Recomendador de Músicas.ipynb ☆

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

Comentário Compartilhar

+ Código + Texto

RAM Disco

Recomendador de músicas utilizando K-Means

```
[1] import pandas as pd
    import numpy as np
```

```
[2] dados = pd.read_csv('/content/Dados_totais.csv')
    dados_generos = pd.read_csv('/content/data_by_genres.csv')
    dados_anos = pd.read_csv('/content/data_by_year.csv')
```

```
[3] dados.head(2)
```

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	n
0	0.285	2000	0.00239	Coldplay	0.429	266773	0.661	0	3AJwUDP919kvQ9QcozQPxg	0.000121	11	0.2340	-7.227	1	Ye
1	0.613	2000	0.14300	OutKast	0.843	270507	0.806	1	0l3q5fE6wg7LIfHGngUTnV	0.000000	4	0.0771	-5.946	0	Jack

```
[4] dados["year"].unique()
```

```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020])
```

```
[5] dados.shape
```

```
(20311, 19)
```

```
[6] dados.drop(["explicit", "key", "mode"], axis = 1, inplace=True)
    dados.shape
```

```
(20311, 16)
```

```
[7] dados.isnull().sum()
```

```
valence           0
year              0
acousticness      0
artists           0
danceability      0
duration_ms       0
energy            0
id                0
instrumentalness  0
liveness          0
loudness          0
name              0
popularity        0
speechiness       0
tempo             0
artists_song      0
dtype: int64
```

```
[8] dados.isna().sum()
```

```
valence           0
year              0
acousticness      0
artists           0
danceability      0
duration_ms       0
energy            0
id                0
instrumentalness  0
liveness          0
loudness          0
name              0
```

```
popularity          0
speechiness         0
tempo               0
artists_song        0
dtype: int64
```

[9]
```
dados_generos.head(2)
```

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
0	1	21st century classical	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.3616	-31.514333	0.040567	75.336500	0.103783	27.833333	6
1	1	432hz	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.1310	-16.854000	0.076817	120.285667	0.221750	52.500000	5

[10]
```
dados_generos.drop(["key", "mode"], axis = 1, inplace=True)
dados_generos.shape
```

```
(2973, 12)
```

[11]
```
dados_generos.isnull().sum()
```

```
genres              0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
dtype: int64
```

[12]
```
dados_generos.isna().sum()
```

```
genres              0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
dtype: int64
```

[13]
```
dados_anos.head(2)
```

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.344878	0.20571	-17.048667	0.073662	101.531493	0.379327	0.653333	2
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.434195	0.24072	-19.275282	0.116655	100.884521	0.535549	0.140845	10

[14]
```
dados_anos['year'].unique()
```

```
array([1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931,
       1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942,
       1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953,
       1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964,
       1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975,
       1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986,
       1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997,
       1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
       2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
       2020])
```

[15]
```
dados_anos = dados_anos[dados_anos["year"] >= 2000]
dados_anos['year'].unique()
```

```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020])
```

[16]
```
dados_anos.drop(["key", "mode"], axis = 1, inplace=True)
dados_anos.head(2)
```

```
<ipython-input-16-e368ae42d938>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dados_anos.drop(["key", "mode"], axis = 1, inplace=True)
```

	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity
79	2000	0.289323	0.590918	242724.642638	0.625413	0.101168	0.197686	-8.247766	0.089205	118.999323	0.559475	46.684049
80	2001	0.286842	0.583318	240307.796010	0.626986	0.107214	0.187026	-8.305095	0.089182	117.765399	0.541479	48.750125

```
[17] dados_anos.reset_index()
```

	index	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity
0	79	2000	0.289323	0.590918	242724.642638	0.625413	0.101168	0.197686	-8.247766	0.089205	118.999323	0.559475	46.684049
1	80	2001	0.286842	0.583318	240307.796010	0.626986	0.107214	0.187026	-8.305095	0.089182	117.765399	0.541479	48.750125
2	81	2002	0.282624	0.576160	239503.283000	0.641270	0.088048	0.193911	-7.686640	0.084308	119.239738	0.542397	48.655500
3	82	2003	0.256471	0.575763	244670.575230	0.660165	0.083049	0.196976	-7.485545	0.093926	120.914622	0.530504	48.626407
4	83	2004	0.280559	0.567680	237378.708037	0.648868	0.077934	0.202199	-7.601655	0.094239	121.290346	0.524489	49.273143
5	84	2005	0.255764	0.572281	237229.588205	0.653209	0.090194	0.190082	-7.466159	0.093334	121.617967	0.532531	50.953333
6	85	2006	0.279986	0.568230	234042.914359	0.650326	0.077701	0.188289	-7.265501	0.085847	121.798615	0.520028	51.313846
7	86	2007	0.254081	0.563414	241049.962564	0.668305	0.072957	0.196127	-7.044536	0.084347	124.087516	0.516794	51.075897
8	87	2008	0.249192	0.579193	240107.315601	0.671461	0.063662	0.198431	-6.843804	0.077356	123.509934	0.527542	50.630179
9	88	2009	0.261929	0.564190	238140.013265	0.670749	0.075872	0.205252	-7.046015	0.085458	123.463808	0.507170	51.440816
10	89	2010	0.242687	0.572488	242811.804563	0.681778	0.082981	0.199701	-6.909904	0.081031	123.570215	0.520895	52.730159
11	90	2011	0.273183	0.552867	236998.787308	0.648301	0.103772	0.203309	-7.574986	0.087479	121.483997	0.472454	53.307387
12	91	2012	0.249953	0.570882	245807.457584	0.656571	0.085206	0.189733	-7.260550	0.081742	121.781736	0.462709	52.655013
13	92	2013	0.257488	0.571148	242267.661437	0.645597	0.098365	0.199631	-7.472039	0.093849	120.806829	0.454741	54.047065
14	93	2014	0.249313	0.589948	233728.314713	0.648795	0.076570	0.191822	-7.067440	0.084061	122.305263	0.463049	55.543142
15	94	2015	0.253952	0.593774	230029.046606	0.627064	0.106787	0.188856	-7.625639	0.096779	120.115411	0.432098	56.700608
16	95	2016	0.284171	0.600202	221396.510295	0.592855	0.093984	0.181170	-8.061056	0.104313	118.652630	0.431532	59.647190
17	96	2017	0.286099	0.612217	211115.696787	0.590421	0.097091	0.191713	-8.312630	0.110536	117.202740	0.416476	63.263554
18	97	2018	0.267633	0.663500	206001.007133	0.602435	0.054217	0.176326	-7.168785	0.127176	121.922308	0.447921	63.296243
19	98	2019	0.278299	0.644814	201024.788096	0.593224	0.077640	0.172616	-7.722192	0.121043	120.235644	0.458818	65.256542
20	99	2020	0.219931	0.692904	193728.397537	0.631232	0.016376	0.178535	-6.595067	0.141384	124.283129	0.501048	64.301970

```
[18] dados_anos.isna().sum()
```

```
year              0
acousticness      0
danceability      0
duration_ms       0
energy            0
instrumentalness  0
liveness          0
loudness          0
speechiness       0
tempo             0
valence           0
popularity        0
dtype: int64
```

```
[19] dados_anos.isnull().sum()
```

```
year              0
acousticness      0
danceability      0
duration_ms       0
energy            0
instrumentalness  0
liveness          0
loudness          0
speechiness       0
tempo             0
valence           0
popularity        0
dtype: int64
```

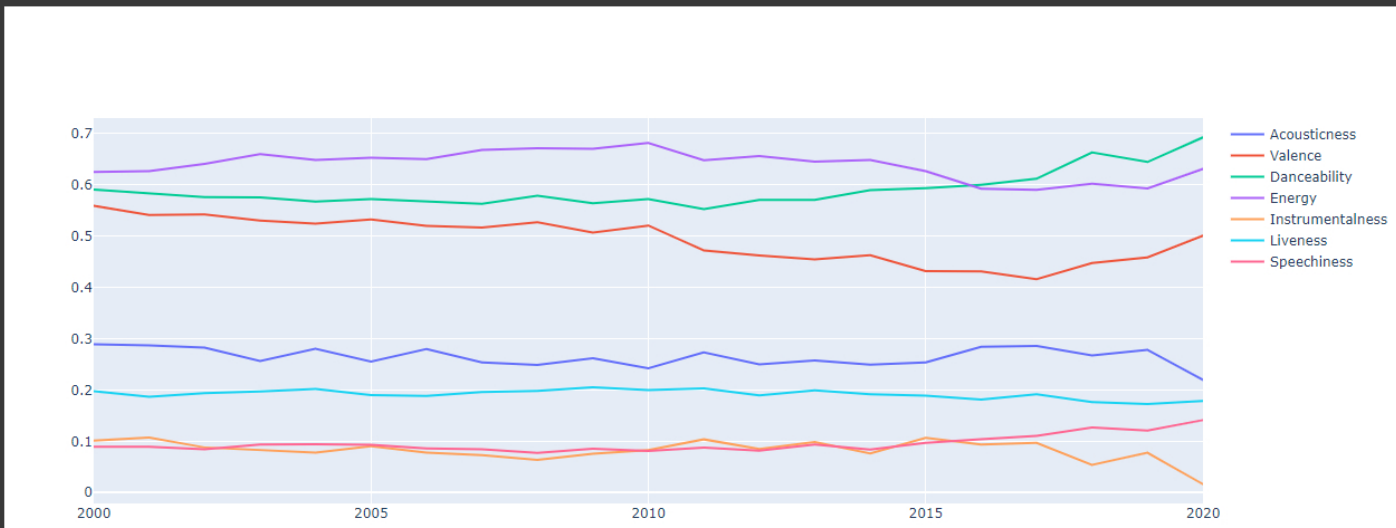
Análise Gráfica

```
[20] import plotly.express as px
```

```
[21] fig = px.line(dados_anos, x = 'year', y = 'loudness', markers = True, title = 'Variação do Loudness conforme os anos')
     fig.show()
```

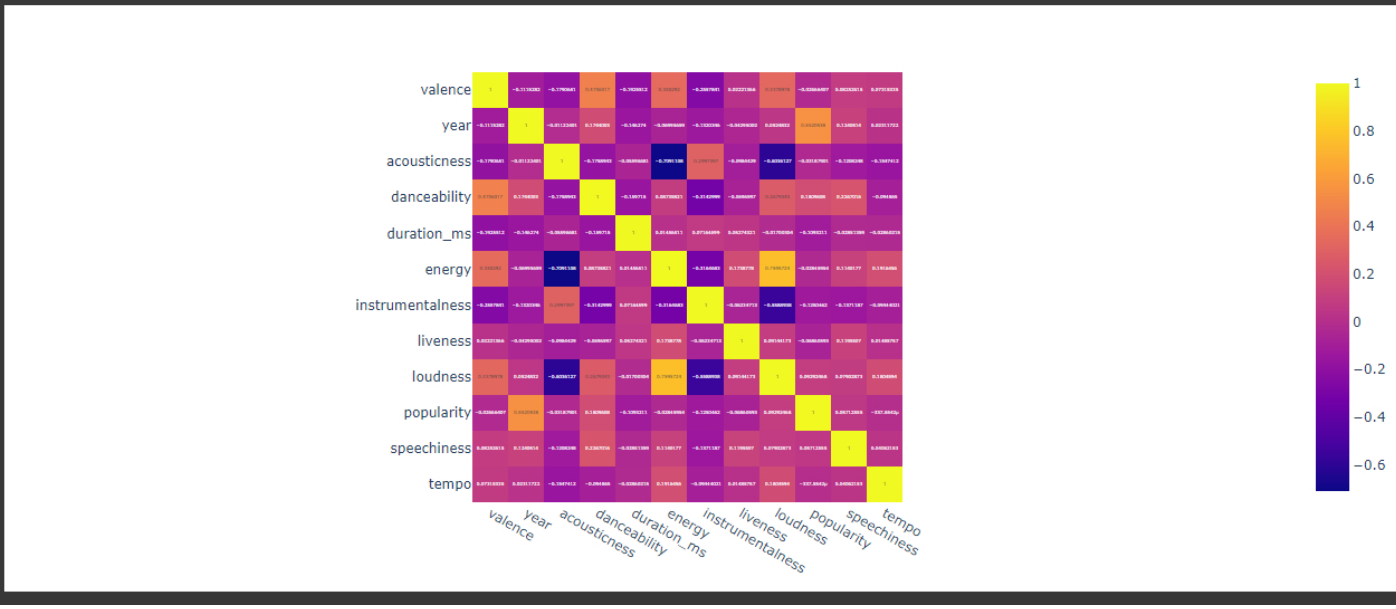
[22] `import plotly.graph_objects as go`

[23]
```
fig = go.Figure()
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['acousticness'], name = 'Acousticness'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['valence'], name = 'Valence'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['danceability'], name = 'Danceability'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['energy'], name = 'Energy'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['instrumentalness'], name = 'Instrumentalness'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['liveness'], name = 'Liveness'))
fig.add_trace(go.Scatter(x = dados_anos['year'], y = dados_anos['speechiness'], name = 'Speechiness'))
fig.show()
```



[24]
```
fig = px.imshow(dados.corr(), text_auto=True)
fig.show()
```

```
<ipython-input-24-fa7b692f5384>:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of
```



Clusterização por gênero

```
[25] dados_generos['genres'].value_counts().sum()
```

```
2973
```

```
[26] dados_generos1 = dados_generos.drop('genres', axis=1)
     dados_generos1
```

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity
0	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514333	0.040567	75.336500	0.103783	27.833333
1	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854000	0.076817	120.285667	0.221750	52.500000
2	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180000	0.047000	133.444000	0.975000	48.000000
3	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288965	0.107872	112.857352	0.513604	20.859882
4	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479387	0.082851	112.110362	0.448249	45.820071
...	...	...	...	...	...	...	...	...	...	...	...
2968	0.222625	0.547082	2.580991e+05	0.610240	0.143872	0.204206	-11.295878	0.061088	125.494919	0.596155	33.778943
2969	0.161000	0.863000	2.063200e+05	0.909000	0.000000	0.108000	-5.985000	0.081300	119.038000	0.845000	58.000000
2970	0.263261	0.748889	3.060728e+05	0.622444	0.257227	0.089678	-10.289222	0.038778	101.965222	0.824111	46.666667
2971	0.993000	0.705667	1.984173e+05	0.172667	0.468633	0.179667	-11.453333	0.348667	91.278000	0.739000	0.000000
2972	0.421038	0.629409	1.716717e+05	0.609369	0.019248	0.255877	-9.854825	0.050491	126.366087	0.808544	30.261905

2973 rows × 11 columns

```
[27] from sklearn.pipeline import Pipeline
     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA
```

```
[28] SEED = 1224
     np.random.seed(1224)

     pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2, random_state=SEED))])
```

```
[29] genre_embedding_pca = pca_pipeline.fit_transform(dados_generos1)
     projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding_pca)
```

```
[30] projection
```

	x	y
0	5.910268	-0.011146
1	2.787093	4.498483
2	-0.757538	-1.258495
3	1.020521	-0.931690
4	0.753911	-0.999861
...	...	...
2968	-0.475136	-0.017126
2969	-3.026756	-1.304983
2970	-0.832511	-1.089726
2971	1.774790	-3.695233
2972	-1.028069	-1.390709

2973 rows × 2 columns

```
[31] from sklearn.cluster import KMeans

     kmeans_pca = KMeans(n_clusters=5, verbose=True, random_state=SEED)
     kmeans_pca.fit(projection)

     dados_generos['cluster_pca'] = kmeans_pca.predict(projection)
     projection['cluster_pca'] = kmeans_pca.predict(projection)
```

```
Iteration 22, inertia 3509.923997363926.
Iteration 23, inertia 3508.6659460439305.
Iteration 24, inertia 3507.576714118868.
Iteration 25, inertia 3506.338287272816.
Iteration 26, inertia 3505.313827224235.
Iteration 27, inertia 3504.8047979287217.
Iteration 28, inertia 3504.5524335160226.
Converged at iteration 28: center shift 0.00012946374322485388 within tolerance 0.0002709886847099649.
Initialization complete
Iteration 0, inertia 5726.817439830062.
Iteration 1, inertia 4002.8186163759915.
Iteration 2, inertia 3677.022813030337.
```

```
Iteration 3, inertia 3594.023594573494.
Iteration 4, inertia 3568.0392461810325.
Iteration 5, inertia 3559.4047642395535.
Iteration 6, inertia 3554.535649933391.
Iteration 7, inertia 3550.9000086430456.
Iteration 8, inertia 3547.931977312026.
Iteration 9, inertia 3546.186206463716.
Iteration 10, inertia 3544.593079901218.
Iteration 11, inertia 3544.157310008578.
Iteration 12, inertia 3543.941753998973.
Converged at iteration 12: center shift 0.00010369698781414579 within tolerance 0.0002709886847099649.
Initialization complete
Iteration 0, inertia 4375.718400720178.
Iteration 1, inertia 3663.371869736929.
Iteration 2, inertia 3584.8228907672656.
Iteration 3, inertia 3560.552577179441.
Iteration 4, inertia 3552.6535536541846.
Iteration 5, inertia 3550.9863322071333.
Iteration 6, inertia 3548.841299825511.
Iteration 7, inertia 3546.343296544779.
Iteration 8, inertia 3544.272114110863.
Iteration 9, inertia 3541.0832411176907.
Iteration 10, inertia 3538.703786254462.
Iteration 11, inertia 3536.9610470989546.
Iteration 12, inertia 3535.1468370885514.
Iteration 13, inertia 3533.5433662533487.
Iteration 14, inertia 3530.7882440270614.
Iteration 15, inertia 3527.196024419155.
Iteration 16, inertia 3522.5419691521615.
Iteration 17, inertia 3518.2447958096354.
Iteration 18, inertia 3516.6343144695356.
Iteration 19, inertia 3515.6221060844655.
Iteration 20, inertia 3513.9973981796215.
Iteration 21, inertia 3511.188703670774.
Iteration 22, inertia 3509.97825268788.
Iteration 23, inertia 3508.792500367633.
Iteration 24, inertia 3507.658797555081.
Iteration 25, inertia 3506.3651721284323.
Iteration 26, inertia 3505.313827224235.
Iteration 27, inertia 3504.8047979287217.
Iteration 28, inertia 3504.5524335160226.
Converged at iteration 28: center shift 0.00012946374322485388 within tolerance 0.0002709886847099649.
/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
```

[32] projection

	x	y	cluster_pca
0	5.910268	-0.011146	3
1	2.787093	4.498483	3
2	-0.757538	-1.258495	4
3	1.020521	-0.931690	1
4	0.753911	-0.999861	1
...	...	...	...
2968	-0.475136	-0.017126	2
2969	-3.026756	-1.304983	4
2970	-0.832511	-1.089726	4
2971	1.774790	-3.695233	1
2972	-1.028069	-1.390709	4

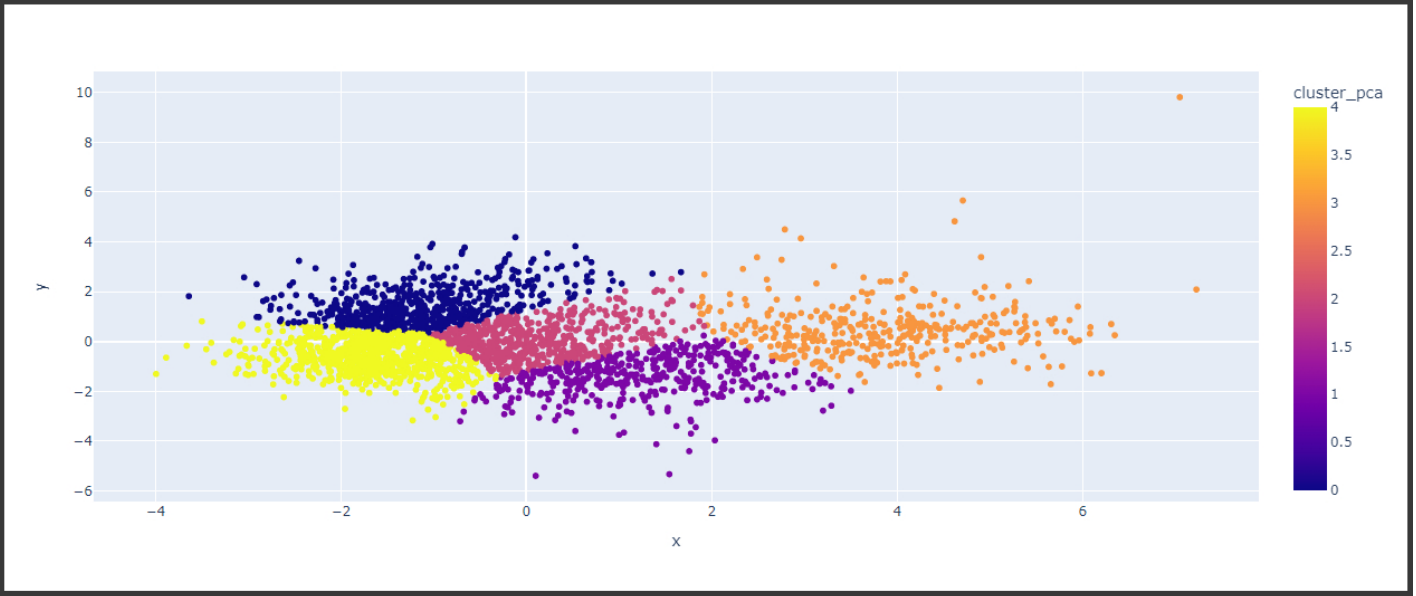
2973 rows × 3 columns

[33]
```
projection['generos'] = dados_generos['genres']
projection
```

	x	y	cluster_pca	generos
0	5.910268	-0.011146	3	21st century classical
1	2.787093	4.498483	3	432hz
2	-0.757538	-1.258495	4	8-bit
3	1.020521	-0.931690	1	[]
4	0.753911	-0.999861	1	a cappella
...	...	...	...	...
2968	-0.475136	-0.017126	2	zolo
2969	-3.026756	-1.304983	4	zouglou
2970	-0.832511	-1.089726	4	zouk
2971	1.774790	-3.695233	1	zurich indie
2972	-1.028069	-1.390709	4	zydeco

2973 rows × 4 columns

```
[34] fig = px.scatter(
         projection, x='x', y='y', color='cluster_pca', hover_data=['x', 'y', 'generos']
     )
     fig.show()
```




```
[35] # como saber se esses clusters estão realmente bem separados?
     # precisamos utilizar a taxa de variância explicada para saber o quanto os nossos dados são explicados dentro dos clusters

     pca_pipeline[1].explained_variance_ratio_

     # [1] é a posição do pca. A [0] é do StandardScaler.
     # sempre a primeira posição do array vai explicar melhor do que a segunda.
```

```
array([0.34986105, 0.14284565])
```

```
[36] pca_pipeline[1].explained_variance_ratio_.sum()

     # podemos notar que quase 50% dos dados estão sendo explicados em cada cluster.
     # 0.50 foi dado com um valor razoável.
```

```
0.4927066994726641
```

```
[37] # para sabermos quantas variáveis são explicadas por cada coluna.

     pca_pipeline[1].explained_variance_

     # A coluna X explica 3.84 e a Y, 1.57
```

```
array([3.84976644, 1.57183087])
```

```
[38] pca_pipeline[1].explained_variance_.sum()

     # são 5 colunas sendo explicadas de 11 totais.
```

```
5.421597305805697
```

Clusterização Por Música

```
[39] # total de artistas

     dados['artists'].value_counts()
```

```
Drake                  170
Taylor Swift           156
Eminem                 147
Kanye West             136
BTS                    122
                      ...
Of Monsters and Men      7
The Living Tombstone     7
Clean Bandit             7
Nelly Furtado            6
Empire of the Sun        6
Name: artists, Length: 875, dtype: int64
```

```
[40] # total de músicas de artistas

     dados['artists_song'].value_counts()
```

```
Coldplay - Yellow                                  1
Juice WRLD - Intro                                 1
```

```
Carrie Underwood - The Champion - Bonus Track                    1
YoungBoy Never Broke Again - Cross Me (feat. Lil Baby and Plies)  1
Twenty One Pilots - Morph                                        1
                                                                 ..
Taylor Swift - Come Back...Be Here                               1
Banda El Recodo - Me Gusta Todo De Ti                            1
Michael Bublé - The Christmas Song                               1
Glee Cast - Smooth Criminal (Glee Cast Version) (feat. 2CELLOS)  1
Eminem - Darkness                                                1
Name: artists_song, Length: 20311, dtype: int64
```

[41]
```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(dtype=int)
colunas_ohe = ohe.fit_transform(dados[['artists']]).toarray()
dados2 = dados.drop('artists', axis=1)

dados_musicas_dummies = pd.concat([dados2, pd.DataFrame(colunas_ohe, columns=ohe.get_feature_names_out(['artists']))], axis=1)
dados_musicas_dummies
```

	valence	year	acousticness	danceability	duration_ms	energy	id	instrumentalness	liveness	loudness	...	artists_Zara Larsson	artists_Ze
0	0.285	2000	0.00239	0.429	266773	0.661	3AJwUDP919kvQ9QcozQPxg	0.000121	0.2340	-7.227	...	0	
1	0.613	2000	0.14300	0.843	270507	0.806	0l3q5fE6wg7LIfHGngUTnV	0.000000	0.0771	-5.946	...	0	
2	0.400	2000	0.00958	0.556	216880	0.864	60a0Rd6pjrkxjPbaKzXjfq	0.000000	0.2090	-5.870	...	0	
3	0.543	2000	0.00664	0.545	233933	0.865	6ZOBP3NvfbU4SZcrnt1k6	0.000011	0.1680	-5.708	...	0	
4	0.760	2000	0.03020	0.949	284200	0.661	3yfqSUWxFvZELEM4PmlwIR	0.000000	0.0454	-4.244	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
20306	0.187	2020	0.57400	0.445	176250	0.330	0PUkanqCGTb6qseXPKOw1F	0.000000	0.1020	-8.121	...	0	
20307	0.240	2020	0.81500	0.467	179188	0.429	6Iw6fQVKZi0fWtEQBrTFcP	0.000052	0.1770	-8.689	...	0	
20308	0.466	2020	0.31000	0.562	253613	0.686	308prODCCD0O660tlktbUi	0.022500	0.1250	-8.480	...	0	
20309	0.522	2020	0.20400	0.598	230600	0.472	2f8y4CuG57UJEmkG3ujd0D	0.000015	0.1080	-10.991	...	0	
20310	0.195	2020	0.00998	0.671	337147	0.623	5SiZJoLXp3WOl3J4C8IK0d	0.000008	0.6430	-7.161	...	0	

20311 rows × 890 columns

[42]
```
# Comparando colunas

dados.shape
```
```
(20311, 16)
```

[43]
```
dados_musicas_dummies.shape
```
```
(20311, 890)
```

[44]
```
pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=0.7, random_state=SEED))])

music_embedding_pca = pca_pipeline.fit_transform(dados_musicas_dummies.drop(['id', 'name', 'artists_song'], axis=1))
projection_m = pd.DataFrame(data=music_embedding_pca)
```

[45]
```
pca_pipeline[1].n_components_
```
```
612
```

[46]
```
kmeans_pca_pipeline = KMeans(n_clusters=50, verbose=False, random_state=SEED)
kmeans_pca_pipeline.fit(projection_m)

dados['cluster_pca'] = kmeans_pca_pipeline.predict(projection_m)
projection_m['cluster_pca'] = kmeans_pca_pipeline.predict(projection_m)
```
```
/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
```

[47]
```
projection_m['artist'] = dados['artists']
projection_m['song'] = dados['artists_song']
```

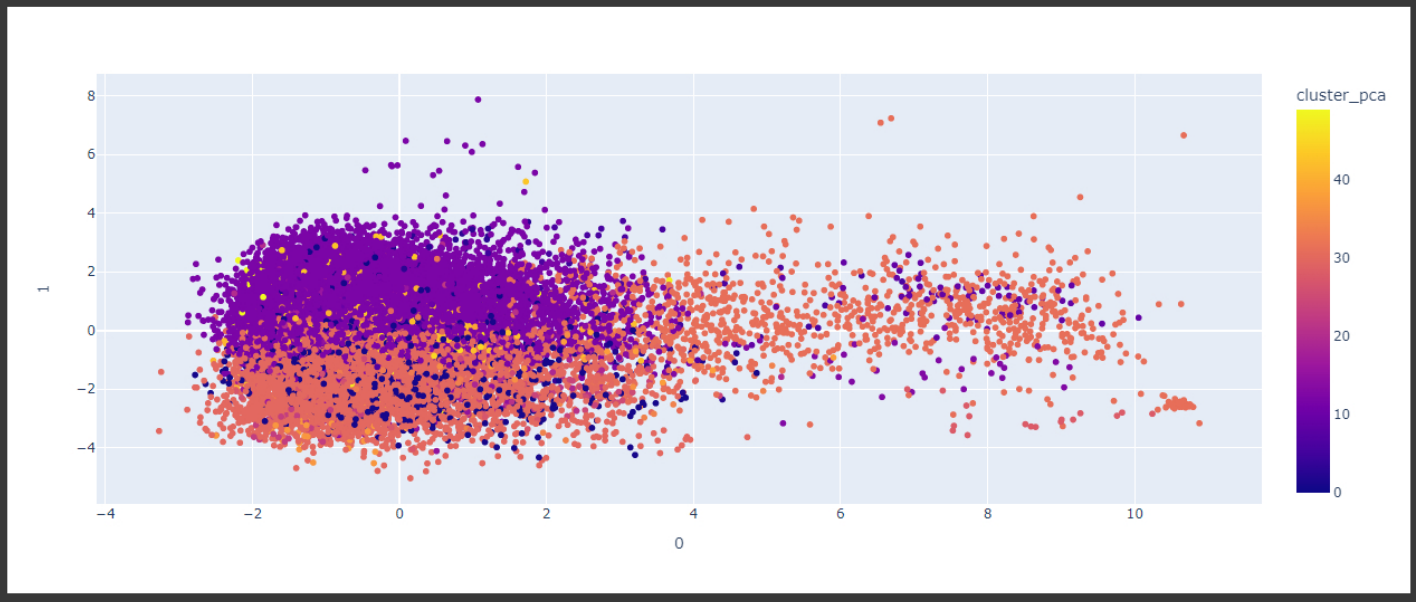
[48]
```
projection_m
```

	0	1	2	3	4	5	6	7	8	9	...	605	606	607	608	609
0	0.174796	0.731252	2.186797	-0.767192	0.594847	0.315968	-0.412322	-0.479171	-2.700668	2.109956	...	-0.009111	0.005154	-0.034027	-0.032617	2.241192e-11
1	-1.358420	0.052935	-1.789973	1.938444	0.403606	1.023898	-1.172118	0.821698	-1.716897	0.252498	...	-0.031986	0.029708	-0.016322	-0.030234	-2.395758e-12
2	-0.972077	0.658094	0.757500	-0.277350	-0.400393	0.402941	1.292960	1.839192	-1.562236	1.410677	...	0.017097	0.017361	-0.010979	-0.012781	4.273933e-12

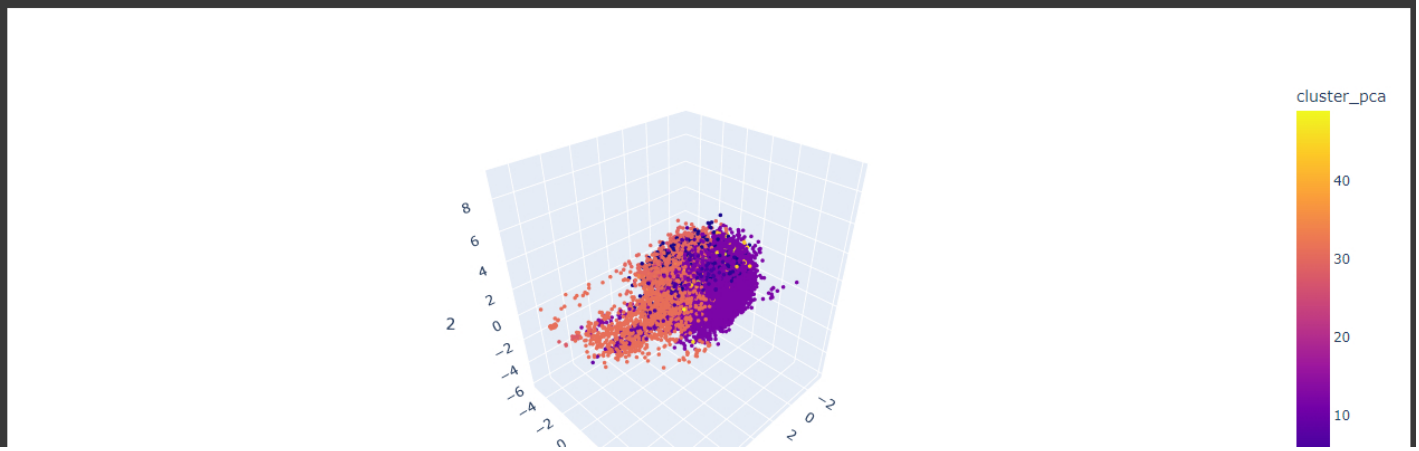
3	-0.926464	1.292091	0.398499	-0.997738	0.202219	0.620859	-0.527689	1.135529	-1.343893	1.020964	...	0.089337	0.164777	-0.064148	-0.040911	5.152080e-11
4	-1.710077	-0.383502	-1.258562	1.346428	1.223591	1.744579	0.094311	0.479412	-2.125376	0.389565	...	-0.011937	-0.004081	-0.017478	-0.010033	2.254040e-12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20306	1.346699	-2.775889	0.986755	-1.704967	0.103935	-0.726582	0.253808	0.520234	-0.064513	0.204971	...	-0.235355	-0.054256	0.016922	-0.125947	4.217717e-11
20307	1.269387	-2.444619	0.971408	-0.115303	-0.719597	-1.823579	-0.545186	-1.475719	-0.338390	0.272129	...	-0.183026	0.074605	0.005622	0.000054	4.323403e-11
20308	0.781916	-1.164865	0.705256	-0.947160	1.145828	-0.472667	-0.356604	0.972512	0.900317	-0.674016	...	-0.033420	-0.157564	0.213849	-0.065180	-2.580928e-11
20309	0.002454	-3.056407	-0.272934	1.436355	-0.604453	0.251492	-1.361710	-0.633245	0.280076	0.521145	...	-0.784640	0.080684	0.038379	0.108493	5.473520e-11
20310	-0.957008	-1.208016	1.103144	4.128170	1.026126	0.085505	1.588383	0.567041	-1.026283	0.266124	...	-0.012069	-0.003818	-0.017675	-0.010372	2.372825e-12

20311 rows x 615 columns

```
[49] fig = px.scatter(
         projection_m, x=0, y=1, color='cluster_pca', hover_data=[0, 1, 'song']
     )
     fig.show()
```




```
[50] fig = px.scatter_3d(
        projection_m, x=0, y=1, z=2, color='cluster_pca',hover_data=['song'])
     fig.update_traces(marker_size = 2)
     fig.show()
```



[51] `pca_pipeline[1].explained_variance_ratio_`

```
0.00112861, 0.00112861, 0.0011286 , 0.0011286 , 0.00112859,
0.00112859, 0.00112859, 0.00112858, 0.00112858, 0.00112857,
0.00112857, 0.00112856, 0.00112856, 0.00112856, 0.00112856,
0.00112856, 0.00112856, 0.00112856, 0.00112856, 0.00112855,
0.00112855, 0.00112855, 0.00112854, 0.00112854, 0.00112853,
0.00112852, 0.00112852, 0.00112851, 0.00112851, 0.00112851,
0.00112851, 0.00112851, 0.00112851, 0.00112851, 0.00112851,
0.00112851, 0.00112851, 0.0011285 , 0.0011285 , 0.00112849,
0.00112849, 0.00112848, 0.00112848, 0.00112847, 0.00112847,
0.00112846, 0.00112846, 0.00112846, 0.00112845, 0.00112845,
0.00112845, 0.00112845, 0.00112845, 0.00112845, 0.00112845,
0.00112844, 0.00112844, 0.00112843, 0.00112843, 0.00112843,
0.00112842, 0.00112842, 0.00112842, 0.0011284 , 0.0011284 ,
0.0011284 , 0.0011284 , 0.0011284 , 0.0011284 , 0.0011284 ,
0.0011284 , 0.0011284 , 0.0011284 , 0.0011284 , 0.0011284 ,
0.0011284 , 0.0011284 , 0.00112839, 0.00112839, 0.00112839,
0.00112839, 0.00112838, 0.00112838, 0.00112838, 0.00112837,
0.00112837, 0.00112837, 0.00112836, 0.00112836, 0.00112835,
0.00112834, 0.00112834, 0.00112834, 0.00112834, 0.00112834,
0.00112834, 0.00112834, 0.00112834, 0.00112834, 0.00112834,
0.00112834, 0.00112834, 0.00112834, 0.00112834, 0.00112834,
0.00112834, 0.00112834, 0.00112834, 0.00112834, 0.00112834,
0.00112834, 0.00112834, 0.00112834, 0.00112834, 0.00112833,
0.00112833, 0.00112833, 0.00112833, 0.00112832, 0.00112831,
0.00112831, 0.00112831, 0.00112831, 0.0011283 , 0.0011283 ,
0.0011283 , 0.00112829, 0.00112828, 0.00112828, 0.00112828,
0.00112828, 0.00112828, 0.00112828, 0.00112828, 0.00112828,
0.00112828, 0.00112828, 0.00112828, 0.00112828, 0.00112828,
0.00112828, 0.00112828, 0.00112828, 0.00112828, 0.00112828,
0.00112828, 0.00112828, 0.00112827, 0.00112827, 0.00112827,
0.00112826, 0.00112826, 0.00112826, 0.00112825, 0.00112825,
0.00112825, 0.00112824, 0.00112824, 0.00112823, 0.00112823,
0.00112823, 0.00112823, 0.00112823, 0.00112823, 0.00112823,
0.00112823, 0.00112823, 0.00112823, 0.00112823, 0.00112823,
0.00112823, 0.00112823, 0.00112823, 0.00112823, 0.00112823,
0.00112823, 0.00112823, 0.00112823, 0.00112823, 0.00112823,
0.00112823, 0.00112822, 0.00112822, 0.00112822, 0.00112821,
0.00112821, 0.00112821, 0.00112821, 0.0011282 , 0.0011282 ,
0.0011282 , 0.00112819, 0.00112819, 0.00112819, 0.00112817,
0.00112817, 0.00112817, 0.00112817, 0.00112817, 0.00112817,
0.00112817, 0.00112817, 0.00112817, 0.00112817, 0.00112817,
0.00112817, 0.00112817, 0.00112817, 0.00112817, 0.00112817,
0.00112817, 0.00112817, 0.00112817, 0.00112817, 0.00112817,
0.00112817, 0.00112817, 0.00112817, 0.00112817, 0.00112817,
0.00112816, 0.00112816, 0.00112815, 0.00112815, 0.00112815,
0.00112814, 0.00112814, 0.00112814, 0.00112814, 0.00112813,
0.00112813, 0.00112813, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112812, 0.00112812, 0.00112812, 0.00112812,
0.00112812, 0.00112811, 0.00112811, 0.0011281 , 0.00112809,
0.00112809, 0.00112809, 0.00112809, 0.00112809, 0.00112808,
0.00112808, 0.00112808, 0.00112807, 0.00112807, 0.00112806,
0.00112806, 0.00112806])
```

[52] `pca_pipeline[1].explained_variance_ratio_.sum()`

```
0.7000008463187031
```

[53] `pca_pipeline[1].explained_variance_`

```
1.00112957, 1.00112278, 1.00112014, 1.00111593, 1.00111097,
1.00111014, 1.00110688, 1.00110297, 1.00109604, 1.00109356,
1.00109101, 1.00108428, 1.00108428, 1.00108428, 1.00108428,
1.00108428, 1.00108301, 1.00108221, 1.00107783, 1.00107571,
1.0010706 , 1.00106976, 1.00106442, 1.00106081, 1.00105482,
1.00105039, 1.00104303, 1.00103844, 1.00103686, 1.00103494,
1.00103494, 1.00103494, 1.00103494, 1.00103494, 1.00103494,
1.00103494, 1.00103494, 1.00102964, 1.00102745, 1.00102014,
1.00101695, 1.00101286, 1.00101005, 1.00100567, 1.0009992 ,
1.0009946 , 1.00099319, 1.00099024, 1.00098878, 1.00098577,
1.00098561, 1.0009856 , 1.00098538, 1.00098238, 1.00098191,
1.00097695, 1.00097308, 1.00097064, 1.00096714, 1.0009639 ,
1.0009583 , 1.00095688, 1.00095352, 1.00093972, 1.00093629,
1.00093629, 1.00093629, 1.00093629, 1.00093629, 1.00093629,
1.00093629, 1.00093629, 1.00093629, 1.00093629, 1.00093629,
1.00093629, 1.00093629, 1.00093314, 1.00092943, 1.00092822,
1.00092715, 1.00092395, 1.00092036, 1.00092016, 1.00091724,
1.00091592, 1.0009112 , 1.00090694, 1.00090263, 1.00089665,
1.00088696, 1.00088696, 1.00088696, 1.00088696, 1.00088696,
1.00088696, 1.00088696, 1.00088696, 1.00088696, 1.00088696,
1.00088696, 1.00088696, 1.00088696, 1.00088696, 1.00088696,
1.00088696, 1.00088696, 1.00088696, 1.00088696, 1.00088696,
1.00088696, 1.00088696, 1.00088696, 1.00088696, 1.00087952,
1.00087918, 1.00087433, 1.00087367, 1.00087066, 1.00086458,
1.00086236, 1.00086069, 1.0008569 , 1.00085485, 1.00085116,
1.00084748, 1.00083907, 1.00083765, 1.00083765, 1.00083765,
```

```
1.00083765, 1.00083765, 1.00083765, 1.00083765, 1.00083765,
1.00083765, 1.00083765, 1.00083765, 1.00083765, 1.00083765,
1.00083765, 1.00083765, 1.00083765, 1.00083765, 1.00083765,
1.00083644, 1.0008313 , 1.00082607, 1.00082326, 1.00082201,
1.00081971, 1.00081677, 1.00081218, 1.00081034, 1.00080643,
1.0008046 , 1.00080014, 1.00079807, 1.00078834, 1.00078834,
1.00078834, 1.00078834, 1.00078834, 1.00078834, 1.00078834,
1.00078834, 1.00078834, 1.00078834, 1.00078834, 1.00078834,
1.00078834, 1.00078834, 1.00078834, 1.00078834, 1.00078834,
1.00078834, 1.00078834, 1.00078834, 1.00078834, 1.00078834,
1.00078834, 1.00078451, 1.00077816, 1.00077598, 1.00077502,
1.00077274, 1.0007717 , 1.00077055, 1.00076633, 1.00075988,
1.00075852, 1.00075534, 1.00075242, 1.00075068, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073903,
1.00073903, 1.00073903, 1.00073903, 1.00073903, 1.00073268,
1.0007255 , 1.00072453, 1.00072259, 1.00072084, 1.00071798,
1.00071216, 1.00070986, 1.00070776, 1.00070654, 1.00070412,
1.0006998 , 1.00069713, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068973, 1.00068973, 1.00068973, 1.00068973,
1.00068973, 1.00068121, 1.00067902, 1.00067108, 1.00066742,
1.0006665 , 1.00066471, 1.00066314, 1.00066069, 1.0006592 ,
1.00065675, 1.00065402, 1.00064997, 1.00064884, 1.00064043,
1.00064043, 1.00064043])
```

[54]
```
pca_pipeline[1].explained_variance_.sum()
```

```
620.9313218688688
```

Recomendações

[55]
```
nome_musica = "Ed Sheeran - Shape of You"
```

[56]
```
from sklearn.metrics.pairwise import euclidean_distances

cluster = list(projection_m[projection_m['song'] == nome_musica]['cluster_pca'])[0]
musicas_recomendadas = projection_m[projection_m['cluster_pca'] == cluster][[0,1,'song']]
x_musica = list(projection_m[projection_m['song'] == nome_musica][0])[0]
y_musica = list(projection_m[projection_m['song'] == nome_musica][1])[0]

#distancias euclidianas
distancias = euclidean_distances(musicas_recomendadas[[0, 1]], [[x_musica, y_musica]])
musicas_recomendadas['id'] = dados['id']
musicas_recomendadas['distancias'] = distancias
recomendadas = musicas_recomendadas.sort_values('distancias').head(10)
recomendadas
```

	0	1	song	id	distancias
2463	-0.826954	-2.939691	Ed Sheeran - Shape of You	7qiZfU4dY1lWllzX7mPBI3	0.000000
13421	-0.826575	-2.915302	$uicideBoy$ - For the Last Time	240audWazVjwvwh7XwfSZE	0.024393
2942	-0.863727	-2.965997	Marshmello - Be Kind (with Halsey)	3Z8FwOEN59mRMxDCtb8N0A	0.045213
5371	-0.874338	-2.927113	The Kid LAROI - WRONG (feat. Lil Mosey)	1EWkw4Fa6IlnsAihLUIFFM	0.049025
13682	-0.878031	-2.944654	YoungBoy Never Broke Again - House Arrest Tingz	4Boj4bBiDv6Ur1zKEvLXBI	0.051317
15392	-0.792012	-2.987829	$uicideBoy$ - Putrid Pride	4CwA4CdTQqNv18QSAW3Srk	0.059483
15388	-0.784530	-2.897926	Pop Smoke - Paranoia (feat. Gunna & Young Thug)	3QqJ44HiM84iyZS0G2IKIP	0.059533
5271	-0.789801	-2.986457	Ed Sheeran - South of the Border (feat. Camila...	4vUmTMuQqjdnvlZmAH61Qk	0.059728
18479	-0.904140	-2.933036	21 Savage - 1.5	2wOXxtHZgRkkrkEbKLzzqs	0.077472
2660	-0.869469	-3.011060	Tyga - Taste (feat. Offset)	5IaHrVsrferBYDm0bDyABy	0.083072

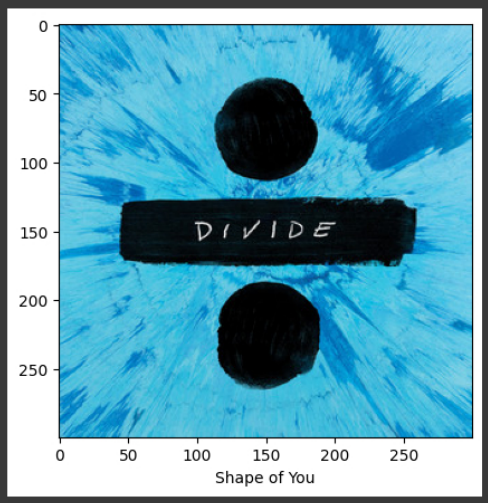
[57]
```
!pip install spotipy
```

```
Requirement already satisfied: spotipy in /usr/local/lib/python3.10/dist-packages (2.23.0)
Requirement already satisfied: redis>=3.5.3 in /usr/local/lib/python3.10/dist-packages (from spotipy) (5.0.1)
Requirement already satisfied: requests>=2.25.0 in /usr/local/lib/python3.10/dist-packages (from spotipy) (2.31.0)
Requirement already satisfied: six>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spotipy) (1.16.0)
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-packages (from spotipy) (2.0.7)
Requirement already satisfied: async-timeout>=4.0.2 in /usr/local/lib/python3.10/dist-packages (from redis>=3.5.3->spotipy) (4.0.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25.0->spotipy) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25.0->spotipy) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25.0->spotipy) (2023.7.22)
```

[58]
```
import spotipy
from spotipy.oauth2 import SpotifyOAuth
from spotipy.oauth2 import SpotifyClientCredentials

scope = "user-library-read playlist-modify-private"
OAuth = SpotifyOAuth(
```

```
                        scope = scope,
                        redirect_uri = 'http://localhost:5000/callback',
                        client_id = '9bc3788e2fe44ef58732f0dd07ed8573',
                        client_secret = '68c40f57aaec40b790b6f3fad2040d8d'
                        )

    client_credentials_manager = SpotifyClientCredentials(client_id='9bc3788e2fe44ef58732f0dd07ed8573', client_secret= '68c40f57aaec40b790b6f3fad2040d8d')
    sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

```
[59] import matplotlib.pyplot as plt
     from skimage import io

     # achando o ID
     id = dados[dados['artists_song'] == nome_musica]['id'].iloc[0]

     # requisitando a API
     track = sp.track(id)
     url = track['album']['images'][1]['url']
     name = track['name']

     # Mexendo com a imagem
     image = io.imread(url)
     plt.imshow(image)
     plt.xlabel(name, fontsize = 10)
     plt.show()
```



Recomendador

```
[60] def recommend_id(playlist_id):
       url = []
       name = []
       for i in playlist_id:
         track = sp.track(i)
         url.append(track['album']['images'][1]['url'])
         name.append(track['name'])
       return name, url
```

```
[61] name, url = recommend_id(recomendadas['id'])
     name, url
```

```
(['Shape of You',
  'For the Last Time',
  'Be Kind (with Halsey)',
  'WRONG (feat. Lil Mosey)',
  'House Arrest Tingz',
  'Putrid Pride',
  'Paranoia (feat. Gunna & Young Thug)',
  'South of the Border (feat. Camila Cabello & Cardi B)',
  '1.5',
  'Taste (feat. Offset)'],
 ['https://i.scdn.co/image/ab67616d00001e02ba5db46f4b838ef6027e6f96',
  'https://i.scdn.co/image/ab67616d00001e02f795a2962820664b4112901e',
  'https://i.scdn.co/image/ab67616d00001e02fdf2e993e10e67396b3bf759',
  'https://i.scdn.co/image/ab67616d00001e02891f0552bed344dc2bfe322f',
  'https://i.scdn.co/image/ab67616d00001e028bd6e34012a7ab6311050d0c',
  'https://i.scdn.co/image/ab67616d00001e0207aa1426cb2b3cfd8ad67c64',
  'https://i.scdn.co/image/ab67616d00001e0246e1307c35579c3483ea7b03',
  'https://i.scdn.co/image/ab67616d00001e0273304ce0653c7758dd94b259',
  'https://i.scdn.co/image/ab67616d00001e02280689ecc5e4b2038bb5e4bd',
  'https://i.scdn.co/image/ab67616d00001e02a4d73b32e40487605c73cffe'])
```

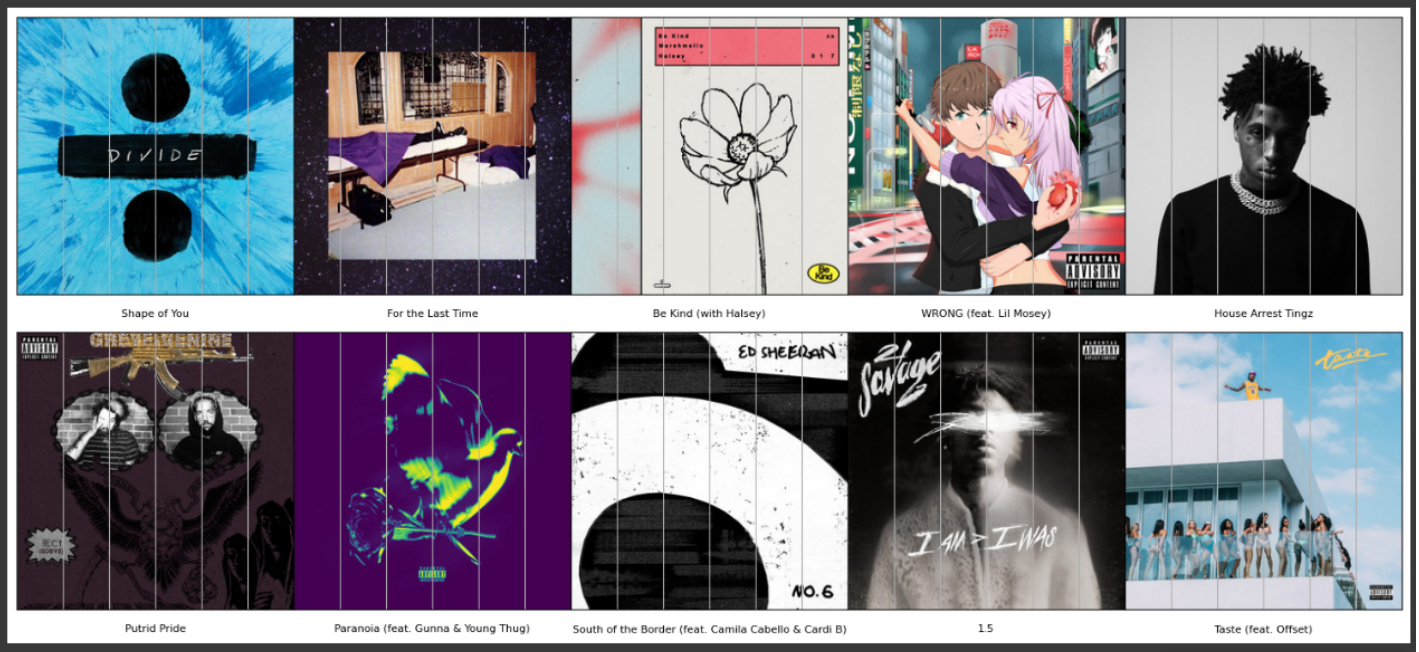
Gerando Imagens da Playlist

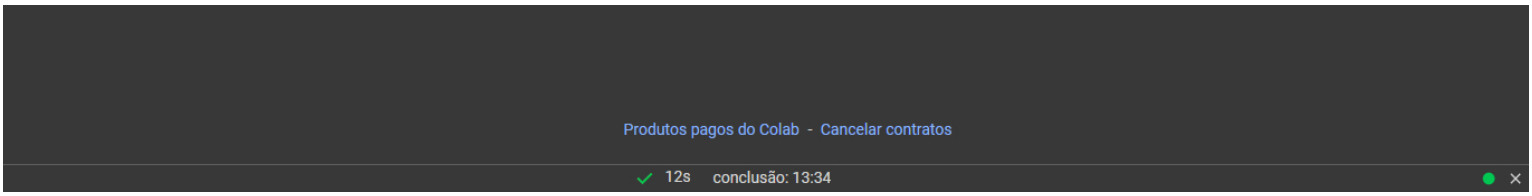
```
[65] def visualize_songs(name, url):
```

```
    plt.figure(figsize=(15,10))
    columns = 5

    for i, u in enumerate(url):
      # define o ax como subplot, com a divisão que retorna inteiro do número urls pela colunas + 1 (no caso, 6)
      ax = plt.subplot(len(url) // columns + 1, columns, i + 1)

      # Lendo a imagem com o Scikit Image
      image = io.imread(u)

      # Mostra a imagem
      plt.imshow(image)

      # Para deixar o eixo Y invisível
      ax.get_yaxis().set_visible(False)

      # xticks define o local que valor trocar os rótulos do eixo x, nesse caso, deixar os pontos de marcação brancos.
      plt.xticks(color = 'w', fontsize = 0.1)

      # yticks define o local que valor trocar os rótulos do eixo y, nesse caso, deixar os pontos de marcação brancos.
      plt.yticks(color = 'w', fontsize = 0.1)

      # Colocando o nome da música no eixo x
      plt.xlabel(name[i], fontsize = 8)

      # Faz com que todos os parâmetros se encaixem no tamanho da imagem definido
      plt.tight_layout(h_pad = 0.7, w_pad = 0)

      # Ajusta os parâmetros de layout da imagem.
      # wspace = A largura do preenchimento entre subparcelas, como uma fração da largura média dos eixos.
      # hspace = A altura do preenchimento entre subparcelas, como uma fração da altura média dos eixos.
      plt.subplots_adjust(wspace=None, hspace=None)

      # Remove os ticks - marcadores, do eixo x, sem remover o eixo todo, deixando o nome da música.
      plt.tick_params(bottom = False)

      # Tirar a grade da imagem gerada automaticamente
      plt.grid(visible=None)
    plt.show()
```

```
[66] visualize_songs(name, url)
```

12s




Produtos pagos do Colab - Cancelar contratos
12s conclusão: 13:34