Quem estiver usando COLAB

```
from google.colab import drive
# Montar o Google Drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.a

Enter your authorization code:
··········
Mounted at /content/drive
```

```
# lib de estrutura e análise de dados.
import pandas as pd
# importa database em um arquivo .csv (necessário indicar o diretório)
base = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Aula 4/TopTracksSpotify2017.csv')
# exibe os valores
base.head()
```

	id	name	artists	danceability	energy	key	loudness	mode	speechiness	acousticness	inst
0	7qiZfU4dY1lWllzX7mPBI	Shape of You	Ed Sheeran	0.825	0.652	1.0	-3.183	0.0	0.0802	0.5810	
1	5CtI0qwDJkDQGwXD1H1cL	Despacito - Remix	Luis Fonsi	0.694	0.815	2.0	-4.328	1.0	0.1200	0.2290	
2	4aWmUDTfIPGksMNLV2rQP	Despacito (Featuring Daddy Yankee)	Luis Fonsi	0.660	0.786	2.0	-4.757	1.0	0.1700	0.2090	
3	6RUKPb4LETWmmr3iAEQkt	Something Just Like This	The Chainsmokers	0.617	0.635	11.0	-6.769	0.0	0.0317	0.0498	
4	3DXncPQOG4VBw3QHh3S81	I'm the One	DJ Khaled	0.609	0.668	7.0	-4.284	1.0	0.0367	0.0552	

Quem estiver usando ANACONDA

1 cell hidden

INÍCIO TRATAMENTO DE DADOS

```
# exibe o nome das colunas
base.columns
```

```
colunas_usadas = ['danceability', 'energy', 'key', 'loudness',
       'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'duration_ms', 'time_signature']


# seleciono apenas as colunas com informações relevantes
colunas_usadas
```

```
['danceability',
 'energy',
 'key',
 'loudness',
 'mode',
 'speechiness',
 'acousticness',
 'instrumentalness',
 'liveness',
 'valence',
 'tempo',
 'duration_ms',
 'time_signature']
```

```
# COLAB
base = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Aula 4/TopTracksSpotify2017.csv', usecols = colunas_usadas)


# anteriormente nos usamos o conceito de padronização, onde os dados foram escalonados.
# para esse exemplo vamos usar o conceito de normalização (0 - 1).
from sklearn.preprocessing import MinMaxScaler
scaler_x = MinMaxScaler()
# inicializamos nosso scaler com os dados, retiramos o PRICE que é nosso objetivo.
base[['danceability', 'energy', 'key', 'loudness',
       'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'duration_ms', 'time_signature']] = scaler_x.fit_transform(base[['danceability', 'energy', 'key', 'loudnes
       'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence', 'tempo', 'duration_ms', 'time_signature']])
# para cada coluna ele recebe o scaler dos valores de X.
# visualização dos dados normalizados
base.head()
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	du
0	0.847534	0.522184	0.090909	0.913192	0.0	0.139774	0.835910	0.000000	0.127515	0.960218	0.167892	
1	0.651719	0.800341	0.181818	0.786896	1.0	0.237371	0.329246	0.000000	0.125755	0.826097	0.111456	
2	0.600897	0.750853	0.181818	0.739576	1.0	0.359980	0.300459	0.000000	0.175050	0.863605	0.823537	
3	0.536622	0.493174	1.000000	0.517648	0.0	0.020844	0.071309	0.000069	0.305835	0.408957	0.224297	
4	0.524664	0.549488	0.636364	0.791749	1.0	0.033104	0.079081	0.000000	0.313380	0.823824	0.047322	

```
# para o preço que seria considerado nosso Y faremos tb a normalização.
scaler_y = MinMaxScaler()
base[['danceability']] = scaler_y.fit_transform(base[['danceability']])
```

```
# visualização dos dados normalizados
base.head()
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	du
0	0.847534	0.522184	0.090909	0.913192	0.0	0.139774	0.835910	0.000000	0.127515	0.960218	0.167892	
1	0.651719	0.800341	0.181818	0.786896	1.0	0.237371	0.329246	0.000000	0.125755	0.826097	0.111456	
2	0.600897	0.750853	0.181818	0.739576	1.0	0.359980	0.300459	0.000000	0.175050	0.863605	0.823537	
3	0.536622	0.493174	1.000000	0.517648	0.0	0.020844	0.071309	0.000069	0.305835	0.408957	0.224297	
4	0.524664	0.549488	0.636364	0.791749	1.0	0.033104	0.079081	0.000000	0.313380	0.823824	0.047322	

```
# na variável X nos temos os atributos previsores, logo não queremos os valores do preço nela.
X = base.drop('danceability', axis = 1) # apaga a coluna do preço.
y = base.danceability
```

```
# visualizar os atributos.
X.head()
```

	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	tim
0	0.522184	0.090909	0.913192	0.0	0.139774	0.835910	0.000000	0.127515	0.960218	0.167892	0.384366	
1	0.800341	0.181818	0.786896	1.0	0.237371	0.329246	0.000000	0.125755	0.826097	0.111456	0.356880	
2	0.750853	0.181818	0.739576	1.0	0.359980	0.300459	0.000000	0.175050	0.863605	0.823537	0.353352	
3	0.493174	1.000000	0.517648	0.0	0.020844	0.071309	0.000069	0.305835	0.408957	0.224297	0.460011	
4	0.549488	0.636364	0.791749	1.0	0.033104	0.079081	0.000000	0.313380	0.823824	0.047322	0.693131	

```
# visualizar os atributos.
y.head()
```

```
0    0.847534
1    0.651719
2    0.600897
3    0.536622
4    0.524664
Name: danceability, dtype: float64
```

```
# ajuste dos dados para criação das classes.
previsores_colunas = colunas_usadas[1:17] # posição 0 é o preço
previsores_colunas
```

```
['energy',
 'key',
 'loudness',
 'mode',
 'speechiness',
 'acousticness',
 'instrumentalness',
 'liveness',
 'valence',
 'tempo',
 'duration_ms',
 'time_signature']
```

FIM TRATAMENTO DE DADOS.

```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
import tensorflow as tf
```

```
tf.__version__
```

```
'1.15.2'
```

```
# vamos criar um for para passar nossas colunas para um array que será usado com o X.
# como os valores de X são numéricos, vamos usar um feature_column.numeric.
colunas = [tf.feature_column.numeric_column(key = c) for c in previsores_colunas]
```

```
# visualizar os valores das colunas.
print(colunas[2])
```

```
NumericColumn(key='loudness', shape=(1,), default_value=None, dtype=tf.float32, normalizer_fn=None)
```

Separar a base de dados

Vamos separa a base entre treinamento e testes

```
# agora precisamos criar uma base para testes e outra para o treinamento, normalmente usado 30, 70.
from sklearn.model_selection import train_test_split
X_treinamento, X_teste, y_treinamento, y_teste = train_test_split(X, y, test_size = 0.3)
```

```
# visualizar o array X de treino.
X_treinamento.shape
```

```
(70, 12)
```

```
# visualizar o array Y de treino.
y_treinamento.shape
```

```
(70,)
```

```
# visualizar o array X de teste.
X_teste.shape
```

```
(30, 12)
```

FUNÇÃO DE TREINAMENTO

```
# define a função de treinamento.
# passamos as entradas dos valores X e Y que foram separados para testes.
# batch_size é definido com 32 para que o alg não carregue de uma só vez todos os dados.
#
funcao_treinamento = tf.estimator.inputs.pandas_input_fn(x = X_treinamento,
                                                        y = y_treinamento,
                                                        batch_size = 32
```

```
                                          batch_size = 32,
                                          num_epochs = None,
                                          shuffle = True)
```

FUNÇÃO DE TESTE

```
# define a função de teste.
funcao_teste = tf.estimator.inputs.pandas_input_fn(x = X_teste,
                                                   y = y_teste,
                                                   batch_size = 32,
                                                   num_epochs = 10000,
                                                   shuffle = False)
```

ALGORITMO - REGRESSÃO LINEAR

```
# função do regressor.
regressor = tf.estimator.LinearRegressor(feature_columns=colunas)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmpq84db5pf
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmpq84db5pf', '_tf_random_seed': None, '_save_summary_steps': 100, '_save_che
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute': None, '
```

TREINAMENTO...

```
# com o regressor podemos fazer o treinamento.
regressor.train(input_fn=funcao_treinamento, steps = 10000)
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/training/training_util.py:236: Variable.initialized_
Instructions for updating:
Use Variable.read_value. Variables in 2.X are initialized automatically both in eager and graph (inside tf.defun) contexts.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_estimator/python/estimator/inputs/queues/feeding_queue_runner.py
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_estimator/python/estimator/inputs/queues/feeding_functions.py:50
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/feature_column/feature_column_v2.py:305: Layer.add_v
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling BaseResou
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_estimator/python/estimator/canned/linear.py:308: to_float (from
Instructions for updating:
Use `tf.cast` instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/array_ops.py:1475: where (from tensorflow.python
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/training/monitored_session.py:882: start_queue_runne
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Saving checkpoints for 0 into /tmp/tmpq84db5pf/model.ckpt.
INFO:tensorflow:loss = 15.06291, step = 1
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable):
INFO:tensorflow:global_step/sec: 305.378
INFO:tensorflow:loss = 0.6026597, step = 101 (0.331 sec)
INFO:tensorflow:global_step/sec: 383.591
INFO:tensorflow:loss = 0.48607284, step = 201 (0.263 sec)
INFO:tensorflow:global_step/sec: 340.987
INFO:tensorflow:loss = 0.7062148, step = 301 (0.292 sec)
INFO:tensorflow:global_step/sec: 351.556
INFO:tensorflow:loss = 0.7458534, step = 401 (0.286 sec)
INFO:tensorflow:global_step/sec: 367.156
```

```
INFO:tensorflow:global_step/sec: 367.156
INFO:tensorflow:loss = 0.4673788, step = 501 (0.277 sec)
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable):
INFO:tensorflow:global_step/sec: 348.113
INFO:tensorflow:loss = 0.5876535, step = 601 (0.282 sec)
INFO:tensorflow:global_step/sec: 335.978
INFO:tensorflow:loss = 0.5779847, step = 701 (0.304 sec)
INFO:tensorflow:global_step/sec: 349.867
INFO:tensorflow:loss = 0.35653055, step = 801 (0.282 sec)
INFO:tensorflow:global_step/sec: 342.908
INFO:tensorflow:loss = 0.6800269, step = 901 (0.286 sec)
INFO:tensorflow:global_step/sec: 361.892
INFO:tensorflow:loss = 0.4379784, step = 1001 (0.278 sec)
INFO:tensorflow:global_step/sec: 388.338
INFO:tensorflow:loss = 0.4846738, step = 1101 (0.260 sec)
INFO:tensorflow:global_step/sec: 362.028
INFO:tensorflow:loss = 0.8479563, step = 1201 (0.275 sec)
INFO:tensorflow:global_step/sec: 356.745
INFO:tensorflow:loss = 0.6384973, step = 1301 (0.279 sec)
INFO:tensorflow:global_step/sec: 365.047
INFO:tensorflow:loss = 0.73553175, step = 1401 (0.286 sec)
INFO:tensorflow:global_step/sec: 360.929
INFO:tensorflow:loss = 0.6088743, step = 1501 (0.272 sec)
INFO:tensorflow:global_step/sec: 372.318
INFO:tensorflow:loss = 0.43618086, step = 1601 (0.265 sec)
INFO:tensorflow:global_step/sec: 377.558
INFO:tensorflow:loss = 0.33748716, step = 1701 (0.269 sec)
INFO:tensorflow:global_step/sec: 365.469
INFO:tensorflow:loss = 0.515324, step = 1801 (0.268 sec)
INFO:tensorflow:global_step/sec: 379.761
INFO:tensorflow:loss = 0.5078869, step = 1901 (0.262 sec)
INFO:tensorflow:global_step/sec: 365.97
INFO:tensorflow:loss = 0.7551081, step = 2001 (0.275 sec)
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable):
INFO:tensorflow:global_step/sec: 362.424
INFO:tensorflow:loss = 0.6637558, step = 2101 (0.280 sec)
INFO:tensorflow:global_step/sec: 350.996
INFO:tensorflow:loss = 0.4985267, step = 2201 (0.285 sec)
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable):
INFO:tensorflow:global_step/sec: 361.901
INFO:tensorflow:loss = 0.612228, step = 2301 (0.276 sec)
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable):
INFO:tensorflow:global_step/sec: 367.299
```

```
INFO:tensorflow:loss = 0.318715, step = 2401 (0.274 sec)
INFO:tensorflow:global_step/sec: 364.619
INFO:tensorflow:loss = 0.41287994, step = 2501 (0.272 sec)
INFO:tensorflow:global_step/sec: 379.345
INFO:tensorflow:loss = 0.4601137, step = 2601 (0.266 sec)
INFO:tensorflow:global_step/sec: 373.386
INFO:tensorflow:loss = 0.33889997, step = 2701 (0.263 sec)
INFO:tensorflow:global_step/sec: 370.809
INFO:tensorflow:loss = 0.501478, step = 2801 (0.275 sec)
INFO:tensorflow:global_step/sec: 379.512
INFO:tensorflow:loss = 0.52347434, step = 2901 (0.261 sec)
INFO:tensorflow:global_step/sec: 361.384
INFO:tensorflow:loss = 0.8358942, step = 3001 (0.281 sec)
INFO:tensorflow:global_step/sec: 365.201
INFO:tensorflow:loss = 0.4524899, step = 3101 (0.273 sec)
INFO:tensorflow:global_step/sec: 339.755
INFO:tensorflow:loss = 0.4218226, step = 3201 (0.296 sec)
INFO:tensorflow:global_step/sec: 327.184
INFO:tensorflow:loss = 0.5360174, step = 3301 (0.295 sec)
INFO:tensorflow:global_step/sec: 367.384
INFO:tensorflow:loss = 0.6250687, step = 3401 (0.284 sec)
INFO:tensorflow:global_step/sec: 361.507
INFO:tensorflow:loss = 0.59657526, step = 3501 (0.268 sec)
INFO:tensorflow:global_step/sec: 379.205
INFO:tensorflow:loss = 0.6881917, step = 3601 (0.267 sec)
INFO:tensorflow:global_step/sec: 359.347
INFO:tensorflow:loss = 0.46599433, step = 3701 (0.279 sec)
INFO:tensorflow:global_step/sec: 369.808
INFO:tensorflow:loss = 0.5261489, step = 3801 (0.280 sec)
INFO:tensorflow:global_step/sec: 369.363
INFO:tensorflow:loss = 0.42467988, step = 3901 (0.262 sec)
INFO:tensorflow:global_step/sec: 363.709
INFO:tensorflow:loss = 0.60812515, step = 4001 (0.277 sec)
INFO:tensorflow:global_step/sec: 373.847
INFO:tensorflow:loss = 0.48772526, step = 4101 (0.261 sec)
INFO:tensorflow:global_step/sec: 349.967
INFO:tensorflow:loss = 0.49208063, step = 4201 (0.289 sec)
INFO:tensorflow:global_step/sec: 361.005
INFO:tensorflow:loss = 0.49653152, step = 4301 (0.272 sec)
INFO:tensorflow:global_step/sec: 378.448
INFO:tensorflow:loss = 0.33538777, step = 4401 (0.274 sec)
INFO:tensorflow:global_step/sec: 377.453
```

```
INFO:tensorflow:loss = 0.44626743, step = 4501 (0.262 sec)
INFO:tensorflow:global_step/sec: 342.044
INFO:tensorflow:loss = 0.7130273, step = 4601 (0.289 sec)
INFO:tensorflow:global_step/sec: 339.296
INFO:tensorflow:loss = 0.9449788, step = 4701 (0.297 sec)
INFO:tensorflow:global_step/sec: 338.195
INFO:tensorflow:loss = 0.85044324, step = 4801 (0.303 sec)
INFO:tensorflow:global_step/sec: 341.468
INFO:tensorflow:loss = 0.6653259, step = 4901 (0.281 sec)
INFO:tensorflow:global_step/sec: 367.017
INFO:tensorflow:loss = 0.8402523, step = 5001 (0.278 sec)
INFO:tensorflow:global_step/sec: 370.626
INFO:tensorflow:loss = 0.68081677, step = 5101 (0.264 sec)
INFO:tensorflow:global_step/sec: 377.321
INFO:tensorflow:loss = 0.87028193, step = 5201 (0.267 sec)
INFO:tensorflow:global_step/sec: 371.677
INFO:tensorflow:loss = 0.34389257, step = 5301 (0.269 sec)
INFO:tensorflow:global_step/sec: 387.547
INFO:tensorflow:loss = 0.72975075, step = 5401 (0.261 sec)
INFO:tensorflow:global_step/sec: 357.392
INFO:tensorflow:loss = 0.55152595, step = 5501 (0.277 sec)
INFO:tensorflow:global_step/sec: 381.731
INFO:tensorflow:loss = 0.845948, step = 5601 (0.259 sec)
INFO:tensorflow:global_step/sec: 357.153
INFO:tensorflow:loss = 0.50020397, step = 5701 (0.282 sec)
INFO:tensorflow:global_step/sec: 377.763
INFO:tensorflow:loss = 0.5048729, step = 5801 (0.270 sec)
INFO:tensorflow:global_step/sec: 359.676
INFO:tensorflow:loss = 0.5006413, step = 5901 (0.273 sec)
INFO:tensorflow:global_step/sec: 369.435
INFO:tensorflow:loss = 0.61612606, step = 6001 (0.280 sec)
INFO:tensorflow:global_step/sec: 311.514
INFO:tensorflow:loss = 0.6315159, step = 6101 (0.312 sec)
INFO:tensorflow:global_step/sec: 346.822
INFO:tensorflow:loss = 0.529412, step = 6201 (0.287 sec)
INFO:tensorflow:global_step/sec: 332.43
INFO:tensorflow:loss = 0.40272892, step = 6301 (0.300 sec)
INFO:tensorflow:global_step/sec: 341.936
INFO:tensorflow:loss = 0.6275656, step = 6401 (0.291 sec)
INFO:tensorflow:global_step/sec: 376.308
INFO:tensorflow:loss = 0.5658413, step = 6501 (0.269 sec)
INFO:tensorflow:global_step/sec: 325.078
```

```
INFO:tensorflow:loss = 0.46921533, step = 6601 (0.304 sec)
INFO:tensorflow:global_step/sec: 340.416
INFO:tensorflow:loss = 0.62076014, step = 6701 (0.294 sec)
INFO:tensorflow:global_step/sec: 344.307
INFO:tensorflow:loss = 0.69967264, step = 6801 (0.297 sec)
INFO:tensorflow:global_step/sec: 335.504
INFO:tensorflow:loss = 0.54543626, step = 6901 (0.297 sec)
INFO:tensorflow:global_step/sec: 340.843
INFO:tensorflow:loss = 0.32576382, step = 7001 (0.295 sec)
INFO:tensorflow:global_step/sec: 331.43
INFO:tensorflow:loss = 0.4732263, step = 7101 (0.296 sec)
INFO:tensorflow:global_step/sec: 343.524
INFO:tensorflow:loss = 0.46608585, step = 7201 (0.289 sec)
INFO:tensorflow:global_step/sec: 342.552
INFO:tensorflow:loss = 0.54833156, step = 7301 (0.292 sec)
INFO:tensorflow:global_step/sec: 355.55
INFO:tensorflow:loss = 0.6329608, step = 7401 (0.283 sec)
INFO:tensorflow:global_step/sec: 334.963
INFO:tensorflow:loss = 0.56207675, step = 7501 (0.297 sec)
INFO:tensorflow:global_step/sec: 342.642
INFO:tensorflow:loss = 0.47871095, step = 7601 (0.292 sec)
INFO:tensorflow:global_step/sec: 338.925
INFO:tensorflow:loss = 0.36805856, step = 7701 (0.295 sec)
INFO:tensorflow:global_step/sec: 344.376
INFO:tensorflow:loss = 0.5934015, step = 7801 (0.290 sec)
INFO:tensorflow:global_step/sec: 357.578
INFO:tensorflow:loss = 0.6276143, step = 7901 (0.280 sec)
INFO:tensorflow:global_step/sec: 345.8
INFO:tensorflow:loss = 0.7121178, step = 8001 (0.290 sec)
INFO:tensorflow:global_step/sec: 352.386
INFO:tensorflow:loss = 0.362469, step = 8101 (0.285 sec)
```

```
# calculo das metricas de treinamento.
metricas_treinamento = regressor.evaluate(input_fn=funcao_treinamento, steps = 10000)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2020-04-28T18:33:12Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpq84db5pf/model.ckpt-10000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Evaluation [1000/10000]
INFO:tensorflow:Evaluation [2000/10000]
INFO:tensorflow:Evaluation [3000/10000]
INFO:tensorflow:Evaluation [4000/10000]
INFO:tensorflow:Evaluation [5000/10000]
INFO:tensorflow:Evaluation [6000/10000]
INFO:tensorflow:Evaluation [7000/10000]
INFO:tensorflow:Evaluation [8000/10000]
INFO:tensorflow:Evaluation [9000/10000]
INFO:tensorflow:Evaluation [10000/10000]
INFO:tensorflow:Finished evaluation at 2020-04-28-18:33:36
INFO:tensorflow:Saving dict for global step 10000: average_loss = 0.017598214, global_step = 10000, label/mean = 0.66980094, los
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 10000: /tmp/tmpq84db5pf/model.ckpt-10000


INFO:tensorflow:global step/sec: 364.856
```

TESTES...

```
INFO:tensorflow:Loss for final step: 0.8297609.
```

```
# calculo das metricas de testes. (função de avaliação)
metricas_teste = regressor.evaluate(input_fn=funcao_teste, steps = 10000)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2020-04-28T18:33:40Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpq84db5pf/model.ckpt-10000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Evaluation [1000/10000]
INFO:tensorflow:Evaluation [2000/10000]
INFO:tensorflow:Evaluation [3000/10000]
INFO:tensorflow:Evaluation [4000/10000]
INFO:tensorflow:Evaluation [5000/10000]
INFO:tensorflow:Evaluation [6000/10000]
INFO:tensorflow:Evaluation [7000/10000]
INFO:tensorflow:Evaluation [8000/10000]
INFO:tensorflow:Evaluation [9000/10000]
INFO:tensorflow:Finished evaluation at 2020-04-28-18:34:03
INFO:tensorflow:Saving dict for global step 10000: average_loss = 0.044323128, global_step = 10000, label/mean = 0.62359107, los
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 10000: /tmp/tmpq84db5pf/model.ckpt-10000
```

```
# exibindo valores da métrica.
metricas_treinamento
```

```
{'average_loss': 0.017598214,
 'global_step': 10000,
 'label/mean': 0.66980094,
 'loss': 0.56314284,
 'prediction/mean': 0.6600661}
```

```
# exibindo valores da métrica. (valores para análise de resultados.)
metricas_teste
```

```
{'average_loss': 0.044323128,
 'global_step': 10000,
 'label/mean': 0.62359107,
 'loss': 1.4183401,
 'prediction/mean': 0.67811567}
```

PREVISÃO...

```
# função de previsão, gera os valores para cada um dos valores de X.
funcao_previsao = tf.estimator.inputs.pandas_input_fn(x = X_teste, shuffle = False)


previsoes = regressor.predict(input_fn=funcao_previsao)


# exibe valores previstos.
list(previsoes)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpq84db5pf/model.ckpt-10000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
[{'predictions': array([0.85198224], dtype=float32)},
 {'predictions': array([0.62288594], dtype=float32)},
 {'predictions': array([0.5665797], dtype=float32)},
 {'predictions': array([0.5752864], dtype=float32)},
 {'predictions': array([0.7422537], dtype=float32)},
 {'predictions': array([0.6310452], dtype=float32)},
 {'predictions': array([0.7804037], dtype=float32)},
 {'predictions': array([0.6404171], dtype=float32)},
 {'predictions': array([0.8282819], dtype=float32)},
 {'predictions': array([0.65532637], dtype=float32)},
 {'predictions': array([0.8542887], dtype=float32)},
 {'predictions': array([0.6400654], dtype=float32)},
 {'predictions': array([0.6603379], dtype=float32)},
 {'predictions': array([0.7073367], dtype=float32)},
 {'predictions': array([0.6310492], dtype=float32)},
 {'predictions': array([0.72443116], dtype=float32)},
 {'predictions': array([0.5688299], dtype=float32)},
 {'predictions': array([0.6192843], dtype=float32)},
 {'predictions': array([0.5694825], dtype=float32)},
 {'predictions': array([0.66267145], dtype=float32)},
 {'predictions': array([0.62840915], dtype=float32)},
 {'predictions': array([0.7364763], dtype=float32)},
 {'predictions': array([0.81638044], dtype=float32)},
 {'predictions': array([0.52159953], dtype=float32)},
 {'predictions': array([0.68060404], dtype=float32)},
 {'predictions': array([0.72323275], dtype=float32)},
 {'predictions': array([0.7231647], dtype=float32)},
 {'predictions': array([0.62855995], dtype=float32)},
 {'predictions': array([0.63377136], dtype=float32)},
 {'predictions': array([0.71856046], dtype=float32)}]
```

```
# para o calculo do erro, vamos criar um array, que irá receber cada um dos valores gerados pela função previsão.
valores_previsoes = []
for p in regressor.predict(input_fn=funcao_previsao):
```

```
for p in regressor.predict(input_fn = funcao_previsao):
    valores_previsoes.append(p['predictions'])
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpq84db5pf/model.ckpt-10000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```

```
# exibe array com os valores.
# com esse valores iremos calcular os erros MSE e MAE.
valores_previsoes
```

```
[array([0.85198224], dtype=float32),
 array([0.62288594], dtype=float32),
 array([0.5665797], dtype=float32),
 array([0.5752864], dtype=float32),
 array([0.7422537], dtype=float32),
 array([0.6310452], dtype=float32),
 array([0.7804037], dtype=float32),
 array([0.6404171], dtype=float32),
 array([0.8282819], dtype=float32),
 array([0.65532637], dtype=float32),
 array([0.8542887], dtype=float32),
 array([0.6400654], dtype=float32),
 array([0.6603379], dtype=float32),
 array([0.7073367], dtype=float32),
 array([0.6310492], dtype=float32),
 array([0.72443116], dtype=float32),
 array([0.5688299], dtype=float32),
 array([0.6192843], dtype=float32),
 array([0.5694825], dtype=float32),
 array([0.66267145], dtype=float32),
 array([0.62840915], dtype=float32),
 array([0.7364763], dtype=float32),
 array([0.81638044], dtype=float32),
 array([0.52159953], dtype=float32),
 array([0.68060404], dtype=float32),
 array([0.72323275], dtype=float32),
 array([0.7231647], dtype=float32),
 array([0.62855995], dtype=float32),
 array([0.63377136], dtype=float32),
 array([0.71856046], dtype=float32)]
```

```
# como os valores foram normalizados, precisamos desnormaliza-los.
import numpy as np
# colocar no formato de matriz.
valores_previsoes = np.asarray(valores_previsoes).reshape(-1,1)
```

```
valores_previsoes
```

```
array([[0.85198224],
       [0.62288594],
       [0.5665797 ],
       [0.5752864 ],
       [0.7422537 ],
       [0.6310452 ],
       [0.7804037 ],
       [0.6404171 ],
       [0.8282819 ],
       [0.65532637],
       [0.8542887 ],
       [0.6400654 ],
       [0.6603379 ],
       [0.7073367 ],
       [0.6310492 ],
       [0.72443116],
       [0.5688299 ],
       [0.6192843 ],
       [0.5694825 ],
       [0.66267145],
       [0.62840915],
       [0.7364763 ],
       [0.81638044],
       [0.52159953],
       [0.68060404],
       [0.72323275],
       [0.7231647 ],
       [0.62855995],
       [0.63377136],
       [0.71856046]], dtype=float32)
```

```
valores_previsoes = scaler_y.inverse_transform(valores_previsoes)
valores_previsoes
```

```
array([[0.85198224],
       [0.62288594],
       [0.5665797 ],
       [0.5752864 ],
       [0.7422537 ],
       [0.6310452 ],
       [0.7804037 ],
       [0.6404171 ],
       [0.8282819 ],
       [0.65532637],
       [0.8542887 ],
       [0.6400654 ],
       [0.6603379 ],
       [0.7073367 ],
       [0.6310492 ],
       [0.72443116],
       [0.5688299 ],
       [0.6192843 ],
       [0.5694825 ],
       [0.66267145],
       [0.62840915],
       [0.7364763 ],
       [0.81638044],
       [0.52159953],
       [0.68060404],
       [0.72323275],
       [0.7231647 ],
       [0.62855995],
       [0.63377136],
       [0.71856046]], dtype=float32)
```

```
y_teste
```

```
4     0.524664
39    0.433483
61    0.635277
9     0.713004
55    0.636771
71    0.871450
49    0.762332
97    0.707025
0     0.847534
59    0.837070
42    0.949178
84    0.857997
43    0.729447
1     0.651719
2     0.600897
19    0.778774
90    0.793722
93    0.726457
11    0.590433
80    0.692078
87    0.523169
98    0.597907
57    0.666667
66    0.083707
20    0.426009
34    0.766816
70    0.385650
99    0.000000
86    0.346786
6     0.571001
Name: danceability, dtype: float64
```

```
y_teste.shape
```

```
(30,)
```

```
y_teste2 = y_teste.values.reshape(-1,1)
y_teste2.shape
```

```
(30, 1)
```

```
y_teste2 = scaler_y.inverse_transform(y_teste2)
```

```
y_teste2
```

```
array([[0.52466368],
       [0.43348281],
       [0.63527653],
       [0.71300448],
       [0.6367713 ],
       [0.87144993],
       [0.76233184],
       [0.70702541],
       [0.84753363],
       [0.83707025],
       [0.94917788],
       [0.85799701],
       [0.72944694],
       [0.65171898],
       [0.60089686],
       [0.77877429],
       [0.79372197],
       [0.7264574 ],
       [0.59043348],
       [0.69207773],
       [0.52316891],
       [0.59790732],
       [0.66666667],
       [0.08370703],
       [0.42600897],
       [0.76681614],
       [0.38565022],
       [0.        ],
       [0.34678625],
       [0.57100149]])
```

```
from sklearn.metrics import mean_absolute_error
```

```
mae = mean_absolute_error(y_teste2, valores_previsoes)
```

```
mae
```

```
0.15964591643616638
```

Conclusão

Chegamos a conclusão que nivel de dansabilidade de uma track do spotify influência diretamente no seu posicionamento das musicais mais escutas, top100, no spotify