Hit PREDICTOR

Binary classification on a Spotify Dataset

Texto

Presentación del conjunto de datos :

Estudio de las características que caracterizan a los temas más escuchados estos últimos años (TOPLIST) en SPOTIFY .
Seleccionaremos las canciones más populares (track_popularity > 70) para determinar si existen unas características sónicas (features) propias que las definan.
Cual alquimistas intentaremos encontrar " la fórmula mágica" que determina la popularidad o no de una canción.
Una vez tengamos esta "fórmula" , testaremos en otro dataset para ver si descubrimos canciones que por "fórmula" deberían ser populares pero no lo fueron.

El dataset con el entrenaremos al modelo contiene 32000 canciones y sus características sónicas (Tono, escala, y coef.sintéticos) así como su popularidad.
El dataset con el que "testaremos" el modelo contiene 1M. canciones con sus características y su popularidad.

TRACK_POPULARITY : Popularidad de una canción. Se entiende que se estima según las escuchas. Cuanto más cercano a 100, más popular ha sido la canción (más escuchas).
KEY : Clave de la canción por semitonos . Así, 1 es DO, 2 es DO#, 3 es RE..etc hasta SI = 11. Var. Categórica

LOUDNESS : "Volumen" medio para una canción en dB .

MODE : Tono MAYOR o MENOR de la canción . 1 si es MAYOR , 0 si es tono menor . Var. Categórica

SPEECHNESS : Determina la presencia de "partes" habladas en la canción. Así para podcasts / RAP/ Spoken words este suele ser > 70 así como valores < 30 suelen definir temas musicales. ACCOUSTICNESS: Acusticidad (vs electricidad) de las canciones . Canciones con instrumentos acústicos tendrán valores cercanos a 1 y presuponemos que temas más con instrumentos y timbres más eléctricos o electrónicos tendrán valores cercanos a 0 .

INSTRUMENTALNESS: Define la presencia o no de partes vocales en la canción. Cuanto más se acerca a 1 más instrumental es la canción. Esperamos valores cercanos a 1 para JAZZ, CLASSICA, ETC

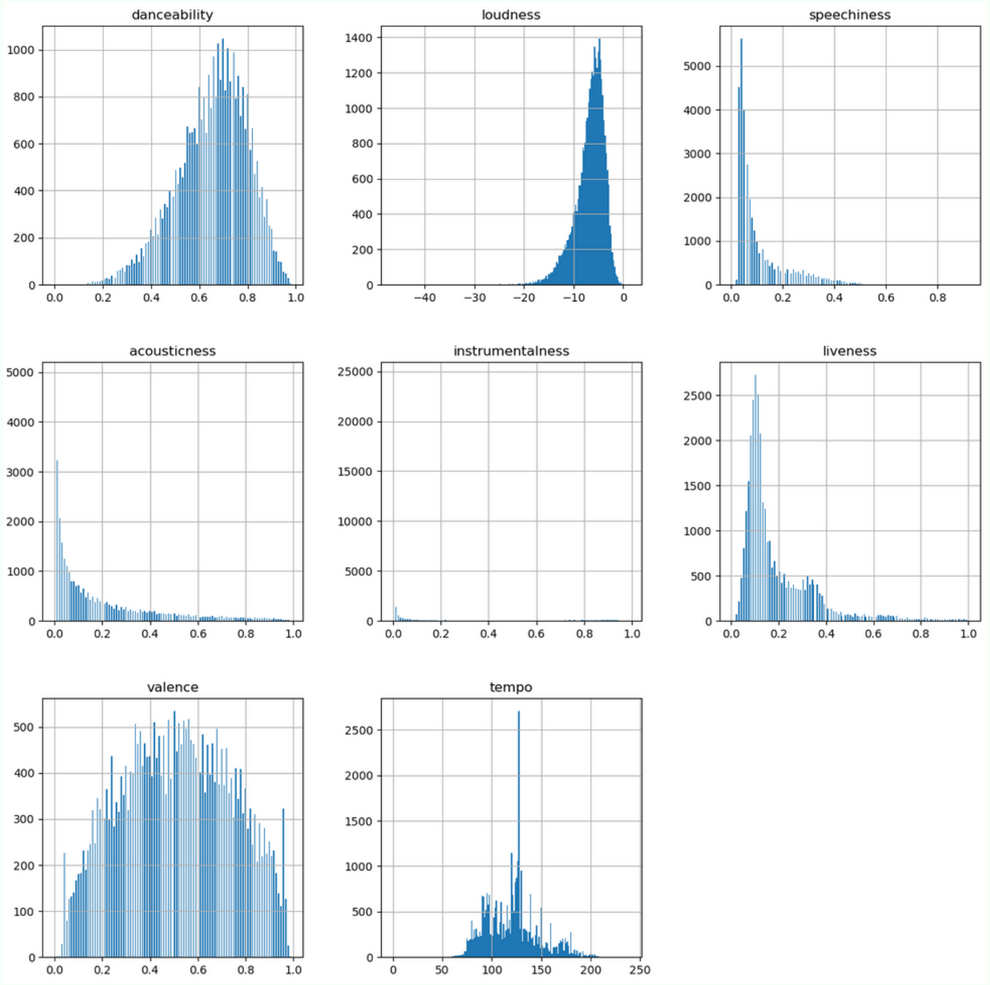
LIVENESS: Define la presencia o no de público durante la grabación, define si la canción ha sido gravada en Directo o en estudio.

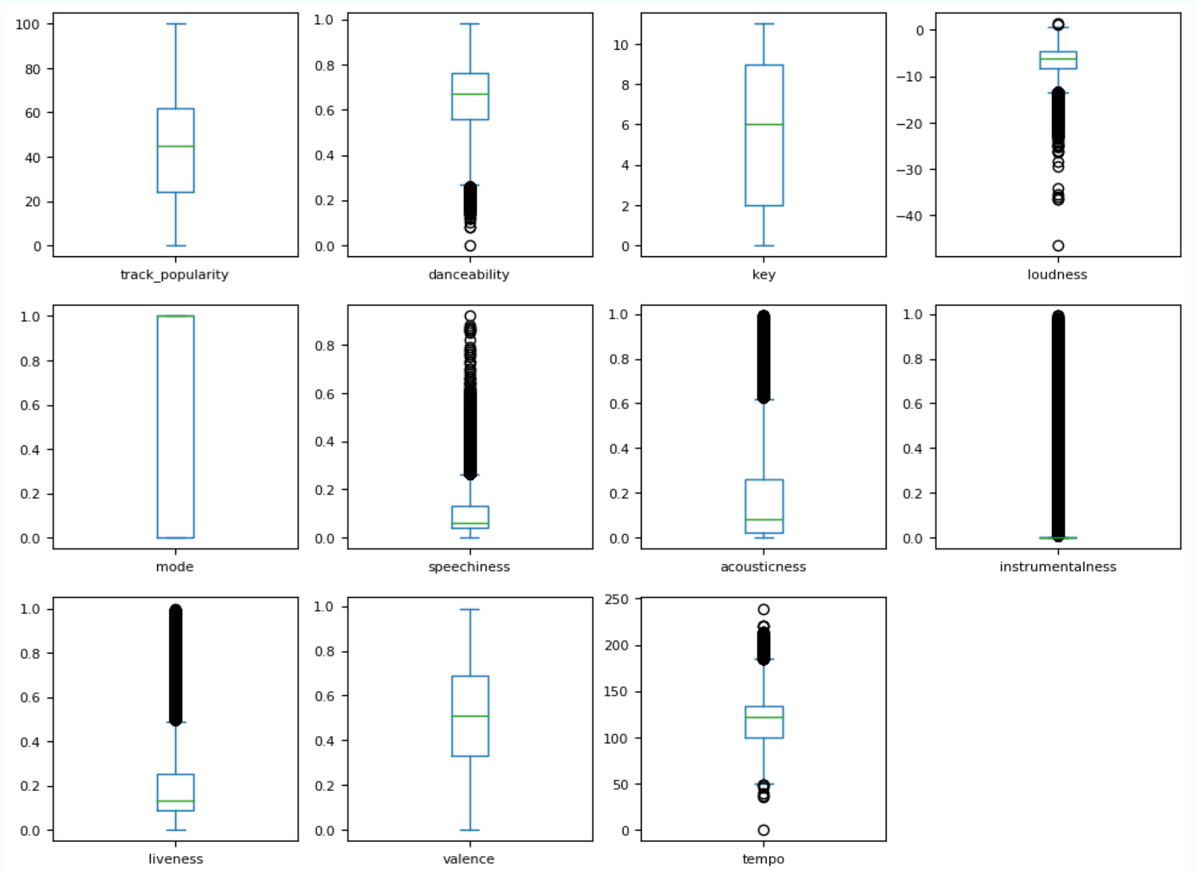
VALENCE : Define si una canción es " triste" (cercana a 0) o "alegre" (cercana a 1)

TEMPO : Tempo de la canción en BPM

DURATION : Duración de la canción en Msegundos

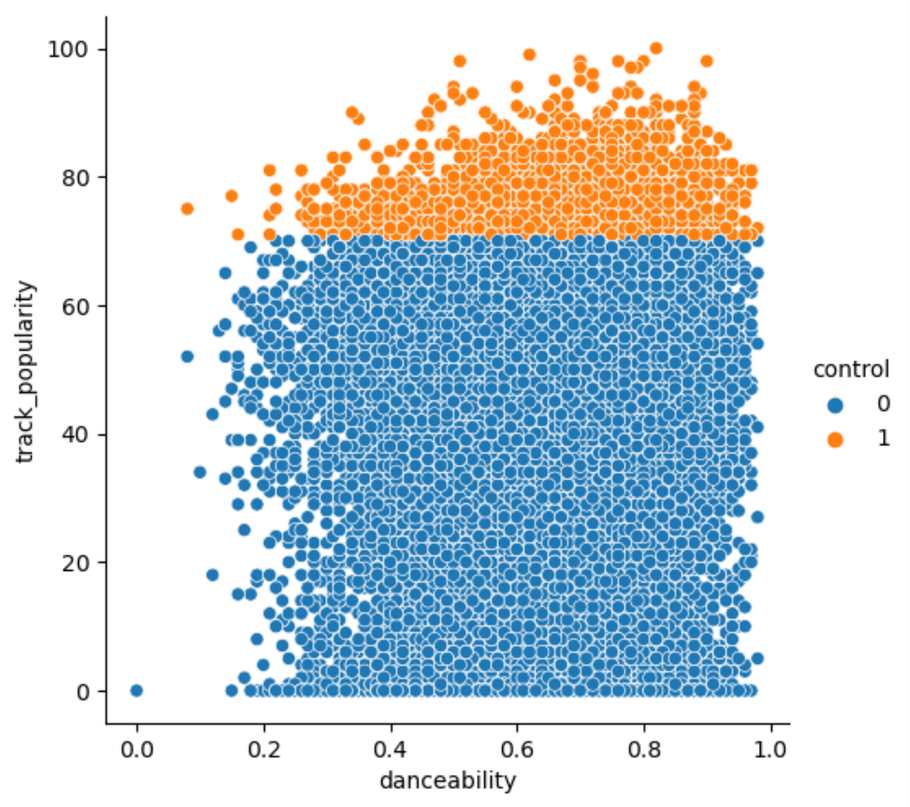
columns_numericas = df.select_dtypes(include=['float64'])

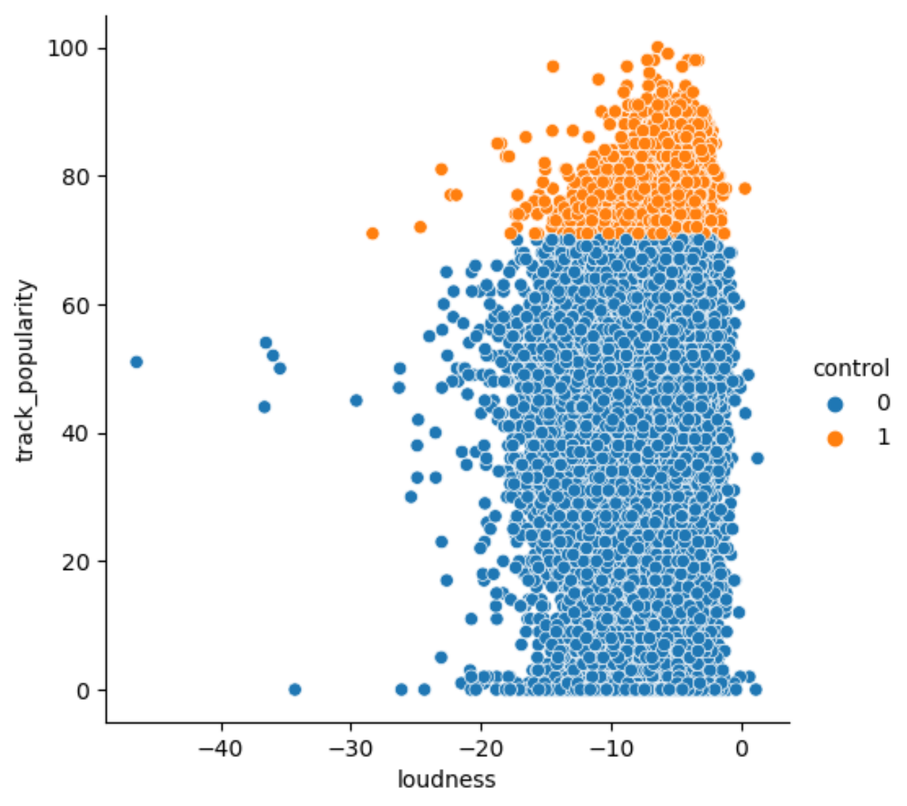


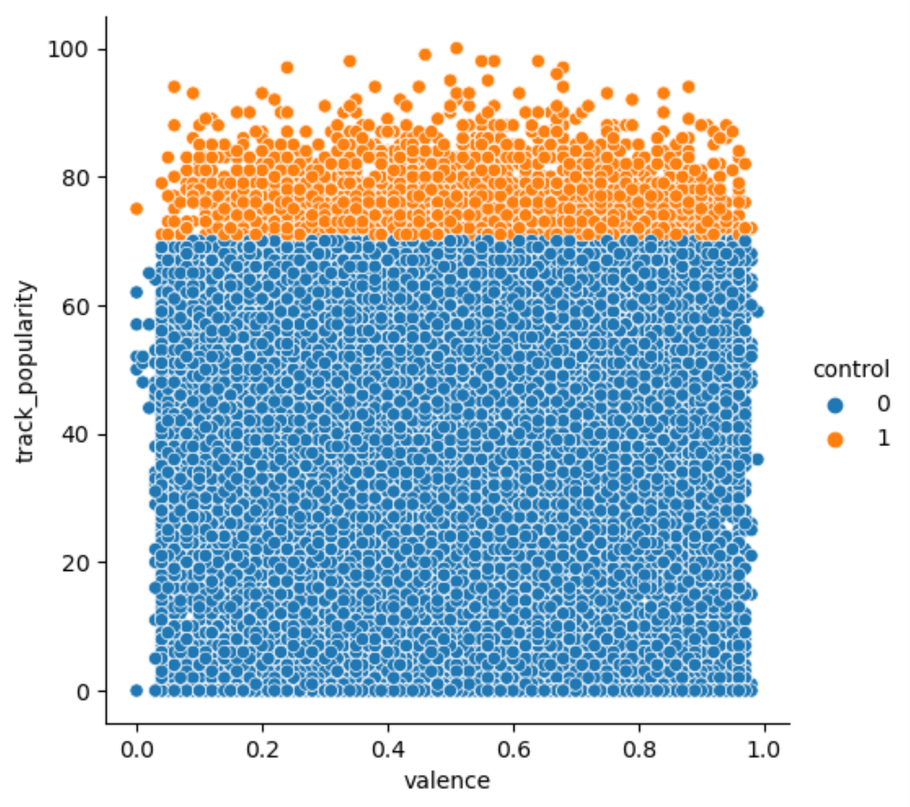

track_popularity
danceability
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo

```
df['control'] = df['track_popularity'].apply(lambda x: 1 if x > 70 else  0)
```

	track_popularity	danceability	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	control
0	66	0.75	6	-2.63	1	0.06	0.10	0.0	0.07	0.52	122.04	0
1	67	0.73	11	-4.97	1	0.04	0.07	0.0	0.36	0.69	99.97	0
2	70	0.68	1	-3.43	0	0.07	0.08	0.0	0.11	0.61	124.01	0
3	60	0.72	7	-3.78	1	0.10	0.03	0.0	0.20	0.28	121.96	0
4	69	0.65	1	-4.67	1	0.04	0.08	0.0	0.08	0.72	123.98	0


100
80
60
40
20
0
track_popularity
0.0
0.2
0.4
0.6
0.8
1.0
danceability
control
0
1


100
80
60
40
20
0
track_popularity
−40
−30
−20
−10
0
loudness
control
0
1


100
80
60
40
20
0
track_popularity
0.0
0.2
0.4
0.6
0.8
1.0
valence
control
0
1


tr
track_popularity
danceability
key
loudness
mode
speechiness
acousticness
ins
instrumentalness
liveness
valence
tempo
control
100.00% 13.59% 6.85% 1.78% -6.10% 7.65% 8.45% -1.44% -4.77% -5.17% -2.49%
13.59% 100.00% 2.45% 12.97% -10.14% 19.94% -5.01% -10.10% -10.01% 30.93% -15.15%
6.85% 2.45% 100.00% 0.78% -13.40% 2.79% -0.21% -3.06% 2.01% -0.24% -5.88%
1.78% 12.97% 0.78% 100.00% -1.61% -1.49% -30.68% -18.01% 4.43% 24.03% 5.30%
-6.10% -10.14% -13.40% -1.61% 100.00% -9.38% 3.96% -2.63% -2.38% 0.59% 3.77%
7.65% 19.94% 2.79% -1.49% -9.38% 100.00% 1.09% -3.89% -0.83% 3.08% 17.75%
8.45% -5.01% -0.21% -30.68% 3.96% 1.09% 100.00% 3.26% -2.82% -9.01% -5.72%
-1.44% -10.10% -3.06% -18.01% -2.63% -3.89% 3.26% 100.00% -0.73% -7.27% 1.55%
-4.77% -10.01% 2.01% 4.43% -2.38% -0.83% -2.82% -0.73% 100.00% 0.84% 1.35%
-5.17% 30.93% -0.24% 24.03% 0.59% 3.08% -9.01% -7.27% 0.84% 100.00% 1.16%
-2.49% -15.15% -5.88% 5.30% 3.77% 17.75% -5.72% 1.55% 1.35% 1.16% 100.00%
track_popularity
danceability
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
control

No vemos grandes diferencias en cuanto a correlaciones x feature y grupo de control. Si acaso :

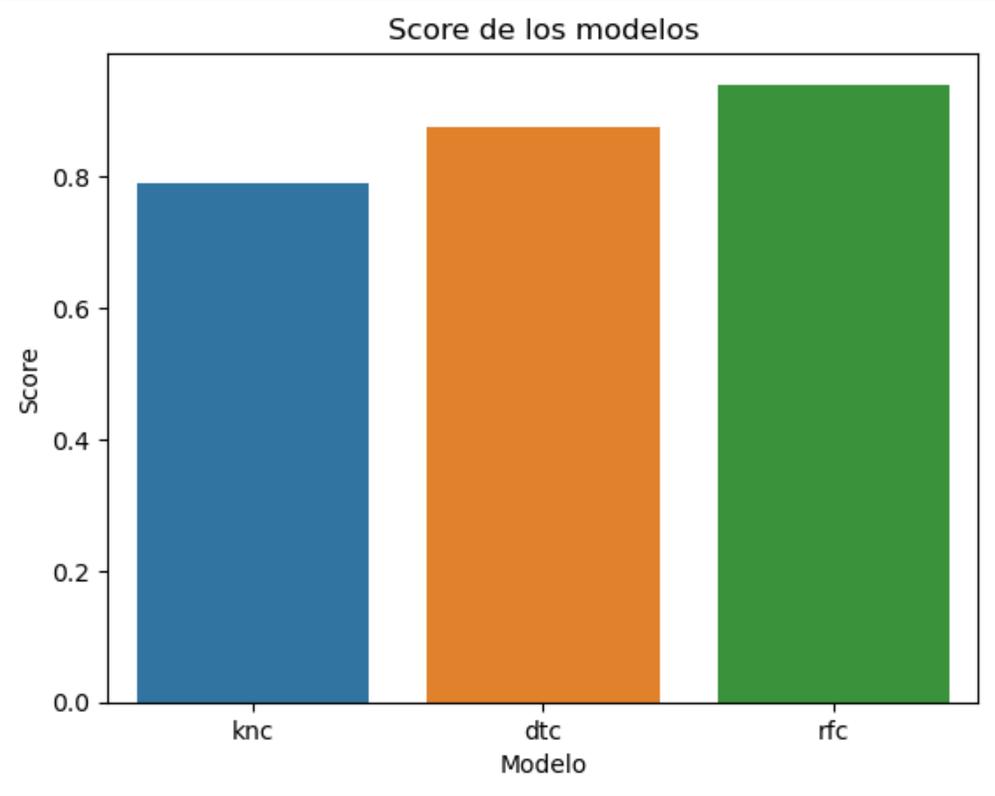
Danceability : 0.13 TOP vs 0.03 BOTTOM ...las canciones TOP suelen ser ligeramente más bailables.

Speechness : 0.07 TOP vs 0.004 BOTTOM ...las canciones TOP suelen incluir más partes "habladas", mas "Lyrics".

Instrumentalness : -0.014 TOP vs - 0.09 BOTTOM .. las canciones TOP son menos instrumentales que las canciones BOTTOM. Confirma en cierta forma la importancia de las lyrics en cuanto a la popularidad de una canción (que no calidad)

Valence : -0.051 TOP vs 0.02 BOTTOM ...las canciones TOP guardan una mínima correlación negativa con Valence y las canciones BOTTOM guardan una mínima correlación positiva con VALENCE. las canciones TOP suelen ser más alegres que las canciones BOTTOM.

T



Scores analísis previo Modelos CrossValidation

y = df['control']

```
1 y.value_counts()
```

```
0    28404
1     4424
Name: control, dtype: int64
```

```
from imblearn.over_sampling import SMOTE

# Implement SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X, y)
```

```
1 y_train_smote.value_counts()
```

```
   28404
   28404
me: control, dtype: int64
```

Utilizaremos lazypredict para ver que algoritmo es más adecuado.

```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(X_train_scaled, X_test_scaled, y_train, y_test)
models
```

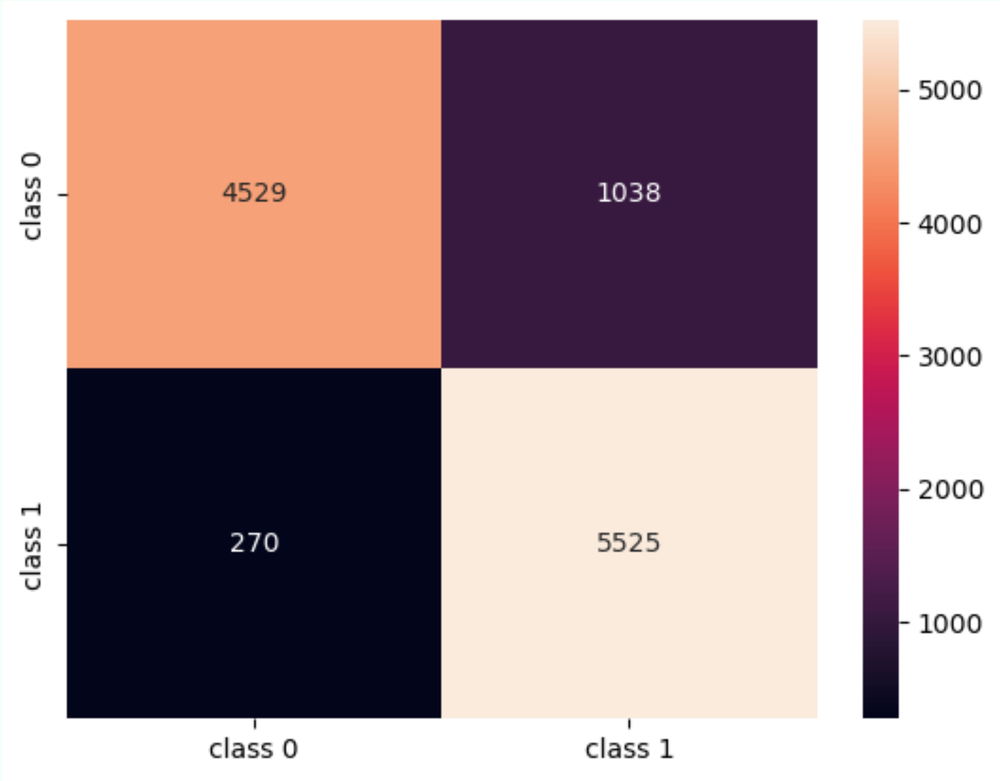
Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
RandomForestClassifier	0.95	0.95	0.95	0.95	15.84
ExtraTreesClassifier	0.94	0.94	0.94	0.94	5.38
BaggingClassifier	0.93	0.93	0.93	0.93	4.49
XGBClassifier	0.91	0.91	0.91	0.91	0.71
DecisionTreeClassifier	0.88	0.88	0.88	0.88	0.54
LGBMClassifier	0.88	0.88	0.88	0.88	0.74
ExtraTreeClassifier	0.84	0.84	0.84	0.84	0.08
KNeighborsClassifier	0.83	0.83	0.83	0.83	3.09
NuSVC	0.79	0.79	0.79	0.79	527.00
SVC	0.73	0.73	0.73	0.73	185.11
AdaBoostClassifier	0.70	0.70	0.70	0.70	3.05
CalibratedClassifierCV	0.63	0.63	0.63	0.63	29.52
LogisticRegression	0.63	0.63	0.63	0.62	0.15
LinearSVC	0.63	0.63	0.63	0.62	14.98
LinearDiscriminantAnalysis	0.63	0.62	0.62	0.62	0.80
RidgeClassifier	0.63	0.62	0.62	0.62	0.08
RidgeClassifierCV	0.63	0.62	0.62	0.62	0.12
NearestCentroid	0.62	0.62	0.62	0.62	0.08
SGDClassifier	0.62	0.61	0.61	0.61	0.27

]:
```python
#  Parametros sobre los que vamos a iterar CV

param_grid = {'algorithm': ['auto'],
'leaf_size': [100,30,20,10],
    'metric': ['minkowski'],
        'n_neighbors': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,100,200,500],
            'p': [2,3,4],
                'weights': ['uniform']}
```

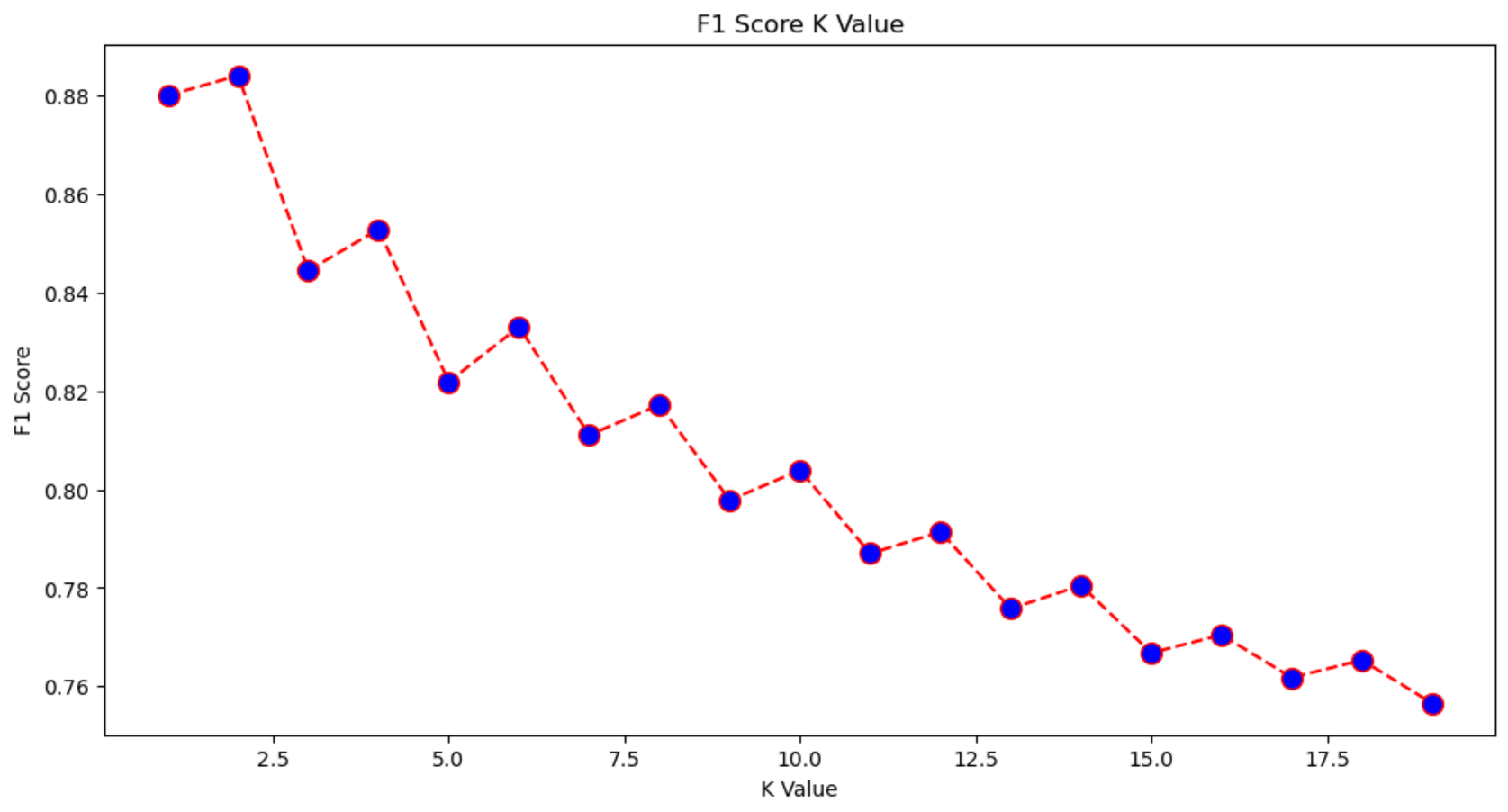
]:
```python
# RandomizedSearchCV y fit()del modelo KNN

grid_search1 = RandomizedSearchCV(knn, param_grid, n_iter=20, cv=5, verbose = 2, scoring='accuracy', n_jobs=-1)



grid_search1.fit(X_train_scaled, y_train)
grid_search1.best_params_

```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

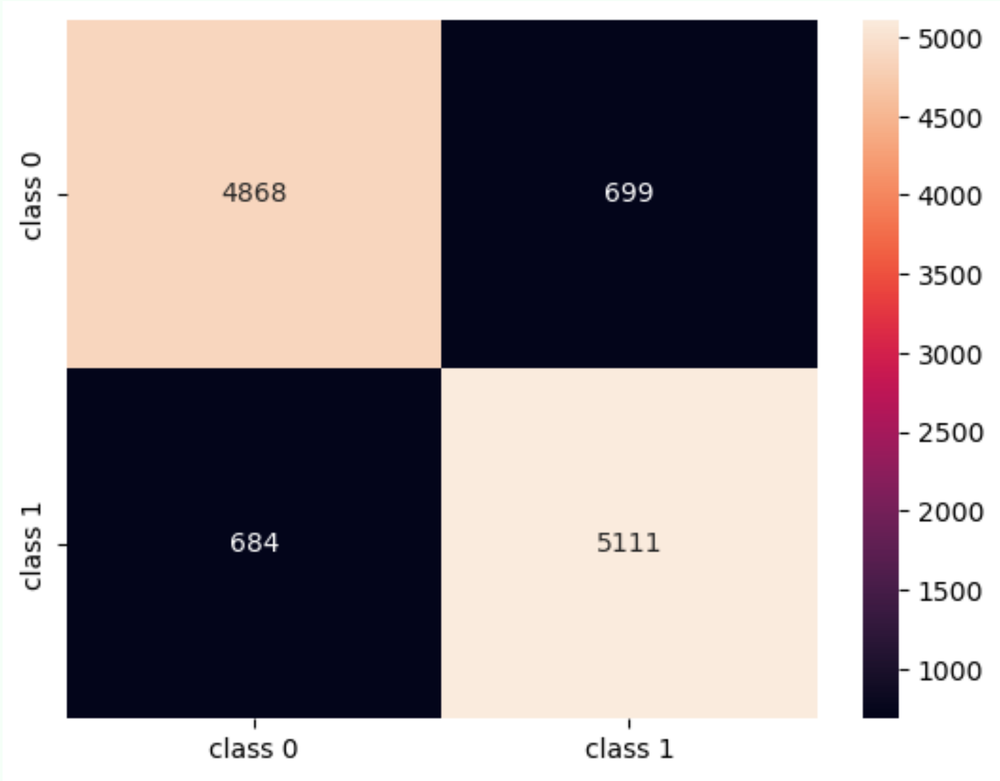
]:
```
{'weights': 'uniform',
 'p': 2,
 'n_neighbors': 1,
 'metric': 'minkowski',
 'leaf_size': 10,
 'algorithm': 'auto'}
```



Heatmap Accuracy para Best Model



Método K.Elbow para k.óptimo = 2



Heatmap Accuracy para K = 2

	precision	recall	f1-score	support
0	0.94	0.81	0.87	5567
1	0.84	0.95	0.89	5795
accuracy			0.88	11362
macro avg	0.89	0.88	0.88	11362
weighted avg	0.89	0.88	0.88	11362

	precision	recall	f1-score	support
0	0.88	0.87	0.88	5567
1	0.88	0.88	0.88	5795
accuracy			0.88	11362
macro avg	0.88	0.88	0.88	11362
weighted avg	0.88	0.88	0.88	11362

	popularity	danceability	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	control	predicted_popularity
0	68	0.48	4	-10.06	1	0.04	0.69	0.00	0.12	0.14	133.41	0	0
1	50	0.57	3	-10.29	1	0.03	0.48	0.00	0.10	0.52	140.18	0	0
2	57	0.41	3	-13.71	1	0.03	0.34	0.00	0.09	0.14	139.83	0	0
3	58	0.39	10	-9.85	1	0.04	0.81	0.00	0.08	0.51	204.96	0	0
4	54	0.43	6	-5.42	0	0.03	0.07	0.02	0.11	0.22	171.86	0	0

```
df2['control'] = df2['popularity'].apply(lambda x: 1 if x > 70 else  0)
```

```
predicted_popularity = best_model_knn.predict(df3)
```

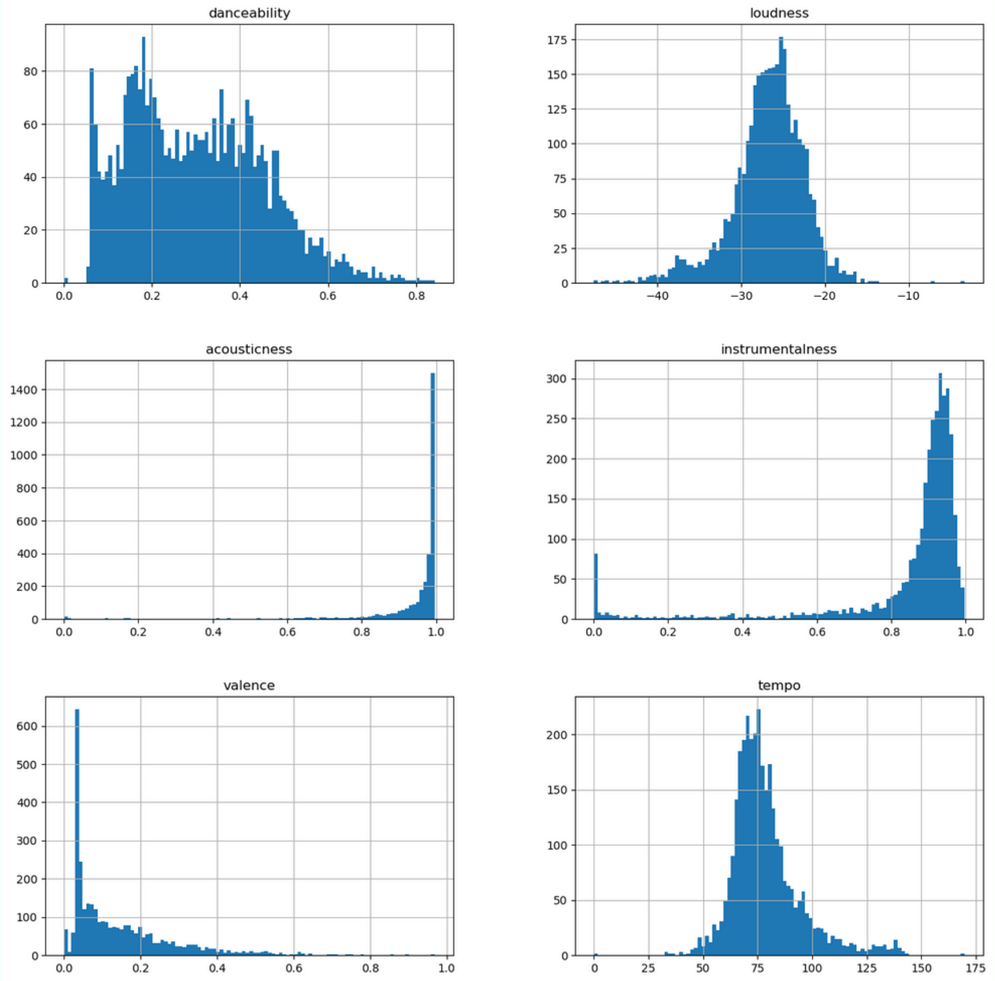
 # Buscamos las "perlas", canciones que por características (segun algoritmo de class por features sónicas) deberían ser populares (predicted_popularity = 1) y que sin embargo no alcanzaron popularidad (control = 0)

df_pearl = df2.query('control == 0 & predicted_popularity == 1 & 30 <popularity < 60')

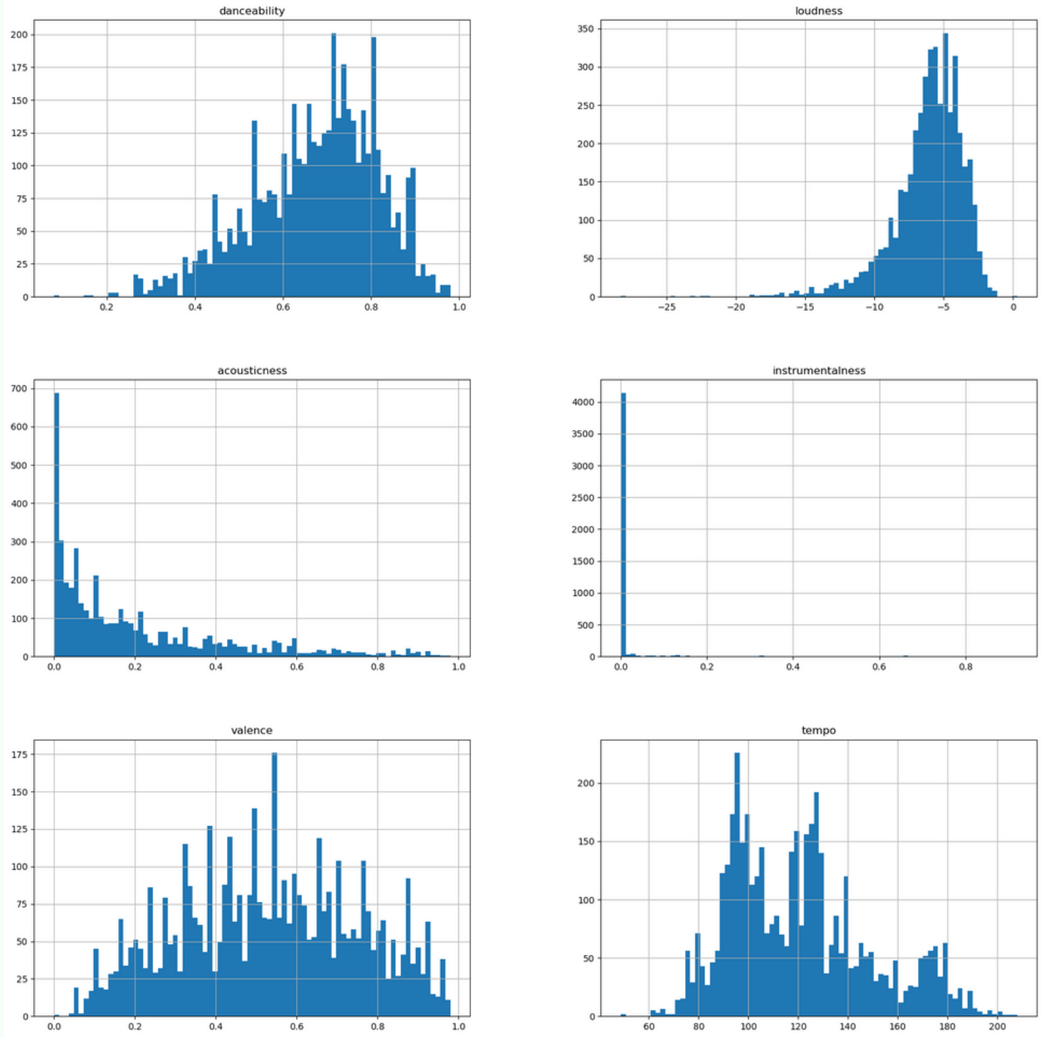
```
1 df_pearl.head()
```

	popularity	danceability	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	control	predicted_popularity
2733	53	0.07	9	-24.01	0	0.04	0.86	0.92	0.10	0.05	80.49	0	1
2745	50	0.28	4	-14.32	0	0.03	0.66	0.98	0.10	0.08	39.37	0	1
2751	45	0.51	1	-23.25	1	0.04	0.99	0.88	0.14	0.05	73.36	0	1
2754	39	0.08	11	-21.73	1	0.04	0.85	0.83	0.10	0.03	56.58	0	1
2764	38	0.11	11	-22.85	1	0.04	0.92	0.89	0.09	0.03	72.66	0	1

Son 3000 canciones “ olvidadas” aprox.



HITS POPULARES



PERLAS OLVIDADAS