ĐỒ ÁN CUỐI KÌ

I. Mô tả bài toán:

Bài toán phân lớp nhị phân: Một bài nhạc có những đặc trưng sau:

- Artist_name: tên nghệ sĩ.

- Track_name: tên bài hát.

- Genre: thể loại.
- Track id: gán ngẫu nhiên
- Popularity: mức độ phổ biến theo đánh giá người dùng
- Danceability: mô tả mức độ phù hợp của một bản nhạc để khiêu vũ dựa trên sự kết hợp của các yếu tố âm nhạc bao gồm nhịp độ, độ ổn định nhịp điệu, độ mạnh nhịp đập và sự đều đặn tổng thể. Khi giá trị đạt 0,0 bài nhạc không phù hợp để khiêu vũ, và ngược lại 1,0 thì phù hợp.
- Energy: là một thước đo từ 0.0 đến 1, 0 và biểu diễn một thước đo cường độ và hoạt động. thông thường, những bài nhạc mang đầy năng lượng sẽ nhanh, to và ồn. Ví dụ, dark metal có điểm cao, trong khi khúc dạo đầu của a bach prelude đạt điểm khá thấp. Các đặc điểm cảm nhận góp phần vào thuộc tính này bao gồm dải động, độ ồn cảm nhận được, âm sắc, tốc độ khởi phát và entropy chung.
- Key: nốt tổng thể ước tính của bản nhạc. Các số nguyên ánh xạ tới các nốt nhạc bằng cách sử dụng ký hiệu lớp nốt nhạc tiêu chuẩn. Ví dụ. 0 = C, 1 = C#/D giáng, 2 = D, và lần lượt. Nếu không có nốt nào được phát hiện, giá trị là -1.
- Loudness: âm lượng tổng thể của một bản nhạc tính bằng decibel (db). Giá trị độ ồn được tính trung bình trên toàn bộ bản nhạc và hữu ích để so sánh độ ồn tương đối của bản nhạc. Độ to là chất lượng của âm thanh đó là tương quan tâm lý chính của sức mạnh thể chất (biên độ). giá trị phạm vi điển hình giữa -60 và 0 db.
- Mode: cho biết âm giai (trưởng hay thứ) của một bản nhạc

- Speechiness: phát hiện sự hiện diện của các từ được nói trong một bản nhạc. bản ghi âm giống giọng nói độc quyền hơn (chương trình trò chuyện e.g., sách nói, thơ), càng gần với giá trị thuộc tính 1.0. giá trị trên 0.66 mô tả các bản nhạc có thể được tạo hoàn toàn bằng lời nói. giá trị từ 0,33 đến 0.66 mô tả các bản nhạc có thể chứa cả nhạc và lời nói, theo từng phần hoặc theo lớp, kể cả các trường hợp như nhạc rap. giá trị dưới 0,33 nhiều khả năng đại diện cho âm nhạc và các bản nhạc không giống lời nói khác.
- Acousticness: đo độ tin cậy từ 0.0 đến 1.0 cho dù bản nhạc có âm thanh hay không. 1.0 thể hiện độ tin cậy cao bản nhạc là âm thanh.
- Instrumentalness: dự đoán xem một bản nhạc không chứa giọng hát nào. âm thanh “ooh” và “aah” được coi là nhạc cụ trong bối cảnh này. Bản nhạc rap hoặc lời nói rõ ràng là "giọng hát". Càng gần giá trị nhạc cụ là 1,0, thì càng có nhiều khả năng bản nhạc không chứa nội dung giọng hát. Giá trị trên 0,5 được dự định để đại diện cho các bản nhạc cụ, nhưng sự tự tin sẽ cao hơn khi giá trị tiếp cận 1,0. sự phân bố các giá trị cho tính năng này trông giống như sau:
- Liveness: phát hiện sự hiện diện của khán giả trong bản ghi. giá trị độ sống cao hơn thể hiện khả năng gia tăng mà bản nhạc được biểu diễn trực tiếp. Giá trị trên 0,8 cung cấp khả năng cao rằng bản nhạc đang hoạt động.
- Valence: Phép đo từ 0.0 đến 1.0 mô tả tính tích cực của âm nhạc được truyền tải bởi một bản nhạc. các bản nhạc có giá trị cao có âm thanh tích cực hơn (ví dụ: vui vẻ, vui vẻ, hưng phấn), trong khi bản nhạc có giá trị thấp có âm thanh tiêu cực hơn (chẳng hạn như buồn, chán nản, tức giận).
- Tempo: nhịp độ ước tính tổng thể của một bản nhạc theo nhịp mỗi phút (bpm). theo thuật ngữ âm nhạc, nhịp độ là tốc độ hoặc tốc độ của một bản nhạc và bắt nguồn trực tiếp từ thời lượng nhịp trung bình.
- Duration_ms: thời lượng của bản nhạc tính bằng mili giây.

Dựa vào đặc trưng popularity trong bộ dữ liệu ta tự quy định. nếu popularity lớn hơn 60 thì thuộc nhóm nhạc trendy ngược lại nếu nhỏ hơn 60 thì là không trend.

II. Mô tả về bộ dữ liệu:

1. Cách thức xây dựng bộ dữ liệu

APIs:

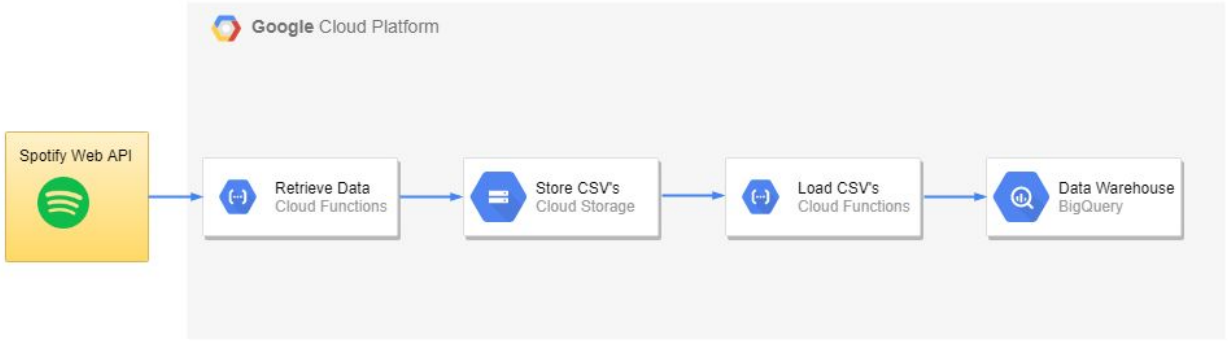
- Spotify web api

GCP Services:

- Cloud functions
- Cloud scheduler
- Cigquery

Thư viện python:

- Pandas, numpy – data analysis
- Matplotlib, seaborn – data visualization
- Spotipy – access to spotify web api



2. Số lượng, độ đa dạng:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   genre             232725 non-null  object
 1   artist_name       232725 non-null  object
```

3. Thao tác tiền xử lí dữ liệu:

- Load data

```
[ ] df=pd.read_csv('SpotifyFeatures.csv')
    df.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	Major	0.0525
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	Minor	0.0868
2	Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	Minor	0.0362
3	Movie	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tIvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	Major	0.0395
4	Movie	Fabien Nataf	Ouverture	0IusIXpMROHdEPvSl1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	Major	0.0456

- Lóm tắt cơ bản dữ liệu:

```
df.describe()
```

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	232725.000000	232725.000000	232725.000000	2.327250e+05	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000
mean	41.127502	0.368560	0.554364	2.351223e+05	0.570958	0.148301	0.215009	-9.569885	0.120765	117.666585	0.454917
std	18.189948	0.354768	0.185608	1.189359e+05	0.263456	0.302768	0.198273	5.998204	0.185518	30.898907	0.260065
min	0.000000	0.000000	0.056900	1.538700e+04	0.000020	0.000000	0.009670	-52.457000	0.022200	30.379000	0.000000
25%	29.000000	0.037600	0.435000	1.828570e+05	0.385000	0.000000	0.097400	-11.771000	0.036700	92.959000	0.237000
50%	43.000000	0.232000	0.571000	2.204270e+05	0.605000	0.000044	0.128000	-7.762000	0.050100	115.778000	0.444000
75%	55.000000	0.722000	0.692000	2.657680e+05	0.787000	0.035800	0.264000	-5.501000	0.105000	139.054000	0.660000
max	100.000000	0.996000	0.989000	5.552917e+06	0.999000	0.999000	1.000000	3.744000	0.967000	242.903000	1.000000

- Kiểm tra các đặc trưng trong bộ dữ liệu:

```
print(df.keys())
```

```
Index(['genre', 'artist_name', 'track_name', 'track_id', 'popularity',
       'acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
       'speechiness', 'tempo', 'time_signature', 'valence'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   genre             232725 non-null  object 
 1   artist_name       232725 non-null  object 
 2   track_name        232725 non-null  object 
 3   track_id          232725 non-null  object 
 4   popularity        232725 non-null  int64  
 5   acousticness      232725 non-null  float64
 6   danceability      232725 non-null  float64
 7   duration_ms       232725 non-null  int64  
 8   energy            232725 non-null  float64
 9   instrumentalness  232725 non-null  float64
 10  key               232725 non-null  object 
 11  liveness          232725 non-null  float64
 12  loudness          232725 non-null  float64
 13  mode              232725 non-null  object 
 14  speechiness       232725 non-null  float64
 15  tempo             232725 non-null  float64
 16  time_signature    232725 non-null  object 
 17  valence           232725 non-null  float64
dtypes: float64(9), int64(2), object(7)
memory usage: 32.0+ MB
```

dtypes: float64(9), int64(2), object(7)

- Đếm số lượng phân loại biến categorical với numerical:

```
categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :\n\n', categorical)
```

4. Phân chia (split):

Chia train-test thành 80/20

```
[ ]  training = df.sample(frac = 0.8,random_state = 420)
     X_train = training[features]
     y_train = training['popularity']
     X_test = df.drop(training.index)[features]
```

```
[ ]  X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size = 0.2, random_state = 420)
```

III. Thao tác xử lí dữ liệu:

- Kiểm tra giá trị thiếu NaN bằng hàm isnull():

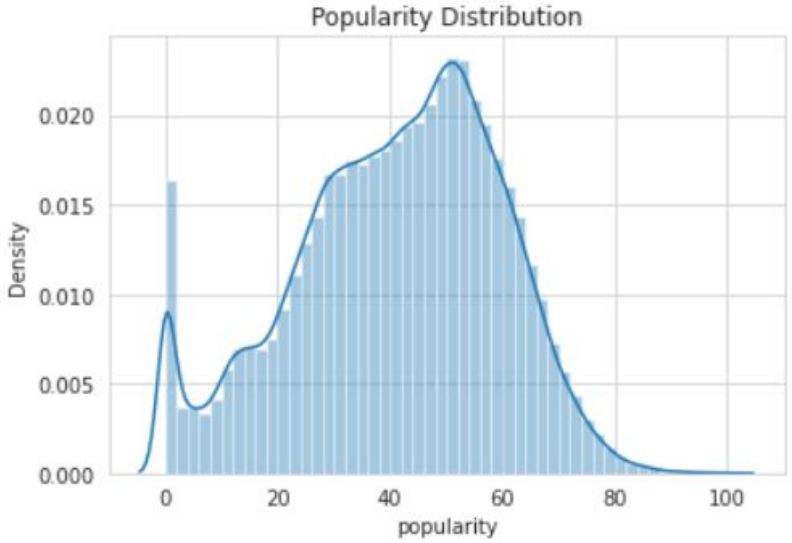
```
df.isnull().sum()
```

```
genre               0
artist_name         0
track_name          0
track_id            0
popularity          0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
time_signature      0
valence             0
dtype: int64
```

- Sử dụng thư viện seaborn để dễ dàng hình dung mật độ phân bố các đặc trưng:

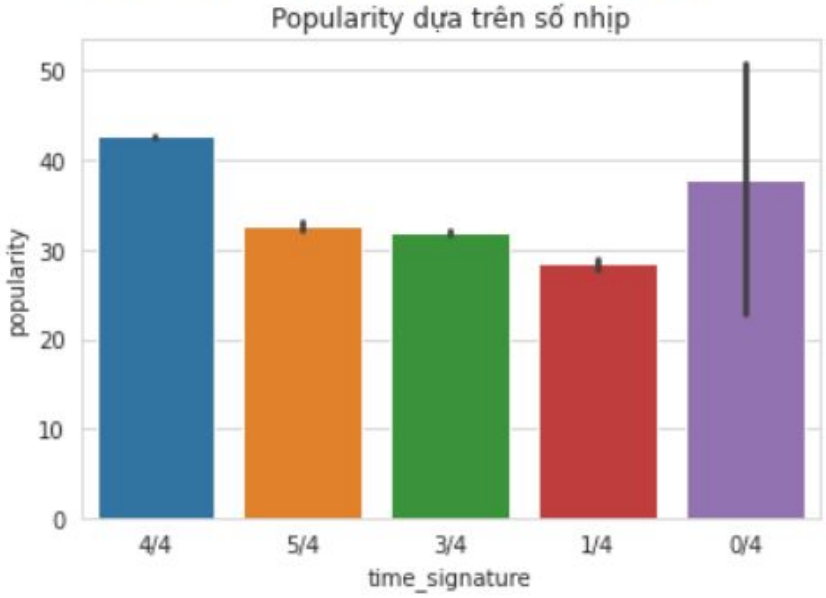
```
[ ]  sns.distplot(df['popularity']).set_title('Popularity Distribution')
```

```
Text(0.5, 1.0, 'Popularity Distribution')
```



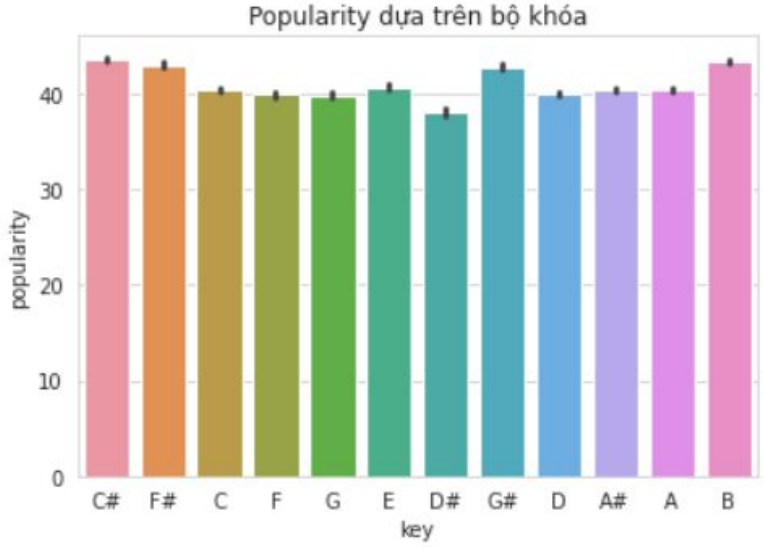

```
sns.barplot(x='time_signature', y='popularity', data=df)
plt.title('Popularity dựa trên số nhịp')
```

```
Text(0.5, 1.0, 'Popularity dựa trên số nhịp')
```



```
sns.barplot(x='key',y='popularity', data=df)
plt.title('Popularity dựa trên bộ khóa')
```

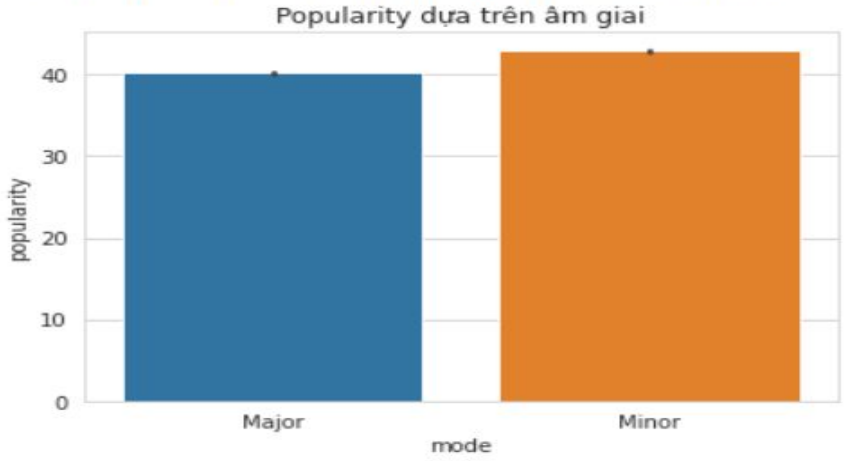
```
Text(0.5, 1.0, 'Popularity dựa trên bộ khóa')
```



Key là key master là bộ khóa định vị nốt nhạc bắt đầu thường là G key hoặc F key

```
sns.barplot(x='mode', y='popularity', data=df)
plt.title('Popularity dựa trên âm giai')
```

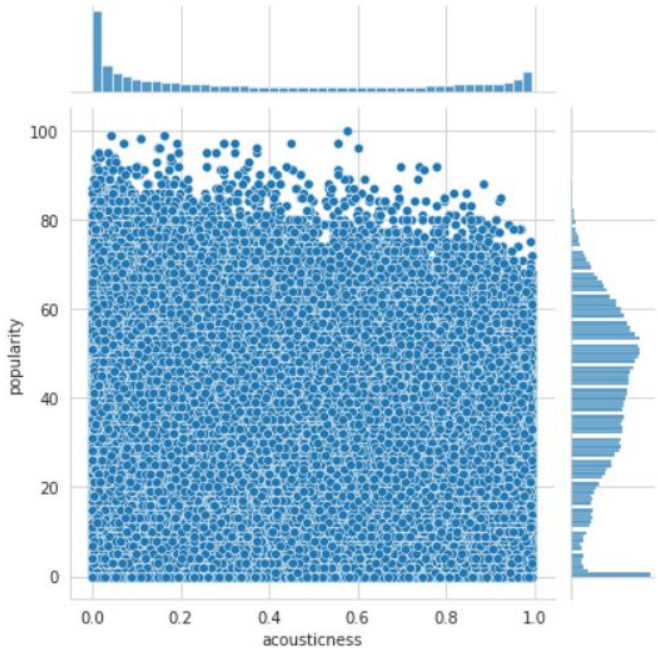
```
Text(0.5, 1.0, 'Popularity dựa trên âm giai')
```



Major là trưởng Minor là thứ

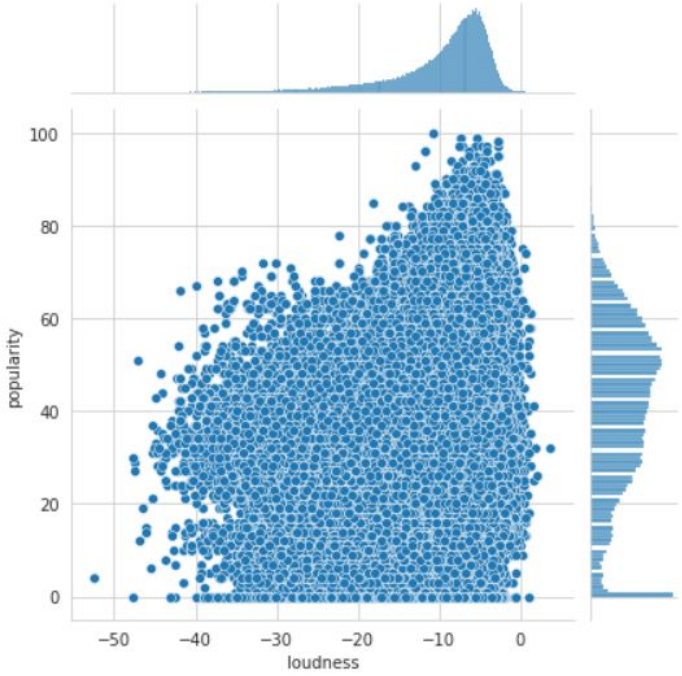
```
sns.jointplot(x='acousticness', y='popularity', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7fd9101ea828>
```

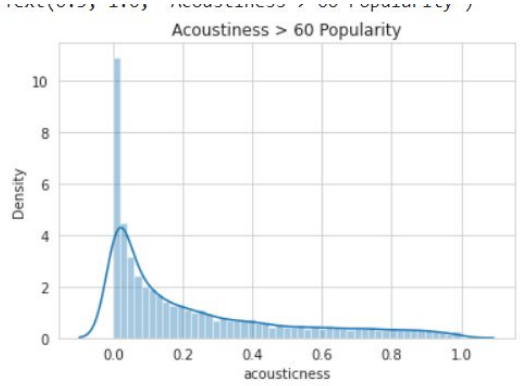


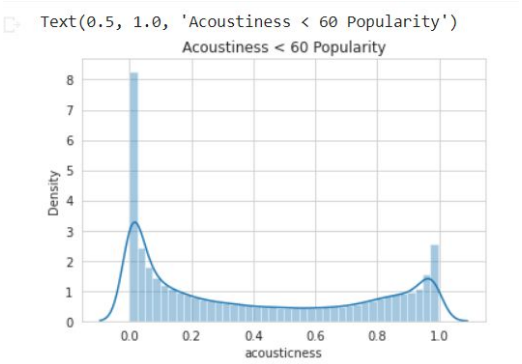

```
sns.jointplot(x='loudness', y='popularity', data=df)
```

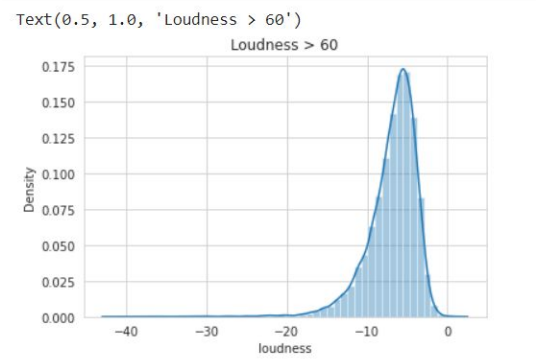
```
<seaborn.axisgrid.JointGrid at 0x7fd90fe1ad30>
```

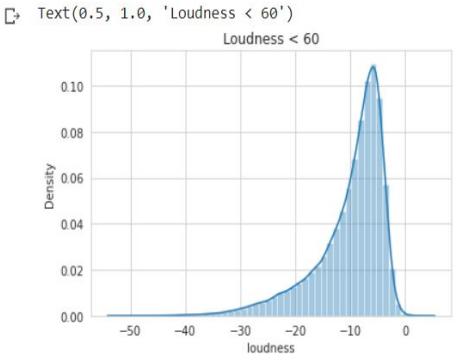


- Dựa vào mật độ phân bổ giá trị của feature popularity trong data, tự phân định giá trị 60 sẽ là giá trị phân lớp









Bây giờ phải chuyển định dạng của 3 biến Categorical (Key, Mode, Time-signature) sang kiểu số

```
[203] L_keys=df['key'].unique()
      for i in range(len(L_keys)):
        df.loc[df['key']==L_keys[i],'key']=i
```

Nhạc lí có 12 nốt tính luôn thăng giáng, set nốt La(A) là 0 lần lượt đến nốt G# là 11

```
[204] df.sample(10)
```

ıck_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo
GtsZH	49	0.074500	0.765	234507	0.486	0.000010	4	0.282	-13.819	Major	0.0469	87.054
2Ncoth	48	0.000299	0.627	194293	0.776	0.000050	4	0.498	-3.520	Major	0.0253	130.003
ıGiEVJ	77	0.163000	0.755	181102	0.769	0.004020	9	0.360	-5.569	Major	0.0399	114.011
/vCq0	47	0.000023	0.507	213087	0.990	0.000000	2	0.347	-3.517	Minor	0.1210	135.053
SIhRcr	35	0.005180	0.515	194875	0.806	0.000660	9	0.130	-7.703	Minor	0.0404	135.098
ɛejNS4	52	0.293000	0.595	235123	0.428	0.000126	7	0.109	-11.991	Major	0.0463	132.985
Nwgaz	52	0.311000	0.886	112500	0.742	0.000011	0	0.241	-4.582	Major	0.1990	130.000
qnt9u0	22	0.083400	0.887	256013	0.692	0.000000	2	0.174	-5.313	Major	0.0391	96.991
/VRhD	68	0.055400	0.762	278720	0.641	0.000000	0	0.454	-6.784	Major	0.2090	90.089
?OuUY	18	0.702000	0.342	145787	0.989	0.000000	8	0.191	-3.397	Major	0.4260	172.118

```
[205] df.loc[df['mode']=='Major','mode'] = 1
      df.loc[df['mode'] =='Minor','mode'] = 0
```

Mode : trưởng(major) set =1 còn thứ (minor) set =0

```
[206] df.sample(10)
```

ty	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature
53	0.08890	0.900	236227	0.532	0.000000	2	0.0533	-5.516	1	0.0744	140.023	4/4
31	0.01390	0.850	170509	0.500	0.000384	7	0.0947	-6.708	1	0.0665	160.020	4/4
7	0.74300	0.517	93307	0.704	0.000000	8	0.0635	-5.918	0	0.3020	82.970	5/4
48	0.03390	0.683	244507	0.666	0.000026	9	0.0938	-4.541	0	0.1870	97.043	4/4
59	0.55400	0.641	166533	0.294	0.000035	2	0.1100	-13.337	1	0.0366	132.709	4/4
37	0.83400	0.308	385587	0.221	0.000005	8	0.2470	-17.725	0	0.0676	87.163	4/4

```
[207] L_ts=df['time_signature'].unique()
      for i in range(len(L_ts)):
        df.loc[df['time_signature']==L_ts[i],'time_signature'] = i
```

```
[208] df.sample(5)
```

ty	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
49	0.9900	0.384	288627	0.0452	0.930000	1	0.6770	-24.145	0	0.0605	175.088	2	0.0642
46	0.8030	0.390	155333	0.4600	0.000000	10	0.1020	-8.711	0	0.0455	147.335	0	0.6940
21	0.9450	0.796	75933	0.0866	0.000000	8	0.0778	-14.727	1	0.0374	99.910	0	0.6540
57	0.4230	0.660	169520	0.6110	0.000011	2	0.1070	-10.309	1	0.0392	113.898	0	0.7490
33	0.0693	0.805	201627	0.6650	0.000085	9	0.0773	-6.553	0	0.0538	94.974	0	0.7910

```
[209] df.loc[df['popularity']<60,'popularity'] = 0
      df.loc[df['popularity']>=60,'popularity'] = 1
      df.loc[df['popularity']==1]
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalnes
135	R&B	Mary J. Blige	Be Without You - Kendu Mix	2YegxR5As7BeQuVp2U6pek	1	0.08300	0.724	246333	0.689	0.00000
136	R&B	Rihanna	Desperado	6KFaHC9G178beAp7P0Vi5S	1	0.32300	0.685	186467	0.610	0.00000
137	R&B	Yung Bleu	Ice On My Baby (feat. Kevin Gates) - Remix	6muW8cSjJ3rusKJ0vH5olw	1	0.06750	0.762	199520	0.520	0.00000
138	R&B	Surfaces	Heaven Falls / Fall on Me	7yHqOZfsXYlicyoMt62yC6	1	0.36000	0.563	240597	0.366	0.00243
139	R&B	Olivia O'Brien	Love Myself	4XzgjxGKqULifVf7mnDIQK	1	0.59600	0.653	213947	0.621	0.00000
...	...	...	...	...	...	...	...	...	...	
228206	Soul	Natalie Cole	Bachata Rosa	0qLqqJYI7wW8w93fgqd2LN	1	0.77000	0.672	242200	0.424	0.00000
228385	Soul	Paolo Nutini	Jenny Don't Be Hasty	3G3GZnfjmypPEdkItAofp3	1	0.00362	0.764	207133	0.816	0.00000
			For You - Recorded At							

IV. Mô tả thuật toán

1. Giới thiệu:

- Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh. random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. random forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh. nó nằm ở cơ sở của thuật toán boruta, chọn các tính năng quan trọng trong tập dữ liệu.

2. Thuật toán Random Forests

- Giả sử bạn muốn đi trên một chuyến đi và bạn muốn đi đến một nơi mà bạn sẽ thích.

- Vậy bạn sẽ làm gì để tìm một nơi mà bạn sẽ thích? Bạn có thể tìm kiếm trực tuyến, đọc các bài đánh giá trên blog và các cổng thông tin du lịch hoặc bạn cũng có thể hỏi bạn bè của mình.

- Giả sử bạn đã quyết định hỏi bạn bè và nói chuyện với họ về trải nghiệm du lịch trong quá khứ của họ đến những nơi khác nhau. Bạn sẽ nhận được một số khuyến nghị từ tất cả các bạn. Bây giờ bạn phải tạo danh sách các địa điểm được đề xuất. sau đó, bạn yêu cầu họ bỏ phiếu (hoặc chọn địa điểm tốt nhất cho chuyến đi) từ danh sách các địa điểm được đề xuất bạn đã thực hiện. Địa điểm có số phiếu bầu cao nhất sẽ là lựa chọn cuối cùng của bạn cho chuyến đi.

- Trong quá trình quyết định ở trên, có hai phần. Trước tiên, hãy hỏi bạn bè về trải nghiệm du lịch cá nhân của họ và nhận được đề xuất từ nhiều nơi họ đã ghé thăm. điều này cũng giống như sử dụng thuật toán cây quyết định. Ởđây, mỗi người trong số các bạn chọn những nơi mà họ đã ghé thăm cho đến nay. Phần thứ hai, sau khi thu thập tất cả các khuyến nghị, là thủ tục bỏ phiếu để chọn địa điểm tốt nhất trong danh sách các khuyến nghị. Toàn bộ quá trình nhận được khuyến nghị từ bạn bè và bỏ phiếu cho họ để tìm ra nơi tốt nhất được gọi là thuật toán rừng ngẫu nhiên.

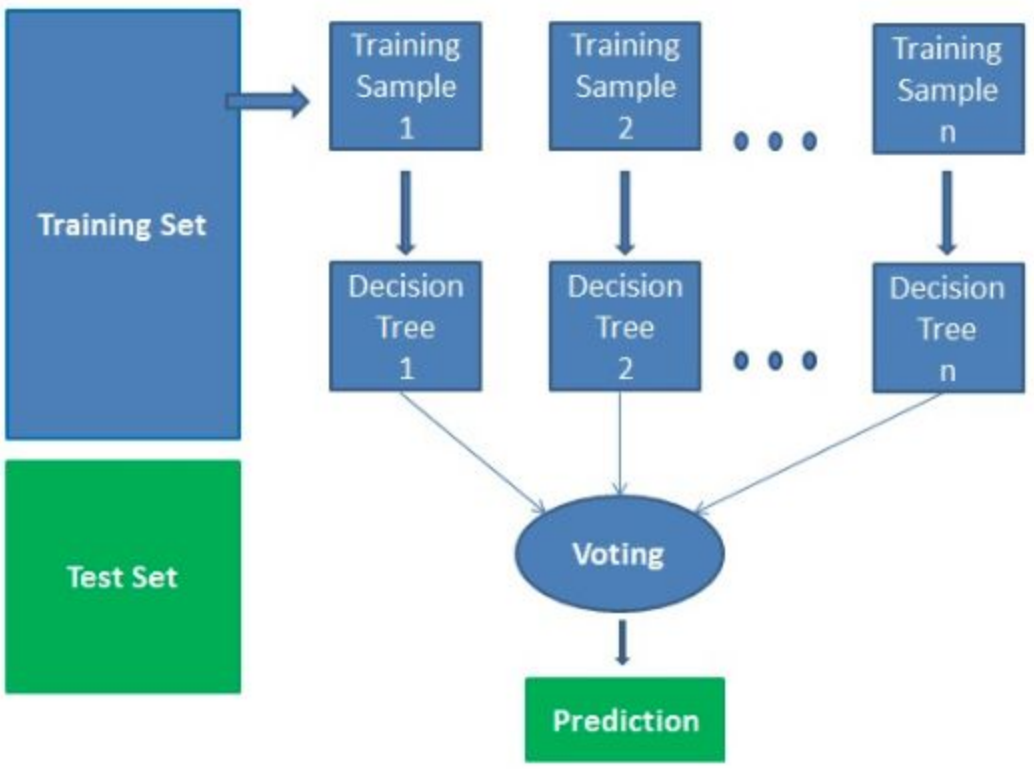
- Về mặt kỹ thuật, nó là một phương pháp tổng hợp (dựa trên cách tiếp cận phân chia và chinh phục) của các cây quyết định được tạo ra trên một tập - dữ liệu được chia ngẫu nhiên. Bộ sưu tập phân loại cây quyết định này còn được gọi là rừng. cây quyết định riêng lẻ được tạo ra bằng cách sử dụng chỉ báo chọn thuộc tính như tăng thông tin, tỷ lệ tăng và chỉ số gini cho từng thuộc tính. mỗi cây phụ thuộc vào một mẫu ngẫu nhiên độc lập. Trong bài toán phân loại, mỗi phiếu bầu chọn và lớp phổ

biến nhất được chọn là kết quả cuối cùng. trong trường hợp hồi quy, mức trung bình của tất cả các kết quả đầu ra của cây được coi là kết quả cuối cùng. nó đơn giản và mạnh mẽ hơn so với các thuật toán phân loại phi tuyến tính khác.

3. Thuật toán hoạt động như thế nào?

Nó hoạt động theo bốn bước:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Hãy bỏ phiếu cho mỗi kết quả dự đoán.
4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



***Ưu điểm**: Random Forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy. Random Forests cũng có thể xử lý các giá trị còn thiếu. Có hai cách để xử lý các giá trị này: sử dụng các giá trị trung bình để thay thế các biến liên tục và tính toán mức trung bình gần kề của các giá trị bị thiếu. Bạn có thể nhận được tầm quan trọng của tính năng tương đối, giúp chọn các tính năng đóng góp nhiều nhất cho trình phân loại.

***Nhược điểm**: Random Forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian. Mô hình khó hiểu hơn so với cây quyết định, nơi bạn có thể dễ dàng đưa ra quyết định bằng cách đi theo đường dẫn trong cây.

Các tính năng quan trọng:

- Random Forests cũng cung cấp một chỉ số lựa chọn tính năng tốt. Scikit-learn cung cấp thêm một biến với mô hình, cho thấy tầm quan trọng hoặc đóng góp tương đối của từng tính năng trong dự đoán. nó tự động tính toán điểm liên quan của từng tính năng trong giai đoạn đào tạo. Sau đó, nó cân đối mức độ liên quan xuống sao cho tổng của tất cả các điểm là 1.

- Điểm số này sẽ giúp bạn chọn các tính năng quan trọng nhất và thả các tính năng quan trọng nhất để xây dựng mô hình.

- Random Forests sử dụng tầm quan trọng của gini hoặc giảm tạp chất trung bình (mdi) để tính toán tầm quan trọng của từng tính năng. Gini tầm quan trọng còn được gọi là tổng giảm trong tạp chất nút. đây là mức độ phù hợp hoặc độ chính xác của mô hình giảm khi bạn thả biến. Độ lớn càng lớn thì biến số càng có ý nghĩa. ở đây, giảm trung bình là một tham số quan trọng cho việc lựa chọn biến. Chỉ số gini có thể mô tả sức mạnh giải thích tổng thể của các biến. Random Forests và cây quyết định random forests là một tập hợp của nhiều cây quyết định. Cây quyết định sâu có thể bị ảnh hưởng quá mức, nhưng random forests ngăn cản việc lấp đầy bằng cách tạo cây trên các tập con ngẫu nhiên. Cây quyết định nhanh hơn tính toán. random forests khó giải thích, trong khi cây quyết định có thể diễn giải dễ dàng và có thể chuyển đổi thành quy tắc.

V. Cài đặt tinh chỉnh tham số

1. Thư viện sklearn:

- Scikit-learn (sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

- Để cài đặt scikit-learn trước tiên phải cài thư viện scipy (scientific python). những thành phần gồm:

- **Numpy**: gói thư viện xử lý dãy số và ma trận nhiều chiều
- **Scipy**: gói các hàm tính toán logic khoa học
- **Matplotlib**: biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
- **Ipython**: notebook dùng để tương tác trực quan với python
- **Sympy**: gói thư viện các kí tự toán học
- **Pandas**: xử lý, phân tích dữ liệu dưới dạng bảng

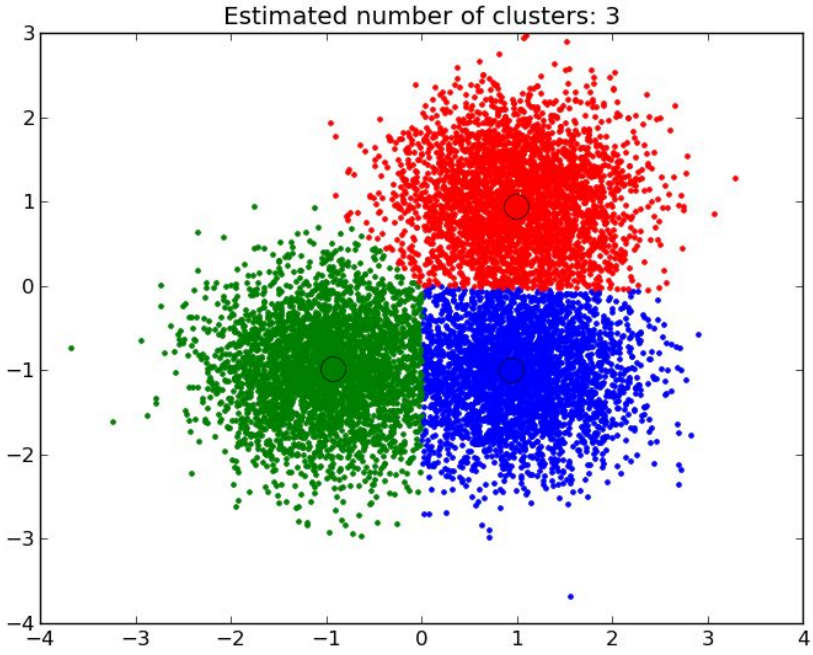
- Những thư viện mở rộng của scipy thường được đặt tên dạng scikits. như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

- Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

- Mặc dù được viết cho python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của c để tăng hiệu suất làm việc. Ví dụ như: numpy (tính toán ma trận), lapack, libsvm và cython.

Nhóm thuật toán

Thư viện tập trung vào việc mô hình hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện numpy và pandas



Hình demo của thuật toán phân cụm

sau đâu là một số nhóm thuật toán được xây dựng bởi thư viện scikit-learn:

- **Clustering**: nhóm thuật toán phân cụm dữ liệu không gán nhãn. ví dụ thuật toán kmeans
- **Cross validation**: kiểm chéo, đánh giá độ hiệu quả của thuật toán học giám sát sử dụng dữ liệu kiểm thử (validation data) trong quá trình huấn luyện mô hình.
- **Datasets**: gồm nhóm các bộ dữ liệu được tích hợp sẵn trong thư viện. hầu như các bộ dữ liệu đều đã được chuẩn hóa và mang lại hiệu suất cao trong quá trình huấn luyện như iris, digit, ...
- **Dimensionality reduction**: mục đích của thuật toán này là để giảm số lượng thuộc tính quan trọng của dữ liệu bằng các phương pháp như tổng hợp, biểu diễn dữ liệu và lựa chọn đặc trưng. ví dụ thuật toán pca (principal component analysis).
- **Ensemble methods**: các phương pháp tập hợp sử dụng nhiều thuật toán học tập để có được hiệu suất dự đoán tốt hơn so với bất kỳ thuật toán học cấu thành nào.
- **Feature extraction**: trích xuất đặc trưng. mục đích là để định nghĩa các thuộc tình với dữ liệu hình ảnh và dữ liệu ngôn ngữ.
- **Feature selection**: trích chọn đặc trưng. lựa chọn các đặc trưng có ý nghĩa trong việc huấn luyện mô hình học giám sát.
- **Parameter tuning**: tinh chỉnh tham số. các thuật toán phục vụ việc lựa chọn tham số phù hợp để tối ưu hóa mô hình.
- **Manifold learning**: các thuật toán học tổng hợp và phân tích dữ liệu đa chiều phức tạp.

- **Supervised models**: học giám sát. mảng lớn các thuật toán học máy hiện nay. ví dụ như linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines và decision trees.

2. Tinh chỉnh tham số

Giữ lại các Features thuộc numberical như sau

```
[211] features = ["acousticness", "danceability", "duration_ms", "energy", "instrumentalness", "key", "liveness", "mode", "speechiness", "tempo", "tim
```

Chia train-test thành 80/20

```
[212] training = df.sample(frac = 0.8,random_state = 420)
      X_train = training[features]
      y_train = training['popularity']
      X_test = df.drop(training.index)[features]
```

```
[213] X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size = 0.2, random_state = 420)
```

```
[214] RFC_Model = RandomForestClassifier()
      RFC_Model.fit(X_train, y_train)
      RFC_Predict = RFC_Model.predict(X_valid)
      RFC_Accuracy = accuracy_score(y_valid, RFC_Predict)
```

VI. Đánh giá:

	Model	Accuracy
1	RandomForestClassifier	0.945134
3	DecisionTreeClassifier	0.885971
0	LogisticRegression	0.842706
2	KNeighborsClassifier	0.826431
4	LinearSVC	0.760000

	Model	AUC
1	RandomForestClassifier	0.840730
3	DecisionTreeClassifier	0.824929
2	KNeighborsClassifier	0.607270
4	LinearSVC	0.563868
0	LogisticRegression	0.500000

- Thuật toán randomforestclassifier (rdc) dẫn đầu thể hiện độ phù hợp ở bài toán này với 94.51% accuracy (*) và 84.07% auc (**). tiếp theo đó là thuật toán decision tree với 88.59% accuracy và 82.49% auc.

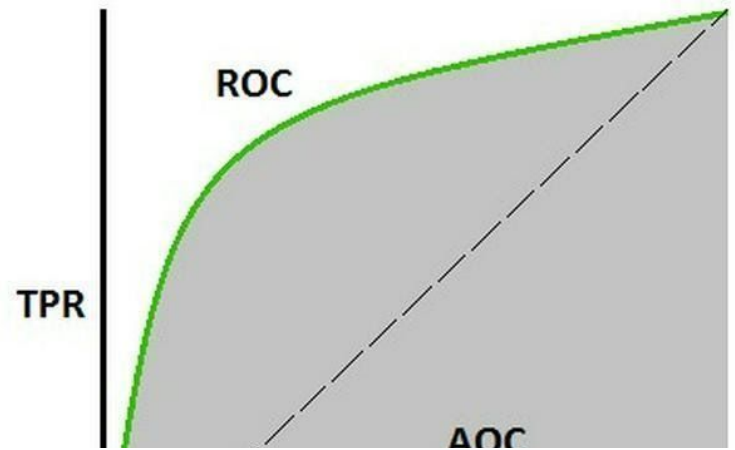
***Accuracy (độ chính xác)**: độ gần của các phép đo với một giá trị cụ thể, trong khi độ chính xác phép đo (precision) là mức độ gần của các phép đo với nhau.

độ chính xác (accuracy) có hai định nghĩa:

1. Thông thường hơn, đó là một mô tả về các *lỗi hệ thống*, thước đo sai lệch thống kê; độ chính xác thấp gây ra sự khác biệt giữa kết quả và giá trị "đúng". iso gọi đây là trueness.
2. Ngoài ra, iso định nghĩa độ chính xác là mô tả sự kết hợp của cả hai loại lỗi quan sát ở trên (ngẫu nhiên và có hệ thống), vì vậy độ chính xác cao đòi hỏi cả độ chính xác cao và độ chính xác cao.

****Auc:** auc - roc là một phương pháp tính toán hiệu suất của một mô hình phân loại theo các ngưỡng phân loại khác nhau. Giả sử với bài toán phân loại nhị phân (2 lớp) sử dụng hồi quy logistic (logistic regression), việc chọn các ngưỡng phân loại [0..1] khác nhau sẽ ảnh hưởng đến khả năng phân loại của mô hình và ta cần tính toán được mức độ ảnh hưởng của các ngưỡng. auc là từ viết tắt của area under the curve còn roc viết tắt của receiver operating characteristics. roc là một đường cong biểu diễn xác suất và auc biểu diễn mức độ phân loại của mô hình. Auc-roc còn được biết đến dưới cái tên auroc (area under the receiver operating characteristics). Ý nghĩa của auroc có thể diễn giải như sau: là xác suất rằng một mẫu dương tính được lấy ngẫu nhiên sẽ được xếp hạng cao hơn một mẫu âm tính được lấy ngẫu nhiên. Biểu diễn theo công thức, ta có auc = p(score(x+) > score(x-)). chỉ số auc càng cao thì mô hình càng chính xác trong việc phân loại các lớp.

- Đường cong roc biểu diễn các cặp chỉ số (tpr, fpr) tại mỗi ngưỡng với tpr là trục tục và fpr là trục hoành.



Các chỉ số sử dụng trong AUC - ROC:

- TPR (true positive rate/sentivity/recall): biểu diễn tỷ lệ phân loại chính xác các mẫu dương tính trên tất cả các mẫu dương tính, được tính theo công thức:

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

- TPR càng cao thì các mẫu dương tính càng được phân loại chính xác.

specificity: biểu diễn tỷ lệ phân loại chính xác các mẫu âm tính trên tất cả các mâu âm tính, được tính theo công thức:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

- FPR (false positive rate/fall-out): biểu diễn tỷ lệ gắn nhãn sai các mẫu âm tính thành dương tính trên tất cả các mẫu âm tính, được tính theo công thức:

$$FPR = 1 - Specificity$$

$$= \frac{FP}{TN + FP}$$

- Có thể thấy specificity tỷ lệ nghịch với FPR. FPR càng cao thì specificity càng giảm và số lượng các mẫu âm tính bị gắn nhãn sai càng lớn.

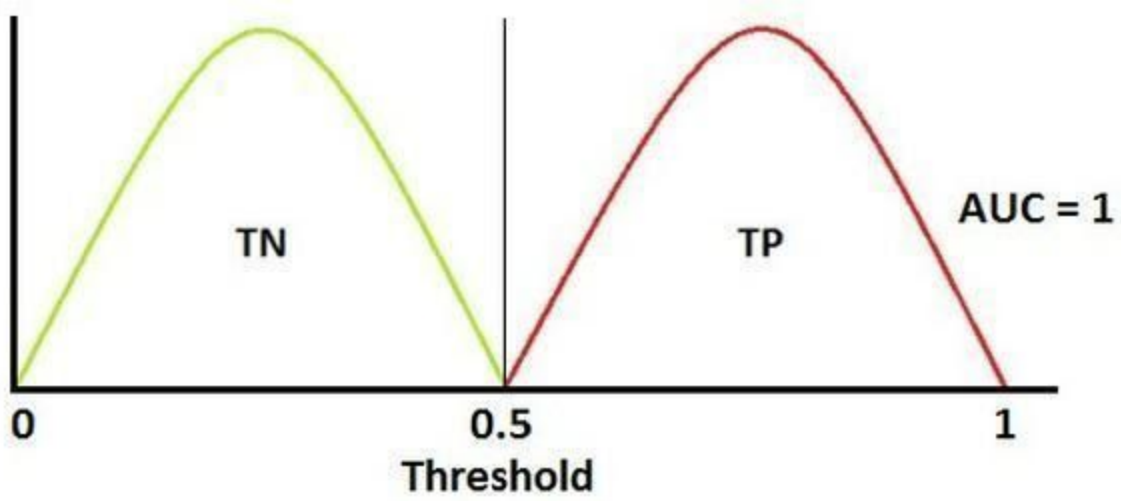
- Đây chính là các chỉ số dùng để tính toán hiệu suất phân loại của mô hình. Để hợp chúng lại thành 1 chỉ số duy nhất, ta sử dụng đường cong roc để hiển thị từng cặp (TPR, FPR) cho các ngưỡng khác nhau với mỗi điểm trên đường cong biểu diễn 1 cặp (TPR, FPR) cho 1 ngưỡng, sau đó tính chỉ số auc cho đường cong này. Chỉ số auc chính là con số thể hiện hiệu suất phân loại của mô hình.

- Đánh giá mô hình qua chỉ số AUC:

Như đã nói ở trên, chỉ số auc càng gần 1 thì mô hình càng phân loại chính xác. auc càng gần 0.5 thì hiệu suất phân loại càng tệ còn nếu gần 0 thì mô hình sẽ phân loại ngược kết quả (phân loại dương tính thành âm tính và ngược lại). Giờ ta sẽ biểu diễn các trường hợp này qua các đồ thị.

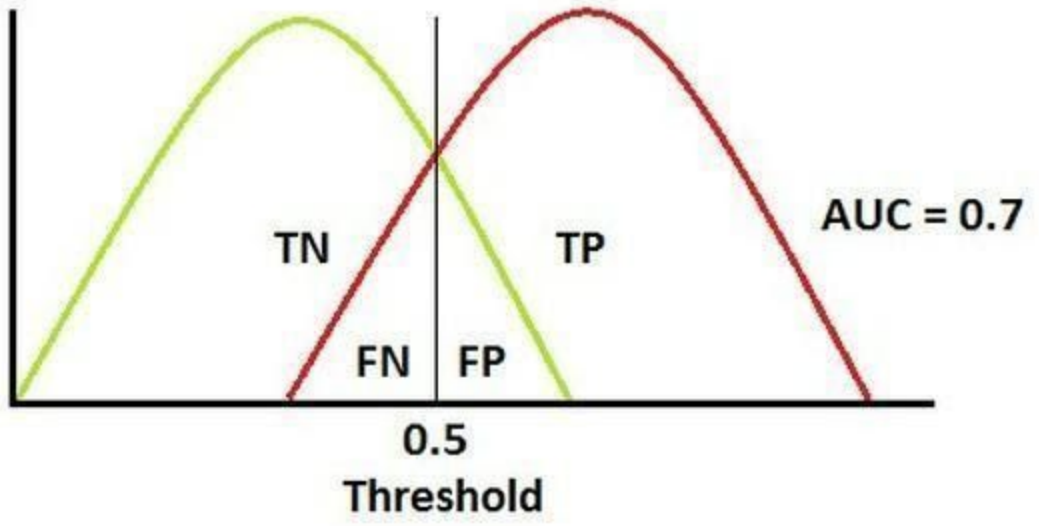
Ghi chú: đường cong màu đỏ biểu diễn phân phối của các mẫu dương tính, đường cong màu xanh lá biểu diễn phân phối của các mẫu âm tính:

- AUC = 1



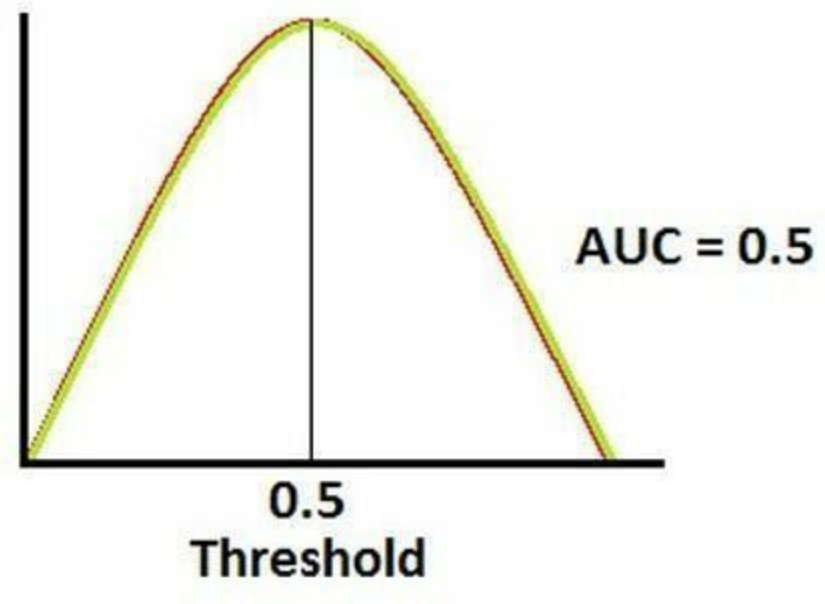
Đây là trường hợp tốt nhất. mô hình phân loại hoàn toàn chính xác khi 2 đường cong không chồng lên nhau. tuy nhiên trường hợp này rất khó xảy ra và chỉ tồn tại trên lý thuyết.

- Auc = 0.7



Khi 2 đường cong chồng lên nhau, việc phân loại sẽ xảy ra 2 dạng lỗi đó là FP (type 1 error) và FN (type 2 error). ta có thể thay đổi giá trị của 2 chỉ số lỗi này bằng cách thay đổi ngưỡng. có thể thấy đường cong roc đã hạ xuống một chút, tuy nhiên nó vẫn nằm ở góc trên bên trái của đồ thị, tức là hiệu suất phân loại vẫn ổn định.

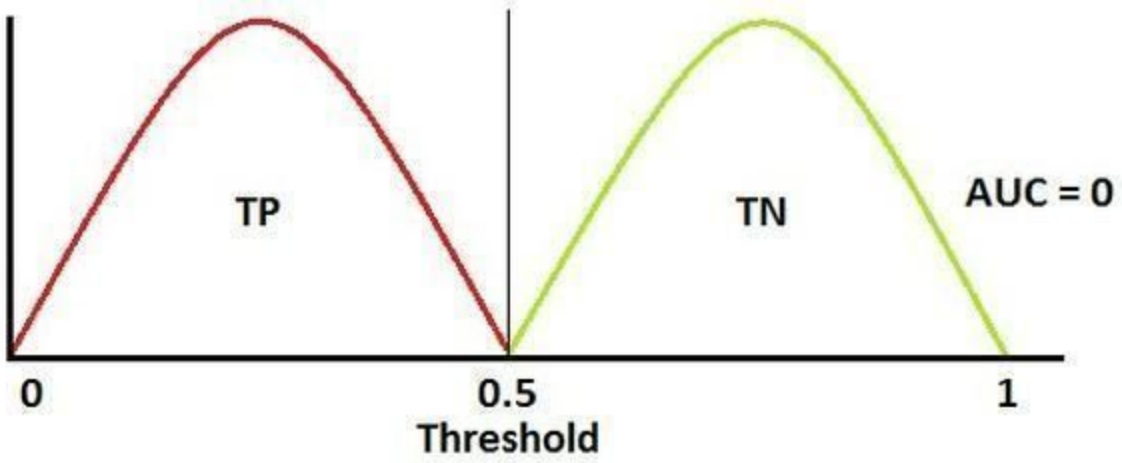
- auc = 0.5



Đây là trường hợp tệ nhất. mô hình hoàn toàn không có khả năng phân loại giữa 2 lớp.

đường cong roc ở trường hợp này là một đường thẳng tạo với ox một góc 45 độ, biểu diễn một mô hình phân loại một cách ngẫu nhiên. mô hình phân loại ngẫu nhiên thường được sử dụng như một đường cơ sở để so sánh giữa các mô hình.

- Auc = 0



Khi auc xập xỉ 0, mô hình phân loại ngược hoàn toàn 2 lớp với việc phân loại âm tính thành dương tính - dương tính thành âm tính. để sửa điều này ta chỉ cần đảo ngược đầu ra của mô hình.

Mối liên hệ giữa specificity - sentivity, TPR - FPR:

- Sentivity và specificity là 2 chỉ số tỷ lệ nghịch với nhau. khi chỉ số sentivity tăng thì chỉ số specificity giảm và ngược lại.
- Khi ta tăng ngưỡng phân loại, số lượng mẫu được gắn nhãn âm tính sẽ tăng lên, từ đó chỉ số specificity tăng và chỉ số sentivity giảm. điều ngược lại cũng đúng.
- Vì sentivity/TFP và FPR đều tỉ lệ nghịch với specificity nên TFP tỷ lệ thuận với FPR.