

نام خدا

تمرین سوم درس هوش مصنوعی

سوال دوم

محمد امین سلطانی چم حیدری

۸۱۰۶۰۱۰۸۱

استاد مربوطه:

دکتر مسعود شریعت پناهی

بهار ۱۴۰۲

بخش دوم: پیش‌بینی گونه (genre) و میزان محبوبیت آهنگ‌های جدید

یک سایت موسیقی برای افزایش مخاطبان خود قصد دارد آهنگ‌های منتشر شده بین سال‌های ۲۰۰۰ تا ۲۰۲۰ را از نظر گونه (genre) و میزان محبوبیت رتبه‌بندی کند. برای این کار از ویژگی‌هایی همچون نام آهنگ، نام خواننده، سال انتشار، انرژی، ریتم و ضربان‌گ استفاده می‌شود. داده‌های مورد نظر در فایل musics.csv ارایه شده است. این فایل شامل ۱۸ ستون است که دو ستون محبوبیت (popularity) و گونه (genre) خروجی‌های مدل و ۱۶ ستون دیگر ورودی‌ها (ویژگی‌ها) را نشان می‌دهند.

required libraries

```
In [10]: ┏ import matplotlib.pyplot as plt  
      import pandas as pd  
      import numpy as np  
      import pylab as pl  
      from sklearn import preprocessing
```

به منظور تربیت مدل برای این سوال ابتدا کتابخانه‌های مورد نظر در Jupyter notebook فراخوانی شده اند.

← نمایش data و رسم نمودار matplotlib

← کار با داده‌های ساختار یافته مانند جداول، فایل csv و excel و ... pandas

← انجام عملیات ریاضی، آماری و عددی Numpy

← تجسم داده‌های علمی و عملی. ترکیبی از numpy و matplotlib می‌باشد Pylab

← یادگیری ماشین و تحلیل داده Sickit_learn

open file

```
In [18]: musics_df0 = pd.read_csv("musics.csv")
musics_df0.head()
```

Out[18]:

	artist	song	duration_ms	explicit	year	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valer
0	Britney Spears	Oops...I Did It Again	211160	False	2000	0.751	0.834	1	-5.444	0	0.0437	0.3000	0.000018	0.3550	0.8
1	blink-182	All The Small Things	167066	False	1999	0.434	0.897	0	-4.918	1	0.0488	0.0103	0.000000	0.6120	0.6
2	Faith Hill	Breathe	250546	False	1999	0.529	0.496	7	-9.007	1	0.0290	0.1730	0.000000	0.2510	0.2
3	Bon Jovi	It's My Life	224493	False	2000	0.551	0.913	0	-4.063	0	0.0466	0.0263	0.000013	0.3470	0.5
4	*NSYNC	Bye Bye Bye	200560	False	2000	0.614	0.928	8	-4.806	0	0.0516	0.0408	0.001040	0.0845	0.6

در مرحله بعد داده های مربوط به آهنگ های منتشر شده که شامل ویژگی های ورودی و خروجی ها می باشد به کمک دستور `pd` از کتابخانه `pandas` باز شده اند.

```
clear data
```

```
In [22]: musics_df = musics_df0
print(musics_df.info())
musics_df.describe()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   artist            2000 non-null    object  
 1   song              2000 non-null    object  
 2   duration_ms       2000 non-null    int64  
 3   explicit          2000 non-null    bool    
 4   year              2000 non-null    int64  
 5   danceability      2000 non-null    float64 
 6   energy             2000 non-null    float64 
 7   key               2000 non-null    int64  
 8   loudness          2000 non-null    float64 
 9   mode              2000 non-null    int64  
 10  speechiness       2000 non-null    float64 
 11  acousticness      2000 non-null    float64 
 12  instrumentalness 2000 non-null    float64 
 13  liveness          2000 non-null    float64 
 14  valence            2000 non-null    float64 
 15  tempo              2000 non-null    float64 
 16  genre              2000 non-null    object  
 17  popularity         2000 non-null    int64  
 18  category           2000 non-null    category
dtypes: bool(1), category(1), float64(9), int64(5), object(3)
memory usage: 269.9+ KB
None
```

```
Out[22]:
```

	duration_ms	year	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liver
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	228748.124500	2009.494000	0.667438	0.720366	5.378000	-5.512434	0.553500	0.103568	0.128955	0.015226	0.18
std	39136.569008	5.85996	0.140416	0.152745	3.615059	1.933482	0.497254	0.096159	0.173346	0.087771	0.14
min	113000.000000	1998.00000	0.129000	0.054900	0.000000	-20.514000	0.000000	0.023200	0.000019	0.000000	0.02
25%	203580.000000	2004.00000	0.581000	0.622000	2.000000	-6.490250	0.000000	0.039600	0.014000	0.000000	0.08
50%	223279.500000	2010.00000	0.676000	0.736000	6.000000	-5.285000	1.000000	0.059850	0.055700	0.000000	0.12
75%	248133.000000	2015.00000	0.764000	0.839000	8.000000	-4.167750	1.000000	0.129000	0.176250	0.000068	0.24
max	484146.000000	2020.00000	0.975000	0.999000	11.000000	-0.276000	1.000000	0.576000	0.976000	0.985000	0.85

در قسمت بعد توصیفی از داده های ورودی مشاهده می شود. این مرحله به ان علت انجام شده که در صورت وجود هرگونه نقص و عیب در داده های ورودی سطر و ستون مربوط به ان داده ورودی حذف شود که البته در این سوال مشاهده می شود هیچ گونه نقصی در داده های ورودی وجود ندارد.

الف) ابتدا ستون های نام آهنگ، سال انتشار و نام خواننده را از دادگان حذف کنید (علت را توضیح دهید). در ستون محبوبیت، میزان محبوبیت هر آهنگ با عددی بین ۰ (کمترین محبوبیت) و ۱۰۰ (بیشترین محبوبیت) نشان داده شده است. برای سادگی، میزان محبوبیت را به پنج دسته تقسیم کنید، به این صورت که اعداد ۰ تا ۲۰ را با ۱ (نامحوب)، اعداد ۲۰ تا ۴۰ را با ۲ (محبوبیت کم) و ... جایگزین کنید.

Delete song, year, artist columns & shifting popularity to 1-5

```
In [38]: musics_df['popularity'] = musics_df['popularity'].replace(0, 1)
musics_df['category'] = pd.cut(musics_df['popularity'], bins=[0, 20, 40, 60, 80, 100], labels=['1', '2', '3', '4', '5'])
print(musics_df['category'])
musics_df = musics_df.drop(['song', 'year', 'artist', 'popularity'], axis=1)
musics_df = musics_df.rename(columns={'category': 'popularity'})
musics_df.head(15)
```

```
0      4
1      4
2      4
3      4
4      4
..
1995    4
1996    4
1997    4
1998    4
1999    5
Name: category, Length: 2000, dtype: category
Categories (5, object): ['1' < '2' < '3' < '4' < '5']
```

```
Out[38]:
```

s	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	genre	popularity
0	False	0.751	0.834	1	-5.444	0	0.0437	0.30000	0.000018	0.3550	0.894	95.053	pop	4
6	False	0.434	0.897	0	-4.918	1	0.0488	0.01030	0.000000	0.6120	0.684	148.726	rock, pop	4
6	False	0.529	0.496	7	-9.007	1	0.0290	0.17300	0.000000	0.2510	0.278	136.859	pop, country	4
3	False	0.551	0.913	0	-4.063	0	0.0466	0.02630	0.000013	0.3470	0.544	119.992	rock, metal	4
0	False	0.614	0.928	8	-4.806	0	0.0516	0.04080	0.001040	0.0845	0.879	172.656	pop	4
3	True	0.706	0.888	2	-6.959	1	0.0654	0.11900	0.000096	0.0700	0.714	121.549	hip hop, pop, R&B	4
0	True	0.949	0.661	5	-4.244	0	0.0572	0.03020	0.000000	0.0454	0.760	104.504	hip hop	5
0	False	0.708	0.772	7	-4.264	1	0.0322	0.02670	0.000000	0.4670	0.861	103.035	pop, rock	4
3	False	0.713	0.678	5	-3.525	0	0.1020	0.27300	0.000000	0.1490	0.734	138.009	pop, R&B	4
3	False	0.720	0.808	6	-5.627	1	0.0379	0.00793	0.029300	0.0634	0.869	126.041	Dance/Electronic	4
9	False	0.617	0.728	7	-7.932	1	0.0292	0.03280	0.048200	0.3600	0.808	139.066	pop	1
3	False	0.745	0.958	7	-9.664	1	0.0287	0.08130	0.324000	0.5330	0.960	129.962	pop	3
3	False	0.822	0.922	11	-5.798	0	0.0989	0.02910	0.325000	0.2520	0.568	163.826	pop	3
0	False	0.586	0.659	0	-7.920	0	0.0304	0.01100	0.000000	0.1060	0.147	111.989	rock, pop	4
0	False	0.689	0.685	3	-5.153	1	0.0478	0.09210	0.000000	0.1190	0.398	160.067	pop, Dance/Electronic	4

در این قسمت ستون های نام آهنگ، سال انتشار و نام خواننده حذف شده اند. علت این امر این است که برای مثال نام آهنگ در هر سطر مقدار متفاوتی دارد لذا پراکندگی داده های این سطر ها زیاد می باشد. پراکندگی زیاد داده ها و همچنین کیفی بودن این مقادیر موجب می شود این ستون ها در تربیت مدل موجب افزایش خطای شود و به طور کلی کمکی به تربیت بهتر مدل نکنند.

همچنین در این قسمت ستون **popularity** که میزان محبوبیت را از ۰ تا ۱۰۰ نشان می داد به مقادیر ۱ تا ۵ شیفت شده است.

ب) ورودی‌ها و خروجی‌های کیفی را به مقادیر کمی تبدیل کنید. تنها ستون‌های مربوط به ویژگی صریح بودن (**explicit**) و خروجی گونه (**genre**) کیفی هستند. ویژگی صریح بودن تنها دارای دو مقدار درست و نادرست است که می‌توانید آن‌ها را به ترتیب به ۰ و ۱ تبدیل کنید

explicit column quantification

```
In [43]: musics_df['explicit'] = musics_df['explicit'].astype(bool)
musics_df['explicit'] = musics_df['explicit'].replace({True: 1, False: 0})
musics_df.head(20)
```

Out[43]:

	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	g
0	211160	0	0.751	0.834	1	-5.444	0	0.0437	0.30000	0.000018	0.3550	0.894	95.053	
1	167066	0	0.434	0.897	0	-4.918	1	0.0488	0.01030	0.000000	0.6120	0.684	148.726	rock
2	250546	0	0.529	0.496	7	-9.007	1	0.0290	0.17300	0.000000	0.2510	0.278	136.859	pop, co
3	224493	0	0.551	0.913	0	-4.063	0	0.0466	0.02630	0.000013	0.3470	0.544	119.992	rock, r
4	200560	0	0.614	0.928	8	-4.806	0	0.0516	0.04080	0.001040	0.0845	0.879	172.656	
5	253733	1	0.706	0.888	2	-6.959	1	0.0654	0.11900	0.000096	0.0700	0.714	121.549	hip hop,
6	284200	1	0.949	0.661	5	-4.244	0	0.0572	0.03020	0.000000	0.0454	0.760	104.504	hip
7	258560	0	0.708	0.772	7	-4.264	1	0.0322	0.02670	0.000000	0.4670	0.861	103.035	pop,
8	271333	0	0.713	0.678	5	-3.525	0	0.1020	0.27300	0.000000	0.1490	0.734	138.009	pop,
9	307153	0	0.720	0.808	6	-5.627	1	0.0379	0.00793	0.029300	0.0634	0.869	126.041	Dance/Elect
10	238759	0	0.617	0.728	7	-7.932	1	0.0292	0.03280	0.048200	0.3600	0.808	139.066	
11	268863	0	0.745	0.958	7	-9.664	1	0.0287	0.08130	0.324000	0.5330	0.960	129.962	
12	306333	0	0.822	0.922	11	-5.798	0	0.0989	0.02910	0.325000	0.2520	0.568	163.826	
13	285960	0	0.586	0.659	0	-7.920	0	0.0304	0.01100	0.000000	0.1060	0.147	111.989	rock
14	294200	0	0.689	0.685	3	-5.153	1	0.0478	0.09210	0.000000	0.1190	0.398	160.067	Dance/Elect
15	284000	0	0.797	0.622	6	-5.642	0	0.2900	0.08070	0.000000	0.0841	0.731	93.020	hip hop,
16	245400	0	0.761	0.716	10	-5.800	0	0.0560	0.39600	0.000000	0.0771	0.649	119.410	
17	214883	0	0.671	0.880	8	-6.149	0	0.0552	0.00181	0.691000	0.2850	0.782	136.953	
18	285426	0	0.740	0.876	6	-6.870	0	0.0369	0.01730	0.001520	0.0785	0.825	127.002	
19	161506	1	0.922	0.909	10	-2.429	0	0.2700	0.02810	0.000000	0.0856	0.309	95.295	hip

در مرحله اول از قسť ب برای کمی سازی ستون **explicit** مطابق شکل بالا مقادیر این ستون به جای **True , 0** ، **False , 1** جایگذاری می شوند.

```

genre column quantification (one hot encoding)

In [6]: ┆ all_genre = set()
          for s in musics_df['genre']:
              all_genre.update(s.split(", "))

          num_genre = len(all_genre)
          genre_ids = {s: i for i,s in enumerate(all_genre)}

          new_genre1 = []
          for s in musics_df['genre']:
              row_arr = [0] * num_genre
              for sp in s.split(", "):
                  if sp in all_genre:
                      row_arr[genre_ids[sp]] = 1
              new_genre1.append(row_arr)

          new_genre1 = np.array(new_genre1)

          print(new_genre1)
          print(genre_ids)
          new_genre1 = pd.DataFrame(new_genre1)

          new_genre1
          [[0 0 1 ... 0 0 0]
           [0 0 1 ... 0 0 0]
           [0 0 1 ... 0 0 0]
           ...
           [0 0 0 ... 0 0 1]
           [0 0 1 ... 0 0 0]
           [0 0 0 ... 0 0 1]
           ...
           [blues': 0, 'World/Traditional': 1, 'pop': 2, 'R&B': 3, 'Folk/Acoustic': 4, 'country': 5, 'rock': 6, 'Dance/Electronic': 7,
            'easy listening': 8, 'set()': 9, 'latin': 10, 'jazz': 11, 'classical': 12, 'metal': 13, 'hip hop': 14}]

Out[6]:
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
2 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
3 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
4 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
...
1995 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1996 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1997 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
1998 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2000 rows × 15 columns

```

برای کمی سازی ستون `genre` با توجه به اینکه مقادیر این ستون مانند مرحله قبل از دو نوع تشکیل نشده اند نمی توان از روش قبل برای کمی سازی استفاده کرد.

در این قسمت برای کمی سازی ابتدا مقادیر این ستون که بین آن ها کاما می باشد جدا سازی می شوند. سپس یک فایل جدید تعریف می شود به این صورت که در هر سطر، ستون مربوط به آن مقدار کیفی برابر یک و سایر ستون ها برابر صفر قرار داده می شوند.

برای این کار می توان از دستور `one hot encoding` و `dummies` یا مطابق شکل بالا از یک حلقه `for` استفاده کرد. از آن جایی که جداسازی داده های هر سطر نیز مد نظر می باشد در اینجا از دو حلقه `for` برای جداسازی و اعمال `one hot encoding` استفاده شده است.

برای مثال در سطر اول چون `genre` آن `pop` است، ستون دوم که نشان دهنده `pop` می باشد برابر یک قرار داده شده و سایر ستون ها صفر هستند. یا در سطر سوم که شامل `pop` و `country` است مقادیر ستون دوم و پنجم برابر یک و سایر ستون ها برابر صفر قرار داده می شوند. به این ترتیب در هر سطر ستون مربوط به آن `genre` برابر یک قرار داده می شود. با توجه به اینکه در فایل ورودی 15 زانر متفاوت وجود دارد یک ماتریس 15 ستونه حاصل می شود.

Convert columns to a vector

```
In [7]: def combine_rows(row):
    return row.values.tolist()

new_genre1 = new_genre1.apply(combine_rows, axis=1)
new_genre_col = new_genre1
print(new_genre1.head())
new_genre = pd.DataFrame(new_genre_col)
musics_df = musics_df.assign(new_genre=new_genre)
musics_df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

out[7]:

	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	genre	popularity	new_genre
0	0	0.751	0.834	1	-5.444	0	0.0437	0.3000	0.000018	0.3550	0.894	95.053	pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.434	0.897	0	-4.918	1	0.0488	0.0103	0.000000	0.6120	0.684	148.726	rock, pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.529	0.496	7	-9.007	1	0.0290	0.1730	0.000000	0.2510	0.278	136.859	pop, country	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.551	0.913	0	-4.063	0	0.0466	0.0263	0.000013	0.3470	0.544	119.992	rock, metal	4	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
0	0	0.614	0.928	8	-4.806	0	0.0516	0.0408	0.001040	0.0845	0.879	172.656	pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
...	
0	0	0.842	0.734	1	-5.065	0	0.0588	0.0427	0.000000	0.1060	0.952	137.958	pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.552	0.702	9	-5.707	1	0.1570	0.1170	0.000021	0.1050	0.564	169.994	pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.847	0.678	9	-8.635	1	0.1090	0.0669	0.000000	0.2740	0.811	97.984	hip hop, country	4	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]
0	0	0.741	0.520	8	-7.513	1	0.0656	0.4500	0.000002	0.2220	0.347	102.998	pop	4	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
0	0	0.695	0.762	0	-3.497	1	0.0395	0.1920	0.002440	0.0863	0.553	120.042	hip hop	5	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

nns

پس از اینکه یک ماتریس 15 ستونه برای نمایش genre هر سطر بدست آمد لا توجه به خواسته‌ی سوال مقادیر هر سطر به یک بردار 1×15 تبدیل می‌شوند و در یک ستون جدید با نام new genre ریخته می‌شوند.
هر بردار از ستون new genre داده‌های ستون genre را با مقادیر 1، 0 نشان می‌دهد.

ج) توزیع دادگان را برای تمام ۱۳ ویژگی (همه ستون‌ها بجز ستون‌های محبوبیت و گونه و سه ویژگی حذف شده) بدست آورید. داده‌های با درجه محبوبیت متفاوت را بر نگاه‌های متفاوت نمایش دهید. پراکنده‌گی داده‌ها را بر حسب ویژگی‌های مختلف نشان دهید. بر پایه این پراکنده‌گی‌ها برداشت خود را از ماهیت داده‌ها بیان کنید (راهنمایی: برای نمایش توزیع داده‌ها می‌توانید از نمودارهای نقشه گرمایی، گسسته، هیستوگرام یا جعبه‌ای استفاده کنید).

```
In [52]: musics_df2 = musics_df
musics_df2 = musics_df2.drop(['new_genre', 'popularity', 'genre'], axis=1)
musics_df2
```

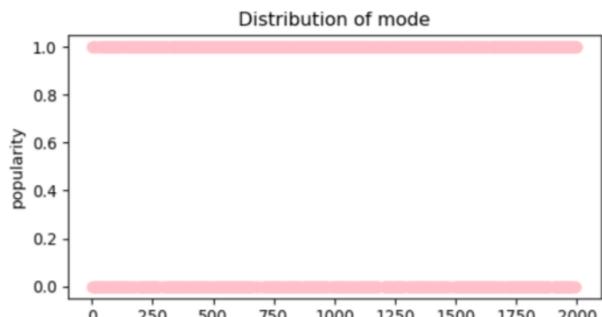
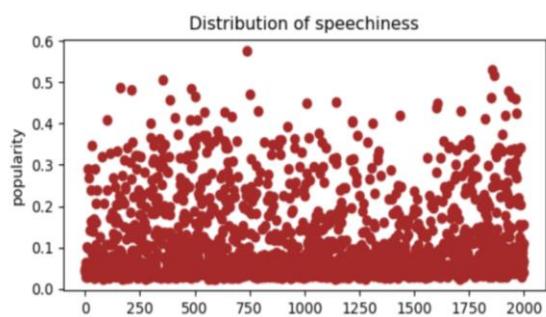
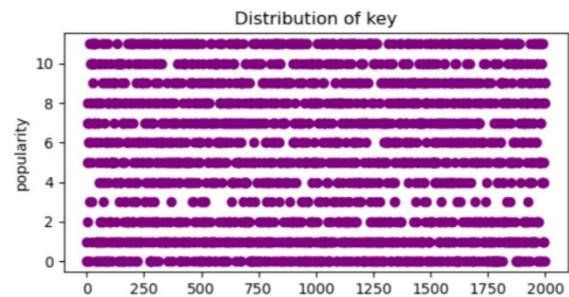
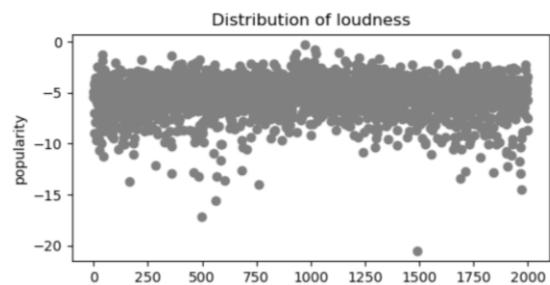
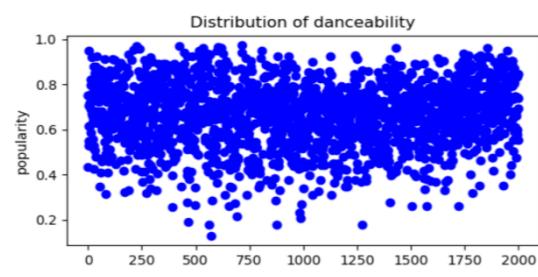
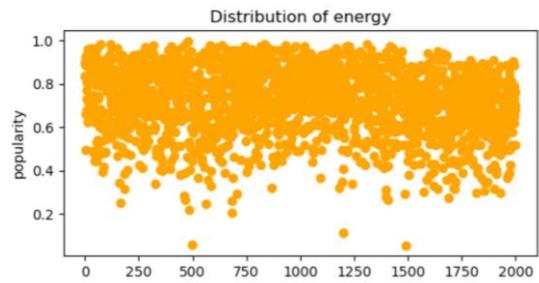
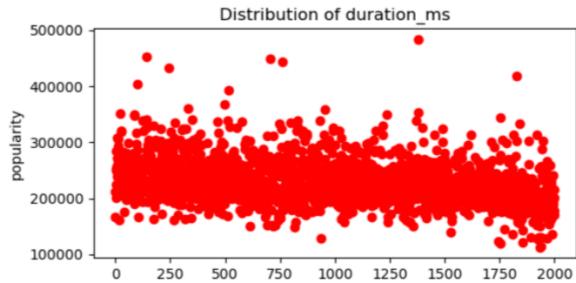
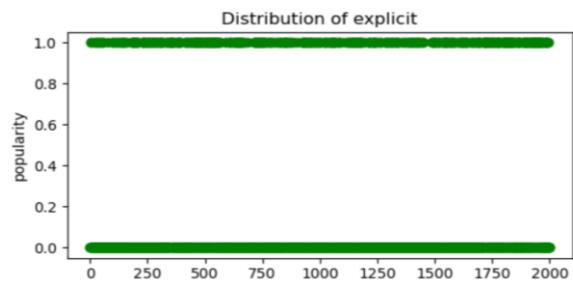
	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo
0	211160	0	0.751	0.834	1	-5.444	0	0.0437	0.3000	0.000018	0.3550	0.894	95.053
1	167066	0	0.434	0.897	0	-4.918	1	0.0488	0.0103	0.000000	0.6120	0.684	148.726
2	250546	0	0.529	0.496	7	-9.007	1	0.0290	0.1730	0.000000	0.2510	0.278	136.859
3	224493	0	0.551	0.913	0	-4.063	0	0.0466	0.0263	0.000013	0.3470	0.544	119.992
4	200560	0	0.614	0.928	8	-4.806	0	0.0516	0.0408	0.001040	0.0845	0.879	172.656
...
1995	181026	0	0.842	0.734	1	-5.065	0	0.0588	0.0427	0.000000	0.1060	0.952	137.958
1996	178426	0	0.552	0.702	9	-5.707	1	0.1570	0.1170	0.000021	0.1050	0.564	169.994
1997	200593	0	0.847	0.678	9	-8.635	1	0.1090	0.0669	0.000000	0.2740	0.811	97.984
1998	171029	0	0.741	0.520	8	-7.513	1	0.0656	0.4500	0.000002	0.2220	0.347	102.998
1999	215280	0	0.695	0.762	0	-3.497	1	0.0395	0.1920	0.002440	0.0863	0.553	120.042

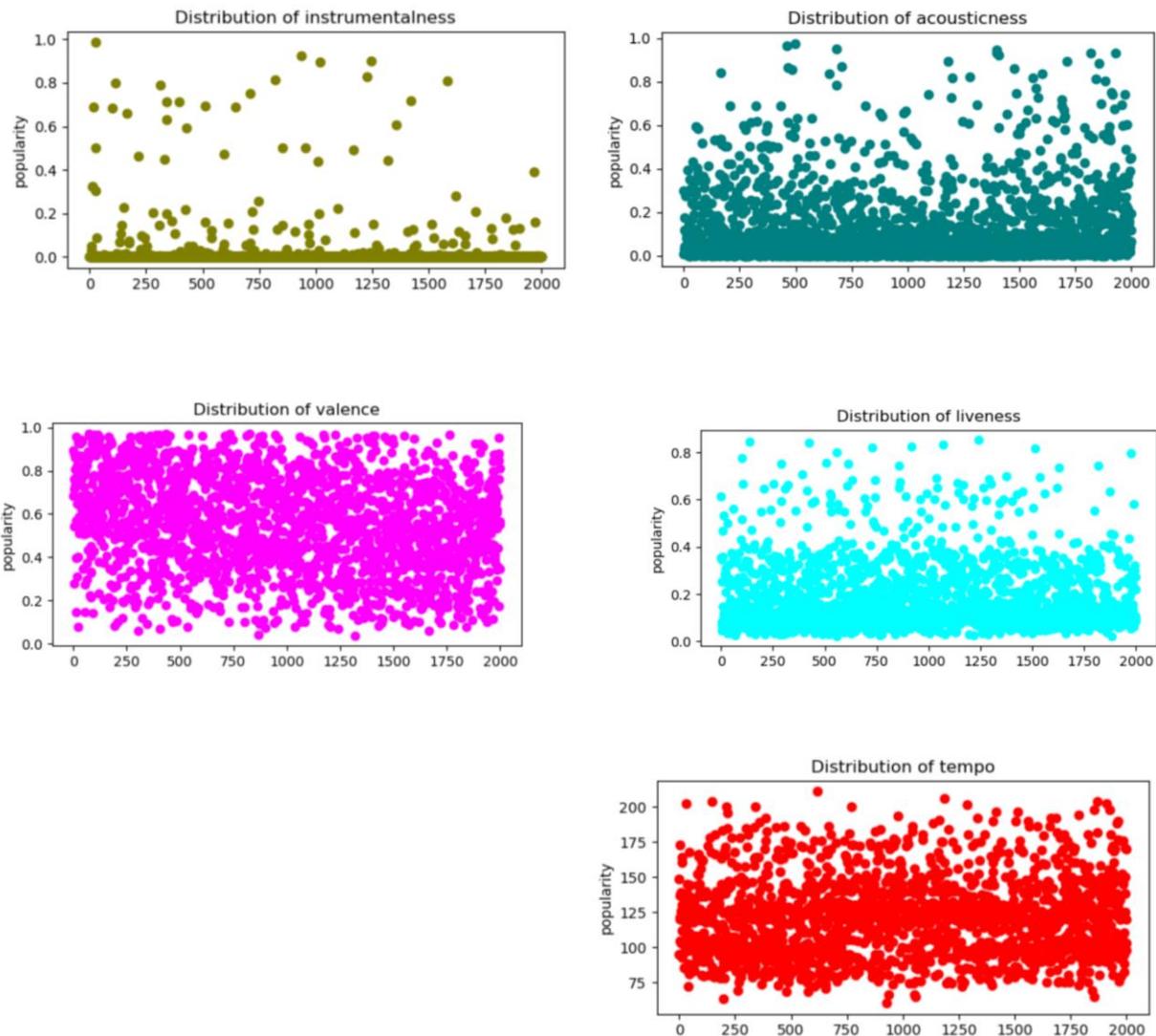
2000 rows × 13 columns

برای نمایش پراکنده‌گی داده‌ها بر حسب ویژگی ابتدا ستون‌های خواسته شده حذف می‌شوند یک data frame جدید شامل ویژگی‌هایی که قرار است توزیع دادگان آن‌ها محاسبه شود ساخته می‌شود.

```
In [54]: colors = ['red', 'green', 'blue', 'orange', 'purple', 'gray', 'pink', 'brown', 'teal', 'olive', 'cyan', 'magenta']
fig, axs = plt.subplots(nrows=14, figsize=(6, 40))
for i, col in enumerate(musics_df2.columns):
    axs[i].scatter(musics_df.index, musics_df2[col], color=colors[i%len(colors)])
    axs[i].set_title(f'Distribution of {col}')
    axs[i].set_ylabel("popularity")
plt.tight_layout()
plt.show()
```

برای نمایش توزیع دادگان از دستور plt.subplots از کتابخانه matplotlib استفاده می‌شود و به کمک یک حلقه for برای هر ویژگی نمودار نقطه‌ای توزیع داده‌ها بر حسب محبوبیت بدست می‌آید که در ادامه این نمودارها برای هر کدام از ویژگی‌ها نمایش داده شده‌اند.





برای داده‌های دو بعدی، پراکندگی نشان‌دهنده‌ی این است که داده‌ها به چه شکلی درون فضا پخش شده‌اند و چقدر از نواحی فضا خالی هستند. در واقع، پراکندگی داده‌ها نشان‌دهنده‌ی شدت و تنوع توزیع داده‌ها در مجموعه داده‌ها است. همچنین، پراکندگی داده‌ها می‌تواند در مدل‌هایی مانند SVM و KNN، در تنظیم پارامترهای مانند تعداد همسایگان در KNN و پارامتر C در SVM مؤثر باشد.

در توزیع داده‌های این سوال برای مثال ویژگی‌های valence و tempo با پراکندگی خوب در کل فضا و عدم تمرکز در ناحیه مشخصی موجب تربیت بهینه مدل می‌شوند در صورتی که در ویژگی‌هایی مانند اکثیریت داده‌ها در یک ناحیه تمرکز دارد در حالی که تعداد کمی از داده‌ها در نواحی دیگر پخش شده‌اند و این عامل موجب می‌شود داده‌های این ویژگی‌ها نتوانند تمامی شرایط موجود در داده‌ها را پوشش دهند و در نهایت دقت کاهش یابد.

د) ستون گونه را کنار گذاشته و در صورت نیاز تمام ستون ها به جز محبوبیت را نرمال سازی کنید. سپس رابطه و تاثیر گذاری هر ویژگی را بر میزان محبوبیت آهنگ پیدا کنید. (راهنمایی: می توانید از معیارهای آماری مانند همبستگی یا correlation استفاده کنید تا میزان تاثیر داده ها بر روی خروجی و حتی رابطه آن ها با یکدیگر را مشاهده کنید. برای نمایش هم می توانید از plot در کتابخانه seaborn یا از نقشه گرمایی استفاده کنید).

separating x,y & normalizing & calculating correlation factor

```
In [81]: from sklearn.preprocessing import StandardScaler
musics_df0 = musics_df0.drop(['song', 'year', 'artist', 'genre', 'category'], axis=1)

X = musics_df0.iloc[:, :-1]
y = musics_df0.iloc[:, -1]

scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

corr_matrix = musics_df0.corr(numeric_only=True)
corr_matrix
```

	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness
duration_ms	1.000000	0.123595	-0.060057	-0.078763	-0.002560	-0.079912	-0.003848	0.066998	0.010923	-0.004208	0.024941
explicit	0.123595	1.000000	0.248845	-0.162462	0.003320	-0.089829	0.049576	0.417343	-0.033523	-0.082522	0.008884
danceability	-0.060057	0.248845	1.000000	-0.104038	0.032731	-0.033315	-0.067528	0.145590	-0.065429	0.023207	-0.126413
energy	-0.078763	-0.162462	-0.104038	1.000000	-0.003446	0.651016	-0.040651	-0.057018	-0.445469	0.037861	0.156761
key	-0.002560	0.003320	0.032731	-0.003446	1.000000	-0.007474	-0.153182	0.007147	0.002365	-0.008173	-0.033071
loudness	-0.079912	-0.089829	-0.033315	0.651016	-0.007474	1.000000	-0.028133	-0.076388	-0.310039	-0.104925	0.102159
mode	-0.003848	0.049576	-0.067528	-0.040651	-0.153182	-0.028133	1.000000	-0.000077	0.005744	-0.038613	0.025439
speechiness	0.066998	0.417343	0.145590	-0.057018	0.007147	-0.076388	-0.000077	1.000000	0.000394	-0.062954	0.061172
acousticness	0.010923	-0.033523	-0.065429	-0.445469	0.002365	-0.310039	0.005744	0.000394	1.000000	-0.005214	-0.110043
instrumentalness	-0.004208	-0.082522	0.023207	0.037861	-0.008173	-0.104925	-0.038613	-0.062954	-0.005214	1.000000	-0.034897
liveness	0.024941	0.008884	-0.126413	0.156761	-0.033071	0.102159	0.025439	0.061172	-0.110043	-0.034897	1.000000
valence	-0.116870	-0.045455	0.403178	0.334474	0.036977	0.232150	-0.074681	0.073605	-0.128128	-0.015192	0.019040
tempo	-0.028603	0.013221	-0.173418	0.153719	-0.001431	0.080709	0.048434	0.057747	-0.103660	0.034608	0.028636
popularity	0.050186	0.046902	-0.003538	-0.013894	0.014641	0.030809	-0.021329	0.021040	0.024641	-0.048381	-0.010228

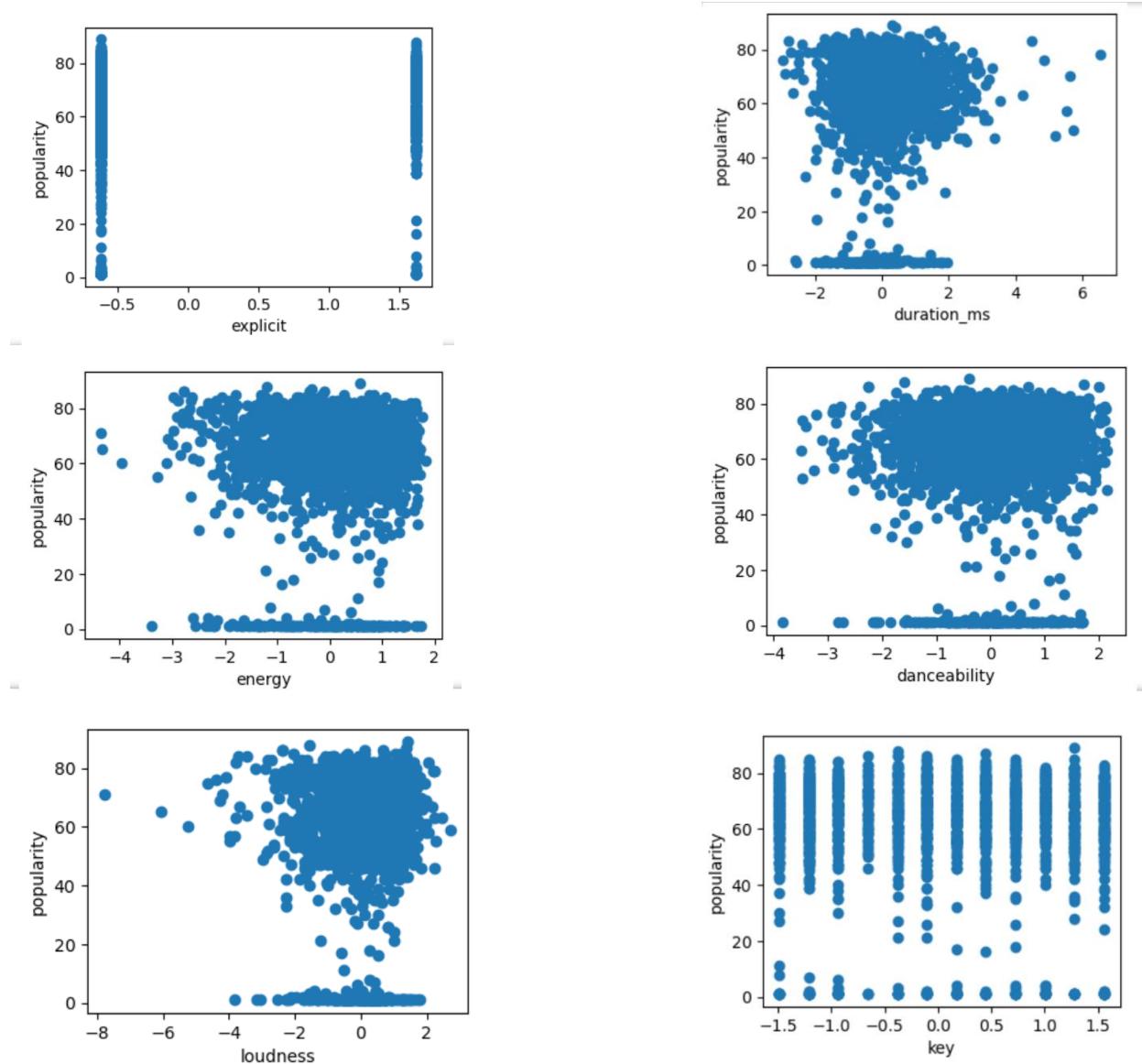
داده های بالا ضریب correlation را برای ویژگی ها بر حسب سک دیگر و محبوبیت نشان می دهد.

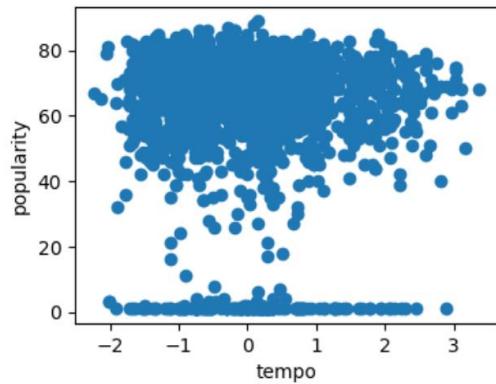
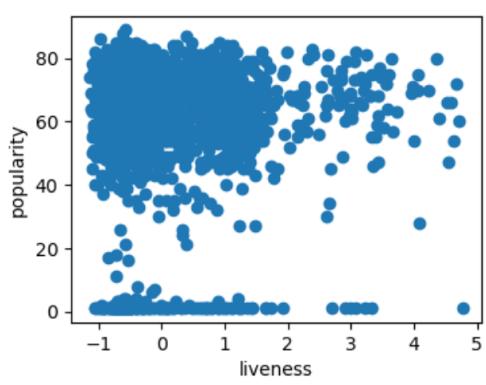
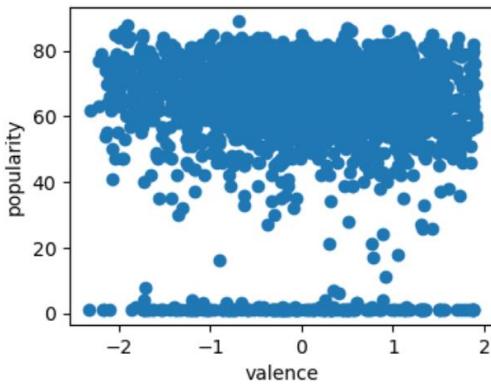
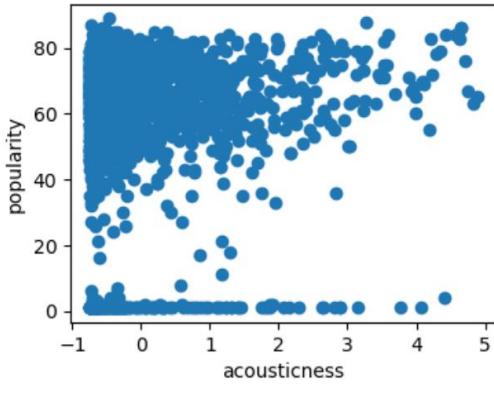
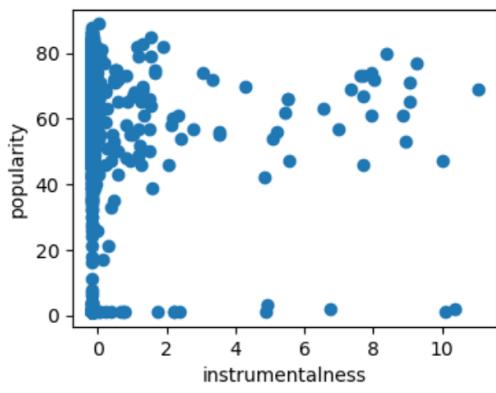
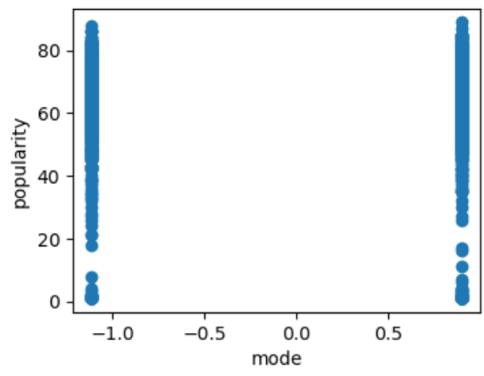
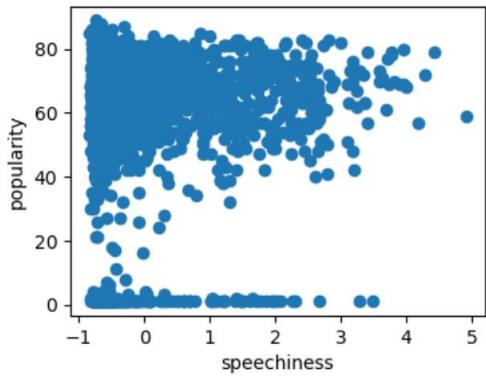
مطابق دستور بالا همه ی ویژگی های ورودی که شامل 13 ستون می باشد درون متغیر X و ستون popularity درون متغیر y ریخته می شوند. سپس همه ی ستون های ورودی با استفاده از دستور StandardScaler از کتابخانه ی normalize، sklearn corr می شوند و با استفاده از دستور ضریب همبستگی برای همه داده ها محاسبه می شود.

در جدول بالا مشاهده می شود که رابطه و تاثیر گذاری هر ویژگی بر روی میزان محبوبیت و همچنین تاثیر آن بر روی سایر ویژگی ها با استفاده از ضریب correlation نشان داده شده است. ستون آخر نمایانگر تاثیر گذاری هر ویژگی بر روی میزان محبوبیت می باشد.

ضریب همبستگی (Correlation coefficient) نشان دهنده میزان رابطه خطی بین دو متغیر است. این ضریب بین دو متغیر به صورت عددی در بازه -1 تا 1 تعریف می‌شود. اگر ضریب همبستگی برابر با 1 باشد، به این معناست که دو متغیر کاملاً مرتبط و همبستگی مثبت دارند، یعنی هر چه ارزش یکی افزایش یابد، ارزش دیگری نیز افزایش پیدا می‌کند. اگر ضریب همبستگی برابر با -1 باشد، به این معناست که دو متغیر کاملاً معکوس و همبستگی منفی دارند، یعنی هر چه ارزش یکی افت کاهش یابد، ارزش دیگری افزایش پیدا می‌کند. و در صورتی که ضریب همبستگی برابر با صفر باشد، نشان دهنده عدم وجود همبستگی بین دو متغیر است، یعنی تغییرات یکی از متغیرها با تغییرات دیگری همراه نیست.

در ادامه نمودار تاثیر گذاری هر ویژگی بر روی میزان محبوبیت رسم شده است.





۵) داده ها را به سه بخش آموزش (training)، آزمایش (test) و ارزیابی (validation) تقسیم کنید. پیشنهاد می شود ۸۰٪ کل داده ها به آموزش، ۱۰٪ به آزمایش و ۱۰٪ به ارزیابی اختصاص داده شود.

```
define X , y

In [196]: musics_df3 = musics_df
musics_df3 = musics_df3.drop(['new_genre','genre'], axis=1)
feature_df = musics_df3[['duration_ms','explicit','danceability','energy','key','loudness','mode','speechiness','acousticness']]
X = np.asarray(feature_df)
Scaler= preprocessing.StandardScaler().fit(X)
X =Scaler.transform(X.astype(float))
print('X : \n',X[0:10])
print('y : \n',y[0:10])

X :
[[ -0.44951624 -0.61665408  0.59525376  0.74413046 -1.21134817  0.03540329
-1.11339196 -0.62274794  0.98697358 -0.17331627  1.23571936  1.558025566
-0.92986699]
[ -1.57646805 -0.61665408 -1.66288167  1.15668497 -1.48803803  0.30751937
0.8981563 -0.56969739 -0.68466912 -0.17351798  3.06315751  0.59920732
1.06094401]
[ 0.5571088 -0.61665408 -0.98615338 -1.46925722  0.44879094 -1.8078469
0.8981563 -0.77565836  0.25415127 -0.17351798  0.49621132 -1.23948612
0.62077942]
[ -0.10875222 -0.61665408 -0.82943736  1.26146072 -1.48803803  0.74983733
-1.11339196 -0.59258194 -0.59234505 -0.17336413  1.17883413 -0.0348249
-0.00484257]
[ 0.7204304 -0.61665408 -0.38065966  1.35968798  0.72548079  0.36546043
-1.11339196 -0.5405716 -0.50867636 -0.16166597 -0.68771262  1.48232363
1.94854313]
[ 0.63856196  1.62165473  0.27469826  1.09774861 -0.93465832 -0.7483511
0.8981563 -0.39702305 -0.05744247 -0.17241939 -0.7908171  0.73507137
0.05290886]
[ 1.41723572  1.62165473  2.00569797 -0.38875732 -0.10458876  0.65620043
-1.11339196 -0.48232001 -0.56984106 -0.17351798 -0.9657392  0.94339624
-0.57931541]
[ 0.76193013 -0.61665408  0.28894517  0.33812444  0.44879094  0.64585381
0.8981563 -0.74237174 -0.59003695 -0.17351798  2.03211264  1.40808052
-0.63380828]
[ 1.08838171 -0.61665408  0.32456245 -0.27743308 -0.10458876  1.02816139
-1.11339196 -0.01630732  0.83117671 -0.17351798 -0.22907542  0.8256474
0.66343462]
[ 2.008386713 -0.61665408  0.37442663  0.57386987  0.17210109 -0.05926828
0.8981563 -0.68307994 -0.69834462  0.16038981 -0.83774742  1.43703562
0.21952379]]
y :
[4. 4. 4. 4. 4. 5. 4. 4. 4.]
```

ابتدا مطابق شکل بالا ویژگی های ورودی در ماتریس ۱۳ ستونه X و داده های خروجی در ماتریس تک ستونه y ریخته می شوند.

separating X,y to train,test,validation data

```
In [36]: from sklearn.model_selection import train_test_split
musics_df3['popularity'] = musics_df3['popularity'].astype('float')
y = np.asarray(musics_df3['popularity'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1111)
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)
print ('val set:', X_val.shape, y_val.shape)

Train set: (1600, 13) (1600,)
Test set: (200, 13) (200,)
val set: (200, 13) (200,)
```

مطابق خواسته سوال داده های ورودی با دستور `sklearn train_test_split` از کتابخانه `train_test_split` به سه بخش تقسیم می شوند که ۸۰ درصد داده ها در قسمت `train` ، `test` ، `validation` درصد در `train` و `test` و `validation` بخش ۱۰ درصد در بخش `test` ریخته می شود.

و) با استفاده از داده‌های آموزش و به کمک الگوریتم‌های درخت تصمیم‌گیری، جنگل تصادفی، k نزدیک‌ترین همسایه و بردار پشتیبان، مدل خود را برای پیش‌بینی محبوبیت آهنگ‌ها تربیت کنید. پس از آن، عملکرد مدل را بر روی داده‌های آزمایش (با استفاده از ماتریس سردرگمی) بررسی کنید. بهترین مدل را انتخاب کرده و دلیل برتری آن را بر مدل‌های (الگوریتم‌های) دیگر شرح دهید. آیا دقت بهترین مدل را مناسب می‌دانید یا خیر؟ پیشنهادهای خود را برای افزایش دقت ارائه دهید.

مدل درخت تصمیم‌گیری:

Decision Tree Classifier

```
In [246]: M from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import confusion_matrix

model = DecisionTreeClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

accuracy = (cm[0,0]+cm[1,1]+cm[2,2]+cm[3,3]+cm[4,4])/len(y_test)

print('Confusion Matrix:\n', cm)
print('Accuracy:', accuracy)

Confusion Matrix:
[[ 1  0  8  8  1]
 [ 2  0  0  2  0]
 [ 3  0  13 18  4]
 [19  0  25 80  6]
 [ 1  0  2  6  1]]
Accuracy: 0.475
```

مدل جنگل تصادفی:

Random Forest Classifier

```
In [247]: M from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import confusion_matrix

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
accuracy = (cm[0,0]+cm[1,1]+cm[2,2]+cm[3,3]+cm[4,4])/len(y_test)

print('Confusion Matrix:\n', cm)
print('Accuracy:', accuracy)

Confusion Matrix:
[[ 0  0  1 17  0]
 [ 1  0  0  3  0]
 [ 0  0  3 35  0]
 [ 0  0  3 127  0]
 [ 0  0  0   9  1]]
Accuracy: 0.655
```

: KNN مدل

KNN Classifier

```
In [248]: └─▶ from sklearn.neighbors import KNeighborsClassifier
      from sklearn.metrics import confusion_matrix

      knn = KNeighborsClassifier()
      knn.fit(X_train, y_train)

      y_pred = knn.predict(X_test)
      cm = confusion_matrix(y_test, y_pred)

      accuracy = (cm[0,0]+cm[1,1]+cm[2,2]+cm[3,3]+cm[4,4])/len(y_test)

      print('Confusion Matrix:\n', cm)
      print('Accuracy:', accuracy)

Confusion Matrix:
[[ 2  0  4 12  0]
 [ 0  0  1  3  0]
 [ 6  0  4 28  0]
 [ 8  0 24 97  1]
 [ 0  0  4  6  0]]
Accuracy: 0.515
```

: SVM مدل

SVM model

```
In [249]: └─▶ from sklearn.svm import SVC
      from sklearn.metrics import confusion_matrix

      model = SVC(kernel='linear', C=1.0, gamma='auto')
      model.fit(X_train, y_train)

      y_pred = model.predict(X_test)
      cm = confusion_matrix(y_test, y_pred)

      accuracy = (cm[0,0]+cm[1,1]+cm[2,2]+cm[3,3]+cm[4,4])/len(y_test)

      print('Confusion Matrix:\n', cm)
      print('Accuracy:', accuracy)

Confusion Matrix:
[[ 0  0  0 18  0]
 [ 0  0  0  4  0]
 [ 0  0  0 38  0]
 [ 0  0  0 130  0]
 [ 0  0  0 10  0]]
Accuracy: 0.65
```

باتوجه به اینکه تقسیم داده ها به گروه های تست، تمرین و ارزیابی به صورت تصادفی بوده است، در هر مرحله اجرای کد، مقادیر ماتریس سردرگمی و دقت های حاصل شده برای هر مدل متفاوت خواهد بود. لذا اعداد ذکر شده در مدل های بالا ثابت نمی باشند و به ازای هربار اجرای کد تغییر می کنند. بنابراین امکان دارد در هر مرحله یکی از مدل ها بیشترین دقت را حاصل کند. ولی به طور کلی و با 50+ بار اجرای کد، مدل های SVM , Random Forest Classifier بیشترین دقت را حاصل می کنند.

مقایسه مدل های تربیت شده :

Model	Accuracy (%)
Decision Tree Classifier	47.5
Random Forest Classifier	65.5
KNN	51.5
SVM	65

در مدل های تربیت شده معمولاً بیشترین دقت مربوط به مدل های SVM و Random Forest Classifier می باشد. ولی با توجه به مقایسه ماتریس های سردرگمی نتیجه گیری می شود که مدل SVM همه می داده ها را در بازه ی محبوبیت 40-60 پیشگویی کرده است و صرفاً در این سوال با توجه به تراکم خروجی ها در این بازه مقدار دقت آن بیشتر از سایر مدل ها شده است. به عبارت دیگر با توجه به اینکه هر داده ورودی به آن وارد شود خروجی ثابتی را برای آن پیش بینی می کند لذا مدل کارآمدی نیست.

بنابراین به نظر می رسد برای این مسئله مدل **Random Forest Classifier** بهترین مدل می باشد که دقت آن به 65.5 نیز می رسد.

برای افزایش دقت مدل (DTS (Decision Tree Classifier)، می توان از رویکردهای زیر استفاده کرد:

- ۱ - تنظیم پارامترهای مدل مانند min_sample_leaf یا max_depth
- ۲ - استفاده از روش های ensemble که در این روش، چندین مدل DTS با پارامترهای مختلف آموزش داده می شوند و سپس نتایج آن ها ترکیبه دسته بندی نهایی ترکیب می شوند.
- ۳ - استفاده از روش های feature selection که این روش ها باعث کاهش تعداد فیچرها و افزایش دقت مدل DTS می شوند.
- ۴ - اصلاح overfitting با روش هایی مانند regularization و pruning

ز) امتیازی: بعد از آموزش، مواردی را که به غلط توسط مدل پیش‌بینی شده جدا کنید. با استفاده از تحلیل آماری یا تفسیر بصری علت این پیش‌بینی غلط را توضیح دهید و با ذکر دلیل، در جهت بهبود دقت مدل تلاش کنید.

False prediction

```
In [260]: false_predictions = X_test[y_test != y_pred]
false_predictions = pd.DataFrame(false_predictions)
false_predictions
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.645469	1.621655	-0.003117	0.174412	-1.211348	-0.800086	0.898156	0.888673	1.200473	-0.173518	-0.470838	1.658947	-1.231643
1	-1.101755	-0.616654	0.851698	0.835809	-1.211348	0.422367	0.898156	-0.506245	-0.216124	-0.173518	-0.221965	1.790282	0.069229
2	-0.482588	-0.616654	-1.043141	0.220252	-0.104589	0.707934	0.898156	-0.417827	-0.528872	-0.173518	0.410883	-0.904355	0.213404
3	1.090759	-0.616654	-1.028894	0.390512	0.725481	1.059719	0.898156	-0.294043	-0.532911	-0.173518	-0.754553	-1.624434	0.292409
4	0.420476	-0.616654	0.481278	-1.763939	-1.488038	-1.480376	0.898156	-0.468797	-0.380000	-0.173370	1.022400	-2.108110	-0.079730
...
65	-0.675116	-0.616654	0.474155	-0.545921	-0.381279	-0.085135	-1.113392	0.389373	-0.477517	-0.173257	-0.860502	0.436170	1.849175
66	-0.417492	-0.616654	1.535550	1.091200	-1.211348	0.306485	0.898156	0.129322	-0.680053	7.006069	-0.027602	0.436170	0.174013
67	0.583357	-0.616654	0.288945	0.331576	-0.104589	-0.764390	-1.113392	0.285353	0.461880	-0.173457	0.681088	-0.261265	0.292668
68	0.289645	1.621655	1.200747	-0.650697	-1.211348	0.428575	-1.113392	-0.147373	1.500526	-0.172474	-0.549055	0.952454	-0.672452
69	-0.526880	-0.616654	1.150883	-1.377578	-0.104589	-1.977014	-1.113392	-0.693482	-0.696729	-0.157563	-0.699090	-0.460532	-0.004805

علت داده‌هایی که به غلط پیش‌بینی شده اند می‌تواند ناشی از داده‌های آموزشی کم، داده‌های آموزشی ناهمگن، پارامترهای نامناسب مدل، انتخاب مدل نامناسب و ... باشد.

در این مسئله علت اصلی پیش‌بینی غلط و پایین بودن دقت مدل عدم پراکندگی خروجی محبوبیت می‌باشد. اگر به داده‌های سوال دقت شود بیشترین خروجی را بازه یا گروه 4 که شامل محبوبیت 60-80 می‌باشد تشکیل داده اند در صورتی که ممکن است داده‌هایی که برای تست به صورت تصادفی جدا می‌شوند خروجی در این بازه قرار نداشته باشند و این موضوع باعث کاهش دقت می‌شود.

برای حل این مشکل می‌توان از روش **k-fold cross validation** برای انتخاب داده‌های تست، تمرین و ارزیابی استفاده کرد که این موضوع دقت را به میزان قابل توجهی افزایش خواهد داد.

همچنین ممکن است مدل‌های انتخاب شده برای تربیت این مدل مناسب نبوده باشند. لذا می‌توان از مدل‌های پیچیده‌تری مانند شبکه‌های عصبی یا شبکه‌های عصبی عمیق استفاده کرد.

ح) این بار ستون محبوبیت را نادیده گرفته و بجای آن ستون گونه را در نظر بگیرید. در این ستون پر تکرارترین و کم تکرارترین گونه را پیدا کنید. طبیعتاً هر آهنگ میتواند همزمان دارای گونه‌های مختلفی باشد. بیشترین تعداد گونه مربوط به یک آهنگ را پیدا کنید. اگر آهنگی برای مثال در دو گونه hip hop و pop به صورت همزمان قرار گرفته باشد، درصد احتمال قرار گیری در ژانر R&B را محاسبه کنید. (راهنمایی: از قانون بیز استفاده کنید).

```
{'pop': 0, 'Folk/Acoustic': 1, 'rock': 2, 'jazz': 3, 'hip hop': 4, 'classical': 5, 'Dance/Electronic': 6, 'World/Traditional': 7, 'metal': 8, 'latin': 9, 'country': 10, 'blues': 11, 'set()': 12, 'R&B': 13, 'easy listening': 14}
```

ماتریس new_genre که تعریف شد هر درایه آن شامل مقادیر صفر و یک بود. مثلاً اگر یک نمونه jazz می‌بود درایه شماره ۳ از این بردار یک و بقیه درایه‌ها صفر بودند.

calculating iteration of each genre

```
In [22]: ┆ column_sums = new_genre1.sum()  
          print(column_sums)
```

```
0      1633  
1        20  
2      234  
3        2  
4      778  
5        1  
6      390  
7      10  
8      66  
9      64  
10     21  
11     4  
12     22  
13     452  
14      7  
dtype: int64
```

```
In [23]: import pandas as pd

genre_counts = musics_df4['new_genre'].value_counts()

most_common_genre = genre_counts.index[0]
least_common_genre = genre_counts.index[-1]

print("Most common genre:", most_common_genre)
print("Least common genre:", least_common_genre)
```

Most common genre: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Least common genre: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

برای اینکه پر تکرار ترین و کم تکرار ترین گونه پیدا شوند از ماتریس new_genre1 که همان بردار های

`new_genre` پیش از تبدیل به فرم برداری می باشد استفاده می شود. از آنجایی که در این ماتریس ستون ها شامل `genre` های متفاوت می باشند و در هر نمونه درایه ستون `genre` آن برابر یک می باشد لذا اگر مقادیر هر ستون با یکدیگر جمع شوند در پایان ستونی که بیشترین مقدار یک را نشان می داد حاصل جمع آن از همه بیشتر است و آن ستون نشان دهنده پر تکرار ترین `genre` می باشد. به عبارت دیگر مشاهده می شود که کدام تکرار عدد یک در بین ستون ها یا همان ژانر ها بیشتر از بقیه بوده است.

طبعتا برای پیدا کردن کم تکرار ترین `genre` نیز ستونی که جمع مقادیر آن از همه کمتر باشد یعنی `genre` مربوط به آن کمتر از سایر تکرار شده است.

همانطور که مشاهده می شود پر تکرار ترین ستون مربوط به ستون شماره 0 از ماتریس `new_genre1` می باشد که مطابق اختصاص اعداد به هر `genre` این ستون معرف `pop` است. یعنی ژانر `pop` به صورت تکی یا هنگامی که همراه با یک یا چند ژانر دیگر آمده است پر تکرار ترین ژانر می باشد. همانطور که مشاهده می شود این ژانر 1633 بار به صورت تکی یا مشترک مورد استفاده قرار گرفته است. همچنین کم تکرار ترین ستون مربوط به ستون 5 است که مطابق اختصاص اعداد این ستون معرف `classical` می باشد یعنی ژانر `classical` کم تکرار ترین ژانر می باشد. این ژانر تنها یک بار مورد استفاده قرار گرفته است و لذا کم تکرار ترین ژانر می باشد.

Most common genre: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 Least common genre: [0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]

همچنین بیشترین گونه مربوط به یک آهنگ مربوط به زمانی است که فقط ژانر آن `pop` باشد و کمترین گونه مربوط به یک آهنگ مربوط به زمانی است که ژانر آن تلفیقی از `hip-hop` و `country` باشد که خب منطقی به نظر می رسد.

قسمت بعدی این سوال باید با استفاده از قانون بیز در حالتی که یک موزیک در ژانر `hip-hop`, `pop`, `R&B` قرار دارد احتمال قرار داشتن آن در ژانر `R&B` محاسبه شود.

قانون بیز:

$$\frac{\text{احتمال رخداد دسته } \# \text{ از میان کل دسته ها}^{\#} \text{ (احتمال رخداد } y \text{ برای ورودی } X^{\#})}{\text{احتمال رخداد } X \text{ از میان همه ورودی ها}} = \text{احتمال تعلق ورودی } X \text{ به دسته } \#$$

calculate probability according to naive bayesian

```
In [29]: count_h_p = ((new_genre1.iloc[:, 0] == 1) & (new_genre1.iloc[:, 4] == 1)).sum()
count_R = (new_genre1.iloc[:, 13] == 1).sum()
count_R_h_p = ((new_genre1.iloc[:, 13] == 1) & (new_genre1.iloc[:, 0] == 1) & (new_genre1.iloc[:, 4] == 1)).sum()

P_h_p_given_R = count_R_h_p / count_R
P_R = count_R / len(new_genre1)
P_h_p = count_h_p / len(new_genre1)

P_h_p_given_R = P_h_p_given_R * P_R / P_h_p

print('probability that if it was in the pop and hip-hop category, it would also be in R&B:', P_h_p_given_R)
```

probability that if it was in the pop and hip-hop category, it would also be in R&B: 0.39619651347068147

برای استفاده از قانون بیز احتمالات مطابق زیر تعریف می شوند:

احتمال تعلق X^* به دسته y #

($P_{h_p_given_R}$) احتمال اینکه ورودی از نوع pop , hip hop از نوع R&B نیز باشد.

احتمال رخداد X^* برای ورودی y #

($P_{h_p_given_R}$) احتمال اینکه یک ورودی شامل pop , hip hop نیز باشد

احتمال رخداد y از میان کل دسته ها:

(P_R) احتمال اینکه در میان کل دسته ها ورودی از نوع R&B باشد.

احتمال رخداد X^* از میان همه ورودی ها:

(P_{h_p}) احتمال اینکه ورودی از نوع Pop و hip hop باشد.

probablity that if it was in the pop and hip-hop category, it would also be in R&B: 0.39619651347068147

بنابر این احتمال تعلق X^* (R&B) به دسته $y\#$ (pop, hip hop) مطابق محاسبات برابر با 0.3962 می باشد که مطابق قانون بیز احتمالات کم تر از 0.5 صفر در نظر گرفته می شوند یعنی برای ورودی جدید طبق قانون بیز اگر ورودی شامل pop , hip hop باشد R&B نخواهد بود.

باتوجه به اینکه این احتمالات با استفاده از ستون new_genre1 و اعداد اختصاص یافته به هر ژانر انجام شده است و اینکه در هر بار اجرای کد اعداد اختصاص یافته به ژانر های pop, hip hop , R&B متفاوت می باشد باید در سه خط اول برای محاسبه احتمالات آن ها جایگذاری شود. مثلا در اینجا چون pop ستون شماره 0 ماتریس new_genre است برای محاسبه احتمال آن در خط اول شماره ستون آن یعنی 0 جایگذاری شده است ولی در صورت اجرای مجدد کد این عدد امکان دارد تغییر کند و محاسبات دچار خطا شود.

(احتمال اشتباه یا به صورت nan نشان داده می شود)

ت) امتیازی: برای پیش‌بینی گونه موسیقی، دادگان را مانند قسمت (۵) به ۳ بخش تقسیم کنید. پس از تغییر و تبدیل دادگان، با استفاده از الگوریتم درخت تصمیم‌گیری مدل خود را تربیت کنید و در پایان ماتریس سردرگمی را تشکیل داده و تحلیل کنید.

define output and attach labels

```
In [31]: M from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
musics_df5 = musics_df
musics_df5.genre = le.fit_transform(musics_df5.genre)
musics_df5 = musics_df5.drop(['new_genre', 'popularity'], axis=1)
musics_df5
```

	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	genre
0	211160	0	0.751	0.834	1	-5.444	0	0.0437	0.3000	0.000018	0.3550	0.894	95.053	30
1	167066	0	0.434	0.897	0	-4.918	1	0.0488	0.0103	0.000000	0.6120	0.684	148.726	54
2	250546	0	0.529	0.496	7	-9.007	1	0.0290	0.1730	0.000000	0.2510	0.278	136.859	36
3	224493	0	0.551	0.913	0	-4.063	0	0.0466	0.0263	0.000013	0.3470	0.544	119.992	53
4	200560	0	0.614	0.928	8	-4.806	0	0.0516	0.0408	0.001040	0.0845	0.879	172.656	30
...
1995	181026	0	0.842	0.734	1	-5.065	0	0.0588	0.0427	0.000000	0.1060	0.952	137.958	30
1996	178426	0	0.552	0.702	9	-5.707	1	0.1570	0.1170	0.000021	0.1050	0.564	169.994	30
1997	200593	0	0.847	0.678	9	-8.635	1	0.1090	0.0669	0.000000	0.2740	0.811	97.984	17
1998	171029	0	0.741	0.520	8	-7.513	1	0.0656	0.4500	0.000002	0.2220	0.347	102.998	30
1999	215280	0	0.695	0.762	0	-3.497	1	0.0395	0.1920	0.002440	0.0863	0.553	120.042	14

2000 rows × 14 columns

با توجه به اینکه پس از اعمال one hot encoding درایه های ستون genre به صورت بردار تعریف شد و امکان تعریف خروجی به صورت بردار 15 درایه ای وجود ندارد و همچنین مدل را دچار مشکل میکند برای تربیت مدل با استفاده از خروجی genre در این قسمت به هر کدام از داده های خروجی با استفاده از دستور `LabelEncoder` یک label زده می شود. اگرچه در این روش به دلیل اینکه خروجی ها زیاد هستند هم خطای زیاد خواهد بود ولی از حالتی که خروجی به عنوان برداری از صفر و یک ها باشد بهتر است.

define new y and separating data

```
In [35]: M musics_df5['genre'] = musics_df5['genre'].astype('float')
y2 = np.asarray(musics_df5['genre'])
y2[0:20]
```

```
Out[35]: array([30., 54., 36., 53., 30., 21., 14., 40., 33., 0., 30., 30., 30.,
       54., 31., 21., 30., 30., 30., 14.])
```

```
In [38]: M from sklearn.model_selection import train_test_split

X_train, X_test, y2_train, y2_test = train_test_split(X, y2, test_size=0.1)
X_train, X_val, y2_train, y2_val = train_test_split(X_train, y2_train, test_size=0.1111)
print ('Train set:', X_train.shape, y2_train.shape)
print ('Test set:', X_test.shape, y2_test.shape)
print ('Val set:', X_val.shape, y2_val.shape)
```

```
Train set: (1600, 13) (1600,)
Test set: (200, 13) (200,)
Val set: (200, 13) (200,)
```

داده ها به سه بخش train, test , validation تقسیم می شوند.

```
In [82]: ┌─▶ from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import confusion_matrix

      model = DecisionTreeClassifier(max_depth=100, min_samples_split=50)
      model.fit(X_train, y2_train)

      y2_pred = model.predict(X_test)
      cm = confusion_matrix(y2_test, y2_pred)
      accuracy = np.trace(cm) / np.sum(cm)

      print('Confusion Matrix:\n', cm)
      print("accuracy", accuracy)
```

Confusion Matrix:

$$\begin{bmatrix} [1 & 0 & 0 \dots & 0 & 0 & 0] \\ [0 & 0 & 0 \dots & 0 & 0 & 0] \\ [0 & 0 & 0 \dots & 0 & 0 & 0] \\ \dots \\ [0 & 0 & 0 \dots & 0 & 0 & 0] \\ [0 & 0 & 0 \dots & 0 & 0 & 0] \\ [0 & 0 & 0 \dots & 0 & 0 & 0] \end{bmatrix}$$

accuracy 0.315

همانگونه که پیش بینی می شد در این مدل با توجه به اینکه به خروجی ها `label` زده شد و تعداد حالت های خروجی زیاد شد دقت مدل به اندازه قابل توجهی کاهش یافته است. همچنین ماتریس سردرگمی با توجه به این که 25 حالت برای خروجی `label` زده شده است و به عبارتی 25 خروجی متفاوت تعریف شده است یک ماتریس 25×25 خواهد بود که با این روش تربیت مدل خطای زیادی دارد.

باتوجه به پیچیدگی داده ها و ابعاد زیاد ماتریس سردرگمی برای تربیت این مدل سایر روش ها از جمله شبکه های عصبی روش مناسب تری می باشند.