

# بە نام خداوند بخشنده‌ی مهربان



عنوان پژوهش و گزارش:

(بیماری‌های قلبی UCL)

نگارش:

نگار درویشی

۱۴۰۰ فروردین

## پکیده

امروزه بیماری های قلبی بسیار رایج و یکی از دلایل اصلی مرگ و میر به شمار می رود. تشخیص درست و به موقع این بیماری بسیار حائز اهمیت است.

بر اساس گزارش سازمان بهداشت جهانی ۴۱,۳ درصد کل مرگ های سال ۲۰۰۵ در ایران ناشی از بیماری های قلبی عروقی بوده است. و با تاسف پیش بینی می شود تا سال ۲۰۳۰ این میزان به ۴۴,۸ درصد برسد.

ما تصمیم گرفته ایم تا با استفاده از یک مجموعه داده ارائه شده توسط UCL ، در شناسایی وجود بیماری قلب تمرکز کنیم. در مرحله اول ، ما مجموعه داده های خود را با استفاده از روش های مختلف استخراج ویژگی ها تجزیه و تحلیل کرده ایم.

## فهرست مطالب

۱. آیا داده پرت در دیتاست وجود دارد؟ در صورت وجود آنها را حذف کنید ..... ۲۴
۲. بررسی کنید آیا تعداد نمونه ها در هر کلاس متوازن است؟ ..... ۲۵
۳. نمونه های موجود در دیتاست را با نسبت ۸۰ به ۲۰ به دو بخش داده های آموزشی و داده های تست تقسیم بندی کنید ..... ۲۹
۴. قضیه بیز را در حداقل یک پاراگراف بیان کنید . سپس دسته بند های Gaussian Bernoulli و Naive Bayes ، Multinomial Naive Bayes ، Naive Bayes را با یکدیگر مقایسه کنید و بیان کنید هر کدام از این دسته بندها بیشتر در کجا کاربرد دارند؟ ..... ۳۰
۵. با در نظر گرفتن فیچر ها target ، chol ، trestbps و lage ..... یک دسته بند Bayes را از پایه پیاده سازی کنید ..... ۳۸
۶. پس پیاده سازی Bayes Naive Gaussian و آموزش آن بر روی داده های آموزشی ( ۸۰ درصد دیتاست) . نتایج را برای داده های تست ( ۲۰ درصد باقی دیتاست) بررسی کنید ..... ۳۹
۷. با استفاده از پکیج GaussianNB و sklearn یک مدل بسازید و بر روی داده های آموزشی ، ترین کنید ..... ۴۰

۷. بررسی کنید که در سه معیار مطرح شده مدلی که با استفاده از پکیج ساخته اید و مدلی که خود پیاده سازی کرده اید به چه صورتی عمل کرده اند..... ۴۱
۸. کلاسیفایر **SVM** را با استفاده از پکیج **sklearn** بر سه فیچر مطرح شده در سوال ۴ با استفاده از داده های آموزشی ترین کنید..... ۴۱
۹. حداقل دو حالت مختلف را برای کرنل در **SVM** ساخته شده با پکیج در نظر بگیرید..... ۴۳
۱۰. دسته بند **SVM** را با استفاده از پکیج **sklearn** بسازید و با در نظر گرفتن کلیه فیچرهای دیتابست بر روی داده های آموزشی ترین کنید..... ۴۷
۱۱. برای سوال ۱۰ ، یکبار مدل را با **Validation Cross fold-5** اجرا کنید..... ۴۸
۱۲. با استفاده از پکیج **sklearn** دسته بند را **NN-K ۲** را بسازید ..... ۴۹
۱۳. بررسی کنید در سوال ۱۲ تعداد همسایه ها **K** چه نقشی ایفا میکند ؟ زیاد شدن همسایه ها خوب است ؟ چگونه میتوان مشخص کرد چه تعداد همسایه برای مسئله ما مناسب است؟..... ۵۱

۱۴. در سوال ۱۲ به جای استفاده از تمامی فیچرها فقط از سه فیچر `thalach`، `trestbps`، `chol` استفاده کنید و مدل را بسازید.....

۵۴.....

۱۵. تفاوت بین روش های کلاس بندی پارامتری و غیرپارامتری.....

۱۶. معیار **Matthews Correlation Coefficient** (MCC) چیست و در چه جاهایی استفاده میشود؟.....

۶۰.....

۶۳.....

۶۳.....

## مقدمه

در ابتدای قرن بیستم ۱۰ درصد کل مرگ و میرها به علت بیماری های قلبی عروقی بود. در انتهای همین قرن موارد مرگ و میر ناشی از بیماری های قلبی به ۲۵ درصد افزایش یافت و پیش بینی می شود با توجه به روند کنونی تا سال ۲۰۲۵ میلادی بیشتر از ۳۵ تا ۶۰ درصد موارد مرگ و میر در جهان از بیماری های قلبی عروقی ناشی شود. با توجه به شیوع و سهمی که بیماری های قلبی در مرگ و میر انسان ها دارند در این پژوهش اطلاعات مربوط به بیماری های قلبی با استفاده از تکنیک های داده کاوی مورد تحلیل قرار گرفته است تا بتوان با استفاده از این اطلاعات به مدل و الگوی دقیق تری جهت پیش بینی بیماری قلبی دست یافت.

## استخراج و تجسم ویژگی های داده ها

### مجموعه داده و شرح مسئله

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from scipy.linalg import svd

pd.options.display.max_columns = 50

df = pd.read_csv('heart.csv')

```

در اینجا ، ما نمونه ای از شکل ظاهری داده ها را نشان خواهیم داد:

```
df.head()
```

```
<style scoped>.dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
  vertical-align: top;
}
```

```
.dataframe thead th {
  text-align: right;
}
```

```
</style>
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

### ارزیابی مجموعه داده و کیفیت داده

ما نحوه تفسیر انواع ویژگیها در مجموعه

داده توسط **pandas** را بررسی خواهیم

کرد:

```
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age            303 non-null int64
sex            303 non-null int64
cp             303 non-null int64
trestbps       303 non-null int64
chol           303 non-null int64
fb              303 non-null int64
restecg         303 non-null int64
thalach         303 non-null int64
exang           303 non-null int64
oldpeak         303 non-null float64
slope           303 non-null int64
ca              303 non-null int64
thal            303 non-null int64
target          303 non-null int64

```

**pandas** در صفحه‌ی قبل، می‌توانیم ببینیم که

تمام ویژگی‌های ما را عددی تشخیص می‌دهد، چه اعداد صحیح و چه شناور. با این حال، این گمراه کننده است، زیرا ۸ ویژگی ما در واقع طبقه‌بندی شده‌اند و تعداد آنها نشان دهنده نگاشت‌های پدیده‌ای قلب انسان در دنیای واقعی است.

بنابراین، حتی اگر مقادیر این ویژگی‌ها را به صورت عددی دسته‌بندی کنیم، نمی‌توانیم آنها را به عنوان عددی در تحلیل خود بگنجانیم، از جمله برای جنسیت متغیر، نه ۰ و نه ۱ بهتر از دیگری است و نه متغیر نسبت طبقه‌بندی، زیرا مقدار ۰ فقط یک قرارداد داخلی در این مجموعه داده است که زنان را نشان می‌دهد. بنابراین، هرگونه محاسبه "عددی" با استفاده از این متغیر توسط توابع پاندا عملی ارزیابی می‌شود، در حالی که برای دنیای واقعی نادرست است.

```
df.describe()
```

```
<style scoped>.dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th { vertical-align: top; }
```

```
.dataframe thead th { text-align: right; }
```

```
</style>
```

	age	trestbps	chol	thalach	oldpeak	sex_male	cp_atypical	cp_non_anginal	cp_as
count	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.0
mean	54.523649	131.60473	247.155405	149.560811	1.059122	0.679054	0.165541	0.280405	0.476
std	9.059471	17.72662	51.977011	22.970792	1.166474	0.467631	0.372297	0.449958	0.500
min	29.000000	94.000000	126.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	48.000000	120.000000	211.000000	133.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	56.000000	130.000000	242.500000	152.500000	0.800000	1.000000	0.000000	0.000000	0.000
75%	61.000000	140.000000	275.250000	166.000000	1.650000	1.000000	0.000000	1.000000	1.000
max	77.000000	200.000000	564.000000	202.000000	6.200000	1.000000	1.000000	1.000000	1.000

</style>

cp_asymptomatic	fbs_true	restecg_st_t	restecg_hypertrophy	exang_yes	slope_flat	slope_downsloping	ca_1
296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000
0.476351	0.14527	0.496622	0.013514	0.327703	0.462838	0.466216	0.219595
0.500286	0.35297	0.500835	0.115655	0.470171	0.499461	0.499702	0.414673
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000	0.000000
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

</style>

ng_yes	slope_flat	slope_downsloping	ca_1	ca_2	ca_3	thal_fixed	thal_reversible	target
00000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000	296.000000
7703	0.462838	0.466216	0.219595	0.128378	0.067568	0.060811	0.388514	0.540541
0171	0.499461	0.499702	0.414673	0.335077	0.251427	0.239388	0.488238	0.499198
0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
0000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000
0000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

ما چهار تفاوت مهم در کیفیت داده را مشاهده می

کنیم:

با توجه به این منبع ، باید در ویژگی های **ca** و

مقادیر از دست رفته وجود داشته باشد ، که به

عنوان "؟" نشان داده می شود. با این حال ، به نظر

نمی رسد که در مجموعه داده های ما وجود داشته

باشد.

از آمار خلاصه ، می توان دریافت که **thal** مقادیری

بین ۰ تا ۳ می گیرد که اشتباه است ، زیرا در

توضیحات مجموعه داده ذکر شده است که فقط می

تواند مقادیر ۳ ، ۶ یا ۷ را بدست آورد.

از مجموعه داده ما مقادیری بین ۰ تا ۳ می گیرد ،

اما در توضیحات مجموعه داده باید مقادیر بین ۱ تا ۴

باشد.

شیب از مجموعه داده های ما مقادیر بین ۰ و ۲ را در

بر می گیرد ، اما در توضیحات مجموعه داده باید

مقادیر بین ۱ تا ۳ باشد.

ما تحقیق عمیق تری در این مورد انجام خواهیم داد:

```
df['ca'].value_counts()
```

0	175
1	65
2	38
3	20
4	5

Name: ca, dtype: int64

ما تازه دریافتیم که مقادیر گمشده برای ویژگی **ca** هنوز در مجموعه داده های ما وجود دارد، با این حال، به جای اینکه به عنوان '?' نشان داده شوند، آنها به عنوان مقادیری خارج از نگاشت های طبقه بندی شده توصیف شده در توصیف مجموعه داده شده اند.

در اینجا، مقادیر از دست رفته مقدار "4" را اشغال می کنند، که کیفیت داده را به خطر می اندازد، زیرا **ca** فقط می تواند مقادیر صحیح بین

- ۰ تا ۳ را بدست آورد.

از آنجا که فقط ۵ ورودی در مجموعه داده وجود دارد که دارای این خطای است، ما آنها را به طور کلی از مجموعه داده حذف خواهیم کرد.

```
df.drop(index = (df[df.ca == 4]).index, inplace = True)
```

برای مقادیر از دست رفته در **thal** نیز همین اتفاق می افتد: آنها به جای "?" به عنوان "۰" نشان داده می شوند. برای اینکه کیفیت مجموعه داده بالا حفظ شود، مجدداً تمام ورودی های دارای چنین مقادیر اشتباه را از مجموعه داده حذف خواهیم کرد.

با این وجود ، در اینجا یک مشکل دیگر نیز وجود دارد **thal** ، که قبلاً ذکر شد: نگاشتهای طبقه بندی شده **thal** کاملاً خارج از محدوده و درهم آمیخته هستند ، در مقایسه با آنچه انتظار می رود ( فقط مقادیر صحیح ۳ ، ۶ یا ۷ باید وجود داشته باشد) .

با استفاده از این منبع ، بار دیگر متوجه شدیم که داده ها خود به خطر نیفتاده اند و فقط لازم است نگاشت های آن دوباره انجام شود ، که در زیر انجام داده ایم:

```
df['thal'].value_counts()
```

```
2    163
3    115
1     18
0      2
Name: thal, dtype: int64
```

```
df.drop(index = (df[df.thal == 0]).index, inplace = True)
```

```
df.loc[df.thal == 1, 'thal'] = 6
df.loc[df.thal == 3, 'thal'] = 7
df.loc[df.thal == 2, 'thal'] = 3
```

حال ، اگر بررسی کنیم ، خواهیم دید که ویژگی **thal** از مشکلات کیفیت داده های خود پاک شده است و مطابق انتظار نشان داده می شود:

```
df['thal'].value_counts()
```

```
3    163
7    115
6     18
Name: thal, dtype: int64
```

برای سومین مسئله داده ، مقادیر **ca** را با

کدگذاری های طبقه بندی مناسب کد می کنیم:

```
df['cp'].value_counts()
```

```
0    141
2     83
1     49
3     23
Name: cp, dtype: int64
```

```
df.loc[df.cp == 0, 'cp'] = 4
```

```
df.loc[df.cp == 3, 'cp'] = 7
```

```
df.loc[df.cp == 2, 'cp'] = 3
df.loc[df.cp == 1, 'cp'] = 2
df.loc[df.cp == 7, 'cp'] = 1
```

ما همین کار را با رمزگذاری مقادیر شیب با توجه به کدگذاری های طبقه بندی مناسب انجام خواهیم داد:

```
df['slope'].value_counts()
```

```
2    138  
1    137  
0     21  
Name: slope, dtype: int64
```

```
df.loc[df.slope == 2, 'slope'] = 3  
df.loc[df.slope == 1, 'slope'] = 2  
df.loc[df.slope == 0, 'slope'] = 1
```

سرانجام ، ما بررسی خواهیم کرد که پس از حذف ورودی های حاوی داده های گمشده / اشتباه ، تعداد

ورودی های داده را ترک کرده ایم:

```
df.shape
```

```
(296, 14)
```

## تجزیه و تحلیل داده های اکتشافی (EDA) – تجسم

ابتدا لیستی از تمام اسامی ویژگی های عددی را ذخیره خواهیم کرد تا در نتایج تجزیه و تحلیل متغیرهای عددی با نتایج متغیرهای طبقه ای مخلوط نشوند.

```
numerical_columns = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

به طور خاص ، تجزیه و تحلیل PCA بسیار حساس

به موارد زیر است:

دور در صفات ، توزیع ویژگی توزیع نرمال یک متغیره

نیست ، متغیرهای غیر همبسته

اگر متغیرهای ما همبستگی بسیار ضعیفی داشته باشند ،

به خودی خود مشکلی ایجاد نمی کند ، با این حال ،

در این حالت ، PCA چندان مفید نیست ، زیرا روابط

بین داده ها واریانس داده ها را به خوبی نمی گیرد و

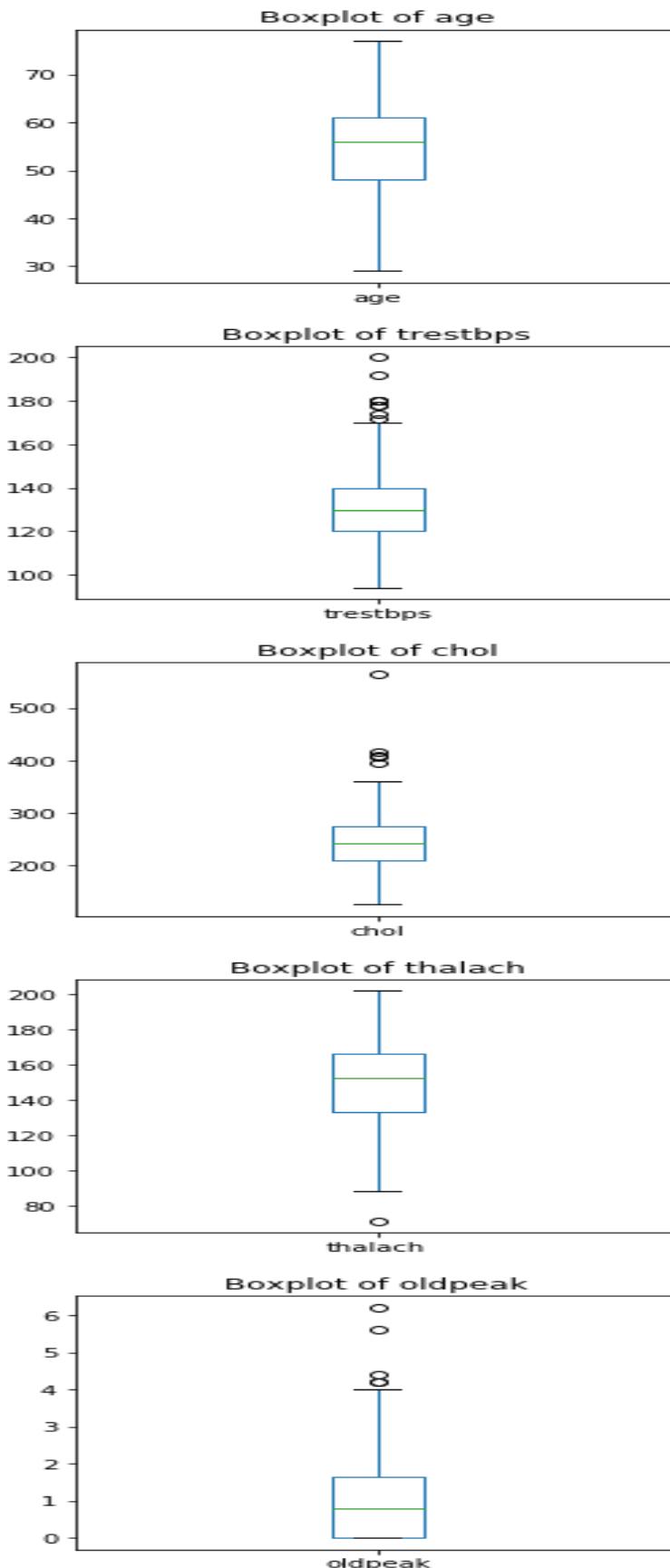
ما از پیچیدگی بسیار کم نخواهیم کرد

با انجام PCA ، زیرا برای اطمینان از نمایش مناسب داده های خود ، به هر حال ما باید از بیشتر اجزای PCA استفاده کنیم. قبل از انجام تجزیه و تحلیل PCA ، باید هنگام تجزیه و تحلیل داده ها به دنبال این دستورالعمل ها باشیم. در اینجا جعبه های جعبه ای برای هر ۵ متغیر عددی ایجاد شده است ، تا بینند کدام متغیرها دارای فاصله هستند و اندازه آنها چقدر است.

```
fig = plt.figure(figsize=(4, 23))
for i in range(5):
    # Creating a subplot placeholder corresponding to an attribute
    plt.subplot(5, 1, i+1)

    # Make a space between the different rows of plots
    plt.subplots_adjust(hspace = 0.25)

    df[numerical_columns[i]].plot(kind = 'box');
    plt.title('Boxplot of ' + str(numerical_columns[i]));
```

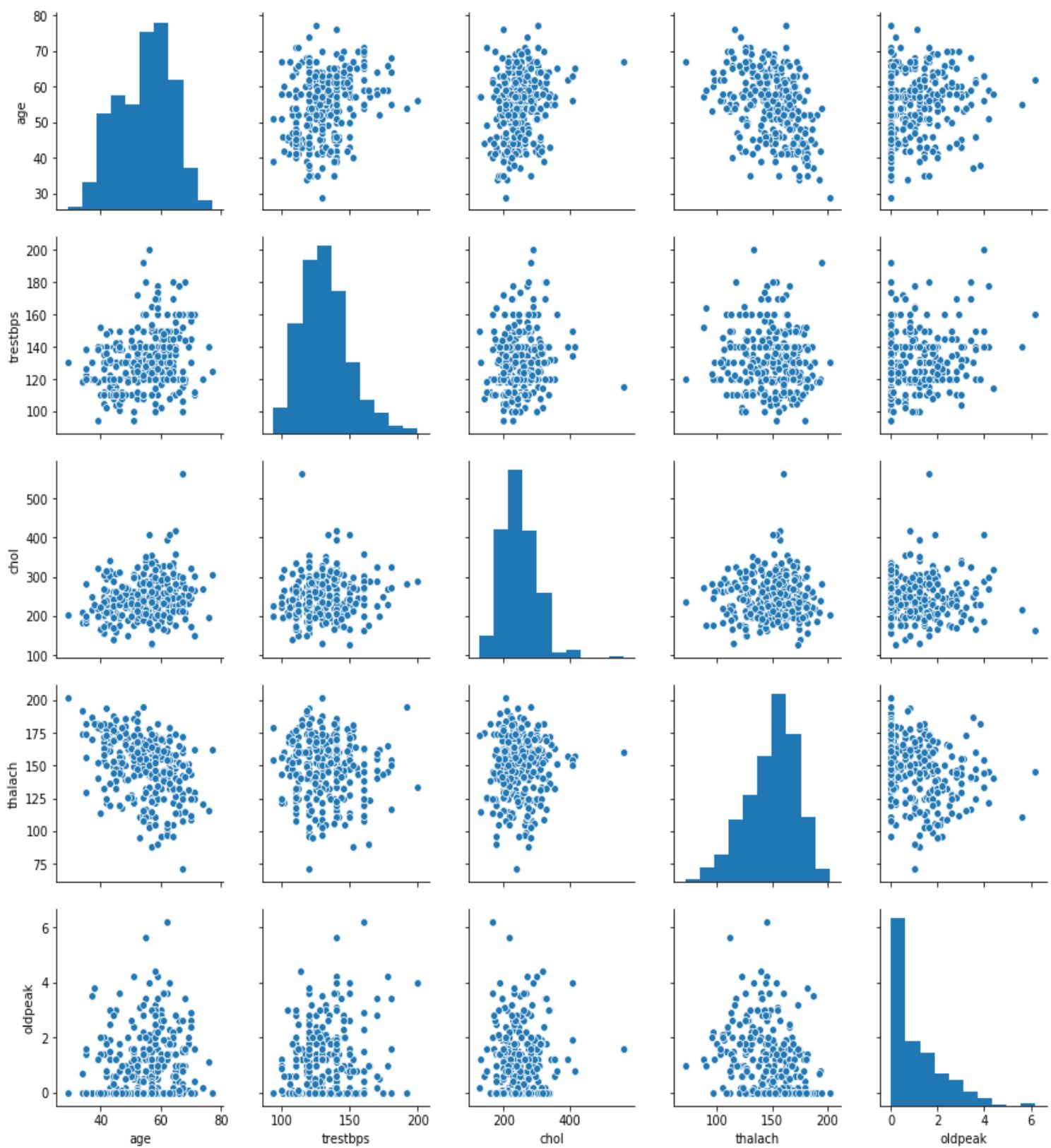


از این طرح ، نتیجه می گیریم که متغیرهای عددی ما ، به استثنای سن ، محدوده هایی را در خود نگه می دارند. این ممکن است هنگام تجسم خوشه های PCA مشکلاتی را ایجاد ، بنابراین بر کل هنجار Frobenius تسلط پیدا می کند و PCA به عوامل اصلی محرك اجزای مختلف تبدیل می شود.

برای اینکه بتوانیم به سرعت یک نمای کلی از توزیع های هیستوگرام متغیرهای عددی منفرد را ببینیم (برای کشف اینکه آیا صفات توزیع غیرمدادی و عادی دارند) ، و همچنین روابط زوجی بین آنها (برای کشف همبستگی های نهایی) ، جفت زیر را ترسیم کرده ایم طرح زیر:

```
grid = sb.pairplot(data = df.loc[:, numerical_columns]);
grid.fig.suptitle("Pair-by-pair interactions between features, and univariate distribution histograms", y = 1.02);
```

Pair-by-pair interactions between features, and univariate distribution histograms



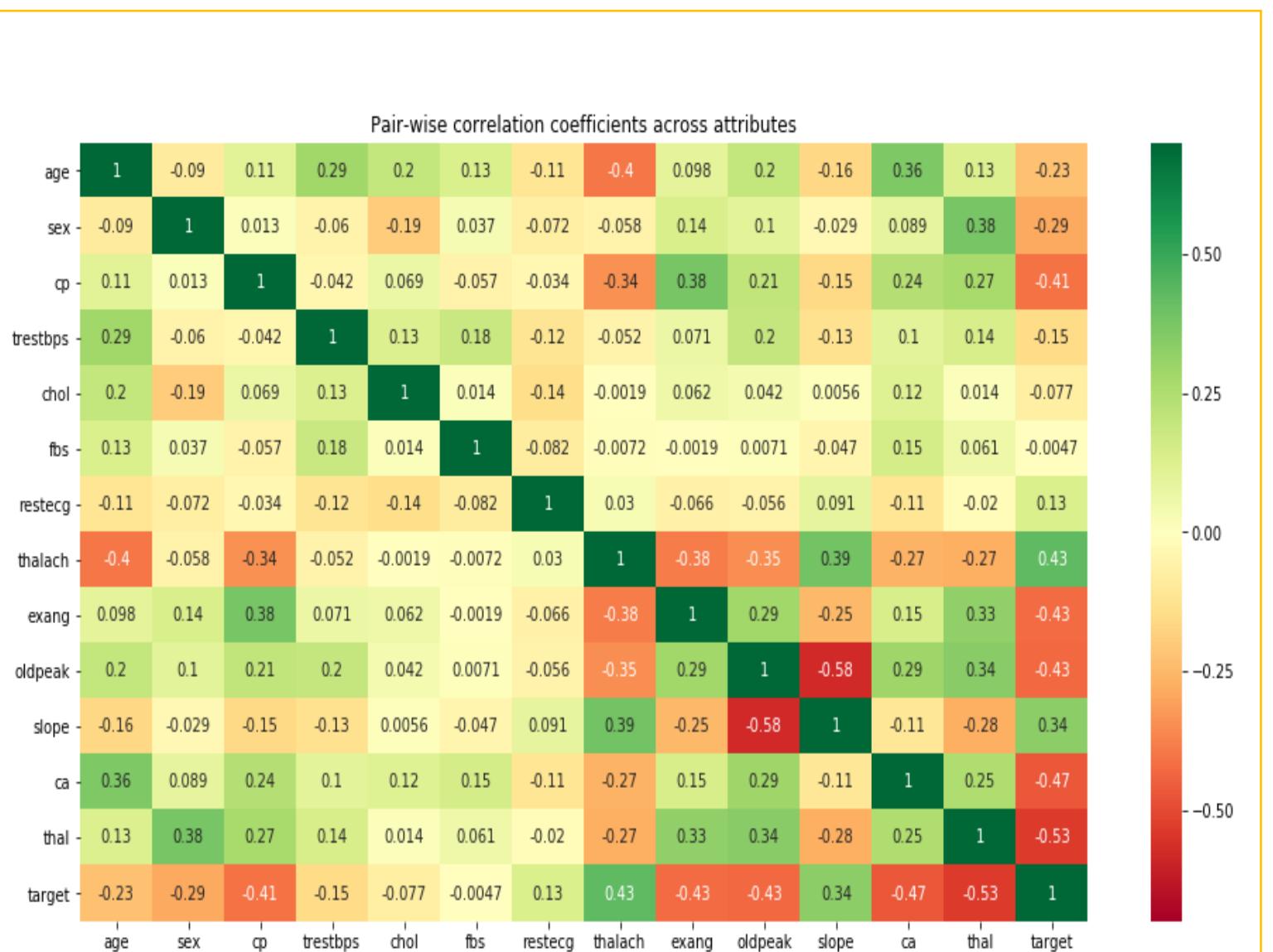
از اینجا ، می بینیم که به نظر نمی رسد همه ویژگی های عددی ما در یک ساختار توزیع نرمال توزیع شده اند. به نظر می رسد بیشتر آنها کج هستند (از هیستو گرام ها می توان فهمید) ، حتی **oldpeak** ساختاری مشابه توزیع **gama** دارد.

با رفتن بیشتر ، ما به ضرایب همبستگی زوجی بین همه ویژگی هایمان علاقه مند هستیم ، بنابراین برای اینکه ببینیم آیا از انجام تجزیه و تحلیل **PCA** سود حاصل می شود ، از آنجا که همبستگی زیاد بین برخی از مولفه ها به معنی خوب بودن این دو مولفه است.

هر همبستگی فراتر از  $|0.3|$  (یا  $+/-$ ) را می توان همبستگی متوسطی دانست.

```
plt.figure(figsize = [16, 8])

sb.heatmap(df.corr(), cmap = 'RdYlGn', annot = True, vmax = 0.7, vmin = -0.7);
plt.title('Pair-wise correlation coefficients across attributes');
```



ما می توانیم متوجه شویم که فقط تعداد کمی از ویژگی های ما با هم همبستگی

متوسطی دارند ، که بیشتر آنها ویژگی های طبقه ای با همان دامنه ممکن از مقادیر را

دارند.

## تحلیل مولفه اصلی (PCA) مجموعه داده

قسمت نهایی این تجزیه و تحلیل استفاده از تجزیه و تحلیل PCA بر روی این مجموعه داده ها خواهد بود.

ما باید با توجه به نیازهای PCA چالش های احتمالی مجموعه داده خود را یادآوری کنیم:

برخی از ویژگی های ما حاوی نقاط پرت است هیچ یک از ویژگی های عددی ما توزیع نمی شود به نظر نمی رسد که بین اکثر ویژگی های ما رابطه قوی یا متوسطی وجود داشته باشد. صرف نظر از این ، ما معتقدیم که تجزیه و تحلیل PCA می تواند در تعیین اهمیت متغیرهای ما در توضیح واریانس درون مجموعه داده مفید باشد. ما علاقه مندیم ببینیم آیا از طریق چنین تحولی می توان به کاهش عمدۀ ابعاد دست یافت.

قبل از ادامه تحلیل ، باید توجه داشته باشیم که بسیاری از متغیرهای ما ، حتی اگر عددی به نظر برسند ، در واقع متغیرهای طبقه ای با نقشه برداری عددي هستند. برای اینکه PCA نتایج دقیقی را ارائه دهد ، ما یک کدگذاری خارج از K را بر روی هر یک از این متغیرها انجام خواهیم داد:

```
df.head()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {  
    vertical-align: top;  
}
```

```
.dataframe thead th {  
    text-align: right;  
}
```

```
</style>
```

	age	sex	cp	trestbps	chol	fbp	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	0	150	0	2.3	1	0	6	1
1	37	1	3	130	250	0	1	187	0	3.5	1	0	3	1
2	41	0	2	130	204	0	0	172	0	1.4	3	0	3	1
3	56	1	2	120	236	0	1	178	0	0.8	3	0	3	1
4	57	0	4	120	354	0	1	163	1	0.6	3	0	3	1

```
df['sex_male'] = df.sex  
df.drop(columns = 'sex', inplace = True)
```

```
df = pd.get_dummies(data = df, columns = ['cp'], drop_first=True)  
df.rename({'cp_2': 'cp_atypical', 'cp_3' : 'cp_non_anginal', 'cp_4': 'cp_asymptomatic'}, axis = 'columns', inplace = True)
```

```
df['fbs_true'] = df.fbs  
df.drop(columns = 'fbs', inplace = True)
```

```
df = pd.get_dummies(data = df, columns = ['restecg'], drop_first=True)  
df.rename({'restecg_1': 'restecg_st_t', 'restecg_2' : 'restecg_hypertrophy'}, axis = 'columns', inplace = True)
```

```
df['exang_yes'] = df.exang  
df.drop(columns = 'exang', inplace = True)
```

```
df = pd.get_dummies(data = df, columns = ['slope'], drop_first=True)  
df.rename({'slope_2': 'slope_flat', 'slope_3' : 'slope_downsloping'}, axis = 'columns', inplace = True)
```

```
df = pd.get_dummies(data = df, columns = ['ca'], drop_first=True)
```

```
df = pd.get_dummies(data = df, columns = ['thal'], drop_first=True)  
df.rename({'thal_6': 'thal_fixed', 'thal_7' : 'thal_reversible'}, axis = 'columns', inplace = True)
```

```
df['target_true'] = df.target  
df.drop(columns = 'target', inplace = True)  
df.rename({'target_true': 'target'}, axis = 'columns', inplace = True)
```

در اینجا مشاهده می شود که همه متغیرهای طبقه

بندی شده ما به درستی کدگذاری می شوند ، به

ستونهایی که اخیراً ایجاد شده اند نامهای

پیشنهادی داده می شوند ، تا معانی ویژگی را

بهتر تشخیص دهنند:

```
df.head()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {  
    vertical-align: top;  
}  
  
.dataframe thead th {  
    text-align: right;  
}
```

```
</style>
```

	age	trestbps	chol	thalach	oldpeak	sex_male	cp_atypical	cp_non_anginal	cp_asymptomatic	fbs_true	res
0	63	145	233	150	2.3	1	0	0	0	1	0
1	37	130	250	187	3.5	1	0	1	0	0	1
2	41	130	204	172	1.4	0	1	0	0	0	0
3	56	120	236	178	0.8	1	1	0	0	0	1
4	57	120	354	163	0.6	0	0	0	1	0	1

```
</style>
```

	restecg_st_t	restecg_hypertrophy	exang_yes	slope_flat	slope_downsloping	ca_1	ca_2	ca_3	thal_fixed	thal_rever
0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0

```
</style>
```

	restecg_hypertrophy	exang_yes	slope_flat	slope_downsloping	ca_1	ca_2	ca_3	thal_fixed	thal_reversible	target
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	0	0	1

به جای ۱۳ ویژگی و ۱ متغیر طبقه بندی مانند قبل ،

اکنون تعداد ویژگی ها را به ۲۰ افزایش داده ایم:

`df.shape`

(296, 21)

برای انجام تجزیه و تحلیل PCA با سهولت تر ، ما ویژگی های مجموعه داده را در یک ماتریس داده  $X$  کد

پردار لا ذخیرہ میں شود:

```
raw_data = df.get_values()
```

```
# Notice that raw_data both contains the information we want to store in a matrix X and the information that we wish to store  
# in y (the class labels, whether the patient has a heart disease or not).
```

```
print(raw_data)
```

```

[[ 63. 145. 233. ... 1. 0. 1.]
 [ 37. 130. 250. ... 0. 0. 1.]
 [ 41. 130. 204. ... 0. 0. 1.]
 ...
 [ 68. 144. 193. ... 0. 1. 0.]
 [ 57. 130. 131. ... 0. 1. 0.]
 [ 57. 138. 236. ... 0. 0. 0.]

```

```
# We start by making the data matrix X by indexing into data.  
cols = range(0, len(df.columns) - 1)
```

```
print(cols)
```

```
range(0, 20)
```

```
X = raw_data[:, cols]
```

```
print(x)
```

```

[[ 63. 145. 233. ... 0. 1. 0.]
 [ 37. 130. 250. ... 0. 0. 0.]
 [ 41. 130. 204. ... 0. 0. 0.]
 ...
 [[ 68. 144. 193. ... 0. 0. 1.]
 [ 57. 130. 131. ... 0. 0. 1.]
 [ 57. 130. 236. ... 0. 0. 0.]]

```

```
# We can extract the attribute names that came from the columns  
attributeNames = np.asarray(df.columns[cols])
```

```
print(attributeNames)
```

```
[ 'age' 'trestbps' 'chol' 'thalach' 'oldpeak' 'sex_male' 'cp_atypical'  
 'cp_non_anginal' 'cp_asymptomatic' 'fbs_true' 'restecg_st_t'  
 'restecg_hypertrophy' 'exang_yes' 'slope_flat' 'slope_downsloping' 'ca_1'  
 'ca_2' 'ca_3' 'thal_fixed' 'thal_reversible' ]
```

```
# The class labels are numerical already, and can be directly imported into y
classLabels = raw_data[:, -1]
classNames = np.unique(classLabels)
```

```
# This is the class index vector y:  
y = class_labels.astype(int)
```

```
print(v)
```

# We can determine the number of data objects and number of attributes using the shape of X  
**N, M = X.shape**

```
# Finally, the last variable that we need to have the dataset in the  
# "standard representation" for the course, is the number of classes, C:  
C = len(classNames)
```

اکنون که داده های مربوط به بیماری های قلبی را به صورت نمایش داده های استاندارد قالب بندی کرده ایم ، باید آن را نرمال سازی و با نمره  $Z$  استاندارد کنیم ،

پس از آن نمایشی از تجزیه و تحلیل PCA خود در زیر خواهیم داشت:

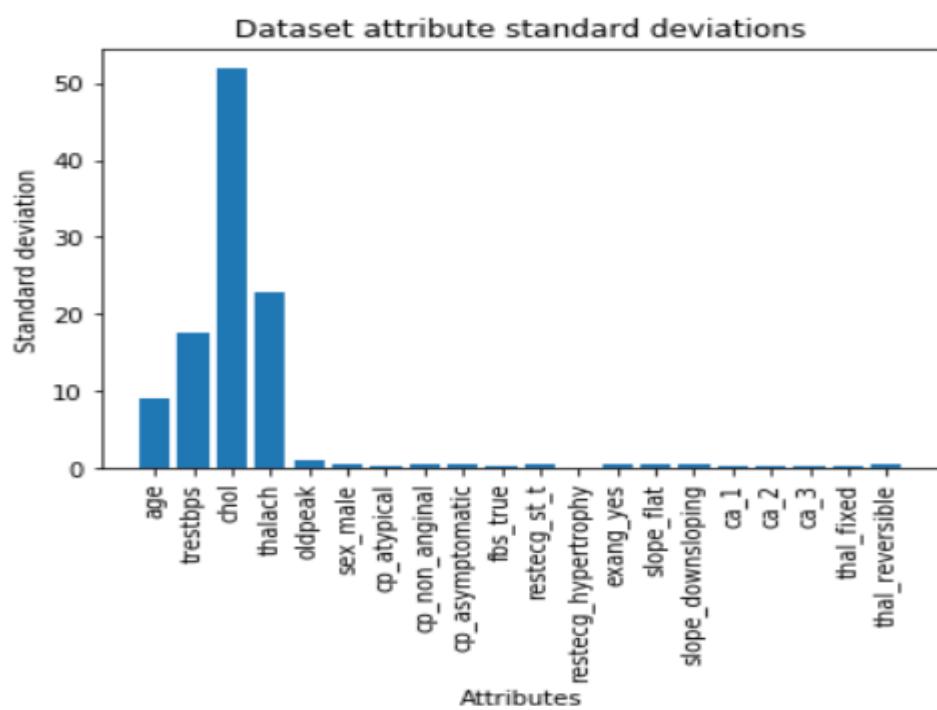
```
range_ = np.arange(1, X.shape[1] + 1)

plt.bar(range_, np.std(X, axis = 0));

plt.xticks(range_, attributeNames);
plt.xticks(rotation = 90);
plt.xlabel('Attributes');

plt.ylabel('Standard deviation');

plt.title('Dataset attribute standard deviations');
```



به وضوح می توان دریافت که در مقیاس اختلاف زیادی با صفات عددی وجود دارد ،

در مقایسه با صفات طبقه بندی. مقیاس آنها حتی در میان ویژگی های عددی نیز بسیار

متفاوت است.

بنابراین ، اگر می خواهیم یک نمایش داده بی طرفانه

داشته باشیم ، باید قبل از انجام PCA ، کل مجموعه

داده را استاندارد کنیم:

```
## Investigate how standardization affects PCA
```

```
# Subtract the mean from the data
```

```
Y1 = X - np.ones(shape = (N, 1))*X.mean(axis = 0) # Y normalized
```

```
# Subtract the mean from the data and divide by the attribute standard
# deviation to obtain a standardized dataset:
```

```
Y2 = X - np.ones(shape = (N, 1))*X.mean(axis = 0)
```

```
Y2 = Y2*(1/np.std(Y2,0)) # Y normalized and standardized
```

```
# Store the two in a cell, so we can just loop over them:
```

```
Ys = [Y1, Y2]
```

```
titles = ['Zero-mean', 'Zero-mean and unit variance']
```

```
threshold = 0.9
```

```
# Choose two PCs to plot (the projection)
```

```
i = 0 # PC1
```

```
j = 1 # PC2
```

```
for k in range(2):
```

```
# Obtain the PCA solution by calculate the SVD of either Y1 or Y2
```

```
U,S,V = svd(Ys[k], full_matrices=False)
```

```
V=V.T # For the direction of V to fit the convention in the course we transpose
```

```
# For visualization purposes, we flip the directionality of the
```

```
# principal directions such that the directions match for Y1 and Y2.
```

```
if k==1: V = -V; U = -U;
```

```
# Compute variance explained
```

```
rho = (S*S) / (S*S).sum()
```

```
# Compute the projection onto the principal components
```

```
Z = U*S;
```

```
# Make the plots
```

```
plt.figure(figsize=(10,15))
plt.subplots_adjust(hspace=.4)
```

```
nrows = 3
```

```
ncols = 2
```

```
for k in range(2):
```

```
# Obtain the PCA solution by calculate the SVD of either Y1 or Y2
U,S,V = svd(Ys[k], full_matrices=False)
```

```
V=V.T # For the direction of V to fit the convention in the course we transpose
```

```
# For visualization purposes, we flip the directionality of the
```

```
# principal directions such that the directions match for Y1 and Y2.
```

```
if k==1: V = -V; U = -U;
```

```
# Compute variance explained
```

```
rho = (S*S) / (S*S).sum()
```

```
# Compute the projection onto the principal components
Z = U*S;
```

```
# Plot projection
```

```
plt.subplot(nrows, ncols, 1+k)
```

```
C = len(classNames)
```

```
for c in range(C):
```

```
plt.plot(Z[y==c,i], Z[y==c,j], 'o', alpha = 0.7)
```

```
plt.xlabel('PC'+str(i+1))
```

```
plt.ylabel('PC'+str(j+1))
```

```
plt.title(titles[k] + '\n' + 'Projection' )
```

```
plt.legend(['No disease', 'Disease'])
```

```
plt.axis('equal')
```

```
# Plot attribute coefficients in principal component space
plt.subplot(nrows, ncols, 3+k)
```

```
for att in range(V.shape[1]):
```

```
plt.arrow(0,0, V[att,i], V[att,j])
```

```
plt.text(V[att,i], V[att,j], attributeNames[att])
```

```
plt.xlim([-1,1])
plt.ylim([-1,1])
```

```
plt.xlabel('PC'+str(i+1))
plt.ylabel('PC'+str(j+1))
plt.grid()
```

```
# Add a unit circle
```

```
plt.plot(np.cos(np.arange(0, 2*np.pi, 0.01)),
         np.sin(np.arange(0, 2*np.pi, 0.01)));
```

```
plt.title(titles[k] + '\n'+'Attribute coefficients')
```

```
plt.axis('equal')
```

```
# Plot cumulative variance explained
plt.subplot(nrows, ncols, 5+k);
```

```
plt.plot(range(1,len(rho)+1), rho, 'x-')
plt.plot(range(1,len(rho)+1), np.cumsum(rho), 'o-')
plt.plot([1,len(rho)], [threshold, threshold], 'k--')
```

```
plt.title('Variance explained by principal components');
```

```
plt.xlabel('Principal component');
```

```
plt.ylabel('Variance explained');
```

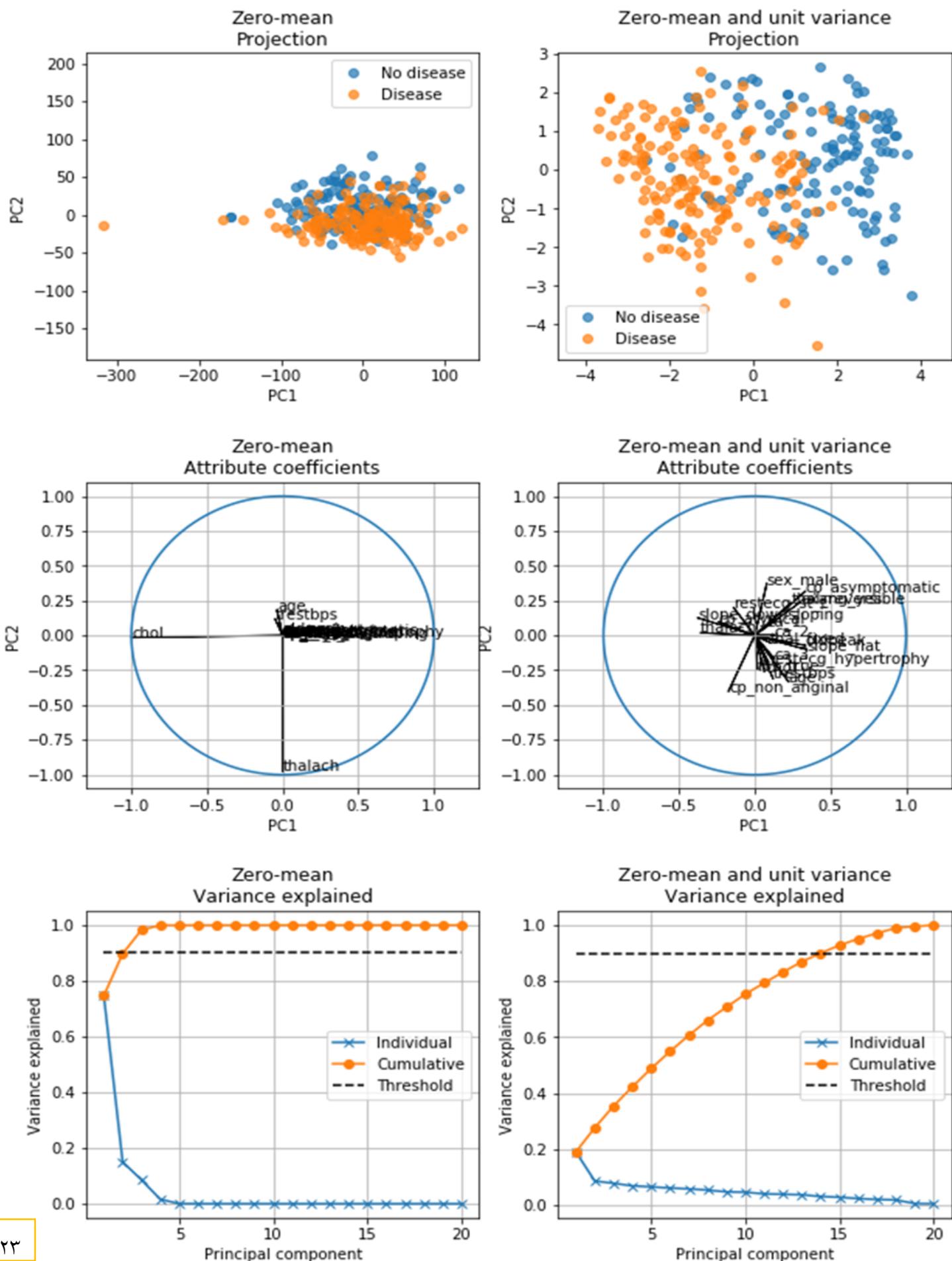
```
plt.legend(['Individual','Cumulative','Threshold'])
```

```
plt.grid()
```

```
plt.title(titles[k]+'\n'+'Variance explained')
```

```
plt.suptitle('Heart disease dataset: Effect of standardization', y = 0.935, fontsize = 13);
```

### Heart disease dataset: Effect of standardization



نتایج حاصل از تجزیه و تحلیل استاندارد PCA نشان می دهد که ، بدون استاندارد سازی ، خوشه ها در قسمت بزرگی از مساحت کل با هم همپوشانی دارند.

با این حال ، هنگامی که استاندارد سازی می شود ، دو مولفه اصلی تفکیک واضح داده ها را نشان می دهد。( با استفاده از برچسب های کلاس برداری ۷)

بردارهای ضریب ویژگی پیش بینی شده روی صفحه PC1-PC2 نشان می دهد که طول آنها (هنجارها) مشابه یکدیگر هستند ، با این حال جهت آنها به طور گسترده ای مشخص است. همه اینها نشان می دهد که بیشتر ویژگی ها از نظر ایجاد پیش بینی داده ها بر روی PC1-PC2 مهم هستند و این ویژگی ها ارتباط کمی با یکدیگر دارند.

PCA موفق به کاهش زیادی در ابعاد نشد ، زیرا ۱۴ (از ۲۰) مولفه برای حفظ ۹۰٪ از واریانس داده در مجموعه داده کاهش یافته مورد نیاز است.

۱. آیا داده پرت در دیتاست وجود دارد ؟ در صورت وجود آنها را حذف کنید.

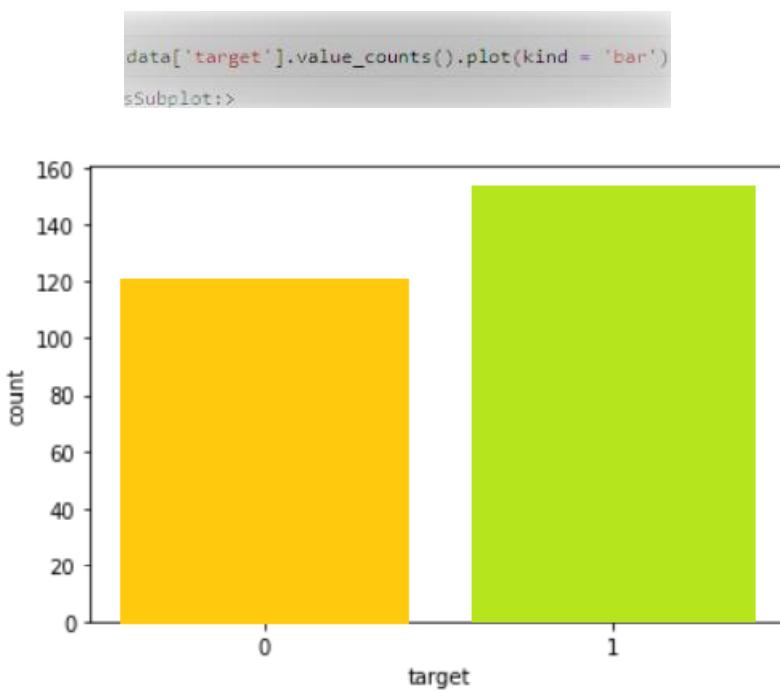
با توجه به مطالبی که تا به اینجا گفته شد میخواهیم پاسخ سوال آیا داده پرت در دیتاست وجود دارد ؟ در صورت وجود آنها را حذف کنید را بیان کنیم.

بله، در دیتاست داده‌ی پرت وجود داشت زیرا در نمودار توزیع داده‌ها نقاط و داده‌هایی دیدیم که در محدوده‌ی مشخصی از مرکز توزیع داده‌ها وجود نداشتند و فاصله‌ی زیادی هم با مرکز توزیع و هم با محدوده‌ی دیگر داده‌ها داشتند. داده‌ها استاندارد سازی شدند. ما داده‌ها را نرمالسازی کردیم و برای اینکه داده‌های پرت را حذف کنیم داده‌ای که مثلاً بیرون بازه‌ی (۳ و -۳) بود را از آن صرف نظر و حذف کردیم. توجه کنید که همیشه میتوانیم هر بازه‌ی معقولی را بسته به مسئله و توزیع داده‌های خود انتخاب کنیم.

۱. بررسی کنید آیا تعداد نمونه‌ها در هر کلاس متوازن است؟ (به

صورت مختصر توضیح دهید اگر داده‌ها متوازن نباشد چه مشکلاتی ممکن است پیش بیاید و چه راه حل‌هایی برای آن وجود دارد).

در اینجا اگر در هیستوگرامی که با توجه به مقادیری که target دارد به وجود آوردیم . می دانیم که مقدار صفر برای target یعنی بیماری قلبی نداریم و یک برای آن یعنی بیماری قلبی داریم. با مشاهده ای این هیستوگرام فهمیدیم که وقتی داده ها کلاس بندی شدند و کلاس صفر و یک برای داده ها در این نمودار می بینیم ، بیانگر این است که داده های موجود در هر دو کلاس به صورت متوازن با یکدیگر وجود دارند. و کلاس ها با یکدیگر تفاوت و اختلاف زیادی ندارند. در غیر اینصورت اگر تعداد مشاهدات مربوط به یک کلاس به طور چشمگیری کمتر از مشاهداتی باشد که به کلاس دیگر تعلق دارند. می گفتیم داده ها نامتوازن هستند.



این مشکل بیشتر در سناریوهایی که کشف ناهنجاری‌ها در آن‌ها حیاتی است، مثل کشف تقلب در سیستم بانکی، تشخیص بیماری‌های نادر و غیره اهمیت زیادی پیدا می‌کند. در چنین وضعیتی، مدل پیشگویانه‌ای که با به کارگیری الگوریتم‌های یادگیری ماشین ایجاد شده است، جهت دار و یک طرفه شده و دقت آن بسیار پایین خواهد بود. این اتفاق بدین خاطر می‌افتد که الگوریتم‌های یادگیری ماشین معمولاً طوری طراحی شده‌اند که با کاهش خطأ، دقت مدل را افزایش دهند. بنابراین، این الگوریتم‌ها توزیع/نسبت یک کلاس نسبت به کل کلاس‌ها، یا توازن کلاس‌ها را در محاسبات خود به حساب نمی‌آورند.

رویکردهای متنوعی برای حل مشکل داده‌های نامتوازن وجود دارد که تکنیک‌های نمونه‌برداری مختلفی را به کار می‌گیرند.

الگوریتم‌های یادگیری ماشین در مواجهه با دیتاست‌های نامتوازن، طبقه‌بندی‌های نامناسبی را ایجاد می‌کنند. در یک دیتاست نامتوازن اگر رویدادی که می‌خواهیم پیش‌بینی کنیم به کلاس اقلیت تعلق داشته باشد و نرخ آن رویداد کمتر از ۵ درصد باشد، معمولاً یک رویداد نادر محسوب می‌شود.

مثال: در یک دیتاست مربوط به کشف تقلب، داده‌های زیر را داریم:

– تعداد کل مشاهدات: ۱۰۰۰

– تعداد مشاهدات تقلب: ۲۰

– تعداد سایر مشاهدات (بدون تقلب): ۹۸۰

– نرخ رویداد (تقلب): ۲ درصد

سوال اصلی که در حین تحلیل داده‌ها با آن مواجه می‌شویم، این است که با توجه به تعداد کم مشاهدات مربوط به کلاس نادر، چگونه به یک دیتاست متوازن دست یابیم که در آن تعداد مناسبی از نمونه‌های مشاهدات ناهنجار موجود باشد.

## حل مشکل کلاس های نامتوازن در الگوریتم های پیش بینی

الف) رویکرد در سطح داده: تکنیک های **Resampling** .

**Random Under Sampling** .

**Random Over Sampling** .

**Cluster-Based Over Sampling** .

**Informed Over Sampling: Synthetic Minority Over Sampling Technique** .

**Modified synthetic minority oversampling technique (MSMOTE)** .

ب) تکنیک های الگوریتمی تجمعی ( **Algorithmic Ensemble Techniques** )

**Bagging Based** .

**Boosting-Based** .

**Adaptive Boosting- Ada Boost** .

**Gradient Tree Boosting** .

**XG Boost** .

۲. نمونه های موجود در دیتاست را با نسبت ۸۰ به ۲۰ به دو بخش داده های

آموزشی و داده های تست تقسیم بندی کنید . برای این کار میتوانید از

### پکیج `sklearn` ا بهره ببرید .

مجموعه داده ها را به نسبت ۸۰ (داده های آموزشی) به ۲۰ (داده های تست) تفکیک کردیم به کمک پکیج .

```
import numpy as np
from sklearn.model_selection import train_test_split
labels= data['target']
features=data.drop('target',axis=1)
x_train,x_test,y_train,y_test=train_test_split(features,labels,test_size=0.2,random_state=42)
```

X_train	فیچر های داده ترین
y_train	لیبل های داده ترین
X_test	فیچر های داده تست
y_test	لیبل های داده تست

```
[1] data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.950624	0.679881	1.969864	0.762694	-0.255910	2.390484	-1.004171	0.015417	-0.69548	1.085542	-2.270822	-0.713249	-2.145324	0.913019
1	-1.912150	0.679881	1.000921	-0.092585	0.072080	-0.416945	0.897478	1.630774	-0.69548	2.119067	-2.270822	-0.713249	-0.512075	0.913019
2	-1.471723	-1.465992	0.031978	-0.092585	-0.815424	-0.416945	-1.004171	0.975900	-0.69548	0.310399	0.974740	-0.713249	-0.512075	0.913019
3	0.179877	0.679881	0.031978	-0.662770	-0.198030	-0.416945	0.897478	1.237849	-0.69548	-0.206364	0.974740	-0.713249	-0.512075	0.913019
4	0.289984	-1.465992	-0.936965	-0.662770	2.078611	-0.416945	0.897478	0.582975	1.43311	-0.378618	0.974740	-0.713249	-0.512075	0.913019

۳. قضیه بیز را در حداقل یک پاراگراف بیان کنید . سپس دسته بند های

**Naive** ، **Multinomial Naive Bayes** ، **Gaussian Bernoulli Naive Bayes**

را با یکدیگر مقایسه کنید و بیان کنید هر کدام از این دسته بندها

بیشتر در کجا کاربرد دارند .

بیز ساده را می توان یک مدل برمبانی احتمال شرطی در نظر گرفت. فرض کنید  $X = (x_1, \dots, x_n)$  برداری از  $n$  ویژگی را بیان کند که به صورت متغیرهای مستقل هستند. به این ترتیب می توان احتمال رخداد  $C_k$  یعنی  $p(C_k | x_1, \dots, x_n)$  را به عنوان یکی از حالت های کلاس رخدادهای مختلف به ازاء  $k$  های متفاوت، به شکل زیر نمایش داد.

$$p(C_k | X) = \frac{p(C_k) p(X | C_k)}{p(X)}$$

رابطه ۱

همانطور که دیده می شود رابطه بالا همان قضیه بیز است. به عنوان یادآوری قضیه بیز را براساس احتمالات پیشامدهای «پیشین» (Prior)، «پسین» (Posterior)، «درستنمایی» (Likelihood) و «شواهد» (Evidence) در رابطه زیر بازنویسی می کنیم.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

به این ترتیب برای محاسبه احتمال  $p(C_k | x_1, \dots, x_n)$  کافی است از «احتمال توام» (Joint Probability) کمک بگیریم و به کمک احتمال شرطی با توجه به استقلال متغیرها، آن را ساده کنیم.

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k), \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

با فرض استقلال مولفه‌ها یا ویژگی‌های  $x_i$ ‌ها از یکدیگر می‌توان احتمالات را به شکل ساده‌تری نوشت. کافی است رابطه زیر را در نظر بگیریم.

$$p(x_i \mid x_{i+1}, \dots, x_n, C_k) \approx p(x_i \mid C_k).$$

به این ترتیب احتمال توام را به صورت حاصلضرب احتمال شرطی می‌توان نوشت.

$$\begin{aligned} p(C_k \mid x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\approx p(C_k) p(x_1 \mid C_k) p(x_2 \mid C_k) p(x_3 \mid C_k) \dots \\ &= p(C_k) \prod_{i=1}^n p(x_i \mid C_k), \end{aligned}$$

رابطه ۲

نکته: در رابطه ۱ مخرج کسر در همه محاسبات یکسان و ثابت است در نتیجه می‌توان احتمال شرطی را متناسب با احتمال توام در نظر گرفت. در رابطه بالا این تناسب را با علامت  $\propto$  نشان داده‌ایم.

با توجه به نکته گفته شده، و رابطه ۲ می‌توانیم احتمال شرطی معرفی شده در رابطه ۱ را به صورت زیر بدست آوریم. در نتیجه احتمال تعلق یک مشاهده به دسته یا گروه  $C_k$  با توجه به مشاهدات  $X$  مطابق با رابطه زیر مشخص خواهد شد.

$$p(C_k \mid x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i \mid C_k)$$

توجه داشته باشید که در اینجا احتمال شواهد (مشاهدات) به صورت  $Z = p(X) = \sum_k p(C_k) p(X \mid C_k)$  در نظر گرفته شده است. واضح است که  $Z$  به شواهد و مشاهدات  $x_1, \dots, x_n$  وابسته است.

در قسمت قبل با نحوه محاسبه مدل احتمالاتی بیز آشنا شدیم. اما در این بخش به کمک «قواعد تصمیم» (Decision Rule)، دسته‌بند بیز را ایجاد و کامل می‌کنیم. یکی از اساسی‌ترین قواعد تصمیم، انتخاب فرضیه محتمل‌تر است. به این ترتیب از بین تصمیمات مختلف، آن کاری را انجام می‌دهیم که براساس شواهد جمع‌آوری شده، بیشترین احتمال رخداد را دارد. این قاعده را «حداکثر پسین» (Maximum Posterior) یا به اختصار MP می‌نامند. به این ترتیب دسته بند بیز را می‌توان به صورت تابعی از تصمیمات  $C_k$  در نظر گرفت که بوسیله تابع  $\hat{y}$  تخمین زده می‌شود. حداکثرسازی این تابع را به صورت زیر نشان می‌دهیم.

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i \mid C_k).$$

در نتیجه با توجه به توزیع‌های مختلفی که ممکن است نمونه تصادفی داشته باشد (نمونه از آن جامعه آمده باشد) یعنی  $p(x_i \mid C_k)$  می‌توان پارامترها را محاسبه و یا برآورد کرد.

اگر مشاهدات و داده‌ها از نوع پیوسته باشند، از مدل احتمالی با توزیع گاوی یا نرمال برای متغیرهای مربوط به شواهد می‌توانید استفاده کنید. در این حالت هر دسته یا گروه دارای توزیع گاوی است. به این ترتیب اگر  $k$  دسته یا کلاس داشته باشیم می‌توانیم برای هر دسته میانگین و واریانس را محاسبه کرده و پارامترهای توزیع نرمال را برای آن‌ها برآورد کنیم. فرض کنید که  $\mu_k$  میانگین و  $\sigma_k^2$  واریانس دسته  $k$  ام یعنی  $C_k$  باشد. همچنین  $n$  مشاهدات حاصل از متغیرهای تصادفی  $X$  در نظر بگیرید. از آنجایی که توزیع  $X$  در هر دسته گاوی (نرمال) فرض شده است، خواهیم داشت:

$$p(x = v | C_k) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

### دسته بند بیز ساده چندجمله‌ای (Multinomial Naive Bayes)

بیز ساده چندجمله‌ای، به عنوان یک دسته‌بند متنی بسیار به کار می‌آید. در این حالت برحسب مدل احتمالی یا توزیع چند جمله‌ای، برداری از  $n$  ویژگی برای یک مشاهده به صورت  $(x_1, \dots, x_n)$  با احتمالات  $(p_1, \dots, p_n)$  در نظر گرفته می‌شود. مشخص است که در این حالت بردار  $X$  بیانگر تعداد مشاهداتی است که ویژگی خاصی را دارا هستند. به این ترتیبتابع درستنمایی در چنین مدلی به شکل زیر نوشته می‌شود.

$$p(\mathbf{x} | C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

اگر مدل بیز ساده را براساس لگاریتم تابع درستنمایی بنویسیم، به صورت یک دسته‌بند خطی درخواهد آمد.

$$\begin{aligned} \log p(C_k | \mathbf{x}) &\propto \log \left( p(C_k) \prod_{i=1}^n p_{ki}^{x_i} \right) \\ &= \log p(C_k) + \sum_{i=1}^n x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x} \end{aligned}$$

واضح است که در این حالت  $w_{ki} = \log p_{ki}$  و  $b = \log p(C_k)$  است.

### دسته بند بیز ساده برنولی (Bernoulli Naive Bayes)

در این قسمت به بررسی توزیع برنولی و دسته‌بندی بیز خواهیم پرداخت. به شکلی این نوع از دسته‌بندی متن‌های کوتاه داشته، به همین دلیل محبوبیت بیشتری نیز دارد. در این مدل در حالت چند متغیره، فرض بر این است که وجود یا ناموجود بودن یک ویژگی در نظر گرفته شود. برای مثال با توجه به یک لغتنامه مربوط به اصطلاحات ورزشی، متن دلخواهی مورد تجزیه و تحلیل قرار می‌گیرد و بررسی می‌شود که آیا کلمات مربوط به لغتنامه ورزشی در متن وجود دارند یا خیر. به این ترتیب مدل تابع درستنمایی متن براساس کلاس‌های مختلف  $C_k$  به شکل زیر نوشته می‌شود.

$$p(\mathbf{x} | C_k) = \prod_{i=1}^n p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

مشخص است که منظور از  $p_{ki}$  احتمال تولید مشاهده  $x_i$  از کلاس  $k$  است.

## مدل های طبقه بندی

### طبقه بندی - ۱

برای این مشکل طبقه بندی ، ما به طور طبیعی تصمیم گرفته ایم که پیش بینی اینکه آیا یک بیمار بیماری قلب دارد یا خیر (یعنی اگر متغیر "هدف" ۰ یا ۱ باشد) را از طریق یک طبقه بندی با استفاده از همه ۲۰ ویژگی دیگر مجموعه داده حل کنیم.) پس از انجام رمزگذاری خارج از k، زیرا این همان هدفی است که مجموعه داده برای آن جمع آوری شده است. بنابراین مسئله یک مشکل طبقه بندی باینری است. ما برای مقابله با خطاهای احتمالی که ممکن است از اختلاف زیاد مقیاس ناشی شوند، مقادیر ۲۰ ویژگی خود را مجدداً استاندارد خواهیم کرد.

### طبقه بندی - ۲

برای وظیفه فعلی خود ، ما عملکرد سه طبقه بندی کننده را با هم مقایسه خواهیم کرد: یک رگرسیون لجستیک ، یک طبقه بندی ANN و یک خط مبنای ارائه دهنده پیش بینی تمام مشاهدات به عنوان متعلق به بزرگترین کلاس در مجموعه داده های آموزشی. از آنجا که مراحل مدل سازی از نزدیک مراحل انجام شده برای مدلهای ما در مسئله رگرسیون را منعکس می کند ، ما فقط تفاوتها را ذکر خواهیم کرد ، در غیر این صورت به زودی مراحل مشابه را بدون توضیحات بیشتر اطلاع خواهیم داد. طبقه بندی ANN براساس آزمایش آزمایشی تخمین خطای تعیین با استفاده از اعتبارسنجی متقابل بر روی تعدادی از الگوریتم های مختلف ، به عنوان روش طبقه بندی دوم ما انتخاب شد. نتایج به زودی در جدول ۴ زیر نشان داده شده است ، نشان می دهد که شبکه های عصبی ملی با کمترین خطای طبقه بندی غلط بر روی داده های ما پیش بینی کرده اند. یک واقعیت جالب کوچک که باید توجه شود این است که طبقه بندی کننده Naïve-Bayes در حال دستیابی به عملکرد بسیار نزدیک به ANN است ، که این اتفاق نادر برای این روش است. با این حال ، اگر به آن فکر کنیم ، دلیل اینکه Naïve-Bayes در برخی از مجموعه های داده عملکرد ضعیفی کسب می کند ، به این فرض است که همه ویژگی ها به طور مستقل با هم مرتبط هستند. اگرچه در مورد ما ، بسیاری از ویژگی ها در واقع ارتباط بسیار کمی با هم دارند ، بنابراین نیاز Naïve-Bayes را برآورده می کنند و در نتیجه عملکرد بسیار خوبی با این الگوریتم طبقه بندی دارند.

ما برای محاسبه و آموزش مدل های خود برای به دست آوردن دقیق ترین

اندازه گیری خطای تعمیم ، مجدداً از اعتبارسنجی دو سطحه  $K_1 = 10$

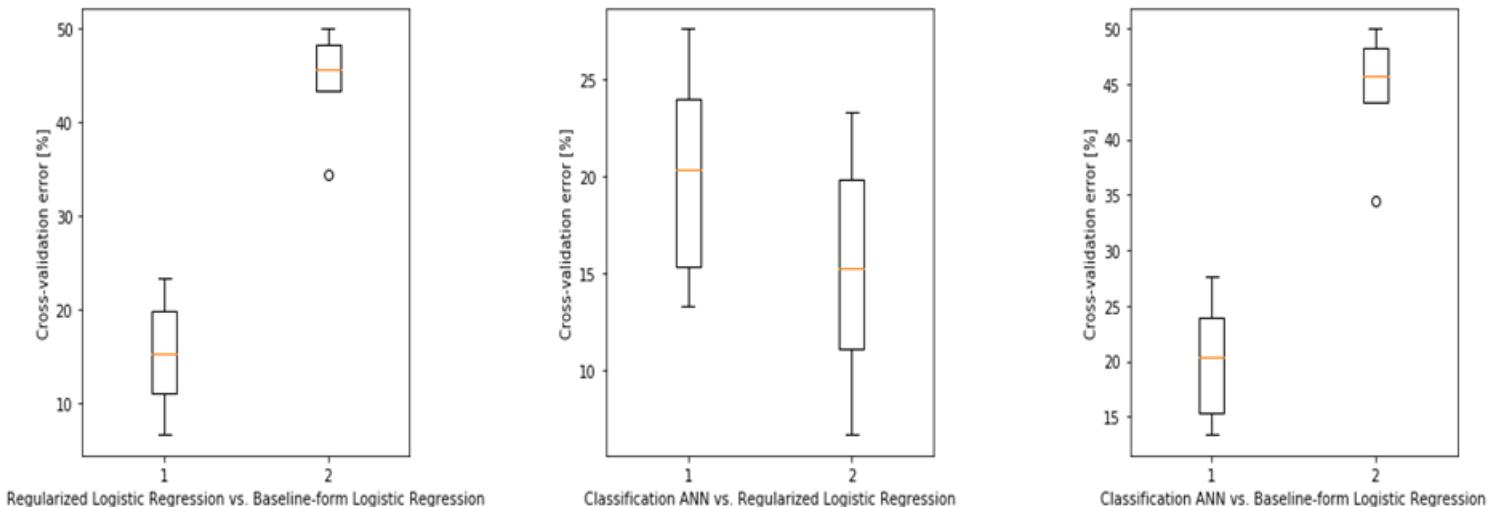
(خارجی) و  $K_2 = 10$  برابر (داخلی) استفاده خواهیم کرد. برای رگرسیون لجستیک ، خروجی آن یک تابع خطی خواهد بود که به یک تابع سیگموئید منتقل می شود ، بنابراین احتمال بین ۰ و ۱ را ایجاد می کند که یک مشاهدات متعلق به کلاس خاصی از داده های ما است. ما آستانه تمایز کلاس را در ۵,۰ قرار می دهیم ، و ما با جستجوی (دوباره همانطور که قبل) برای یک  $\lambda$  که مدل واریانس آموزش را با حداقل مقدار تعصب معرفی شده به حداقل می رساند ، مدل را منظم می کنیم. مانند قبل ، این به عنوان قدرت ۱۰ بین ۱۰-۵ و ۱۰۹ انتخاب خواهد شد. همانطور که قبل ، یک رهگیری در مدل رگرسیون لجستیک نصب می شود. برای طبقه بندی **ANN** ، ما یک شبکه با ۱ لایه خطی پنهان با استفاده از یک تابع فعال سازی **tanh** و ۱ لایه خروجی خطی با یک تابع انتقال **Sigmoid** نهایی آموزش خواهیم داد تا به عنوان طبقه بندی کننده ما عمل کند. پارامتر کنترل کننده پیچیدگی مجدداً تعداد واحدهای پنهان در لایه پنهان خواهد بود که از ۱ تا ۵ امکان پذیر است. از آنجا که داده های ما باینری طبقه بندی می شوند ، تابع تلفات این بار به تابع **BCELoss** (از دست دادن طبقه بندی باینری) تغییر یافته است. . تعداد شبکه های مراکت مجدداً به ۳ و حداقل تعداد دوره های ما دوباره ۱۰۰۰۰ تنظیم شده است. سرانجام ، برای پایه ما ، ما بزرگترین کلاس موجود در داده های آموزش را پیش بینی می کنیم و "کورکورانه" همه مشاهدات آزمون را متعلق به چنین پیش بینی می کنیم.

### طبقه بندی - ۳

ما برای هر یک از چین های بیرونی ، مقدار بهینه پارامترهای کنترل کننده پیچیدگی برای طبقه بندی **ANN** و مدل های رگرسیون لجستیک ، همراه با ارائه خطاهای طبقه بندی باینری تعمیم تخمین زده شده  $\{test\_s \wedge E\}$  تست مدل ها ، در تقسیم آزمون مجموعه داده های متقابل ما ارزیابی شده است. برای هر ارزیابی ، ما از شاخص های تقسیم اعتبارسنجی مجدداً استفاده می کنیم تا مقایسه آماری خطاهای را امکان پذیر کنیم.

## طبقه بندی - ۴

با استفاده از همان روش ارزیابی آماری ، همانطور که قبلاً برای مدل های رگرسیون انجام شده بود ، در مورد تفاوت میانگین ها در خطای تعمیم سه مدل ، ما ارزیابی آماری سه مدل طبقه بندی خود را محاسبه خواهیم کرد:



نتایج نشان می دهد که هر دو طبقه بندی **ANN** و رگرسیون لجستیکی منظم نتایج بسیار خوبی ارائه می دهند ، به طور کلی خطای تعمیم را به زیر نتایج سطح پایه کاهش می دهد. ما متوجه می شویم که بهترین نتایج برای شبکه **ANN** با استفاده از کمتر از حداکثر تعداد واحد پنهان محاسبه شده است، که نشان می دهد شبکه **ANN** برای این کار بیش از حد پیچیده نیست. همچنین ، قدرت تنظیم قاعده  $\lambda$  در تمام موارد "برنده" به عنوان مقدار ۱۰ انتخاب شده است ، بنابراین نشان می دهد که برخی از اتصالات اضافی در مدل غیر منظم ما وجود دارد ، با این حال این مسئله حل می شود. ما به درستی تعیین کرده ایم که تمام خطاهای مدل پایه در زیر آستانه ۵۰٪ قرار دارند (که انتظار می رود ، زیرا اگر به طور تصادفی کلاس رایج تری را روی یک مجموعه داده حاوی دو کلاس پیش بینی کنیم ، بالاترین حالت خطای این تابع زمانی است که هر دو کلاس قرار داشته باشند) داشتن تعداد عناصر یکسان ، بنابراین یک خط مبنا حداکثر خطای ۵۰٪ را نشان می دهد. بر اساس شکل ۶ ، واضح است که هم رگرسیون لجستیکی منظم و هم طبقه بندی **ANN** به طور چشمگیری در این کار طبقه بندی مجموعه داده ساده عمل می کنند ، و پس از حداقل استفاده از پیچیدگی بیش از حد ، از نظر آماری هیچ یک از آنها به طور قابل توجهی بهتر از دیگری نیست - کنترل پارامترها از آنجا که آموزش صحیح **ANN** چندین ساعت به طول انجامید ، در حالی که رگرسیون لجستیک در کمتر از ۱ ثانیه آموزش دیده است ، می توانیم استدلال کنیم که ، برای مجموعه داده های ما ، نصب **ANN** غیرضروری و مصرف کننده منابع است.

آموزش مدل طبقه بندی ما بر روی یک رگرسیون لجستیکی منظم با لامبدا بهینه ای که در بالا یافت می شود به شرح زیر است  $\lambda = 10$  : ، ما معادله خطی زیر را بدست می آوریم (که سپس برای تولید نتیجه احتمال نهايی ما در تابع سیگموئید قرار می گيرد):

$$\check{y} = \sigma(0.18 + 0.01x_{age} - 0.21x_{trestbps} - 0.11x_{chol} + 0.31x_{thalach} \\ - 0.34x_{oldpeak} - 0.47x_{sex-male} - 0.06x_{cp-atypical} + 0.16x_{cp-nonanginal} - 0.53x_{cp-asymptomatic} + 0.09x_{fbs-true} + 0.15x_{restecg-st-t} - 0.04x_{restecg-hypertrophy} - 0.28x_{exang-yes} - 0.22x_{x-thal-0.56} - 0.35x_{ca-3} - 0.03x_{thal-0.56} - 0.59x_{ca-2} - 0.59x_{ca-1})$$

برگشت پذیر  $\{\text{thal}\}$

می توانیم متوجه شویم که ، از آنجا که داده ها قبل از نصب مدل نرمال و استاندارد شده اند ، وزن ها به همان ترتیب اندازه محاسبه و نشان داده می شوند. در این شرایط ، تفسیر درست است که به عنوان مثال مقدار "thalach" قوی ترین مؤلفه مثبت تأثیرگذار در معادله است (معقول ، زیرا حداکثر ضربان قلب ثبت شده احتمالاً با افزایش احتمال بیماری قلبی ارتباط مثبت خواهد داشت) ، یا اینکه احتمال ابتلا به بیماری قلبی در مردان به میزان قابل توجهی کمتر از احتمال زنان از آنجا که  $x_{sex-male}$  ضریب منفی دارد . در مقایسه با وزن هایی که قبلًا برای مسئله رگرسیون انتخابی ماند شده است ، می توانیم متوجه شویم که هم ارتباط و هم میزان تأثیر (و علامت) بر روی خروجی در بیشتر ویژگی های ما تغییر کرده است. ما تعیین کرده ایم که وظیفه طبقه بندی ما در پیش بینی اینکه بیماران مبتلا به بیماری عروق قلب هستند یا نه بر اساس پارامترهای مجموعه داده جمع آوری شده ، می تواند حل شود و با استفاده از روش های یادگیری ماشین رگرسیون لجستیک یا ANN های طبقه بندی ، به یک سطح دقیق بالا برسیم. در مورد تجزیه و تحلیل مسئله رگرسیون ، ما استدلال می کنیم که نتایج مدل ما فقط از طریق جمع آوری داده های جدید بهبود می یابد ، زیرا مجموعه داده و بسیاری از ویژگی های آن بسیار ضعیف با هم ارتباط دارند.

۴. در نظر گرفتن فیچر ها `chol`، `trestbps` و `thalach` target یک

دسته بند `Bayes Naive Gaussian` را از پایه پیاده سازی کنید. (بدون

استفاده از پکیج) برای این کار شما نیاز است که در دیتاست آموزشی خود

اعضاي مختلف قاعده بيز را محاسبه کنید.

دو دسته داده‌ی آموزشی و داده‌ی تست در کلاس `guassclf` داریم.

در حالت کلی طبقه بندی **Naive Bayes** در مراحل زیر احتمال یک رویداد را محاسبه می‌کند:

مرحله ۱: محاسبه احتمال قبلی برای برچسب‌های کلاس داده شده

مرحله ۲: احتمال **Likelihood** را با هر ویژگی برای هر کلاس پیدا کنیم.

مرحله ۳: این مقدار را در فرمول **Bayes** قرار دهید و احتمال خلفی را محاسبه کنیم.

مرحله ۴: با توجه به اینکه ورودی مربوط به کلاس احتمال بالاتر است، بررسی کدام کلاس احتمال

بیشتری دارد.

برای ساده سازی محاسبه احتمال قبلی و خلفی می‌توانید از دو جدول فراوانی و احتمال استفاده کنید. هر

دو جدول به شما کمک می‌کنند تا احتمال قبل و بعد را محاسبه کنید. جدول فرکانس شامل برچسب‌ها

برای همه ویژگی‌ها است.

```

from sklearn.metrics import classification_report

import numpy as np
import math
data2 = data[['chol','trestbps','thalach']]
X_train, X_test, y_train, y_test = train_test_split(data2,
                                                    data['target'], test_size=0.2, shuffle=True)

class gaussClf:
    def separate_by_classes(self, X, y):
        self.classes = np.unique(y)
        classes_index = {}
        subdatasets = {}
        cls, counts = np.unique(y, return_counts=True)
        self.class_freq = dict(zip(cls, counts))
        print(self.class_freq)
        for class_type in self.classes:
            classes_index[class_type] = np.argwhere(y==class_type)
            subdatasets[class_type] = X[classes_index[class_type], :]
            self.class_freq[class_type] = self.class_freq[class_type]/sum(list(self.class_freq.values()))
        return subdatasets

    def fit(self, X, y):
        separated_X = self.separate_by_classes(X, y)
        self.means = {}
        self.std = {}
        for class_type in self.classes:
            self.means[class_type] = np.mean(separated_X[class_type], axis=0)[0]
            self.std[class_type] = np.std(separated_X[class_type], axis=0)[0]

    def calculate_probability(self, x, mean, stdev):
        exponent = math.exp(-((x - mean) ** 2 / (2 * stdev ** 2)))
        return (1 / (math.sqrt(2 * math.pi) * stdev)) * exponent

    def predict_proba(self, X):
        self.class_prob = {cls:math.log(self.class_freq[cls], math.e) for cls in self.classes}
        for cls in self.classes:
            for i in range(len(self.means)):
                self.class_prob[cls]+=math.log(self.calculate_probability(X[i], self.means[cls][i], self.std[cls][i]), math.e)
        self.class_prob = {cls: math.e**self.class_prob[cls] for cls in self.class_prob}
        return self.class_prob

    def predict(self, X):
        pred = []
        for x in X:
            pred_class = None
            max_prob = 0
            for cls, prob in self.predict_proba(x).items():
                if prob>max_prob:
                    max_prob = prob
                    pred_class = cls
            pred.append(pred_class)
        return pred

```

## ۰. پس پیاده سازی Bayes Naive Gaussian و آموزش آن بر روی داده های

آموزشی (۲۰ درصد دیتاست). نتایج را برای داده های تست (۲۰ درصد باقی

دیتاست) بررسی کنید به عبارت دیگر برای داده ورودی بررسی کنید در

بخش تست لیبل را پیش بینی کنید. با توجه به این لیبل های واقعی را نیز

برای پیش بینی به تابع **predict** نیاز داریم ، این تابع در محله‌ی تست احتمال لگاریتمی کلاس را با

قواعد بیز حساب می‌کند و با مقایسه‌ی نتیجه‌ی حاصل با نتایجی که از داده‌های تست حاصل شد دقیق

و خطای پیش بینی ما معین می‌شود..

```
clf = gaussClf()
clf.fit(X_train.values,y_train.values)
print(classification_report(y_test.values,clf.predict(X_test.values)))
```

		precision	recall	f1-score	support
0	101	0.50	0.04	0.07	28
1	128	0.52	0.97	0.67	30
accuracy				0.52	58
macro avg		0.51	0.50	0.37	58
weighted avg		0.51	0.52	0.38	58

## ۶. با استفاده از پکیج GaussianNB و sklearn یک مدل بسازید و بر روی

داده‌های آموزشی ، ترین کنید سپس بر روی داده‌های تست همانند سوال (۵)

سه معیار را گزارش دهید.

```
import pandas as pd
df=pd.read_csv(r"C:\Users\Tmfk\Desktop\heart.csv")
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
features = df[['age','thalach','chol']]
labels = df['target']
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn import datasets
from sklearn.metrics import confusion_matrix
iris=datasets.load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(features,labels,test_size=0.2,random_state=0)
gnb=GaussianNB()
mnb=MultinomialNB()
y_pred_gnb=gnb.fit(x_train,y_train).predict(x_test)
cnf_matrix_gnb = confusion_matrix(y_test, y_pred_gnb)
cnf_matrix_gnb
```

```
array([[14, 13],
       [ 5, 29]], dtype=int64)
```

```
from sklearn.metrics import f1_score
f1_score(y_test, y_pred_gnb,)
```

```
0.7631578947368423
```

## ۷. بررسی کنید که در سه معیار مطرح شده مدلی که با استفاده از پکیج ساخته اید و مدلی که خود پیاده سازی کرده اید به چه صورتی عمل کرده اند.

با بررسی دوباره‌ی جواب‌های سوال‌های پنج . شش در می‌یابیم که پکیج دقیق‌تر و بهترین جواب را برگردانده است و سرعت و دقت در پروسه‌ی یک کار بیشترین است نسبت به کارها و محاسباتی که خودمان به صورت دستی پی‌می‌گیریم زیرا خود این پکیج‌ها توسط برنامه نویسان و محققان خبره‌بنا شده است و دقت عمل بیشتری دارند.

## ۸. کلاسیفایر SVM را با استفاده از پکیج `sklearn` بر سه فیچر مطرح شده در سوال (۴) با استفاده از داده‌های آموزشی ترین کنید . سپس بر روی داده‌های تست سه معیار Precision ، score F1 ، Recall را گزارش کنید .

```
from sklearn.svm import SVC  
  
clf = SVC()  
clf.fit(X_train.values,y_train.values)  
print(classification_report(y_test.values,clf.predict(X_test.values)))
```

	precision	recall	f1-score	support
0	0.68	0.54	0.60	24
1	0.72	0.82	0.77	34
accuracy			0.71	58
macro avg	0.70	0.68	0.69	58
weighted avg	0.70	0.71	0.70	58

ما عملکرد و ویژگی های انواع مدل های یادگیری بدون ناظارت را بر روی مجموعه داده بیماری قلبی انتخاب شده ارزیابی خواهیم کرد.

```
import numpy as np
import pandas as pd
import seaborn as sb

import matplotlib.pyplot as plt
from matplotlib.pyplot import figure, show, hist, plot, legend, xlabel, title, ylim, imshow, bar, xticks, yticks, cm, subplot

from sklearn.mixture import GaussianMixture
from sklearn import model_selection
from sklearn.cluster import k_means
from sklearn.neighbors import NearestNeighbors

from scipy.cluster.hierarchy import linkage, fcluster, dendrogram
from scipy import stats

from toolbox_02450 import clusterplot, clusterval, gausKernelDensity

from similarity import binarize2
from apyori import apriori

import warnings
```

مجموعه داده ما در ابتدا برای اهداف طبقه بندی جمع آوری شده است ، دارای ۱۴ ویژگی: ۵ عددی و ۹ طبقه بندی. از میان ۱۴ ویژگی ، یک متغیر «هدف» وجود دارد که باید در یک تنظیم طبقه بندی پیش بینی شود. از آنجا که مجموعه داده اصلی ما شامل ۹ متغیر طبقه ای است ، هنگام انجام تجزیه و تحلیل یادگیری بدون ناظارت ، لازم است که آنها را با استفاده از روش **K-out-of-K** کدگذاری کنید ، به طوری که روش های یادگیری ماشین نگاشت های دسته عددی را به عنوان مقادیر پیوسته درک نکنند. پس از این تغییر ویژگی ، از مجموع ۱۴ متغیر اصلی ، با ۳۰ متغیر (شامل متغیر معیار) مواجه می شویم.

برای خوش بندی و تشخیص دور ، ما متغیرهای پیش بینی را به یک ماتریس **X** جدا خواهیم کرد ، و متغیر معیار در یک بردار جداگانه **y** قرار می گیرد ، اما برای استخراج ارتباط ، زیرا ما همچنین علاقه مند به کشف قوانین بین مقادیر متغیرهای پیش بینی و مقادیر متغیر معیار (**y**) ، ما از یک **K** دو امکان برای متغیر ملاک "هدف" (که به این معناست که آیا یک فرد دارای بیماری قلب است) رمزگذاری می شود و آنها را در ماتریس ما قرار می دهد **X** برای کامل بودن

## ۹. حداقل دو حالت مختلف را برای کرnel در SVM ساخته شده با پکیج در نظر

بگیرید و نتایج آن را گزارش دهید . آیا کرnel های مختلف نتایج مختلفی ارایه دادند ؟ به صورت کلی علت استفاده از کرnel ها در SVM چیست ؟ توضیح دهید .

ابتدا قبل از آن به سراغ تراکم هسته گوسی (Gaussian Kernel Density) می رویم.

محاسبه تراکم با استفاده از دانسیته هسته گوسی به انتخاب عرض هسته  $\lambda$ ، که به آن پهنانی باند نیز گفته می شود ، نیاز دارد که می تواند به طور مشابه با حساسیت فرمول احتمال نسبت به مناطق بسیار کم یا ضعیف داده مرتبط باشد. پهنانی باند هسته بسیار زیاد به راحتی می تواند از بین برود ، در حالی که پهنانی باند بسیار پایین باعث می شود مدل تراکم در برابر تفاوت بین مناطق با چگالی بالا و چگالی کم بسیار محکم باشد.

روشی برای انتخاب عرض ایده آل  $\lambda$  از طریق روش تأیید اعتبار متقاطع است ، جایی که ما احتمال ورود به سیستم را برای محاسبه تراکم نقاط داده داده مجموعه آزمایش می کنیم ، و ما به طور موثر  $\lambda$  را انتخاب می کنیم که این ثبت را به حداکثر می رساند. اندازه گیری احتمال با توجه به اینکه ، برای مجموعه داده ما ، پهنانی باند مطلوب  $0.5 = \lambda$  است ، ما اکنون به ارائه ۲۰ نقطه داده با کمترین تراکم و پرتوهای بالقوه می پردازیم.

```

outlier_observations = np.zeros((3,20))

### Gaussian Kernel density estimator

# cross-validate kernel width by leave-one-out-cross-validation
# (efficient implementation in gausKernelDensity function)
# evaluate for range of kernel widths

widths = X.var(axis=0).max() * (2.0**np.arange(-10,3))
logP = np.zeros(np.size(widths))

for i,w in enumerate(widths):
    # print('Fold {:2d}, w={:f}'.format(i,w))
    density, log_density = gausKernelDensity(X,w)
    logP[i] = log_density.sum()

# figure(1)
# title('Log likelihood of data using various kernel widths')
# plot(widths, logP)

val = logP.max()
ind = logP.argmax()
width = widths[ind]
print('Optimal estimated width is: {:.2f}'.format(width))

# evaluate density for estimated width
density, log_density = gausKernelDensity(X,width)

# Sort the densities
i = (density.argsort(axis=0)).ravel()
density = density[i].reshape(-1,)

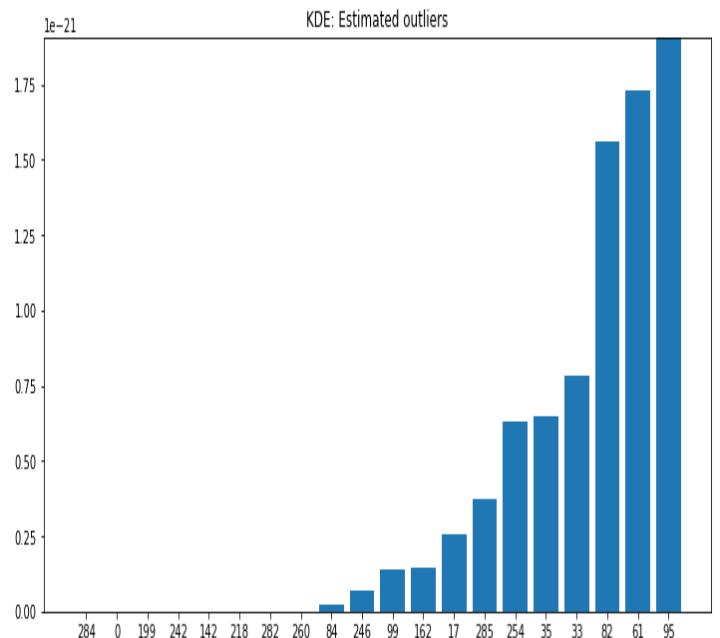
# Plot possible outliers
i = i.astype(str)

print()

plt.figure(figsize = [12, 6])
plt.title('KDE: Estimated outliers')
for k in range(0,20):
    plt.bar(i[k], density[k], color = sb.color_palette()[0])
outlier_observations[0,:] = i[:20]

```

Optimal estimated width is: 0.50



- ما کرنل خطی و غیر خطی را به کار بردیم و

نتیجه را بررسی می کنیم.

```
from sklearn.svm import SVC
```

```
clf = SVC(kernel = 'linear')
clf.fit(X_train.values,y_train.values)
print(classification_report(y_test.values,clf.predict(X_test.values)))
```

	precision	recall	f1-score	support
0	0.75	0.62	0.68	24
1	0.76	0.85	0.81	34
accuracy			0.76	58
macro avg	0.76	0.74	0.74	58
weighted avg	0.76	0.76	0.75	58

```
from sklearn.svm import SVC
```

```
clf = SVC(kernel = 'poly')
clf.fit(X_train.values,y_train.values)
print(classification_report(y_test.values,clf.predict(X_test.values)))
```

	precision	recall	f1-score	support
0	0.71	0.62	0.67	24
1	0.76	0.82	0.79	34
accuracy			0.74	58
macro avg	0.74	0.72	0.73	58
weighted avg	0.74	0.74	0.74	58

در اینجا کرنل های مختلف نتیجه های مختلفی ارائه دادند و کرنل خطی در اینجا بهتر عمل می کند.

عملکرد **kernel** روشی است که برای گرفتن داده به عنوان ورودی و تبدیل به فرم مورد نیاز پردازش داده استفاده می شود. "**kernel**" به دلیل مجموعه ای از توابع ریاضی مورد استفاده در ماشین بردار پشتیبانی پنجره ای را برای دستکاری داده ها استفاده می کند. بنابراین ، عملکرد هسته به طور کلی مجموعه داده های آموزش را به گونه ای تغییر می دهد که سطح تصمیم گیری غیر خطی قادر به تبدیل شدن به یک معادله خطی در تعداد بیشتری از فضاهای بعد باشد. اساساً ، این محصول داخلی را در یک بعد ویژگی استاندارد بین دو نقطه برمی گرداند.

معادله عملکرد استاندارد هسته:

$$\begin{aligned} K(\bar{x}) &= 1, \text{if } \|\bar{x}\| \leq 1 \\ K(\bar{x}) &= 0, \text{Otherwise} \end{aligned}$$

برای پیاده سازی توابع هسته ، اول از همه ما باید کتابخانه "scikit-learn" را با استفاده از ترمینال خط فرمان نصب کنیم.

**Gaussian Kernel**: هسته گوسی: برای انجام تحول استفاده می شود ، در صورتی که دانش قبلی در مورد داده ها وجود نداشته باشد.

$$K(x, y) = e^{-\left(\frac{\|x-y\|^2}{2\sigma^2}\right)}$$

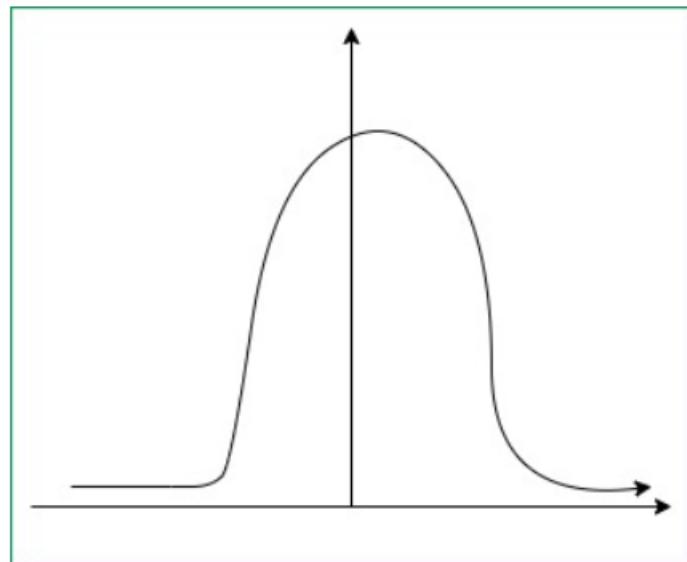
عملکرد پایه شعاعی هسته گوسی (RBF): همان عملکرد هسته فوق ، افزودن روش پایه شعاعی برای بهبود تحول.

$$K(x, y) = e^{-\gamma \|x - y\|^2}$$

$$K(x, x1) + K(x, x2) \text{ (Simplified - Formula)}$$

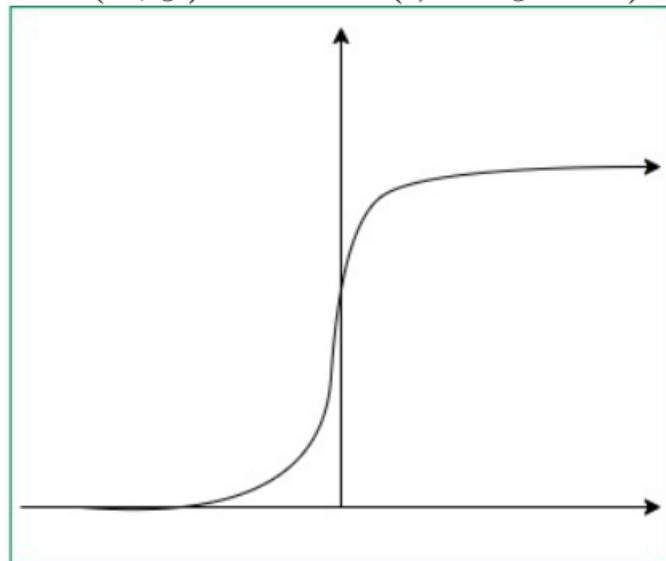
$$K(x, x1) + K(x, x2) > 0 \text{ (Green)}$$

$$K(x, x1) + K(x, x2) = 0 \text{ (Red)}$$



هسته سیگموئید: این عملکرد معادل مدل دو لایه ای از شبکه عصبی پرسپترون است که به عنوان عملکرد فعال سازی نورونهای مصنوعی استفاده می شود.

$$K(x, y) = \tanh(\gamma \cdot x^T y + r)$$



## ۱. دسته بند SVM را با استفاده از پکیج `sklearn` بسازید و با در نظر گرفتن

کلیه فیچرهای دیتاست بر روی داده های آموزشی ترین کنید سپس نتایج را

بر روی داده های تست ، ارزیابی کنید.

زمانی که کل داده های آموزشی را برای ترین انتخاب می کنیم ، بنابراین مدل ما روی تمام فیچر ها و داده ها ساخته شده و خطای آن کمتر و دقتش بالاتر می رود و بنابراین مدل پیش بینی ما دقیق تر می شود و روی داده های تست خطای کمتر و دقت بیشتری دارد.

```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(data.drop(['target'],axis=1),
                                                    data['target'], test_size=0.2, shuffle=True)

clf = SVC(kernel = 'linear')
clf.fit(X_train.values,y_train.values)
print(classification_report(y_test.values,clf.predict(X_test.values)))

precision    recall   f1-score   support

          0       0.96      0.78      0.86      32
          1       0.78      0.96      0.86      26

accuracy                           0.86      58
macro avg       0.87      0.87      0.86      58
weighted avg    0.88      0.86      0.86      58
```

۱۱. برای سوال (۱۰) یکبار مدل را با Validation Cross fold-5 اجرا کنید و

نتایج را گزارش دهید . (در اینجا برای فولد کردن داده ها از کل دیتاست

استفاده میکنیم.)

```
precision= 0.81  
recall= 0.89  
f1-score= 0.85
```



```
mean score: 0.8380676328502416  
score std: 0.04456023563792447
```

ابتدا این محاسبه می شود.

سپس این محاسبه می شود.

۱۰. با استفاده از پکیج `sklearn` دسته بند را K-NN را بسازید . با به

کارگیری تمامی فیچرها موجود در دیتاست آموزشی ، مدل را ترین کنید

سپس بر روی دیتاست تست ، ارزیابی کنید.

قبل از پاسخ به سوال ابتدا معرفی میکنیم : تراکم نزدیک ترین همسایگان (K-NN)

معیار تراکم **KNN** ، برخلاف **KDE** ، به احتمالات متکی نیست و تراکم نقطه را با استفاده از میانگین فاصله تا **K** نزدیکترین همسایگان محاسبه می کند. به طور مستقیم ، هرچه این میانگین فاصله کمتر باشد ، نزدیکترترین همسایگان (به طور متوسط) نزدیکتر خواهند بود و منطقه با جمعیت بیشتری از داده ها برخوردار خواهد شد. بنابراین ، تراکم **KNN** معکوس فاصله متوسط تا **K** نزدیکترین همسایه است. در شکل ۶ ، ۲۰ محتمل ترین محتوا از مجموعه داده ما نشان داده شده است.

```
### K-neighbors density estimator
# Neighbor to use:
K = 5

# Find the k nearest neighbors
knn = NearestNeighbors(n_neighbors=K).fit(X)
D, i = knn.kneighbors(X)

density = 1./(D.sum(axis=1)/K)

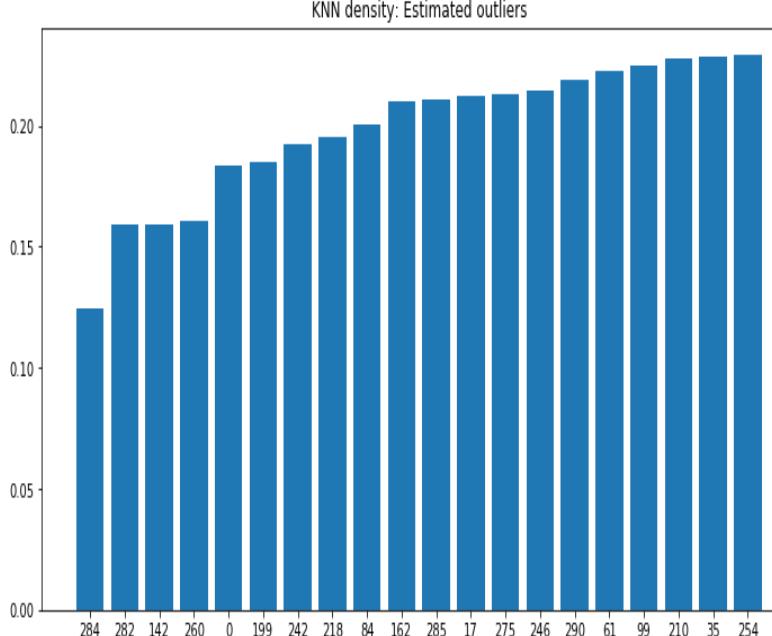
# Sort the scores
i = density.argsort()
density = density[i]

# Plot possible outliers
i = i.astype(str)

print()

plt.figure(figsize = [12, 6])
plt.title('KNN density: Estimated outliers')
for k in range(0,20):
    plt.bar(i[k], density[k], color = sb.color_palette()[0])

outlier_observations[1,:] = i[:20]
```



### (KNN-ARD) نزدیکترین همسایگان- K میانگین تراکم نسبی

متريک KNN ARD فرمول چگالي KNN را يك گام جلو تر می برد و به جای اينكه در نظر بگيريد کدام نقاط داراي تراكم داده های پراكنده هستند ، فرمول ARD اين اندازه گيري های چگالي را نسبت به آنچه برای نقاط اطراف معمول است ، انجام می دهد. بنابراین ، اگر يك نقطه داراي تراكم داده های پراكنده باشد ، با اين حال نزدیکترین همسایگان آن نيز تراكم داده های پراكنده دارند ، اين دیگر به عنوان سیگنال های دور از ذهن در نظر گرفته نمی شود. شکل ۷ ۲۰ نقطه داده را نشان می دهد که تراكم مساحت آنها بسیار متفاوت از همسایگان آن است

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
print(classification_report(y_test.values,clf.predict(X_test.values)))

precision    recall   f1-score   support
          0       0.73      0.50      0.59       32
          1       0.56      0.77      0.65       26

accuracy                           0.62      58
macro avg       0.64      0.63      0.62      58
weighted avg    0.65      0.62      0.62      58
```

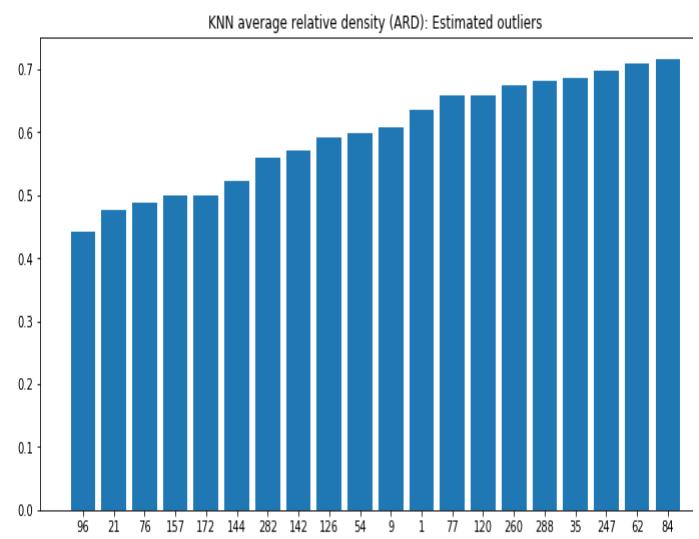
```
### K-nearest neighbor average relative density
# Compute the average relative density
knn = NearestNeighbors(n_neighbors=K).fit(X)
D, i = knn.kneighbors(X)
density = 1./D.sum(axis=1)/K
avg_rel_density = density/(density[i[:,1:]].sum(axis=1)/K)

# Sort the avg_rel.densities
i_avg_rel = avg_rel_density.argsort()
avg_rel_density = avg_rel_density[i_avg_rel]

# Plot possible outliers
i_avg_rel = i_avg_rel.astype(str)

print()

plt.figure(figsize = [12, 6])
plt.title('KNN average relative density (ARD): Estimated outliers')
for k in range(0,20):
    plt.bar(i_avg_rel[k], avg_rel_density[k], color = sb.color_palette()[0])
outlier_observations[2,:] = i_avg_rel[:20]
```



۱۳. بررسی کنید در سوال (۱۲) تعداد همسایه ها  $k$  چه نقشی ایفا میکند؟ زیاد شدن همسایه ها

خوب است؟ پگونه میتوان مشخص کرد چه تعداد همسایه برای مسئله ما مناسب است؟

کیفیت  $kNN$  یا  $k$ -نزدیکترین همسایگان، یک الگوریتم طبقه بندی است. با این حال، این با طبقه بندی هایی که قبلًا توصیف شد متفاوت است زیرا این یک زبان آموز تنبل است.

یادگیرنده تنبل چیست؟

یک یادگیرنده تنبل در طول فرآیند آموزش کار دیگری به جز ذخیره اطلاعات آموزش انجام نمی دهد. فقط وقتی داده های جدید بدون برچسب وارد می شوند، این نوع از یادگیرندهای به دنبال طبقه بندی هستند. از طرف دیگر، یک یادگیرنده مشتاق، یک مدل طبقه بندی را در حین آموزش ایجاد می کند. وقتی داده های جدید بدون برچسب وارد می شوند، این نوع از فراگیران داده ها را در مدل طبقه بندی تغذیه می کنند.

فقط داده های آموزش دارای برچسب را ذخیره می کند.

وقتی داده های جدید بدون برچسب وارد می شوند،  $kNN$  در ۲ مرحله اساسی کار می کند:

اول، آن را به نزدیکترین نقاط برچسب داده داده آموزش - به عبارت دیگر، نزدیکترین همسایگان  $k$ .

دوم، با استفاده از کلاس های همسایگان،  $kNN$  ایده بهتری از نحوه طبقه بندی داده های جدید بدست می آورد.

چگونه  $kNN$  می فهمد چه چیزی نزدیکتر است؟

برای داده های مداوم،  $kNN$  از یک معیار فاصله مانند فاصله اقلیدسی استفاده می کند. انتخاب معیار فاصله تا حد زیادی به داده ها بستگی دارد. برخی حتی یادگیری معیار فاصله را براساس داده های آموزش پیشنهاد می دهند.

برای داده های گستته ، ایده تبدیل داده های گستته به داده های مداوم است. ۲ نمونه از این موارد عبارتند از:

استفاده از فاصله **Hamming** به عنوان معیاری برای "نزدیکی" ۲ رشته متن.

تبدیل داده های گستته به ویژگی های باینری.

این ۲ موضوع **Stack Overflow** دارای پیشنهادات بیشتری در زمینه برخورد با داده های گستته است:

طبقه بندی **KNN** با داده های طبقه بندی شده

استفاده از **k-NN** در  $\mathbb{R}$  با مقادیر طبقه ای

هنگامی که همسایه ها اختلاف نظر دارند ، **kNN** چگونه داده های جدید را طبقه بندی می کند؟

**kNN** وقتی همه همسایه ها یک کلاس هستند اوقات راحتی دارد. شهود این است که اگر همه همسایگان موافق باشند ، پس داده جدید به احتمال زیاد در همان کلاس قرار می گیرد.

وقتی همسایگان کلاس یکسانی ندارند **kNN** چگونه کلاس را تعیین می کند؟

۲ روش معمول برای مقابله با این موارد عبارتند از:

با اکثریت آرا **simple** همسایگان رای گیری کنید. هر کلاسی که بیشترین تعداد آرا را داشته باشد ، برای نقطه داده

جدید به کلاس تبدیل می شود.

رأی مشابهی بگیرید به جز اینکه وزن بیشتری به همسایگان نزدیک خود بدھید. یک روش ساده برای این کار استفاده از فاصله متقابل به عنوان مثال است. اگر همسایه ۵ واحد فاصله دارد ، رای او را  $1/5$  بسازید. هرچه همسایه دورتر می

شود ، فاصله متقابل نیز کوچکتر می شود ... دقیقاً همان چیزی که ما می خواهیم!

این یادگیری تحت نظارت است ، زیرا **kNN** یک مجموعه داده آموزشی با برچسب ارائه می شود.

چرا از **kNN** استفاده می کنیم؟

سهولت درک و پیاده سازی ۲ دلیل اصلی در استفاده از **kNN** است. بسته به اندازه گیری فاصله ، **kNN** می تواند کاملاً دقیق باشد.

**kNN** هنگام تعیین نزدیکترین همسایگان در یک مجموعه داده بزرگ ، می تواند از نظر محاسباتی گران شود.

داده های پر سر و صدا می توانند طبقه بندی **kNN** را از بین ببرند.

ویژگی هایی که دامنه مقادیر بیشتری دارند می توانند نسبت به ویژگی هایی که دامنه کمتری دارند بر معیار فاصله مسلط شوند ، بنابراین مقیاس گذاری ویژگی ها مهم است.

از آنجا که پردازش داده به تعویق افتاده است ، **kNN** معمولاً به طبقه بندی کننده های مشتاق نیاز به ذخیره سازی بیشتری دارد.

انتخاب یک معیار فاصله خوب برای دقت **kNN** بسیار مهم است.

کجا استفاده می شود؟

تعدادی از پیاده سازی **kNN** وجود دارد:

طبقه بندی نزدیکترین همسایه **MATLAB**

**scikit-learn KNeighboursClassifier**

**R**-طبقه بندی نزدیکترین همسایه در

مقدار و شیب نمودار **precision** و **recall** وقتی که تعداد همسایه ها می یابد تغییر اما **score-f1** تقریبا ثابت است.

می توان از **validation-cross-kfold** کمک بگیریم. تعداد مناسب خوشه ها ممکن است ۲ یا بیشتر باشد (با توجه به خوشه بندی سلسله مراتبی شاید ۴ یا ۸) و از روش خوشه بندی سلسله مراتبی مجموعه داده از رویکرد **GMM** استفاده می کنیم.

۱۴. در سوال (۱۲) به جای استفاده از تمامی فیچرها فقط از سه فیچر `chol` ،

استفاده کنید و مدل را بسازید . سپس نتایج ارزیابی `thalach`، `trestbps`

گزارش دهید .

نتایج حاصل از مدلی که با سه فیچر ساختیم:

`precision= 0.666 , recall= 0.727 , f1-score= 0.692`

۱۵. تفاوت بین روش های کلاس بندی پارامتری و غیرپارامتری را به صورت خلاصه

بیان کنید . هر کدام بهتر است در چه مواقعی استفاده شوند ؟

در حالت عمومی آمار پارامتریک مستلزم پیش فرضهایی در مورد جامعه ای که از آن نمونه گیری صورت گرفته می باشد. به عنوان مهمترین پیش فرض در آمار پارامتریک فرض می شود که توزیع جامعه نرمال است اما آمار ناپارامتریک مستلزم هیچگونه فرضی در مورد توزیع نیست. به همین خاطر بسیاری از تحقیقات علوم انسانی که با مقیاس های کیفی سنجیده شده و قادر توزیع (Free of distribution) هستند از شاخصهای آمارا ناپارامتریک استفاده می کنند.

فنون آمار پارامتریک شدیداً تحت تاثیر مقیاس سنجش متغیرها و توزیع آماری جامعه است. اگر متغیرها از نوع اسمی و ترتیبی بوده حتما از روش های ناپارامتریک استفاده می شود. اگر متغیرها از نوع فاصله ای و نسبی باشند در صورتیکه فرض شود توزیع آماری جامعه نرمال یا بهنجار است از روش های پارامتریک استفاده می شود در غیراینصورت از روش های ناپارامتریک استفاده می شود.

## خلاصه آزمونهای پارامتریک

آزمون **t**اتک نمونه : برای آزمون فرض پیرامون میانگین یک جامعه استفاده می شود. در بیشتر پژوهش هائی که با مقیاس لیکرت انجام می شوند جهت بررسی فرضیه های پژوهش و تحلیل سوالات تخصصی مربوط به آنها از این آزمون استفاده می شود.

آزمون **t**وابسته : برای آزمون فرض پیرامون دو میانگین از یک جامعه استفاده می شود. برای مثال اختلاف میانگین رضایت کارکنان یک سازمان قبل و بعد از تغییر مدیریت یا زمانی که نمرات یک کلاس با پیش آزمون و پس آزمون سنجش می شود.

آزمون **t**دو نمونه مستقل : جهت مقایسه میانگین دو جامعه استفاده می شود. در آزمون **t** برای دو نمونه مستقل فرض می شود واریانس دو جامعه برابر است. برای نمونه به منظور بررسی معنی دار بودن تفاوت میانگین نمره نظرات پاسخ دهنده کان بر اساس جنسیت در خصوص هر یک از فرضیه های پژوهش استفاده می شود.

آزمون **t**ولچ : این آزمون نیز مانند آزمون **t** دو نمونه جهت مقایسه میانگین دو جامعه استفاده می شود. در آزمون **t** ولچ فرض می شود واریانس دو جامعه برابر نیست. برای نمونه به منظور بررسی معنی دار بودن تفاوت میانگین نمره نظرات پاسخ دهنده کان بر اساس جنسیت در خصوص هر یک از فرضیه های پژوهش استفاده می شود.

آزمون **t**هتلینگ : برای مقایسه چند میانگین از دو جامعه استفاده می شود. یعنی دو جامعه براساس میانگین چندین صفت مقایسه شوند.

تحلیل واریانس (**ANOVA**) : از این آزمون به منظور بررسی اختلاف میانگین چند جامعه آماری استفاده می شود. برای نمونه جهت بررسی معنی دار بودن تفاوت میانگین نمره نظرات پاسخ دهنده کان بر اساس سن یا تحصیلات در خصوص هر یک از فرضیه های پژوهش استفاده می شود.

تحلیل واریانس چند عاملی (**MANOVA**) : از این آزمون به منظور بررسی اختلاف چند میانگین از چند جامعه آماری استفاده می شود.

تحلیل کوواریانس چند عاملی (**MANCOVA**) : چنانچه در **MANCOVA** بخواهیم اثر یک یا چند متغیر کمکی را حذف کنیم استفاده می شود.

## خلاصه آزمونهای ناپارامتریک

آزمون علامت تک نمونه : برای آزمون فرض پیرامون میانگین یک جامعه استفاده می شود.

آزمون علامت زوجی : برای آزمون فرض پیرامون دو میانگین از یک جامعه استفاده می شود.

ویلکاکسون : همان آزمون علامت زوجی است که در آن اختلاف نسبی تفاوت از میانگین لحاظ می شود.

من-ویتنی : به آزمون **U** نیز موسوم است و جهت مقایسه میانگین دو جامعه استفاده می شود.

کروسکال-والیس : از این آزمون به منظور بررسی اختلاف میانگین چند جامعه آماری استفاده می شود. به

آزمون **H** نیز موسوم است و تعمیم آزمون **U** مان-ویتنی می باشد. آزمون کروسکال-والیس معادل

روش پارامتریک آنالیز واریانس تک عاملی است.

فریدمن : این آزمون معادل روش پارامتریک آنالیز واریانس دو عاملی است که در آن  $k$  تیمار به صورت

تصادفی به  $n$  بلوک تخصیص داده شده اند.

کولموگروف-اسمیرنف : نوعی آزمون نیکوئی برآذش برای مقایسه یک توزیع نظری با توزیع مشاهده شده است.

آزمون تقارن توزیع : در این آزمون شکل توزیع مورد سوال قرار می گیرد. فرض بدیل آن است که توزیع متقارن نیست.

آزمون میانه : جهت مقایسه میانه دو جامعه استفاده می شود و برای  $k$  جامعه نیز قابل تعمیم است.

مک نمار : برای بررسی مشاهدات زوجی درباره متغیرهای دو ارزشی استفاده می شود.

آزمون **Q** کوکران : تعمیم آزمون مک نمار در  $k$  نمونه وابسته است.

ضریب همبستگی اسپیرمن : برای محاسبه همبستگی دو مجموعه داده که به صورت ترتیبی قرار دارند استفاده می شود.

قبل از انتخاب یک آزمون آماری بایستی به سوالات زیر پاسخ داد:

۱- چه تعداد متغیر مورد بررسی قرار می گیرد؟

۲- چند گروه مفایسه می شوند؟

۳- آیا توزیع ویژگی مورد بررسی در جامعه نرمال است؟

۴- آیا گروه های مورد بررسی مستقل هستند؟

۵- سوال یا فرضیه تحقیق چیست؟

۶- آیا داده ها پیوسته، رتبه ای و یا مقوله ای **Categorical** هستند؟

قبل از ادامه این مبحث لازم است مفهوم چند واژه آماری را یاد آور شویم که زیاد وقت گیر نیست.

۱- جامعه آماری: به مجموعه کاملی از افراد یا اشیاء یا اجزاء که حداقل در یک صفت مورد علاقه مشترک باشند، گفته می شود.

۲- نمونه آماری: نمونه بخشی از یک جامعه آماری تحت بررسی است که با روشنی که از پیش تعیین شده است انتخاب می شود، به قسمی که می توان از این بخش، استنباطهایی درباره کل جامعه بدست آورد.

۳- پارامتر و آماره: پارامتر یک ویژگی جامعه است در حالی که آماره یک ویژگی نمونه است. برای مثال میانگین جامعه یک پارامتر است. حال اگر از جامعه نمونه گیری کنیم و میانگین نمونه را بدست آوریم، این میانگین یک آماره است.

۴- برآورد و آزمون فرض: برآوردهایی و آزمون فرض دو روشنی هستند که برای استنباط درمورد پارامترهای مجهول دو جمعیت به کار می روند.

۵- متغیر: ویژگی یا خاصیت یک فرد، شئ و یا موقعیت است که شامل یک سری از مقادیر با دسته بندیهای متناسب است. قد، وزن، گروه خونی و جنس نمونه هایی از متغیر هستند. انواع متغیر می تواند کمی و کیفی باشد.

۶- داده های کمی مانند قد، وزن یا سن درجه بندی می شوند و به همین دلیل قابل اندازه گیری می باشند. داده های کمی نیز خود به دو دسته دیگر تقسیم می شوند:

الف: داده های فاصله ای (**Interval data**)

ب: داده های نسبتی (**Ratio data**)

۷-داده های فاصله ای: به عنوان مثال داده هایی که متغیر IQ (ضریب هوشی) را در پنج نفر توصیف می کنند عبارتند از: ۱۱۰، ۸۰، ۷۵، ۹۷ و ۱۱۷، چون این داده ها عدد هستند پس داده های ما کمی اند اما می دانیم که IQ نمی تواند صفر باشد و صفر در اینجا فقط مبنایی است تا سایر مقادیر IQ در فاصله ای منظم از صفر و یکدیگر قرار گیرند پس این داده ها فاصله ای اند.

۸-داده های نسبتی: داده هایی هستند که با عدد نوشه می شوند اما صفر آنها واقعی است. اکثریت داده های کمی این گونه اند و حقیقتاً دارای صفر هستند. به عنوان مثال داده هایی که متغیر طول پاره خط بر حسب سانتی متر را توصیف می کنند عبارتند از: ۲۰، ۱۵، ۳۵، ۸ و ۲۳، چون این داده ها عدد هستند پس داده های ما کمی اند و چون صفر در اینجا واقعاً وجود دارد این داده نسبتی تلقی می شوند.

۹-داده های کیفی مانند جنس، گروه خونی یا ملیت فقط دارای نوع هستند و قابل بیان با استفاده از واحد خاصی نیستند. داده های کیفی خود به دو دسته دیگر تقسیم می شوند:

الف: داده های اسمی (Nominal data)

ب: داده های رتبه ای (Ordinal data)

۱۰-داده های رتبه ای: Ordinal مانند کیفیت درسی یک دانش آموز (ضعیف، متوسط و قوی) و یا رتبه بندی هتل ها (یک ستاره، دو ستاره و ...)

۱۱-داده های اسمی (nominal) که مربوط به متغیر یا خواص کیفی مانند جنس یا گروه خونی است و بیانگر عضویت در یک گروها category خاص می باشد. (داده مقوله ای)

۱۲-متغیر تصادفی گسسته و پیوسته: به عنوان مثال تعداد تصادفات جاده ای در روز یک متغیر تصادفی گسسته است ولی انتخاب یک نقطه به تصادف روی دایره ای به مرکز مبدأ مختصات و شعاع ۳ یک متغیر تصادفی پیوسته است.

۱۳-گروه: یک متغیر می تواند به لحاظ بررسی یک ویژگی خاص در یک گروه و یا دو و یا بیشتر مورد بررسی قرار گیرد. نکته ۱: دو گروه می تواند وابسته و یا مستقل باشد. دو گروه وابسته است اگر ویژگی یک مجموعه افراد قبل و بعد از وقوع یک عامل سنجدیده شود. مثلا میزان رضایت شغلی کارکنان قبل و بعد از پرداخت پاداش و همچنین اگر در مطالعات تجربی افراد از نظر برخی ویژگی ها در یک گروه با گروه دیگر همسان شود.

۱۴-جامعه نرمال: جامعه ای است که از توزیع نرمال تبعیت می کند.

۱۵-توزیع نرمال: یکی از مهمترین توزیع ها در نظریه احتمال است. و کاربردهای بسیاری در علوم دارد.

فرمول این توزیع بر حسب دو پارامتر امید ریاضی و واریانس بیان می شود. منحنی رفتار این تابع تا حد زیادی شبیه به زنگ های کلیسا می باشد. این منحنی دارای خواص بسیار جالبی است برای مثال نسبت به محور عمودی متقارن می باشد، نیمی از مساحت زیر منحنی بالای مقدار متوسط و نیمه دیگر در پایین مقدار متوسط قرار دارد و اینکه هرچه از طرفین به مرکز مختصات نزدیک می شویم احتمال وقوع بیشتر می شود.

سطح زیر منحنی نرمال برای مقادیر متفاوت مقدار میانگین و واریانس فراگیری این رفتار آنقدر زیاد است که دانشمندان اغلب برای مدل کردن متغیرهای تصادفی که با رفتار آنها آشنایی ندارند، از این تابع استفاده می‌کنند. به عنوان مثال در یک امتحان درسی نمرات دانش آموزان اغلب اطراف میانگین بیشتر می‌باشد و هر چه به سمت نمرات بالا یا پایین پیش برویم تعداد افرادی که این نمرات را گرفته اند کمتر می‌شود. این رفتار را سهولت می‌توان با یک توزیع نرمال مدل کرد.

اگر یک توزیع نرمال باشد مطابق قضیه چی بی شف ۲۶,۶۸٪ مشاهدات در فاصله میانگین، مثبت و منفی یک انحراف معیار قرار دارد. و ۴۴,۹۵٪ مشاهدات در فاصله میانگین، مثبت و منفی دو انحراف معیار قرار دارد. و ۷۳,۹۹٪ مشاهدات در فاصله میانگین، مثبت و منفی سه انحراف معیار قرار دارد.

نکته ۱: واضح است که داده‌های رتبه‌ای دارای توزیع نرمال نمی‌باشند.

نکته ۲: وقتی داده‌ها کمی هستند و تعداد نمونه نیز کم است تشخیص نرمال بودن داده‌ها توسط آزمون کولموگروف – اسمیرنوف مشکل خواهد شد.

۱۶- آزمون پارامتریک: آزمون‌های پارامتریک، آزمون‌های هستند که توان آماری بالا و قدرت پرداختن به داده‌های جمع آوری شده در طرح‌های پیچیده را دارند. در این آزمون‌ها داده‌ها توزیع نرمال دارند. (مانند آزمون تی).

۱۷- آزمون‌های غیرپارامتری: آزمون‌هایی می‌باشند که داده‌ها توزیع غیر نرمال داشته و در مقایسه با آزمون‌های پارامتری از توان تشخیصی کمتری برخوردارند. (مانند آزمون من – ویتنی و آزمون کروسکال و والیس)

نکته ۳: اگر جامعه نرمال باشد از آزمون‌های پارامتریک و چنانچه غیر نرمال باشد از آزمون‌های غیر پارامتری استفاده می‌نمائیم.

نکته ۴: اگر نمونه بزرگ باشد، طبق قضیه حد مرکزی جتی اگر جامعه نرمال نباشد می‌توان از آزمون‌های پارامتریک استفاده نمود.

## ۱۶. معیار Matthews Correlation Coefficient (MCC) چیست و در چه

### جاهای استفاده میشود؟

ضریب همبستگی (MCC) یا ضریب phi در یادگیری ماشین به عنوان معیار کیفیت طبقه بندی های باینری (دو کلاسه) استفاده می شود.

ضریب مثبت و منفی درست و نادرست را در نظر می گیرد و به طور کلی به عنوان یک معیار متعادل در نظر گرفته می شود که حتی اگر کلاس ها از اندازه های بسیار متفاوت باشند می توانند مورد استفاده قرار گیرد. MCC در اصل ضریب همبستگی بین طبقه بندی باینری مشاهده و پیش بینی شده است. مقداری بین  $-1$  و  $+1$  برمی گرداند. ضریب  $+1$  یک پیش بینی کامل را نشان می دهد،  $0$  هیچ چیزی بهتر از پیش بینی تصادفی نیست و  $-1$  اختلاف کلی بین پیش بینی و مشاهده را نشان می دهد. با این حال، اگر MCC برابر با  $-1$ ،  $0$  یا  $+1$  نباشد، این یک شاخص قابل اعتماد نیست که یک پیش بینی شبیه حدس تصادفی است زیرا MCC به مجموعه داده وابسته است. MCC برای جدول احتمالی  $2 \times 2$  با آمار مربع کای ارتباط نزدیک دارد.

$$|MCC| = \sqrt{\frac{\chi^2}{n}}$$

که  $n$  تعداد کل مشاهدات است.

در حالی که روش کاملی برای توصیف ماتریس سردرگمی مثبت و منفی توسط یک عدد وجود ندارد، اما ضریب همبستگی متیوها به عنوان یکی از بهترین معیارها در نظر گرفته می شود. اقدامات دیگر، مانند نسبت پیش بینی های صحیح (که اصطلاحاً به آنها اصطلاح می شود)، وقتی دو کلاس از اندازه های بسیار متفاوت باشند، مفید نیستند. به عنوان مثال، اختصاص دادن هر شی به مجموعه بزرگتر به نسبت بالایی از پیش بینی های صحیح دست می یابد، اما به طور کلی یک طبقه بندی مفید نیست.

MCC را می توان مستقیماً از ماتریس سردرگمی با استفاده از فرمول محاسبه کرد:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

در این معادله ،  $TP$  تعداد مثبت واقعی ،  $TN$  تعداد منفی واقعی ،  $FP$  تعداد مثبت کاذب و  $FN$  تعداد منفی کاذب است. اگر هر یک از چهار جمع مخرج صفر باشد ، مخرج را می توان خودسرانه روی یک تنظیم کرد. این منجر به ضریب همبستگی متیو صفر می شود که می تواند مقدار محدود کننده درستی باشد.

MCC را می توان با فرمول محاسبه کرد:

$$MCC = \sqrt{PPV \times TPR \times TNR \times NPV} - \sqrt{FDR \times FNR \times FPR \times FOR}$$

با استفاده از ارزش اخباری مثبت ، نرخ مثبت واقعی ، نرخ منفی واقعی ، ارزش اخباری منفی ، نرخ کشف کاذب ، نرخ منفی کاذب ، نرخ مثبت کاذب و میزان حذف کاذب.

فرمول اصلی ارائه شده توسط متیوز این بود:

$$N = TN + TP + FN + FP$$
$$S = \frac{TP + FN}{N}$$
$$P = \frac{TP + FP}{N}$$
$$MCC = \frac{TP/N - S \times P}{\sqrt{PS(1 - S)(1 - P)}}$$

این برابر با فرمول فوق است. به عنوان یک ضریب همبستگی، ضریب همبستگی **Matthews** میانگین هندسی ضرایب رگرسیون مسئله و دوگانه آن است. ضرایب رگرسیون مولفه‌های ضریب همبستگی ماتیوز و آمار **Youden** هستند.

برخی از دانشمندان ادعا می‌کنند که ضریب همبستگی متیو بهترین آموزنده برای تعیین کیفیت پیش‌بینی طبقه‌بندی باینری در زمینه ماتریس سردرگمی است.

ضریب همبستگی **Matthews** به حالت چند کلاسه تعمیم داده شده است.

$$MCC = \frac{\sum_k \sum_l \sum_m C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl}) (\sum_{k' \neq k} \sum_{l'} C_{k'l'})} \sqrt{\sum_k (\sum_l C_{lk}) (\sum_{k' \neq k} \sum_{l'} C_{l'k'})}}$$

وقتی بیش از دو برچسب وجود داشته باشد، **MCC** دیگر بین  $-1$  و  $+1$  نخواهد بود. در عوض حداقل مقدار بسته به توزیع واقعی بین  $-1$  تا  $+1$  خواهد بود. حداکثر مقدار همیشه  $+1$  است.

با تعریف متغیرهای میانی این فرمول به راحتی قابل درک است:

- $t_k = \sum_i C_{ik}$
- $p_k = \sum_i C_{ki}$
- $c = \sum_k C_{kk}$
- $s = \sum_i \sum_j C_{ij}$

تعداد دفعاتی که کلاس  $k$  واقعاً رخ داده است ،

تعداد دفعات پیش‌بینی کلاس  $k$  ،

تعداد کل نمونه‌ها به درستی پیش‌بینی شده است ،

تعداد کل نمونه‌ها این اجازه می‌دهد فرمول به صورت زیر بیان شود:

$$MCC = \frac{cs - \vec{t} \cdot \vec{p}}{\sqrt{s^2 - \vec{p} \cdot \vec{p}} \sqrt{s^2 - \vec{t} \cdot \vec{t}}}$$

## بحث و نتیجه گیری

ما نتایج حاصل از تکنیک های مختلف یادگیری بدون نظارت بر مجموعه داده های بیماری قلبی UCL را تعیین کرده ایم. خوش بندی سلسله مراتبی مجموعه داده ما از رویکرد GMM مناسب تر است). با این حال ، حتی در این قوانین با اعتماد به نفس بالا ، باید سعی کنیم کمی دقیق باشیم که چه چیزی باعث ایجاد این فرکانس بالا یا اطمینان به قوانین می شود: آیا این سوگیری مجموعه داده است یا پدیده هایی در دنیای واقعی است؟

## مأخذ

**\_آمار ریاضی و کاربردهای آن \_ {جان فرونڈ}**

**\_داده کاوی و مفاهیم \_ {ژیاوه هان}**

**-دانشنامه ویکی پدیا**

