گزارش تمرین سری دوم

درس داده کاوی

جناب آقای دکتر فراهانی وجناب آقای دکتر خرد پیشه

دستیار آموزشی جناب آقای علی شریفی

زینب خسروی۹۹۴۲۲۰۶۷

مقدمه

داده کاوی به بررسی و تجزیه وتحلیل مقادیر زیادی از داده ها به منظورکشف اطلاعاتی که از طریقالگوریتم ها با هدف خاصی ارائه می شود گاهی تکنیک های داده کاوی می تواند با استفاده از داده های موجود زمینه های ابتلا به بیماری پیشگیری و درمان بیماری ها را در پزشکی فراهم کند و در تشخیص به موقع کمک کند .

هدف استفاده از نتایج حاصل از داده کاوی جهت پیش بینی دقیق تر و تصمیم گیری موثر تر در درمان بیماری هست .

ابتدا داده ها را فراخوانی کردم میانگین استاندارد وماکسیمم ومینیمم داده ها را بدست آوردم واطلاعات داده ها را استخراج کردم تا داده ها را اول بشناسیم بعد کریلیشن بین ویزگی ها را گرفتم که چه رابطه ای می توان بینشون پیدا کرد بعد داده های پرت حذف کردم بعد با استفاده از نمودار های دیگر سعی کردم چه اطلاعات دیگری می توانم بدست بیاورم.بررسی کردم تعداد نمونه ها را در هر کلاس متوازن هست .

داده ها را به نسبت ۸۰به ۲۰ به داده های آموزشی و تست تقسیم کردم قاعده بیز بیان کردم

knn,svmبیان کردم و الگویتم اجرا کردم توضیحی برای همسایگی ها بیان کردم

تفاوت کلاس بندی های پارامتری و غیر پارامتری را توضیح دادم .

نتایج نشان داده در کلاسبندی ها از لحاظ متغییردر مورد بیماری قلبی استرس و جنسیت و سن و فشار خون وقند خون به ترتیب بیشترین اهمیت در پیش بینی پاسخ به بیماری دارند.

ما داده ای داریم که اگر بیماران مبتلا به بیماری قلبی هستند یا نه بر اساس ویژگی های موجود در آن طبقه بندی شوند

ما سعی خواهیم کرد از این داده ها برای ایجاد مدلی استفاده کنیم که بیمار را به این بیماری مبتلا می کند یا خیر.

ابتدا با پکیج پانداس به بررسی دیتا ست می پردازیم

```
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
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
data=pd.read_csv('/content/drive/My Drive/heart.csv')
```

+ Code + Text

شمارش ومیانگین و استانداردومینیمم و ماکسیمم ویزگی ها را بدست آورده

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

داده صدا میکنیم

```
data
```

	age	sex	cp	trestbps	chol	fbs	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

اطلاعات داده استخراج کنیم

```
[9]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

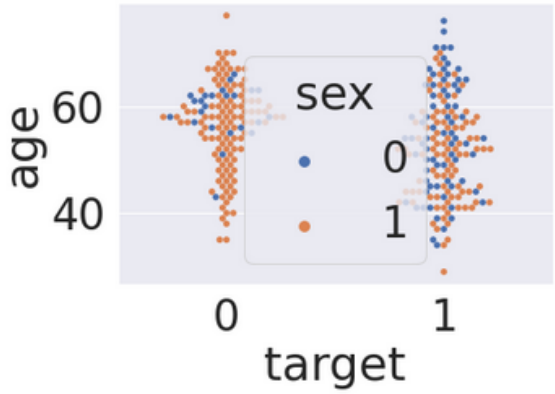
کریلیشن همه ویژگی ها را در نمودار هیت مپ نشان داده است

```
[ ]  plt.figure(figsize=(20,10))
     sns.heatmap(data.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f10aafb7b10>
```


```
[148] print(data['sex'].value_counts())
      sns.swarmplot(data['target'],data['age'],hue=data['sex'])
```

```
1    207
0     96
Name: sex, dtype: int64
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following vari
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fc57b4fc210>
```



در اینجا سن و جنسیت چه تاثیری در بیماری قلبی دارند یا نه را نشان می دهد.

نوع داده ها را بررسی می کنیم

```
[10] data.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

داده های نال یا پوچ از بین می بریم

```
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

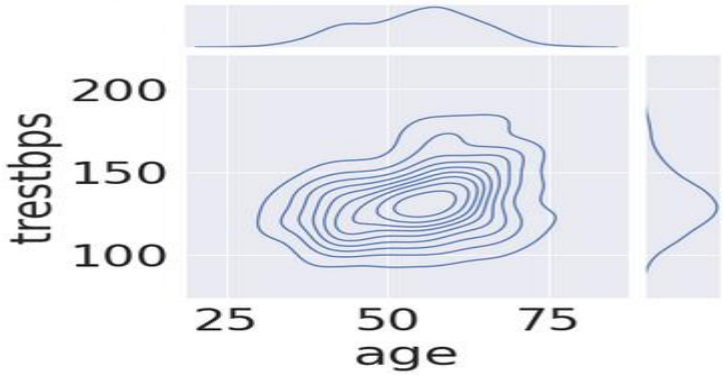
دیسکریب همه ویژگی ها را بدست آورده یعنی تعداد و میانگین و واریانس و مینیمم و ماکسیمم وچهارک ها

```
[150] data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

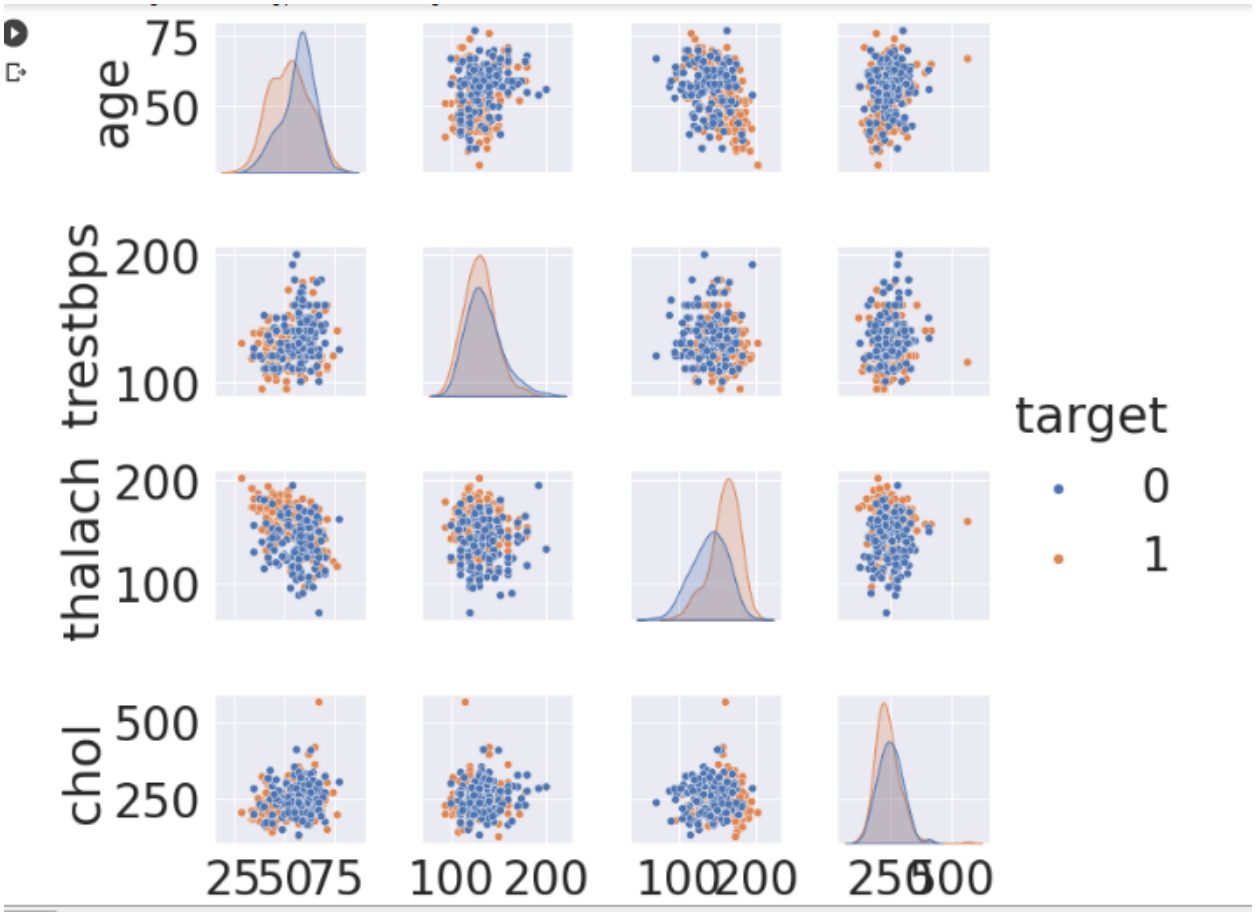
```
[151] sns.jointplot(data.age, data.trestbps, kind='kde');
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:1668: UserWarning: Tight
  f.tight_layout()
```



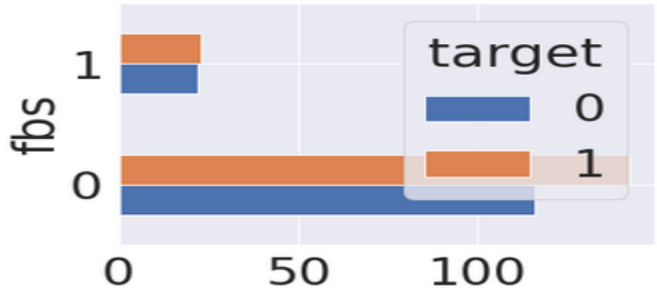
کواریانس یا کوریلیشن فشار خون و سن بدست آوردیم که میشه لایک لیهود شون که احتمال رخداد کلاس ها برابر هستند

```
sns.pairplot(data[['age','trestbps','thalach','chol','target']],hue='target',size=2.5);
```



توزیع داده های سن و فشار خون و چربی خون والکل برای بیماری قلبی مبتلا شوند یانه را نشان می دهد

```
[ ]  pd.crosstab(data.fbs,data.target).plot(kind='barh');
```



قند خون و بیماری قلبی

بررسی می کنیم تعدادنمونه ها در هر کلاس متوازن هست؟

برای نرمالایز کردن نباید دامنه تغییرات زیاد باشد.چون کارمون خراب می کند .یه موقع هست دامنه تغییرات بین ۰-۱هست یه موقع هست دامنه تغییرات بین یک میلیون تا منفی یک میلیون هست که باید نرمال سازی یا استاندارد کنیم.که باید ستون ها رامنهای میانگین تقسیم بر استاندارد ش کنیم.که نوع ستون ها در این جا float64 ,int64بود.که در یک لوپ انداختیم تا نرمالایز انجام شود.

```
for cols in data.columns:
    if data[cols].dtype=='int64' or data[cols].dtype == 'float64':
        data[cols] = ((data[cols] - data[cols].mean()) /(data[cols].std()))
```

```
[13] data.shape

     (303, 14)
```

در ابتدا داده ها این مقدار سطر دستون بودند.

بعد داده های پرت بررسی میکنیم وحذف میکنیم داده های پرت به سه طریق حذف میکنیم

فاصله چهارکی

روش های کلاس ترینگ

دادههایی که در شعاع سه برابر انحراف معیار حول مرکز میانگین وجود ندارند را داده های پرت در نظر بگیرید.

:

```
[84] for cols in data.columns:
         if data[cols].dtype == 'int64' or data[cols].dtype == 'float64':
            upper_range = data[cols].mean() + 3 * data[cols].std()
            lower_range = data[cols].mean() - 3 * data[cols].std()

            indexs = data[(data[cols] > upper_range) | (data[cols] < lower_range)].in
            data=data.drop(indexs)
```

```
data.shape
```

```
(287, 14)
```

در این جا از روش آخری استفاده کردیم برایش شرط گذاشتیم و دراپ کردیم.و بعددیتا شیپ که زدیم

از۳۰۳ به۲۸۷ رسید.

دیتاست را با نسبت ۸۰به۲۰ به دو بخش داده های آموزشی وداده های تست تقسیم بندی کردم با پکیج

Sklearn

```
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_trainفیچر های داده ترین

y_trainلیبل های داده ترین

X_test فیچر های داده تست

y_test لیبل های داده تست

عنوان یا ویزگی داده ها

```
[ ] 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

قانون بیز

بیشترین احتمال کلاسهای موجود که ارائه می شود توسط بردار ویژگی به صورت های مختلف رفتار می کند یک متغییر تصادفی با احتمال تعریف می شود که هر چه از مرکز دورتر از دو سمت باشد شانس کمتری دارد را توزیع گوسین یا نرمال می نامند .تابع چگالی احتمال بر توزیع حاکم هست با احتمال داده ها کار می کنیم میانگین و واریانس یا مورد انتظار با هم برابر هستند.زنگوله ای شکل هستند.احتمال تعلق یک سمپل به کلاس می سنجیم سمپل تصادفی انتخاب می کنیم برای همه کلاس ها و آن که احتمال بیشتری دارد را انتخاب می کنیم.

وقتی می خواهیم بدانیم از چه توزیعی پیروی می کند ولی پارامتر هایش را نمی دانیم با قانون بیز داده هایی که داریم لایکلیهود و احتمال پیشین حساب می کنیم ومقایسه می کنیم وتصمیم می گیریم که نیاز به توابع احتمال داریم.که لایکلیهود احتمال این ها را می سنجدکه می خواهیم با استفاده از داده هایی که داریم اگر داده جدید به ما داده شد چقدر احتمال دارد که از این توزیع آمده است

از روش های پارامتری و شبیه پارامتری ونا پارامتری استفاده میکنیم.

در روش پارامتری فرض میکنیم تابع توزیع نرمال هست دنبال پارامتر های میانگین و واریانس هستیم پارامتر ها را طوری تنظیم می کنیم که تابع لایکلیهود بتواند توزیع داده ها را به شکل خوبی نشان دهد

لایکلیهود پارامتر ها را طوری تنظیم می کنیم که احتمال رخداد سمپل های مربوط به کلاس بیشترین مقدار ممکن باشد.اگر فرض کنیم توزیع نرمال هست میانگین نمونه ها تخمینی از میانگین جامعه خواهد بود وواریانس نمونه ها هم تخمینی از واریانس جامعه خواهد بود.

درروش سمی پارامتری فرض میکنیم تابع توزیع احتمال کلاسی بدست می آوریم که تلفیقی از چند توزیع نرمال هست توزیع ها را باهم جمع می کنیم ممکنه وزن دار باشد که از روش تخمین استفاده می کنیم

در روش نا پارامتری نمی دانیم از چه توزیعی پیروی می کند پارامتر ها را هم نداریم می خواهیم لایکلیهود تخمین بزنیم بدون اینکه از توزیع شناخت داشته باشیمکه یکی از راه هاش کشیدن هیستگرام هست فراوانی داده ها را در ارتفاع میله نشان می دهیم راه دیگر نزدیکترین همسایه هست که نزدیکترین داده مورد نظر به کدام کلاس نزدیکتر هست لایکلیهود بدست می آوریم ضعف این روش این هست که اگر خیلی نزدیک یا خیلی دور باشد تخمین خوبی برای لایکلیهود نیست که در این روش های نا پارامتری تابع سراسری دا گلوبال نداریم ونگاه مون یک نگاه محلی هست روش دیگری هم هست به نام پارزن ویندوز که اطراف داده ها پنجره ای در نظر میگیرد شعاع محدود می کندبعد فیکس می کند داده هایی که در آن افتاده تخمینی برای لایکلیهود هست

حالا برای همه روش های ناپارامتری و سمی پارامتری و پارامتری لایکلیهود تخمین زدیم می خواهیم در موردش تصمیم گیری کنیم سراغ قانون بیز می رویم مقایسه می کنیم که برای کدام کلاس هست

نیو بیز وقتی استفاده می شود که یک اصل مشترک یا ویژگی خاص برای کلاس هست در یک محیط یادگیری تحت نظارت کارامد هست برای تخمین پارامتر هاش از روی حد اکثر احتمال استفاده میکند مزیت این هست که فقط به تعداد کمی از داده های آموزش برای طبقه بندی نیاز دارداز نظر محاسباتی کارامد اجرا آسان قدرتمند سریع ودقیق هست.

مولتی نومیال نیو بیز هم یک روش نظارت شده هست با یک یا چند برچسب کلاس از یک کلاس ثابت یا از پیش تعریف شده ظبقه بندی می کند از تک تک داده ها ی موجود در ویژگی ها استفاده می کند وقتی که ابعاد ویژگی ها بالا هست مشکلات از نظر عملکرد طبقه بندی وسرعت پایین محاسبات هست

برنولی دارای پیچیدگی زمانی مشابه مدل چند جمله ای است مدل برنولی به عنوان کسری از داده های کلاس تخمین می زند از اطلاعات باینری استفاده می کند تعدادی هم نا دیده گرفته می شود برای همین همطبقه بندی در داده های طولانی اشتباهات زیادی دارند

چهار ویژگی را در نظر گرفتیم از دیتاست استفاده می کنیم از پایه پیاده سازی می کنیم

```
data["trestbps"].describe()
```

```
count    287.000000
mean      -0.034175
std        0.968852
min       -2.145254
25%       -0.662770
50%       -0.092585
75%        0.477601
max        2.758344
Name: trestbps, dtype: float64
```

```
[ ] data["chol"].describe()
```

```
count    287.000000
mean      -0.034001
std        0.885974
min       -2.320322
25%       -0.680369
50%       -0.101562
75%        0.535126
max        2.850354
Name: chol, dtype: float64
```

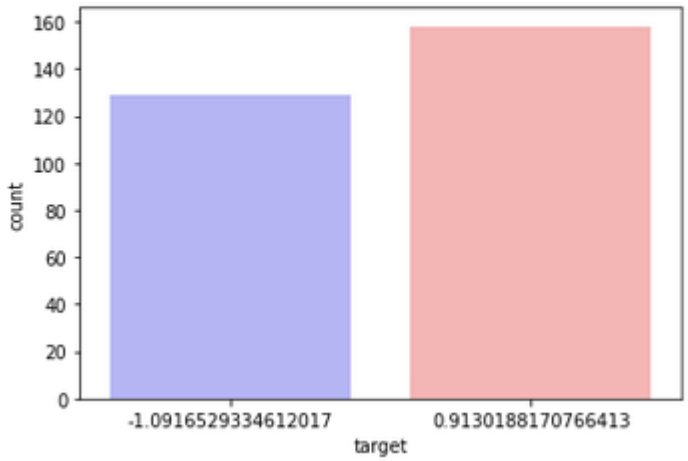
```
[ ] data["target"].describe()
```

```
count    287.000000
mean       0.011964
std        0.998948
min       -1.091653
25%       -1.091653
50%        0.913019
75%        0.913019
max        0.913019
Name: target, dtype: float64
```

```
[ ] data.target.value_counts()
```

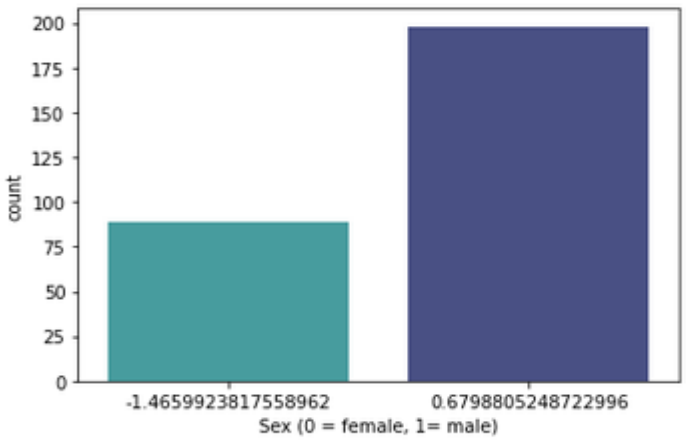
```
 0.913019    158
-1.091653    129
Name: target, dtype: int64
```

```
sns.countplot(x="target", data=data, palette="bwr")
plt.show()
```



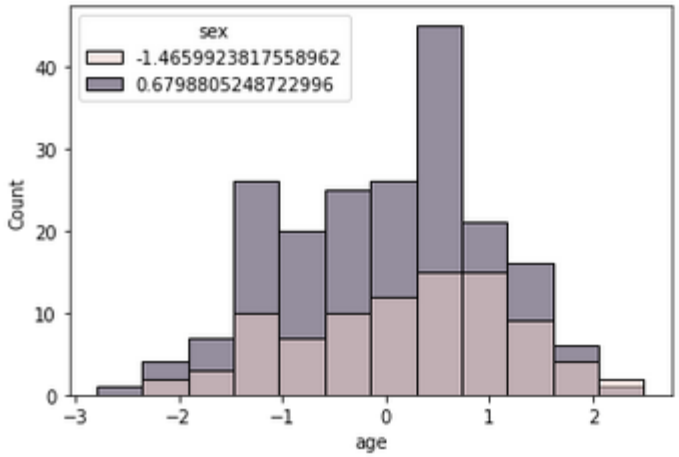
تعدادمبتلا شدن به بیماری قلبی براساس جنسیت بررسی می کنیم که چه تعداد زن هاو چه تعدادمردها مبتلا شدند

```
sns.countplot(x='sex', data=data, palette="mako_r")
plt.xlabel("Sex (0 = female, 1= male)")
plt.show()
```



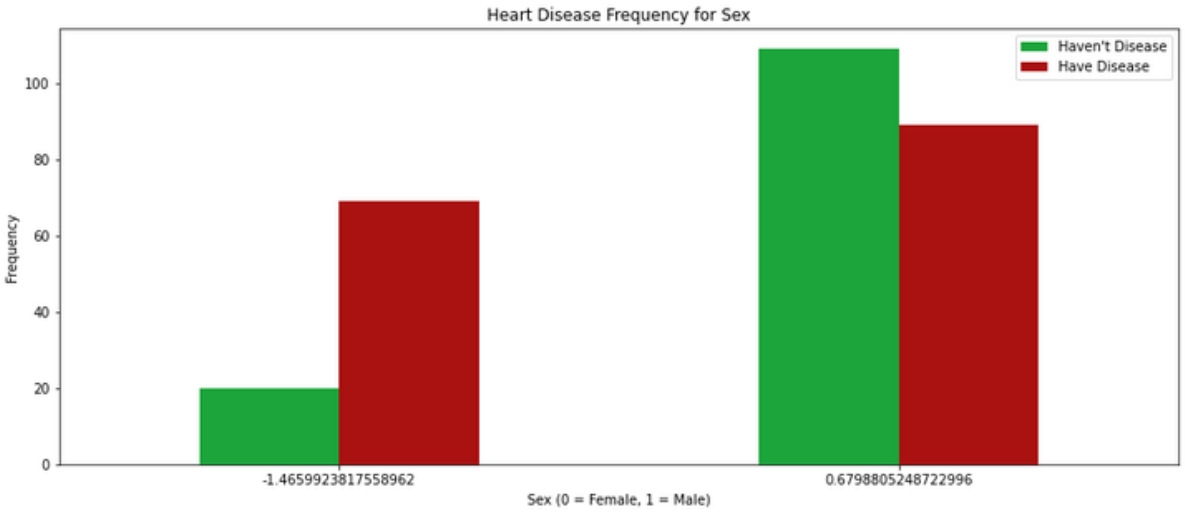
```
] sns.histplot(data=data, x="age", hue="sex")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faa583f9f50>
```



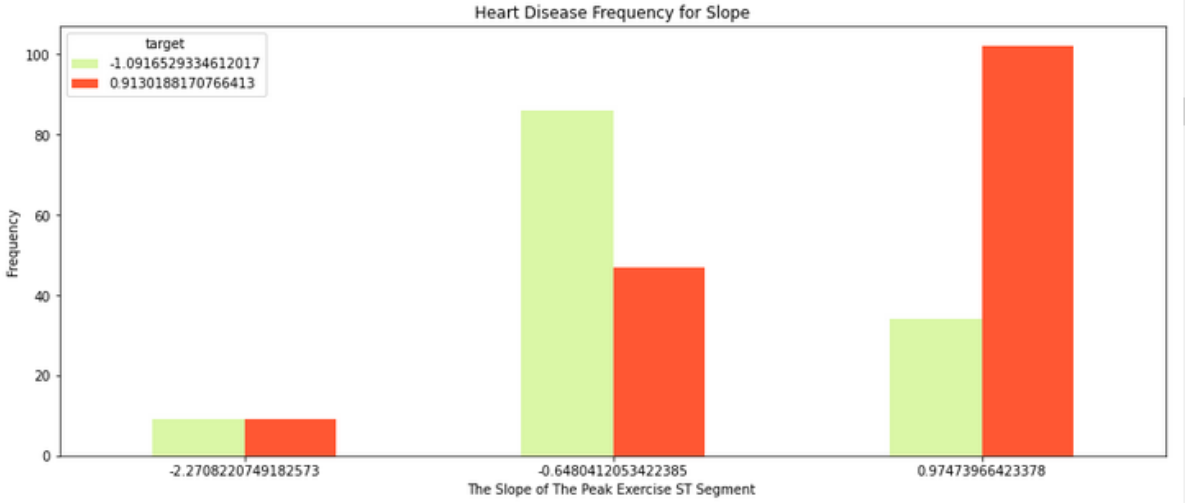
توزیع سن براساس جنسی (۰ = زن ، ۱ = مرد

```
[ ] pd.crosstab(data.sex,data.target).plot(kind="bar",figsize=(15,6),color=['#1CA53B','#AA111
    plt.title('Heart Disease Frequency for Sex')
    plt.xlabel('Sex (0 = Female, 1 = Male)')
    plt.xticks(rotation=0)
    plt.legend(["Haven't Disease", "Have Disease"])
    plt.ylabel('Frequency')
    plt.show()
```



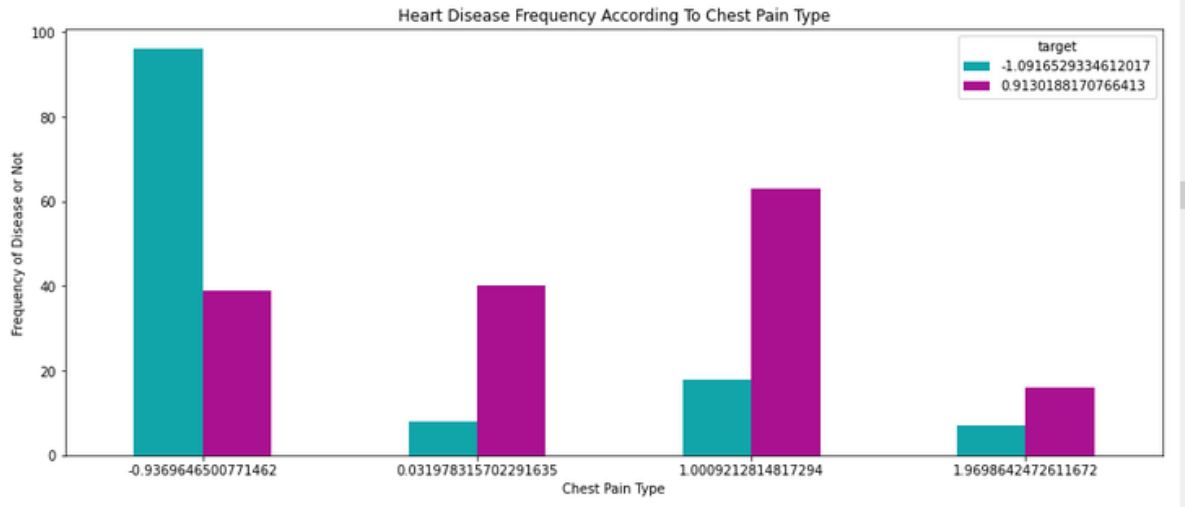
زن ها بیشتر مبتلا به بیماری قلبی می شوند یا مردان از روی نمودار مشخص هست. بر اساس جنسیت

```
[ ] pd.crosstab(data.slope,data.target).plot(kind="bar",figsize=(15,6),color=['#DAF7A6','#FF5
    plt.title('Heart Disease Frequency for Slope')
    plt.xlabel('The Slope of The Peak Exercise ST Segment ')
    plt.xticks(rotation = 0)
    plt.ylabel('Frequency')
    plt.show()
```



کسانی ورزش می کنندشیب نشان میدهد که بیماری قلبی دارند یا نه

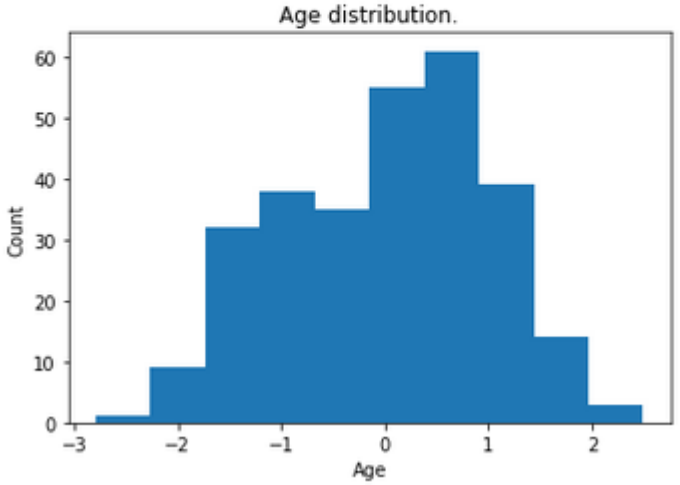
```
[ ] pd.crosstab(data.cp,data.target).plot(kind="bar",figsize=(15,6),color=['#11A5AA','#AA1190
    plt.title('Heart Disease Frequency According To Chest Pain Type')
    plt.xlabel('Chest Pain Type')
    plt.xticks(rotation = 0)
    plt.ylabel('Frequency of Disease or Not')
    plt.show()
```



بیماری قلبی براساس نوع درد

```
[ ] plt.hist(data['age'])
    plt.xlabel('Age')
    plt.ylabel('Count')
    plt.title('Age distribution.')
```

```
Text(0.5, 1.0, 'Age distribution.')
```



مقدار از دست رفته نداریم

```
[ ] a = pd.get_dummies(data['cp'], prefix = "cp")
    b = pd.get_dummies(data['thal'], prefix = "thal")
    c = pd.get_dummies(data['slope'], prefix = "slope")
```

```
[ ] frames = [data, a, b, c]
    data = pd.concat(frames, axis = 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

از آنجا که 'slope' و 'thal'، 'cp' متغیرهای طبقه ای هستند ، ما آنها را به متغیرهای ساختگی تبدیل خواهیم کرد

```
[ ] data = data.drop(columns = ['cp', 'thal', 'slope'])
    data.head()
```

	age	sex	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca	target	cp_-0.9369646500771462	cp_0.03197831570229163 5	cp_1.00092
0	0.950624	0.679881	0.762694	-0.255910	2.390484	-1.004171	0.015417	-0.69548	1.085542	-0.713249	0.913019	0	0	
1	-1.912150	0.679881	-0.092585	0.072080	-0.416945	0.897478	1.630774	-0.69548	2.119067	-0.713249	0.913019	0	0	
2	-1.471723	-1.465992	-0.092585	-0.815424	-0.416945	-1.004171	0.975900	-0.69548	0.310399	-0.713249	0.913019	0	1	
3	0.179877	0.679881	-0.662770	-0.198030	-0.416945	0.897478	1.237849	-0.69548	-0.206364	-0.713249	0.913019	0	1	
4	0.289984	-1.465992	-0.662770	2.078611	-0.416945	0.897478	0.582975	1.43311	-0.378618	-0.713249	0.913019	1	0	

بعد متغییر های ساختگی دراپ می کنیم

```
from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_si:
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
```

```
Number of mislabeled points out of a total 75 points : 4
```

از قاعده گوسین بیز استفاده می کنیم فرض میکنیم داده ها توزیع نرمال دارند با توزیع نرمال کار می کنید.

```
[ ] data.groupby('target').mean()
```

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
-1.091653	0.247307	0.347187	-0.448738	0.093499	0.063705	-0.003448	-0.134425	-0.462455	0.476070	0.40654	-0.333549	0.393514	0.374184
0.913019	-0.208284	-0.257241	0.375401	-0.138416	-0.113773	-0.026037	0.127190	0.390105	-0.385622	-0.38625	0.307140	-0.447113	-0.305334

این ویزگی همان کلاس ها مون هستند که بیماری قلبی دارند یا ندارند

برای ساختن مدل از رگرسیون لاجستیک استفاده میکنیم.چون مقادیر مون بین صفر ویک نیست

ما می توانیم از کتابخانه Sklearnاستفاده کنیم یا انکه خودمان توابع بنویسیم اول توابع می نویسیم

```
[ ] y = data.target.values
    x_data = data.drop(['target'], axis = 1)
```

```
[ ] x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values
```

```
[ ] x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)
```

```
[ ] x_train = x_train.T
    y_train = y_train.T
    x_test = x_test.T
    y_test = y_test.T
```

```
[ ] def initialize(dimension):

        weight = np.full((dimension,1),0.01)
        bias = 0.0
        return weight,bias
```

```python
def sigmoid(z):

    y_head = 1/(1+ np.exp(-z))
    return y_head
```

```python
def forwardBackward(weight,bias,x_train,y_train):
    # Forward

    y_head = sigmoid(np.dot(weight.T,x_train) + bias)
    loss = -(y_train*np.log(y_head) + (1-y_train)*np.log(1-y_head))
    cost = np.sum(loss) / x_train.shape[1]

    # Backward
    derivative_weight = np.dot(x_train,((y_head-y_train).T))/x_train.shape[1]
    derivative_bias = np.sum(y_head-y_train)/x_train.shape[1]
    gradients = {"Derivative Weight" : derivative_weight, "Derivative Bias" : derivative_

    return cost,gradients
```

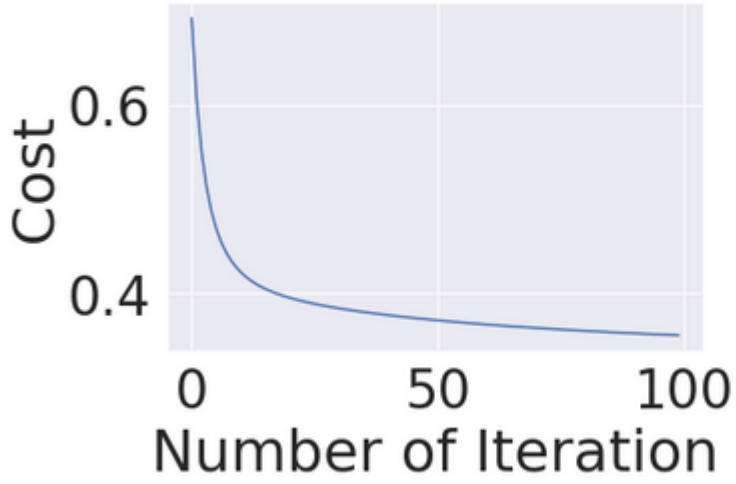
```python
[ ] def update(weight,bias,x_train,y_train,learningRate,iteration) :
        costList = []
        index = []

        #for each iteration, update weight and bias values
        for i in range(iteration):
            cost,gradients = forwardBackward(weight,bias,x_train,y_train)
            weight = weight - learningRate * gradients["Derivative Weight"]
            bias = bias - learningRate * gradients["Derivative Bias"]

            costList.append(cost)
            index.append(i)

        parameters = {"weight": weight,"bias": bias}

        print("iteration:",iteration)
        print("cost:",cost)

        plt.plot(index,costList)
        plt.xlabel("Number of Iteration")
        plt.ylabel("Cost")
        plt.show()

        return parameters, gradients
```

```
[ ] def predict(weight,bias,x_test):
        z = np.dot(weight.T,x_test) + bias
        y_head = sigmoid(z)

        y_prediction = np.zeros((1,x_test.shape[1]))

        for i in range(y_head.shape[1]):
            if y_head[0,i] <= 0.5:
                y_prediction[0,i] = 0
            else:
                y_prediction[0,i] = 1
        return y_prediction
```

```
[ ] def logistic_regression(x_train,y_train,x_test,y_test,learningRate,iteration):
        dimension = x_train.shape[0]
        weight,bias = initialize(dimension)

        parameters, gradients = update(weight,bias,x_train,y_train,learningRate,iteration)

        y_prediction = predict(parameters["weight"],parameters["bias"],x_test)

        print("Manuel Test Accuracy: {:.2f}%".format((100 - np.mean(np.abs(y_prediction - y_test))*100)))
```

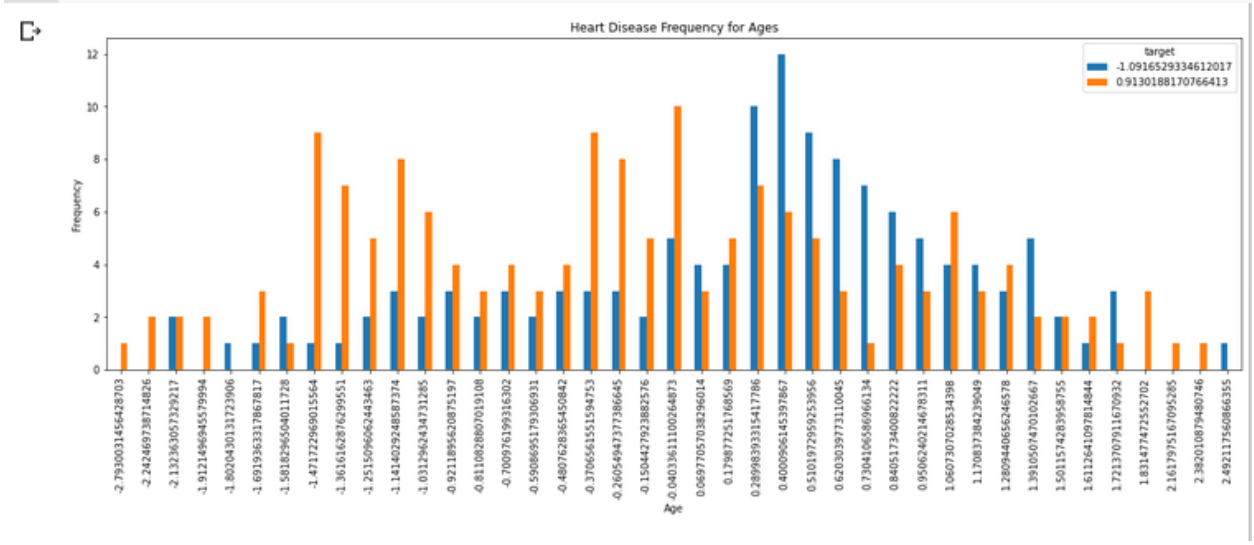
```
logistic_regression(x_train,y_train,x_test,y_test,1,100)
```

```
iteration: 100
cost: 0.3556455127179851
```




```
Manuel Test Accuracy: 87.93%
```

```
pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('heartDiseaseAndAges.png')
plt.show()
```



میتوانیم از ناپارامتری استفاده کنیم که باهیستوگرام وروش پارزن ویندوز اطراف داده ها پنجره در نظر بگیریم یا محدوده ای در نظر بگیریم شعاع محدوده فیکس کنیم وبرای تخمین لایکلیهود استفاده کنیم.

ولی خود سوال به ما گفته شده گوسین بیز استفاده شود

در روش پارامتری ایده اصلی این هست که مجموعه ای از پارانتر ها ثابت هستند که یک مدل احتمالاتی تعیین می کنند از توزیع نرمال به صورت تقریب استفاده میکنیم این روش وابسته به جمعیت هست که دو پارامتر توزیع نرمال میانگین و واریانس هست

در روش نا پارامتری مجموعه ای از پارامتر ها ثابت نشده اند وتوزیعی نیست که استفاده کنیم برای همی به این روش توزیع آزاد هم می گویند

برای پیدا کردن فاصله اطمینان که در مورد یک میانگین وجود دارد روش پارامتری محاسبه می کند حاشیه خطا با یک فرمول و تخمینی از میانگینمون با میانگین نمونه

روش های پارامتری موثر ترو کارامد تر از روش های غیر پارامتری هستند

```
[53] x = data[['age','sex','ca','trestbps','chol','fbs','restecg','thalach','exang','oldpeak']]
     y = data['target']

     X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.2, random_state=42)
```

```
[55] from sklearn.svm import SVC
```

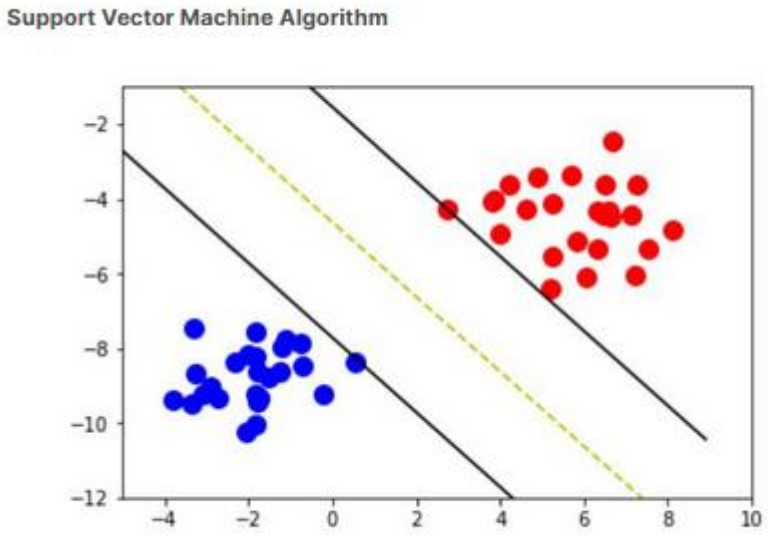
```
model_svc = 'Support Vector Classifier'
svc =  SVC(kernel='rbf', C=2)
svc.fit(X_train, y_train)
svc_predicted = svc.predict(X_test)
svc_conf_matrix = confusion_matrix(y_test, svc_predicted)
svc_acc_score = accuracy_score(y_test, svc_predicted)
print("confussion matrix")
print(svc_conf_matrix)
print("-----------------------------------------")
print("Accuracy of Support Vector Classifier:",svc_acc_score*100,'\n')
print("-----------------------------------------")
```

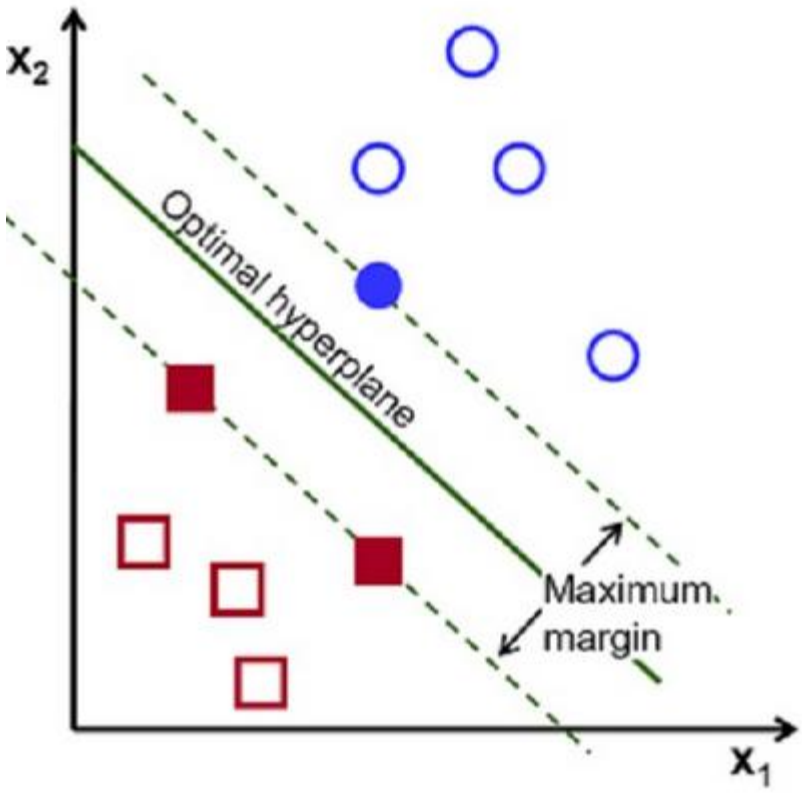
```
confussion matrix
[[19  4]
 [ 6 29]]
-----------------------------------------
Accuracy of Support Vector Classifier: 82.75862068965517

-----------------------------------------
```

```
[ ] from sklearn.svm import SVC
```

```
svm = SVC(random_state = 1)
svm.fit(x_train.T, y_train.T)

acc = svm.score(x_test.T,y_test.T)*100
accuracies['SVM'] = acc
print("Test Accuracy of SVM Algorithm: {:2f}%".format(acc))
```

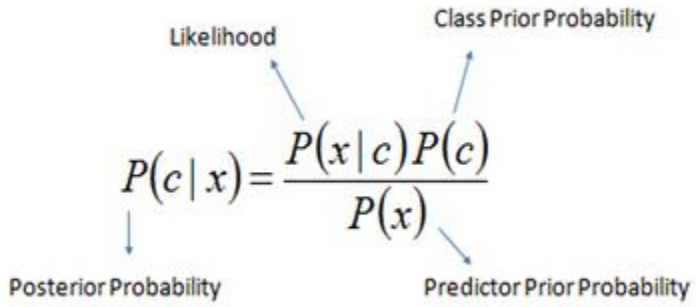
یکی دیگر از روش هایی که برای دسته بندی استفاده می شود هدفمون این هست که یک خط یا هایپر پلین پیدا کنیم که داده هامون جدا کندیعنی ویژگی هامون توسط خط یا صفحه جدا شوند گاهی اوقات جدا می شوند گاهی نه و باید برایشون راهی پیدا کنیم. شرایط نرم می کنیم فضا را بزرگ می کنیم تاتوانایی هاش بالا برود یا به جای ضرب داخلی کرنر های متفاوت می گذاریم وجدا می کنیم.خط باید بیشترین مارجین داشته باشد برای اینکه با اطمینان بگوییم که برای کلاس یک هست یا کلاس دو





ادامه قانون هم بیز :

Naive Bayes Algorithm



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

دراین جا عملا لایکلیهود باید حساب کنیم در همه کلاس ها یکسان هست برای همینevidence

در نظر نمی گیریم پس می شود احتمال کلاس درلایکلیهود حساب کنیم تاposterior

بدست آوریم وچون احتمال کلاس ها را هم برابر در نظر بگیریم می توانیم احتمال کلاس ها را هم صرف نظر کنیم که فقط باید لایکلیهود بدست آوریم

در حالت تک متغییره از این فرمول استفاده می کنیم

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

یعنی هر کدام از ویژگی های ما با یک بردار یکدونه ای بدست می آید که در این حالت میانگین و واریانس در فرمول قرار می دهیم.که منظور میانگین داده های کلاس هست و واریانس مربوط به داده های کلاس که در این مسأله دو کلاس ۰و۱ داریم یک بار برای کلاس یک و یک بار هم برای کلاس صفر میانگین و واریانس حساب میکنیم داده در این جا قرار می دهیم داده انگار با یک ویژگی کار می کنیم مثلا قند خون یا فشار خون که بین ۰تا۲۰۰ هست نمونه مون یا یک ویژگی نمایندگی میشه عدد قند خون یا فشار خون اینجا قرار می دهیم.

و احتمال آن نمونه را بدست می آوریم واحتمال این که مربوط به کلاس صفر باشه چقدر هست و احتمال این که مربوط به کلاس یک باشه چقدر هست. یعنی با این احتمال بیماری قلبی دارد یا ندارد.

حالا در این جا از یک متغییره خارج می شود به چند متغییره باید برویم که این جا باید از مفهوم کواریانس باید استفاده کنیم و در فرمول جایگذاری کنیم که اینجا چندتا میانگین و واریانس باید بدست آوریم برای هر کلاس

که در اینجا بردار چند تایی داریم

```
model_nb = 'Naive Bayes'
nb = GaussianNB()
nb.fit(X_train,y_train)
nbpred = nb.predict(X_test)
nb_conf_matrix = confusion_matrix(y_test, nbpred)
nb_acc_score = accuracy_score(y_test, nbpred)
print("confussion matrix")
print(nb_conf_matrix)
print("--------------------------------------------")
print("Accuracy of Naive Bayes model:",nb_acc_score*100,'\n')
print("--------------------------------------------")
model_nb = 'Naive Bayes'
nb = GaussianNB()
nb.fit(X_train,y_train)
nbpred = nb.predict(X_test)
nb_conf_matrix = confusion_matrix(y_test, nbpred)
nb_acc_score = accuracy_score(y_test, nbpred)
print("confussion matrix")
print(nb_conf_matrix)
print("--------------------------------------------")
print("Accuracy of Naive Bayes model:",nb_acc_score*100,'\n')
print("--------------------------------------------")
```

```
confussion matrix
[[19  4]
 [ 6 29]]
--------------------------------------------
Accuracy of Naive Bayes model: 82.75862068965517

--------------------------------------------
confussion matrix
[[19  4]
 [ 6 29]]
```

```
model_knn = 'K-NeighborsClassifier'
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predicted = knn.predict(X_test)
knn_conf_matrix = confusion_matrix(y_test, knn_predicted)
knn_acc_score = accuracy_score(y_test, knn_predicted)
print("confussion matrix")
print(knn_conf_matrix)
print("--------------------------------------------")
print("Accuracy of K-NeighborsClassifier:",knn_acc_score*100,'\n')
print("--------------------------------------------")
print(classification_report(y_test,knn_predicted))
```

در روش نزدیکترین همسایه به ازای هر کلاس الگو ها را میانگین می گرفتیم داده مرکزی بدست می آوردیم

فاصله داده با مرکز می سنجیم وبعد می گفتیم ان که نزدیکتر هست به آن تعلق دارداگر داده ای مورد سوال باشدفاصله اش را با همه کلاس ها می سنجیم که متعلق به کدام هست اگر داده نویز داشته باشد تصمیم گیری با خطا همراه هست این روش به داده های نویزی حساس هست اگر مرز تصمیم نباشد بر اساس شباهت و فاصله کلاس بندی می شوند

تعداد همسایه ها بستگی به داده دارد به طور کلی هر چه همسایه بیشتر باشد با خطا کمتری طبقه بندی می شوند

برای دسته بندی های دو کلاسه بهتر است که تعداد همسایه یک عدد فرد باشد

نتیجه:

استرس سابقه خانوادگی و سن تاثیر زیاد دارد در بیماری قلبی

چربی خون‌وقند چربی شکمی و الکل پی ولیوزیاد می خواهند تاثیر شون کم هست