



Shahid Beheshti
University

دانشگاه شهید بهشتی
دانشکده علوم ریاضی
گروه علوم کامپیوتر

گزارش تمرین های سری دوم

(دیتاست بیماران قلبی)

درس داده کاوی

اساتید محترم:

جناب آقای دکتر هادی فراهانی و جناب آقای دکتر سعید رضا خردپیشه

آموزشیار محترم: جناب آقای علی شریفی

جواد تدین ۹۹۴۲۲۰۴۱

بهار ۱۴۰۰

Jupyter Tadayon-99422041-heart Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

In [6]: data = pd.read_csv('D:/tadayon-heart/heart.csv')

In [7]: data

Out[7]:
```

	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

در این بخش ما با استفاده از دیتاست بیماری های قلبی UCL که در آدرس قرار دارد . به بررسی سوالات زیر میپردازیم .

۱. در ابتدا به بررسی دیتاست با استفاده از پکیج pandas بپردازید .

- آیا داده پرت در دیتاست وجود دارد ؟ در صورت وجود آنها را حذف کنید .
- بررسی کنید آیا تعداد نمونه ها در هر کلاس متوازن است ؟ (به صورت مختصر توضیح دهید اگر داده ها متوازن نباشد چه مشکلاتی ممکن است پیش بیاید و چه راه حل هایی برای آن وجود دارد .)

داده پرت (Outlier) به مشاهداتی گویند که نسبت به یک نقطه مرکزی (مثل میانگین) فاصله زیادی برحسب یک شاخص پراکندگی (مثال انحراف معیار) داشته باشد. این ایده از خصوصیات توزیع نرمال گرفته شده است. در صورتی که داده ها دارای توزیع نرمال یا طبیعی باشند، احتمال اینکه مقداری خارج از فاصله سه برابر انحراف معیار از میانگین قرار گیرد، بسیار کوچک خواهد بود. در نتیجه اگر به چنین مشاهده ای برخوردیم، آن را داده پرت و در نتیجه مشاهده ناهنجار یا نامتعارف تلقی خواهیم کرد. در شکل نقاطی را نشان می دهد که فاصله بسیار زیادی از مرکز توزیع دارند که در واقع همان نقاط پرت موجود در داده ها می باشند . برای حذف داده های پرت، هر نمونه ای که فاصله ای بیش از ۵,۲ از مبدا صفر داشت را از داده ها حذف کردیم. در شکل مشاهده می کنید که پس از حذف این نقاط، تعداد نقاط پرت موجود در مجموعه داده ها کاهش یافته است.

302 57 0 1 130 236 0 0 174 0 0.0 1 1 2 0

303 rows x 14 columns

```
In [8]: data = data.sample(frac=1).reset_index(drop=True)
```

```
In [9]: data
```

```
Out[9]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
1	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
2	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
3	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
4	29	1	1	130	204	0	0	202	0	0.0	2	0	2	1
...
298	54	1	0	110	206	0	0	108	1	0.0	1	1	2	0
299	64	1	2	140	335	0	1	158	0	0.0	2	0	2	0
300	48	1	1	130	245	0	0	180	0	0.2	1	0	2	1
301	42	0	2	120	209	0	1	173	0	0.0	1	0	2	1
302	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1

300 48 1 1 130 245 0 0 180 0 0.2 1 0 2 1

301 42 0 2 120 209 0 1 173 0 0.0 1 0 2 1

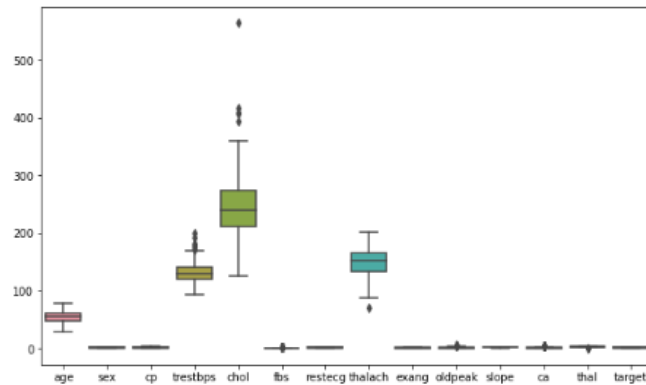
302 48 0 2 130 275 0 1 139 0 0.2 2 0 2 1

303 rows x 14 columns

```
In [10]: plt.figure(figsize=(10, 6))
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(data=data)
```

```
Out[10]: <AxesSubplot:>
```



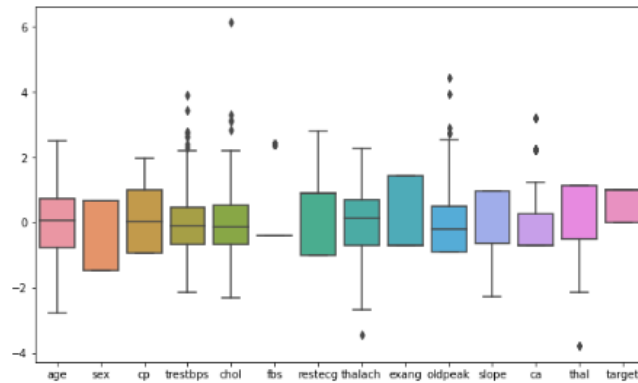
```
In [13]: # for i in data:
#         scaled_data[i] = (data[i]-data[i].mean())/data[i].std()
# data = pd.read_csv('Desktop/heart.csv')
cols_to_norm = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
data[cols_to_norm] = data[cols_to_norm].apply(lambda x: (x - x.mean()) / x.std())
```

```
In [14]: data['chol'].std()
```

Out[14]: 1.0

```
In [15]: plt.figure(figsize=(10, 6))
sns.boxplot(data=data)
```

Out[15]: <AxesSubplot:>



```
if(s == True):
    out_idx.append(j)
```

In [19]: len(out_idx)

Out[19]: 35

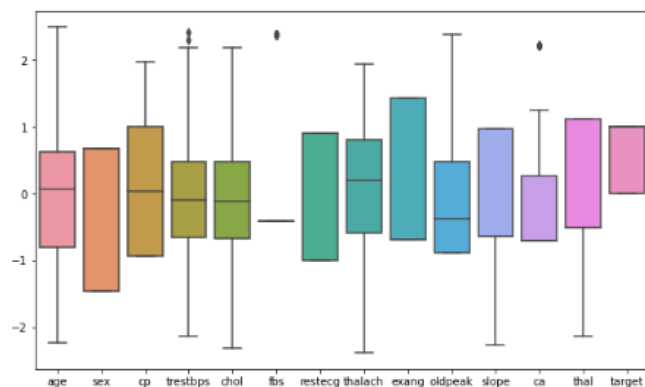
In [20]: y.drop(out_idx, inplace=True)

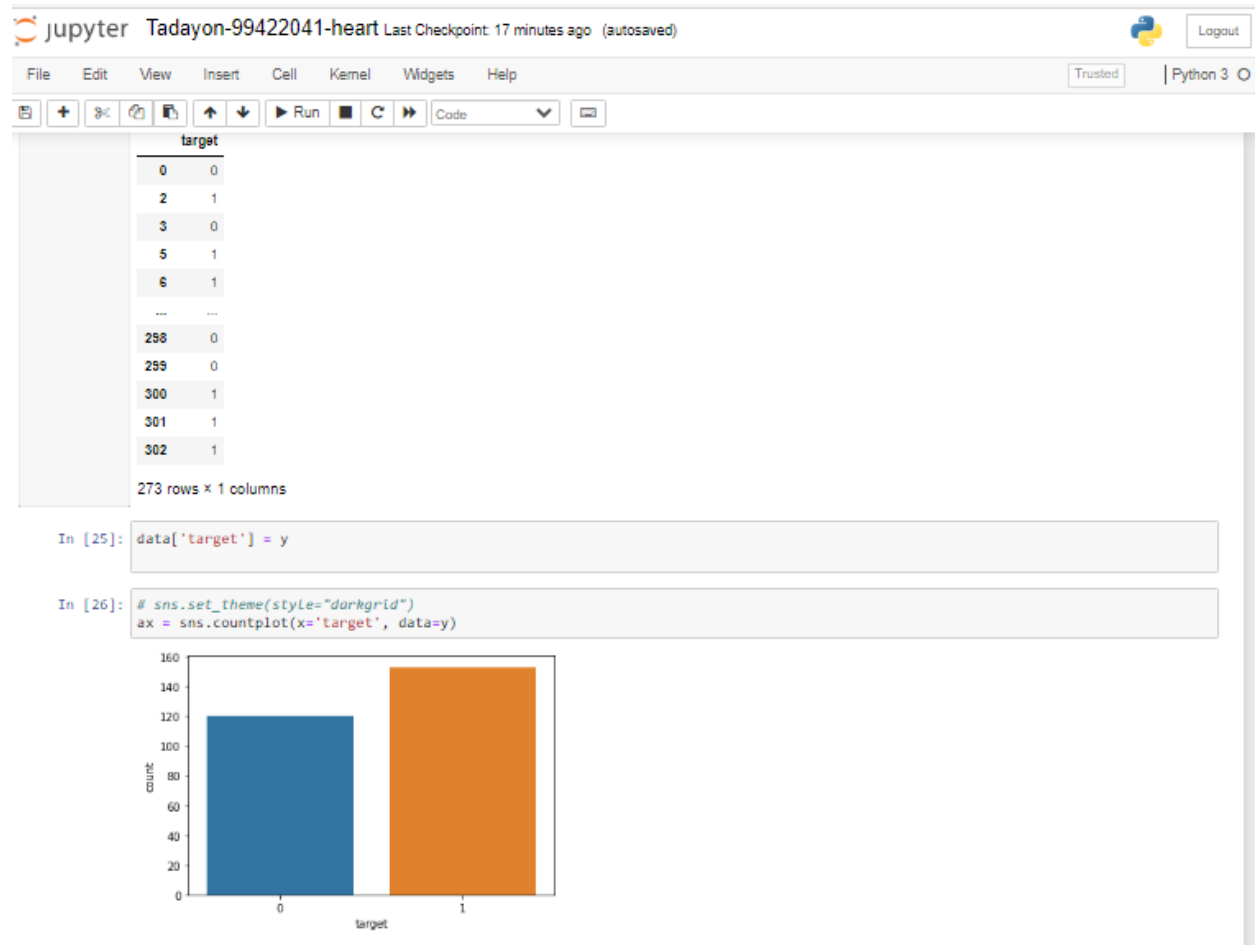
In [21]: data.drop(out_idx, inplace=True)

In [22]: # y=y.reset_index()

In [23]: plt.figure(figsize=(10, 6))
sns.boxplot(data=data)

Out[23]: <AxesSubplot:>





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** بهره ببرید .

```
In [27]: from sklearn.model_selection import train_test_split
```

```
In [28]: # data = pd.read_csv('Desktop/heart.csv')
# data = data.apply(lambda x: (x - x.mean()) / x.std())
p = data[data['target'] == 1]
n = data[data['target'] == 0]
```

```
In [29]: # data = pd.read_csv('Desktop/heart.csv')
# cols_to_norm = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
# data[cols_to_norm] = data[cols_to_norm].apply(lambda x: (x - x.mean()) / x.std())
p_y = p['target']
n_y = n['target']
del p['target']
del n['target']
y = pd.DataFrame(data['target'])
del data['target']
# x = data
```

```
In [30]: X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.2, random_state=42)
```

```
In [31]: y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)
X_train = X_train.reset_index(drop=True)
X_test = X_test.reset_index(drop=True)
```

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

قضیه بیز از روشی برای دسته بندی پدیده ها بر پایه احتمال استفاده میکند. و احتمال رخداد پیشامد A به شرط B برابر است با احتمال رخداد پیشامد B به شرط A ضرب در احتمال رخداد پیشامد A تقسیم بر احتمال رخداد پیشامد B

$$P(A|B) = \frac{P(A|B)P(A)}{P(B)}$$

دسته بندی به این صورت انجام می شود که : احتمال این که یک نمونه در هر کدام از دسته ها باشد را بدست می آوریم و نمونه را در دسته ای قرار میدهیم که مقدار احتمال آن بیشتر است. درقضیه بیز فرمول بالا به صورت زیر تعریف میشود:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

که در آن C دسته مورد نظر و X مقادیر نمونه است.

- $P(c | x)$ احتمال پیش بینی کننده (ویژگی) است.
- $P(c)$ احتمال قبلی کلاس است.
- $P(x | c)$ این احتمال است که احتمال کلاس پیش بینی کننده داده شده است.
- $P(x)$ احتمال قبلی پیش بینی کننده است.

در مثال زیر بهتر میتوانیم هر کدام از این احتمالات را با مثال بدست آوریم:

$p(x|c)$ احتمال این که نمونه

$p(c)$ از تقسیم تعداد برجسب های آن دسته به نسبت کل داده ها بدست می آید

$p(x)$ احتمال قبلی که احتمال وقوع مقدار مورد نظر برای نمونه را نشان می دهد

$$P(x | c) = P(\text{Sunny} | \text{Yes}) = 3 / 9 = 0.33$$

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3



Likelihood Table		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(x) = P(\text{Sunny}) = 5 / 14 = 0.36$$

$$P(c) = P(\text{Yes}) = 9 / 14 = 0.64$$

قضیه بیز بسته به داده هایی که داریم میتواند به صورت های مختلفی به کار برده شود. مثلا برای داده های پیوسته و دارای توزیع نرمال از الگوریتم دسته بند بیز گاوسی استفاده می شود. در ادامه سه دسته بند را معرفی و با هم مقایسه می نمایم.

• gaussian naive bayse

برای داده های پیوسته و با توزیع نرمال مناسب است و احتمال آن با فرمول زیر محاسبه می شود

در این حالت هر دسته یا گروه دارای توزیع گاوسی است. به این ترتیب اگر کلاس داشته باشیم می توانیم برای هر دسته میانگین و واریانس را محاسبه کرده و پارامترهای توزیع نرمال را برای آن ها برآورد

کنیم. فرض کنید که μ_k میانگین و σ_k^2 واریانس دسته k ام یعنی C_k باشد. همچنین v را مشاهدات حاصل از متغیرهای تصادفی X در نظر بگیرید. از آنجایی که توزیع X در هر دسته گاوسی (نرمال) فرض شده است، خواهیم داشت:

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

• Multinomial naive Bayes

برای زمانی که **feature vectors** ها دارای خاصیت احتمال چندجمله ای هستند مناسب است.

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 $X=(x_1, \dots, x_n)$ 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 ترتیب تابع درستنمایی در چنین مدلی به شکل زیر نوشته می‌شود.

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

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

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

• Bernoulli naive Bayes

در این دسته بند ویژگی ها مستقل هستند.

این نوع از دسته‌بند بیز بیشترین کاربرد را در 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 کلاس های مختلف C_k به شکل زیر نوشته می‌شود.

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

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

۴. با در نظر گرفتن فیچر ها $chol$ ، $trestbps$ ، $thalach$ و لیبل $target$ یک دسته بند **Gaussian Naive Bayes** ^۲ را از پایه پیاده سازی کنید. (بدون استفاده از پکیج) برای این کار شما نیاز است که در دیتاست آموزشی خود اعضای مختلف قاعده بیز را محاسبه کنید.

```
X_test = X_test.reset_index(drop=True)
```

```
In [33]: p = X_train[y_train['target'] == 1]
p_y = y_train[y_train['target'] == 1]
n = X_train[y_train['target'] == 0]
n_y = y_train[y_train['target'] == 0]
```

```
In [34]: y_train['target']
```

```
Out[34]: 0      1
1      1
2      1
3      1
4      1
..
213    1
214    0
215    1
216    1
217    1
Name: target, Length: 218, dtype: int64
```

```
In [35]: means_p = pd.DataFrame(columns=['chol', 'thalach', 'trestbps'])
```

```
In [36]: means_n = pd.DataFrame(columns=['chol', 'thalach', 'trestbps'])
```

```
In [37]: stds_p = pd.DataFrame(columns=['chol', 'thalach', 'trestbps'])
stds_n = pd.DataFrame(columns=['chol', 'thalach', 'trestbps'])
```

```
In [38]: means_p = means_p.append({'chol' : p['chol'].mean(),
'thalach' : p['thalach'].mean(),
'trestbps' : p['trestbps'].mean()} ,
ignore_index=True)
```

```
In [39]: means_n = means_n.append({'chol' : n['chol'].mean(),
'thalach' : n['thalach'].mean(),
'trestbps' : n['trestbps'].mean()} ,
ignore_index=True)
```

```
In [40]: stds_p = stds_p.append({'chol' : p['chol'].std(),
'thalach' : p['thalach'].std(),
'trestbps' : p['trestbps'].std()} ,
ignore_index=True)
```

```

In [41]: stds_n = stds_n.append({'chol' : n['chol'].std(),
                                'thalach' : n['thalach'].std(),
                                'trestbps' : n['trestbps'].std()} ,
                                ignore_index=True)

In [42]: from math import sqrt, pi, exp

In [43]: mean_chol_ = X_train['chol'].mean()
mean_thalach = X_train['thalach'].mean()
mean_trestbps = X_train['trestbps'].mean()

std_chol = X_train['chol'].std()
std_thalach = X_train['thalach'].std()
std_trestbps = X_train['trestbps'].std()

In [44]: # pred = (1/(sqrt(2*pi)*stds['chol'][0]))*exp(-(x-means['chol'][0])**2/(2*stds['chol'][0]**2))

In [45]: # pdf_chol_p = (1/(sqrt(2*pi)*stds_p['chol']))*exp(-(x['chol'][i]-means_p['chol'])**2/(2*stds_p['chol']**2))
# pdf_thalach_p = (1/(sqrt(2*pi)*stds_p['thalach']))*exp(-(x['thalach'][i]-means_p['thalach'])**2/(2*stds_p['thalach']**2))
# pdf_trestbps_p = (1/(sqrt(2*pi)*stds_p['trestbps']))*exp(-(x['trestbps'][i]-means_p['trestbps'])**2/(2*stds_p['trestbps']**2))

In [46]: pdfs_chol_p = []
pdfs_thalach_p = []
pdfs_trestbps_p = []

In [47]: for i in range(len(X_test)):
    tmp = (1/(sqrt(2*pi)*stds_p['chol']))*exp(-(X_test['chol'][i]-means_p['chol'])**2/(2*stds_p['chol']**2))
    pdfs_chol_p.append(tmp)

    tmp = (1/(sqrt(2*pi)*stds_p['thalach']))*exp(-(X_test['thalach'][i]-means_p['thalach'])**2/(2*stds_p['thalach']**2))
    pdfs_thalach_p.append(tmp)

    tmp = (1/(sqrt(2*pi)*stds_p['trestbps']))*exp(-(X_test['trestbps'][i]-means_p['trestbps'])**2/(2*stds_p['trestbps']**2))
    pdfs_trestbps_p.append(tmp)

In [48]: pdfs_chol_n = []
pdfs_thalach_n = []
pdfs_trestbps_n = []

In [49]: for i in range(len(X_test)):
    tmp = (1/(sqrt(2*pi)*stds_n['chol']))*exp(-(X_test['chol'][i]-means_n['chol'])**2/(2*stds_n['chol']**2))
    pdfs_chol_n.append(tmp)

    tmp = (1/(sqrt(2*pi)*stds_n['thalach']))*exp(-(X_test['thalach'][i]-means_n['thalach'])**2/(2*stds_n['thalach']**2))
    pdfs_thalach_n.append(tmp)

    tmp = (1/(sqrt(2*pi)*stds_n['trestbps']))*exp(-(X_test['trestbps'][i]-means_n['trestbps'])**2/(2*stds_n['trestbps']**2))
    pdfs_trestbps_n.append(tmp)

In [50]: prior = len(p)/(len(p) + len(n))

In [51]: positive = []

for i in range(len(pdfs_chol_p)):
    tmp = pdfs_chol_p[i]*pdfs_thalach_p[i]*pdfs_trestbps_p[i]*prior
    positive.append(tmp)

negative = []

for i in range(len(pdfs_chol_p)):
    tmp = pdfs_chol_n[i]*pdfs_thalach_n[i]*pdfs_trestbps_n[i]*(1-prior)
    negative.append(tmp)

In [52]: pred = []
for i in range(len(positive)):
    if(positive[i][0]>negative[i][0]):
        pred.append(1)
    else:
        pred.append(0)

In [53]: acc = 0
for i in range(len(y_test)):
    if(y_test['target'][i] == pred[i]):
        acc += 1
acc = acc/len(pred)


In [54]: acc


Out[54]: 0.76363636363637

```

۵. پس پیاده سازی Gaussian Naive Bayes و آموزش آن بر روی داده های آموزشی (۸۰ 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 دارید معیار های زیر گزارش دهید .

- F1 score
- Recall
- Precision

```
jupyter Tadayon-99422041-heart Last Checkpoint: 44 minutes ago (autosaved)  Logout
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 
```

```
In [55]: tp = 0
fp = 0
fn = 0
for i in range(len(pred)):
    if(pred[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1_score = 2*precision*recall/(precision+recall)

In [56]: print('precision=',precision)
print('recall=',recall)
print('f1-score=',f1_score)

precision= 0.7105263157894737
recall= 0.9310344827586207
f1-score= 0.8059701492537312
```

۶. با استفاده از پکیج sklearn و GaussianNB یک مدل بسازید و بر روی داده های آموزشی، ترین کنید سپس بر روی داده های تست همانند سوال (۵) سه معیار را گزارش دهید.

```
jupyter Tadayon-99422041-heart Last Checkpoint: an hour ago (autosaved) Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted

f1-score= 0.8059701492537312

In [57]: from sklearn.naive_bayes import GaussianNB

In [58]: X = X_train[['chol', 'thalach', 'trestbps']]

In [59]: X

Out[59]:
      chol  thalach  trestbps
0 -0.236617  0.452000  0.363564
1 -0.390965 -1.687256  0.591638
2  0.342190 -0.290191 -0.092585
3 -1.046947 -0.071899  0.477601
4  0.940291  0.321025 -0.662770
...
213 -1.374937 -1.163356 -1.232956
214 -0.448846  1.368824  1.161824
215 -0.603195  0.888583 -0.662770
216 -0.024388  1.325166 -0.092585
217 -1.027653  0.539317 -0.662770

218 rows x 3 columns

In [60]: clf = GaussianNB()
clf.fit(np.reshape(np.array(X), (len(X), -1)), np.reshape(np.array(y_train), (len(X), )))

Out[60]: GaussianNB()

In [61]: pred_1 = clf.predict(np.reshape(np.array(X_test[['chol', 'thalach', 'trestbps']]), (len(X_test), -1)))

In [62]: tp = 0
fp = 0
fn = 0
for i in range(len(pred_1)):
    if(pred_1[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred_1[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred_1[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision_1 = tp/(tp+fp)
recall_1 = tp/(tp+fn)
f1_score_1 = 2*precision_1*recall_1/(precision_1+recall_1)

In [63]: print('precision=',precision_1)
print('recall=',recall_1)
print('f1-score=',f1_score_1)

precision= 0.7105263157894737
recall= 0.9318344827586207
f1-score= 0.8059701492537312
```

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.

```
jupyter Tadayon-99422041-heart Last Checkpoint: a minute ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [63]: print('precision=',precision_1)
print('recall=',recall_1)
print('f1-score=',f1_score_1)

precision= 0.7105263157894737
recall= 0.9310344827586207
f1-score= 0.8059701492537312
```

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. سپس بر روی داده های تست سه معیار Recall ، Precision ، F1 score ، را گزارش کنید.

```
jupyter Tadayon-99422041-heart Last Checkpoint: 4 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [65]: from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

In [66]: svm = make_pipeline(StandardScaler(), SVC(gamma='auto'))

In [67]: svm.fit(np.reshape(np.array(X), (X.shape[0], -1)), np.reshape(np.array(y_train), (X.shape[0], )))

Out[67]: Pipeline(steps=[('standardscaler', StandardScaler()),
('svc', SVC(gamma='auto'))])

In [68]: pred_svm = svm.predict(np.reshape(np.array(X_test[['chol', 'thalach', 'trestbps'])), (X_test.shape[0],-1)))

In [69]: pred_svm

Out[69]: array([1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1], dtype=int64)

In [70]: tp = 0
fp = 0
fn = 0
for i in range(len(pred_1)):
    if(pred_svm[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred_svm[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred_svm[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision_svm = tp/(tp+fp)
recall_svm = tp/(tp+fn)
f1_score_svm = 2*precision_1*recall_1/(precision_1+recall_1)

In [71]: print('precision=',precision_svm)
print('recall=',recall_svm)
print('f1-score=',f1_score_svm)

precision= 0.717948717948718
recall= 0.9655172413793104
f1-score= 0.8059701492537312
```

۹. حداقل دو حالت مختلف را برای کرنل در SVM ساخته شده با پکیج در نظر بگیرید و نتایج آن را گزارش دهید. آیا کرنل های مختلف نتایج مختلفی ارایه دادند؟ به صورت کلی علت استفاده از کرنل ها در SVM چیست؟ توضیح دهید.

```
In [72]: from sklearn import svm
```

```
In [73]: # svm = make_pipeline(StandardScaler(), SVC(gamma='auto'))
```

```
In [74]: lin_svm = svm.SVC(kernel='linear', gamma=2)
```

```
In [75]: lin_svm.fit(np.reshape(np.array(X), (X.shape[0], -1)), np.reshape(np.array(y_train), (X.shape[0], )))
```

```
Out[75]: SVC(gamma=2, kernel='linear')
```

```
In [76]: pred_lin_svm = lin_svm.predict(np.reshape(np.array(X_test[['chol', 'thalach', 'trestbps']]), (X_test.shape[0], -1)))
```

```
In [77]: pred_lin_svm
```

```
Out[77]: array([1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
        1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1], dtype=int64)
```

```
In [78]: tp = 0
fp = 0
fn = 0
for i in range(len(pred_lin_svm)):
    if(pred_lin_svm[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred_lin_svm[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred_lin_svm[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision_lin_svm = tp/(tp+fp)
recall_lin_svm = tp/(tp+fn)
f1_score_lin_svm = 2*precision_1*recall_1/(precision_1+recall_1)
```

```
In [79]: print('precision=', precision_lin_svm)
print('recall=', recall_lin_svm)
print('f1-score=', f1_score_lin_svm)
```

```
precision= 0.7105263157894737
recall= 0.9310344827586207
f1-score= 0.8059701492537312
```

```
In [80]: rbf_svm = svm.SVC(kernel='rbf', gamma=2)
```

```
In [81]: rbf_svm.fit(np.reshape(np.array(X), (X.shape[0], -1)), np.reshape(np.array(y_train), (X.shape[0], )))
```

```
Out[81]: SVC(gamma=2)
```

```
In [82]: pred_rbf_svm = rbf_svm.predict(np.reshape(np.array(X_test[['chol', 'thalach', 'trestbps']]), (X_test.shape[0], -1)))
```

```
In [83]: pred_rbf_svm
```

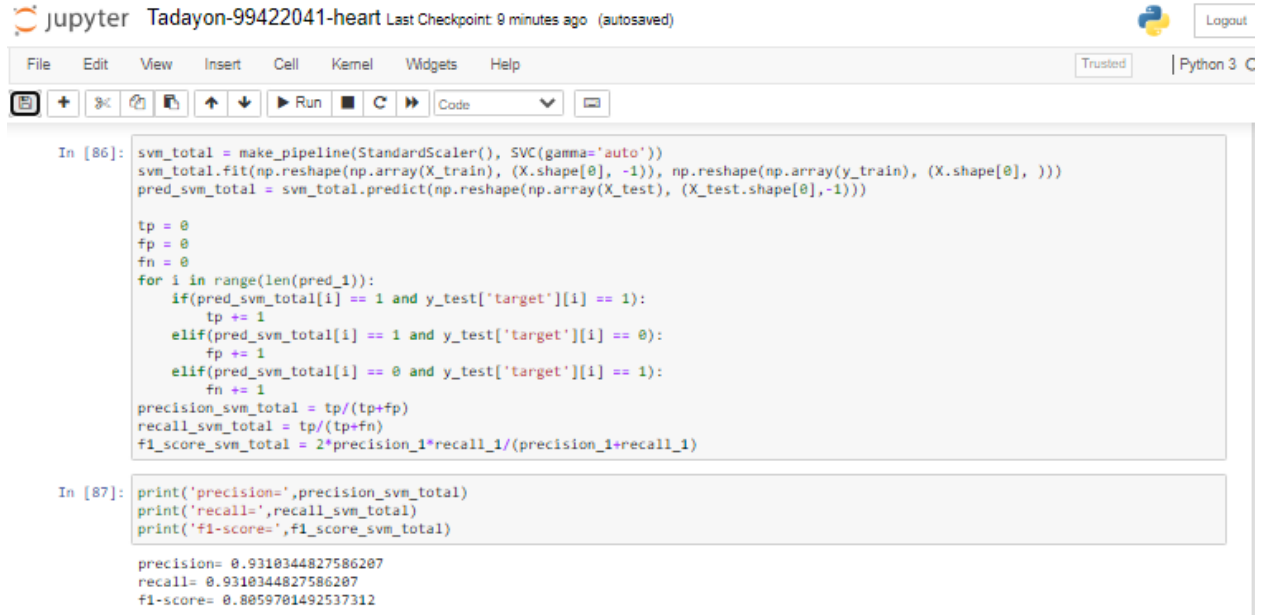
```
Out[83]: array([1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
        0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
        1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1], dtype=int64)
```

```
In [84]: tp = 0
fp = 0
fn = 0
for i in range(len(pred_rbf_svm)):
    if(pred_rbf_svm[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred_rbf_svm[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred_rbf_svm[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision_rbf_svm = tp/(tp+fp)
recall_rbf_svm = tp/(tp+fn)
f1_score_rbf_svm = 2*precision_1*recall_1/(precision_1+recall_1)
```

```
In [85]: print('precision=', precision_rbf_svm)
print('recall=', recall_rbf_svm)
print('f1-score=', f1_score_rbf_svm)
```

```
precision= 0.6842105263157895
recall= 0.896551724137931
f1-score= 0.8059701492537312
```


۱۰. 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 .



The image shows a Jupyter Notebook interface with the following components:

- Header:** Jupyter logo, username 'Tadayon-99422041-heart', last checkpoint '9 minutes ago (autosaved)', and a 'Logout' button.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes buttons for file operations, a 'Run' button, and a dropdown menu currently set to 'Code'.
- Code Cells:**
 - In [86]:** Contains code to create an SVM pipeline, fit it on training data, predict on test data, and calculate precision, recall, and F1 score. The code uses `StandardScaler` and `SVC` from `sklearn`.

```
svm_total = make_pipeline(StandardScaler(), SVC(gamma='auto'))
svm_total.fit(np.reshape(np.array(X_train), (X.shape[0], -1)), np.reshape(np.array(y_train), (X.shape[0], )))
pred_svm_total = svm_total.predict(np.reshape(np.array(X_test), (X_test.shape[0], -1)))

tp = 0
fp = 0
fn = 0
for i in range(len(pred_1)):
    if(pred_svm_total[i] == 1 and y_test['target'][i] == 1):
        tp += 1
    elif(pred_svm_total[i] == 1 and y_test['target'][i] == 0):
        fp += 1
    elif(pred_svm_total[i] == 0 and y_test['target'][i] == 1):
        fn += 1
precision_svm_total = tp/(tp+fp)
recall_svm_total = tp/(tp+fn)
f1_score_svm_total = 2*precision_1*recall_1/(precision_1+recall_1)
```
 - In [87]:** Contains code to print the calculated metrics.

```
print('precision=', precision_svm_total)
print('recall=', recall_svm_total)
print('f1-score=', f1_score_svm_total)
```
- Output:** The output of the second cell shows the calculated values:

```
precision= 0.9310344827586207
recall= 0.9310344827586207
f1-score= 0.8059701492537312
```

۱۱. برای سوال (۱۰) ، یکبار مدل را با 5-fold Cross Validation اجرا کنید و نتایج را گزارش دهید . (در این جا برای فولد کردن داده ها از کل دیتاست استفاده میکنیم .)

```
In [89]: data = pd.read_csv('D:/tadayon-heart/heart.csv')
data = data.sample(frac=1).reset_index(drop=True)
cols_to_norm = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
data[cols_to_norm] = data[cols_to_norm].apply(lambda x: (x - x.mean()) / x.std())
```

```
In [90]: y = data['target']
del data['target']
```

```
In [91]: from sklearn.model_selection import KFold, cross_validate
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score
```

```
In [92]: clf = svm.SVC(random_state=42, gamma='auto')
scoring = { 'precision' : make_scorer(precision_score),
            'recall' : make_scorer(recall_score),
            'f1_score' : make_scorer(f1_score)}
```

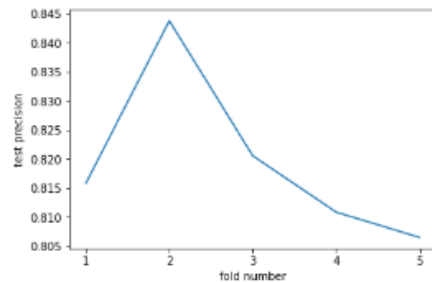
```
In [93]: scores = cross_validate(clf, data, y, cv=5, scoring=scoring)
```

```
In [94]: scores['test_precision']
```

```
Out[94]: array([0.81578947, 0.84375      , 0.82051282, 0.81081081, 0.80645161])
```

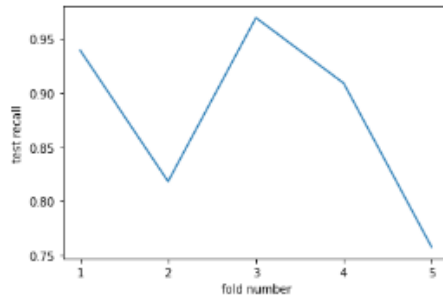
```
In [95]: plt.plot(['1','2','3','4','5'], scores['test_precision'])
plt.ylabel('test precision')
plt.xlabel('fold number')
```

```
Out[95]: Text(0.5, 0, 'fold number')
```



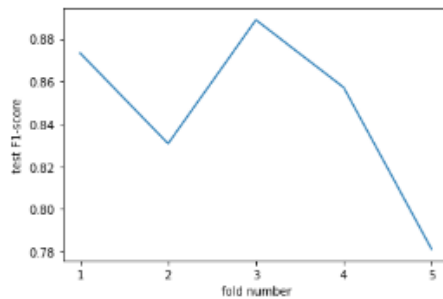
```
In [96]: plt.plot(['1','2','3','4','5'], scores['test_recall'])
plt.ylabel('test recall')
plt.xlabel('fold number')
```

Out[96]: Text(0.5, 0, 'fold number')



```
In [97]: plt.plot(['1','2','3','4','5'], scores['test_f1_score'])
plt.ylabel('test F1-score')
plt.xlabel('fold number')
```

Out[97]: Text(0.5, 0, 'fold number')



```
In [98]: avg_precision = sum(scores['test_precision'])/len(scores['test_precision'])
avg_recall = sum(scores['test_recall'])/len(scores['test_recall'])
avg_f1_score = sum(scores['test_f1_score'])/len(scores['test_f1_score'])
```

```
In [99]: print('precision=',avg_precision)
print('recall=',avg_recall)
print('f1-score=',avg_f1_score)
```

```
precision= 0.8194629435822135
recall= 0.8787878787878789
f1-score= 0.8462580826841389
```

۱۲. با استفاده از پکیج **sklearn** دسته بند را **K-NN** ^۳ را بسازید . با به کارگیری تمامی فیچرها موجود در دیتاست آموزشی ، مدل را ترین کنید سپس بر روی دیتاست تست ، ارزیابی کنید .

.....

```
In [100]: from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(data, y)
```

```
Out[100]: KNeighborsClassifier(n_neighbors=3)
```

```
In [101]: scoring = { 'precision' : make_scorer(precision_score),
                    'recall' : make_scorer(recall_score),
                    'f1_score' : make_scorer(f1_score)}
```

```
In [102]: scores = cross_validate(neigh, data, y, cv=5, scoring=scoring)
```

```
In [103]: avg_precision = sum(scores['test_precision'])/len(scores['test_precision'])
avg_recall = sum(scores['test_recall'])/len(scores['test_recall'])
avg_f1_score = sum(scores['test_f1_score'])/len(scores['test_f1_score'])
```

```
In [104]: print('precision=', avg_precision)
print('recall=', avg_recall)
print('f1-score=', avg_f1_score)
```

```
precision= 0.8144588499596989
recall= 0.8545454545454547
f1-score= 0.8329898781591354
```

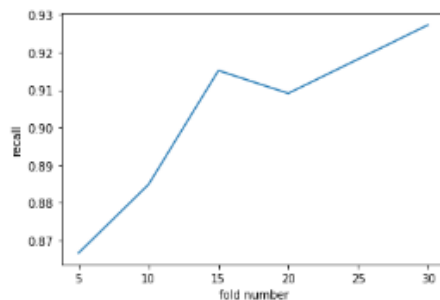
```
In [105]: nneighs = [5, 10, 15, 20, 30]
recalls = []
precisions = []
f1_scores = []
accs = []
```

```
In [106]: from sklearn.neighbors import KNeighborsClassifier
for i in nneighs:
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(data, y)

    scoring = { 'precision' : make_scorer(precision_score),
                'recall' : make_scorer(recall_score),
                'f1_score' : make_scorer(f1_score),
                'accuracy' : make_scorer(accuracy_score)}
    scores = cross_validate(neigh, data, y, cv=5, scoring=scoring)
    precisions.append(sum(scores['test_precision'])/len(scores['test_precision']))
    recalls.append(sum(scores['test_recall'])/len(scores['test_recall']))
    f1_scores.append(sum(scores['test_f1_score'])/len(scores['test_f1_score']))
    accs.append(sum(scores['test_accuracy'])/len(scores['test_accuracy']))
```

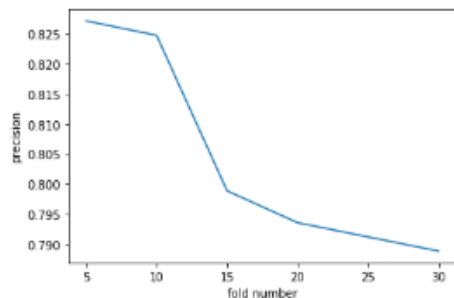
```
In [107]: plt.plot(nneighs, recalls)
plt.ylabel('recall')
plt.xlabel('fold number')
```

```
Out[107]: Text(0.5, 0, 'fold number')
```



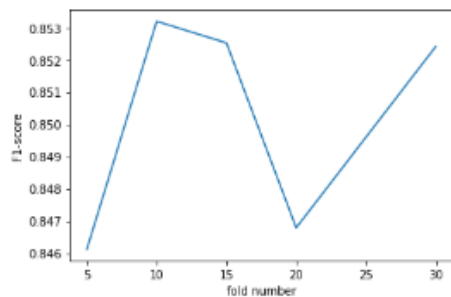
```
In [108]: plt.plot(nneighs, precisions)
plt.ylabel('precision')
plt.xlabel('fold number')
```

Out[108]: Text(0.5, 0, 'fold number')

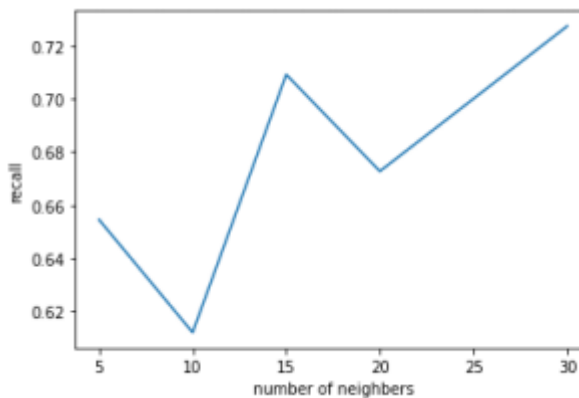
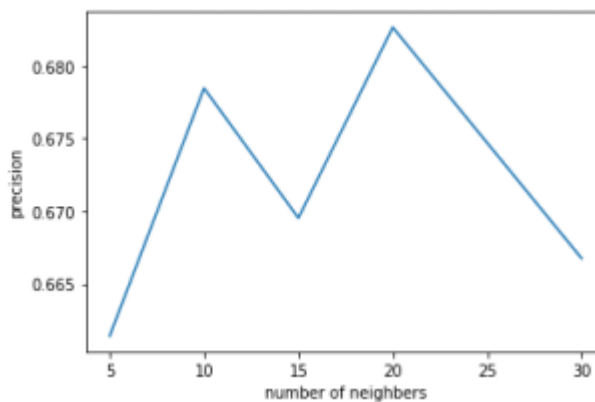


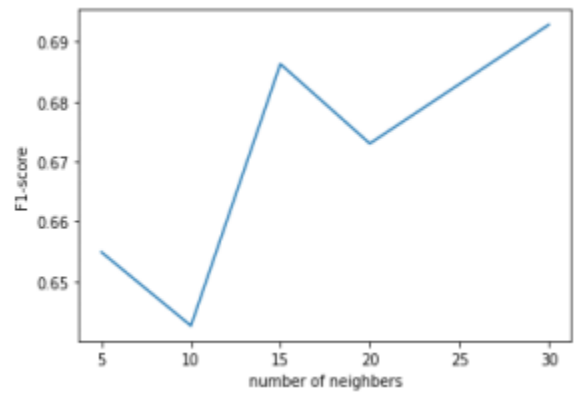
```
In [109]: plt.plot(nneighs, f1_scores)
plt.ylabel('f1-score')
plt.xlabel('fold number')
```

Out[109]: Text(0.5, 0, 'fold number')



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؟





همانطور که در شکل نشان داده شده است، مقدار precision با افزایش تعداد همسایه ها تغییر معناداری نمی کند، ولی مقادیر recall و f1-score با افزایش تعداد همسایه ها افزایش می یابد.

۱۴. در سوال (۱۲) به جای استفاده از تمامی فیچرها فقط از سه فیچر chol ، trestbps ، thalach استفاده کنید و مدل را بسازید . سپس نتایج ارزیابی را گزارش دهید .

```
1 data2 = data[['chol','trestbps','thalach']]
2 X_train, X_test, y_train, y_test = train_test_split(data2,
3 data['target'], test_size=0.2, shuffle=True, random_state=42)
4
5 clf = KNeighborsClassifier(n_neighbors=3)
6 clf.fit(X_train, y_train)
7 print(classification_report(y_test.values, clf.predict(X_test.values)))
```

	precision	recall	f1-score	support
0	0.54	0.83	0.66	23
1	0.83	0.54	0.66	35
accuracy			0.66	58
macro avg	0.68	0.68	0.66	58
weighted avg	0.71	0.66	0.66	58

۱۵. تفاوت بین روش های کلاس بندی پارامتری و غیر پارامتری را به صورت خلاصه بیان کنید . هر کدام بهتر است در چه مواقعی استفاده شوند ؟

در یک مدل پارامتری ، تعداد پارامترها با توجه به اندازه نمونه ثابت می شود. در یک مدل غیر پارامتری ، تعداد (موثر) پارامترها می توانند با اندازه نمونه رشد کنند. در یک رگرسیون OLS ، تعداد پارامترها همیشه به طول β خواهد بود، به علاوه یک واریانس. یک شبکه عصبی با معماری ثابت و بدون پوسیدگی وزن یک مدل پارامتریک است. اما اگر دچار فروپاشی وزن هستید ، مقدار پارامتر پوسیدگی که با اعتبار سنجی متقابل انتخاب می شود ، با داده های بیشتر ، به طور کلی کوچکتر می شود. این می تواند به عنوان افزایش تعداد موثر پارامترها با افزایش اندازه نمونه تفسیر شود.

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

Matthews Correlation Coefficient : پارامتری است که برای ارزیابی کارایی الگوریتم‌های یادگیری ماشین از آن استفاده می‌شود. این پارامتر بیان‌گر کیفیت کلاس‌بندی برای یک مجموعه باینری می‌باشد. بنابراین مواقعی از این معیار استفاده می‌گردد که classification ما همیشه دو بخشی باشد.