گزارش تمرین شماره ۲

درس داده کاوی

جناب آقای دکتر فراهانی

جناب آقای دکتر خرد پیشه

نسرین هداوند سوری

۱۴۰۰/۰۱/۱۷

بررسی مجموعه داده بیماری های قلبی:

بررسی سوال ۱ »

ابتدا به بررسی دیتا ست می پردازم و دیتاست خواندم و در بعضی از دیتا ست ها یکسری سلول هستند که مقدار ندارند و اون سطری که این سلول ها داخل آنها هست باید آن ها را drop کنم در pandas می شود اون سطرهایی که بعضی سلول ها شامل مقدار nan هستند آنها را drop کرد . اگر مقدار نداشته باشد نمی شود drop کرد ابتدا با nan ان را replace می کنم و سپس dropna می کنم و سپس سطر های تکراری حذف کردم و reset index ترتیب قبلی را ایجاد می کنم و یک بررسی روی مجموعه داده می کنم .

```
from scipy import stats

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('E:\Arshad\DataMining\Tamrin2\heart.csv')
data = data.replace("",np.nan)
data = data.dropna()
data = data.drop_duplicates().reset_index(drop=True)
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.00
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2.31
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0.61
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2.00
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2.00
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.00
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.00

ستون های دیتاست تبدیل می کنیم به داده با توزیع گوسی یا توزیع نرمال با میانگین صفر و انحراف استاندارد ۱ و تمام اونهایی که از این مقدار خیلی پرت هستن و انتظار داریم با این توزیع بین ۱ و ۱- تغییر کند و وقتی داده هایی تبدیل به این توزیع کنیم و اون داده هایی که بیش تر از ۳ و کم تر از ۳- هستند پیدا کردم و آنها را drop کردم .با استفاده از zscore دیتاست گرفتم و یک numpy array بر می گرداند و به این ترتیب تبدیل شده دیتا ست به این توزیع گوسی انجام می شود و سطرها را با پیدا کردن آن drop کردم و ازتعداد ۳۰۲ تا ۲۸۷ تا باقی موند که ۱۵ سطر داشتیم که داخل آن یکی از سلول های آن داده پرت برای آن ویژگی محسوب می شود

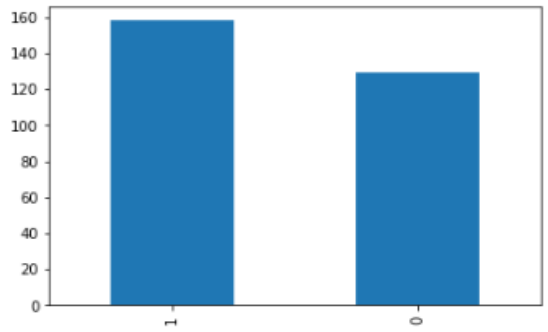
```
1 z = np.abs(stats.zscore(data))
2 rows = np.where((z > 3) | (z < -3))[0].tolist()
3
4 data = data.drop(rows)
5 data.shape
```

```
(287, 14)
```

سپس بررسی اینکه آیا تعداد نمونه ها متوازن هست یا نه ، که با بررسی ویژگی target اینکه اگر صفر باشه یعنی بیماری قلبی نیست و اگر یک بیماری قلب هست . و بله متوازن هستند و اگر اختلاف زیادی با هم داشتن نا متوازن بودن

```
data['target'].value_counts().plot(kind = 'bar')
```

```
<AxesSubplot:>
```



بررسی سوال ۲ »

نمونه های موجود در دیتاست با نسبت ۸۰ به ۲۰ به دو بخش داده های آموزشی و داده های تست تقسیم بندی کنم . با استفاده از پکیج sklearn از model_selection استفاده کردم(برای ایجاد رگرسیون و مدل خطی از کتابخانه Sklearn و زیر کتابخانه model_selectionاستفاده می شود. همچنین با استفاده از datasets به مجموعه‌ای از داده‌های موجود در این کتابخانه دسترسی خواهیم داشت. با انتخاب train_test_split نیز به امکان تفکیک داده‌ها به دو بخش آموزشی و آزمایشی بوجود می‌آید) به عنوان ورودی اول داده x می دهم و به عنوان ورودی دوم داده y بهش می دهم و تست سایز ۲۰ درصد و با shaffle جای سطر ها را عوض می کنم .

```
from sklearn.model_selection import train_test_split

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

بررسی سوال ۳ »

قضیه بیز :

نکاتی درباره قضیه بیز : تست ها همان رویداد نیستند- تست ها کامل نیستند- تست ها، احتمال تست را معین می کنندو نه احتمال واقعی – موارد مثبت نادرست موجب انحراف نتایج می شوند و افراد مختلف اعداد طبیعی را ترجیح می دهند- حتی علم نیز یک تست است .

کاربردهای قضیه بیز :قضیه بیز نتایج حاصل از تست‌ها را به احتمال‌های واقعی رویداد تبدیل می‌کند. با استفاده از قضیه بیز می‌توان:خطاهای اندازه‌گیری را اصلاح کرد: اگر مقدار احتمال واقعی و شانس بروز مثبت نادرست و منفی نادرست را بدانیم، می‌توانید خطاهای اندازه‌گیری را اصلاح کنیم.احتمال واقعی را به احتمال اندازه‌گیری شده تست ارتباط دهیم: قضیه بیز به ما امکان می‌دهد که PR(A|X) یعنی احتمال وقوع رویداد A را با فرض شرط X به PR(X|A) اتصال دهیم که شانس وقوع شرط X با فرض اتفاق افتادن رویداد A است. برای مثال با دانستن

نتایج تست ماموگرافی و شناخت نرخ خطا می‌توانیم احتمال واقعی رخداد سرطان را تشخیص دهیم.

قضیه بیز : اگر پیشامدهای B_1، B_2، . . .، و B_k افرازی از فضای نمونه‌ای S را تشکیل دهند

و به‌ازای $i = ۱, ۲, \cdots, k$، $P(B_i) \neq ۰$، آنگاه برای هر پیشامد A در S، به قسمی که $P(A) \neq ۰$،

$$P(B_r|A) = \frac{P(B_r) \cdot P(A|B_r)}{\sum_{i=۱}^{k} P(B_i) \cdot P(A|B_i)}, \quad r = ۱, ۲, \cdots, k$$

استفاده از قضیه بیز برای کلاس بندی

فرض کنید که K تا کلاس داریم. فرض کنید π_k توزیع پیشین کلاس kام باشد. یعنی احتمال اینکه یک داده‌ی تصادفی عضو کلاس kام باشد. فرض کنید $f_k(x) = \Pr(X = x|Y = k)$ توزیع متغیر ورودی برای داده های کلاس kام باشد. آنگاه طبق قضیه بیز خواهیم داشت:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{3.1}$$

تخمین π_k از روی نمونه های آموزشی بسیار ساده است. کافی است محاسبه کنیم که چه کسری از داده ای آموزشی برچسب کلاس kام را دارند. اما تخمین $f_k(x)$ به این سادگی ها نیست؛ مگر اینکه فرض کنیم دارای توزیع ساده ای باشد. به $\Pr(Y = k|X = x)$ احتمال پسین می گوییم. به ازای هر داده ی جدید این احتمال پسین را برای هر کدام از K کلاس بدست می آوریم و نهایتا داده را به کلاسی نسبت می دهیم که احتمال پسین بیشتری داشته باشد. این قانون تصمیم گیری دارای کمترین نرخ خطا است.

دسته بند بیز ساده گاوسی (Gaussian Naive Bayes)

اگر مشاهدات و داده‌ها از نوع پیوسته باشند، از مدل احتمالی با توزیع گاوسی یا نرمال برای متغیرهای مربوط به شواهد می‌توانید استفاده کنید. در این حالت هر دسته یا گروه دارای توزیع گاوسی 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 μ_k میانگین و σ_k^2 واریانس دسته kام 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:

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

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 $X = (x_1, \ldots, x_n)$ با احتمالات $(p_1, \ldots, p_n)$ 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.

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

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

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

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

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

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

نکته: با توجه به استقلال مشاهدات، تابع درستنمایی به صورت حاصلضرب نوشته شده است.

بررسی سوال ۴»

با در نظر گرفتن chol و trestbps و thalach و لیبل target یک دسته بند GaussianNB بدون استفاده از پکیج پیاده سازی و دردیتاست آموزشی اعضای مختلف قاعده بیز محاسبه شود.

```
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.
        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
```

بررسی سوال ۵ »

سپس پیاده سازی GaussianNB و آموزش آن بر روی آموزشی ۸۰ درصد دیتاست و نتایج 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.

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

```
{0: 101, 1: 128}
              precision    recall  f1-score   support

           0       0.50      0.04      0.07        28
           1       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
```

بررسی سوال ۶ »

همان کار سوال قبل با sklearn استفاده کنم به این شکل که یک Object از کلاس GaussianNB از sklearn.naive_bayes استفاده کردم و فیت کردم روی X_train و y_train

```
from sklearn.naive_bayes import GaussianNB

clf = GaussianNB()
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       0.72      0.54      0.62        24
           1       0.72      0.85      0.78        34

    accuracy                           0.72        58
   macro avg       0.72      0.70      0.70        58
weighted avg       0.72      0.72      0.72        58
```

بررسی سوال ۷ »

سایکیت لرن یه پکیج برای استفاده عموم میسازه طبیعتا سعی میکنه همچیو بهترین حالت ممکن پیش ببره. بعنوان مثال اگر تحقیقات نشون داده باشه فلان پیش پردازش روی داده ها میتونه خروجی رو بهتر کنه، اون پیش پردازش حتما در پکیج سایکیت لرن بطور پیش فرض اتفاق میفته ...

منتها وقتی ما برنامه نویسی میکنیم صرفا هدف خروجی گرفتنه، و دنبال این نیستیم که بهترین حالت ممکن اتفاق بیفته (هر کدوم از پکیج های سایکیت لرن نتیجه مداتها کار تحقیقاتیه ...) و نتیجا خروجی آن بهتر است .

بررسی سوال ۸ »

کلاسیفایر svm 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 precision، recall، f1 score 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
```

بررسی سوال ۹ »

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

در این سوال من ۲ تا کرنل linear و polynomial بررسی کردم و Performance مورد بررسی قرار دادم .

یکی دیگر از راه هایی که برای دسته‌بندی چنین داده هایی استفاده می شود، استفاده از توابع هسته(kernel function) برای گسترش فضاست.

حالا اینکه کرنل در svm کارش این هست که فضای داده را عوض می کند و می برد در دستگاه دیگه و یا همان فضای دیگه در واقع داده ها را نگاشت می دهد در فضای دیگه و کار طبقه بندی در فضای جدید انجام می دهد. به عنوان مثال یک فضای دو بعدی در نظر بگیریم با فرض اینکه یک دایره داشته باشیم و یک دسته داخل دایره و دسته دیگر روی محیط دایره و بخواهیم طبقه بندی روی این دو تا انجام بدهیم در واقع اون مرز تصمیم گیری همون دایره است و یک چیز پیچیده است و کرنل بزنیم بریم در فضای سه بعدی و مرز تصمیم گیری تبدیل می شود به یک صفحه و مرز تصمیم گیری ساده تر می شود. و فضای را نگاشت می دهیم و با مرز تصمیم گیری معادله اش را معکوس می دهیم به فضای اولیه یا طبق توضیحات استاد او ن خط تصمیم تبدیل می کنیم به یک رویه و داده ها راحت تر می شود جدا شودو به جای ضرب داخلی x, xi کرنل را قرار می دهیم .

یکی از مزیت های استفاده از تابع‌های کرنل برای گسترش فضا در مقایسه با استفاده از توان های بالاتر ویژگی ها (features)، مزیت محاسباتی است که در حالت استفاده از کرنل تنها نیاز به محاسبه $\binom{n}{2}$ تابع کرنل داریم. اما وقتی برای گسترش فضا از توان های چندم ویژگی ها استفاده می کنیم، ممکن است محاسبات بسیار زیاد شود.

```
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
```

بررسی سوال ۱۰ »

دسته بند svm را با استفاده از پکیج sklearn بسازید و با در نظر گرفتن کلیه فیچر ها دیتاست بر روی داده آموزشی ترین کنید نتایج بر روی داده تست ارزیابی کنید .

طبیعتا معیار ها خیلی بهتر می شود چون روی کل داده است.

```python
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
```

بررسی سوال ۱۱ »

برای سوال ۱۰ یکبار مدل را با 5 – fold crossvalidtion اجرا کنید و نتایج گزارش دهید – با استفاده از کل دیتا ست .

برای این سوال طبق درس :

توازن بایاس-واریانس در ارزیابی k-fold CV

قبلا گفته شد که CV به لحاظ محاسباتی به LOOCV ارجحیت دارد. مزیت دیگر CV به LOOCV این است که CV تخمین دقیق‌تری از نرخ خطای تست ارائه می‌دهد و این به دلیل توازن بایاس-واریانس است.
همان‌طور که گفته شد، روش ارزیابی ممکن است نرخ خطای تست را بیش‌تر تخمین بزند، چون از تمام داده‌های آموزشی برای آموزش استفاده نمی‌کند و بنابراین دارای بایاس زیادی است. برعکس،، روش LOOCV به دلیل استفاده از تقریبا تمام داده‌ها برای آموزش، تخمین ناریبی از خطای تست می‌دهد. با این منطق می‌توان نتیجه گرفت که CV با k=5 یا k=10 بایاس متوسطی دارد زیرا تعداد داده‌هایی که برای آموزش استفاده می‌کند $\frac{(k-1)n}{k}$ داده‌هاست که کمتر از داده‌های استفاده شده در LOOCV و بیشتر از داده‌های روش ارزیابی است. بنابراین اگر تنها هدف ما کاهش بایاس باشد، روش LOOCV ارجح است. اما بایاس تنها منبع ایجاد خطای تخمین نیست و باید به واریانس تخمین هم توجه کنیم. روش LOOCV دارای واریانس بیشتری نسبت به روش CV است زیرا در روش LOOCV میانگین خروجی n مدل را محاسبه می‌کنیم. با توجه به اینکه این n مدل دارای هم‌پوشانی بسیار زیاد در داده‌های آموزشی هستند، خروجی‌هایشان بسیار به هم وابسته است. این وابستگی بین خروجی‌ها در روش CV کم‌تر است زیرا در روش CV بین داده‌های آموزشی هر k مدل، هم‌پوشانی کم‌تری وجود دارد. از آن‌جایی که میانگین مقادیری که بسیار به هم وابسته هستند، دارای واریانس زیادی است، بنابراین تخمین روش LOOCV دارای واریانس زیادی است.
به طور خلاصه، انتخاب k در CV میزان بایاس و واراینس را تغییر می‌دهد. انتخاب k=5 یا k=10 انتخاب‌های معقولی هستند زیرا نه بایاس بالایی دارند و نه واریانس بالا.

--

دیتا ست به ۵ قسمت قسمت می کنیم و در ۵ مرحله مختلف آموزش می دهیم و به این شکل که هر بار 1/5 این داده ها را به عنوان ۲۰ درصد داده validation در نظر می گیریم و برای مثال اگر ۱۰۰ تا داده باشه بار اول ۲۰ تا اول برای validation استفاده می شود و ۸۰ تا بعدی برای ترین و ۲۰ تا بعدی و ۶۰ تا و ۲۰ تا بعدی ترین و....

```
from sklearn.model_selection import cross_val_score

clf = SVC(kernel='linear')
scores = cross_val_score(clf, X_train, y_train, cv=5)

print(f"mean score: {scores.mean()}")
print(f"score std: {scores.std()}")
```

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

بررسی سوال ۱۲ «

```
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
```

بررسی سوال ۱۳ «

با تعداد همسایه های مختلف بررسی کردم و ۳ بهترین score را می داد و تعداد همسایه حداقل فاصله بین اونها وجود داشته باشد و یک دیتا یا sample در فضای داده داشته باشم که مثلا این حداقل فاصله در نظر بگیریم . تعداد همسایه بالا برود عضویت در کلاس سخت و بر عکس . الگوریتم supervise و تعداد دسته ها از قبل مشخص است .

بررسی سوال ۱۴ «

```
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,random_state=42)

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       0.54      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
```

بررسی سوال ۱۵ «

در حوزه «تجزیه و تحلیل آماری داده‌ها» (Statistical Data Analysis)، توزیع جامعه آماری که نمونه از آن گرفته شده، مهم است زیرا هر چه اطلاعات بیشتر در زمینه رفتار داده‌ها و شکل پراکندگی و توزیع آن‌ها وجود داشته باشد، نتایج قابل اعتمادتر و دقیق‌تر خواهند بود. در مقابل، وجود اطلاعات کم از توزیع جامعه آماری مربوط به نمونه، باعث کاهش اعتماد به نتایج حاصل از روش‌های معمول (پارامتری) آماری می‌شود. بنابراین در این حالت مجبور به استفاده از روش‌های ناپارامتری هستیم که برای اجرای آن‌ها فرضیاتی در مورد توزیع داده‌ها وجود ندارد. به همین علت به روش‌های ناپارامتری گاهی «روش‌های توزیع-آزاد» (Distribution-free Methods) نیز می‌گویند.

آمار پارامتری و روش‌های تجزیه و تحلیل مرتبط

داده‌های پارامتری به نمونه‌ای گفته می‌شود که از توزیع جامعه آماری آن مطلع هستیم. معمولا این توزیع آماری برای داده‌های کمی، نرمال یک یا چند متغیره در نظر گرفته می‌شود. در این حالت از آزمون‌های آماری پارامتری مثل آزمون T، آزمون F و یا آزمون Z استفاده می‌کنیم. همچنین برای اندازه‌گیری میزان همبستگی بین متغیرهای دو یا چند بعدی نیز از ضریب همبستگی پیرسون استفاده خواهیم کرد.

اگر حجم نمونه در روش‌های تجزیه و تحلیل آمار پارامتری بزرگ انتخاب شود، معمولا توان آزمون مناسب خواهد بود و به راحتی می‌توان نتایج حاصل از آزمون فرض را به جامعه نسبت داد.

آمار ناپارامتری و روش‌های تجزیه و تحلیل مرتبط

اگر توزیع جامعه آماری نامشخص باشد و از طرفی حجم نمونه نیز کوچک باشد بطوری که نتوان از قضیه حد مرکزی برای تعیین توزیع حدی یا مجانبی جامعه آماری، استفاده کرد، از تحلیل‌های ناپارامتری استفاده می‌شود، زیرا در این حالت کارآمدتر از روش‌های پارامتری هستند. به این ترتیب در زمانی که توزیع جامعه مشخص نباشد و یا حجم نمونه کم باشد، روش‌ها و آزمون‌های ناپارامتری نسبت به روش‌ها و آزمون‌های پارامتری از توان آزمون بیشتری برخوردارند و نسبت به آن‌ها ارجح هستند.

بهتر است شرایط بهره‌گیری از روش‌های ناپارامتری را به صورت زیر لیست کنیم:

برای داده‌ها، نتوان توزیع آماری مناسبی در نظر گرفت.

وجود داده‌های پرت (Outlier)، وجود چند نما و ... امکان انتخاب توزیع نرمال را برایشان میسر نمی‌کند.

کم بودن حجم نمونه برآورد پارامترهای توزیع نرمال مانند میانگین و بخصوص واریانس را دچار مشکل می‌کند و در عمل امکان بررسی توزیع نرمال به علت حجم کم نمونه برای جامعه وجود ندارد.

روش‌های ناپارامتری در چنین موقعیت‌هایی می‌تواند راهگشا باشد و به محقق و «تحلیل‌گر داده» (Data Scientist) برای شناخت داده‌ها یاری برساند.
باید توجه داشت که اگر توزیع جامعه آماری قابل تحقیق و تعیین باشد، اجرای روش‌های پارامتری بر روش‌های ناپارامتری ارجح هستند زیرا در این حالت روش‌های پارامتری نسبت به روش‌های ناپارامتری از دقت بیشتری برخوردارند. بنابراین فقط زمانی که از توزیع جامعه آماری مطلع نیستم، به اجبار از روش‌های ناپارامتری استفاده خواهیم کرد. البته اگر حجم نمونه بزرگ باشد، در اکثر موارد، نتایج حاصل از آزمون‌های پارامتری و ناپارامتری با یکدیگر همخوانی دارند.

کاربردهای رگرسیون:

مدل های رگرسیونی برای مقاصدی چند مشتمل بر موارد زیر مورد استفاده قرار می گیرند .

۱- توصیف داده ها ۲- برآورد پارامترها ۳- پیشگویی و برآورد ۴- کنترل

دو بخش اصلی رگرسیون در آمار وجود دارد : پارامتری و ناپارامتری. در رگرسیون پارامتری نوع ارتباط بین متغیر های وابسته و مستقل شناخته شده است، اما ممکن است پارامترها مقادیری را شامل شوند که ناشناخته بوده و صلاحیت برآورد مجموعهء داده ها را نداشته باشند. برای مثال یک خط راست برازش داده شده،

$f(x)=ax+b$

بر حسب یک دسته از نقاط،

$\{(xi , \hat{y}i)\} : i=1,...,p$

رگرسیون پارامتری می باشد چرا که نوع ارتباط وابستگی y را روی x نشان می دهد هر چند تمام مقادیر a و b نیستند. نوعاً در هر مسئله پارامتری معین، پارامترهای آزاد بهتر از متغیر های وابسته و مستقل دارای تفسیر معنادار هستند،

معـدل گیـری موضـعی برآورگرهـا ی کرنـل، رگرسـیون ناپـارامتری نیرومنـد، رگرسـیون و همـوار سـازی دسـته هـای باریـک، اسـتناج آمـاری بـرای رگرسـیون ناپـارامتری در تجزیـه و تحلیـل داده هـا ، رگرسـیون چنـد متغیـر ناپـارامتری بـه انضـمام مـدلهای رگرسـیون افزایشـی ، رگرسـیون ناپـارامتری تعمـیم یافتـه و مـدلهای تعمـیم یافتـه افزایشـی. رگرسـیون ناپـارامتری معمـولاً در فرضـیات خطـی آزاد می باشد و شـما را بـه شـرح داده هـای بصـری ، سـاختار غیرپوششـی در داده هـا کـه ممکـن اسـت بـه نحـوی گمشـده باشـد ، قـادر مـی سازد.

بنابراین خیلی از روشهای رگرسـیون ناپـارامتری هنگـامی کـه تعـداد متغیـر هـای مسـتقل در مـدل زیـاد مـی باشـد بـه خـوبی اجـرا نمـی شوند. پراکندگی داده ها در ایـن مجموعـه سـبب مـی شـود بـر آوردهـای واریـانس بـه انـدازه غیـر قابـل پـذیرش بـزرگ شـود، مگـر آنکـه حجم نمونه فوق العاده بـزرگ باشـد. قابلیـت تفسـیر یکـی دیگـر از مسـایل رگرسـیون ناپـارامتری اسـت کـه بـر پایـه کرنـل و هموارسـازی برآوردگرهای خط sp می باشد. اطلاعـات ایـن بـرآورد گرهـا شـامل رابطـه بـین متغیرهـای مسـتقل و وابسـته مـی باشـد کـه اغلـب درک آنها دشوار است

بررسی سوال ۱۶ »

منحنی های ROC برای مقایسه مدل ها بهترین انتخاب هستند. با این حال ، معیارهای اسکالر همچنان در بین جامعه یادگیری ماشین محبوب هستند و چهار رایج ترین آنها عبارتند از: دقت ، یادآوری ، دقت و نمره F1 (*accuracy*, *recall*, *precision*, and *F1-score*) ,یک معیار اسکالر دیگر برای طبقه بندی باینری وجود دارد و آن ضریب همبستگی Matthews (MCC) است.

در طبقه بندی باینری ما دو کلاس داریم: به اصـطلاح کلاسـهای مثبت و منفی. مفید اسـت که در مورد معیارهای طبقه بندی با اسـتفاده از ماتریس سـردرگمی صحبت کنیم ، که ما پس از تعیین آسـتانه طبقه بندی برای طبقه بندی باینری خود ، به این حسـاب می پردازیم. ماتریس سـردرگمی دارای ۴ مقدار اسـت که مربوط به ۴ ترکیب کلاس واقعی و پیش بینی شـده اسـت. در اینجا یک ماتریس سـردرگمی معمولی وجود دارد ، که TP، FP، FN و TN نشان دهنده چهار ترکیب است.

		True/Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Confusion Matrix

مثال : به یک مثال اسباب بازی نگاه کنیم: داده های ما تصاویری از حیوانات خانگی است ، یا یک سگ ( ) ، یا یک گربه (). طبقه بندی کننده ما در هر عکس حیوان خانگی را تشخیص می دهد و ما می خواهیم عملکرد آن را بسنجیم.

در اینجا ماتریس سردرگمی طبقه بندی عکس ما وجود دارد. در مجموع ۲۴ عکس ، ۱۸ + ۲ = ۲۰ عکس سگ و ۳ + ۱ = ۴ عکس گربه داریم.

		True/Actual	
		Positive ( )	Negative ( )
Predicted	Positive ( )	18 (TP)	3 (FP)
	Negative( )	2 (FN)	1 (TN)

Dogs are "positive"

Precision, Recall, and F1-score

precision نسبت true positivesشناسایی شده یا به سادگی TP / (TP + FP) است. عکس های سگ طبقه مثبت هستند و ۱۸ عکس از ۱۸ + ۳ عکس که به عنوان سگ طبقه بندی شده اند در واقع حاوی سگ هستند. بنابراین دقت ۲۱/۱۸ = ۸۶٪ است. recallتعداد مثبت های واقعی است که به درستی طبقه بندی شده اند (TP / (TP + FN)). از ماتریس فوق به راحتی می توان دریافت که ۲۰ مثبت وجود دارد و ۱۸ مورد از آنها با موفقیت شناسایی می شوند. بنابراین ، recall ۱۸ / (۱۸ + ۲) ، یا ۹۰٪ است.، نمره F1 میانگین هماهنگی precision و recall است. این ۸۸٪ محاسبه می شود. نگاهی دوباره به ماتریس ، به ویژه طبقه بندی عکس گربه ها بیندازید. از هر ۴ عکس گربه فقط ۱ مورد با موفقیت شناسایی شد. علاوه بر این ، از ۳ عکس طبقه بندی شده ۲ گربه در واقع سگ هستند. پس چرا نمره F1 اینقدر بالا است؟

precision و recall (،نمره F1 ، که تابعی از این دو است) یک کلاس ، کلاس مثبت را کلاس مورد نظر ما می دانیم. آنها فقط از سه مقدار در ماتریس سردرگمی استفاده می کنند: TP، FP و FN. مقدار چهارم - TN - در این سنجه ها استفاده نمی شود. می توانید هر مقداری را در سلول TN قرار دهید - ۰ ، ۱۰۰ ، بی نهایت - و precision و recall و نمره F1 تغییری نمی کند.

اکنون در ماتریس سردرگمی بیایید "گربه" را یک کلاس مثبت بدانیم ، در اینجا ماتریس سردرگمی جدید است. این دقیقاً همان طبقه بندی قبلی است.

		True/Actual	
		Positive ( )	Negative ( )
Predicted	Positive ( )	1 (TP)	2 (FP)
	Negative( )	3 (FN)	18 (TN)

Cats are "positive"

یک محاسبه سریع نشان می دهد که اکنون precision ٪۳۳ است ، ٪۲۵ recallو F1 نیز ٪۲۹ است طبقه بندی کننده ما در طبقه بندی گربه ها وحشتناک است.

Accuracy

در کل (TP + FP) + (FN + TN) = 20 + 4 = 24 نمونه وجود دارد و TP + TN = 19 به درستی طبقه بندی شده اند. بنابراین Accuracy ٪۷۹ قابل توجه است. اما این کاملا گمراه کننده است ، از آنجایی که ٪۹۰ سگها به طور دقیق طبقه بندی می شوند ، این فقط ٪۲۵ برای گربه ها است. اگر به طور متوسط ٪۹۰ و ٪۲۵ باشید ، به طور متوسط ٪۵۷.۵ دقت خواهید داشت که بسیار کمتر از دقت ٪۷۹ طبقه بندی کننده است. و دلیلش؟ در مجموعه داده های ما نمونه های سگ بسیار بیشتری نسبت به نمونه های گربه وجود دارد. کلاسها نامتعادل هستند.

برای دیدن تأثیر عدم تعادل کلاس بر دقت ، تصور کنید که اکنون به جای ۴ عکس گربه ، ۱۰۰ مجموعه از این ۴ عکس در مجموع ۴۰۰ عکس داشته باشیم. از آنجا که ما از یک طبقه بندی کننده یکسان استفاده می کنیم ، ۱۰۰ عکس از ۴۰۰ عکس به درستی طبقه بندی می شوند و ۳۰۰ طبقه بندی نامناسب انجام می شود. در اینجا ماتریس سردرگمی مربوطه وجود دارد:

		True/Actual	
		Positive (🐶)	Negative (🐱)
Predicted	Positive (🐶)	18 (TP)	300 (FP)
	Negative(🐱)	2 (FN)	100 (TN)

So many cats!

A quick calculation shows that the accuracy is now a much lower $(100+18)/(400+20)=28\%$, because cats are now the majority class. A new class proportion will also influence the precision (but not the recall check!), and thus the F1-score.

Matthews Correlation Coefficient to the Rescue

برخی از معیارهای کلاسیک را مشاهده کرده ایم: accuracy به عدم تعادل کلاس حساس است. دقت ، recallو نمره F1 نامتقارن است. پس چه کاری باید انجام شود؟ اگر هر دو کلاس مورد نظر باشد ، می توان یک مسئله طبقه بندی باینری را به

عنوان یک مسئله چند طبقه با ۲ کلاس در نظر گرفت و سپس معیارهای مربوط به چند کلاس را محاسبه کرد: micro- or macro-averaged precision, recall, and F1-score.

برای طبقه بندی باینری ، یک راه حل دیگر (و مسلماً ظریف تر) وجود دارد: با کلاس واقعی و کلاس پیش بینی شده به عنوان دو متغیر (باینری) رفتار کنید و ضریب همبستگی آنها را محاسبه کنید (به روشی مشابه با محاسبه ضریب همبستگی بین هر دو متغیر). هرچه همبستگی بین مقادیر واقعی و پیش بینی شده بیشتر باشد ، پیش بینی بهتر خواهد بود. این ضریب phi (ϕ) است که در طبقه بندی ها اعمال می شود و ضریب همبستگی Matthews (MCC) را دوباره تعویض می کنیم.

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

برخی از خصوصیات خوب MCC را می توان به راحتی از این فرمول بدست آورد: وقتی طبقه بندی کننده کامل باشد (FP = FN = 0) مقدار MCC 1 است ، این نشان دهنده همبستگی مثبت کامل است. برعکس ، وقتی طبقه بندی کننده همیشه طبقه بندی نادرستی انجام می دهد (TP = TN = 0) ، مقدار -۱ به دست می آید که نشان دهنده همبستگی منفی کامل است (در این حالت ، برای بدست آوردن طبقه بندی ایده آل می توانید به راحتی نتیجه طبقه بندی را معکوس کنید). در حقیقت ، مقدار MCC همیشه بین ۱ تا -۱ است ، با ۰ به این معنی است MCC نیز کاملاً متقارن است ، بنابراین هیچ کلاسی از کلاس دیگر مهمتر نیست. اگر مثبت و منفی را تغییر دهید ، باز هم همان مقدار را خواهید گرفت.

MCC هر چهار مقدار را در ماتریس سردرگمی در نظر می گیرد ، و مقدار زیاد (نزدیک به ۱) به این معنی است که هر دو کلاس به خوبی پیش بینی شده است ، حتی اگر یک کلاس به طور نامتناسبی کمتر از (یا بیش از حد) نشان داده شود.

We're all set now! Let's calculate the MCC for our original classifier. Here's the confusion matrix:

		True/Actual	
		Positive (🐶)	Negative (🐱)
Predicted	Positive (🐶)	18 (TP)	3 (FP)
	Negative(🐱)	2 (FN)	1 (TN)

Plugging in the numbers, we find:

$$MCC = \frac{18 \times 1 - 3 \times 2}{\sqrt{(18+3)(18+2)(1+3)(1+2)}} = 0.17$$

```
>>>
>>> from sklearn.metrics import matthews_corrcoef, confusion_matrix
>>> confusion_matrix(y_true, y_pred)
array([[ 1,  3],
       [ 2, 18]])
>>> matthews_corrcoef(y_true, y_pred)
0.1690308509457033
>>>
>>>
```

یکی از روش‌های بررسی و ارزیابی عملکرد دسته‌بندی دو دویی، «نمودار مشخصه عملکرد (Receiver Operating Characteristic) «یا به اختصار منحنی ROC است. کارایی الگوریتم‌های «دسته‌بندهای دو دویی (Binary Classifier) «معمولا توسط شاخص‌هایی به نام «حساسیت (Sensitivity) «یا «بازیابی (Recall) «سنجیده می‌شود. اما در نمودار ROC هر دوی این شاخص‌ها ترکیب شده و به صورت یک منحنی نمایش داده می‌شوند. اغلب برای بررسی کارایی الگوریتم‌های دسته‌بندی یا ایجاد داده‌های رسته‌ای از منحنی ROC استفاده می‌کنند

جدول ۳: شاخص‌های ارزیابی دسته‌بندی

نام شاخص	شرح	نحوه محاسبه
Accuracy (ACC)	صحت یا دقت	$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$
Balanced accuracy (BA)	صحت یا دقت متعادل	$\mathrm{BA} = \frac{TPR+TNR}{2}$
F1 score	امتیاز اف وان- میانگین توافقی دقت و حساسیت	$\mathrm{F_1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$
Matthews correlation coefficient (MCC)	ضریب همبستگی ماتیوس	$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$
Fowlkes–Mallows index (FM)	شاخص فولکس-مالوز	$\mathrm{FM} = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}} = \sqrt{PPV \cdot TPR}$
Informedness or bookmaker informedness (BM)	آگاهی بخشی یا نشانگر آگاهی بخشی	$\mathrm{BM} = \mathrm{TPR} + \mathrm{TNR} - 1$
Markedness (MK) or deltaP	علامت‌داری یا دلتای پی	$\mathrm{MK} = \mathrm{PPV} + \mathrm{NPV} - 1$

فضای ROC