تمرین شماره 1

علیرضا حسینی

شماره دانشجویی : ۸۱۰۱۰۱۱۴۲

یادگیری ماشین

دکتر ابوالقاسمی و دکتر توسلی پور

زمستان 1401

فهرست مطالب

۱-۱- سوال 1 : طبقه بندی دو کلاسه برای تابع تویع کوشی 5
۱-۲- سوال 2 : مرز تصمیم برای مساله 2 کلاسه رایلی..... 9
۱-۳- سوال 3 : مرز تصمیم برای مساله 2 کلاسه گوسی..... 9
۱-۴- سوال 4 : متغیر تصادفی پواسن..... 15
۱-۵- سوال 5 : آشنایی با طبقه بند naiive bayes..... 18
-۶-۱ سوال 6 : طبقه بندی تصاویر جنگل و دریا..... 28

فهرست اشکال

شکل (۱-۱) کد پایتون رسم P(w1|x) و P(w2|x) 6
شکل (۱-۲) P(w1|x) و P(w2|x) روی یک محور 6
شکل (۱-۳) منحنی P(w1|x) و p(w2|x) با استفاده از ماتریس مینیمم ریسک 9
شکل (۱-۴) رسم داده های سوال 3 در matplotlib 10
شکل (۱-۵) کد پایتون رسم داده ها و مرز تصمیم 12
شکل (۱-۶) مرز تصمیم برای حالت برابر بودن احتمال prior 12
شکل (۱-۷) مرز تصمیم برای حالت مینیم ریسک 13
شکل (۱-۸) مرز تصمیم برای حالت (p1= 1/3 , p2 = 2/3) 14
شکل (۱-۹) نمایی از داده های penguins.csv 21
شکل (۱-۱۰) کد پایتون جایگزینی x ها با NAN و حذف آن از دیتافریم 21
شکل (۱-۱۱) نمایی از دیتافریم پس از حذف سطر های حاوی 'X' 21
شکل (۱-۱۲) کد پایتون کد گذاری داده های هدف و تقسیم داده ها به 2 بخش ویژگی و هدف 22
شکل (۱-۱۳) کد پایتون نرمال سازی داده های X 22
شکل (۱-۱۴) کد پایتون تقسیم داده ها به تست و آموزش 22
شکل (۱-۱۵) محاسبه احتمال prior 23
شکل (۱-۱۶) محاسبه میانگین و واریانس 23
شکل (۱-۱۷) تعریف تابع naiive bayes 23
شکل (۱-۱۸) استفاده از طبقه بند جهت پیش بینی روی مجموعه تست 23
شکل (۱-۱۹) استفاده از کتاب خانه sklearn جهت محاسبه متریک ها 24
شکل (۱-۲۰) کد پایتون محاسبه متریک ها بنابر تعریف 24
شکل (۱-۲۱) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی بدون استفاده از کتابخانه) 25
شکل (۱-۲۲) کد پایتون استفاده از GussianNB و گرفتن خروجی ها بر روی داده های تست 25
شکل (۱-۲۳) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی با استفاده از کتابخانه) 25
شکل (۱-۲۴) کد پایتون 1vsall 27
شکل (۱-۲۵) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی با استفاده از کتابخانه و 1vsALL) 27
شکل (۱-۲۶) کد پایتون اختصاص لیبل به تصاویر 28
شکل (۱-۲۷) تشکیل لیست ویژگی ها با مقدار میانگین کانال آبی تصاویر دیتاست 29
شکل (۱-۲۸) کد پایتون رسم ویژگی ها بر حسب لیبل ها 29
شکل (۱-۲۹) منحنی ویژگی ها بر حسب لیبل ها 30
شکل (۱-۳۰) کد پایتون تعریف طبقه بند بیز 30
شکل (۱-۳۱) کد پایتون پیدا کردن بهترین threshold 31
شکل (۱-۳۲) کد پایتون تست طبقه بند دریا و جنگل 31

شکل (۱-۳۳) مقادیر بهترین ترشولد و دقت و Cm و pre و recall در طبقه بند دریا و جنگل 31
شکل (۱-۳۴) پیدا کردن تصاویری که شبکه عملکرد مناسبی روی آن ها نداشته است. 32
شکل (۱-۳۵) تصاویر دریا هایی که جنگل تشخیص داده شده است..... 32
شکل (۱-۳۶) تصاویر جنگل هایی که دریا تشخیص داده شده است..... 33

۱-۱- سوال 1 : طبقه بندی دو کلاسه برای تابع تویع کوشی

برای نشان دادن اینکه p(w1|x) = p(w2|x)، باید دو احتمال پسین را با هم مقایسه کنیم.

$$p(w1) = p(w2) = \frac{1}{2}$$

$$p(wi\ |x) = \frac{p(x|wi)p(wi)}{p(x)}$$

ما می توانیم مخرج را نادیده بگیریم زیرا برای هر دو معادله یکسان است.

با توجه به p(w1) = p(w2)، فقط باید نشان دهیم که p(x|w1) = p(x|w2) برای نقطه (a1 + a2) / 2 برقرار است.

برای اینکه مقادیر و تابع توزیع داده شده را در معادله وارد میکنیم.

$$p(x|wi) = \frac{1}{\pi b}\frac{1}{(1+\frac{x-ai}{b}^2)}$$

در نقطه (a1 + a2) / 2 داریم:

$$p(x|w1) = \frac{1}{\pi b}\frac{1}{(1+\frac{\frac{a1+a2}{2}-a1}{b}^2)}$$

$$p(x|w1) = \frac{1}{\pi b}\frac{1}{(1+\frac{a2-a1}{2b}^2)}$$

$$p(x|w2) = \frac{1}{\pi b}\frac{1}{(1+\frac{\frac{a2+a1}{2}-a2}{b}^2)}$$

$$p(x|w2) = \frac{1}{\pi b}\frac{1}{(1+\frac{a2-a1}{2b}^2)}$$

بنابراین p(x|w1) = p(x|w2) در به ازای x= (a1+a2)/2 برابر است.این بدان معنی است که مرز تصمیم بین

دو کلاس در نقطه (a1 + a2) / 2 است.

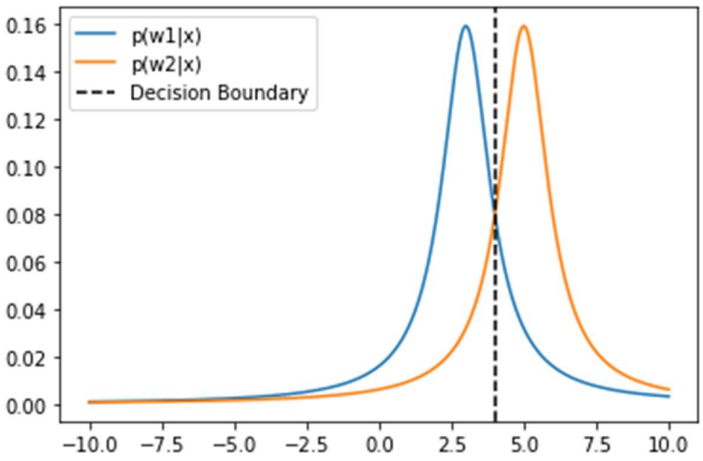
برای رسم احتمالات p(w1|x) و p(w2|x)، می‌توانیم مقادیر داده شده را جایگزین a1، a2 و b در عبارات بالا کرده و آنها را به عنوان تابعی از x رسم کنیم.

$$p(wi|x) = \frac{1}{\pi b}\frac{1}{\left(1+\frac{x-ai^2}{b}\right)}\frac{1}{2}$$

برای رسم از matplotlib در پایتون استفاده میکنیم.

```python
import numpy as np
import matplotlib.pyplot as plt

a1 = 3
a2 = 5
b = 1

x = np.linspace(-10, 10, 1000)

p_w1_given_x = 0.5/(np.pi * b) * 1/(1 + ((x-a1)/b)**2)
p_w2_given_x = 0.5/(np.pi * b) * 1/(1 + ((x-a2)/b)**2)

plt.plot(x, p_w1_given_x, label='p(w1|x)')
plt.plot(x, p_w2_given_x, label='p(w2|x)')
plt.axvline((a1+a2)/2, color='black', linestyle='--', label='Decision Boundary')
plt.legend()
plt.show()
```

شکل (۱-۱) کد پایتون رسم P(w1|x) و P(w2|x)



شکل (۱-۲) P(w1|x) و P(w2|x) روی یک محور

در ادامه برای محاسبه **حداقل احتمال خطا** داریم:

$$p(error) = \int_{-inf}^{\frac{a1+a2}{2}} p(w2|x)\,dx \quad + \int_{\frac{a1+a2}{2}}^{inf} p(w2|x)\,dx$$

$$p(error) = \int_{-inf}^{\frac{a1+a2}{2}} \frac{1}{\pi b}\frac{1}{\left(1+\frac{x-a1^2}{b}\right)}\frac{1}{2}dx + \int_{\frac{a1+a2}{2}}^{inf} dx \;\; \frac{1}{\pi b}\frac{1}{\left(1+\frac{x-a2^2}{b}\right)}\frac{1}{2}$$

به دلیل تقارن داریم :

$$p(error) = 2\int_{-inf}^{\frac{a1+a2}{2}} \frac{1}{\pi b}\frac{1}{\left(1+\frac{x-a1^2}{b}\right)}\frac{1}{2}dx$$

$$p(error) = 2\frac{1}{2\pi}(\arctan\frac{a1-a2}{2} + \arctan\,(inf)$$

$$p(error) = \frac{1}{2} - \frac{1}{\pi}\arctan\left(\frac{a1-a2}{2b}\right)$$

جهت محاسبه **حداکثر مقدار خطا** : حداکثر خطا زمانی رخ می دهد که میانگین دو کلاس یکسان باشد، یعنی a2 = a1. در این حالت، مرز تصمیم در x = a1 = a2 است و میزان خطا برابر با حداقل مقدار است که ½ است. (به عبارت دیگر ارگمان arctan وقتی صفر شود مقدار خطا بیشینه است که یا a1 = a2 باشد و 2 توزیع یکسان یا پارامتر b برابر بینهایت شود.

توجه داشته باشید که نتیجه فوق با این شهود سازگار است که طبقه بندی دو کلاس با میانگین های مختلف آسان تر از دو کلاس با میانگین یکسان است، زیرا مورد اول منجر به جدایی گسترده تر بین دو کلاس می شود.

طراحی طبقه بند:

در حالت کلی برای طبقه بند داریم.

$$w1 \quad ; \quad \frac{p(x|w1)}{p(x|w2)} > \frac{p(w1)}{p(w2)}$$

$$w2 \quad ; \quad other\ wise$$

حالت اول P(w1) = P(w2) : طرف سمت راست نامساوی 1 شده و همان تصویر 1 و مقدار x = (a1+a2)/2

به عنوان مرز تصمیم گیری انتخاب میشود و احتمال خطا به صورت زیر محاسبه میشود.

$$p(error) = \int R2\ p(w1|x)p(x)dx + \int R1\ p(w2|x)p(x)dx$$

که با توجه به تقارن R1 و R2 و جایگذاری عبارات فوق در نهایت به این نتیجه میرسیم که احتمال خطا در

این حالت برای طبقه بند بیز همان مینیمم خطا محاسبه شده در بخش قبل خواهد بود.

$$p(error) = \frac{1}{2} - \frac{1}{\pi}\arctan\left(\frac{a1 - a2}{2b}\right)$$

طراحی طبقه بند با کمینه کردن ریسک

با توجه به مساله فوق برای به حداقل رساندن ریسک با وزن های داده شده، باید از قانون تصمیم گیری زیر

استفاده کنیم:

اگر P(w1|x) * L21 > P(w2|x) * L12، سپس w2 انتخاب شود.

اگر P(w1|x) * L21 < P(w2|x) * L12، آنگاه w1 انتخاب شود.

اگر P(w1|x) * L21 = P(w2|x) * L12، سپس w1 یا w2 را به طور تصادفی انتخاب شود.

با استفاده از قضیه بیز، می توانیم احتمالات پسین را به صورت زیر محاسبه کنیم:

سپس، می‌توانیم این احتمالات بعدی را به قاعده تصمیم متصل کنیم و مرز تصمیم را به صورت زیر بدست

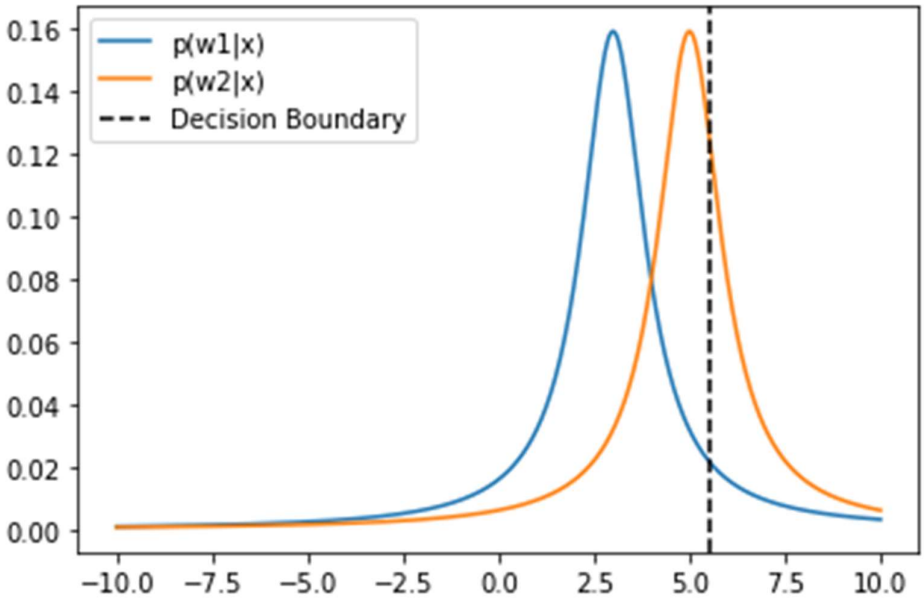
آوریم:

$$½ > \frac{p(x|w1)}{p(x|w2)}$$

مرز تصمیم گیری در حالت تساوی به دست میاید.

$$\frac{1}{\pi b}\frac{1}{(1 + \frac{x - a2^2}{b})} = 2\ \frac{1}{\pi b}\frac{1}{(1 + \frac{x - a1^2}{b})}$$

x = 4 ± (2/3) sqrt(8) = 5.51



شکل (۳-۱) منحنی P(w1|x) و p(w2|x) با استفاده از ماتریس مینیمم ریسک

۲-۱- سوال 2 : مرز تصمیم برای مساله 2 کلاسه رایلی

با توجه به فرض تساوی توزیع پیشین داریم مرز تصمیم به صورت زیر میشود.

$$p(x|w1) = p(x|w2)$$

$$\frac{x}{\delta 1^2} e^{-(x^2/2\delta 1^2)} = \frac{x}{\delta 2^2} e^{-(x^2/2\delta 2^2)}$$

$$x^2 \left(\frac{1}{2\delta 2^2} - \frac{1}{2\delta 2^2}\right) = \ln \frac{\delta 1^2}{\delta 2}$$

$$x = \left(\frac{\ln \frac{\delta 1^2}{\delta 2}}{\left(\frac{1}{2\delta 2^2} - \frac{1}{2\delta 2^2}\right)} \right)^{\frac{1}{2}}$$

۳-۱- سوال 3 : مرز تصمیم برای مساله 2 کلاسه گوسی

ابتدا باید میانگین و واریانس را محاسبه کنیم.

برای اینکار نقاط هر کلاس را روی محور های x و y مشخص کرده و برای هر کلاس میانگین روی x و y را محاسبه کرده و برای محاسبه کواریانس هم تمامی x ها را از میانگین اش کم کرده (همچنین برای y) و حاصل ضرب این دو را تقسیم بر(تعداد sample های آن کلاس منهای 1) میکنیم.

بدین ترتیب برای هر کلاس به صورت زیر محاسبه میشود. (تعداد نمونه ها)

$$mxi = \frac{1}{n}\sum xi \quad , \quad myi = \frac{1}{n}\sum yi$$

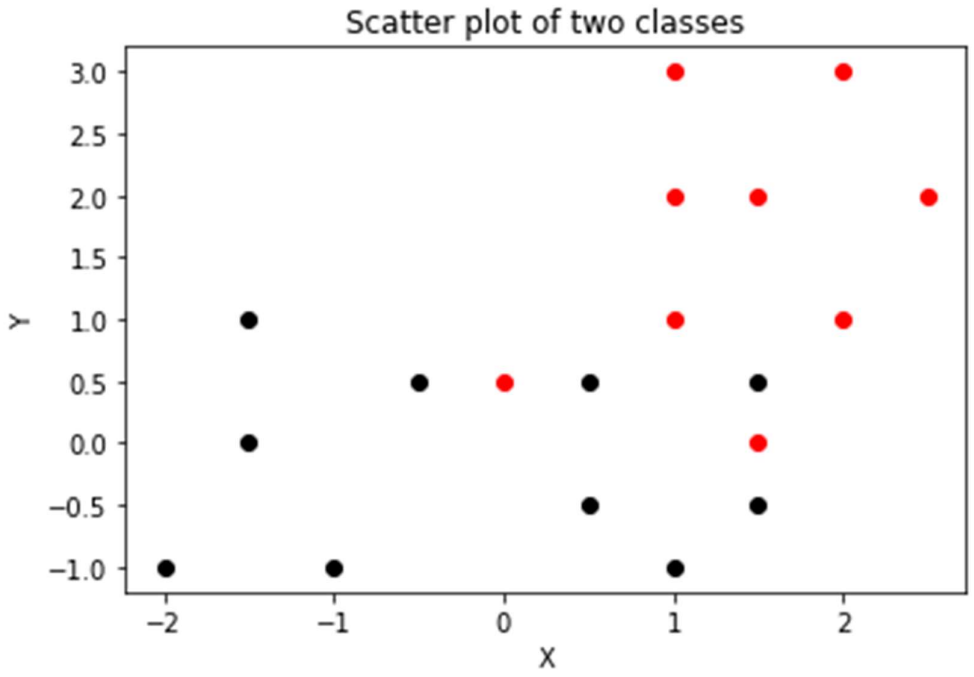
$$covi = \frac{1}{n-1}\sum (xi - mxi)(yi - myi)$$

با توجه به توضیحات فوق برای کلاس 1 داریم :

$$mx1 = -0.15\,, my1 = -0.15\,, cov1 = 0.0028$$

و برای کلاس 2 داریم :

$$mx2 = 1.39\,, my2 = 1.62\,, cov2 = 0.232$$



شکل (۴-۱) رسم داده های سوال 3 در matplotlib

مرز تصمیم با شرط احتمال پیشین مساوی و برابر 0.5 :

برای تشخیص مرز تصمیم نیاز به ماتریس کواریانس میباشد (var x1و var x2 و var y1 و var y2) و پس از محاسبه آن باید با استفاده از مرز تصمیم دو .discriminant func را به صورت زیر به دست آورده و با هم برابر قرار دهیم .

$$gi(x) = x^t Wi\, x + Wi^t x + wi0$$

$$\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1},$$

$$w_{i0} = -\frac{1}{2}\mu_i^t \Sigma_i^{-1} \mu_i - \frac{1}{2}\ln\,|\Sigma_i| + \ln\,P(\omega_i)$$

با توجه به روابط فوق برای کلاس 1 و 2 مقادیر رامحاسبه میکنیم.

$$\mathrm{W}_1 = \begin{bmatrix} -0.32206438 & 0.00178034 \\ 0.00178034 & -0.99503472 \end{bmatrix}$$
$$w_1 = [-0.09608521. -0.29797631]$$
$$w_{10} = -0.5985502971752543$$

$$g1 = -0.32\,x1^2 + 0.002\,x1\,y1 - 0.995\,y1^2 - 0.09\,x1 - 0.29\,y1 - 0.598$$

به همین ترتیب برای کلاس 2 :

$$g2 = -1.15\,x2^2 + 0.54\,x2\,y2 - 0.57\,y2^2 + 2.33\,x2 + 1.08\,y2 - 2.75$$

(تمامی محاسبات در متلب محاسبه شده است)

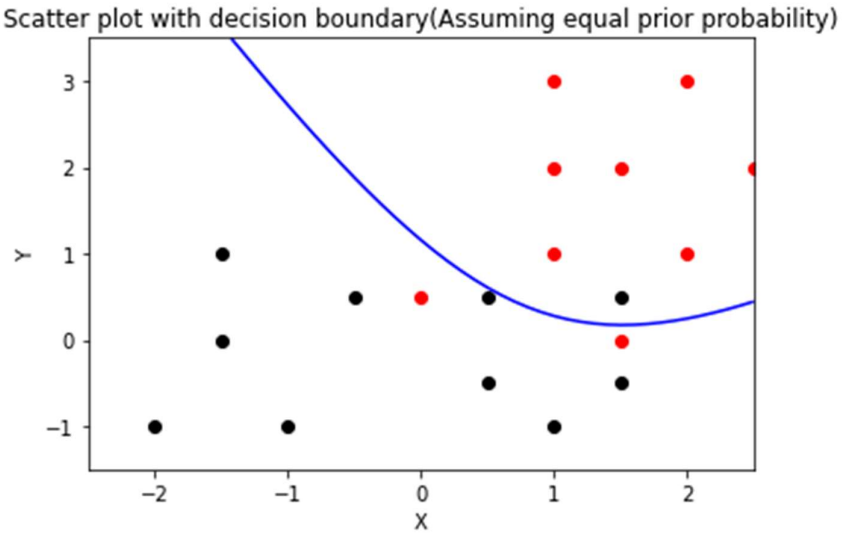
به دلیل یکسان بودن prior ها با برابر قرار دادن g2=g1 مرز تصمیم مشخص میشود.

که معادله در کد زیر آمده است.

برای رسم از کد زیر استفاده میکنیم . ابتدا معادله را به صورت تابع تعریف میکنیم.

```
####### p1 = p2 = 0.5
# Data for class 1
x1 = [-2.0, -1.0, -1.5, -1.5, -0.5, 0.5, 0.5, 1.5, 1.5, 1.0]
y1 = [-1.0, -1.0, 0, 1, 0.5, 0.5, -0.5, -0.5, 0.5, -1]
# Data for class 2
x2 = [0, 1.5, 1, 2, 1, 1.5, 2.5, 1, 2]
y2 = [0.5, 0, 1, 1, 2, 2, 2, 3, 3]

# Define function for decision boundary
def boundary(x, y):
    return 0.83*x**2 - 0.538*x*y - 2.42*x - 0.425*y**2 - 1.37*y + 2.152

plt.scatter(x1, y1, c='black')
plt.scatter(x2, y2, c='red')
x_range = np.arange(-2.5, 2.6, 0.1)
y_range = np.arange(-1.5, 3.6, 0.1)
# Create meshgrid
xx, yy = np.meshgrid(x_range, y_range)
# Evaluate decision boundary function at all points in meshgrid
zz = boundary(xx, yy)
# Plot decision boundary as contour plot
plt.contour(xx, yy, zz, levels=[0], colors='blue')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter plot with decision boundary(Assuming equal prior probability)')
plt.show()
```

شکل (۵-۱) کد پایتون رسم داده ها و مرز تصمیم



شکل (۶-۱) مرز تصمیم برای حالت برابر بودن احتمال prior

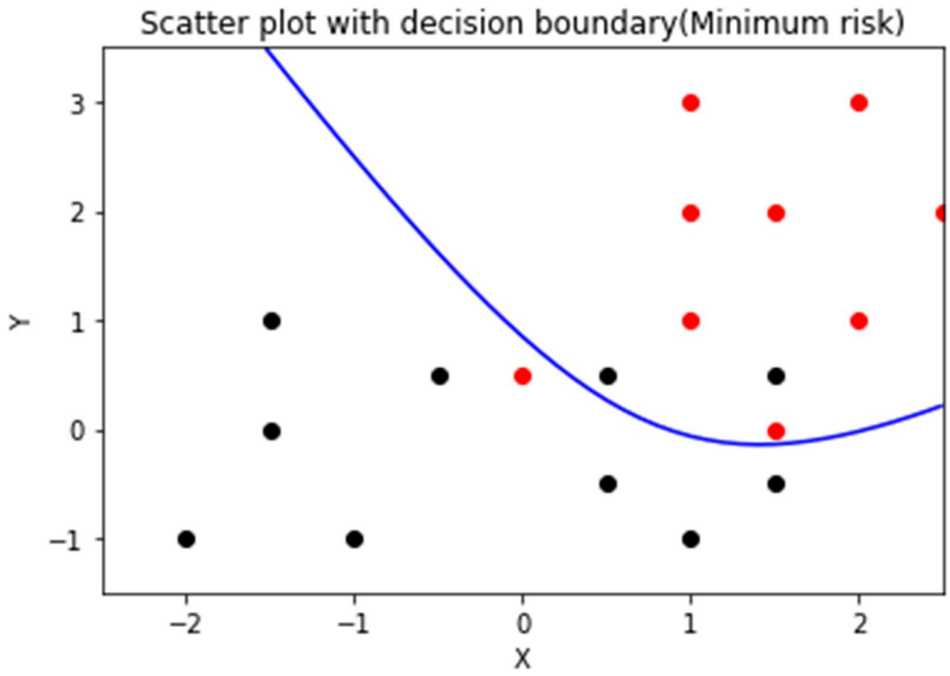
برای محاسبه خطای آموزشی تجربی باید گفت که در کلاس 1 ، یک نقطه در مرز تصمیم کلاس 2 میباشد و 10 درصد خطا دارد . در کلاس نیز 2 تا از 9 تا اشتباه تصمیم گیری میشود.

مرز تصمیم با کاهش ریسک داده شده :

وجود ریسک باعث میشود مقادیر w10 و w20 تغییر کند (مقادیر $\ln(\lambda 21 - \lambda 11)$ به w10 اضافه شده و $\ln(\lambda 21 - \lambda 11)$ به w20 اضافه میشود. (برای آنکه بتوانیم رسم کنیم a=1 فرض شده است)

بدین ترتیب مرز تصمیم معادله جدیدی میشود:

$$0.83\ x^2 - 0.538\ x\ y - 2.42\ x - 0.425\ y^2 - 1.37\ y + 1.468 = 0$$



شکل (۷-۱) مرز تصمیم برای حالت مینیم ریسک

مرز تصمیم P(w1) = 1/3 و P(w2) = 2/3 :

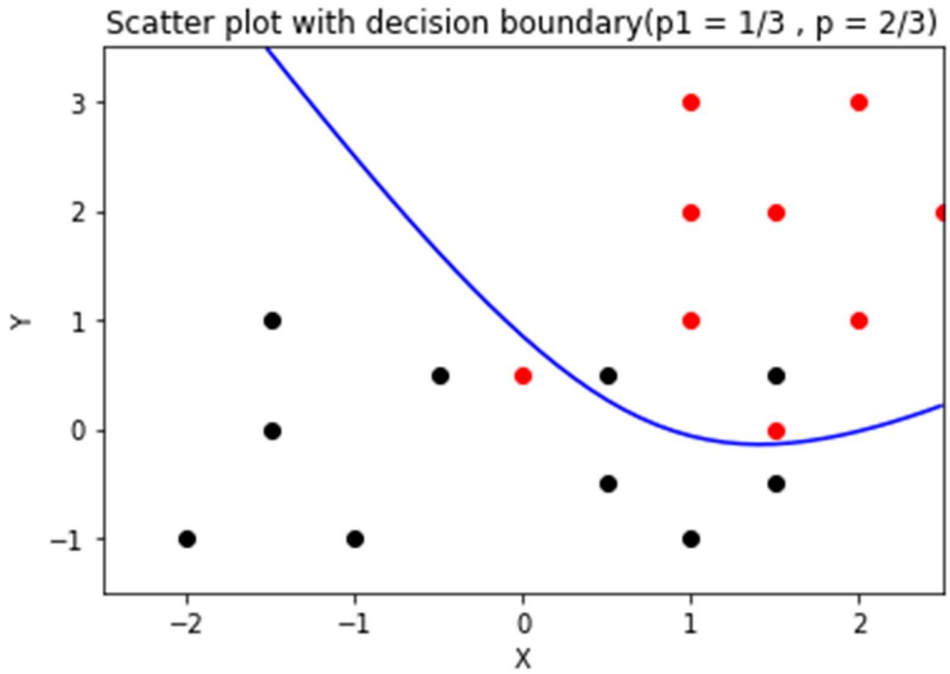
اگر در فرمول wi0 دقت شود ln pwi داریم و نسبت به حالت اول محاسبات متفاوت میشود چرا که معادلات قبلی با 0.5 محاسبه شده بود اما تفاوت فقط در یک constant اخر g ها متفاوت میشود.

پس از انجام محاسبات داریم :

$$g1 = -0.32 * x1^2 + 0.002 * x1 * y1 - 0.995 * y1^2 - 0.09 * x1 - 0.29 * y1 - 1.005$$
$$g2 = -1.15 * x2^2 + 0.54 * x2 * y2 - 0.57 * y2^2 + 2.33 * x2 + 1.08 * y2 - 2.47$$

بنابراین مرز تصمیم به صورت زیر میشود.

0.83 x^2 - 0.538 x y - 2.42 x - 0.425 y^2 - 1.37 y + 1.468 = 0



شکل (۸-۱) مرز تصمیم برای حالت (p1= 1/3 , p2 = 2/3)

در این حالت نیز 1/9 خطا در کلاس 2 و 20 درصد خطا در کلاس 1 داشته و در مجموع 19 نقطه 3/19 خطا داریم.

۴-۱- سوال 4 : متغیر تصادفی پواسن

با توجه به اینکه داده های ما متغیر تصادفی پواسن با پارامتر λ میباشد.

تابع احتمال برای مجموعه ای از n متغیر تصادفی پواسون مستقل و با توزیع یکسان (i.d.) به صورت زیر است:

$$L(\lambda|D) = P(X1 = x1\ , \dots , Xn = xn\ ) = \prod \frac{e^{-\lambda}\ \lambda^{xi}}{xi\ !} \quad ; i = 1,2,..n$$

از طرفین Ln میگیریم :

$$\ln(\ L(\lambda|D))\ = P(X1 = x1\ , \dots , Xn = xn\ ) = \sum Ln \frac{e^{-\lambda}\ \lambda^{xi}}{xi\ !}$$

$$\ln(\ L(\lambda|D)) = n\lambda + \ \lambda \sum xi\ - \sum \ln(xi\ !)$$

برای به دست آوردن تخمینگر بیشینه درستنمایی (MLE) پارامتر λ، تابع log-lihood را نسبت بهλ مشتق گرفته و برابر صفر قرار میدهیم.

$$\lambda = \frac{\sum xi}{n} \quad ; i = 1\ , 2, \dots , n$$

بنابراین، MLE با پارامتر λ صرفاً میانگین نمونه داده های مشاهده شده است.

در بخش ب با توجه به اینکه توزیع قبلی L یک توزیع گاما است، می توانیم بنویسیم:

$$p(\lambda) = Gamma(\lambda|\alpha, \beta) = c\ \lambda^{\alpha-1} e^{-\beta\lambda}$$

$$p(\lambda|D) = \frac{\big(p(D|\lambda)\ p(\lambda)\big)}{p(D)}$$

$$\big(p(D|\lambda)\ p(\lambda)\big) = c\ \ \lambda^{\Sigma xi\ +\ \alpha-1} e^{-\lambda(n+\beta)}$$

$$p(D) = \int p(D|\lambda) p(\lambda) d\lambda$$

از آنجایی که انتگرال به صورت تحلیلی قابل حل نیست، می توانیم از روش های عددی برای تقریب آن استفاده

کرد ولی چون یک ضریب است در نهایت از آن صرف نظر میکنیم مهم این است که posterior هم توزیع گاما با پارامتر های جدید مشخص شده دارد. ($c'\ \lambda^{\Sigma xi + \alpha - 1} e^{-\lambda(n+\beta)}$) و بنابراین توزیع گاما برای پواسن یک conjucate prior میباشد. (c' یک ثابت نرمال کننده توزیع میباشد)

MAP (Maximum A Posteriori) برای λ مقدار L است که توزیع prosterior را با توجه به داده های مشاهده شده به حداکثر می رساند.

از سؤال قبلی، ما به دست آوردیم که توزیع peosterir با توجه به احتمال پواسون و یک گاما قبل، این است:

$$p(\lambda|D) = \ c'\ \lambda^{\Sigma xi + \alpha - 1} e^{-\lambda(n+\beta)}$$

$$\lambda\_MAP = \text{argmax}(\lambda)\ p(\lambda|D)$$

با گرفتن لگاریتم توزیع و مشتق نسبت به λ ، داریم:

$$d/d\lambda\ \ln(p(\lambda|D)) = (\Sigma xi + \alpha - 1) / \lambda - (n + \beta) = 0$$

با حل λ به دست می آوریم:

$$\lambda\_MAP = (\Sigma xi + \alpha - 1) / (n + \beta)$$

البته راه درست تر طبق راهنمایی میتواند این باشد که توزیع گاما را بنویسیم و با توجه به اینکه MLP یک نقطه میدهد طبق نکته راهنمایی گفته شده جایگذاری کنیم (چرا که نقطه ماکزیمم توزیع posterior که گاما میباشد همان نقطه MAP میباشد.)

در مورد اینکه MAP به MLE میرسد باید گفت که بله، برآوردگر MAP با نزدیک شدن تعداد نقاط داده به بی نهایت، تحت شرایط خاص، به MLE نزدیک می شود.

به طور کلی، MLE با یافتن مقدار پارامتری که تابع احتمال را بر اساس داده های مشاهده شده به حداکثر می رساند، به دست می آید. از سوی دیگر، برآوردگر MAP شامل گنجاندن دانش قبلی در مورد پارامتر در فرآیند تخمین با به حداکثر رساندن توزیع posteror پارامتر است.

هنگامی که توزیع قبلی به طور مناسب انتخاب می شود و مقدار اطلاعات قبلی خیلی قوی نیست، با افزایش مقدار داده های مشاهده شده، تأثیر توزیع قبلی بر توزیع پسین کاهش می یابد. در نتیجه، توزیع posterior بیشتر و بیشتر حول حداکثر تابع احتمال، که MLE است، متمرکز می شود. در این مورد، برآوردگر MAP با نزدیک شدن تعداد نقاط داده به بی نهایت، به MLE نزدیک می شود.

با این حال، اگر توزیع قبلی خیلی قوی باشد یا مقدار داده خیلی کم باشد، توزیع پسین ممکن است به طور قابل توجهی تحت تأثیر توزیع قبلی قرار گیرد و برآوردگر MAP ممکن است حتی با افزایش تعداد نقاط داده به MLE همگرا نشود. بنابراین، رابطه بین برآوردگر MAP و MLE به قدرت توزیع قبلی و مقدار داده های مشاهده شده بستگی دارد.

انتخاب بین برآورد MLE و MAP بستگی به در دسترس بودن و قدرت دانش قبلی در مورد پارامتر در حال برآورد دارد. به طور کلی، MLE زمانی مناسب تر است که اطلاعات قبلی در دسترس نباشد یا ضعیف باشد، در حالی که تخمین MAP زمانی مناسب تر است که درجاتی از اطلاعات قبلی در دسترس باشد.

MLE یک رویکرد فراوان گرا است که به دنبال تخمین پارامتری است که تابع احتمال را بر اساس داده های مشاهده شده به حداکثر می رساند. دانش قبلی در مورد پارامتر را در فرآیند تخمین گنجانده نیست، و بنابراین برای برآورد صرفاً به داده های مشاهده شده متکی است. MLE به ویژه در شرایطی که تعداد نمونه بزرگ است و اطلاعات قبلی در مورد پارامتر در حال تخمین وجود ندارد یا کمی وجود دارد مفید است. در چنین مواردی، MLE احتمالاً تخمین قابل اعتماد تری از مقدار پارامتر واقعی است.

از سوی دیگر، تخمین MAP شامل گنجاندن دانش قبلی در مورد پارامتر در فرآیند تخمین با به حداکثر

رساندن توزیعposterior پارامتر است. توزیع قبلی نشان دهنده دانش یا باور قبلی در مورد پارامتر است و بر اساس داده های مشاهده شده برای به دست آوردن توزیع پسین به روز می شود. تخمین MAP به ویژه در شرایطی مفید است که درجاتی از دانش قبلی در مورد پارامتر موجود است و این دانش را می توان در قالب یک توزیع قبلی بیان کرد. در چنین مواردی، تخمین MAP می تواند به تخمین های دقیق تری از پارامتر نسبت به MLE منجر شود.

به طور خلاصه، MLE زمانی ترجیح داده می شود که اطلاعات قبلی ضعیف باشد یا در دسترس نباشد، در حالی که تخمین MAP زمانی ترجیح داده می شود که اطلاعات قبلی در دسترس باشد و بتوان آن را در فرآیند تخمین گنجاند. انتخاب بین این دو روش در نهایت به ماهیت داده ها و میزان دانش قبلی موجود در مورد پارامتر در حال برآورد بستگی دارد.

۵-۱- سوال 5 : آشنایی با طبقه بند naiive bayes

درباره naiive bayes :

طبقه بندی کننده naiive بیز یک الگوریتم طبقه بندی پرکاربرد در یادگیری ماشین است که بر اساس قضیه بیز است. این یک مدل احتمالی است که برچسب کلاس یک داده ورودی داده شده را با محاسبه احتمال آن داده ورودی متعلق به هر کلاس پیش بینی می کند.

طبقه بندی کننده naiive بیز زیرمجموعه ای از طبقه بندی کننده بیزی است. طبقه‌بندی‌کننده بیزی یک مدل احتمالی است که توزیع احتمال را بر روی مجموعه‌ای از کلاس‌ها به یک داده ورودی معین اختصاص می‌دهد. طبقه بندی کننده بیزی از قضیه بیز برای محاسبه احتمال هر کلاس با توجه به داده های ورودی استفاده می کند.

فرمول قضیه بیز به صورت زیر است:

$$P(C|X) = P(X|C) * P(C) / P(X)$$

جایی که P(C|X) احتمال کلاس C داده‌های ورودی X، P(X|C) احتمال مشاهده داده‌های ورودی X با توجه به کلاس C، P(C) احتمال قبلی کلاس C و P است. (X) احتمال مشاهده داده های ورودی X است.

طبقه‌بندی‌کننده Naive Bayes به این دلیل که یک فرض قوی را ایجاد می‌کند که ویژگی‌ها یا ویژگی‌های داده‌های ورودی مستقل از یکدیگر هستند، می‌گویند که اغلب در عمل درست نیست. طبقه بندی کننده Naive Bayes احتمال هر کلاس را با توجه به داده های ورودی با ضرب احتمالات هر ویژگی با توجه به کلاس محاسبه می کند. فرمول طبقه بندی کننده Naive Bayes به شرح زیر است:

$$P(C|X) = P(X1|C) * P(X2|C) * ... * P(Xn|C) * P(C) / P(X)$$

جایی که P(Xi|C) احتمال ویژگی i با کلاس C است و P(C) احتمال قبلی کلاس C است.

طبقه بندی کننده naiive بیز الگوریتمی ساده تر و سریعتر از طبقه بندی کننده بیزی است. به منابع محاسباتی و ذخیره سازی کمتری نیاز دارد و می تواند مجموعه داده های بزرگی را با ابعاد بالا مدیریت کند. همچنین نسبت به ویژگی های نامربوط و داده های پر نویز حساسیت کمتری دارد. علاوه بر این، طبقه‌بندی‌کننده Naive Bayes کمتر مستعد پیش پردازش است، که زمانی اتفاق می‌افتد که مدل به خوبی با داده‌های آموزشی مطابقت داشته باشد و روی داده‌های دیده نشده جدید ضعیف عمل کند. طبقه بندی کننده naiive بیز همچنین به طور گسترده در کارهای طبقه بندی متن، مانند فیلتر کردن هرزنامه و تجزیه و تحلیل احساسات، که در آن داده های ورودی از ویژگی های مجزا، مانند کلمات یا اصطلاحات، و فرکانس آنها تشکیل شده است، استفاده می شود.

به عبارت دیگر طبقه بندی کننده Naive Bayes یک انتخاب مناسب برای مجموعه داده هایی است که دارای ویژگی های مجزا و تعداد زیادی نمونه هستند، مانند وظایف طبقه بندی متن. همچنین برای مجموعه‌های داده‌ای که درجه بالایی از عدم تعادل کلاس را نشان می‌دهند، مناسب است، جایی که برخی از کلاس‌ها نمونه‌های بسیار کمی دارند. با این حال، ممکن است بهترین انتخاب برای مجموعه‌های داده با ویژگی‌های پیوسته یا آنهایی

که به مرز تصمیم‌گیری پیچیده‌تر و غیرخطی بین کلاس‌ها نیاز دارند، نباشد. علاوه بر این، ذکر این نکته مهم است که طبقه‌بندی کننده Naive Bayes بر فرض استقلال ویژگی تکیه می‌کند، که ممکن است در همه موارد صادق نباشد و منجر به کاهش دقت مدل شود.

هزینه اصلی مرتبط با استفاده از طبقه‌بندی کننده naiive بیز، همانطور که چندین بار گفته شد فرض قوی استقلال ویژگی آن است. در عمل، بسیاری از ویژگی ها به یکدیگر وابسته هستند و تعاملات آنها ممکن است بر پیش بینی برچسب کلاس تأثیر بگذارد. این می تواند منجر به کاهش دقت مدل شود. علاوه بر این، طبقه‌بندی‌کننده Naive Bayes ممکن است برای مجموعه‌های داده با ویژگی‌های پیوسته مناسب نباشد، جایی که فرض استقلال ممکن است برقرار نباشد. هزینه دیگر سوگیری است که توسط احتمالات قبلی کلاس ها ارائه می شود. اگر احتمالات قبلی نماینده توزیع واقعی طبقات نباشد، مدل ممکن است ضعیف عمل کند. در نهایت، طبقه‌بندی Naive Bayes ممکن است برای کارهایی که مرز تصمیم‌گیری بین کلاس‌ها پیچیده و غیرخطی است، مناسب نباشد.

در اینجا یک دیتاست مربوط به پنگوئن ها داده شده است که ابتدا پیش‌پردازش شده و سپس به کمک الگوریتم Naiive bayes عمل طبقع بندی را بر روی آن انجام میدهیم.

پیش پردازش داده ها

ابتدا به کمک کتابخانه پانداز داده ها را لود کرده و بخشی از داده ها را به کمک دستور head. مشاهده میکنیم.

	species	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	39.1	18.7	181	3750
1	Adelie	39.5	17.4	186	3800
2	Adelie	40.3	18	195	3250
3	Adelie	x	x	x	x
4	Adelie	36.7	19.3	193	3450

شکل (۹-۱) نمایی از داده های penguins.csv

بر اساس مجموعه داده مشاهده شده ، به نظر می رسد برخی از پنگوئن ها دارای مقادیر گم شده ای هستند که به عنوان "x" نشان داده شده است. ما باید این مقادیر از دست رفته را با NaN جایگزین کرده و سپس ردیف های مربوطه را از مجموعه داده حذف کنیم.

```
df = df.replace('x', pd.np.nan)
df = df.dropna()
```

شکل (۱۰-۱) کد پایتون جایگزینی x ها با NAN و حذف آن از دیتافریم

	species	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	39.1	18.7	181	3750
1	Adelie	39.5	17.4	186	3800
2	Adelie	40.3	18	195	3250
4	Adelie	36.7	19.3	193	3450
5	Adelie	39.3	20.6	190	3650

شکل (۱۱-۱) نمایی از دیتافریم پس از حذف سطر های حاوی ’X‘

بخش species که لیبل های ما میباشند ، قبل از اینکه بتوانیم از آن برای طبقه‌بندی استفاده کنیم، باید کدگذاری شود. ما می توانیم از LabelEncoder از کتابخانه sklearn برای رمزگذاری گونه ها به مقادیر عددی استفاده کنیم.

در نهایت نیز ما باید مجموعه داده را به دو بخش تقسیم کنیم: ماتریس ویژگی X و بردار هدف y. ماتریس ویژگی شامل ویژگی هایی است که ما برای طبقه بندی استفاده خواهیم کرد، در حالی که بردار هدف حاوی برچسب های مربوطه است (در این مورد، مقادیر گونه های کدگذاری شده).

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
df['species'] = encoder.fit_transform(df['species'])
```

```
X = df.drop('species', axis=1)
y = df['species']
```

شکل (۱۲-۱) کد پایتون کد گذاری داده های هدف و تقسیم داده ها به 2 بخش ویژگی و هدف

در نهایت، باید مقادیر ویژگی ها را به گونه ای مقیاس بندی کنیم که میانگین و واریانس واحد صفر داشته باشند. این می تواند عملکرد طبقه بندی کننده Naive Bayes را بهبود بخشد.

```
from sklearn.preprocessing import StandardScaler

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

شکل (۱۳-۱) کد پایتون نرمال سازی داده های X

در بخش پایانی پیش پردازش نیز لازم است داده ها را به 2 دسته آموزش و تست تقسیم کنیم که در اینجا 70 درصد داده ها را به صورت رندوم به داده های آموزشی نسبت میدهیم.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

شکل (۱۴-۱) کد پایتون تقسیم داده ها به تست و آموزش

طراحی طبقه بند بدون استفاده از کتابخانه های آماده :

در ایتدا ، باید احتمالات prior را برای هر کلاس محاسبه کنیم، که به سادگی نسبت هر کلاس در مجموعه آموزشی است.

```
class_priors = np.zeros(3)
for i in range(3):
    class_priors[i] = np.sum(y_train == i) / len(y_train)
```

شکل (۱۵-۱) محاسبه احتمال prior

همچنین لازم است برای هر کلاس، میانگین و واریانس هر ویژگی در مجموعه آموزشی را محاسبه کنیم.

```
class_means = np.zeros((3, X_train.shape[1]))
class_variances = np.zeros((3, X_train.shape[1]))

for i in range(3):
    X_class = X_train[y_train == i]
    class_means[i] = np.mean(X_class, axis=0)
    class_variances[i] = np.var(X_class, axis=0)
```

شکل (۱۶-۱) محاسبه میانگین و واریانس

اکنون می‌توانیم تابعی را تعریف کنیم که بردار ویژگی را می‌گیرد و کلاس پیش‌بینی‌شده را بر اساس الگوریتم Naive Bayes برمی‌گرداند.

```
def naive_bayes(x):
    log_prob = np.zeros(3)
    for i in range(3):
        log_prob[i] = np.sum(-0.5 * np.log(2*np.pi*class_variances[i]) - 0.5 * ((x-class_means[i])**2) / class_variances[i]) + np.log(class_priors[i])
    return np.argmax(log_prob)
```

شکل (۱۷-۱) تعریف تابع naiive bayes

در ادامه می توانیم از طبقه بندی کننده برای پیش بینی مجموعه تست و ارزیابی عملکرد آن استفاده کنیم.

```
y_pred = np.zeros(len(y_test))
for i in range(len(X_test)):
    y_pred[i] = naive_bayes(X_test[i])
```

شکل (۱۸-۱) استفاده از طبقه بند جهت پیش بینی روی مجموعه تست

از کتابخانه SKlearn برای محاسبه متریک ها در نهایت استفاده میکنیم.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average=None)
recall = recall_score(y_test, y_pred, average=None)
cm = confusion_matrix(y_test, y_pred)
```

شکل (۱۹-۱) استفاده از کتاب خانه sklearn جهت محاسبه متریک ها

البته میتوانستیم اینجا نیز بدون استفاده از هیچ کتابخانه و بنابر تعریف نیز metric ها را محاسبه کنیم .

```
# Accuracy
accuracy = (y_pred == y_true).sum() / len(y_true)

# Confusion Matrix
classes = np.unique(y_true)
conf_mat = np.zeros((len(classes), len(classes)), dtype=int)
for i, true_label in enumerate(classes):
    for j, pred_label in enumerate(classes):
        conf_mat[i, j] = np.sum((y_true == true_label) & (y_pred == pred_label))

# Precision
precision = np.zeros(len(classes))
for i, label in enumerate(classes):
    precision[i] = conf_mat[i, i] / np.sum(conf_mat[:, i])

# Recall
recall = np.zeros(len(classes))
for i, label in enumerate(classes):
    recall[i] = conf_mat[i, i] / np.sum(conf_mat[i, :])
```

شکل (۲۰-۱) کد پایتون محاسبه متریک ها بنابر تعریف

در نهایت با استفاده از naiive bayes و طراحی آن بدون استفاده از کتابخانه ها ، متریک ها روی داده تست به صورت زیر میباشد.

```
accuracy:  0.9509803921568627
precision:  [0.94       0.88235294 1.        ]
recall:  [0.95918367 0.83333333 1.        ]
cm:  [[47  2  0]
 [ 3 15  0]
 [ 0  0 35]]
```

شکل (۲۱-۱) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی بدون استفاده از کتابخانه)

طراحی طبقه بند با استفاده از کتابخانه Sklearn :

برای اینکار میتوانیم از ما می توانیم از کلاس GaussianNB از Scikit-learn برای آموزش یک طبقه بندی کننده Naive Bayes در مجموعه آموزشی استفاده کنیم. (از همان داده های قبل استفاده میکنیم تا قابلیت مقایسه بهتری داشته باشیم تا اینکه مجددا به صورت رندوم 70 درصد داده جدید را برای آموزش برداریم)

```
clf = GaussianNB()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

شکل (۲۲-۱) کد پایتون استفاده از GussianNB و گرفتن خروجی ها بر روی داده های تست

در نهایت به همان طریق قبل متریک ها را مقایسه میکنیم و نتایج به صورت زیر است.

```
accuracy1:  0.9509803921568627
precision1:  [0.94       0.88235294 1.        ]
recall1:  [0.95918367 0.83333333 1.        ]
cm1:  [[47  2  0]
 [ 3 15  0]
 [ 0  0 35]]
```

شکل (۲۳-۱) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی با استفاده از کتابخانه)

دو روش، پیاده سازی Naive Bayes از ابتدا و استفاده از کتابخانه Scikit-learn، از نظر پیچیدگی و مقدار

کد مورد نیاز برای پیاده سازی آنها متفاوت است.

پیاده سازی Naive Bayes از ابتدا می تواند تمرین خوبی برای به دست آوردن درک عمیق تر از نحوه عملکرد الگوریتم باشد و بتوانید آن را مطابق با نیازهای خاص شخصی سازی کنید. با این حال، به تلاش زیادی برای کدنویسی نیاز دارد، به خصوص اگر بخواهیم آن را برای چندین کلاس یا با انواع مختلف توزیع احتمال پیاده سازی کنیم.

از سوی دیگر، استفاده از کتابخانه Scikit-learn، راه بسیار ساده‌تر و راحت‌تری را برای پیاده‌سازی Naive Bayes، به‌ویژه زمانی که با مجموعه داده‌های پیچیده سروکار داریم، ارائه می‌کند. همچنین توابع توزیع احتمال داخلی مختلفی را ارائه می‌کند که می‌توان از آنها برای جا دادن داده‌ها استفاده کرد و به راحتی می‌تواند چندین کلاس را مدیریت کند.

از نظر عملکرد، پیاده سازی Scikit-learn احتمالاً کارآمدتر از پیاده سازی از ابتدا خواهد بود، زیرا برای مجموعه داده های بزرگ بهینه شده است و از تکنیک های محاسباتی موازی بهره می برد.

در اینجا ماتریس آشفتگی در هر 2 حالت یکی شد ، یکی از دلایل این امر می تواند این باشد که پیاده سازی Scikit-learn بر اساس همان الگوریتم Naive Bayes است که در پیاده سازی از ابتدا استفاده شده است، و هر دو به درستی الگوریتم را با مفروضات و تنظیمات پارامتر یکسان اعمال می کنند. این منجر به همان دقت و ماتریس آشفتگی می شود.

دلیل دیگر می تواند این باشد که مجموعه داده نسبتاً ساده است و نیازی به ویژگی های پیشرفته یا سفارشی سازی ندارد که تنها با پیاده سازی سفارشی قابل دستیابی است. در این حالت، هر دو روش قادر به طبقه‌بندی صحیح مجموعه داده‌ها بدون نیاز به ویژگی‌های اضافی یا سفارشی‌سازی هستند.

به صورت 1vs all از طبقه بند طراحی شده استفاده کنیم:

برای اینکار از کد زیر استفاده میکنیم (تبدیل به 3 تا باینری classifier میشود)

```
# Compute performance metrics for each class
classes = np.unique(y_test)
accuracy = []
precision = []
recall = []
cm = []
for c in classes:

    y_test_c = (y_test == c)
    y_pred_c = (y_pred == c)
    # Compute metrics for the current class
    accuracy_c = accuracy_score(y_test_c, y_pred_c)
    precision_c = precision_score(y_test_c, y_pred_c)
    recall_c = recall_score(y_test_c, y_pred_c)
    cm_c = confusion_matrix(y_test_c, y_pred_c)
    # Append metrics to the lists
    accuracy.append(accuracy_c)
    precision.append(precision_c)
    recall.append(recall_c)
    cm.append(cm_c)

# Convert the lists to arrays
accuracy3 = np.array(accuracy)
precision3 = np.array(precision)
recall3 = np.array(recall)
cm3 = np.array(cm)
```

شکل (۲۴-۱) کد پایتون 1vsall

```
print("accuracy1: " , accuracy3)
print("precision1: " , precision3)
print("recall1: " , recall3)
print("cm1: " , cm3)
```

```
accuracy1:  [0.95098039 0.95098039 1.        ]
precision1:  [0.94       0.88235294 1.        ]
recall1:  [0.95918367 0.83333333 1.        ]
cm1:  [[[50  3]
  [ 2 47]]

 [[82  2]
  [ 3 15]]

 [[67  0]
  [ 0 35]]]
```

شکل (۲۵-۱) مقادیر دقت و precision و recall و ماتریس آشفتگی (طراحی با استفاده از کتابخانه و 1vsALL)

در حالت بدون استفاده از کتابخانه SK learn نیز کد و نتیجه حاصله به همین صورت میشود.

۶-۱- سوال 6 : طبقه بندی تصاویر جنگل و دریا

مرحل دادن لیبل به داده های تصویر میباشد برای اینکار از 2 لیست استفاده میکنیم تا در یکی آدرس ها و در دیگری label متناظر آن را ذخیره کنیم . با توجه به تصاویر ، تصاویری که اسم آن ها با s شروع میشود تصویر دریا بوده و تصاویری که اسم آن ها با j شروع میشود تصویر جنگل میباشد.

```
import os

parent_dir = "./image"

path = []
labels =[]

for file in os.listdir(parent_dir):
    if file.startswith("j"):
        label = 0  # Jungle
    elif file.startswith("s"):
        label = 1  # Sea
    else:
        continue
    path.append(file)
    labels.append(label)
```

شکل (۲۶-۱) کد پایتون اختصاص لیبل به تصاویر

ایده اصلی طبقه بندی :

همانطور که میدانید دریا به رنگ آبی میباشد و هیچ جنگلی آبی نمیباشد فلذا میتوانیم در مرحله پیداکردن ویژگی ها ابتدا در تصاویر کانال RGB فقط مقادیر کانال B را جمع کرده و میانگین بگیریم و آن عدد نماینده ویژگی آن باشد.

مقادیر ویژگی را در لیستی به صورت زیر قرار میدهیم.

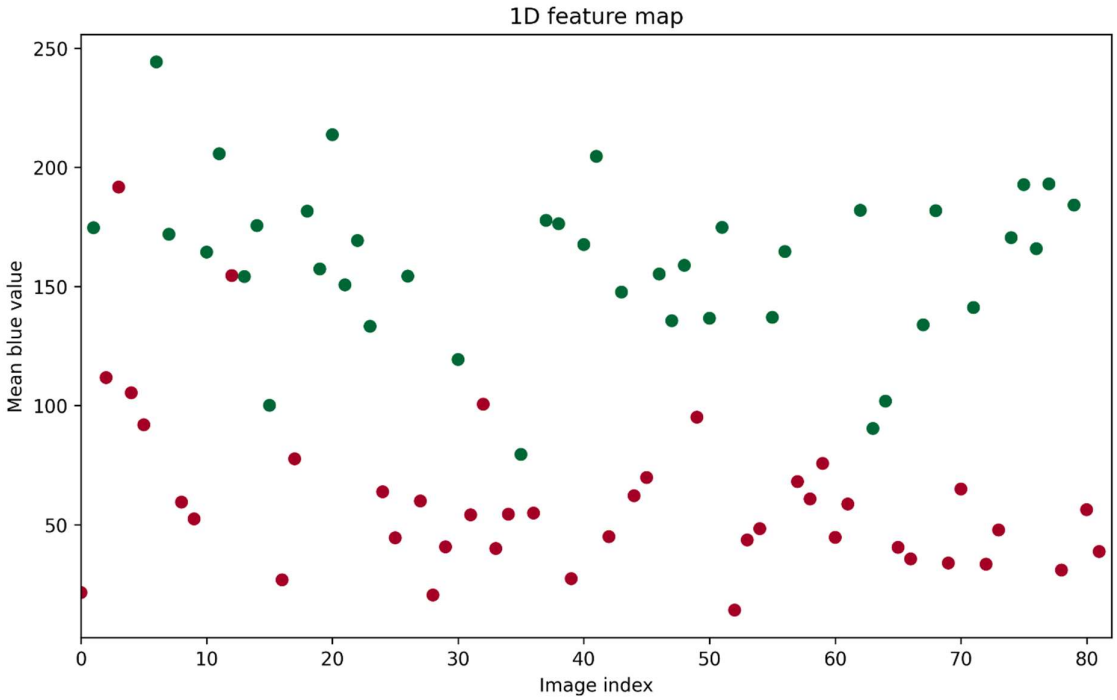
```
blue_means = []

for p in path:
    img = cv2.imread('./image/'+ p)
    blue_means.append(np.mean(img[:, :, 0])) # blue channel mean value
```

شکل (۲۷-۱) تشکیل لیست ویژگی ها با مقدار میانگین کانال آبی تصاویر دیتاست

اگر مقادیر blue_means را رسم کنیم برای 2 کلاس به صورت زیر میشود.

```
import matplotlib.pyplot as plt

# Create a figure
fig, ax = plt.subplots()

# Set the x-axis limits
ax.set_xlim(0, len(blue_means))

# Plot the data points
ax.scatter(range(len(blue_means)), blue_means, c=labels, cmap='RdYlGn')

# Set the axis labels and title
ax.set_xlabel('Image index')
ax.set_ylabel('Mean blue value')
ax.set_title('1D feature map')

# Show and save the plot
fig.set_size_inches(10, 6)
fig.savefig('figure.png', dpi=300, bbox_inches='tight')
plt.show()
```

شکل (۲۸-۱) کد پایتون رسم ویژگی ها بر حسب لیبل ها



شکل (۲۹-۱) منحنی ویژگی ها بر حسب لیبل ها

با توجه به شکل 22 مشاهده میشود مثلا خطی در حدود 110 میتواند مرزی را با دقت خوبی جدا کند ، برای پیدا کردن مقدار دقیق این threshold از طبقه بند بیز استفاده میکنیم . (پس از پیدا کردن threshold تصمیم گیری به این صورت است که اگر مقدار میانگین کانال آبی تصویر از این threshold بیشتر بود تصویر دریا است و در غیر این صورت تصویر جنگل است)

```python
# Define the Bayesian classifier function
def bayesian_classifier(values, labels, threshold):
    predictions = []
    for value in values:
        if value > threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    accuracy = accuracy_score(labels, predictions)
    cm = confusion_matrix(labels, predictions)
    precision = precision_score(labels, predictions)
    recall = recall_score(labels, predictions)
    return accuracy, cm, precision, recall ,  predictions
```

شکل (۳۰-۱) کد پایتون تعریف طبقه بند بیز

عملیات پیدا کردن بهترین threshold بدین صورت است که از 0 تا 256 که بیشترین میانگین برای رنگ آبی میباشد ، دونه دونهچک میکنیم دقت طبقه بند چه قدر است ، بالاترین دقت در optimal threshold رخ میدهد.

```
thresholds = np.linspace(0, 255, 256)
best_threshold = None
best_accuracy = 0
for threshold in thresholds:
    accuracy, _, _, _ , _ = bayesian_classifier(blue_means, labels, threshold)
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_threshold = threshold
```

شکل (۳۱-۱) کد پایتون پیدا کردن بهترین threshold

پس از بدست آوردن بهترین ترشولد که در اینجا 112 میباشد شبکه را تست کرد و متریک های مختلف را محاسبه میکنیم.

```
accuracy, cm, precision, recall , predictions= bayesian_classifier(blue_means, labels, best_threshold)

# Print the results
print('Optimal Threshold:', best_threshold)
print('Accuracy:', accuracy)
print('Confusion Matrix:')
print(cm)
print('Precision:', precision)
print('Recall:', recall)
```

شکل (۳۲-۱) کد پایتون تست طبقه بند دریا و جنگل

```
Optimal Threshold: 112.0
Accuracy: 0.926829268292683
Confusion Matrix:
[[40  2]
 [ 4 36]]
Precision: 0.9473684210526315
Recall: 0.9
```

شکل (۳۳-۱) مقادیر بهترین ترشولد و دقت و Cm و pre و recall در طبقه بند دریا و جنگل

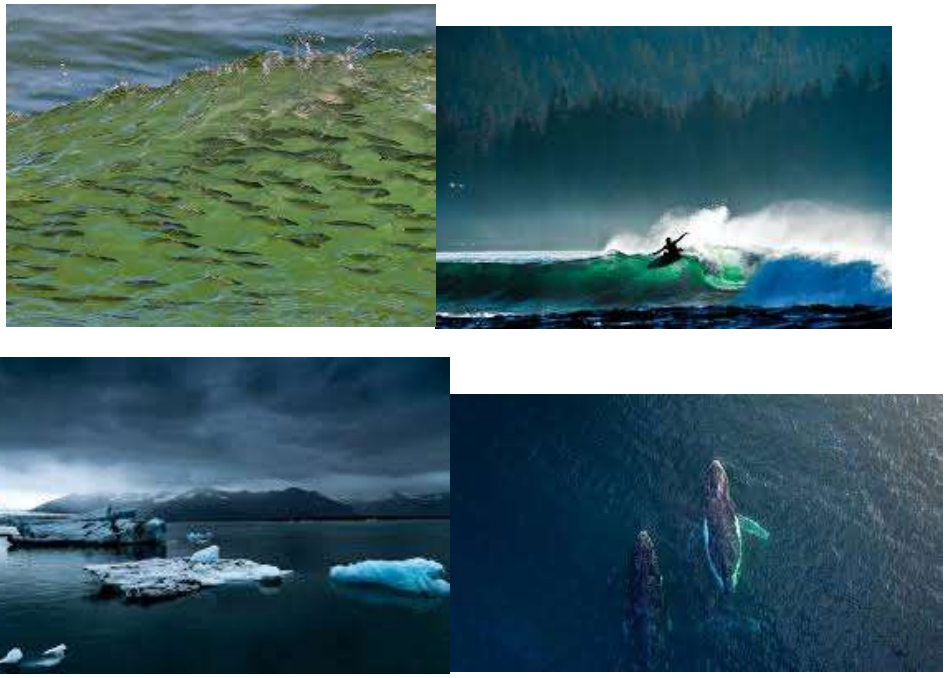
اگر در تصاویر فوق دقت شود یکی از مواردی شبکه برمیگرداند predictions ها میباشد میتوانیم miss match های آن را با labels به دست بیاوریم تا ببینیم که روی کدام تصاویر شبکه درست طبقه بندی نکرده است. (با توجه به ماتریس آشفتگی 4 تا دریا جنگل تشخیص داده شده است و 2 تا جنگل دریا تشخیص داده شده است.)

```
mismatch_indices = [i for i in range(len(labels)) if labels[i] != predictions[i]]
```

```
for i in mismatch_indices:
  print(path[i])
```

```
j45.jpg
j44.jpg
s7.jpg
s27.jpg
s24.jpg
s13.jpg
```

شکل (۳۴-۱) پیدا کردن تصاویری که شبکه عملکرد مناسبی روی آن ها نداشته است.



شکل (۳۵-۱) تصاویر دریا هایی که جنگل تشخیص داده شده است

مشاهده میشود در تصاویر فوق طیف رنگ آبی کاملا مثل سایر تصاویر نبوده و حتی در یک تصویر رنگ سبز حتی غالب تر است و به همین دلیل جدا کننده به خوبی نتوانسته طبقه بندی کند.

شکل (۳۶-۱) تصاویر جنگل هایی که دریا تشخیص داده شده است

همانطور که در اینجا مشاهده میشود نیز این تصاویر تا حدی طیف میانگین آبی است (چرا که رنگ سفید در RGB در همه کانال ها 255 میشود و این باعث بالا رفتن میانگین میشود)

و بنابراین اشتباهات رخ داده منطقی میباشد.