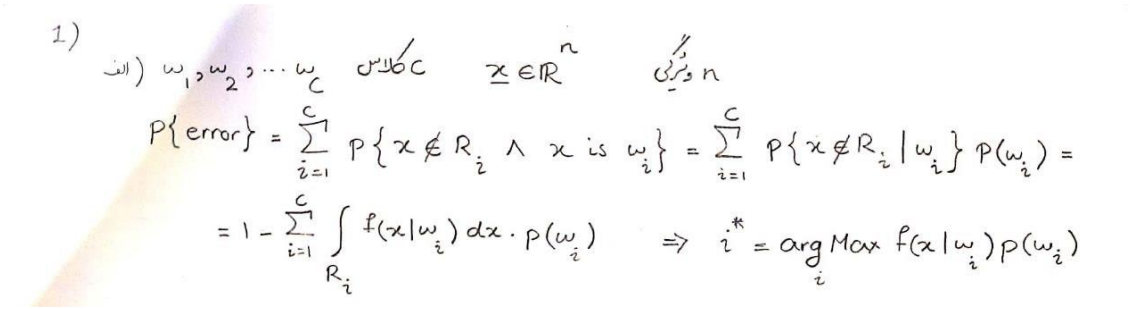
به نام خدا

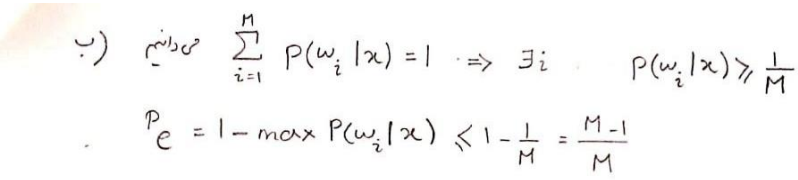
تمرین اول یادگیری ماشین

غزل زمانی‌نژاد

1. الف)

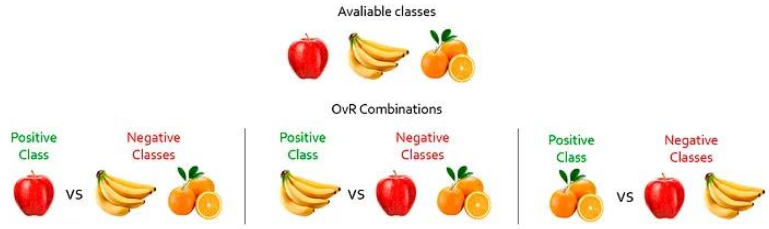


ب)

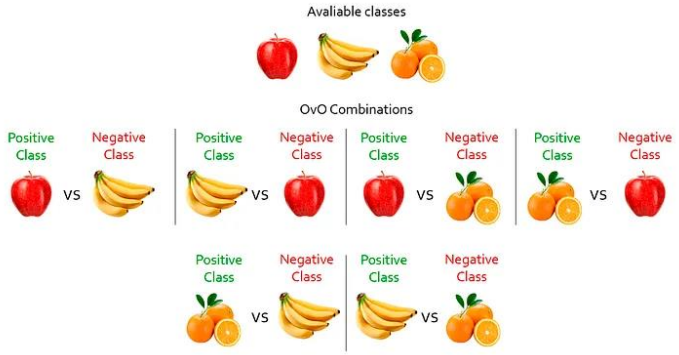


ج) به دو شیوه می‌توانیم این کار را انجام دهیم:

- One-vs-Rest: برای هر یک از کلاس‌ها، آن کلاس را به عنوان کلاس مثبت در نظر گرفته و سایر کلاس‌ها را به عنوان کلاس منفی در نظر می‌گیریم. بدین صورت مسئله به باینری کاهش می‌یابد. اگر مسئله n کلاسه باشد، n نمودار ROC رسم می‌کنیم و در آخر می‌توانیم از مقدار AUC نمودارها میانگین (ساده یا وزن‌دار) بگیریم.



- One-vs-One: در این حالت، همه کلاس‌ها را دو به دو با هم مقایسه و نمودار آن را رسم می‌کنیم. به طور مثال کلاس ۰ را به عنوان کلاس مثبت در نظر می‌گیریم و در نمودارهای مختلف کلاس منفی را ۱،۲، ...، n در نظر می‌گیریم. اگر مسئله n کلاسه باشد، n * (n-1) نمودار خواهیم داشت. در آخر می‌توانیم از مقدار AUC نمودارها میانگین (ساده یا وزن‌دار) بگیریم.



منبع: https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a

د) Naïve Bayes فرض می‌کند میان ویژگی‌های بردار، استقلال وجود دارد. به همین علت، در صورتی که ویژگی‌های دیتاست از یکدیگر مستقل باشند، طبقه‌بند عملکرد خوبی خواهد داشت. به طور مثال ویژگی‌های قد و وزن به نوعی به هم مرتبط هستند و ممکن است در صورت استفاده از این طبقه‌بند، به عملکرد بهینه نرسیم.

علاوه بر فرض بالا، موارد دیگری نیز می‌توانند بر عملکرد این طبقه‌بند تاثیر بگذارند:

- در صورتی که تعداد کافی داده داشته باشیم می‌توانیم به عملکرد بهینه برسیم و در غیر این صورت ممکن است به پاسخ sub-optimal برسیم.
- اگر دیتاست بالانس باشد و درصد کلاس‌ها حدودا با هم برابر باشد، عملکرد بهتری خواهد داشت.

- کیفیت ویژگی‌هایی که از داده جمع‌آوری کرده‌ایم تاثیر بسزایی روی عملکرد طبقه‌بند خواهد داشت.
- اگر پیش‌پردازش مناسبی بر روی داده انجام دهیم، عملکرد طبقه‌بند می‌تواند بهتر شود.
- اگر در محاسبه احتمال‌ها از laplace smoothing استفاده کنیم، عملکرد بهتر خواهد شد.

. .2

2) $p(x|y=1) = \frac{x}{\sigma^2} \exp(-\frac{x^2}{2\sigma^2})$ $x \geq 0$ $\sigma > 0$

$p(x|y=2) = \theta x \exp(-\theta x)$ $x \geq 0$ $\theta > 0$

فرض مسئله $p(y=1) = p(y=2) = 0.5$

$x \in R_1 \quad f(x|\omega_2)p(\omega_2) < f(x|\omega_1)p(\omega_1) \Rightarrow \log f(x|\omega_2) < \log f(x|\omega_1)$

$\Rightarrow \log \frac{x}{\sigma^2} + \frac{-x^2}{2\sigma^2} > \log \theta x + (-\theta x) \Rightarrow -\log \sigma^2 - \frac{x^2}{2\sigma^2} < \log\theta - \theta x$

$\underset{\times(-1)}{\Rightarrow} \log \sigma^2 + \frac{x^2}{2\sigma^2} < \theta x - \log\theta \Rightarrow \frac{1}{2\sigma^2}x^2 - \theta x + \log\theta\sigma^2 < 0$

$\Rightarrow \Delta = \theta^2 - 4(\frac{1}{2\sigma^2})\log\theta\sigma^2 = \theta^2 - \frac{2}{\sigma^2}\log\theta\sigma^2$

برای اینکه جواب داشته باشیم : $\Delta > 0 \Rightarrow \theta^2 - \frac{2}{\sigma^2}\log\theta\sigma^2 > 0$

$x = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{\theta \pm \sqrt{\theta^2 - \frac{2}{\sigma^2}\log\theta\sigma^2}}{\frac{1}{\sigma^2}}$ $\rightarrow r_1 = \sigma^2(\theta - \sqrt{\theta^2 - \frac{2}{\sigma^2}\log\theta\sigma^2})$ $\rightarrow r_2 = \sigma^2(\theta + \sqrt{\theta^2 - \frac{2}{\sigma^2}\log\theta\sigma^2})$

x		r_1		r_2	
y	+	0	−	0	+

جواب نهایی $x \in R_1 \quad r_1 < x < r_2 \wedge \Delta > 0$

$x \in R_2 \quad (0 < x < r_1 \vee x > r_2) \wedge \Delta > 0$

3. الف)

3) الف) prior: $P(\omega_1) + P(\omega_2) = 1 \rightarrow P(\omega_2) = 1 - P(\omega_1)$

$$\ell_1 = \underbrace{\lambda_{11} f(x|\omega_1) P(\omega_1)}_{0} + \underbrace{\lambda_{21}}_{1} f(x|\omega_2) P(\omega_2)$$

$$\ell_2 = \underbrace{\lambda_{22} f(x|\omega_2) P(\omega_2)}_{0} + \underbrace{\lambda_{12}}_{1} f(x|\omega_1) P(\omega_1)$$

$$\rightarrow R = \int_{R_2} p(x|\omega_1) P(\omega_1) + \int_{R_1} p(x|\omega_2) P(\omega_2) \xrightarrow{P(\omega_2) = 1 - P(\omega_1)}$$

$$\int_{R_2} p(x|\omega_1) P(\omega_1) + \int_{R_1} p(x|\omega_2)(1 - P(\omega_1))$$

$$\partial / \partial P(\omega_1) = -1$$

minimize Risk: $\partial R / \partial P(\omega_1) = 0 \rightarrow \int_{R_2} p(x|\omega_1) - \int_{R_1} p(x|\omega_2) = 0$

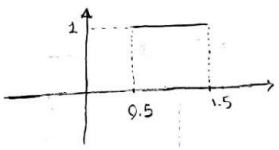
$$\Rightarrow \int_{R_2} p(x|\omega_1)\, dx = \int_{R_1} p(x|\omega_2)\, dx$$

ب)

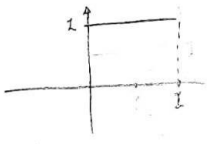
ب) خیر یکتا نیست. توابع مختلف یافت می‌شوند که در تابع صدق می‌کنند. مثال نقض:

$P(\omega_1) = P(\omega_2) = 0.5$

$$p(x|\omega_1) = \begin{cases} 1 & 0.5 \le x \le 1.5 \\ 0 & \text{otherwise} \end{cases}$$



$$p(x|\omega_2) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$



$R_1 = [0, 0.75]$ $\quad \int_{R_1} p(x|\omega_2) = 0.75 = \int_{R_2} p(x|\omega_1) = 0.75$

$R_1 = [0.5, 1]$ $\quad \int_{R_1} p(x|\omega_2) = 0.5 = \int_{R_2} p(x|\omega_1) = 0.5$

بازه های مختلفی پیدا می‌شود که در تابع صدق می‌کند

4) $\hat{\theta}_{MAP} = \underset{\theta}{\arg\max}\ f(x|\theta)\, f(\theta)$

Ln-ize: $\ln \prod_{k=1}^{N} p(x_k|\mu)\, p(\mu) = \ln p(\mu) + \sum_{k=1}^{N} \ln p(x_k|\mu) =$

$= \ln \frac{\mu}{\sigma_\mu^2} + \ln \exp\left(\frac{-\mu^2}{2\sigma_\mu^2}\right) + \sum_{k=1}^{N} \ln\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_k-\mu)^2}{2\sigma^2}\right)\right] =$

$= \ln\mu - \ln\sigma_\mu^2 - \frac{\mu^2}{2\sigma_\mu^2} + \sum_{k=1}^{N} -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x_k-\mu)^2}{2\sigma^2} =$

$= \ln\mu - \ln\sigma_\mu^2 - \frac{\mu^2}{2\sigma_\mu^2} - \frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{N}(x_k-\mu)^2$

$\xrightarrow{\partial/\partial\mu}$ ($\partial/\partial\mu = 0$ and $\partial/\partial\mu = 0$)

$$\frac{1}{\mu} - \frac{1}{2\sigma_\mu^2}\cdot 2\mu - \frac{1}{2\sigma^2}(-2)\sum_{k=1}^{N}(x_k-\mu) = \frac{1}{\mu} - \frac{\mu}{\sigma_\mu^2} + \frac{1}{\sigma^2}\sum_{k=1}^{N}(x_k-\mu) = 0$$

$$\longrightarrow \frac{1}{\mu} - \left(\frac{1}{\sigma_\mu^2} + \frac{N}{\sigma^2}\right)\mu + \frac{1}{\sigma^2}\sum x_k = 0$$

$\cdot(-\mu) \longrightarrow$

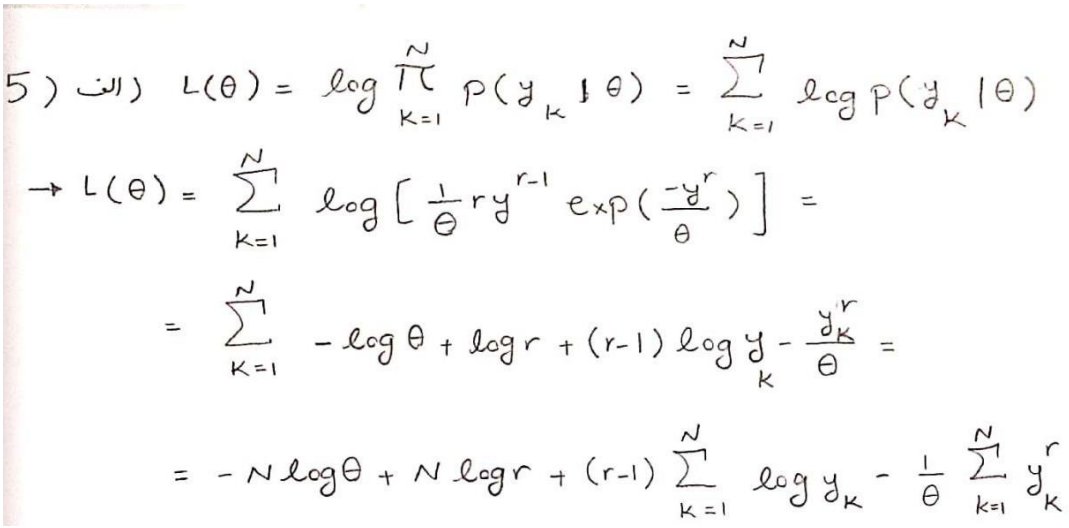
$$\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right)\mu^2 - \left(\frac{1}{\sigma^2}\sum x_k\right)\mu - 1 = 0$$

$$\Delta = b^2 - 4ac = \left[\frac{1}{\sigma^2}\sum x_k\right]^2 - 4\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right)(-1)$$

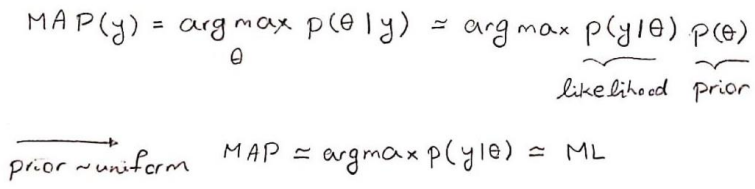
$$\mu = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{\frac{1}{\sigma^2}\sum x_k \pm \sqrt{\left[\frac{1}{\sigma^2}\sum x_k\right]^2 + 4\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right)}}{2\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}\right)} = \frac{Z \pm \sqrt{Z^2 + 4R}}{2R} \Rightarrow$$

$$\mu > 0 : \frac{Z}{2R}\left(1 + \sqrt{1 + \frac{4R}{Z^2}}\right)$$

5. الف)



ب) به طور کلی اگر تتا توزیع uniform داشته باشد (یعنی prior knowledge نداشته باشیم)، MAP به ML میل می‌کند.



6. الف) یک نوع طبقه‌بند است که اساس آن فرضیه Bayes است. همچنین فرض استقلال ویژگی‌ها در آن مطرح است. این فرض باعث راحت‌تر شدن مسئله می‌شود اما در بسیاری از مسائل دنیای واقعی صدق نمی‌کند. مهمترین تفاوت آن با طبقه‌بند Bayes همان فرض استقلال ویژگی‌هاست اما از نظر ساختاری طبقه‌بند Naive ساده‌تر و هزینه محاسباتی آن کمتر است. در مقابل طبقه‌بند Bayes، می‌تواند روابط پیچیده میان متغیرها را در نظر بگیرد و به طور کلی مدل پیچیده‌تری ارائه دهد اما هزینه محاسباتی آن بیشتر است. تصمیم به استفاده کردن از Naïve Bayes به نوع داده و همچنین مسئله بستگی دارد اما در حالت کلی به دلایل زیر از این طبقه‌بند استفاده می‌کنیم:
   - از نظر محاسباتی کارآمد است و زمانی که تعداد زیادی ویژگی داریم یا پیچیدگی وابستگی ویژگی ها دغدغه اصلی نیست، به خوبی کار می‌کند.
   - اجرای آن ساده‌تر است و می‌تواند به دقت کافی برسد.

هزینه‌هایی که بابت این طبقه‌بند می‌پردازیم عبارتند از:

- فرض عدم وابستگی ویژگی‌ها در بسیاری از موارد صحیح نیست و می‌تواند منجر به نتایج اشتباه شود.
- به دلیل در نظر نگرفتن رابطه میان ویژگی‌ها اطلاعات زیادی را از دست می‌دهیم.
- حساسیت نسبت به ویژگی‌های نامربوط
- نسبت به داده imbalance مقاوم نیست و نمی‌تواند اینطور دیتاست‌ها را به خوبی مدل کند.

در شرایطی که روابط میان ویژگی‌ها اهمیت نداشته باشد، مسئله طبقه‌بندی ساده باشد یا منابع محاسباتی محدودی داشته باشیم می‌توانیم از این طبقه‌بند بهره ببریم.

ب) ابتدا دیتاست را می‌خوانیم و ستون اضافه آن را حذف می‌کنیم.

```
1 import pandas as pd
2 import numpy as np
3 import math
```

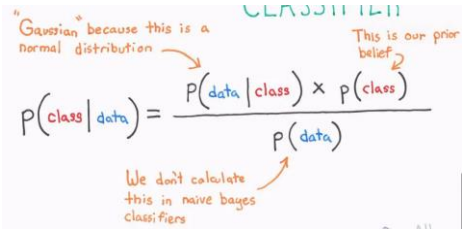
```
1 df = pd.read_csv(r"data.csv")
2 df = df.iloc[:, :-1] # drop last column Unnamed: 32
```

سپس ۰,۸ از داده را به عنوان داده آموزشی و ۰,۲ را به عنوان داده تست در نظر می‌گیریم.

```
1 train_portion = 0.8
2 train_size = int(train_portion * len(df))
3 test_size = len(df) - train_size
4
5 # shuffle data
6 df = df.sample(frac=1, random_state=42)
7 df_train = df[:train_size]
8 df_test = df[train_size:]
9 print(f"number of training data: {len(df_train)}, number of test data: {len(df_test)}")
```

```
number of training data: 455, number of test data: 114
```

بعد مطابق فرمول Naïve Bayes باید مقادیر prior و gaussian را بدست آوریم.



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

احتمال prior هر کلاس:

```
prior_m = (df_train['diagnosis'] == 'M').sum() / train_size
prior_b = (df_train['diagnosis'] == 'B').sum() / train_size
prior = {'M': prior_m, 'B': prior_b}

print(f"malignant prior={prior_m}, benign prior={prior_b}")
```

```
malignant prior=0.3626373626373626, benign prior=0.6373626373626373
```

سپس میانگین و واریانس هر ویژگی را بر اساس کلاس آن محاسبه می‌کنیم تا بتوانیم توزیع گاوسی آن را تشکیل دهیم.

```
def mean(values):
    return values.sum() / len(values)

def var(values):
    column_mean = mean(values)
    return sum((x - column_mean) ** 2 for x in values) / (len(values))

def gaussian(mean, var, x):
    return (1 / np.sqrt(2*np.pi*var)) * np.exp(-(x-mean)**2 / (2*var))

# save the mean and variance of each column, conditioned on class
features = df.columns.tolist()[2:] # features names
mean_dict = {'M':{}, 'B': {}}
var_dict = {'M':{}, 'B': {}}

df_m = df_train.loc[df_train['diagnosis'] == 'M']
df_b = df_train.loc[df_train['diagnosis'] == 'B']

for f in features:
    mean_dict['M'][f] = mean(df_m[f])
    var_dict['M'][f] = var(df_m[f])

    mean_dict['B'][f] = mean(df_b[f])
    var_dict['B'][f] = var(df_b[f])
```

از آنجایی که فرض استقلال ویژگی در naïve bayes برقرار است، likelihood برابر با حاصل ضرب تک تک $p(data|class)$ ها خواهد بود. برای جلوگیری از underflow از لگاریتم این مقادیر استفاده می‌کنیم و بجای ضرب احتمال‌ها، log آنها را جمع می‌کنیم.

```
def likelihood(m, v, features, x):
    # in order to prevent underflow,
    # calculate log p
    p = 0

    for f in features:
        g = gaussian(m[f], v[f], x[f])
        p += np.log(g)
    return p
```

از تابع predict_sample برای پیش‌بینی مقدار یک داده بر اساس احتمال‌های محاسبه شده استفاده می‌کنیم. مقدار prior * likelihood را برای هر کلاس حساب کرده و argmax آن را به عنوان پیش‌بینی در نظر می‌گیریم.

تابع predict_batch برای تخمین همه داده‌های تست استفاده می‌کنیم. در آخر برچسب‌ها را به ۰ و ۱ به ترتیب برای benign و malignant تبدیل می‌کنیم (بهتر بود این کار در ابتدا انجام می‌شد).

```python
def predict_sample(x):

    prob = 0
    class_pred = 'M'
    for c in ['M', 'B']:
        # pass mean and variance conditioned on the class
        p = np.log(prior[c]) + likelihood(mean_dict[c], var_dict[c], features, x)
        if p > prob:
            prob = p
            class_pred = c

    return class_pred
```

```python
def predict_batch(df):
    preds = []
    for index in range(len(df)):
        row = df.iloc[index]
        preds.append(predict_sample(row))

    return preds
```

```python
preds = predict_batch(df_test)
preds_binary = [1 if i == "M" else 0 for i in preds]
true_labels = [1 if i == "M" else 0 for i in df_test['diagnosis']]
```

برای ارزیابی:

در یک حلقه تک تک مقادیر پیش‌بینی داده تست و برچسب اصلی آن را مقایسه می‌کنیم تا مقادیر true positive, true negative, false positive, false negative را حساب کنیم.

```python
tp = 0
tn = 0
fp = 0
fn = 0

for p, t in zip(preds_binary, true_labels):
    # correct prediction
    if p == t:
        if p:
            tp += 1
        else: tn += 1
    # wrong prediction
    else:
        if p:
            fp += 1
        else: fn += 1

```

```python
total = len(df_test)
accuracy = (tp + tn) / total
precision = tp / (tp + fp)
recall = tp / (tp + fn)
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}")
```

```
accuracy = 0.8771929824561403, precision = 0.8367346938775511, recall = 0.8723404255319149
```

```python
confusion = np.array([[tn, fp], [fn, tp]])
print(confusion)
```

```
[[59  8]
 [ 6 41]]
```

Precision معیاری است برای اینکه چند نمونه از موارد مثبت پیش‌بینی شده واقعاً مثبت بوده‌اند و recall معیاری است برای اینکه چه تعداد از نمونه‌های مثبت واقعی به درستی توسط مدل پیش‌بینی شده‌اند. از روی ماتریس confusion می‌توانیم مشاهده کنیم که از مجموع ۶۷ نمونه خوش‌خیم، ۵۹ تا به درستی پیش‌بینی شده و ۸ تا به اشتباه بدخیم پیش‌بینی شده‌اند. همچنین از مجموع ۴۷ نمونه بدخیم، ۴۱ تا به درستی پیش‌بینی شده و ۶ تا به اشتباه خوش‌خیم پیش‌بینی شده‌اند که می‌تواند اثرات مخربی به همراه داشته باشد و موجب مرگ بیمار شود.

accuracy = 0.88,

precision = 0.84,

recall = 0.87

confusion = [[59 8]

[6 41]]

پ) برای اینکه نتایج با بخش قبل قابل مقایسه باشد، توزیع داده آموزش و آزمون را مطابق بخش قبل در نظر می‌گیریم. سپس یک مدل GuassianNB بر روی داده train آموزش می‌دهیم.

```
from sklearn.model_selection import train_test_split

x_train = df_train[features]
y_train = [1 if i == "M" else 0 for i in df_train["diagnosis"]]

x_test = df_test[features]
y_test = [1 if i == "M" else 0 for i in df_test["diagnosis"]]
```

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)
```

سپس با تابع predict داده آزمون را پیش‌بینی می‌کنیم. در آخر از معیارهای ارزیابی موجود در Scikit Learn برای بررسی عملکرد مدل استفاده می‌کنیم.

```
y_pred = nb.predict(x_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

print("accuracy = ", accuracy_score(y_test, y_pred))
print()
print(classification_report(y_test, y_pred))
```

```
accuracy =  0.9210526315789473

              precision    recall  f1-score   support

           0       0.90      0.97      0.94        67
           1       0.95      0.85      0.90        47

    accuracy                           0.92       114
   macro avg       0.93      0.91      0.92       114
weighted avg       0.92      0.92      0.92       114
```

```
print(confusion_matrix(y_test, y_pred))
```

```
[[65  2]
 [ 7 40]]
```

به طور کلی تمامی معیارها نسبت به قسمت قبل بهبود یافته‌اند. دلیل احتمالی: کدهای کتابخانه به نحو بهینه‌تر و با اعمال موارد بیشتر برای رسیدن به دقت بالاتر پیاده‌سازی شده‌اند و نسبت به کدی که از scratch زده شده تفاوت‌هایی دارد. در این مثال مقدار false negative بالا رفته که مطلوب نیست.

7. برای طبقه‌بندی، بعد از بررسی نمونه‌ها دو رنگ به عنوان threshold برای دریا و جنگل در نظر می‌گیریم:

```
image_dir = './image/'

sea_threshold = (0, 128, 255)
forest_threshold = (51, 255, 51)
```

با استفاده از تابع color_features برای هر تصویر، میانگین هر کانال را حساب می‌کنیم.

```
def color_features(image):

    # Calculate the mean pixel values for each channel
    mean_r = np.mean(image[:,:,0])
    mean_g = np.mean(image[:,:,1])
    mean_b = np.mean(image[:,:,2])

    return (mean_r, mean_g, mean_b)
```

برای پیش‌بینی کلاس هر داده، فاصله اقلیدسی بردار ویژگی را با ترشهولدها می‌سنجیم. در اینجا کلاس دریا را با برچسب ۰ و جنگل را با برچسب ۱ نمایش می‌دهیم.

```
def classify(s_threshold, f_threshold, features):
    (mean_r, mean_g, mean_b) = features
    sea_difference = np.linalg.norm(np.array(s_threshold) - np.array([mean_r, mean_g, mean_b]))
    forest_difference = np.linalg.norm(np.array(forest_threshold) - np.array([mean_r, mean_g, mean_b]))

    if sea_difference < forest_difference:
        return 0
    return 1
```

در تابع predict_batch روی تمامی داده‌ها iterate می‌کنیم و با بدست آوردن بردار ویژگی و صدا کردن تابع classify، آن را دسته‌بندی می‌کنیم.

```python
def predict_batch(path):
    x = []
    labels = []  # 0 for sea, 1 for forest
    preds = []

    # Iterate through the images in the dataset
    for f in os.listdir(path):
        # load the image and convert into numpy array
        img = Image.open(path+f)
        img_np = np.asarray(img)
        # extract image features
        img_feature = color_features(img_np)
        pred = classify(sea_threshold, forest_threshold, img_feature)
        label = 0 if f[0] == 's' else 1

        x.append(img_np)
        preds.append(pred)
        labels.append(label)

    return x, labels, preds

```

با این روش توانستیم به دقت ۹۵٪ برسیم.

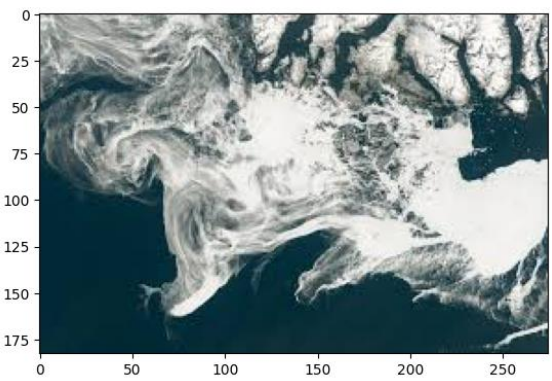
```python
x, y_true, y_pred = predict_batch(image_dir)
```

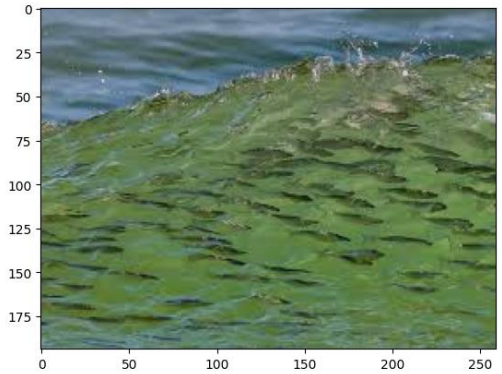
```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

print("accuracy:", accuracy_score(y_true, y_pred))
print()
print(classification_report(y_true, y_pred))
print(confusion_matrix(y_true, y_pred))
```

```
accuracy: 0.9512195121951219

              precision    recall  f1-score   support

           0       0.95      0.95      0.95        40
           1       0.95      0.95      0.95        42

    accuracy                           0.95        82
   macro avg       0.95      0.95      0.95        82
weighted avg       0.95      0.95      0.95        82

[[38  2]
 [ 2 40]]
```

در آخر مثال‌هایی که اشتباه پیش‌بینی شده‌اند را بررسی می‌کنیم:

```python
different_indices = [i for i, (x, y) in enumerate(zip(y_true, y_pred)) if x != y]
print(different_indices)
for i in different_indices:
    plt.figure()
    plt.imshow(x[i])
```






این نمونه‌ها از نظر رنگ با سایر نمونه‌ها تفاوت دارند. مثلا مورد سوم تصویری از دریاست اما سبز رنگ است و به همین دلیل به عنوان جنگل پیش‌بینی شده است. اگر تصاویر هر کلاس همگی در یک طیف رنگی بودند (دریا آبی و جنگل سبز)، دقت بالاتر می‌رفت اما با توجه به دیتاست موجود، باید بردار ویژگی را با دقت بیشتر تشکیل دهیم و نیاز به feature engineering بیشتری داریم.