دانشکده مهندسي برق و کامپیوتر

نام و نام خانوادگی :

امیر اسماعیل زاده نوبری

شماره دانشجویی

40101924

درس یادگیری ماشین

مینی پروژه دوم

استاد درس:

دکتر علیاری

بهار 1403


بسم الله الرحمن الرحیم

Contents

لینک GIT: https://github.com/CAmiren/Machine_learning4022

لینک colab :

https://drive.google.com/drive/folders/1H7JAWrO_wJeoyRBBOWCv3oltXbbxCTD6?usp=sharing

سوال 1

1.

۱. فرض کنید در یک مسألهٔ طبقه‌بندی دوکلاسه، دو لایهٔ انتهایی شبکهٔ شما فعال‌ساز ReLU و سیگموید است. چه اتفاقی می‌افتد؟

در یک مسئلهٔ طبقه‌بندی دوکلاسه، هدف نهایی این است که احتمال تعلق یک نمونه به یکی از دو کلاس را پیش‌بینی کنیم. برای این منظور، معمولاً از یک نرون خروجی با فعال‌ساز سیگموید در لایهٔ آخر استفاده می‌شود. سیگموید مقدار عددی را به یک مقدار بین ۰ و ۱ نگاشت می‌کند که می‌توان آن را به‌عنوان احتمال تفسیر کرد.

اگر دو لایهٔ انتهایی شبکهٔ شما به صورت فعال‌ساز ReLU و سیگموید تنظیم شده باشند:

1. فعال‌ساز ReLU (Rectified Linear Unit):

این فعال‌ساز هر مقدار منفی را به صفر نگاشت می‌کند و هر مقدار مثبت را به خود مقدار نگاشت می‌کند.

$$ReLU(x) = \max(0, \infty)$$

این فعال‌ساز کمک می‌کند که شبکه بتواند مقادیر منفی را به صفر تقلیل دهد و تنها مقادیر مثبت را انتقال دهد.

2. فعال‌ساز سیگموید:

این فعال‌ساز مقدار ورودی را به یک مقدار بین ۰ و ۱ نگاشت می‌کند.

$$\sigma(x) = \frac{1}{e^{-x} + 1}$$

این فعال‌ساز به دلیل این که مقدار خروجی آن بین ۰ و ۱ است، برای مسائل طبقه‌بندی دوکلاسه بسیار مناسب است زیرا می‌توان خروجی آن را به‌عنوان احتمال کلاس مثبت تفسیر کرد.

ترکیب فعال‌ساز ReLU و سیگموید در لایه‌های انتهایی :

لایهٔ قبل از لایهٔ نهایی با فعال‌ساز ReLU مقادیر ورودی منفی را به صفر تبدیل می‌کند و مقادیر ورودی مثبت را بدون تغییر انتقال می‌دهد.

لایهٔ نهایی با فعال‌ساز سیگموید این مقادیر را به یک مقدار بین ۰ و ۱ نگاشت می‌کند.

در نتیجه، مقادیر ورودی منفی که به صفر تبدیل شده‌اند، در لایهٔ سیگموید $\sigma(0) = 0.5$ خواهند داشت.

مقادیر مثبت در لایهٔ سیگموید به یک مقدار بین ۰ و ۱ نگاشت خواهند شد که بستگی به مقدار خود دارند.

نتیجه‌گیری

این ترکیب به طور کلی مشکلی ایجاد نمی‌کند و شبکه به درستی می‌تواند آموزش ببیند. با این حال، مقادیر منفی که به صفر تبدیل شده‌اند، همیشه مقدار احتمال 0.5 خواهند داشت که ممکن است تفسیر خروجی را کمی پیچیده‌تر کند. به هر حال، در عمل، چنین ترکیبی به خوبی کار می‌کند و به شبکه کمک می‌کند که بتواند طبقه‌بندی دوکلاسه را به درستی انجام دهد.

.2

۲. یک جایگزین برای ReLU در معادله ۱ آورده شده است. ضمن محاسبهٔ گرادیان آن، حداقل یک مزیت آن نسبت به ReLU را توضیح دهید.

$$\mathrm{ELU}(x) = \begin{cases} x & x >= 0 \\ \alpha\,(e^x - 1) & x < 0 \end{cases} \qquad (۱)$$

$for\ x \geq 0:$

$$\frac{d}{dx} ELU(x) = 1$$

$for\ x < 0:$

$$\frac{d}{dx} ELU(x) = \alpha e^x$$

مزایای ELU نسبت به ReLU :

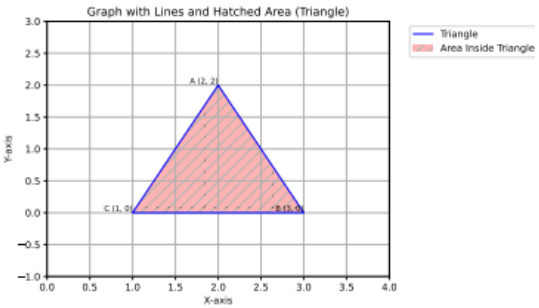
1)یکی از مزایای ELU نسبت به ReLU این است که مشکل "مرگ نرون‌ها" (Dead Neurons) را برطرف می‌کند. در ReLU، اگر ورودی به یک نرون منفی باشد، خروجی صفر می‌شود و گرادیان نیز صفر می‌شود. این مسئله می‌تواند باعث شود که نرون‌ها در طول آموزش به صفر قفل شوند و دیگر به‌روزرسانی نشوند، که به آن "مرگ نرون‌ها" گفته می‌شود.

با توجه به اینکه در ELU برای ورودی‌های منفی مقدار خروجی و گرادیان غیر صفر است، این فعال‌ساز از مرگ نرون‌ها جلوگیری می‌کند. حتی برای ورودی‌های منفی، خروجی غیر صفر و مشتق نسبتاً بزرگ است که باعث به‌روزرسانی وزن‌ها می‌شود.

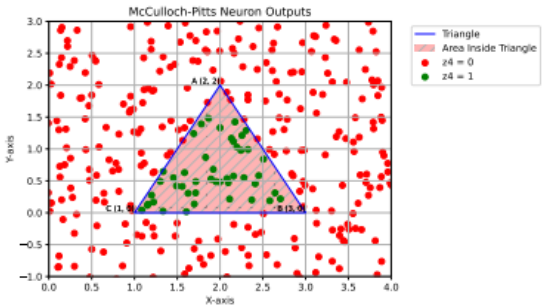
2) یکی دیگر از مزایای ELU این است که میانگین خروجی نرون‌ها نزدیک به صفر است که می‌تواند باعث شود که فرآیند آموزش سریع‌تر و پایدارتر باشد. این ویژگی به بهبود همگرایی مدل کمک می‌کند.

.3

۳. به کمک یک نورون ساده یا پرسپترون یا نورون McCulloch-Pitts[1] شبکه‌ای طراحی کنید که بتواند ناحیهٔ هاشورزدهٔ داخل مثلثی که در نمودار شکل ۱(آ) نشان داده شده را از سایر نواحی تفکیک کند. پس از انجام مرحلهٔ طراحی شبکه (که می‌تواند به‌صورت دستی انجام شود)، برنامه‌ای که در این دفترچه‌کد و در کلاس برای نورون McCulloch-Pitts آموخته‌اید را به‌گونه‌ای توسعه دهید که ۲۰۰۰ نقطهٔ رندوم تولید کند و آن‌ها را به‌عنوان ورودی به شبکهٔ طراحی‌شده توسط شما دهد و نقاطی که خروجی «۱» تولید می‌کنند را با رنگ سبز و نقاطی که خروجی «۰» تولید می‌کنند را با رنگ قرمز نشان دهد. خروجی تولیدشده توسط برنامهٔ شما باید به‌صورتی که در شکل ۱(ب) نشان داده شده است باشد (به محدودهٔ عددی محورهای x و y هم دقت کنید). اثر اضافه‌کردن دو تابع فعال‌ساز مختلف به فرآیند تصمیم‌گیری را هم بررسی کنید.



(آ) نمودار هاشورزدهٔ مورد سوال



(ب) خروجی مطلوب برنامه

شکل ۱: نمودارهای مربوط به بخش «۳» سوال اول و خروجی برنامه.

توضیح مختصر راجع به McCulloch-Pitts :

محاسبه‌گر McCulloch-Pitts، که توسط Warren McCulloch و Walter Pitts در سال ۱۹۴۳ معرفی شد، یکی از قدیمی‌ترین مدل‌های نورون مصنوعی است و پایه‌های نظریه شبکه‌های عصبی را تشکیل می‌دهد. این مدل یک نمایش ریاضی ساده از نورون‌های بیولوژیکی است و اصل اصلی پردازش اطلاعات در مغز را دربر می‌گیرد.

اجزاء یک McCulloch-Pitts-neuron:

1. ورودی‌ها (x): یک نورون مک‌کولوخ-پیتس سیگنال‌های ورودی چندگانه را دریافت می‌کند که به ترتیب به صورت $(x_1, x_2, x_3, \dots, x_n)$ نشان داده می‌شوند و هر کدام با یک وزن مرتبط است.

2. وزن‌ها (w): هر سیگنال ورودی با وزن متناظر (w_i) ضرب می‌شود. وزن‌ها نشان‌دهنده قدرت یا اهمیت سیگنال‌های ورودی برای نورون هستند.

3. آستانه (θ): نورون دارای یک مقدار آستانه (θ)است. این آستانه جمع ورودی‌های وزن‌دار را محاسبه کرده و نتیجه را با آستانه مقایسه می‌کند.

4. تابع فعال‌سازی: تابع فعال‌سازی نورون یک تابع آستانه است. اگر مجموع ورودی‌های وزن‌دار بیشتر از آستانه باشد، نورون "آتش می‌زند" یا سیگنال خروجی تولید می‌کند؛ در غیر این صورت، غیرفعال می‌ماند.

نمایش ریاضی:

با دریافت ورودی‌های $(x_1, x_2, x_3, \dots, x_n)$ و وزن‌های متناظر $(w_1, w_2, w_3, \dots, w_n)$ فعال‌سازی z یک نورون McCulloch-Pitts به شکل زیر محاسبه می‌شود:

$$\begin{cases} 1\,, & if \sum_{i=1}^{n} w_i \,.\, x_i \geq \theta \\ 0\,, & otherwise \end{cases}$$

CODE :

```python
#define muculloch pitts
class McCulloch_Pitts_neuron():

  def __init__(self , weights , threshold):
    self.weights = weights    #define weights
    self.threshold = threshold    #define threshold

  def model(self , x):
    #define model with threshold
    if self.weights @ x >= self.threshold:
        return 1
    else:
        return 0
```

```python
#define model for dataset
def Area(x , y):
  neur1 = McCulloch_Pitts_neuron([-2, -1] , -6)
  neur2 = McCulloch_Pitts_neuron([2, -1] , 2)
  neur3 = McCulloch_Pitts_neuron([0, 1] , 0)
  neur4 = McCulloch_Pitts_neuron([1 , 1 , 1],3)
  #neur4 = neur1 & neur2 & neur3

  z1 = neur1.model(np.array([x, y]))
  z2 = neur2.model(np.array([x, y]))
  z3 = neur3.model(np.array([x, y]))
  z4 = neur4.model(np.array([z1, z2, z3]))
  # 3 bit output

  return list([z4])
```

طبق شکل سوال نقاط مثلث داده شده شامل:

$$\begin{cases} A(2\,,2) \\ B(3\,,0) \\ C(1\,,0) \end{cases}$$

خطوط AB و BC و AC را طبق معادله زیر بدست می آوریم:

$$\begin{cases} m = \dfrac{y_2 - y_1}{x_2 - x_1} \\ y - y_0 = m(x - x_0) \end{cases}$$

$$\begin{cases} m_{AB} = \dfrac{2-0}{2-3} = -2 \\ m_{BC} = \dfrac{0-0}{1-3} = 0 \\ m_{AC} = \dfrac{0-2}{1-2} = 2 \end{cases} \rightarrow \begin{cases} AB: -2x - y = -6 \\ BC\ :\ y = 0 \\ AC: 2x - y = 2 \end{cases}$$

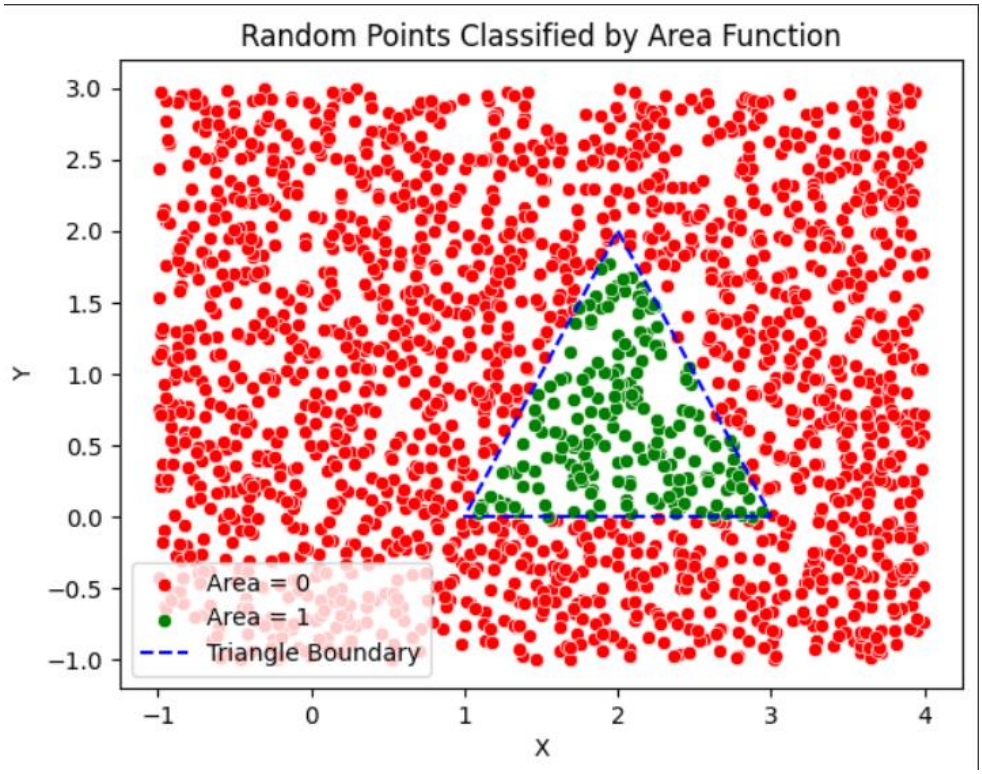
که وزن های نرون ها و آستانه را مشخص میکنند تا موقعیت نسبت به خط ها مشخص شود و نرون 4 ام مشخص می کند که نقطه ی تست شده هر سه شرط را رعایت میکند.

```python
# Generate 2000 random points in the range [-1, 4] for x and [-1, 3] for y coordinates
num_points = 2000
x_values = np.random.uniform(-1, 4, num_points)
y_values = np.random.uniform(-1, 3, num_points)
```

حال با دستور بالا 2000 نقطه با توجه به محدوده خواسته شده سوال تولید می کنیم .

```
red_points = []
green_points = []

# Evaluate data points using the Area function
for i in range(num_points):
    z4_value = Area(x_values[i], y_values[i])
    if z4_value == [0]:  # z5 value is 0
        red_points.append((x_values[i], y_values[i]))
    else:  # z5 value is 1
        green_points.append((x_values[i], y_values[i]))
# Convert lists to NumPy arrays for plotting
red_points = np.array(red_points)
green_points = np.array(green_points)
```

نقاط درون مثلث با 1 value و رنگ سبز ونقاط بیرون با 0 value و رنگ قرمز نمایش میدهیم:



اضافه کردن تابع فعال ساز به مرحله تصمیم گیری :

ما تابع فعال‌سازی ReLU و sigmoid را برای تابع فعالساز انتخاب کردیم

ReLU:

```python
def relu(x):
    return np.maximum(0, x)
```

```python
class ReluNeuron():
    def __init__(self, weights, threshold):
        self.weights = weights  # define weights
        self.threshold = threshold  # define threshold

    def model(self, x):
        # Apply the weights and calculate the linear combination
        linear_combination = np.dot(self.weights, x) - self.threshold
        # Apply the ReLU activation function
        activated_output = relu(linear_combination)
        # Determine the binary output based on the ReLU activation
        return 1 if activated_output > 0 else 0
```

```python
def Area(x, y):
    neur1 = McCulloch_Pitts_neuron([-2, -1], -6)
    neur2 = McCulloch_Pitts_neuron([2, -1], 2)
    neur3 = McCulloch_Pitts_neuron([0, 1], 0)
    neur4 = ReluNeuron([1, 1, 1], 2)  # Use ReLU neuron for z4 with adjusted threshold
```

تابع فعالساز ReLU برای ورودی‌های منفی 0 را برمی‌گرداند و برای ورودی‌های غیر منفی، خود ورودی را برمی‌گرداند.

یک کلاس به نام ReluNeuron ایجاد کردیم تا نورون نهایی با استفاده از فعال‌سازی ReLU را نشان دهد.

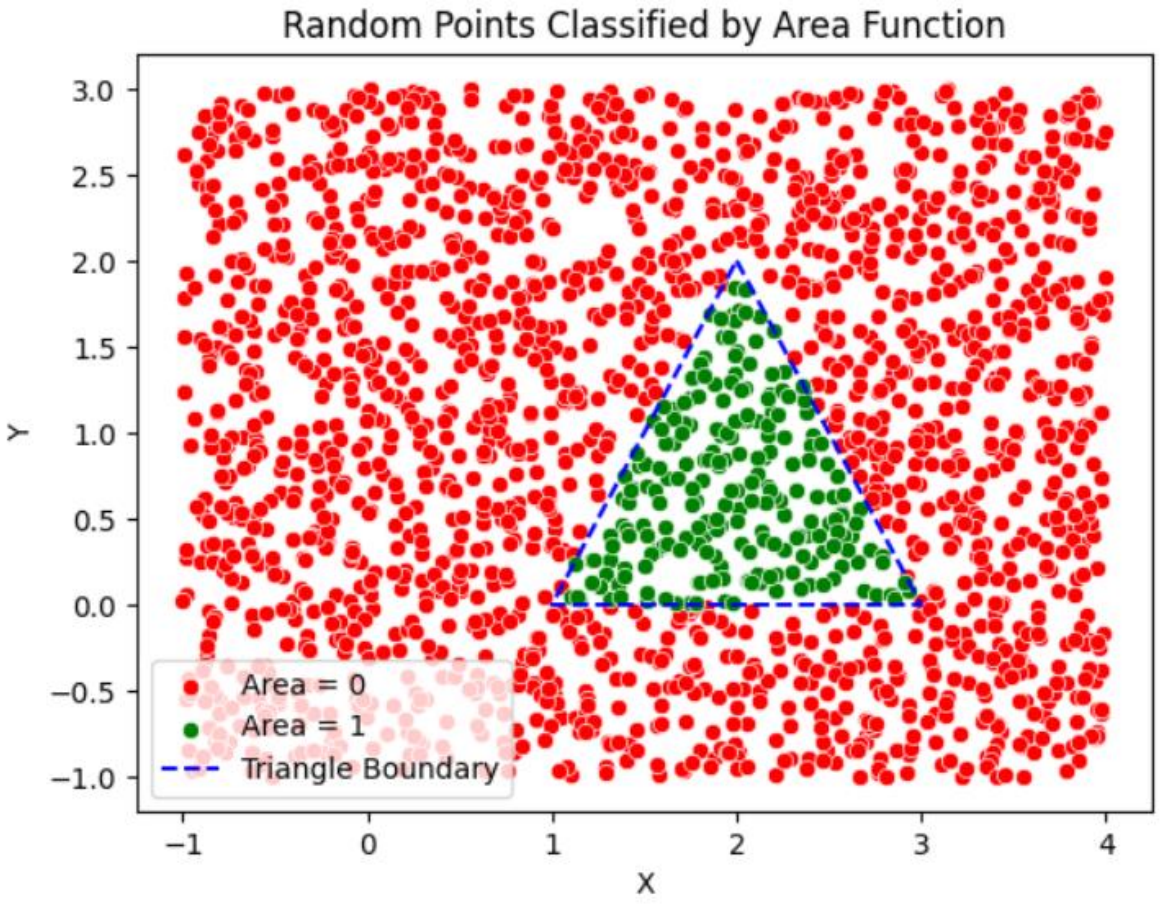
در تابع Area، پیاده‌سازی قبلی نورون نهایی را با ReluNeuron جایگزین کردیم.

تنظیماتی انجام دادیم تا اطمینان حاصل شود که طبقه‌بندی به درستی انجام می‌شود، از جمله تنظیم آستانه برای نورون نهایی به 2.

کلاس McCulloch_Pitts_neuron نورون‌هایی را نشان می‌دهد که با منطق مبتنی بر آستانه کار می‌کنند.

روش model ترکیب خطی ورودی‌ها را محاسبه کرده و آستانه را اعمال می‌کند تا خروجی دودویی تولید کند.

فعال‌سازی ReLU به صورت خارجی، در تابع Area، برای فرآیند تصمیم‌گیری نهایی اعمال می‌شود.



: Sigmoid

```python
# Define the final neuron with sigmoid activation
class SigmoidNeuron():
    def __init__(self, weights, threshold):
        self.weights = weights  # define weights
        self.threshold = threshold  # define threshold

    def model(self, x):
        # Apply the weights and calculate the linear combination
        linear_combination = np.dot(self.weights, x) - self.threshold
        # Apply the sigmoid activation function
        activated_output = sigmoid(linear_combination)
        # Determine the binary output based on the sigmoid activation
        return 1 if activated_output >= 0.5 else 0
```

```python
def Area(x, y):
    neur1 = McCulloch_Pitts_neuron([-2, -1], -6)
    neur2 = McCulloch_Pitts_neuron([2, -1], 2)
    neur3 = McCulloch_Pitts_neuron([0, 1], 0)
    neur4 = SigmoidNeuron([1, 1, 1], 3)  # Use sigmoid neuron for z4 with adjusted threshold
```

تابع فعالساز Sigmoid یک تابع غیر خطی است و مقدار خروجی آن بین 0 و 1 قرار می‌گیرد.

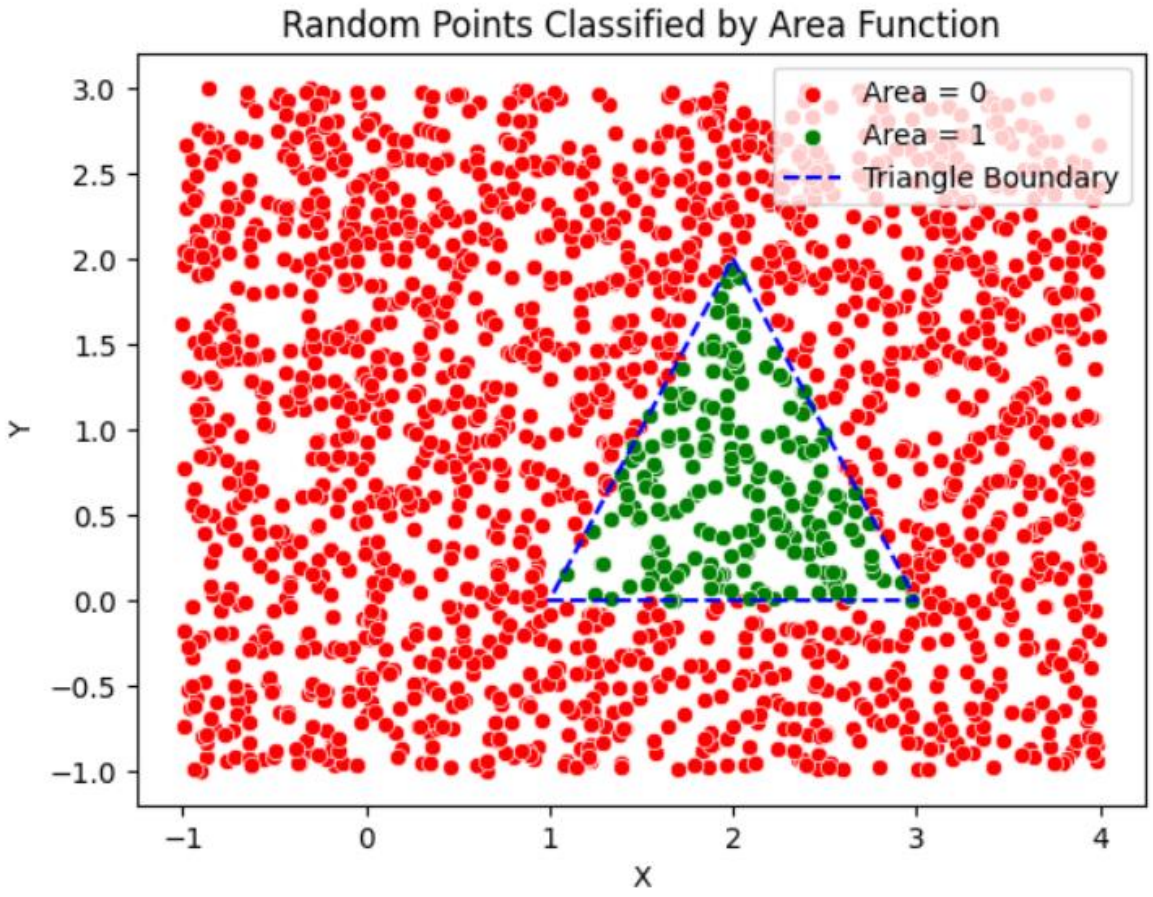
مانند قبل بجای Relu از Sigmoid استفاده می کنیم .

تنظیماتی انجام دادیم تا اطمینان حاصل شود که طبقه‌بندی به درستی انجام می‌شود، از جمله تنظیم آستانه برای نورون نهایی به 3.

کلاس McCulloch_Pitts_neuron نورون‌هایی را نشان می‌دهد که با منطق مبتنی بر آستانه کار می‌کنند.

روش model ترکیب خطی ورودی‌ها را محاسبه کرده و آستانه را اعمال می‌کند تا خروجی دودویی تولید کند.

فعال‌سازی Sigmoid به صورت خارجی، در تابع Area، برای فرآیند تصمیم‌گیری نهایی اعمال می‌شود.



سوال 2

1.

۱. دیتاست CWRU Bearing که در «مینی‌پروژهٔ شمارهٔ یک» با آن آشنا شدید را به خاطر آورید. علاوه بر دو کلاسی که در آن مینی‌پروژه در نظر گرفتید، با مراجعه به صفحهٔ داده‌های عیب در حالت 12k، دو کلاس دیگر نیز از طریق فایل‌های B007_X[۲] و OR007@6_X اضافه کنید. با انجام این کار یک کلاس دادهٔ سالم و سه کلاس از داده‌های دارای سه عیب متفاوت خواهید داشت. در مورد این که هر فایل مربوط به چه نوع عیبی است به‌صورت کوتاه توضیح دهید.

سپس در ادامه، تمام کارهایی که در بخش «۲» سوال دوم «مینی‌پروژهٔ یک» برای استخراج ویژگی و آماده‌سازی دیتا انجام داده بودید را روی دیتاست جدید خود پیاده‌سازی کنید. در قسمت تقسیم‌بندی داده‌ها، یک بخش برای «اعتبارسنجی» به بخش‌های «آموزش» و «آزمون» اضافه کنید و توضیح دهید که کاربرد این بخش چیست.

داده‌های خرابی را برای بلبرینگ سرانداز از یک مجموعه داده استخراج کرده‌اید که با نرخ نمونه‌برداری ۱۲ هزار نمونه در ثانیه جمع‌آوری شده است. داده‌های خاص خرابی که شما استخراج کرده‌اید عبارتند از:

داده خرابی محفظه بیرونی:

1) موقعیت نسبت به منطقه بارگذاری: متمرکز

منطقه بارگذاری متمرکز در ۶:۰۰: این نشان می‌دهد که خرابی در محفظه بیرونی بلبرینگ قرار دارد و در موقعیت ۶:۰۰ نسبت به منطقه بارگذاری متمرکز است. این موقعیت می‌تواند بر روی نحوه‌ای که خرابی بر داده‌های ارتعاشی تأثیر می‌گذارد، تأثیرگذار باشد.

2) داده خرابی توپ:

این نشان می‌دهد که خرابی در یکی از عناصر چرخان (توپ‌ها) بلبرینگ وجود دارد.

3) داده خرابی محفظه داخلی:

این نشان می‌دهد که خرابی در محفظه داخلی بلبرینگ وجود دارد.

درک مکان‌های خرابی:

خرابی محفظه بیرونی: خرابی‌ها در محفظه بیرونی ثابت هستند و حرکت نسبی بین محفظه بیرونی و عناصر چرخان منجر به ضربات دوره‌ای می‌شود که الگوهای ارتعاشی مشخصی را ایجاد می‌کند.

خرابی توپ: خرابی‌ها در عناصر چرخان (توپ‌ها) به ضربات منجر می‌شوند زیرا توپ‌ها بر روی محفظه‌های داخلی و بیرونی حرکت می‌کنند و الگوهای ارتعاشی خاصی تولید می‌کنند.

خرابی محفظه داخلی: خرابی‌ها در محفظه داخلی تحت تأثیر سرعت چرخشی شافت قرار دارند. هنگامی که محفظه داخلی دوران می‌کند، باعث ضربات دوره‌ای با عناصر چرخان می‌شود و الگوهای ارتعاشی منحصر به فردی تولید می‌کند.

فایل‌های داده به فرمت متلب است. هر فایل شامل داده‌های ارتعاشات سمت فن و درایو همراه با سرعت چرخش موتور است. برای تمام فایل‌ها، مورد زیر در نام متغیر نشان می‌دهد:

DE - داده‌های شتاب‌سنج سمت درایو

FE - داده‌های شتاب‌سنج سمت فن

BA - داده‌های شتاب‌سنج پایه

time - داده‌های سری زمانی

RPM - دور در دقیقه در طول آزمایش

داده‌های پایه نرمال ، داده‌های عیب بلبرینگ سمت درایو با نرخ k12 ، داده‌های عیب بلبرینگ ، سمت درایو با نرخ k48 ، داده‌های عیب بلبرینگ سمت فن

با توجه به خواسته های سوال از داده های نرمال با سرعت (1797rmp) وداده های فالت با قطرهای ۰.۰۰۷ اینچ مسیر محفظه داخلی،خارجی و توپ استفاده میکنیم .

```
[('X097_DE_time', (243938, 1), 'double'), ('X097_FE_time', (243938, 1), 'double'), ('X097RPM', (1, 1), 'double')]
[('X105_DE_time', (121265, 1), 'double'), ('X105_FE_time', (121265, 1), 'double'), ('X105_BA_time', (121265, 1), 'double'), ('X105RPM', (1, 1), 'double')]
[('X118_DE_time', (122571, 1), 'double'), ('X118_FE_time', (122571, 1), 'double'), ('X118_BA_time', (122571, 1), 'double'), ('X118RPM', (1, 1), 'double')]
[('X130_DE_time', (121991, 1), 'double'), ('X130_FE_time', (121991, 1), 'double'), ('X130_BA_time', (121991, 1), 'double'), ('X130RPM', (1, 1), 'double')]
0    0
dtype: int64
0    0
dtype: int64
0    0
dtype: int64
0    0
dtype: int64
```

در اینجا فیچر های مرتبط نوشته شده و بنا بر خواسته ی مساله ما فقط از DE_time استفاده کردیم.

تابعی تعریف می کنیم که دیتارا گرفته آن را بر بزند و نمونه با طول N جدا کند . سپس این تابع را به دیتاست اعمال می کنیم و از برای هرکلاس یک ماتریس 100*200 تشکیل میدهیم .

```
Normal.shape , Fault1.shape , Fault2.shape , Fault3.shape
```

```
((100, 200), (100, 200), (100, 200), (100, 200))
```

فیچر های ذکر شده در کتابخانه های numpy و scypy موجود هستند آنها را درون یک کلاس تعریف میکنیم. و به ماتریس های قبل اعمال می کنیم حال یک داده با 14 feature و 400 نمونه داریم .همچنین داده ها را با تعریف تابعی لیبل می زنیم:

```
Combined Features Shape: (400, 14)
Labels: (400,)
```

لیبل ها به شکل زیر در میآیند :

```
print(y)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]
```

داده ها را با استفاده از کتاب خانه sklearn به آموزش ، تست و ارزیابی تقسیم می کنیم و بر میزنیم.

و همچنین آن را استاندارد سازی میکنیم.

در قسمت تقسیم بندی دادهها، اضافه کردن ارزیابی به آموزش و تست میتواند بسیار مفید باشد. این بخش ارزیابی به عنوان مرحلهای میانی بین آموزش و تست عمل میکند و کاربردهای مختلفی دارد:

1. ارزیابی عملکرد مدل:

در مرحله آموزش، مدل با استفاده از دادههای آموزش آموزش داده میشود. سپس، برای ارزیابی عملکرد مدل و جلوگیری از بیشبرازش یا کمبرازش، میتوان از دادههای ارزیابی استفاده کرد. این دادهها به عنوان یک مجموعه میانی برای ارزیابی دقت و کارایی مدل در حالتی که با دادههای آموزش کار نکرده است، استفاده میشوند.

2. انتخاب مدل بهتر:

از دادههای ارزیابی میتوان برای انتخاب بهترین مدل از بین چندین مدل آموزش داده شده استفاده کرد. با ارزیابی عملکرد هر مدل با استفاده از دادههای اعتبارسنجی، میتوان مدلی را که بهترین عملکرد را دارد، انتخاب کرد.

3. تنظیم پارامترها:

برخی از مدل‌ها دارای پارامترهایی هستند که نیاز به تنظیم دقیق دارند. از داده‌های ارزیابی می‌توان برای تنظیم بهینه این پارامترها با استفاده از تکنیک‌هایی مانند ارزیابی متقابل (cross-validation) استفاده کرد.

4. ارزیابی مدل:

استفاده از داده‌های ارزیابی به ما اجازه می‌دهد تا مطمئن شویم که مدل آموزش داده شده قادر به عملکرد مناسب در داده‌های جدید (تست) خواهد بود. این ارزیابی اطمینان ما را از کارایی عملیاتی مدل در موارد واقعی افزایش می‌دهد.

2 .

۲. یک مدل Multi-Layer Perceptron (MLP) ساده با ۲ لایهٔ پنهان یا بیش‌تر بسازید. بخشی از داده‌های آموزش را برای اعتبارسنجی کنار بگذارید و با انتخاب بهینه‌ساز و تابع اتلاف مناسب، مدل را آموزش دهید. نمودارهای اتلاف و Accuracy مربوط به آموزش و اعتبارسنجی را رسم و نتیجه را تحلیل کنید. نتیجهٔ تست مدل روی داده‌های آزمون را با استفاده ماتریس درهم‌ریختگی و classification_report نشان داده و نتایج به‌صورت دقیق تحلیل کنید.

برای استفاده از شبکه عصبی از کتاب خانه Tensorflow.keras استفاده می کنیم .

MLP :

برای این بخش با 2 تابع فعال ساز 2 تابع اتلاف و2 بهینه ساز مختلف استفاده می کنیم

(1

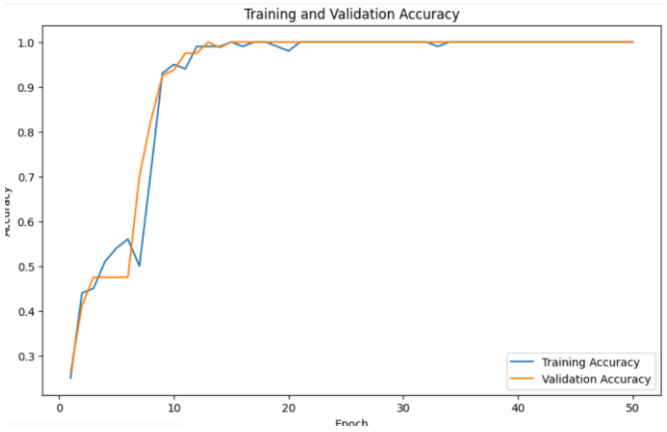
از MLP دولایه با تابع فعال ساز tanh برای دو لایه پنهان و softmax برای لایه خروجی استفاده کرده ایم . همچنین تابع اتلاف Sparse-categorical-cross-entropy که برای لیبل های integer استفاده می شود و بهینه ساز ‘adam’ استفاده کرده ایم:

```
# Define the MLP model
model = Sequential(name="MLP")
model.add(Dense(10, activation='tanh',input_shape=(X_train.shape[1],)))
model.add(Dense(10, activation='tanh'))
model.add(Dense(4,activation='softmax'))
#Summary of the model
model.summary()
```

```
#optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy' , metrics="accuracy")
history = model.fit(X_train, y_train, epochs=100, batch_size=10,
                    steps_per_epoch=10,
                    validation_data=(X_valid, y_valid),
                    )
```

```
10/10 [==============================] - 0s 6ms/step - loss: 0.1240 - accuracy: 1.0000 - val_loss: 0.1334 - val_accuracy: 1.0000
Epoch 96/100
10/10 [==============================] - 0s 6ms/step - loss: 0.1329 - accuracy: 1.0000 - val_loss: 0.1312 - val_accuracy: 1.0000
Epoch 97/100
10/10 [==============================] - 0s 7ms/step - loss: 0.1218 - accuracy: 1.0000 - val_loss: 0.1306 - val_accuracy: 1.0000
Epoch 98/100
10/10 [==============================] - 0s 7ms/step - loss: 0.1208 - accuracy: 1.0000 - val_loss: 0.1270 - val_accuracy: 1.0000
Epoch 99/100
10/10 [==============================] - 0s 7ms/step - loss: 0.1195 - accuracy: 1.0000 - val_loss: 0.1234 - val_accuracy: 1.0000
Epoch 100/100
10/10 [==============================] - 0s 6ms/step - loss: 0.1184 - accuracy: 1.0000 - val_loss: 0.1221 - val_accuracy: 1.0000
```

نمودار Loss و Accuracy :



همانطور که معلوم است دقت برای داده های ارزیابی و آموزش 100 درصد درآمده و اتلاف 0.157 و 0.159 شده است که بسیار مطلوب است .

همچنین classification_report و confusion_matrix به صورت زیر درآمده اند:

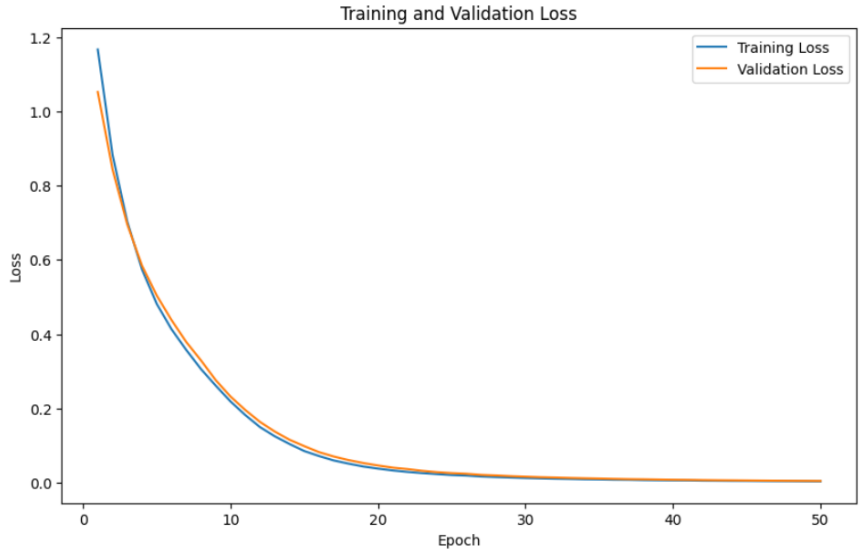
```
3/3 [==============================] - 0s 4ms/step
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        21
         1.0       1.00      1.00      1.00        23
         2.0       1.00      1.00      1.00        16
         3.0       1.00      1.00      1.00        20

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80

Confusion Matrix:
[[21  0  0  0]
 [ 0 23  0  0]
 [ 0  0 16  0]
 [ 0  0  0 20]]
```

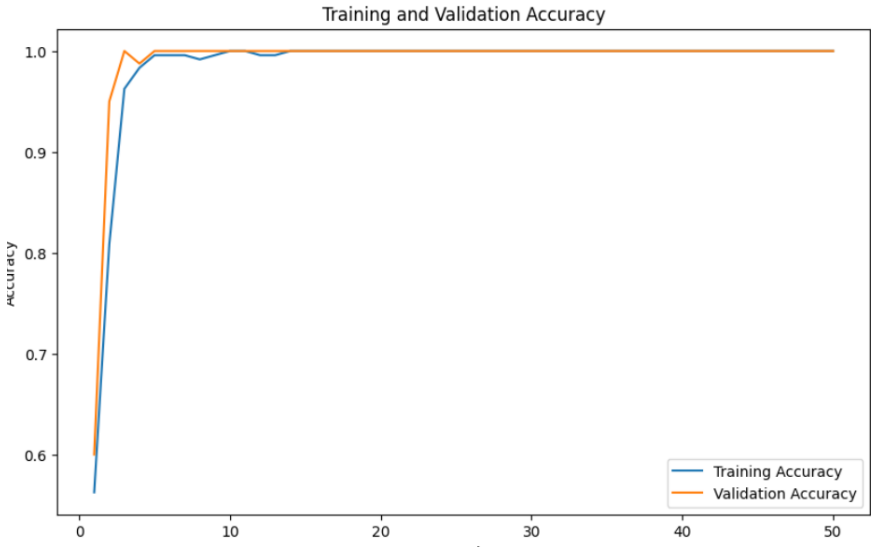
دقت با معیار های مختلف سنجیده شده که می بینیم در همه 100% شده اند و همچنین با توجه به confusion_matix میبینیم که هیچ داده ای missclassified نشده و تماما در قطر اصلی هستند.

: ReLU (2

با استفاده از 2 لایه پنهان با فعالساز ‘ReLU’ و تابع اتلاف ‘categorical_crossentropy’ برای لیبل های ‘One-hot’ شده و بهینه ساز ‘adam’ استفاده میکنیم:

```
# Convert labels to categorical one-hot encoding
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
y_valid = to_categorical(y_valid, num_classes)

# Define the MLP model
model = Sequential(name="MLP")
model.add(Dense(20, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

# Summary of the model
model.summary()

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=10, validation_data=(X_valid, y_valid))
```





.3

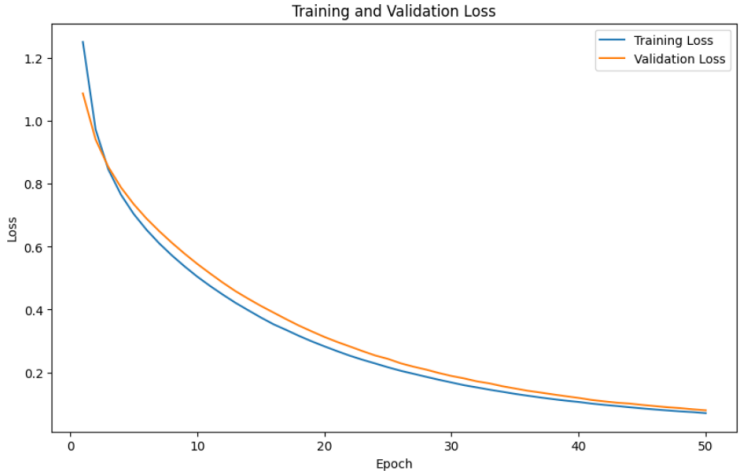
۳. فرآیند سوال قبل را با یک بهینه‌ساز و تابع اتلاف جدید انجام داده و نتایج را مقایسه و تحلیل کنید. بررسی کنید که آیا تغییر تابع اتلاف می‌تواند در نتیجه اثرگذار باشد؟

حال از تابع اتلاف ‘sparse_categorical_crossentropy’ و بهینه ساز ‘sgd’ استفاده میکنیم:

```python
# Define the MLP model
model = Sequential(name="MLP")
model.add(Dense(20, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

# Summary of the model
model.summary()

# Compile the model
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=10, validation_data=(X_valid, y_valid))
```

```
Epoch 45/50
24/24 [==============================] - 0s 8ms/step - loss: 0.0851 - accuracy: 0.9958 - val_loss: 0.0962 - val_accuracy: 0.9875
Epoch 46/50
24/24 [==============================] - 0s 6ms/step - loss: 0.0817 - accuracy: 0.9958 - val_loss: 0.0924 - val_accuracy: 0.9875
Epoch 47/50
24/24 [==============================] - 0s 8ms/step - loss: 0.0786 - accuracy: 0.9958 - val_loss: 0.0888 - val_accuracy: 0.9875
Epoch 48/50
24/24 [==============================] - 0s 6ms/step - loss: 0.0755 - accuracy: 1.0000 - val_loss: 0.0860 - val_accuracy: 0.9875
Epoch 49/50
24/24 [==============================] - 0s 8ms/step - loss: 0.0732 - accuracy: 0.9958 - val_loss: 0.0822 - val_accuracy: 0.9875
Epoch 50/50
24/24 [==============================] - 0s 8ms/step - loss: 0.0700 - accuracy: 0.9958 - val_loss: 0.0791 - val_accuracy: 0.9875
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        21
         1.0       1.00      1.00      1.00        23
         2.0       1.00      1.00      1.00        16
         3.0       1.00      1.00      1.00        20

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80

Confusion Matrix:
[[21  0  0  0]
 [ 0 23  0  0]
 [ 0  0 16  0]
 [ 0  0  0 20]]
```



انتخاب تابع اتلاف و بهینه ساز مناسب بسیار مهم می باشد .

.4

۴. در مورد K-Fold Cross-validation و Stratified K-Fold Cross-validation و مزایای هریک توضیح دهید. سپس با ذکر دلیل، یکی از این روش‌ها را انتخاب کرده و بخش «۲» سوال سوم را با آن پیاده‌سازی کنید و نتایج خود را تحلیل کنید.

: K-Fold Cross-Validation

K-Fold Cross-Validation یک روش نمونه‌گیری مجدد است که برای ارزیابی مدل‌های یادگیری ماشین بر روی مجموعه داده‌های محدود استفاده می‌شود. این روش به پارامتری به نام K وابسته است که به تعداد بخش‌هایی که مجموعه داده‌ها به آن تقسیم می‌شود اشاره دارد.

روش کار به این صورت است:

1. مجموعه داده‌ها به صورت تصادفی مرتب می‌شوند.
2. مجموعه داده‌ها به K بخش (یا fold) تقسیم می‌شوند.
3. برای هر بخش منحصر به فرد:

- آن بخش را به عنوان مجموعه داده تست انتخاب کنید.

- بقیه بخش‌ها را به عنوان مجموعه داده آموزشی انتخاب کنید.
- مدل را بر روی مجموعه آموزشی آموزش داده و بر روی مجموعه تست ارزیابی کنید.
- نمره ارزیابی را نگه داشته و مدل را کنار بگذارید.

4. میانگین نمرات ارزیابی را به عنوان معیار نهایی عملکرد مدل استفاده کنید.

:Stratified K-Fold Cross-Validation

Stratified K-Fold Cross-Validation یک نوع خاص از K-Fold Cross-Validation است که تضمین می‌کند هر بخش نمایانگر نسبت صحیحی از هر کلاس هدف باشد. این روش به ویژه در مسائل طبقه‌بندی با توزیع نامتعادل کلاس‌ها مفید است.

روش کار به این صورت است:

1. مجموعه داده‌ها به صورت تصادفی مرتب می‌شوند.
2. مجموعه داده‌ها به K بخش تقسیم می‌شوند، اما این تقسیم به صورت لایه‌ای انجام می‌شود تا هر بخش تقریباً درصدی از نمونه‌های هر کلاس هدف را داشته باشد که با مجموعه کامل برابر است.
3. برای هر بخش منحصر به فرد:
   - آن بخش را به عنوان مجموعه داده تست انتخاب کنید.
   - بقیه بخش‌ها را به عنوان مجموعه داده آموزشی انتخاب کنید.
   - مدل را بر روی مجموعه آموزشی آموزش داده و بر روی مجموعه تست ارزیابی کنید.
   - نمره ارزیابی را نگه داشته و مدل را کنار بگذارید.
4. میانگین نمرات ارزیابی را به عنوان معیار نهایی عملکرد مدل استفاده کنید.

انتخاب بین K-Fold و Stratified K-Fold Cross-Validation

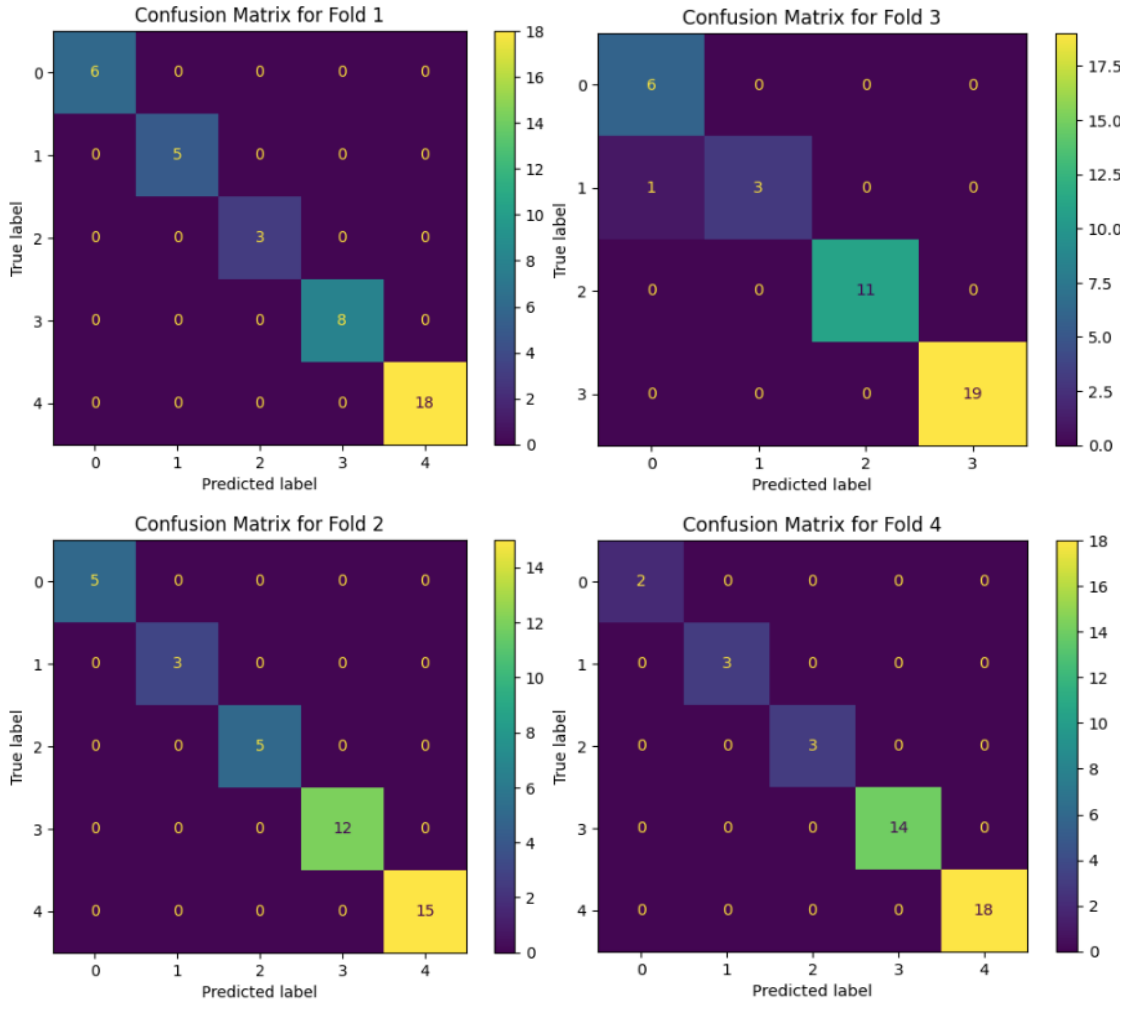
در مسائل طبقه‌بندی، به ویژه زمانی که کلاس‌ها نامتعادل هستند، بهتر است از Stratified K-Fold Cross-Validation استفاده شود. Stratified K-Fold تضمین می‌کند که هر بخش نمایانگر توزیع کلاس‌های داده باشد، که این کار ارزیابی عملکرد مدل را دقیق‌تر و پایدارتر باشد.

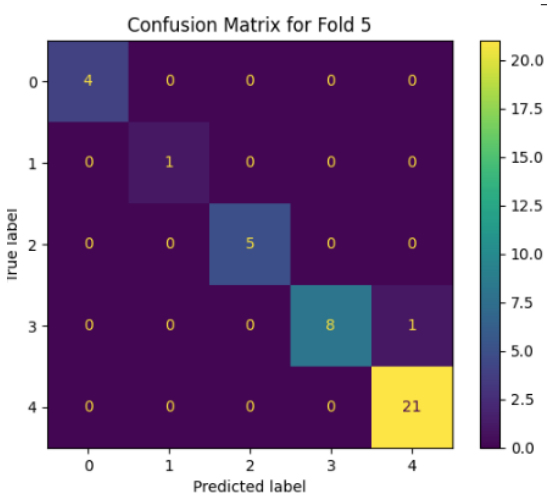
با استفاده از Stratified K-Fold Cross-Validation، اطمینان حاصل می‌شود که ارزیابی مدل شما قوی‌تر و کمتر در معرض تغییرات ناشی از عدم تعادل کلاس‌ها قرار می‌گیرد، که منجر به ارزیابی عملکرد قابل اطمینان‌تری می‌شود.

برای دیتاست سوال 3 k_fold و stratified k-fold زده شده:

k-fold

```
Fold 1 Test Accuracy: 1.0
Fold 2 Test Accuracy: 1.0
Fold 3 Test Accuracy: 0.975
Fold 4 Test Accuracy: 1.0
Fold 5 Test Accuracy: 0.975
```






```
Classification Report for Fold 1:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         8
           4       1.00      1.00      1.00        18

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

```
Fold 2 Test Accuracy: 1.0
Fold 2 Test Accuracy: 1.0
Classification Report for Fold 2:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00        12
           4       1.00      1.00      1.00        15

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

```
Classification Report for Fold 3:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92         6
           1       1.00      0.75      0.86         4
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00        19

    accuracy                           0.97        40
   macro avg       0.96      0.94      0.95        40
weighted avg       0.98      0.97      0.97        40
```

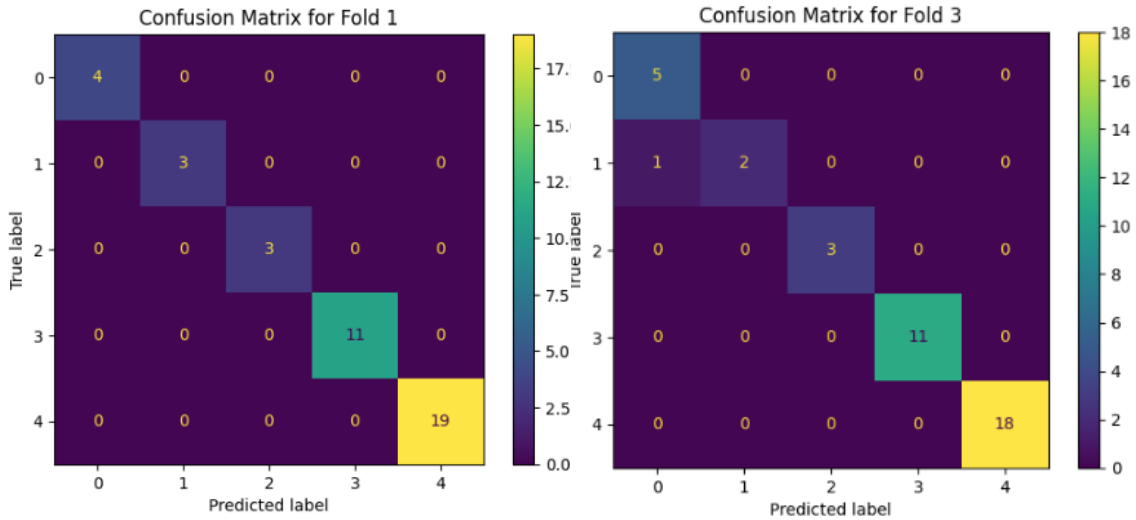
```
Fold 4 Test Accuracy: 1.0
Fold 4 Test Accuracy: 1.0
Classification Report for Fold 4:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00        14
           4       1.00      1.00      1.00        18

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

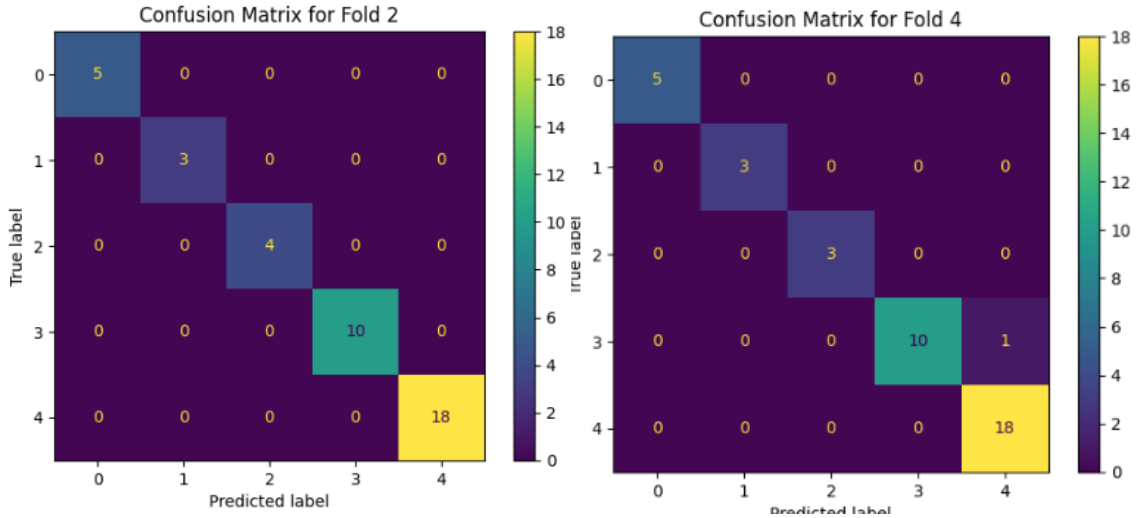
```
Fold 5 Test Accuracy: 0.975
Classification Report for Fold 5:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         5
           3       1.00      0.89      0.94         9
           4       0.95      1.00      0.98        21

    accuracy                           0.97        40
   macro avg       0.99      0.98      0.98        40
weighted avg       0.98      0.97      0.97        40
```

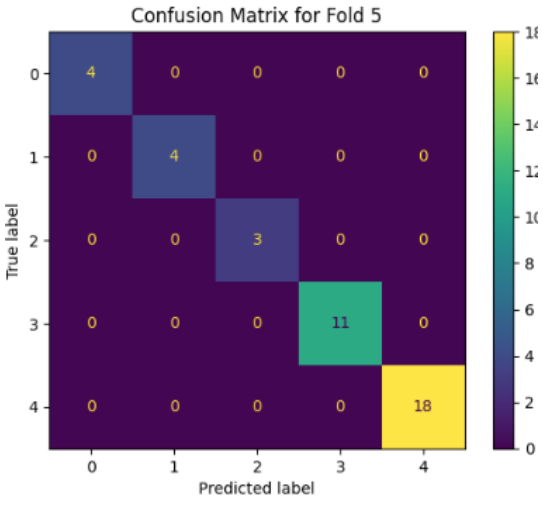
```
Average Accuracy over 5 folds: 0.99
```

در نتایج آورده شده دقت میانگین نزدیک به 100% است که بسیار مطلوب است.

: Statified K-fold







```
Fold 3 Test Accuracy: 0.975
Classification Report for Fold 3:
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.67      0.80         3
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00        18

    accuracy                           0.97        40
   macro avg       0.97      0.93      0.94        40
weighted avg       0.98      0.97      0.97        40


Fold 3 Test Accuracy: 0.975
Fold 4 Test Accuracy: 0.975
Classification Report for Fold 4:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         3
           3       1.00      0.91      0.95        11
           4       0.95      1.00      0.97        18

    accuracy                           0.97        40
   macro avg       0.99      0.98      0.99        40
weighted avg       0.98      0.97      0.97        40


Fold 4 Test Accuracy: 0.975
Fold 5 Test Accuracy: 1.0
Classification Report for Fold 5:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00        18

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

```
Fold 1 Test Accuracy: 1.0
Classification Report for Fold 1:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00        19

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40


Fold 1 Test Accuracy: 1.0
Fold 2 Test Accuracy: 1.0
Classification Report for Fold 2:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00        10
           4       1.00      1.00      1.00        18

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40


Fold 2 Test Accuracy: 1.0
```

```
Average Accuracy over 5 folds: 0.99
```

در نتایج آورده شده دقت میانگین نزدیک به 100% است که بسیار مطلوب است.

.3

در اینجا دو دیتاست طبقه بندی جنگلی و دارو معرفی شده اند

دیتاست دارو

توضیحات دیتاست

فرض کنید شما یک پژوهشگر پزشکی هستید که در حال جمع‌آوری داده‌ها برای یک مطالعه می‌باشد. شما داده‌هایی درباره‌ی مجموعه‌ای از بیماران جمع‌آوری کرده‌اید که همه‌ی آن‌ها از یک بیماری یکسان رنج می‌برند. در طول دوره درمان، هر بیمار به یکی از ۵ دارو پاسخ داده است: داروی A، داروی B، داروی C، داروی X و Y.

وظیفه‌ی ما این است که مدلی بسازیم تا مشخص کند کدام دارو ممکن است برای یک بیمار آینده‌ای که به همان بیماری مبتلا است مناسب باشد. ویژگی‌های این مجموعه داده شامل سن، جنسیت،نسبت سدیم-پتاسیم، فشار خون و کلسترول بیماران است و هدف (متغیر خروجی) دارویی است که هر بیمار به آن پاسخ داده است.

این یک نمونه از طبقه‌بندی چندکلاسه است و ما می‌توانیم از بخش آموزشی مجموعه داده برای ساخت یک مدل درخت تصمیم‌گیری استفاده کنیم و سپس از آن برای پیش‌بینی کلاس یک بیمار ناشناخته یا تجویز دارو به یک بیمار جدید استفاده کنیم.

ویژگی ها:

```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    object
 3   Cholesterol  200 non-null    object
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
None
```

داده های null نداریم و نوع دیتاهای مختلف آورده شده که مشاهده می کنیم داده سن از نوع int64 داده نسبت سدیم-پتاسیم از نوع float64 و بقیه object هستند.تعداد ارزش های unique داده های object به گونه زیر می باشد:

```
Unique counts for object type features:
Sex            2
BP             3
Cholesterol    2
Drug           5
```

داده Drug خروجی یا target ما می باشد که همانطور که میبینیم 5 value مختلف می تواند داشته باشد ، یعنی 5 کلاس داریم.

1 . پیش پردازش

۱. با استفاده از بخشی از داده‌ها، مجموعه‌داده را به دو بخش آموزش و آزمون تقسیم کنید (حداقل ۱۵ درصد از داده‌ها را برای آزمون نگه دارید). توضیح دهید که از چه روشی برای انتخاب بخشی از داده‌ها استفاده کرده‌اید. آیا روش بهتری برای این کار می‌شناسید؟

در ادامه، برنامه‌ای بنویسید که درخت تصمیمی برای طبقه‌بندی کلاس‌های این مجموعه‌داده طراحی کند. خروجی درخت تصمیم خود را با برنامه‌نویسی و یا به‌صورت دستی تحلیل کنید.

داده های از نوع متغیرهای دسته‌بندی Sex (جنسیت)، BP (فشار خون)، Cholesterol (کلسترول) و Drug (دارو) را با استفاده از LabelEncoder به مقادیر عددی تبدیل می‌کنیم.

ویژگی‌ها (Sex, BP, Cholesterol, Na_to_K, Age) را به عنوان X و برچسب‌ها (Drug) را به عنوان y تعریف می‌کند.

داده‌ها را به مجموعه‌های آموزشی (70٪) و آزمایشی (30٪) تقسیم می‌کنیم.

```
((200, 5), (200,))
```

و سپس آن هارا نرمال سازی کردیم.

```
Training set shape: (140, 5) (140,)
Testing set shape: (60, 5) (60,)
```

برای تقسیم مجموعه داده به دو بخش آموزش و آزمون، از روش‌های مختلفی می‌توان استفاده کرد. یکی از رایج‌ترین روش‌ها، استفاده از تابع train_test_split از کتابخانه sklearn است که به راحتی داده‌ها را به دو بخش تقسیم می‌کند. این تابع به طور تصادفی داده‌ها را تقسیم می‌کند، که به کاهش بایاس کمک می‌کند.در این مثال، از ۳۰٪ داده‌ها برای آزمون و ۷۰٪ برای آموزش استفاده شده است. این نسبت معمولاً به عنوان یک استاندارد خوب شناخته می‌شود، اما برای مسائل خاص، می‌توان نسبت‌های متفاوتی استفاده کرد.با استفاده از پارامتر random_state، می‌توانیم تقسیم داده‌ها را بازتولید کنیم، به طوری که در هر اجرا همان نتایج را بدست آوریم.

روش‌های جایگزین:

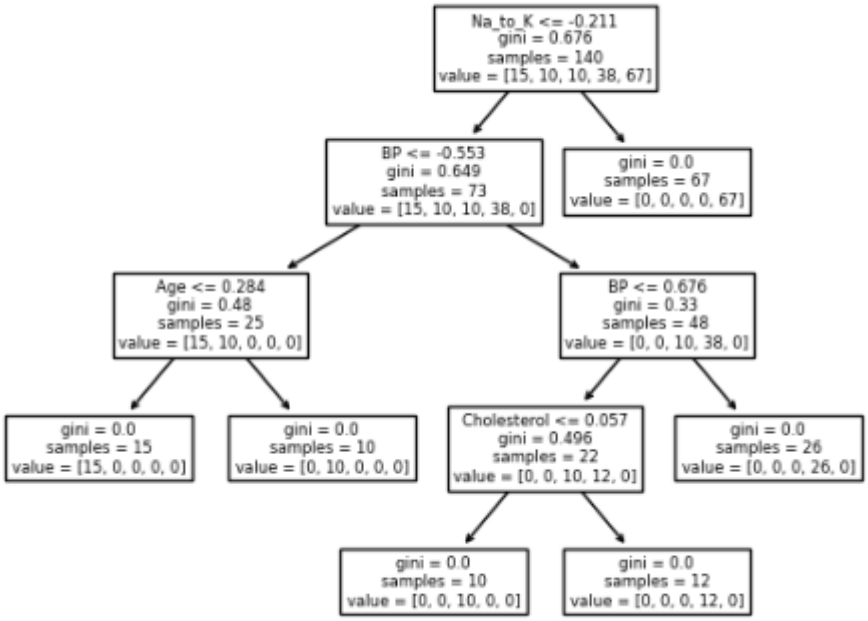
K-Fold Cross-Validation: در این روش، داده‌ها به K بخش تقسیم می‌شوند و مدل K بار آموزش داده می‌شود، هر بار یک بخش به عنوان مجموعه آزمون و بقیه به عنوان مجموعه آموزش در نظر گرفته می‌شوند. این روش دقت مدل را بهبود می‌بخشد زیرا از تمام داده‌ها برای آموزش و آزمون استفاده می‌کند.

Stratified Sampling: در صورتی که داده‌ها دارای کلاس‌های نامتوازن باشند، بهتر است از نمونه‌برداری طبقه‌ای استفاده کنیم تا نسبت کلاس‌ها در مجموعه‌های آموزش و آزمون مشابه باشد. این کار می‌تواند با استفاده از پارامتر stratify در train_test_split انجام شود.

حال با دستور زیر طبقه بند درخت تصمیم را تعریف می کنیم:

```
clf_M = DecisionTreeClassifier(random_state=24)
clf_M.fit(X_train, y_train)
clf_M.score(X_test,y_test)
```

```
1.0
```




```
[Text(0.625, 0.9, 'Na_to_K <= -0.211\ngini = 0.676\nsamples = 140\nvalue = [15, 10, 10, 38, 67]'),
Text(0.5, 0.7, 'BP <= -0.553\ngini = 0.649\nsamples = 73\nvalue = [15, 10, 10, 38, 0]'),
Text(0.25, 0.5, 'Age <= 0.284\ngini = 0.48\nsamples = 25\nvalue = [15, 10, 0, 0, 0]'),
Text(0.125, 0.3, 'gini = 0.0\nsamples = 15\nvalue = [15, 0, 0, 0, 0]'),
Text(0.375, 0.3, 'gini = 0.0\nsamples = 10\nvalue = [0, 10, 0, 0, 0]'),
Text(0.75, 0.5, 'BP <= 0.676\ngini = 0.33\nsamples = 48\nvalue = [0, 0, 10, 38, 0]'),
Text(0.625, 0.3, 'Cholesterol <= 0.057\ngini = 0.496\nsamples = 22\nvalue = [0, 0, 10, 12, 0]'),
Text(0.5, 0.1, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 10, 0, 0]'),
Text(0.75, 0.1, 'gini = 0.0\nsamples = 12\nvalue = [0, 0, 0, 12, 0]'),
Text(0.875, 0.3, 'gini = 0.0\nsamples = 26\nvalue = [0, 0, 0, 26, 0]'),
Text(0.75, 0.7, 'gini = 0.0\nsamples = 67\nvalue = [0, 0, 0, 0, 67]')]
```

```
|--- Na_to_K <= -0.21
|   |--- BP <= -0.55
|   |   |--- Age <= 0.28
|   |   |   |--- class: 0
|   |   |--- Age >  0.28
|   |   |   |--- class: 1
|   |--- BP >  -0.55
|   |   |--- BP <= 0.68
|   |   |   |--- Cholesterol <= 0.06
|   |   |   |   |--- class: 2
|   |   |   |--- Cholesterol >  0.06
|   |   |   |   |--- class: 3
|   |   |--- BP >  0.68
|   |   |   |--- class: 3
|--- Na_to_K >  -0.21
|   |--- class: 4
```

1. Root node :

```
Text(0.625, 0.9, 'Na_to_K <= -0.211\ngini = 0.676\nsamples = 140\nvalue = [15, 10, 10, 38, 67]')
```

با توجه به خروجی های آورده شده بهترین feature انتخاب شده برای گره اول (root node) ویژگی Na to K است که با شرط Na_to_K <= -0.211 به دو دسته تقسیم می شود

معیار Gini Impurity است که برای گره اول عدد 0.676 است که نشان دهنده ناخالصی بالا می باشد مطلوب است که این عدد به صفر برسانیم.

هم چنین 140نمونه در این گره وجود دارند و توزیع کلاس این گره [15, 10, 10, 38, 67] می باشد.

2. First Level Split (Left Branch) :

```
Text(0.5, 0.7, 'BP <= -0.553\ngini = 0.649\nsamples = 73\nvalue = [15, 10, 10, 38, 0]'),
```

این تقسیم مربوط به ویژگی فشار خون (bp) است که با شرط BP <= -0.553 و معیار ناخالصی Gini 0.649 و تعداد نمونه و توزیع کلاس [15, 10, 10, 38, 0] انجام شده.

3. Second Level Split (Left-Left Branch) :

```
Text(0.25, 0.5, 'Age <= 0.284\ngini = 0.48\nsamples = 25\nvalue = [15, 10, 0, 0, 0]'),
```

این تقسیم که مربوط با ویژگی سن و شرط Age <= 0.284 تقسیم داده کرده .تعداد نمونه ها در این گره 25 و میار ناخالصی Gini برابر 0.48 می باشد که به معنی میزان ناخالصی بالا می باشد و تقسیم کلاس [15, 10, 0, 0, 0] می باشد .

4. Third Level Split (Left-Left-Left Branch) :

```
Text(0.125, 0.3, 'gini = 0.0\nsamples = 15\nvalue = [15, 0, 0, 0, 0]'),
```

این تقسیم معیار ناخالصی Gini را به صفرمیرساند یعنی خلوص کلاس کامل است و وکلاس اول را مجزا میکند .تعداد داده ها برابر15 وکلاس به شکل [15, 0, 0, 0, 0] می باشد.

5. Third Level Split (Left-Left-Right Branch) :

```
Text(0.375, 0.3, 'gini = 0.0\nsamples = 10\nvalue = [0, 10, 0, 0, 0]'),
```

همچنین در این تقسیم کلاس دوم مجزا میشود تعداد نمونه ها 10 و تقسیم کلاس [0, 10, 0, 0, 0] او خلوص کلاس کامل است .

6. Second Level Split (Left-Right Branch) :

```
Text(0.75, 0.5, 'BP <= 0.676\ngini = 0.33\nsamples = 48\nvalue = [0, 0, 10, 38, 0]'),
```

در این تقسیم با شرط BP <= 0.676 و معیار Gini 0.33 تعداد48 داده با تقسیم کلاس [0, 0, 10, 38, 0] داریم .

7. Third Level Split (Left-Right-Left Branch):

```
Text(0.625, 0.3, 'Cholesterol <= 0.057\ngini = 0.496\nsamples = 22\nvalue = [0, 0, 10, 12, 0]'),
```

در این تقسیم با شرط Cholesterol <= 0.676 و معیار Gini 0.496 تعداد 22 داده با تقسیم کلاس [0, 0, 10, 12, 0]داریم .

8. Fourth Level Split (Left-Right-Left-Left Branch) :

```
Text(0.5, 0.1, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 10, 0, 0]'),
```

کلاس سوم در اینجا مجزا می شود. تعداد نمونه ها 10 وتقسیم کلاس به شکل [0, 0, 10, 0, 0] و خلوص کامل کلاس .

9. Fourth Level Split (Left-Right-Left-Right Branch) :

```
Text(0.75, 0.1, 'gini = 0.0\nsamples = 12\nvalue = [0, 0, 0, 12, 0]'),
```

همچنین تمام نمونه ها با تعداد12 به کلاس چهارم با معیار gini 0 توزیع کلاس [0, 0, 0, 12, 0] و خلوص کامل کلاس .

10. Third Level Split (Left-Right-Right Branch) :

```
Text(0.875, 0.3, 'gini = 0.0\nsamples = 26\nvalue = [0, 0, 0, 26, 0]'),
```

همچنین کلاس چهارم در اینجا کامل شده با تعداد نمونه 26 و توزیع کلاس به شکل [0, 0, 0, 26, 0] و خلوص کامل کلاس .

11. First Level Split (Right Branch) :

```
Text(0.75, 0.7, 'gini = 0.0\nsamples = 67\nvalue = [0, 0, 0, 0, 67]')]
```

در این گره کلاس پنجم مجزا شده با تعداد داده 67 و شکل توزیع کلاس [0, 0, 0, 0, 67] و خلوص کامل کلاس .

توزیع کلاس در گره‌های برگ:

کلاس 0 (داروی A): عمدتاً در گره‌هایی با شرایط Na_to_K <= -0.211، BP <= -0.553، و Age <= 0.284 مشاهده می‌شود.

کلاس 1 (داروی B): در گره‌هایی با شرایط Na_to_K <= -0.211، BP <= -0.553، و Age > 0.284 دیده می‌شود.

کلاس 2 (داروی C): در گره‌هایی با شرایط Na_to_K <= -0.211، BP > -0.553، و Cholesterol <= 0.057 مشاهده می‌شود.

کلاس 3 (داروی X): در گره‌هایی با شرایط Na_to_K <= -0.211، BP > -0.553، و Cholesterol > 0.057 مشاهده می‌شود.

کلاس 4 (داروی Y): در گره‌هایی با شرایط Na_to_K > -0.211 مشاهده می‌شود.

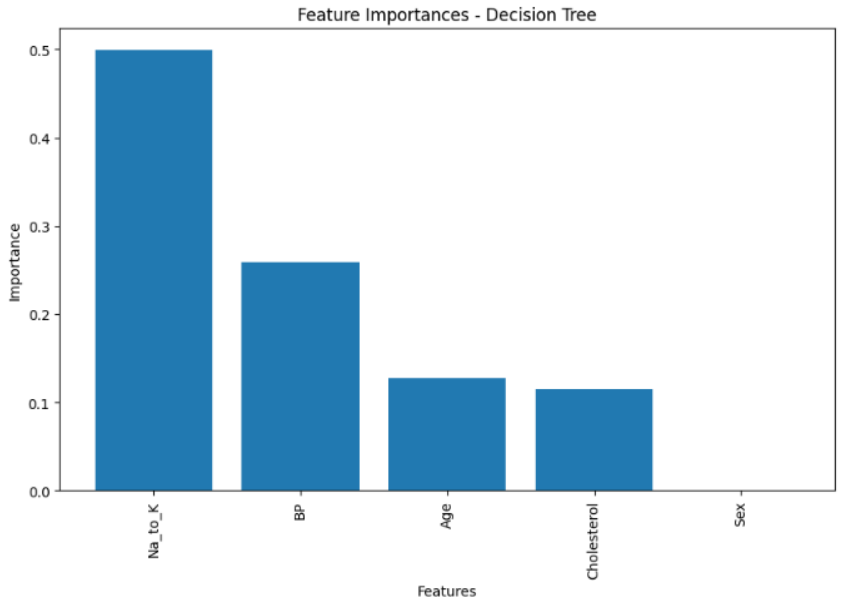
این تحلیل نشان می‌دهد که مدل چگونه پیش‌بینی‌ها را انجام می‌دهد و کدام ویژگی‌ها بیشترین تأثیر را در تعیین کلاس دارند. حضور گره‌های برگ خالص نشان می‌دهد که مدل برای برخی کلاس‌ها به خوبی عمل می‌کند، در حالی که ناخالصی جینی بالا در برخی گره‌ها نشان‌دهنده نواحی است که مدل ممکن است نیاز به بهبود بیشتر داشته باشد یا کلاس‌ها بیشتر مخلوط باشند.

```
print(clf_M.tree_.node_count)
print(clf_M.tree_.n_classes)
print(clf_M.tree_.n_features)
print(clf_M.tree_.n_leaves)
print(clf_M.get_depth())
```

```
11
[5]
5
6
4
```

تعداد گره ها 11 ، تعداد کلاس ها 5 تعداد ویژگی ها 6 تعداد برگ ها 6 و عمق (سطح) 4 است .

همچنین اهمیت ویژگی ها به ترتیب:



.2

۲. با استفاده از ماتریس درهم‌ریختگی و حداقل سه شاخصهٔ ارزیابی مربوط به وظیفهٔ طبقه‌بندی، عمل‌کرد درخت آموزش‌دادهشدهٔ خود را روی بخش آزمون داده‌ها ارزیابی کنید و نتایج را به‌صورت دقیق گزارش کنید.

تأثیر مقادیر کوچک و بزرگ حداقل دو فراپارامتر را بررسی کنید. تغییر فراپارامترهای مربوط به هرس‌کردن چه تأثیری روی نتایج دارد و مزیت آن چیست؟

```
y_pred =clf_M.predict(X_test)
print(accuracy_score(y_pred , y_test))
print(confusion_matrix(y_pred , y_test))
print(classification_report(y_pred , y_test))
```

```
1.0
[[ 8  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  6  0  0]
 [ 0  0  0 16  0]
 [ 0  0  0  0 24]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00        16
           4       1.00      1.00      1.00        24

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```

میبینیم که confusion matrix قطری شده و هیچ داده ای misclassified نشده و همچنین با classification_report می بینیم که با معیار ارزیابی f1_score و recall و precision وaccuracy دقت 100% شده.

تغییر 2 فرا پارامتر (max_depth) و (min_sample_leaf)

1. حداکثر عمق درخت (Max Depth):

الف. عمق کم (مقدار کوچک):

مزایا:

جلوگیری از بیش‌برازش (Overfitting): وقتی عمق درخت کوچک است، مدل به خوبی تعمیم پیدا می‌کند و از یادگیری بیش از حد جزئیات داده‌های آموزش جلوگیری می‌شود.

سرعت بیشتر: مدل سریع‌تر آموزش می‌بیند و زمان پیش‌بینی نیز کاهش می‌یابد.

معایب:

کمتر دقیق: ممکن است مدل نتواند الگوهای پیچیده در داده‌ها را به درستی یاد بگیرد و دقت کمتری داشته باشد.

ب. عمق زیاد (مقدار بزرگ):

مزایا:

دقت بیشتر: مدل می‌تواند الگوهای پیچیده‌تری را یاد بگیرد و دقت بالاتری در داده‌های آموزش داشته باشد.

معایب:

بیش‌برازش: مدل ممکن است بیش از حد به داده‌های آموزش حساس شود و در داده‌های جدید عملکرد خوبی نداشته باشد.

افزایش زمان محاسبه: زمان آموزش و پیش‌بینی افزایش می‌یابد.

2. حداقل نمونه‌ها در یک برگ (Min Samples Leaf):

الف. مقدار کم:

مزایا:

دقت بیشتر: مدل می‌تواند جزئیات بیشتری از داده‌ها را یاد بگیرد و دقت بالاتری داشته باشد.

معایب:

بیش‌برازش: مدل ممکن است بیش از حد به داده‌های آموزش حساس شود و عملکرد خوبی در داده‌های جدید نداشته باشد.

پیچیدگی بیشتر: ساختار درخت پیچیده‌تر می‌شود و زمان آموزش و پیش‌بینی افزایش می‌یابد.

ب. مقدار زیاد:

مزایا:

جلوگیری از بیش‌برازش: با افزایش حداقل نمونه‌ها در یک برگ، مدل تعمیم‌پذیری بهتری پیدا می‌کند.

کاهش پیچیدگی: ساختار درخت ساده‌تر می‌شود و زمان آموزش و پیش‌بینی کاهش می‌یابد.

معایب:

کمتر دقیق: مدل ممکن است نتواند الگوهای جزئی‌تر را یاد بگیرد و دقت کمتری داشته باشد.

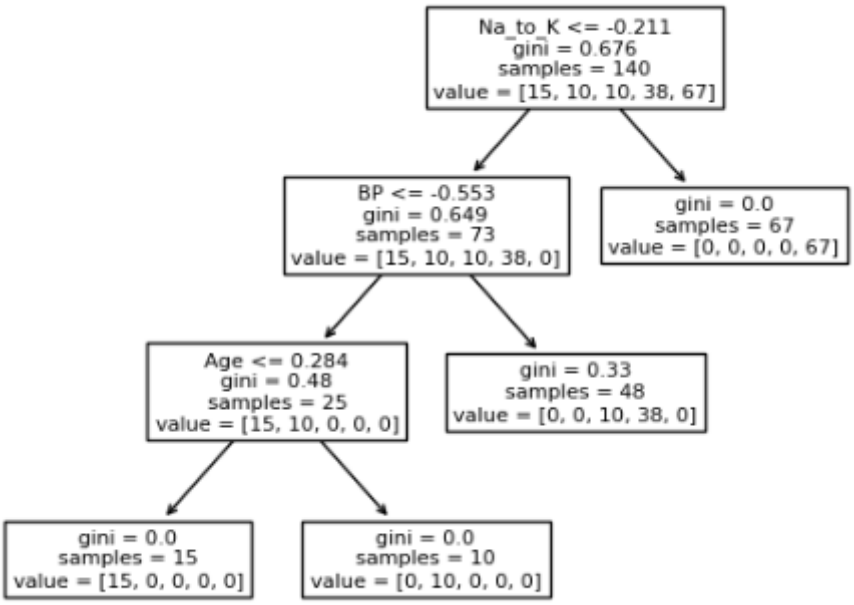
3 .تغییر فراپارامترهای مربوط به هرس کردن و تأثیر آنها

ccp_alpha

مقداردهی به ccp_alpha به مدل این امکان را می‌دهد که تعادل مناسبی بین سادگی و دقت را برقرار کند. مقدار بزرگتر ccp_alpha منجر به هرس کردن بیشتر و یک درخت ساده‌تر می‌شود که ممکن است دقت کمتری داشته باشد اما از بیش‌برازش جلوگیری می‌کند. از طرف دیگر، مقدار کمتر ccp_alpha منجر به حفظ ساختار بیشتر درخت و افزایش دقت روی داده‌های آموزش می‌شود، اما این امر ممکن است باعث بیش‌برازش شود.

با تغییر فراپارامتر های ccp_alpha و max_depth داریم:

```
clf_M2 = DecisionTreeClassifier(random_state=24,ccp_alpha=0.05,max_depth=3)
```




```
0.9
[[ 8  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  0  0  0]
 [ 0  0  6 16  0]
 [ 0  0  0  0 24]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         6
           2       0.00      0.00      0.00         0
           3       1.00      0.73      0.84        22
           4       1.00      1.00      1.00        24

    accuracy                           0.90        60
   macro avg       0.80      0.75      0.77        60
weighted avg       1.00      0.90      0.94        60
```

می بینیم که دقت پایین آمده و کلاس سوم هرس شده است و کلاس بندی نشده است.

.3

۳. توضیح دهید که روش‌هایی مانند جنگل تصادفی و AdaBoost چگونه می‌توانند به بهبود نتایج کمک کنند. سپس، با انتخاب یکی از این روش‌ها و استفاده از فراپارامترهای مناسب، سعی کنید نتایج پیاده‌سازی در مراحل قبلی را ارتقاء دهید.

راهنمایی: می‌توانید از پیوندهای زیر کمک بگیرید:

- sklearn.ensemble.RandomForestClassifier
- sklearn.ensemble.AdaBoostClassifier

روش‌هایی مانند جنگل تصادفی و AdaBoost می‌توانند بهبود نتایج در مسائل یادگیری ماشینی کمک کنند به دلایل زیر:

۱. جنگل تصادفی (Random Forest):

- تنوع بالا: جنگل تصادفی از چندین درخت تصمیم تشکیل شده است که هر کدام به صورت مستقل و به‌طور تصادفی از داده‌ها به‌طور تکراری نمونه‌گیری می‌کنند و درخت‌های مختلفی را تولید می‌کنند. این تنوع باعث می‌شود که مدل از افرازها و الگوهای مختلفی در داده استفاده کند.

- کاهش بیش‌برازش: به دلیل تنوع بالا و تصادفی‌سازی در فرآیند آموزش، جنگل تصادفی به خوبی از بیش‌برازش جلوگیری می‌کند و دارای عملکرد خوبی در داده‌های جدید است.

- انعطاف‌پذیری: جنگل تصادفی می‌تواند با مقادیر پیش‌فرض خوبی به خوبی کار کند و نیازی به تنظیم پارامترهای پیچیده ندارد.

- قابلیت استفاده از ویژگی‌های مختلف: این روش به راحتی می‌تواند با ویژگی‌های مختلف ساختاری و غیرساختاری سازگار باشد و از هر نوع داده‌ای استفاده کند.

۲. AdaBoost (Adaptive Boosting):

- توجه به نمونه‌های دشوار: AdaBoost با تمرکز بر نمونه‌های دشوار و نادرست در فرآیند آموزش، تلاش می‌کند تا در هر مرحله مدل را بهبود دهد و از بیش‌برازش جلوگیری کند.

- اجتماع یادگیری: در هر مرحله، AdaBoost یک مدل ضعیف جدید را به مدل قبلی اضافه می‌کند و تلاش می‌کند با ترکیب این مدل‌های ضعیف، یک مدل قوی و کارآمد بسازد.

- تعمیم‌پذیری بالا: AdaBoost دارای قابلیت تعمیم‌پذیری بالاست و معمولاً به خوبی در مسائل دسته‌بندی با تعداد کمی از داده‌ها عمل می‌کند.

- استفاده از توابع هدف متفاوت: این روش می‌تواند با استفاده از توابع هدف متفاوتی مانند Entropy، Gini Index یا میزان اشتباه دسته‌بندی کند و به دسته‌بندی بهتری دست یابد.

به طور کلی، جنگل تصادفی و AdaBoost از دو روش قدرتمند و مؤثر در یادگیری ماشینی هستند که بهبود نتایج و عملکرد مدل‌های پیش‌بینی را تسهیل می‌کنند.

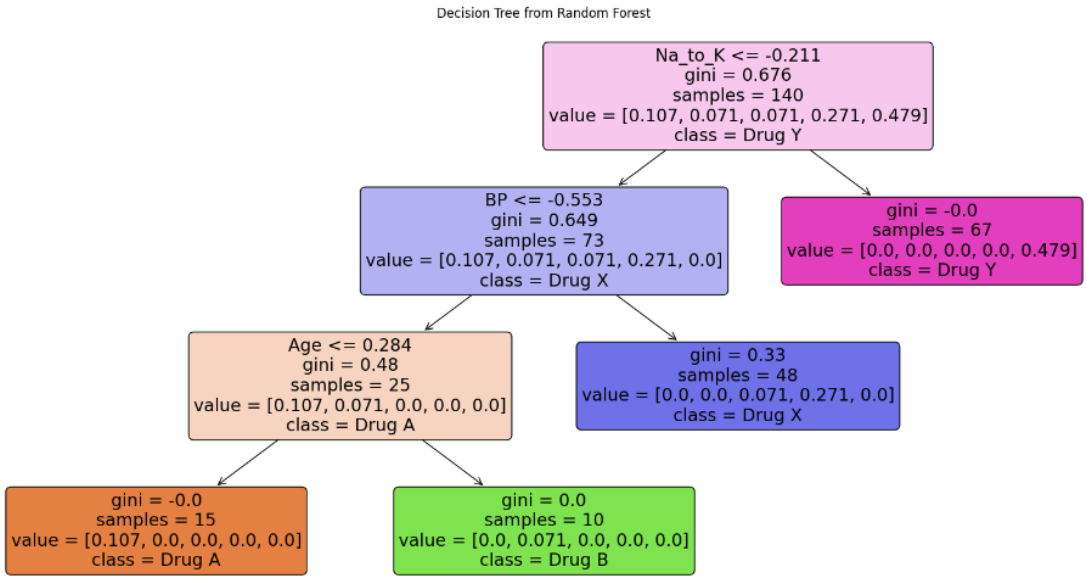
AdaBoost :

اگر داده های قبل با hyper parameter های تغییر داده که دقت 90% داده بود و نتوانسته بود کلاس 3 را تشخیص بدهد را به adaboost بدهیم داریم:

لازم به ذکر هست که مادقت 100% بدون تغییر hyper_parameter ها در بخش 1 گرفتیم.

```
base_estimator = DecisionTreeClassifier(random_state=24,ccp_alpha=0.05,max_depth=3)

# Initialize the AdaBoostClassifier with the custom base estimator
clf_M_ada = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=100, learning_rate=0.1, random_state=24)
```

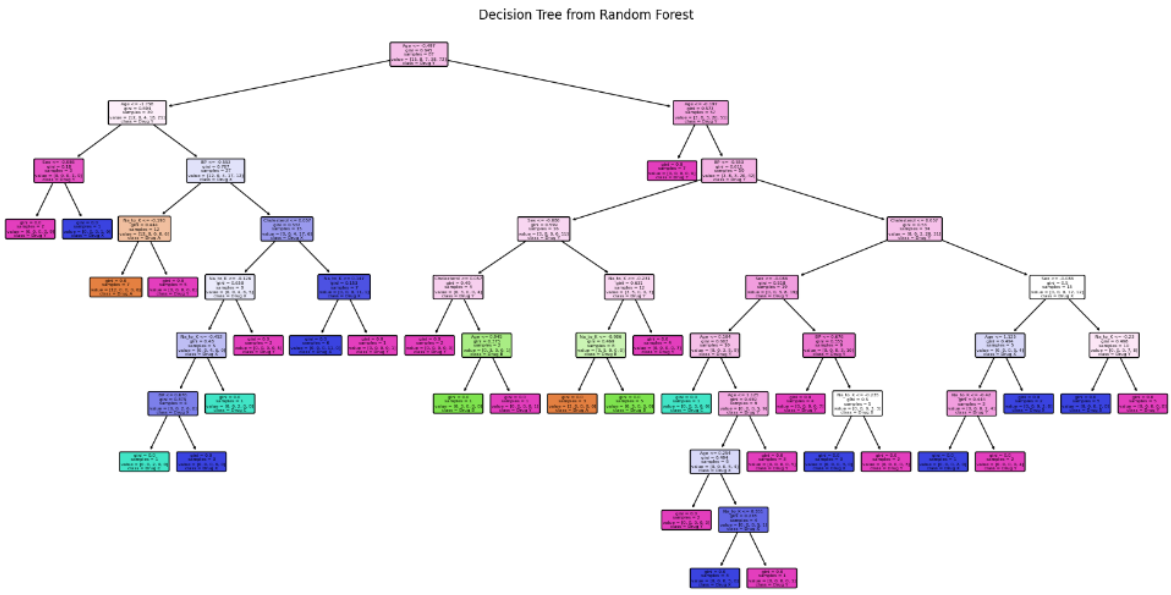
```
Accuracy: 1.0
Confusion Matrix:
 [[ 8  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  6  0  0]
 [ 0  0  0 16  0]
 [ 0  0  0  0 24]]
Classification Report/n:              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00        16
           4       1.00      1.00      1.00        24

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```



Random Forest

همچنین برای Random Forest داریم :

```
Accuracy: 1.0
Confusion Matrix
: [[ 8  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  6  0  0]
 [ 0  0  0 16  0]
 [ 0  0  0  0 24]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00        16
           4       1.00      1.00      1.00        24

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```



که تماما دقت 100% داشته اند

نکته

لازم به ذکر است که این دیتاست دارو با تعداد ویژگی و نمونه کمتر راحت تر حل شد در فایل کد گوگل کلب دیتاست دوم نیز حل شده ولی برای سادگی در گزارش فقط دیتاست اول را آوردم.

سوال 4

۴ سوال چهارم

دیتاست بیماری قلبی را در نظر بگیرید. داده‌ها را به دو بخش آموزش و آزمون تقسیم کرده و ضمن انجام پیش‌پردازش‌هایی که روی آن لازم می‌دانید و با فرض گاوسی‌بودن داده‌ها، از الگوریتم طبقه‌بندی Bayes استفاده کنید و نتایج را در قالب ماتریس درهم‌ریختگی و classification_report تحلیل کنید. تفاوت میان دو حالت Macro و Micro را در کتابخانهٔ سایکیت‌لرن شرح دهید.

درنهایت، پنج داده را به‌صورت تصادفی از مجموعهٔ آزمون انتخاب کنید و خروجی واقعی را با خروجی پیش‌بینی‌شده مقایسه کنید.

داده ها را دریافت کرده و بررسی می کنیم:

```
print(data.columns)
print(data.shape)
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
(1025, 14)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

1025 نمونه وتعداد 13 ویژگی داریم که یکی از آن ها هدف یا ’target‘ هست.

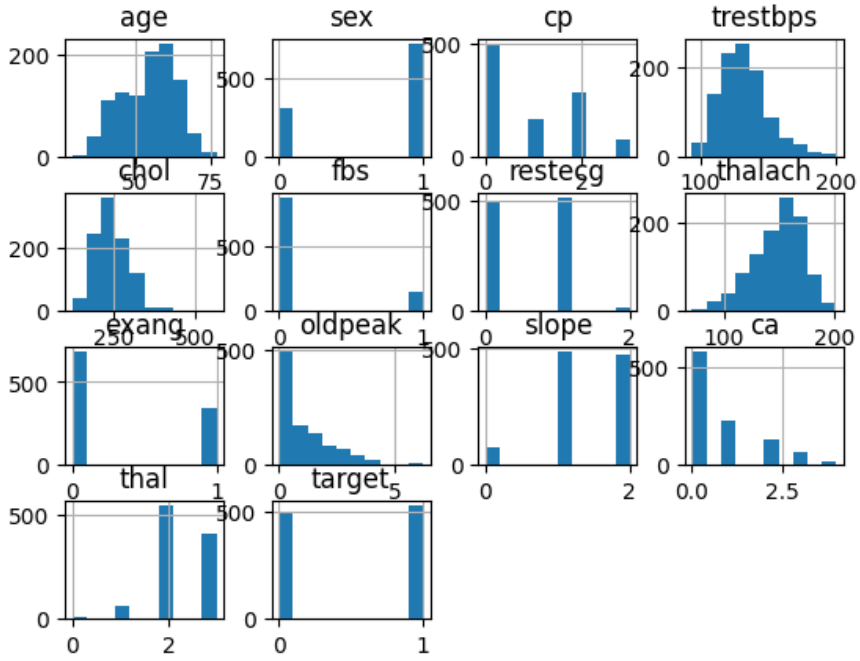
از این 14 داده 13 تای آنها integer و یک float داریم.همچنین داده null ای در دیتاست موجود نیست.

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.754146	2.323902	0.513171
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660	0.500070
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Histogram

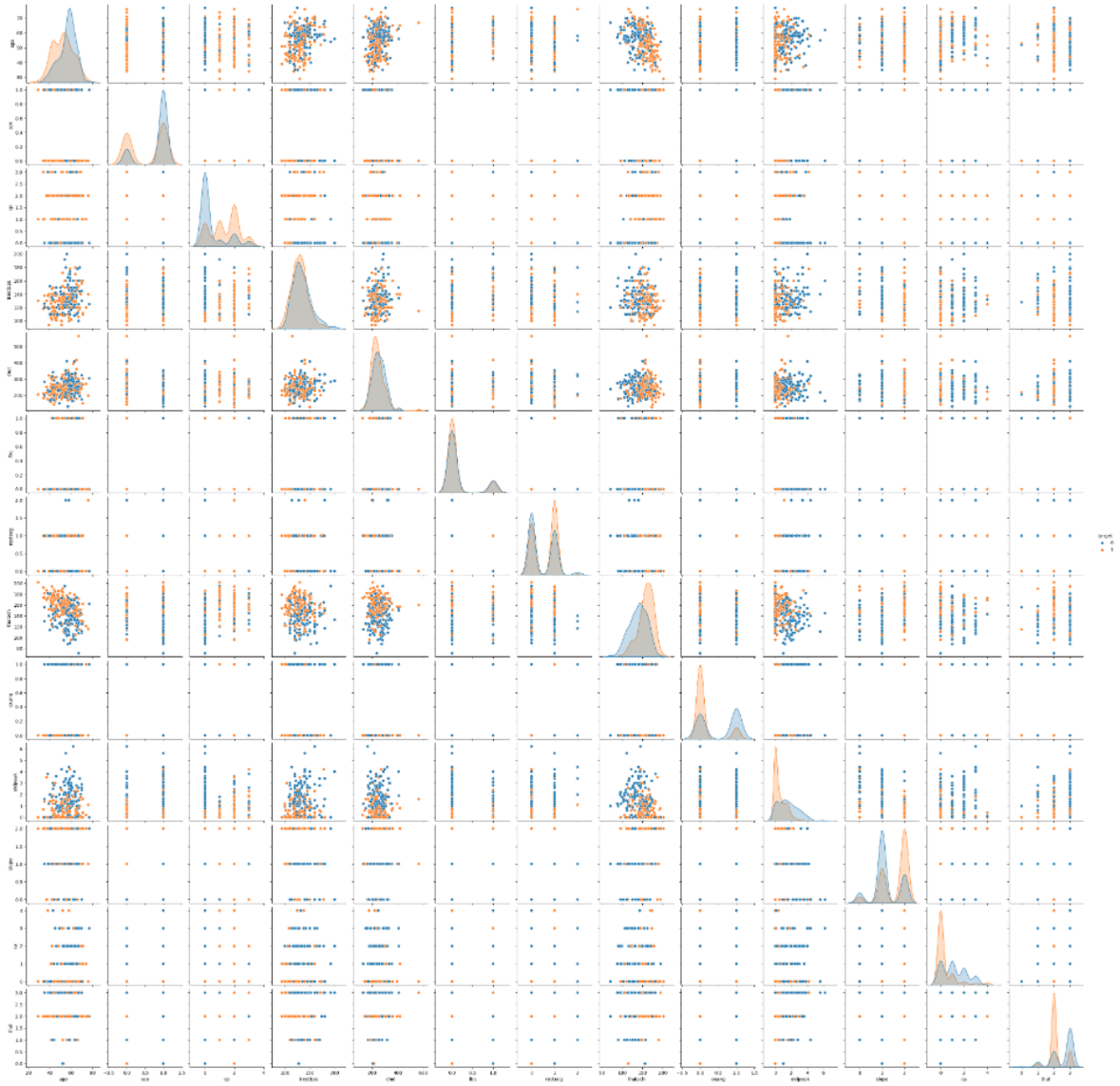
نمودار هیستوگرام را رسم میکنیم



هر هیستوگرام نماینده توزیع مقادیر برای یک ویژگی خاص در مجموعه داده است. می‌توانیم از این هیستوگرام‌ها برای به دست آوردن بینش‌ها درباره توزیع داده‌های خود و شناسایی الگوها یا نقصان‌های موجود استفاده کنید.

Pairplot

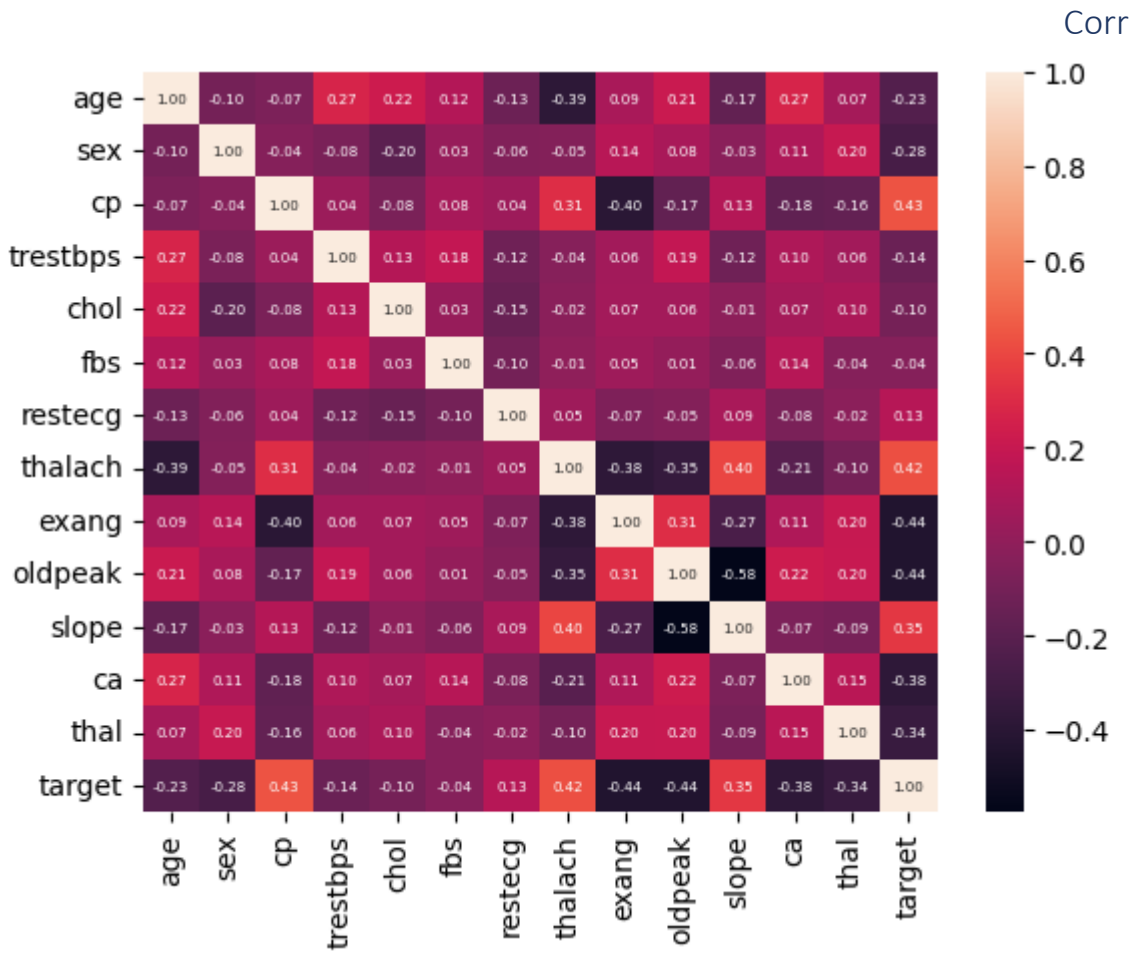
همچنین با تابع sns.pairplot() از کتابخانه Seaborn یک شبکه از نمودارهای جفتی برای هر متغیر در مجموعه داده شما ایجاد می‌کنیم:



این تابع همچنین شامل نمودارهای پراکندگی برای روابط مشترک و هیستوگرام‌ها برای توزیع‌های یک متغیر در قطر است.

پارامتر hue برای مشخص کردن یک متغیر دسته‌ای استفاده می‌شود که برای رنگ‌آمیزی نقاط داده در نمودار استفاده می‌شود و اطلاعات اضافی درباره روابط بین متغیرها را فراهم می‌کند.

بنابراین، sns.pairplot(data, hue='target') یک شبکه از نمودارهای جفتی ایجاد می‌کند که نقاط داده بر اساس متغیر "هدف" رنگ آمیزی شده‌اند. این به شما امکان می‌دهد که روابط بین متغیرها را به صورت تصویری بررسی کنید در حالی که هدف را در نظر می‌گیرید. نمودار های مربوطه در colab آمده است.



با تحلیل ضرایب همبستگی (correlation coefficients) می‌توانیم برخی از روابط میان ویژگی‌ها و متغیر هدف را مشاهده کنیم. برای مثال:

CP (نوع درد قفسه سینه): همبستگی مثبت و معنی‌دار با متغیر هدف (حدود 0.43). این نشان می‌دهد که نوع خاصی از درد قفسه سینه ممکن است با ابتلای به بیماری قلبی مرتبط باشد.

Thalach (ضربان قلب حداکثر): نیز همبستگی مثبت و معنی‌داری با متغیر هدف دارد (حدود 0.42). این نشان می‌دهد که فرکانس ضربان قلب در حداکثر ورزش ممکن است به عنوان یک عامل مهم برای تشخیص بیماری قلبی مورد استفاده قرار گیرد.

ویژگی‌های منفیاً مرتبط با متغیر هدف:

Exang (آنژین ناشی از فعالیت): همبستگی منفی و معنی‌داری با متغیر هدف دارد (حدود -0.44). این نشان می‌دهد که آنژین ناشی از فعالیت ممکن است به عنوان یک علامت برای بیماری قلبی در نظر گرفته شود.

Oldpeak (کاهش ST): همچنین همبستگی منفی و معنی‌داری با متغیر هدف دارد (حدود -0.44). این نشان می‌دهد که کاهش آمپلیتود ST پس از ورزش ممکن است به عنوان یک علامت برای بیماری قلبی در نظر گرفته شود.

ویژگی‌های کم‌تاثیر:

میان برخی از ویژگی‌ها و متغیر هدف، همبستگی ضعیف و یا نامعنی وجود دارد. به عنوان مثال، بین سن (Age) و متغیر هدف، همبستگی معنی‌داری وجود ندارد (حدود -0.23).

ویژگی‌های تکراری:

برخی از ویژگی‌ها با یکدیگر همبستگی معنی‌داری دارند که می‌تواند نشان دهنده وجود ویژگی‌های تکراری باشد. برای مثال، همبستگی معنی‌داری بین سن و فشار خون در استراحت (Trestbps) وجود دارد (حدود 0.27).

این تحلیل‌ها می‌تواند به ما کمک کند تا ویژگی‌های مهم‌تر را برای پیش‌بینی بیماری‌های قلبی شناسایی کرده و به عنوان ورودی‌های اصلی برای مدل‌های طبقه‌بندی مورد استفاده قرار دهیم.

باشد. در این تحلیل، به بررسی اطلاعات بیشتری از هر یک از ویژگی‌ها می‌پردازیم:

این تحلیل ما را قادر می‌سازد تا ویژگی‌های مهم و تاثیرگذار در تشخیص بیماری‌های قلبی را شناسایی کرده و آن‌ها را به عنوان ورودی‌های اصلی برای مدل‌های طبقه‌بندی مورد استفاده قرار دهیم.

.1

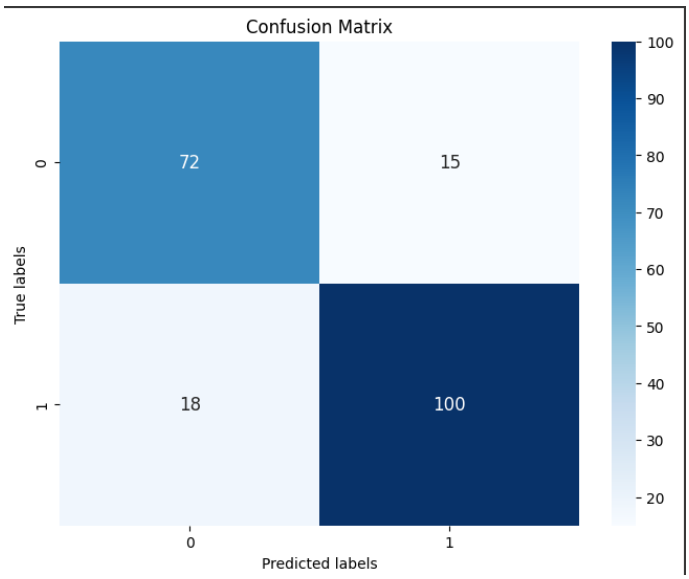
داده ها را با درصد تقسیم 20% به آموزش و تست تقسیم می کنیم .

```
(820, 13) (820, 1) (205, 13) (205, 1)
(820,)
(205,)
```

با GaussianNB() طبقه بند Bayes با توزیع گوسی را تعریف و به داده های آموزش ، فیت می کنیم:
ماتریس درهمریختگی و classification_report :

	precision	recall	f1-score	support
0	0.80	0.83	0.81	87
1	0.87	0.85	0.86	118
accuracy			0.84	205
macro avg	0.83	0.84	0.84	205
weighted avg	0.84	0.84	0.84	205

```
0.8390243902439024
[[ 72  15]
 [ 18 100]]
```



داده ها با دقت بالای 80 درصد با معیار های مختلف نتیجه داده اند.

در زمینه گزارش‌های طبقه‌بندی، اصطلاحات "میکرو" و "ماکرو" به روش‌های متفاوتی که برای محاسبه معیارهایی مانند percision، recall و F1-score استفاده می‌شود، اشاره دارند. در ادامه تفاوت بین این دو روش را بیان می‌کنم:

1. micro average:

به هر زوج نمونه-کلاس یک مشارکت مساوی در معیار کلی می‌دهد (به جز نتیجه‌ی وزن نمونه). به جای جمع کردن معیار برای هر کلاس، این معیارها را به یکدیگر تقسیم می‌کند تا یک نسبت کلی محاسبه شود. میانگین میکرو ممکن است در تنظیمات چند برچسبی، از جمله طبقه‌بندی چند کلاسه که یک کلاس اکثریت باید نادیده گرفته شود، ترجیح داده شود.

: macro average .2

میانگین معیارهای دودویی را محاسبه می‌کند و وزن مساوی به هر کلاس می‌دهد. در مسائلی که کلاس‌های با فراوانی کم همچنان مهم هستند، میانگین ماکرو می‌تواند روشی برای برجسته کردن عملکرد آن‌ها باشد. از طرف دیگر، فرضیه‌ی اینکه تمامی کلاس‌ها به یک اندازه مهم هستند اغلب صحیح نیست، بنابراین میانگین ماکرو به طور زیادی بر عملکرد معمولاً پایین کلاس‌های بافراوانی کم تأثیر خواهد گذاشت.

به طور خلاصه، تفاوت اصلی در چگونگی محاسبه معیارها است: میکرو همه نمونه‌ها را به طور مساوی در نظر می‌گیرد، در حالی که ماکرو-میانگین همه کلاس‌ها را به طور مساوی در نظر می‌گیرد. انتخاب بین میکرو-میانگین و ماکرو-میانگین به توجه به اهداف ارزیابی خاص و ویژگی‌های مجموعه داده وابسته است.

```
print(accuracy_score(y_pred,y_test))
```

```
0.8390243902439024
```

```
print(jaccard_score(y_pred,y_test))
print(jaccard_score(y_pred,y_test,average='micro'))
print(jaccard_score(y_pred,y_test,average='macro'))
```

```
0.7518796992481203
0.7226890756302521
0.718796992481203
```

```
print(f1_score(y_pred,y_test))
print(f1_score(y_pred,y_test,average='micro'))
print(f1_score(y_pred,y_test,average='macro'))
```

```
0.8583690987124463
0.8390243902439024
0.8359642103731723
```

```
print(precision_score(y_pred,y_test))
print(precision_score(y_pred,y_test,average='micro'))
print(precision_score(y_pred,y_test,average='macro'))
```

```
0.8695652173913043
0.8390243902439024
0.8347826086956522
```

```
print(recall_score(y_pred,y_test))
print(recall_score(y_pred,y_test,average='micro'))
print(recall_score(y_pred,y_test,average='macro'))
```

```
0.847457627118644
0.8390243902439024
0.8375219170075978
```

حال 5 داده رندوم از مجموعه تست انتخاب شده تخمین زده و با مقدار واقعی مقایسه میشود:

```
Data Point 1
True Label: 1
Predicted Label: 1

Data Point 2
True Label: 0
Predicted Label: 0

Data Point 3
True Label: 0
Predicted Label: 0

Data Point 4
True Label: 0
Predicted Label: 0

Data Point 5
True Label: 0
Predicted Label: 0
```

مشاهده می شود که تمام دادخ ها به درستی تخمین زده شدند.