به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش مینی پروژه شماره سوم

[ایمان فکری اسکی]

[۹۹۲۹۰۸۳]

استاد : آقای دکتر مهدی علیاری

بهمن ماه ۱۴۰۲

فهرست مطالب

چکیده :

پروژه حاضر به بررسی و ارتباط بین دو رویکرد مهم در حوزه هوش مصنوعی، یعنی شبکه‌های فازی و درخت تصمیم، می‌پردازد. شبکه‌های فازی به عنوان یک سیستم هوشمند با توانمندی در مدل‌سازی اطلاعات ناواضح و عدم قطعیت شناخته شده‌اند. از سوی دیگر، درخت تصمیم به عنوان یک روش تصمیم‌گیری سلسله مراتبی و ساختاری، توانایی خوبی در تجزیه و تحلیل تصمیمات پیچیده و متعدد را ارائه می‌دهند.

در این پروژه، به معرفی مفاهیم اساسی شبکه‌های فازی و درخت تصمیم پرداخته و نقاط اشتراک و تفاوت‌های آنها را بررسی می‌کنیم. همچنین، نحوه استفاده از این دو رویکرد در حل مسائل عملی، از جمله پیش‌بینی، کنترل و تصمیم‌گیری، مورد ارزیابی قرار می‌گیرد.

در ادامه، به بررسی مطالعات موردی و پروژه‌های کاربردی با استفاده از شبکه‌های فازی و درخت تصمیم می‌پردازیم تا کاربردهای عملی این دو روش در مسائل مختلف را مورد بررسی قرار دهیم.

این پروژه نه تنها به درک عمیق‌تر از اصول این دو رویکرد کمک می‌کند بلکه ارائه‌های کاربردی نیز را در زمینه‌های مختلف ارتباطی با هوش مصنوعی و مهندسی سامانه‌ها فراهم می‌سازد.

مقدمه :

در دهه‌های اخیر، تلاش‌های فراوانی برای افزایش کارایی و اطمینان از تصمیمات گرفته شده در سیستم‌ها و مسائل پیچیده صورت گرفته است. در این زمینه، دو رویکرد مهم و مؤثر به نام‌های "شبکه‌های فازی" و "درخت تصمیم" برای مدل‌سازی و تصمیم‌گیری مورد توجه قرار گرفته‌اند. این دو تکنیک، هرکدام با ویژگی‌ها و امکانات خود، در حل مسائل علوم کامپیوتر، مهندسی، و حتی به عنوان ابزارهای تصمیم‌گیری در زندگی روزمره ما مورد استفاده قرار گرفته‌اند.

در این گزارش، به مطالعه و بررسی عمیق این دو رویکرد می‌پردازیم و تلاش می‌کنیم تا به یک درک جامع از نقاط قوت و ضعف هرکدام برسیم. شبکه‌های فازی به عنوان یک ابزار مدل‌سازی پردازش اطلاعات ناواضح و عدم قطعیت شناخته شده‌اند. از سوی دیگر، درخت تصمیم به عنوان یک ساختار تصمیم‌گیری مرتبط و سلسله مراتبی، توانایی مدیریت تصمیمات پیچیده را دارا می‌باشد.

با ارائه یک تحلیل جامع از این دو رویکرد، ما نقشه‌ای دقیق از اینکه چگونه می‌توانند در مواجهه با چالش‌های مختلف، از جمله پیش‌بینی داده‌ها، کنترل سیستم‌ها، و تصمیم‌گیری در شرایط ناپایدار مورد استفاده قرار گیرند، ارائه خواهیم داد.

در طول این گزارش، به بررسی پیشرفت‌های اخیر و کاربردهای عملی این دو تکنیک در حل مسائل واقعی پرداخته و اهمیت آنها در توسعه‌ی راهکارهای هوشمند برای جامعه را مورد بحرانی قرار خواهیم داد.

سوال اول :

با استفاده از کران مرتبه اول (رابطهٔ ۴-۱۱ در [۲]) و کران مرتبه دوم (رابطهٔ ۱۱-۱۰ در [۲])، دو سیستم فازی با غیرفازی‌ساز میانگین و ماکزیمم طراحی کنید که تابع $g(x_1,x_2)=\frac{1}{3+x_1+x_2}$ روی $U=[-1,1]\times[-1,1]$ را به‌شکل یکنواخت و با دقت $\epsilon=0.1$ تقریب بزند. سیستم‌های فازی طراحی‌شده را رسم کرده و با هم مقایسه کنید.

در ابتدا در سوالی که به ما داده شده است باید تعداد توابع تعلق را محاسبه کنیم. در این قسمت دو حالت متفاوت برای این موضوع خواهیم داشت که باید یک بار برای کران اول و بار دیگر برای کران دوم ، مشخصات مربوطه محاسبه شود. به صورت زیر عمل می کنیم :

برای کران مرتبه اول :$\alpha=-1\ \ \beta=1$

$$\|g-f\|_\infty=\sup_{x\in U}|g(x)-f(x)|\le\left\|\frac{\partial g}{\partial x_1}\right\|_\infty h_1+\left\|\frac{\partial g}{\partial x_2}\right\|_\infty h_2\le\epsilon$$

$$\|g-f\|_\infty=\sup_{x\in U}|g(x)-f(x)|\le\left\|\frac{\partial g}{\partial x_1}\right\|_\infty h_1+\left\|\frac{\partial g}{\partial x_2}\right\|_\infty h_2\le\epsilon,\begin{cases}\left\|\frac{\partial g}{\partial x_1}\right\|_\infty=\sup_{x\in U}\left|\frac{\partial g}{\partial x_i}\right|\\ h_i=\max_{1\le j\le N_{i-1}}\left|e_i^{j+1}-e_i^j\right|\end{cases}$$

$$\varepsilon>h\left(\left\|\frac{\partial g}{\partial x_1}\right\|_\infty+\left\|\frac{\partial g}{\partial x_2}\right\|_\infty\right)\to h<\frac{\varepsilon}{\left\|\frac{\partial g}{\partial x_1}\right\|_\infty+\left\|\frac{\partial g}{\partial x_2}\right\|_\infty}$$

مقدار ماکسیمم تابع به ازای x_1=-1 و x_2=-1 محاسبه شده است :

$$\left\|\frac{\partial g}{\partial x_1}\right\|_\infty=\sup_{x\in U}\left|\frac{\partial g}{\partial x_1}\right|=\sup_{x\in U}\left|\frac{-1}{(3+x_1+x_2)^2}\right|=\left|\frac{-1}{(3-1-1)^2}\right|=1$$

$$\left\|\frac{\partial g}{\partial x_2}\right\|_\infty=\sup_{x\in U}\left|\frac{\partial g}{\partial x_2}\right|=\sup_{x\in U}\left|\frac{-1}{(3+x_1+x_2)^2}\right|=\left|\frac{-1}{(3-1-1)^2}\right|=1$$

در نهایت می توانیم با جایگذاری مقادیر بدست آمده در رابطه بالا ، مقدار h و تعداد توابع تعلق را محاسبه کنیم.

$$h<\frac{\varepsilon=0.1}{1+1}\to h<\frac{0.1}{2}\to h<\frac{1}{20}\to h<0.05$$

$$h=\frac{\beta-\alpha}{n}=\frac{1-(-1)}{n}=\frac{2}{n}=0.05\to n=40\to N=n+1=40+1=41$$

بنابراین ما برای این سیستم فازی (کران مرتبه اول)، ۴۱ تابع تعلق مثلثی خواهیم داشت.

توابع تعلق به صورت زیر خواهند بود :

$$\mu_{A^1}(x) = \mu_{A^1}(x; a_1, b_1, c_1) = \mu_{A^1}(x; -1, -1, -1 + h)$$

$$\mu_{Aj}(x) = \mu_{A}j\,(x; a_j, b_j, c_j) = \mu_{A}j\,(x; e^{j-1}, e^{j}, e^{j+1}), \quad \begin{cases} j = 2, \cdots, 40 \\ e^j = \alpha + h(j-1) = -1 + 0/05(j-1) \end{cases}$$

$$\mu_{A^{41}}(x) = \mu_{A^{41}}(x; a_{41}, b_{41}, c_{41}) = \mu_{A^{41}}(x; 1 - h, 1, 1)$$

برای کران مرتبه دوم :$\alpha = -1 \;\; \beta = 1$

$$\|g(x)-f(x)\|_\infty \le \frac{1}{8}\left[\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty h_1^2 + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty h_2^2\right] \le \epsilon, \quad \begin{cases} \left\|\frac{\partial^2 g}{\partial x_i^2}\right\|_\infty = \sup_{x\in U}\left|\frac{\partial^2 g}{\partial x_i^2}\right| \\ h_i = \max_{1\le j\le N_{i-1}}\left|e_i^{j+1} - e_i^j\right| (i = 1, 2) \end{cases} \quad (15)$$

از آن‌جا که دقت تقریب $\epsilon = 0/1$ ذکر شده و فرض شده که $h_1 = h_2 = h$ می‌نویسیم:

$$h^2 < \frac{8\varepsilon}{\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty} \rightarrow h < \sqrt{\frac{8\varepsilon}{\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty}} \quad (16)$$

مقدار ماکسیمم تابع به ازای x_1=-1 و x_2=-1 محاسبه شده است :

$$\left\|\frac{\partial^2 g}{\partial x1^2}\right\| = \sup\left|\frac{\partial^2 g}{\partial x1^2}\right| = \sup\left|\frac{2}{(x1 + x2 + 3)^3}\right| = 2$$

$$\left\|\frac{\partial^2 g}{\partial x2^2}\right\| = \sup\left|\frac{\partial^2 g}{\partial x2^2}\right| = \sup\left|\frac{2}{(x1 + x2 + 3)^3}\right| = 2$$

$$h < \sqrt{\frac{8\varepsilon}{\left\|\frac{\partial^2 g}{\partial x_1^2}\right\|_\infty + \left\|\frac{\partial^2 g}{\partial x_2^2}\right\|_\infty}} = \sqrt{\frac{8 \times 0/1}{2 + 2}} = 0/4472 \quad (18)$$

حال با محاسبهٔ حدود h برای صحیح به‌دست‌آمدن n آن را معادل ۰/۲۵ در نظر می‌گیریم، و تعداد توابع تعلق را محاسبه خواهیم کرد. بنابراین داریم:

$$h = \frac{\beta - \alpha}{n} = \frac{1 - (-1)}{n} = \frac{2}{n} = 0/25 \rightarrow n = 8 \quad (19)$$

بنابراین ما برای این سیستم فازی (کران مرتبه دوم) ، ۹ تابع تعلق مثلثی خواهیم داشت.

توابع تعلق به صورت زیر خواهند بود :

$$\mu_{A^1}(x) = \mu_{A^1}(x; a_1, b_1, c_1) = \mu_{A^1}(x; -1, -1, -1 + h)$$

$$\mu_{A^j}(x) = \mu_{A}j(x; a_j, b_j, c_j) = \mu_A^j(x; e^{j-1}, e^j, e^{j+1}), \quad \begin{cases} j = 2, \cdots, 40 \\ e^j = \alpha + h(j-1) = -1 + 0.25(j-1) \end{cases}$$

$$\mu_{A^9}(x) = \mu_{A^9}(x; a_9, b_9, c_9) = \mu_{A^9}(x; 1 - h, 1, 1)$$

برای غیر فازی ساز میانگین داریم :

کران مرتبه اول

$$f(x) = \frac{\sum_{i_1=1}^{41}\sum_{i_2=1}^{41} g\left(e_1^{i_1}, e_2^{i_2}\right)\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}{\sum_{i_1=1}^{41}\sum_{i_2=1}^{41}\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}$$

کران مرتبه دوم

$$f(x) = \frac{\sum_{i_1=1}^{9}\sum_{i_2=1}^{9} g\left(e_1^{i_1}, e_2^{i_2}\right)\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}{\sum_{i_1=1}^{9}\sum_{i_2=1}^{9}\left[\mu_{A_1^{i_1}}(x_1)\mu_{A_2^{i_2}}(x_2)\right]}$$

حال که محاسبات دستی را انجام داده ایم ، می خواهیم **کد متلب** مربوط به غیر فازی ساز میانگین را در ابتدا برای کران مرتبه اول و سپس برای کران مرتبه دوم بزنیم. پس در ابتدا به سراغ غیر فازی ساز میانگین برای کران مرتبه اول می رویم :

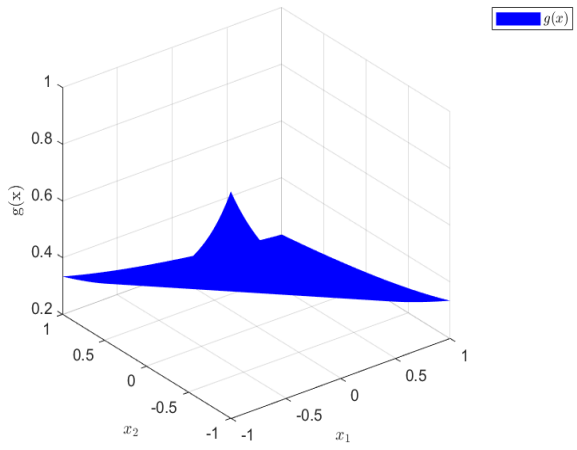
برای غیر فازی ساز میانگین و کران مرتبه اول :

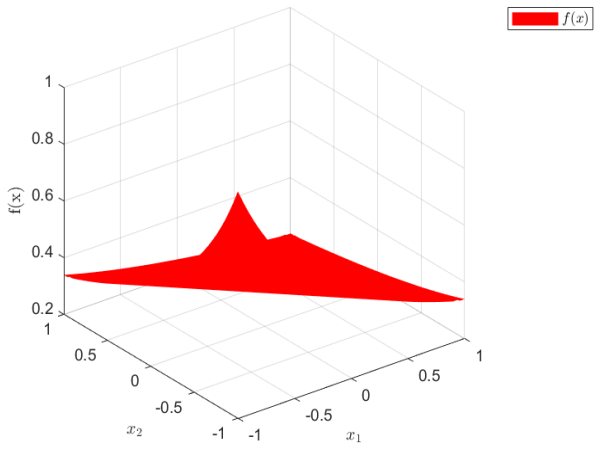
```
clc;
clear;
close all;

%% First order limit
alpha = -1;
beta = 1;
h = 0.05;
```

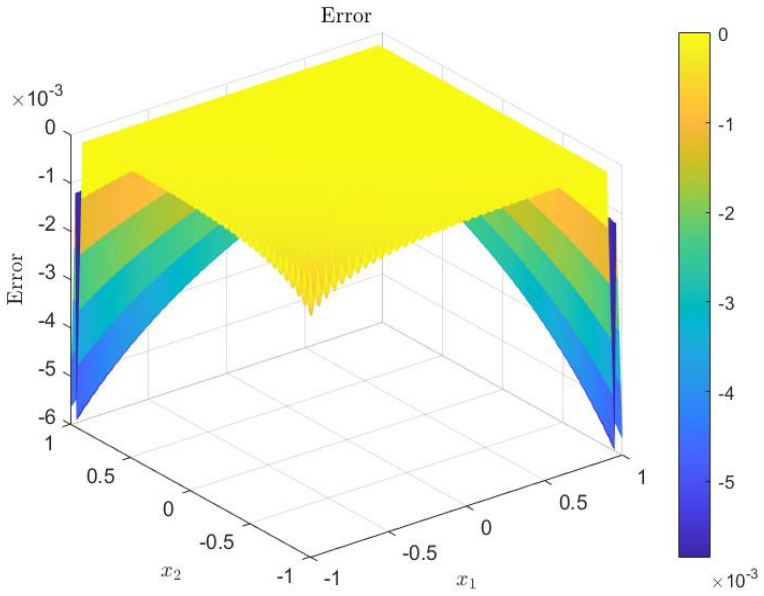
```
N = 41;
x1 = alpha:0.01:beta;
x2 = alpha:0.01:beta;
[x1, x2] = meshgrid(x1, x2);
g_bar = zeros(N*N, 1);
e_i1 = zeros(N, 1);
e_i2 = zeros(N, 1);
num = 0;
den = 0;
k = 1;
trimf = @(x, abc) max(min((x - abc(1)) / (abc(2) -
abc(1)), (abc(3) - x) / (abc(3) - abc(2))), 0);
for i1 = 2:N
    for i2 = 2:N
        e_i1(i1-1,1) = -1 + h*(i1-2);
        e_i2(i2-1,1) = -1 + h*(i2-2);
        if i1 == 2
            mu_A_x1 = trimf(x1, [-1, -1, -1+h]);
        elseif i1 == N
            mu_A_x1 = trimf(x1, [1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1, [-1+h*(i1-3), -1+h*(i1-
2), -1+h*(i1-1)]);
        end
        if i2 == 2
            mu_A_x2 = trimf(x2, [-1, -1, -1+h]);
        elseif i2 == N
            mu_A_x2 = trimf(x2, [1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2, [-1+h*(i2-3), -1+h*(i2-
2), -1+h*(i2-1)]);
        end
        g_bar(k,1) = 1 / (3 + e_i1(i1-1,1) + e_i2(i2-
1,1));
        num = num + g_bar(k,1) * mu_A_x1 .* mu_A_x2;
        den = den + mu_A_x1 .* mu_A_x2;
        k = k + 1;
    end
end
```

```
f_x = num ./ den;
g_x = 1 ./ (3 + x1 + x2);
figure
surf(x1, x2, g_x, 'FaceColor', 'b', 'EdgeColor',
'none');
xlabel('$x_1$', 'Interpreter', 'latex');
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('g(x)', 'Interpreter', 'latex');
legend('$g(x)$', '$f(x)$', 'Interpreter', 'latex');
grid on;
figure
surf(x1, x2, f_x, 'FaceColor', 'r', 'EdgeColor',
'none');
xlabel('$x_1$', 'Interpreter', 'latex');
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('f(x)', 'Interpreter', 'latex');
legend('$f(x)$', 'Interpreter', 'latex');
grid on;
figure
surf(x1, x2, g_x - f_x, 'EdgeColor', 'none');
xlabel('$x_1$', 'Interpreter', 'latex');
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('Error', 'Interpreter', 'latex');
title('Error', 'Interpreter', 'latex');
colorbar;
grid on;
```

در نتیجه نتایج شکل نمودار سیستم فازی و نمودار تابع اصلی به همراه نمودار تابع خطا به صورت زیر خواهند شد







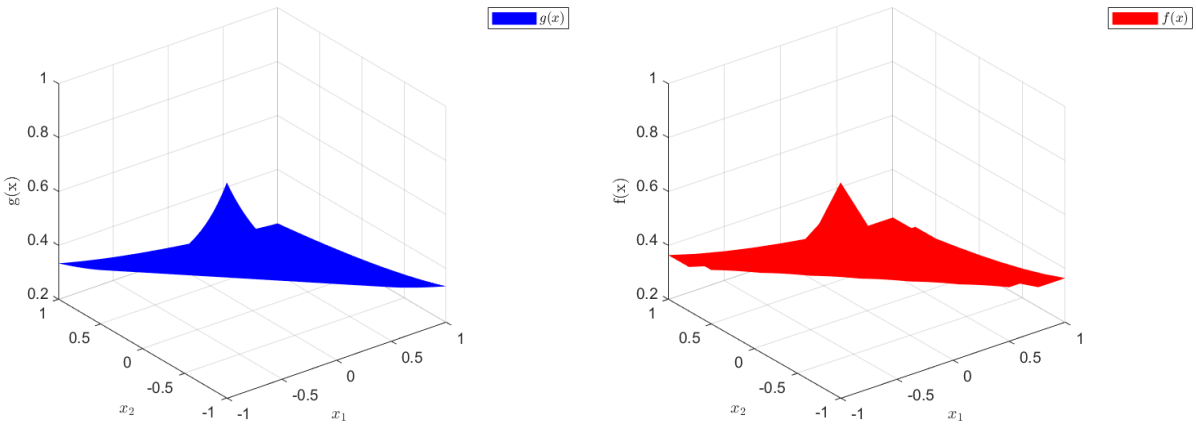
برای غیر فازی ساز میانگین و کران مرتبه دوم :

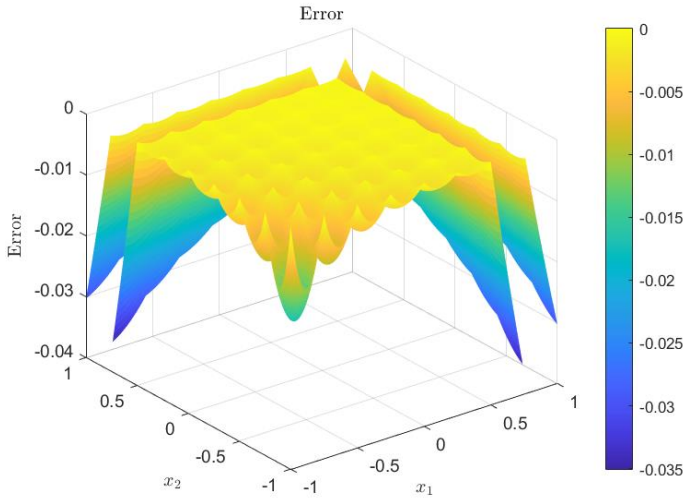
```
%% Second order limit
alpha = -1;
beta = 1;
h = 0.25;
N = 9;
x1 = alpha:0.01:beta;
x2 = alpha:0.01:beta;
[x1, x2] = meshgrid(x1, x2);
g_bar = zeros(N*N, 1);
e_i1 = zeros(N, 1);
e_i2 = zeros(N, 1);
num = 0;
den = 0;
k = 1;
% Define trimf function
trimf = @(x, abc) max(min((x - abc(1)) / (abc(2) -
abc(1)), (abc(3) - x) / (abc(3) - abc(2))), 0);
% Loop to calculate memberships and g_bar
for i1 = 2:N
    for i2 = 2:N
        e_i1(i1-1,1) = -1 + h*(i1-2);
```

```
        e_i2(i2-1,1) = -1 + h*(i2-2);
        if i1 == 2
            mu_A_x1 = trimf(x1, [-1, -1, -1+h]);
        elseif i1 == N
            mu_A_x1 = trimf(x1, [1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1, [-1+h*(i1-3), -1+h*(i1-
2), -1+h*(i1-1)]);
        end
        if i2 == 2
            mu_A_x2 = trimf(x2, [-1, -1, -1+h]);
        elseif i2 == N
            mu_A_x2 = trimf(x2, [1-h, 1, 1]);
        else
            mu_A_x2 = trimf(x2, [-1+h*(i2-3), -1+h*(i2-
2), -1+h*(i2-1)]);
        end
        g_bar(k,1) = 1 / (3 + e_i1(i1-1,1) + e_i2(i2-
1,1));
        num = num + g_bar(k,1) * mu_A_x1 .* mu_A_x2;
        den = den + mu_A_x1 .* mu_A_x2;
        k = k + 1;
    end
end
% Calculate f_x and g_x
f_x = num ./ den;
g_x = 1 ./ (3 + x1 + x2);
figure
surf(x1, x2, g_x, 'FaceColor', 'b', 'EdgeColor',
'none');
xlabel('$x_1$', 'Interpreter', 'latex');
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('g(x)', 'Interpreter', 'latex');
legend('$g(x)$', '$f(x)$', 'Interpreter', 'latex');
grid on;
figure
surf(x1, x2, f_x, 'FaceColor', 'r', 'EdgeColor',
'none');
xlabel('$x_1$', 'Interpreter', 'latex');
```

```
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('f(x)', 'Interpreter', 'latex');
legend('$f(x)$', 'Interpreter', 'latex');
grid on;
figure
surf(x1, x2, g_x - f_x, 'EdgeColor', 'none');
xlabel('$x_1$', 'Interpreter', 'latex');
ylabel('$x_2$', 'Interpreter', 'latex');
zlabel('Error', 'Interpreter', 'latex');
title('Error', 'Interpreter', 'latex');
colorbar;
grid on;
```

در نتیجه نتایج شکل نمودار سیستم فازی و نمودار تابع اصلی به همراه نمودار تابع خطا به صورت زیر خواهند شد





حال به سراغ غیر فازی ساز ماکزیمم می رویم و آن را نیز در دو حالت کران مرتبه اول و کران مرتبه دوم مقایسه می کنیم :

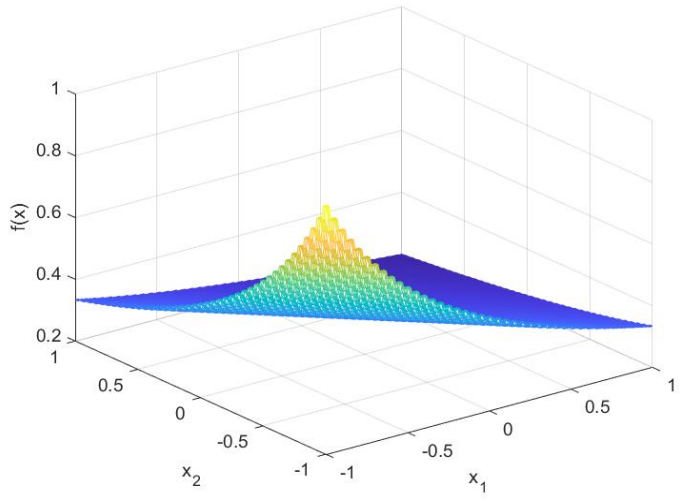
مقادیر h و N که همان مقادیری هستند که در بخش قبل بدست آمده است.

کران مرتبه اول :

```
%% First order limit (max)
alfa=-1;
beta=1;
h=0.05;
N=41;
x1=alfa:0.01:beta;
x2=x1;
[~,n1]=size(x1);
[~,n2]=size(x2);
e1=beta*ones(1,N+1);
e2=beta*ones(1,N+1);
for j=1:N
    e1(j)=alfa+h*(j-1);
    e2(j)=alfa+h*(j-1);
end
f_x=zeros(n1,n2);
for k1=1:n1
    for k2=1:n2


i1=min(find(e1<=x1(1,k1),1,'last'),find(e1>=x1(1,k1),1)
);

i2=min(find(e2<=x2(1,k2),1,'last'),find(e2>=x2(1,k2),1)
);

       if x1(1,k1)>=e1(1,i1)&&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1))&&
x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=.5*(e2(1,i2)+e2(1,1+i2))
           p=0;
           q=0;
       elseif x1(1,k1)>=e1(1,i1)&&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1))&&
```

```
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))&&
x2(1,k2)<=e2(1,1+i2)
            p=0;
            q=1;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1))&&
x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=e2(1,i2)&&
x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            p=1;
            q=0;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1))&&
x1(1,k1)<=e1(1,1+i1)&&
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))&&
x2(1,k2)<=e2(1,1+i2)
            p=1;
            q=1;
        end

        f_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));
    end
end
[x1,x2]=meshgrid(x1,x2);
figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,transpose(f_x));
xlabel('x_1')
ylabel('x_2')
zlabel('f(x)')
```

در نتیجه شکل نمودار سیستم فازی به صورت زیر خواهند شد :

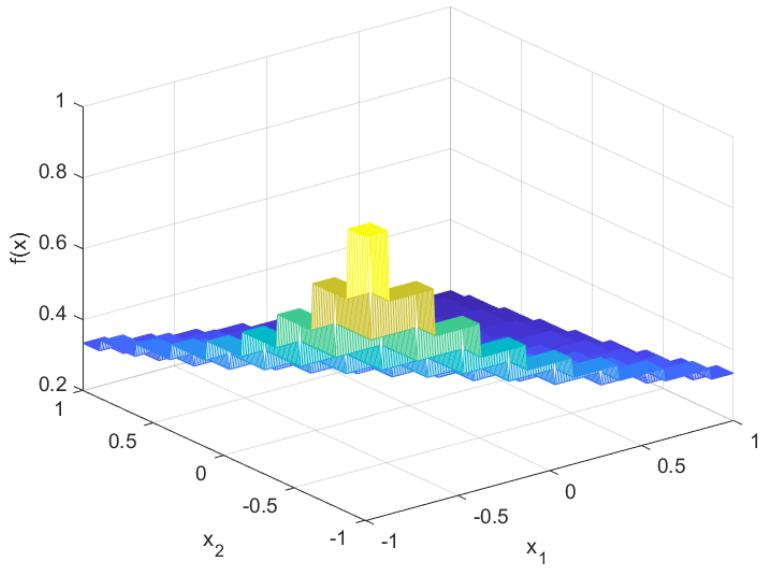


کران مرتبه دوم :

```
%% ?Second order limit (max)
alfa=-1;
beta=1;
h=0.25;
N=9;
x1=alfa:0.01:beta;
x2=x1;
[~,n1]=size(x1);
[~,n2]=size(x2);
e1=beta*ones(1,N+1);
e2=beta*ones(1,N+1);
for j=1:N
    e1(j)=alfa+h*(j-1);
    e2(j)=alfa+h*(j-1);
end
f_x=zeros(n1,n2);
for k1=1:n1
    for k2=1:n2


i1=min(find(e1<=x1(1,k1),1,'last'),find(e1>=x1(1,k1),1)
);

i2=min(find(e2<=x2(1,k2),1,'last'),find(e2>=x2(1,k2),1)
);

        if x1(1,k1)>=e1(1,i1)&&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1))&&
x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=.5*(e2(1,i2)+e2(1,1+i2))
            p=0;
            q=0;
        elseif x1(1,k1)>=e1(1,i1)&&
x1(1,k1)<=.5*(e1(1,i1)+e1(1,1+i1))&&
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))&&
x2(1,k2)<=e2(1,1+i2)
            p=0;
            q=1;
```

```
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1))&&
x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=e2(1,i2)&&
x2(1,k2)<=0.5*(e2(1,i2)+e2(1,1+i2))
            p=1;
            q=0;
        elseif x1(1,k1)>=.5*(e1(1,i1)+e1(1,1+i1))&&
x1(1,k1)<=e1(1,1+i1)&&
x2(1,k2)>=0.5*(e2(1,i2)+e2(1,1+i2))&&
x2(1,k2)<=e2(1,1+i2)
            p=1;
            q=1;
        end

        f_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));
    end
end
[x1,x2]=meshgrid(x1,x2);
figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,transpose(f_x));
xlabel('x_1')
ylabel('x_2')
zlabel('f(x)')
```

در نتیجه شکل نمودار سیستم فازی به صورت زیر خواهند شد :



سوال دوم :

یک برنامهٔ کامپیوتری برای پیاده‌سازی روش جدول جستجو بنویسید. برای کامل و همه‌منظوره‌بودن برنامه، می‌توانید روش پُرکردنِ خانه‌های خالی جدول جستجو را هم در آن در نظر بگیرید. برنامهٔ خود را برای مسألهٔ پیش‌گویی سری زمانی Mackey-Glass که در بخش ۳.۱۲ مرجع [۲] آورده‌شده را به کار گرفته و اجرا کنید. نتایج را به شکلی مناسب نشان دهید.

کد بخش برگرفته از مرجع ۱ :

```
clc;
clear;
close all;
%% Data generation by Mackey-Glass chaotic time series
n=900; % Total number of sampling
% Preallocations
x=zeros (1, n);
dataset_1=zeros (n, 7);
x(1,1:31)=1.3+0.2*rand;

for k=31:n-1
x (1, k+1)=0.2* ((x(1, k-30))/ (1+x (1, k-
30)^10))+0.9*x(1, k);
dataset_1 (k, 2:6)= [x(1, k-3) x(1, k-2) x(1, k-1) x(1,
k) x(1, k+1)];
end
dataset (1:600, 2:6)=dataset_1 (201: 800, 2:6);
t=1:600;

figure1 = figure ('Color', [1 1 1]); plot (t,x
(201:800), 'Linewidth', 2)
grid on
[Number_training, ~]=size (dataset);
Rul=zeros (Number_training/2,6);
Rules_total=zeros (Number_training/2, 6);
%% designing fuzzy system considering two cases:
% (assigning 7 membership functions for each input
variables)
% s=1 ;
% (assigning 15 membership functions for each input
variables)
```

```
% s=2 ;?

for s=1:2
    switch s
        case 1
        num_membership_functions=7; c=linspace (0.5,
1.3,5);
        h=0.2;

membership_functions=cell(num_membership_functions, 2);
        for k=1:num_membership_functions
            if k==1
                membership_functions {k, 1}= [0, 0,
0.3, 0.5];
                membership_functions {k, 2}='trapmf';
            elseif k==num_membership_functions
                membership_functions{k, 1}=[1.3, 1.5,
1.8, 1.8];
                membership_functions {k, 2}='trapmf';
            else
                membership_functions {k, 1}=[c(k-1)-h,
c(k-1), c(k-1)+h];
                membership_functions {k, 2}='trimf';
            end
        end
        case 2
        num_membership_functions=15;
        c=linspace(0.3,1.5, 13);
        h=0.1;

membership_functions=cell(num_membership_functions, 2);
        for k=1:num_membership_functions
            if k==1
                membership_functions{k, 1}=[0, 0, 0.2,
0.3];
                membership_functions{k, 2}='trapmf';
            elseif k==num_membership_functions
```

```
                membership_functions{k, 1}=[1.5, 1.6,
1.8, 1.8];
                membership_functions{k,2}='trapmf';
            else
                membership_functions{k, 1}=[c(k-1)-h,
c(k-1), c(k-1)+h];
                membership_functions{k,2}='trimf';
            end
        end
    end


    %% Assign degree to each rule
    vec_x=zeros (1, num_membership_functions);
    vec=zeros (1,5);
    for t=1: Number_training
        dataset(t, 1)=t;
            for i=2:6
                x=dataset(t, i);
                    for j=1:num_membership_functions
                        if j==1
                        vec_x (1, j) = trapmf(x,
membership_functions {1,1});
                        elseif
j==num_membership_functions
                        vec_x (1, j)=trapmf (x,
membership_functions{num_membership_functions, 1});
                        else
                        vec_x (1, j) = trimf (x,
membership_functions {j,1});
                        end
                    end
                [valu_x, column_x]=max(vec_x);
                vec (1, i-1)=max (vec_x);
                Rules(t, i-1)=column_x;
                Rules(t, 6) =prod(vec);
                dataset (t,7) =prod(vec);
            end
     end
%%  Delete extra rules
```

```
Rules_total(1, 1:6)=Rules(1,1:6);
i=1;
for t=2:Number_training
    m=zeros (1,1);
    for j=1:i
        m(1, j)=isequal(Rules(t, 1:4), Rules_total(j,
1:4));
        if m(1,j)==1 && Rules(t, 6)>=Rules_total (j,6)
            Rules_total(j, 1:6)=Rules (t, 1:6);
        end
    end
    if sum (m)==0
        Rules_total(i+1, 1:6)=Rules(t, 1:6);
        i=i+1;

    end
end



%%
disp('******************************')
disp(['Final rules for ',
num2str(num_membership_functions),' membership
functions for each input variables'])
final_Rules=Rules_total(1:1, :);
%% Create Fuzzy Inference System
Fisname='Prediction controller';
Fistype='mamdani';
Andmethod='prod';
Ormethod='max';
Impmethod='prod';
Aggmethod='max';
Defuzzmethod='centroid';
fis=newfis(Fisname, Fistype, Andmethod, Ormethod,
Impmethod, Aggmethod, Defuzzmethod);
%% Add Variables
for num_input = 1:4
    fis = addInput(fis, [0.1 1.7], "Name", ['x',
num2str(num_input)]);
end
```

```
fis = addOutput(fis,[0.1, 1.7],  'Name', 'x5');
%% Add Membership functions
for num_input = 1:4
    for input_Rul = 1:num_membership_functions
        fis = addMF(fis, ['x', num2str(num_input)],
membership_functions{input_Rul,2},membership_functions{
input_Rul,1}, 'Name', ['A', num2str(input_Rul)]);
    end
end
for input_Rul = 1:num_membership_functions
    fis = addMF(fis,
'x5',membership_functions{input_Rul, 2},
membership_functions{input_Rul, 1}, 'Name', ['MF_',
num2str(input_Rul)]);
end
%% Add Rules
non_zero_rows = any(Rules_total(:, 1:5), 2);  % Find
rows with non-zero rules
fis_Rules = ones(sum(non_zero_rows), 7);
fis_Rules(:, 1:6) = Rules_total(non_zero_rows, 1:6);
fis = addrule(fis, fis_Rules);
%% Prediction of 300 points of chosen dataset
jadval_prediction=zeros(300,2);
f=1;
for i=301:600
    input=dataset(i, 2:6);
    output1=dataset(i, 6);
    x5=evalfis([input(1, 1); input(1, 2); input(1,3);
input(1,4)], fis);
    jadval_prediction(f, :)= [f, x5];
    f=f+1;
end
figure;
plot(jadval_prediction(:,1),jadval_prediction(:,2), 'r-
.', 'Linewidth', 2);
hold on;
grid on
plot(jadval_prediction(:,1),dataset(301: 600, 6), 'b',
'Linewidth', 2);
legend('estimate value', 'real value')
```

```
grid on
end
% Assuming 'fis' is your fuzzy inference system
inputVariableIndex = 1;  % Change this to the index of
the input variable you're interested in

% Plot the membership functions for the specified input
variable
figure;
plotmf(fis, 'output', inputVariableIndex);
grid on
title(['Membership Functions for Input Variable ',
num2str(inputVariableIndex)]);
```

توضیح روند عملکرد :

۱-تولید داده

یک سری زمانی به نام Mackey-Glass تولید می‌شود که به عنوان یک سری زمانی غیرخطی مشهور است. این سری زمانی به صورت بازخوردی از یک فرمول مشخص به دست می‌آید.

۲-طراحی سیستم منطق فازی

دو حالت برای تعریف توابع عضویت برای ورودی‌ها در نظر گرفته شده است:

- حالت اول با ۷ تابع عضویت برای هر ورودی.
- حالت دوم با ۱۵ تابع عضویت برای هر ورودی.

پارامترهای توابع عضویت به صورت دستی تعریف شده‌اند.

۳-تخصیص درجه به هر قانون

برای هر نمونه در مجموعه آموزش، درجه عضویت در هر تابع عضویت محاسبه می‌شود. قانونی که بیشترین درجه عضویت را دارد، به عنوان قانون فعال انتخاب می‌شود.

۴- حذف قوانین اضافی

قوانین اضافی حذف می‌شوند تا سیستم فازی ساده‌تر شود.

۵-ایجاد سیستم منطق فازی

یک سیستم منطق فازی Mamdani ایجاد می‌شود با استفاده از قوانین فازی محاسبه شده. ورودی‌ها و خروجی‌های سیستم منطق فازی تعریف می‌شوند. توابع عضویت برای ورودی‌ها و خروجی‌ها تعریف می‌شوند.

۶-پیش‌بینی مقادیر

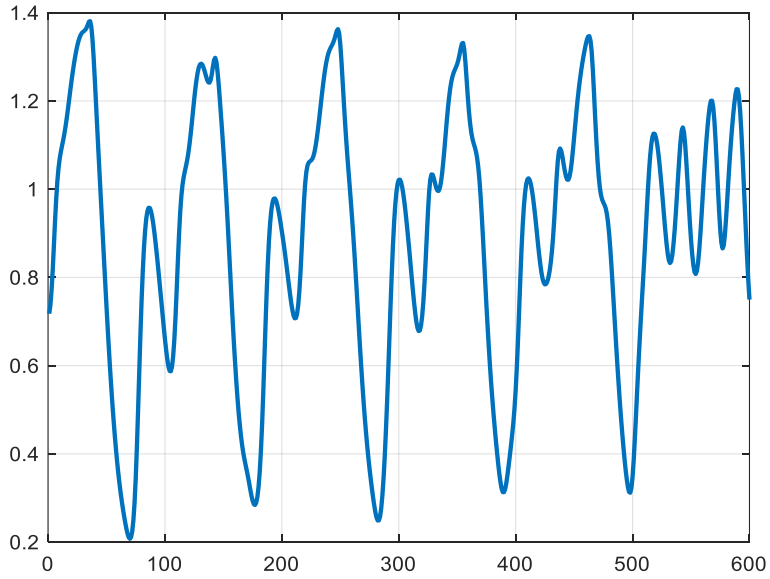
با استفاده از سیستم منطق فازی ایجاد شده، ۳۰۰ نقطه آینده از مجموعه داده پیش‌بینی می‌شوند. نتایج پیش‌بینی با نتایج واقعی مقایسه شده و در یک نمودار نمایش داده می‌شوند.

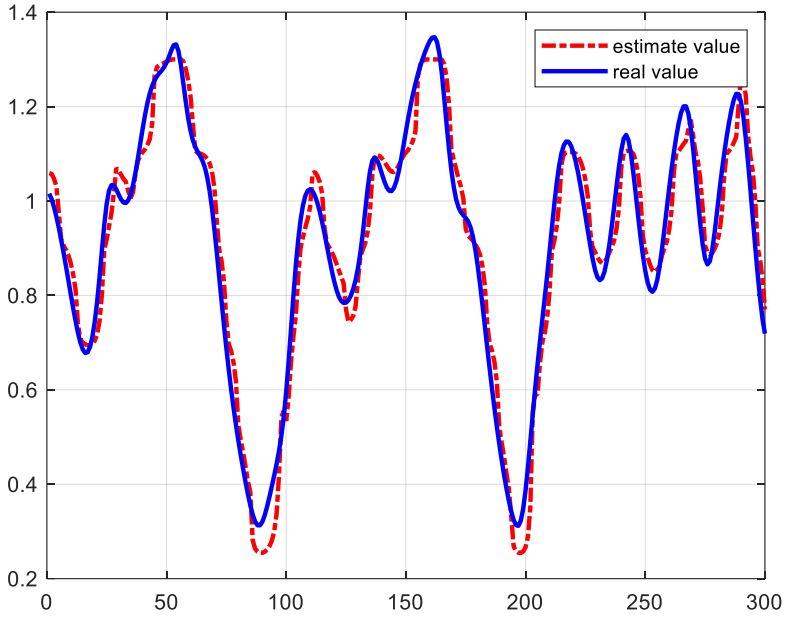
۷- نمایش توابع عضویت

این بخش به شما امکان نمایش توابع عضویت برای یکی از ورودی‌ها را می‌دهد. این توابع عضویت تعیین می‌کنند که ورودی‌ها به چه اندازه در هر مرحله به هر قاعده فازی تعلق دارند.

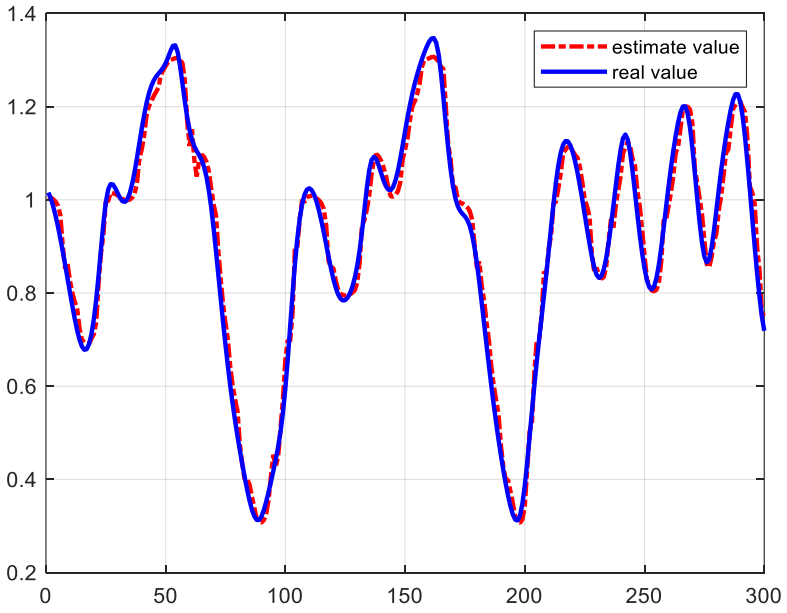
به کلی، کد یک سیستم منطق فازی را ایجاد می‌کند که از آن برای پیش‌بینی مقادیر یک سری زمانی بهره می‌برد.

شکل نتایج :

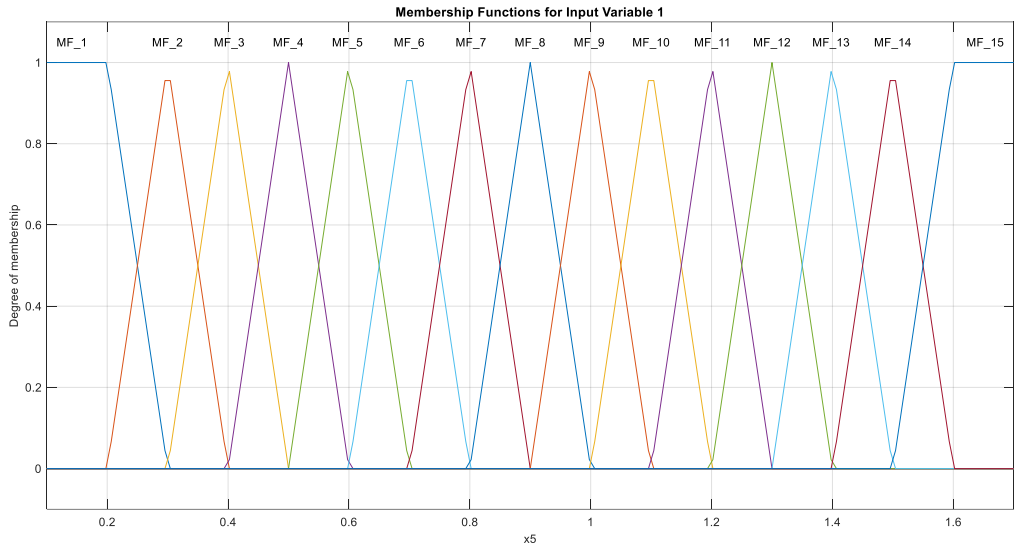




شکل ۱) پیش بینی مدل سری زمانی



شکل ۲) پیش بینی دقیق تر مدل سری زمانی



سوال سوم :

فرض کنید یک سیستم با معادلهٔ دیفرانسیل آورده‌شده در معادله ۱ دارید که قرار است توسط یک شناساگر فازی شناسایی شود.

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \qquad (۱)$$

که در آن تابع نامعلوم $g[u(k)]$ براساس معادله ۲ تعریف می‌شود.

$$g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \qquad (۲)$$

هدف ما این است که عنصر غیرخطی نامعلوم $g[u(k)]$ در معادله ۱ را توسط سیستمی فازی با رابطهٔ معادله ۳ و به‌همراه الگوریتم آموزش گرادیان نزولی (مثلاً روابط (۵.۱۳)، (۸.۱۳) و (۹.۱۳) در مرجع [۲]) تقریب بزنیم. با طراحی و برنامه‌نویسی مناسب این کار را انجام دهید.

$$f(x) = \frac{\sum_{l=1}^{M} \bar{y}^l \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right)\right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right)\right]} \qquad (۳)$$

کد بخش برگرفته از مرجع ۲ :

```
clc;
clear;
close all;
%% Initializing
M=4; %Number of membership functions (Based on 1st step
of fuzzy system design)
```

```
num_training=200; % Number of training
total_num=700;
landa=0.1; % A constant stepsize

% Preallocation
x_bar=zeros (num_training, M);
g_bar=zeros (num_training, M);
sigma=zeros (num_training, M);
y=zeros(total_num, 1);
u=zeros(total_num, 1);
x=zeros(total_num, 1);
y_hat=zeros(total_num, 1);
f_hat=zeros(total_num, 1);
z=zeros(total_num, 1);
g_u=zeros(total_num, 1);

u(1)=-1+2*rand;
y(1)=0;
g_u(1)=0.6*sin(pi*u(1))+0.3*sin(3*pi*u(1))+0.1*sin(5*pi
*u(1));
f_hat(1)=g_u(1);


%% Based on the 1st step of fuzzy system design
u_min=-1;
u_max=1;
h=(u_max-u_min)/(M-1);
for k=1:M
    x_bar(1, k)=-1+h*(k-1);
    u(1,k) =x_bar(1, k);

g_bar(1,k)=0.6*sin(pi*u(1,k))+0.3*sin(3*pi*u(1,k))+0.1*
sin(5*pi*u(1,k));
end

sigma(1,1:M) = (max(u(1,:))-min(u(1,:)))/M;

x_bar(2,:)=x_bar(1, :);
g_bar(2,:)=g_bar(1, :);
sigma(2, :)=sigma(1,:);
x_bar_initial=x_bar(1, :);
```

```
sigma_initial=sigma(1, :);
y_bar_initial=g_bar(1,:);

%% Based on the 2nd and 3rd step of fuzzy system design
for q=2: num_training
for q=2:num_training
    b=0;a=0;
    x(q)=-1+2*rand;
    u(q)=x(q);

g_u(q)=0.6*sin(pi*u(q))+0.3*sin(3*pi*u(q))+0.1*sin(5*pi
*u(q));

    for l=1:M
        z(l)=exp(-((x(q)-x_bar(q,l))/sigma(q, l))^2);
        b=b+z(l);
        a=a+g_bar(q, l)*z(l);
    end

    f_hat (q)=a/b;
    y(q+1)=0.3*y(q)+0.6*y(q-1)+g_u(q);
    y_hat(q+1)=0.3*y(q)+0.6*y(q-1)+f_hat(q);

    for l=1:M
        g_bar(q+1,l)=g_bar(q,l)-landa*(f_hat(q)-
g_u(q))*z(l)/b;
        x_bar(q+1,l)=x_bar(q,l)-landa*((f_hat(q)-
g_u(q))/b)*(g_bar(q,l)-f_hat(q))*z(l)*2*(x(q)-
x_bar(q,l))/(sigma(q,l)^2);
        sigma (q+1,l)=sigma(q, l)-landa*((f_hat(q)-
g_u(q))/b)*(g_bar(q,l)-f_hat(q))*z(l)*2*(x(l)-
x_bar(q,l))^2/(sigma(q,l)^3);
    end
end

x_bar_final=x_bar(num_training,:);
sigma_final=sigma(num_training,:);
g_bar_final=g_bar(num_training,:);

for q=num_training:700
    b=0;
```

```
    a=0;
    x(q)=sin(2*q*pi/200);
    u(q)=x(q);


g_u(q)=0.6*sin(pi*u(q))+0.3*sin(3*pi*u(q))+0.1*sin(5*pi
*u(q));

    for l=1: M
        z(l)=exp(-((x(q)-
x_bar(num_training,l))/sigma(num_training, l))^2);
        b=b+z(l);
        a=a+g_bar(num_training, l)*z(l);
    end
     f_hat(q)=a/b;
     y(q+1)=0.3*y(q)+0.6*y(q-1)+g_u(q);
     y_hat(q+1)=0.3*y(q)+0.6*y(q-1)+f_hat(q);
end

%% Plots and Figures
figure1=figure('Color', [1 1 1]);
plot(1:701, y, 'b', 1:701, y_hat, 'r:', 'Linewidth',
2);
legend('output of the plant', 'output of the
identification model')
axis([0 701 -5 5]);
grid on

figure2=figure('Color', [1 1 1]);
xp=-2:0.001:2;
for l=1:M
    miu_x=exp(-((xp-x_bar(1, l))./(sigma (1,l))).^2);
    plot(xp, miu_x, 'Linewidth', 2);
    hold on
end

xlabel('u');
ylabel('initial MF''s');
axis([-1 1 0 1]);

figure3=figure('Color', [1 1 1]);
```

```
for l=1:M
    miu_x=exp(-((xp-x_bar(num_training, l))./ (sigma
(num_training, l))).^2);
    plot (xp, miu_x, 'Linewidth', 2);
    hold on
end

xlabel('u');
ylabel('final MF''s');
axis ([-1 1 0 1]);
```

این کد یک مدل تطبیقی Adaptive Model برای تخمین خروجی یک سیستم پویا ایجاد می‌کند. در ادامه توضیحات بیشتری در مورد هر بخش از کد آورده شده است:

۱- متغیرها و پارامترها

M :تعداد توابع عضویت مربوط به سیستم فازی.

num_training :تعداد نمونه‌های استفاده شده برای آموزش مدل.

total_num: تعداد کل نمونه‌ها (آموزش و تست).

landa: مقدار ثابت گام آموزش (learning rate) آموزشی که مبتنی بر گرادیان نزولی می باشد.

۲-پیش‌پردازش و مقداردهی اولیه

متغیرها و آرایه‌ها برای ذخیره داده‌ها و پارامترهای مدل ایجاد می‌شوند.

مقادیر اولیه برای ورودی‌ها و خروجی‌ها تعیین می‌شوند.

۳-مقداردهی اولیه بر اساس توابع عضویت

مقادیر اولیه برای توابع عضویت ورودی‌ها براساس توزیع یکسان در بازه [-۱،۱] محاسبه می‌شوند.

این مقادیر به عنوان نقاط میانی اولیه برای توابع عضویت ورودی‌ها استفاده می‌شوند.

۴- آموزش مدل

از الگوریتم تطبیقی برای به‌روزرسانی توابع عضویت ورودی‌ها و سایر پارامترها بر اساس نمونه‌های آموزش استفاده می‌شود. توابع عضویت، میانگین خروجی مدل و ویژگی‌های مرتبط با توابع عضویت به‌روزرسانی می‌شوند.

۵-آزمون مدل

مدل بر روی نمونه‌های آزمون (بعد از آموزش) اجرا می‌شود و خروجی تخمین زده شده به دست می‌آید. خروجی مدل به همراه خروجی واقعی سیستم در یک نمودار نمایش داده می‌شود.

۶- نمودار توابع عضویت اولیه

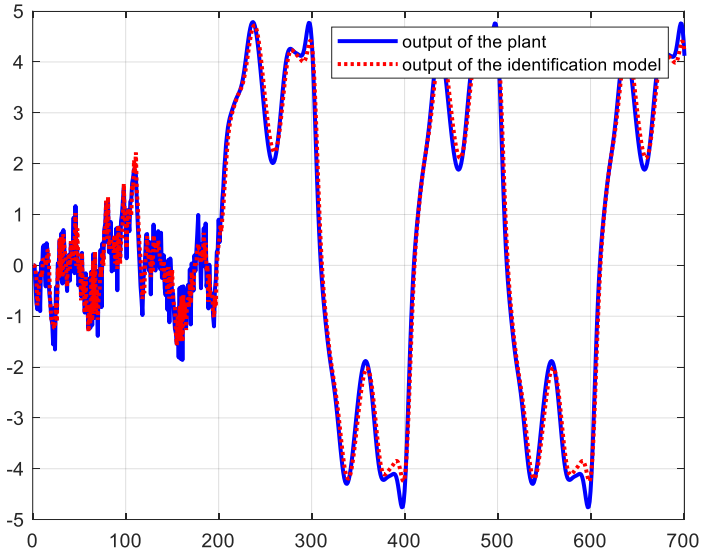
توابع عضویت اولیه برای ورودی‌ها در یک نمودار نمایش داده می‌شوند.

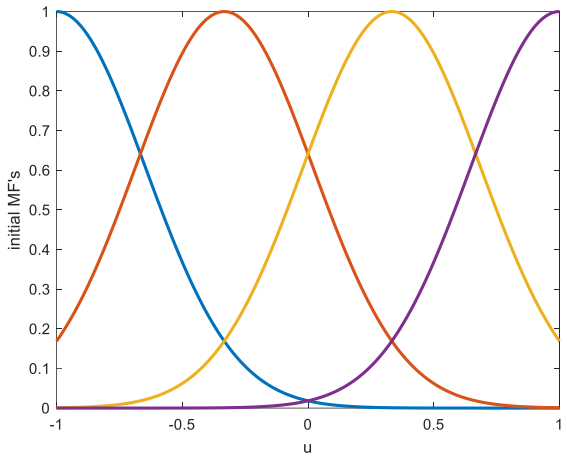
۷- نمودار توابع عضویت نهایی

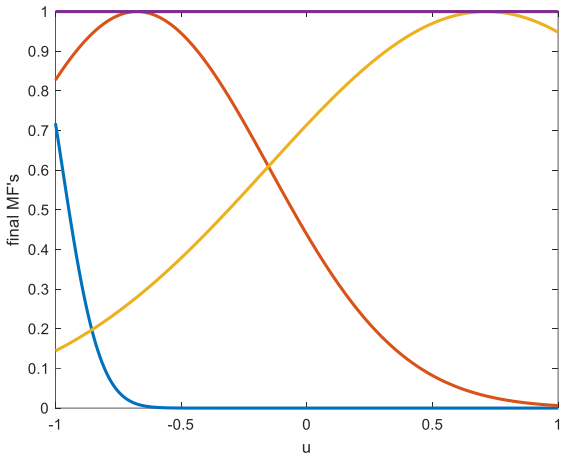
توابع عضویت نهایی برای ورودی‌ها در یک نمودار دیگر نمایش داده می‌شوند.

در کل، این کد یک مدل تطبیقی را با استفاده از توابع عضویت فازی ایجاد می‌کند که خروجی یک سیستم پویا را تخمین می‌زند.

نمودار نتایج :







سوال چهارم :

به سوالات زیر از مبحث درخت تصمیم پاسخ دهید:

۱. با بهره‌گیری از آموزش ارائه‌شده در خصوص کدنویسی درخت تصمیم از ابتدا[۱]، بدون استفاده از کتابخانهٔ سایکیت‌لرن دستوراتی بنویسید که درخت تصمیم یک مجموعه‌دادهٔ مربوط به بیماری کرونا که در این پیوند موجود است را خروجی دهد. اگر می‌توانید این کار را به‌صورتی انجام دهید که اطلاعات بیش‌تری را در خروجی درخت تصمیم خود دریافت کنید. لازم است که تحلیل منطقی از نتیجهٔ درخت تصمیم خود ارائه کنید. می‌توانید این کار را با الگوگرفتن از موارد گفته‌شده در ویدویوهای کلاس و این پیوند انجام دهید.

۲. به انتخاب خود یکی از دو مجموعه‌دادهٔ load_breast_cancer و Drugs[۲] را انتخاب کنید و کار طبقه‌بندی با درخت تصمیم را با استفاده از دستوراتی که آموزش دیده‌اید (کدنویسی از ابتدا و یا کدنویسی با کمک کتابخانهٔ سایکیت‌لرن) انجام دهید. لازم است که توضیحات مختصری از مجموعه‌داده و منطق درخت تصمیم تولیدشده بنویسید. منطق معیاری که استفاده می‌کنید و نتایج آن در قسمت‌های مختلف را به‌صورت کامل تحلیل کنید. هم‌چنین، مسیر مربوط به دو نمونه از داده‌های مجموعهٔ آزمون را نشان داده و تحلیل کنید. اگر از فراپارامتر خاصی مانند فراپارامترهای مخصوص هرس‌کردن استفاده می‌کنید لازم است که حداقل دو مقدار بزرگ و کوچک برای آن در نظر بگیرید و تحلیل خود از تأثیر آن روی نتیجهٔ نهایی را بنویسید.

۳. سوال اختیاری: مجموعه‌دادهٔ مربوط به «میزان امید به زندگی» که در این پیوند آورده شده را فراخوانی کنید و توضیحاتی در مورد آن بنویسید. در ادامه، از دستورات مربوط به درخت تصمیم استفاده کنید و نشان دهید که با تنظیم مناسب پارامترها می‌توان پیش‌بینی مربوط به این دیتاست را روی یک مجموعهٔ آزمون به خوبی انجام داد.

درخت تصمیم یک الگوریتم یادگیری ماشین است که بر اساس یک سری از تصمیم‌ها و شرایط، داده‌ها را به گروه‌ها یا دسته‌های مختلف تقسیم می‌کند. در اینجا یک توضیح کوتاه در مورد اجزای و عملکرد اصلی درخت تصمیم آورده شده است:

۱-گره‌ها

- درخت تصمیم از گره‌های مختلف تشکیل شده است که هر گره به یک سوال یا یک شرط مربوط است.

- دو نوع گره وجود دارد: گره‌های داخلی و گره‌های برگ

- گره‌های داخلی شرایطی را بررسی می‌کنند و بر اساس پاسخ به آن شرط، به یکی از زیرمجموعه‌ها هدایت می‌شوند.

۲- شاخه‌ها

- شاخه‌ها اتصال بین گره‌ها را نشان می‌دهند و نشان‌دهنده گذر از یک گره به گره دیگر است.

- هر شاخه با یک شرط از گره قبلی مرتبط است.

۳- ریشه

- گرهی به نام ریشه وجود دارد که از آن تمام درخت شروع می‌شود.

- ریشه به سوالی مرتبط با شرایط اولیه داده‌ها می‌پردازد.

۴- گره‌های برگ

- گره‌های برگ پیش‌بینی یا دسته‌بندی نهایی را انجام می‌دهند.

- در آنها تصمیم‌ها بر اساس شرایط ایجاد شده در گره‌های داخلی گرفته می‌شود.

۵- شرایط و سوالات

- هر گره داخلی یک شرط یا سوال مرتبط با داده‌ها دارد.

- مثلاً "آیا مقدار ویژگی X بزرگتر از یک حد مشخص است؟".

۶- آموزش

- مدل درخت تصمیم با استفاده از مجموعه‌ی آموزشی آموزش می‌بینند.

- هدف این است که با تقسیم‌بندی داده‌ها در هر گره، درخت تصمیم به بهترین نحو ممکن دسته‌ها را تفکیک کند.

۷- پیش‌بینی

- برای هر نمونه جدید، از درخت تصمیم برای پیش‌بینی دسته‌ای که نمونه به آن تعلق دارد، استفاده می‌شود.

- نمونه از ریشه تا گره‌های برگ پیش‌بینی می‌شود.

۸- افزایش تفسیرپذیری

- درخت تصمیم می‌تواند به دلیل ساختار خود، تفسیرپذیرتر از برخی از مدل‌های مخفی‌لایه‌ای مانند شبکه‌های عصبی باشد.

- با تحلیل شاخص‌ها و شرایط درخت، تصمیم‌گیری مدل قابل فهم‌تر می‌شود.

سوال ۱)

برای حل این سوال می توانیم به صورت کاملا مجزا و با استفاده از روش های گفته شده مانند آنتروپی ، گین هر کدام از ویژگی ها را بدست بیاوریم و با مقایسه یکدیگر ویژگی ریشه ای و قسمت های مختلف درخت را مشخص کنیم. اما در این قسمت ما از کد غیر آماده استفاده می کنیم و به صورت زیر عمل می کنیم سپس تحلیل آن را می نویسیم :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from graphviz import Digraph
!pip install --upgrade --no-cache-dir gdown
!gdown 1UCDlb6gatarImiHiLnrDKDVrUqLQq6oW
data = pd.read_csv('/content/covid.csv')
data
labels = data['Infected']
len(labels), labels.unique(), labels.value_counts()
p = labels.value_counts() / len(labels)
-sum(p * np.log2(p))
def entropy(labels):
    p = labels.value_counts() / len(labels)
    return -sum(p * np.log2(p))

data['Infected'].value_counts()
 entropy_child = 0
 for value in data['Cough'].unique():
```

```
        subset = data[data['Cough'] == value]
        print(subset)
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset['Infected'])
entropy_child
def entropy(labels):
    p = labels.value_counts() / len(labels)
    return -sum(p * np.log2(p))
target = 'Infected'
entropy_parent = entropy(data[target])
entropy_parent

entropy_child = 0
feature = 'Fever'
for value in data[feature].unique():
    subset = data[data[feature] == value]
    display(subset)
    wi = len(subset) / len(data)
    entropy_child += wi * entropy(subset[target])
information_gain = entropy_parent - entropy_child

print(information_gain)
def information_gain(data, feature, target):
    # Entropy of parent
    entropy_parent = entropy(data[target])

    # Entropy of child
    entropy_child = 0
    for value in data[feature].unique():
        subset = data[data[feature] == value]
        #display(subset)
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset[target])

    return entropy_parent - entropy_child

arg=[information_gain(data, feature, 'Infected') for feature in
data.iloc[:, :-1].columns]
def information_gain(data, feature, target):
    # Entropy of parent
    entropy_parent = entropy(data[target])

    # Entropy of child
    entropy_child = 0
    for value in data[feature].unique():
```

```
        subset = data[data[feature] == value]
        wi = len(subset) / len(data)
        entropy_child += wi * entropy(subset[target])

    return entropy_parent - entropy_child
```

```
[36] information_gain(data, 'Fever', 'Infected')

     0.12808527889139443
```

```
[37] information_gain(data, 'Cough', 'Infected')

     0.0391486719030707
```

```
[38] information_gain(data, 'Breathing issues', 'Infected')

     0.39603884492804464
```

```
[39] data.iloc[:, :-1].columns

     Index(['Fever', 'Cough', 'Breathing issues'], dtype='object')
```

```
[40] [information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns]

     [0.12808527889139443, 0.0391486719030707, 0.39603884492804464]
```

```
[41] np.argmax([information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns])

     2
```

تحلیل کد :

در کد بالا پس از ایمپورت کردن اطلاعات در کولب ، داده ها را تقسیم بندی کرده ایم و سپس با استفاده از فرمول آنتروپی ، مقدار گین ویژگی های مختلف را برای مشخص کردن root node بدست آورده ایم. با توجه به کد بالا می بینیم که ویژگی ستون ۲ یعنی ویژگی مشکل تنفسی دارای بیشترین گین می باشد. پس به سراغ تشکیل درخت تصمیم با استفاده از تعریف تابع می رویم :

```
class Node:
  def __init__(self, feature=None, label=None):
    self.feature = feature
    self.label = label
    self.children = {}
  def __repr__(self):
    if self.feature is not None:
      return f'DecisionNode(feature="{self.feature}",
children={self.children})'
    else:
```

```
      return f'LeafNode(label="{self.label}")'
def make_tree(data, target):
  # leaf node?
  if len(data[target].unique()) == 1:
    return Node(label=data[target].iloc[0])
  features = data.drop(target, axis=1).columns
  if len(features) == 0 or len(data) == 0:
    return Node(label=data[target].mode()[0])
  # calculate information gain
  gains = [information_gain(data, feature, target) for feature in
features]
  # greedy search to find best fearure
  max_gains_idx = np.argmax(gains)
  best_features = features[max_gains_idx]
  # make a node
  node = Node(feature=best_features)
  # loop over the best feature
  for value in data[best_features].unique():
    subset = data[data[best_features] == value].drop(best_features,
axis=1)
    # display(subset)
    node.children[value] = make_tree(subset, target)
  return node
```

این کد یک کلاس به نام Node ایجاد می‌کند که از آن برای ساختار داده درخت تصمیم استفاده می‌شود. سپس یک تابع به نام make_tree نیز تعریف شده است که از این کلاس Node برای ساخت درخت تصمیم با توجه به اطلاعات گنجانده شده (Information Gain)در هر ویژگی استفاده می‌کند.

تحلیل کوتاه کد:

۱. کلاس Node

- این کلاس دارای ویژگی‌های feature برای نشان دادن ویژگی در گره,label برای نشان دادن برچسب در گره برگ) و children برای نشان دادن زیردرخت‌های گره) است.

- تابع repr برای نمایش متنی مناسب گره‌ها است.

۲. تابع make_tree

- این تابع یک درخت تصمیم را با استفاده از رویکرد بازگشتی می‌سازد.

- ابتدا چک می‌شود که آیا همه نمونه‌ها در یک دسته‌بندی هستند یا نه. اگر بله، یک گره برگ با برچسب دسته‌بندی ایجاد می‌شود.

- سپس لیست ویژگی‌ها چک می‌شود. اگر هیچ ویژگی‌ای باقی نمانده یا تعداد نمونه‌ها صفر باشد، یک گره برگ با برچسبی برابر با حالت رایج تارگت ایجاد می‌شود.

- اگر موارد بالا نقصانی ایجاد نکنند، اطلاعات گنجانده شده (Information Gain)برای هر ویژگی محاسبه می‌شود.

- با استفاده از یک رویکرد حریصانه (greedy)، ویژگی با بیشترین اطلاعات گنجانده شده انتخاب می‌شود.

- یک گره جدید با این ویژگی به عنوان ویژگی گره ایجاد می‌شود و برای هر مقدار مختلف ویژگی، یک زیردرخت تصمیم بازگشتی ساخته می‌شود.

```
tree = make_tree(data, 'Infected')
tree
```

```
DecisionNode(feature="Breathing issues", children={'No': DecisionNode(feature="Fever", children={'No': LeafNode(label="No"), 'Yes':
DecisionNode(feature="Cough", children={'Yes': LeafNode(label="No")})}), 'Yes': DecisionNode(feature="Fever", children={'Yes':
LeafNode(label="Yes"), 'No': DecisionNode(feature="Cough", children={'Yes': LeafNode(label="Yes")})})})
```

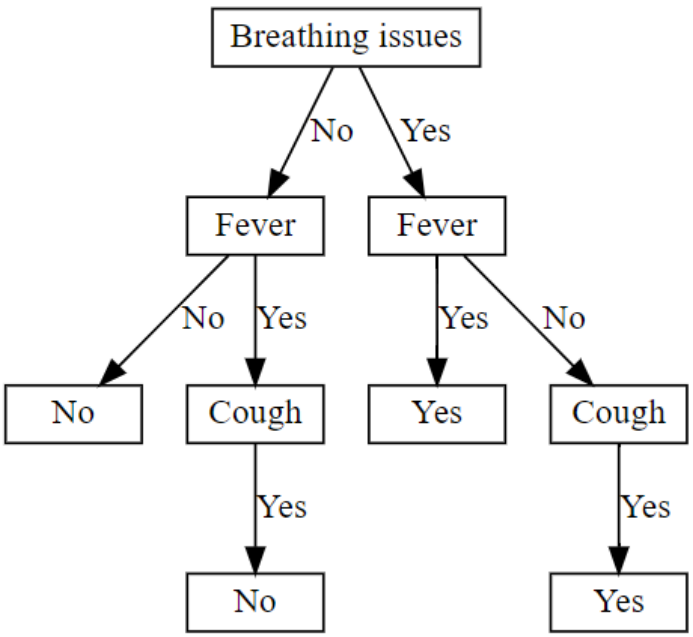
مشکل درون دیتاست می باشد. وقتی فیچر breating_issue به عنوان گره روت انتخاب می‌شه، برای مقادیر فیچر Yes و No هر دو Fever بیش‌ترین مقدار information gain رو دارد. همینجوری نظری هم به دیتاست نگاه کنیم، مشخص هست که فیچر Cough نقشی در ماجرا ندارد.

در نهایت نمودار درختی مربوطه را رسم می کنیم :

```
def visualize_tree(tree, parent=None, node_id=None):
    if node_id is None:
        node_id = '0'
        g = Digraph(node_attr={'shape': 'record', 'height':'.1'})
        g.node(node_id, label=tree.feature)
    else:
        g = parent
        g.node(node_id, label=tree.feature)
    if len(tree.children) == 0:
        g.node(node_id, label=tree.label)
        return g
    for i, (value, child) in enumerate(tree.children.items()):
        child_id = f'{node_id}_{i+1}'
        visualize_tree(child, g, child_id)
```

```
        g.edge(node_id, child_id, label=value)
    return g
g = visualize_tree(tree)
g.render('decision_tree', format='png', view=True)
```

```
visualize_tree(tree)
```



مطابق با جدول بالا وقتی فیچر breating_issue به عنوان گره روت انتخاب می‌شه، برای مقادیر فیچر Yes و No هر دو Fever بیش‌ترین مقدار information gain رو داره. بنابراین چه در صورت yes و در صورت no بودن به ویژگی fever رجوع می کنیم. در صورت no بودن breating_issue و yes بودن fever ویژگی Cough بررسی می شود و در این وضعیت در صورت noب بودن fever شخص کرونایی نیست. از طرفی دیگر نیز در صورت yes بودن breating_issue و fever شخص قطعا دارای کرونا می باشد. در غیر این صورت ویژگی cough بررسی می شود و در صورت yes بودن آن شخص کرونایی تشخیص داده می شود. اما در حالتی که breating_issue ، no باشد ولی fever و cough در این صورت yes باشند، دلیلی بر تشخیص بیماری نمی باشد.

DecisionNode(feature="Breathing issues", children={'No': DecisionNode(feature="Fever", children={'No': LeafNode(label="No"), 'Yes': DecisionNode(feature="Cough", children={'Yes': LeafNode(label="No")})}), 'Yes': DecisionNode(feature="Fever", children={'Yes': LeafNode(label="Yes"), 'No': DecisionNode(feature="Cough", children={'Yes': LeafNode(label="Yes")})})})

سوال ۲)

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
# Example with max_depth as a pruning parameter
max_depth_values = [5 ,10]  # Replace with your desired values
for max_depth in max_depth_values:
    clf = DecisionTreeClassifier(max_depth=max_depth)

    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - Max Depth: {max_depth}')
    plt.savefig(f'decision_tree_max_depth_{max_depth}.png')
    plt.show()
```

این کد برای ایجاد و نمایش درخت تصمیم بر روی داده‌های سرطان پستان استفاده می‌شود. داده‌ها به دو بخش آموزش و آزمون تقسیم می‌شوند. سپس یک مدل درخت تصمیم با عمق‌های مختلف ایجاد می‌شود و درخت تصمیم برای هر عمق با استفاده از تابع plot_tree نمایش داده می‌شود. اینکار به توجه به اطلاعات گنجانده شده در درخت تصمیم و نحوه تصمیم‌گیری در هر گره کمک می‌کند.

۱. مدل با عمق ۵

- این درخت تصمیم با عمق ۵ به صورت گسترده‌تر و کلی‌تر اطلاعات را در اختیار می‌گیرد. این می‌تواند به خاطر کمتر بودن عمق، از برخی اطلاعات خاص و ارتباطات محلی چشم‌پوشی کند.

- با توجه به مقدار عمق ۵، درخت احتمالاً به دنبال اطلاعات مهم و کلان در مورد سرطان پستان است.

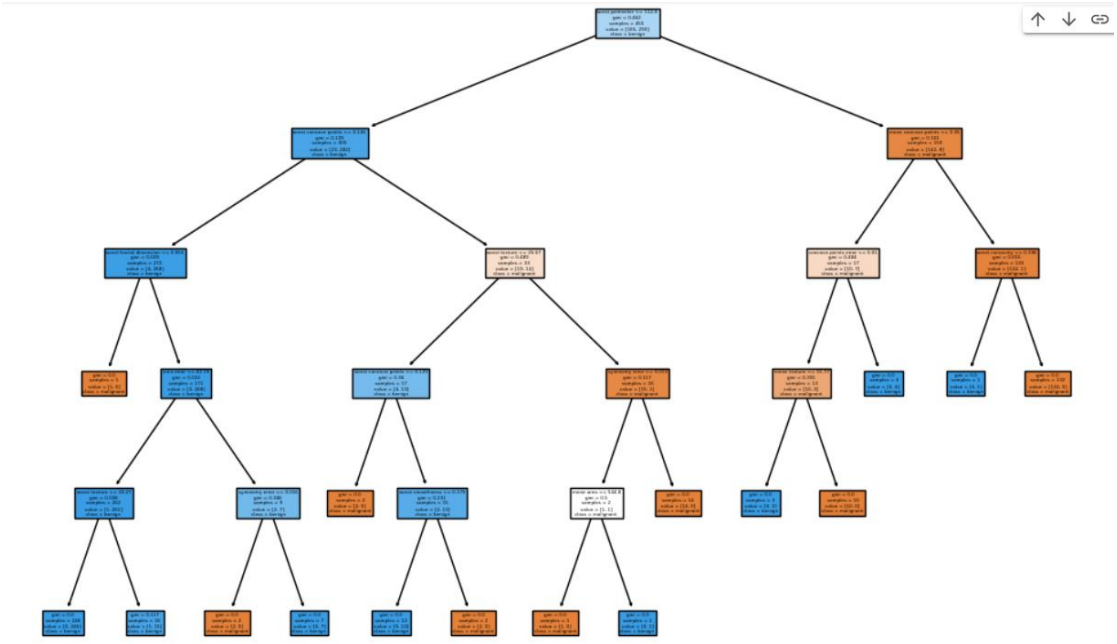
۲. مدل با عمق ۱۰

- این درخت با عمق بیشتر (۱۰)، اطلاعات دقیق‌تر و خاص‌تری را در اختیار می‌گیرد. این ممکن است به دلیل این باشد که در این حالت، درخت قادر به درک اطلاعات محلی و تفاوت‌های کوچک‌تر در داده‌ها می‌شود.

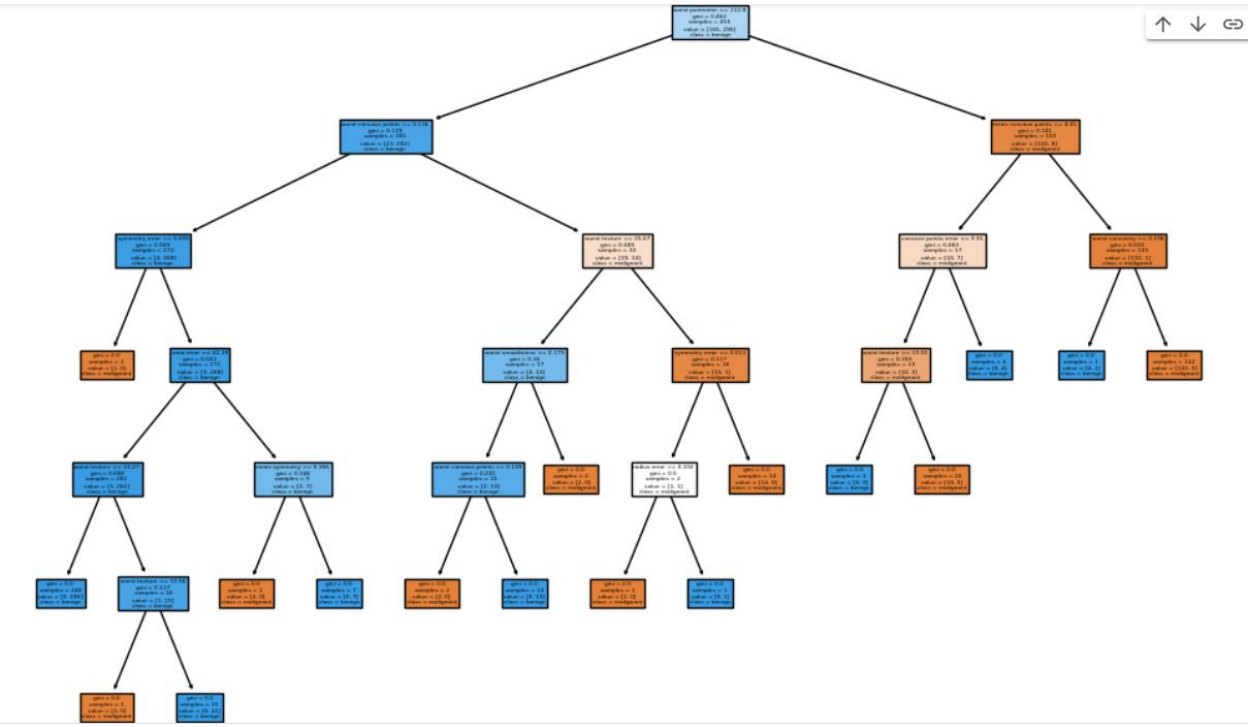
- این عمق بیشتر ممکن است منجر به یادگیری و حفظ جزئیات کمتری در مورد داده‌ها شود و ممکن است باعث افزایش دقت در دسته‌بندی شود.

نمایش نمودار ها :

درخت با عمق ۵ :



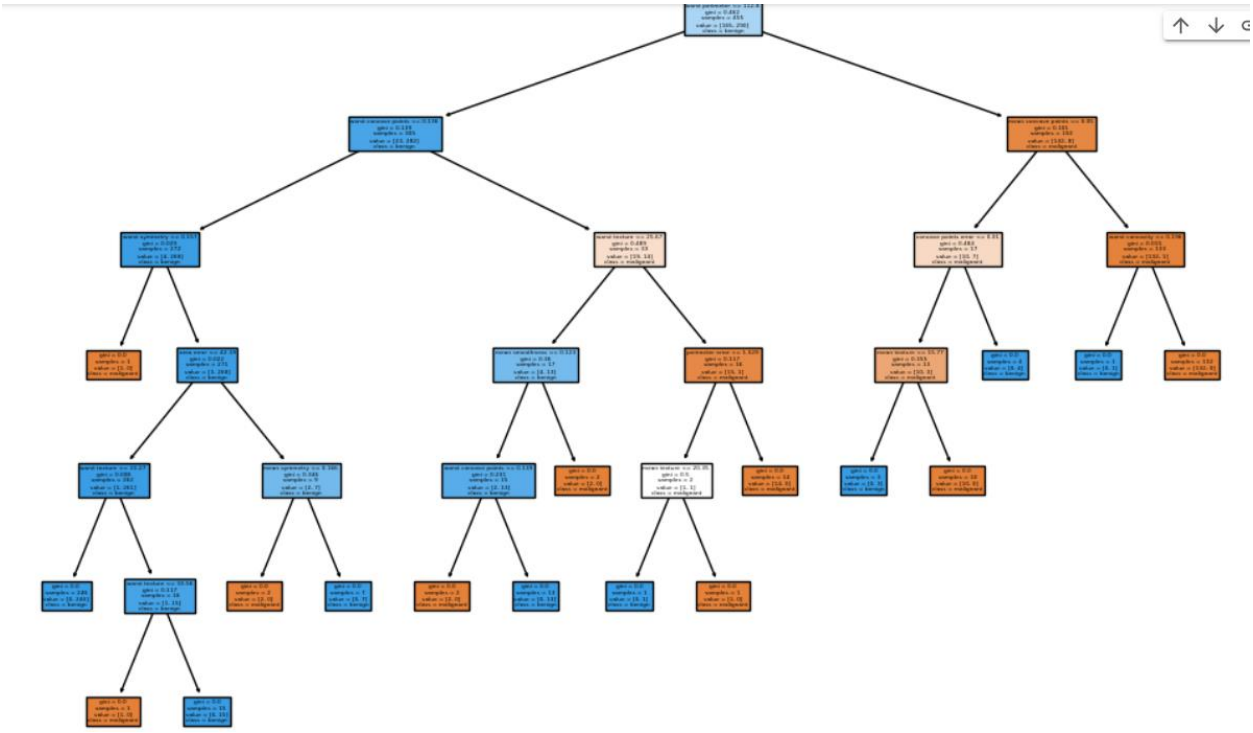
درخت با عمق ۱۰ :



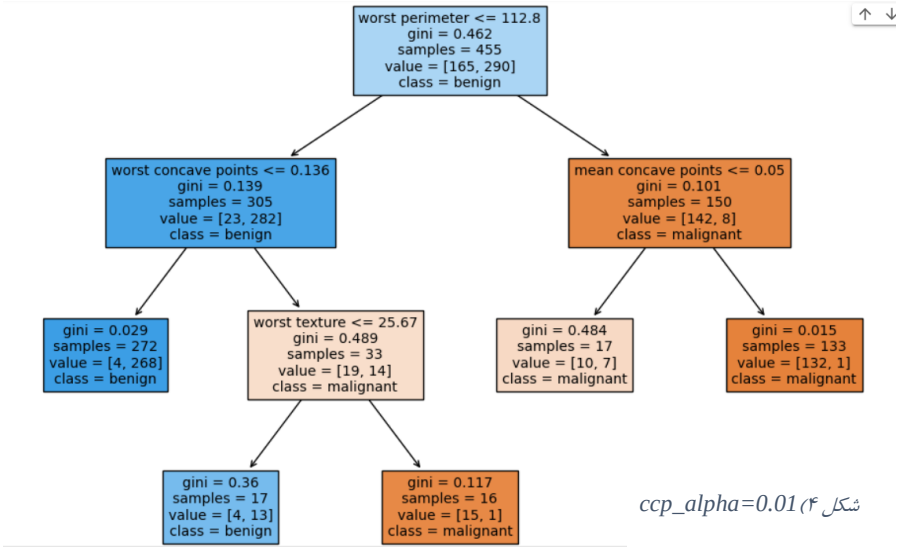
تغییر پارامتر هرس کردن :

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
# Example with ccp_alpha as a pruning parameter
ccp_alpha_values = [0.0, 0.01, 0.02]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)
```

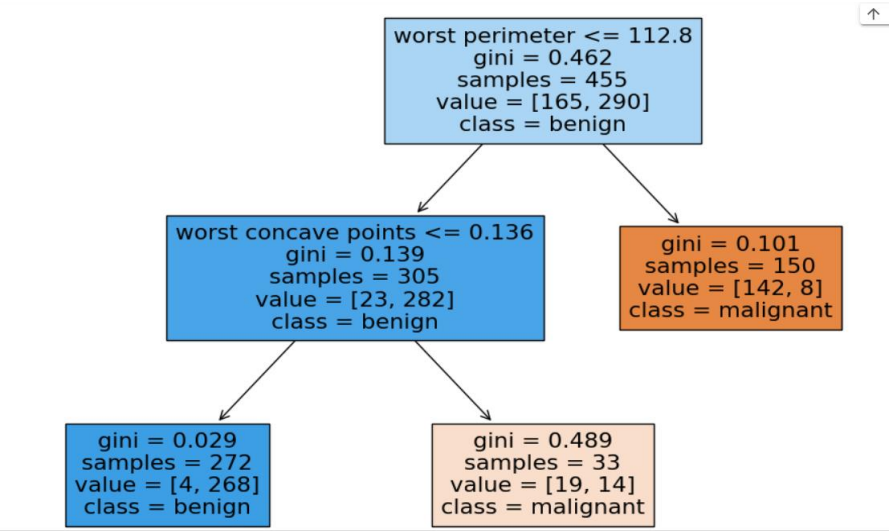
```
    # Train the model
    clf.fit(X_train, y_train)

    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```



شکل ۳) *ccp_alpha = 0*



شکل ۴) *ccp_alpha=0.01*

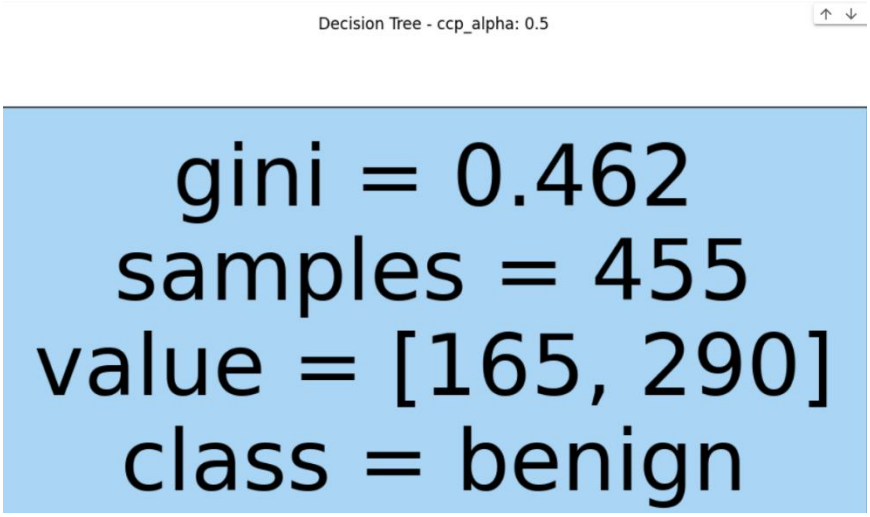


شکل ۵ ccp_alpha=0.02

حال مقدار ccp_alpha را بزرگ در نظر می گیریم :

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn import metrics

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=83)

# Create a decision tree classifier
# You can experiment with different hyperparameters, including pruning-
related ones
# Example with ccp_alpha as a pruning parameter
ccp_alpha_values = [0.5]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

    # Train the model
    clf.fit(X_train, y_train)
```

```
    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```



افزایش مقدار ccp_alpha در الگوریتم درخت تصمیم (Decision Tree) به معنای افزایش عوامل پروده و کاهش اورفیتینگ است. زمانی که شما ccp_alpha را افزایش می‌دهید، مدل مجبور می‌شود بیشترین تلاش را برای تناسب دقیق با داده‌های آموزشی نکند و سعی کند یک مدل ساده‌تر و عمومی‌تر ایجاد کند.

تأثیر افزایش ccp_alpha شامل موارد زیر می‌شود:

۱. کاهش اورفیتینگ: افزایش ccp_alpha باعث می‌شود تا درخت تصمیم کمتر بر داده‌های آموزشی بخاطر برای دقیق نمودن شود. این کاهش اورفیتینگ می‌تواند بهبود عملکرد مدل بر روی داده‌های جدید (داده‌های آزمون) را به ارمغان آورد.

۲. ساختار ساده‌تر درخت : افزایش ccp_alpha باعث می‌شود که درخت ساخته شده ساده‌تر باشد. بخش‌های درخت که با افزایش ccp_alpha اضافه نمی‌شوند، حذف می‌شوند و این باعث می‌شود که درخت کلی ساده‌تر و قابل فهم‌تر باشد.

۳. کاهش دقت در داده‌های آموزشی : افزایش ccp_alpha ممکن است باعث کاهش دقت مدل بر روی داده‌های آموزشی شود، زیرا مدل کمتر به داده‌های آموزشی نزدیک می‌شود.

۴. بهبود تعمیم‌پذیری : با کاهش اورفیتینگ و ساخت یک مدل ساده‌تر، توانمندی مدل در تعمیم به داده‌های جدید و ناشناخته افزایش می‌یابد.

حال به سراغ پیش بینی مسیر مربوط به دو نمونه از داده های مجموعهٔ آزمون می رویم :

```
# Analyze two samples from the test set
sample1 = X_test[0]
sample2 = X_test[1]

# Make predictions for the samples
prediction1 = clf.predict([sample1])[0]
prediction2 = clf.predict([sample2])[0]

# Display the results
print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")

# Sample 1
print("Sample 1:")
print("Features:", sample1)
print("True Label:", y_test[0])
print("Predicted Label:", prediction1)
print("\n")

# Sample 2
print("Sample 2:")
print("Features:", sample2)
print("True Label:", y_test[1])
print("Predicted Label:", prediction2)
```

```
Analysis for Decision Tree with max_depth=10:

Sample 1:
Features: [1.422e+01 2.312e+01 9.437e+01 6.099e+02 1.075e-01 2.413e-01 1.981e-01
 6.618e-02 2.384e-01 7.542e-02 2.860e-01 2.110e+00 2.112e+00 3.172e+01
 7.970e-03 1.354e-01 1.166e-01 1.666e-02 5.113e-02 1.172e-02 1.574e+01
 3.718e+01 1.064e+02 7.624e+02 1.533e-01 9.327e-01 8.488e-01 1.772e-01
 5.166e-01 1.446e-01]
True Label: 0
Predicted Label: 0


Sample 2:
Features: [1.747e+01 2.468e+01 1.161e+02 9.846e+02 1.049e-01 1.603e-01 2.159e-01
 1.043e-01 1.538e-01 6.365e-02 1.088e+00 1.410e+00 7.337e+00 1.223e+02
 6.174e-03 3.634e-02 4.644e-02 1.569e-02 1.145e-02 5.120e-03 2.314e+01
 3.233e+01 1.553e+02 1.660e+03 1.376e-01 3.830e-01 4.890e-01 1.721e-01
 2.160e-01 9.300e-02]
True Label: 0
Predicted Label: 0
```

در این کد، دو نمونه از مجموعه داده آزمون `X_test` با استفاده از یک مدل درخت تصمیم `clf` با پارامتر `max_depth`مشخص شده، تحلیل شده‌اند. این دو نمونه به ترتیب با نام‌های `sample1` و `sample2` شناخته می‌شوند.

سپس برای هر یک از این نمونه‌ها، پیش‌بینی مدل `clf` اعمال شده و نتایج به صورت زیر نمایش داده شده‌اند:

- برای `sample1`:
  - ویژگی‌ها: مقادیر ویژگی‌های این نمونه.
  - برچسب واقعی: برچسب واقعی متناظر با این نمونه از مجموعه داده آزمون (`y_test`).
  - برچسب پیش‌بینی شده: پیش‌بینی مدل برای این نمونه (`prediction1`).
- برای `sample2`:
  - ویژگی‌ها: مقادیر ویژگی‌های این نمونه.
  - برچسب واقعی: برچسب واقعی متناظر با این نمونه از مجموعه داده آزمون (`y_test1`).
  - برچسب پیش‌بینی شده: پیش‌بینی مدل برای این نمونه (`prediction2`).

این اطلاعات به شما این امکان را می‌دهد که نتایج پیش‌بینی مدل را بررسی کنید و با برچسب‌های واقعی مقایسه نمایید.

محاسبه دقت برای دو نمونه درخت تصمیم ارائه شده :

```
# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)

# Display the results
print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")
print("Accuracy:", accuracy)
```

```
Analysis for Decision Tree with max_depth=10:

Accuracy: 0.9210526315789473
```

```
# Set max_depth to 5
max_depth = 5

# Create a decision tree classifier
clf = DecisionTreeClassifier(max_depth=max_depth)

# Train the model
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)

# Display the results
print(f"\nAnalysis for Decision Tree with max_depth={max_depth}:\n")
print("Accuracy:", accuracy)
```

```
Analysis for Decision Tree with max_depth=5:

Accuracy: 0.9210526315789473
```

سواال ۳) اختیاری

توضیحات مربوط به سوال :

اگرچه در گذشته تحقیقات زیادی درباره عوامل مؤثر بر امید زندگی با در نظر گرفتن متغیرهای جمعیتی، ترکیب درآمد و نرخ مرگ و میر انجام شده است، اما مشاهده شده است که تأثیر واکسیناسیون و شاخص توسعه انسانی در گذشته به درستی مورد توجه قرار نگرفته است. همچنین، برخی از تحقیقات گذشته با استفاده از مدل‌های رگرسیون خطی چندگانه براساس داده‌های یک ساله برای تمام کشورها انجام شده‌اند. بنابراین، این موضوع محرکی است برای حل هر دو عامل ذکر شده با فراهم آوردن یک مدل رگرسیون بر پایه مدل اثرات ترکیبی و رگرسیون خطی چندگانه در نظر گرفتن داده‌ها از سال ۲۰۰۰ تا ۲۰۱۵ برای تمام کشورها. واکسیناسیون‌های مهم مانند هپاتیت B، پلیو و دیفتریا نیز در نظر گرفته خواهند شد. به طور خلاصه، این مطالعه بر فاکتورهای واکسیناسیون، فاکتورهای مرگ و میر، فاکتورهای اقتصادی، فاکتورهای اجتماعی و سایر فاکتورهای مرتبط با سلامت تمرکز خواهد داشت. از آنجا که مشاهدات این مجموعه داده بر اساس کشورهای مختلف است، برای یک کشور بهتر است

تا عامل پیش‌بینی‌کننده‌ای که به کاهش امید زندگی منجر می‌شود را تشخیص دهد. این به کشور کمک می‌کند تا بفهمد کدام حوزه باید با اهمیت بیشتری مورد توجه قرار گیرد تا به بهبود بهره‌وری امید زندگی جمعیت خود بپردازد.

این پروژه بر اطمینان از دقت داده‌ها بنا شده است. مخزن داده‌های سازمان جهانی بهداشت (GHO) تحت مختار سازمان بهداشت جهانی (WHO) وضعیت بهداشت و همچنین بسیاری از عوامل مرتبط دیگر برای تمام کشورها را پایش می‌کند. این مجموعه داده‌ها برای اهداف تجزیه و تحلیل داده‌های بهداشت به عموم عرضه شده است. مجموعه داده مربوط به امید زندگی و عوامل بهداشت برای ۱۹۳ کشور از همان وب‌سایت مخزن داده WHO و داده‌های اقتصادی متناظر آن از وب‌سایت سازمان ملل متحد جمع‌آوری شده است. از بین تمام دسته‌های عوامل مرتبط با سلامت، فقط عوامل بحرانی که نماینده بیشتری هستند انتخاب شده‌اند. مشاهده شده است که در ۱۵ سال گذشته، توسعه زیادی در بخش بهداشت صورت گرفته است که منجر به بهبود نرخ مرگ و میر انسانی، به ویژه در کشورهای در حال توسعه در مقایسه با ۳۰ سال گذشته شده است. بنابراین، در این پروژه، برای تحلیل بیشتر، از داده‌ها از سال ۲۰۰۰ تا ۲۰۱۵ برای ۱۹۳ کشور استفاده شده است. فایل‌های داده فردی به یک فایل داده ترکیب شده‌اند. در بررسی اولیه تجزیه و تحلیل داده‌ها، برخی از مقادیر افتراقی دیده شد. چون داده‌ها از WHO بودند، هیچ خطای آشکاری پیدا نکردیم. داده‌های گم‌شده در نرم‌افزار R با استفاده از دستور Missmap مدیریت شدند. نتیجه نشان داد که بیشتر داده‌های گم‌شده مربوط به جمعیت، هپاتیت B و GDP بودند. داده‌های گم‌شده مربوط به کشورهای کمتر شناخته‌شده مانند وانواتو، تونگا، توگو، کیپ ورد و غیره بودند. پیدا کردن تمام داده‌ها برای این کشورها دشوار بود و بنابراین تصمیم گرفته شد که این کشورها را از مجموعه داده نهایی حذف کنیم. فایل ترکیب شده نهایی (مجموعه داده نهایی) شامل ۲۲ ستون و ۲۹۳۸ ردیف بود که به معنای ۲۰ متغیر پیش‌بینی‌کننده بود. تمام متغیرهای پیش‌بینی‌کننده سپس به چندین دسته گسترده تقسیم شدند: عوامل مرتبط با واکسیناسیون، عوامل مرگ و میر، عوامل اقتصادی و عوامل اجتماعی.

شرح کد مربوطه :

در این دیتاست هم داده هایی به شکل string داریم و هم در بخشی از قسمت های دیتاست می بینیم که برخی از داده ها NAN می باشند. برای همین موضوع به صورت زیر عمل می کنیم :

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split
# set style of visualization
sns.set_style("whitegrid")
sns.set_palette("RdBu")
```

در ابتدا که کتابخانه های مختلف را در کولب ایمپورت می کنیم. سپس با استفاده از دستورات مربوط به درخت تصمیم به سراغ رگرسیون داده ها و پیش بینی داده های آزمون می رویم و سپس آن را با داده های واقعی مقایسه می کنیم و میزان خطا را مشخص می کنیم :

```
!pip install --upgrade --no-cache-dir gdown
!gdown 13UXkURa_S_QaHNsBO0m1UzbrVH1cakJK
data = pd.read_csv('/content/Life Expectancy Data.csv')
data.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	

5 rows × 22 columns

```
# first i see some column name with empty space i will fixed it to ease of use
data.columns = data.columns.str.strip()
```

این دستور به شما کمک می‌کند تا نام ستون‌های جدول داده‌هایتان را تمیز کنید. با استفاده از str.strip بر روی نام ستون‌ها، هر فضای خالی در ابتدا یا انتهای نام ستون حذف می‌شود. این کار بهبود خوانایی و دسترسی به داده‌ها را فراهم می‌کند، زیرا احتمال دارد که در نام ستون‌ها فاصله‌های اضافی وجود داشته باشد که ممکن است باعث اشتباه در استفاده از آنها شود.

```
# Size of the data
data.shape
```

```
(2938, 22)
```

```
# A Quick Information about the Data
data.info()
```

این دستور info اطلاعات سریعی در مورد داده‌هایتان ارائه می‌دهد. این شامل تعداد ردیفها، تعداد و نوع داده‌های هر ستون، میزان حافظه مصرفی و اطلاعات مربوط به داده‌های نال null می‌شود. این اطلاعات می‌تواند به شما کمک کند تا داده‌هایتان را بهتر فهمیده و نقاط ضعف یا نقاط قوت ممکن در داده‌ها را شناسایی کنید.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   object 
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   object 
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64  
 10  BMI                              2904 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18  thinness  1-19 years             2904 non-null   float64
 19  thinness 5-9 years               2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

```
# Checking for Null Values
data.isnull().sum()
```

این دستور isnull().sum بررسی می‌کند که هر ستون از داده‌ها دارای چه تعداد مقدار نال null است. این اطلاعات به شما اجازه می‌دهند تا بررسی کنید که آیا هر ستون حاوی داده‌های ناقص است یا خیر. این اطلاعات مهم است زیرا می‌تواند تصمیم‌گیری در مورد نیاز به پر کردن مقادیر ناقص یا انجام پیش‌پردازش داده‌ها را تسهیل کند.

```
Country                             0
Year                                0
Status                              0
Life expectancy                    10
Adult Mortality                    10
infant deaths                       0
Alcohol                           194
percentage expenditure              0
Hepatitis B                       553
Measles                             0
BMI                                34
under-five deaths                   0
Polio                              19
Total expenditure                 226
Diphtheria                         19
HIV/AIDS                            0
GDP                               448
Population                        652
thinness  1-19 years               34
thinness 5-9 years                 34
Income composition of resources   167
Schooling                         163
dtype: int64
```

```
# check if duplicated in data
data.duplicated().any()
```

این دستور duplicated().any بررسی می‌کند که آیا در داده‌ها ردیف تکراری وجود دارد یا خیر. اگر خروجی Trueباشد، این نشان می‌دهد که حداقل یک ردیف در داده‌ها تکرار شده است. این اطلاعات مهم است زیرا ردیف‌های تکراری ممکن است به دلایل مختلفی وجود داشته باشند، مثل داده‌های تکراری یا نقص در جمع‌آوری داده. مدیریت صحیح با ردیف‌های تکراری می‌تواند دقت و قابلیت تفسیر داده‌ها را افزایش دهد.

```
False
```

```
# see quick info of numeric values
data.describe()
```

این دستور describe اطلاعات آماری خلاصه‌ای در مورد ستون‌های عددی داده‌هایتان ارائه می‌دهد. این شامل تعداد، میانگین، انحراف معیار، مینیمم، کارآیی، و ماکزیمم برای هر ستون عددی می‌شود. این اطلاعات به شما کمک می‌کند تا توزیع و مشخصات اصلی داده‌های عددی خود را درک کنید.

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Pol
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2938.000000	2919.0000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247	42.035739	82.5501
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034	160.445548	23.4280
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.000000	3.0000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	0.000000	78.0000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	4.000000	93.0000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	28.000000	97.0000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	2500.000000	99.0000

```
# see quick info of category values
data.describe(include = object)
```

این دستور describe(include = object) اطلاعات خلاصه‌ای از ستون‌های داده‌های دسته‌ای نوع object را نمایش می‌دهد. این اطلاعات شامل تعداد دسته‌ها unique تعداد اجزاء هر دسته top تعداد تکرار بیشترین دسته freq می‌شود. این اطلاعات به شما کمک می‌کنند تا توزیع و ویژگی‌های مهم داده‌های دسته‌ای را درک کنید.

	Country	Status
count	2938	2938
unique	193	2
top	Afghanistan	Developing
freq	16	2426

```
# spliting data to train and test
train, test =  train_test_split(data, test_size = 0.2, random_state = 83)
```

در این دستور، داده‌ها به دو بخش آموزش train و آزمون test تقسیم می‌شوند. این کار با استفاده از train_test_split انجام می‌شود. پارامتر test_size نسبت تعیین کننده‌ی حجم داده‌های آزمون است که در اینجا برابر با ۰,۲ یا ۲۰٪ مجموع داده‌ها است. پارامتر random_state تعیین‌کننده‌ی یک seed برای اطمینان از قابل تکرار بودن تقسیم داده‌ها است، به این ترتیب هر بار که این دستور اجرا می‌شود، نتایج یکسانی به دست می‌آید.

```
def fill_train_with_median():
    return train.fillna(train.median(numeric_only = True))

def fill_test_with_median():
    return test.fillna(test.median(numeric_only = True))

# Apply the function to data
train = fill_train_with_median()
test = fill_test_with_median()
```

این تابع‌های fill_train_with_median و fill_test_with_median به ترتیب داده‌های آموزش train و آزمون test را با مقادیر میانه‌ای median ستون‌های عددی پر می‌کنند. این کار با استفاده از دستور fillna انجام شده و مقادیر ناقص NaN در ستون‌های عددی با میانه آن ستون جایگزین می‌شوند.

در نهایت، تابع‌های پرکردن با میانه به داده‌های آموزش و آزمون اعمال شده و داده‌های اصلی تغییر می‌کنند.

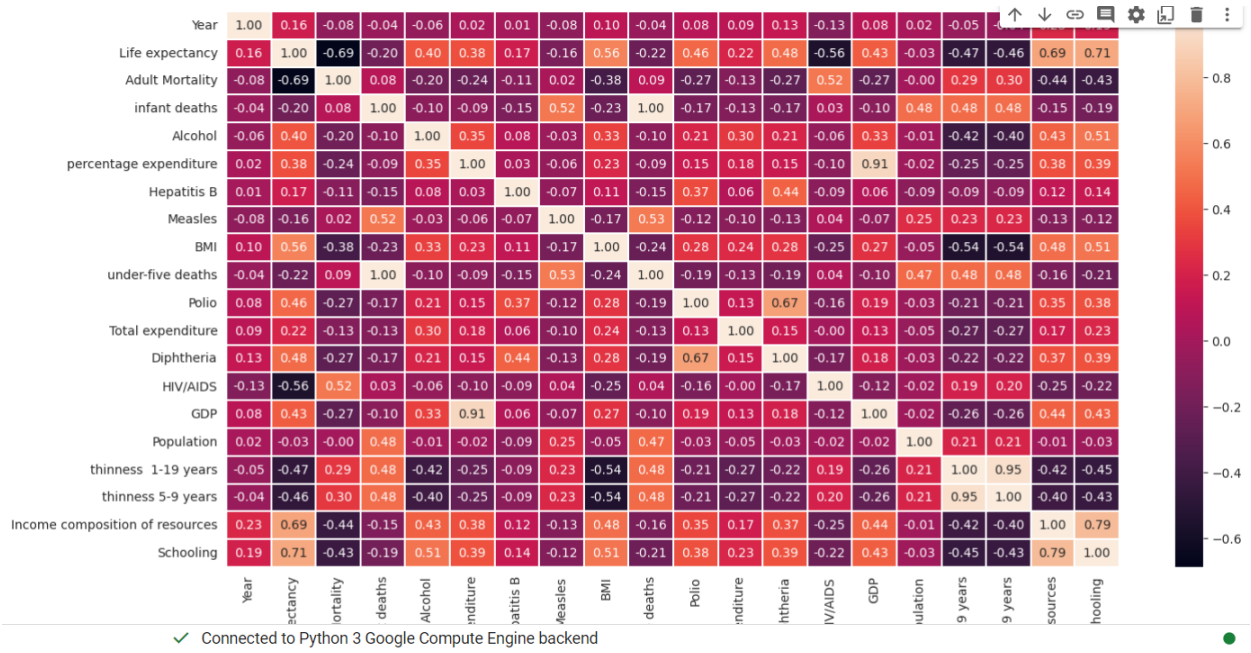
```
train.isna().sum()
```

این دستور isna().sum تعداد مقادیر ناقص NaN در هر ستون از داده‌های آموزش train را برمی‌گرداند. این اطلاعات مفید است تا بررسی کنید که پر کردن با میانه به درستی انجام شده است و دیگر مقادیر ناقص وجود ندارد. اگر تمامی مقادیر در این خروجی صفر باشد، نشان‌دهنده‌ی این است که دیگر مقادیر ناقص در داده‌های آموزش باقی نمانده‌اند.

```
Country                          0
Year                             0
Status                           0
Life expectancy                  0
Adult Mortality                  0
infant deaths                    0
Alcohol                          0
percentage expenditure           0
Hepatitis B                      0
Measles                          0
BMI                              0
under-five deaths                0
Polio                            0
Total expenditure                0
Diphtheria                       0
HIV/AIDS                         0
GDP                              0
Population                       0
thinness  1-19 years             0
thinness 5-9 years               0
Income composition of resources  0
Schooling                        0
dtype: int64
```

```
plt.figure(figsize = (16,8))
sns.heatmap(train.select_dtypes(exclude = object).corr(), annot = True,
fmt = ".2f", linewidths = 0.2)
plt.show()
```

این کد یک نمودار حرارت heatmap از ماتریس همبستگی بین ستون‌های عددی داده‌های آموزش train ایجاد می‌کند. از sns.heatmap برای تولید نمودار حرارت استفاده شده است. پارامترهای annot = True و fmt = 0.2f باعث میشوند تا مقادیر همبستگی روی نمودار نمایش داده شوند و با دقت دو رقم اعشار نمایش داده شوند. ماتریس همبستگی نشان دهنده‌ی ارتباطات آماری بین ستون‌های عددی است.




```
# create object from labelencoder
encoder = LabelEncoder()
for column in ["Country", "Status"]:
    train[column] = encoder.fit_transform(train[column])
    test[column] = encoder.fit_transform(test[column])
```

در این بخش از کد، یک شیء از کلاس LabelEncoder ایجاد می‌شود و سپس برای ستون‌های Country و Status داده‌های آموزش train و آزمون test این LabelEncoder به کار گرفته می‌شود.

عملکرد LabelEncoder این است که به ازای هر دسته مختلف در ستون، یک عدد نسبت می‌دهد. این کار معمولاً برای تبدیل داده‌های دسته‌ای به فرمت قابل استفاده در الگوریتم‌های یادگیری ماشین مفید است.

```
X_train, y_train = train.drop(["Life expectancy"], axis=1).values,
train[["Life expectancy"]].values
X_test, y_test = test.drop(["Life expectancy"], axis=1).values,
test[["Life expectancy"]].values
```

در این بخش، داده‌های آموزش و آزمون برای مدلسازی جدا شده‌اند. X_train حاوی ویژگی‌ها برای آموزش، y_train حاوی مقدار متغیر وابسته برای آموزش، X_test حاوی ویژگی‌ها برای آزمون، و y_testحاوی مقدار متغیر وابسته برای آزمون است.

```
# Scaling train data using min max scaler
scaler = MinMaxScaler()

X_train= scaler.fit_transform(X_train)
```

```
X_test= scaler.transform(X_test)
```

در این بخش از کد، داده‌های ویژگی X_train و X_test برای آموزش و آزمون با استفاده از MinMaxScalerمقیاس‌پذیر شده‌اند. این مقیاس‌دهی به ازای هر ستون، مقادیر را به یک بازه استاندارد (معمولاً [۰, ۱]) تبدیل می‌کند. این کار معمولاً برای بهبود عملکرد الگوریتم‌های یادگیری ماشین که به مقیاس داده حساس هستند، مفید است.

```
y_train
```

نتایج داده ای پیش بینی شده و مقایسه ان ها :

```
array([[74.6],
       [61. ],
       [73.5],
       ...,
       [83. ],
       [80. ],
       [71.7]])
```

```
tree_model = tree.DecisionTreeRegressor(random_state=83)
```

یک مدل رگرسیون درخت تصمیم با استفاده از کلاس DecisionTreeRegressorایجاد شده است.

این مدل به منظور پیش‌بینی یک مقدار عددی، به عنوان مثال، پیش‌بینی امیدراه زندگی، استفاده می‌شود.

```
# Fit the regressor to the training data
tree_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = tree_model.predict(X_test)
```

در این بخش از کد:

- مدل رگرسیون درخت تصمیم با داده‌های آموزش X_train و y_train آموزش داده شده است، از طریق tree_model.fit(X_train, y_train)

- سپس، با استفاده از مدل آموزش دیده بر روی داده‌های آزمون (`X_test`): `y_pred = tree_model.predict(X_test)` ، پیش‌بینی‌ها انجام شده و در y_pred ذخیره شده‌اند.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score
```

```
# Evaluate the model using metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')

# Calculate R-squared
r2 = r2_score(y_test, y_pred)
print(f'R-squared (R²): {r2}')
```

در این بخش از کد، از متریک‌های معمول برای ارزیابی عملکرد مدل استفاده شده است.

- mean_absolute_error و mean_squared_error به ترتیب معیارهای خطای میانگین مطلق و خطای میانگین مربع را بر حسب y_test و y_pred محاسبه می‌کنند.

- سپس، مقادیر این معیارها چاپ می‌شوند.

- r2score برای محاسبه ضریب توافق R-squared بر حسب y_test و y_pred استفاده شده و نتیجه آن نیز چاپ می‌شود. R-squared نشان دهنده‌ی تطابق مدل با داده‌هاست که در بازه [۰, ۱] قرار دارد، که مقادیر بالاتر نشان‌دهنده‌ی عملکرد بهتر مدل است.

```
Mean Absolute Error (MAE): 1.2568027210884352
Mean Squared Error (MSE): 4.606768707482994
R-squared (R²): 0.9462719119906475
```

```
np.random.seed(53)
random_row = np.random.choice(X_test.shape[0], size=10, replace=False)
test2 = X_test[random_row]
label_test2 = y_test[random_row]
y_hat2 = tree_model.predict(test2)
label_test2, y_hat2
# delta = ((label_test2-y_hat2)/label_test2)
# # delta = np.vstack(("% of errore", delta))
# print(delta, "\n")

# label_test2 , y_hat2 = arrays_with_names = np.vstack(("prediction",
label_test2)), np.vstack(("label", y_hat2))
# array = np.hstack((label_test2 , y_hat2))
# print(array)
```

در این بخش از کد، برای تست مدل روی یک زیرمجموعه از داده‌های آزمون `X_test` واحدهای تصادفی انتخاب شده‌اند. سپس بر روی این زیرمجموعه از داده‌ها، پیش‌بینی‌های مدل (`y_hat2`) و برچسب‌های واقعی (`label_test2`) محاسبه شده‌اند.

خروجی داده های تست تخمین زده شده در مقایسه با مقادیر واقعی :

```
(array([[66.4],
        [71.5],
        [74.4],
        [65.1],
        [74.2],
        [72.5],
        [47.7],
        [71.5],
        [69.7],
        [62. ]]),
 array([66. , 71. , 74.6, 65.3, 74.5, 72.5, 52.1, 71.4, 73.6, 61.8]))
```

تغییر پارامترهای رگرسیون درخت تصمیم می‌تواند تأثیر زیادی بر عملکرد مدل داشته باشد. در اینجا، برخی از پارامترهای مهم و تأثیرات آنها بر عملکرد مدل را بررسی می‌کنیم:

۱. `max_depth` (حداکثر عمق درخت)

- افزایش عمق ممکن است باعث برازش بهتر داده‌های آموزش شود، اما اگر به افراز بیش از حد برود، ممکن است باعث برازش بیش از حد و برداشت از داده‌ها شود (`overfitting`)

- کاهش عمق ممکن است باعث ساده‌تر شدن مدل و جلوگیری از برازش بیش از حد شود، اما اگر عمق آن به طور کلی کم باشد، ممکن است از قابلیت یادگیری مشکلات پیچیده تر خودداری کند.

۲. `min_samples_split` حداقل تعداد نمونه‌ها برای تقسیم یک گره

- افزایش این مقدار ممکن است باعث جلوگیری از تقسیم‌های زیاد گره‌ها شود و از برازش بیش از حد جلوگیری کند.

- کاهش این مقدار ممکن است به مدل اجازه دهد تا گره‌ها را به طور دقیق‌تر تقسیم کرده و از اطلاعات بیشتری برخوردار شود، اما اگر به صورت زیاد افزایش یابد، ممکن است باعث برازش بیش از حد شود.

۳. `min_samples_leaf` حداقل تعداد نمونه‌ها در یک برگ

- این پارامتر نیز به مانند `min_samples_split` کنترل می‌کند که آیا مدل تا چه حد از جزئیات داده‌ها برازش کند یا خیر.

- افزایش این مقدار ممکن است باعث جلوگیری از برازش بیش از حد شود.

۴. `max_features` حداکثر تعداد ویژگی‌ها برای جستجو در هر تقسیم

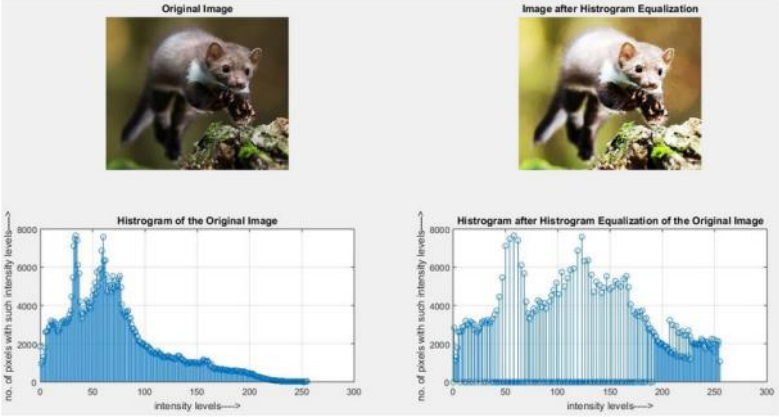
- مشخص‌کننده تعداد ویژگی‌هایی که مدل در هر مرحله از جستجو برای بهترین تقسیم بین گره‌ها استفاده می‌کند.

- افزایش این مقدار ممکن است باعث افزایش تنوع در جستجوها و بهبود عملکرد شود، اما در مواقعی ممکن است باعث افزایش زمان آموزش شود.

همان طور که مشاهده هم شد با تغییر پارامتر های درخت به تخمین بهتری از داده های خروجی آزمون رسیده ایم و از طرفی درصد خطا نیز کاهش یافته است.

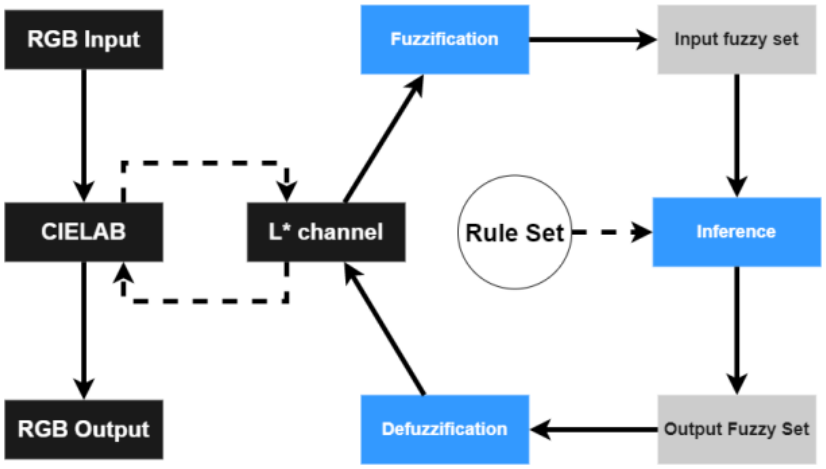
سوال پنجم :

بهبود تصویر[۳] یک روش معمول در پردازش تصویر است و بهبود کنتراست[۴] یک جنبهٔ اصلی آن است. روش‌های سنتی بهبود تصاویر مانند Histogram Equalization ممکن است موجب افزایش یا کاهش بیش از حد کنتراست تصاویر شوند، به خصوص در تصاویر با وضوح کم. هدف این پروژه، توسعهٔ یک سیستم استدلال فازی جدید برای بهبود کنتراست تصاویر است که نقایص روش‌های سنتی را برطرف می‌کند. شکل ۱ نمایی از فرآیند بهبود تصویر را نشان می‌دهد.



شکل ۱: نمایی از فرآیند بهبود تصویر.

طراحی‌شده‌تان را با حداقل یک الگوریتم سنتی در این زمینه در دو شاخصهٔ زمان و کیفیت عمل‌کرد (مانند شاخصهٔ PSNR[۵]) مقایسه کنید. برای سهولت انجام پیاده‌سازی‌ها تنها کافی است که کدهای آوردشده در این دفترچه‌کد گوگل‌کولب را تکمیل کنید.



شکل ۲: نمایی از سیستم فازی مدنظر برای بهبود تصویر در دفترچه‌کد گوگل‌کولب.

توضیحات کلی در مورد سوال :

PSNR به عنوان اختصار Peak Signal-to-Noise Ratio شناخته می‌شود و یک معیار است که برای اندازه‌گیری کیفیت تصاویر و ویدئوها استفاده می‌شود. این معیار معمولاً در زمینه‌های پردازش تصویر، فشرده‌سازی تصویر و انتقال تصاویر مورد استفاده قرار می‌گیرد.

فرمول PSNR به صورت زیر است:

$$PSNR = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

PSNR به عنوان یک معیار نسبی از کیفیت تصویر به کار می‌رود و عدد بالاتر نشان‌دهنده کیفیت بهتر است. این معیار مفید است زمانی که می‌خواهیم تاثیر فشرده‌سازی یا تغییرات در تصویر را اندازه‌گیری کنیم. با این حال، PSNR دارای محدودیت‌ها نیز است. برخی از این محدودیت‌ها عبارتند از:

۱. حساسیت به خطاهای کوچک:PSNR حساس به خطاهای کوچک است و ممکن است در مواردی که تغییرات ناچیزی در تصویر ایجاد شود، ارزیابی ناصحیح داشته باشد.

۲. ناپیوستگی:PSNR نمی‌تواند تغییراتی که به صورت غیر خطی در تصویر رخ می‌دهند را به خوبی ارزیابی کند. به عبارت دیگر، این معیار بر اساس فرض خطی بودن رابطه بین تصاویر عمل می‌کند.

در کل، PSNR یک ابزار مفید است اما برای بررسی جوانب دقیق‌تر و شناخت کیفیت وضوح تصاویر، ممکن است نیاز به استفاده از روش‌های دیگری مانند SSIM (Structural Similarity Index) یا PSNR-HVS (Peak Signal-to-Noise Ratio considering Human Visual System) باشد.

مراجع

[1] https://github.com/MJAHMADEE/MachineLearning2023

[2] Wang, L.X. (1997) A Course in Fuzzy Systems and Control. Prentice-Hall, Englewood Cliffs.