به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش مینی پروژه شماره سوم

فاطمه اسلامی

9819393

استاد : جناب آقای دکتر مهدی علیاری

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، باید تعداد توابع متعلقه را در سوال محاسبه کنیم. این بخش دارای دو حالت متفاوت است که باید یکبار برای محدوده اول و دیگر برای محدوده دوم، مشخصات مربوطه را محاسبه نماییم. این اقدام به دو مرحله تقسیم می‌شود: محاسبه کران اول و محاسبه کران دوم:

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

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

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

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

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)، 41 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) ، 9 تابع تعلق مثلثی خواهیم داشت.

توابع تعلق به صورت زیر خواهند بود :

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

$$\mu_{A^j}(x) = \mu_{Aj}(x; a_j, b_j, c_j) = \mu_A^j(x; e^{j-1}, e^j, e^{j+1}), \quad \begin{cases} j = 2, \cdots, 40 \\ e^j = \alpha + h(j-1) = -1 + 0/25(j-1) \end{cases}$$

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

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 می خواهیم **کد متلب** مربوط به غیر فازی ساز میانگین را در ابتدا برای کران مرتبه اول و سپس برای کران مرتبه دوم بزنیم. در ابتدا به سراغ غیر فازی ساز میانگین برای کران مرتبه اول می رویم :

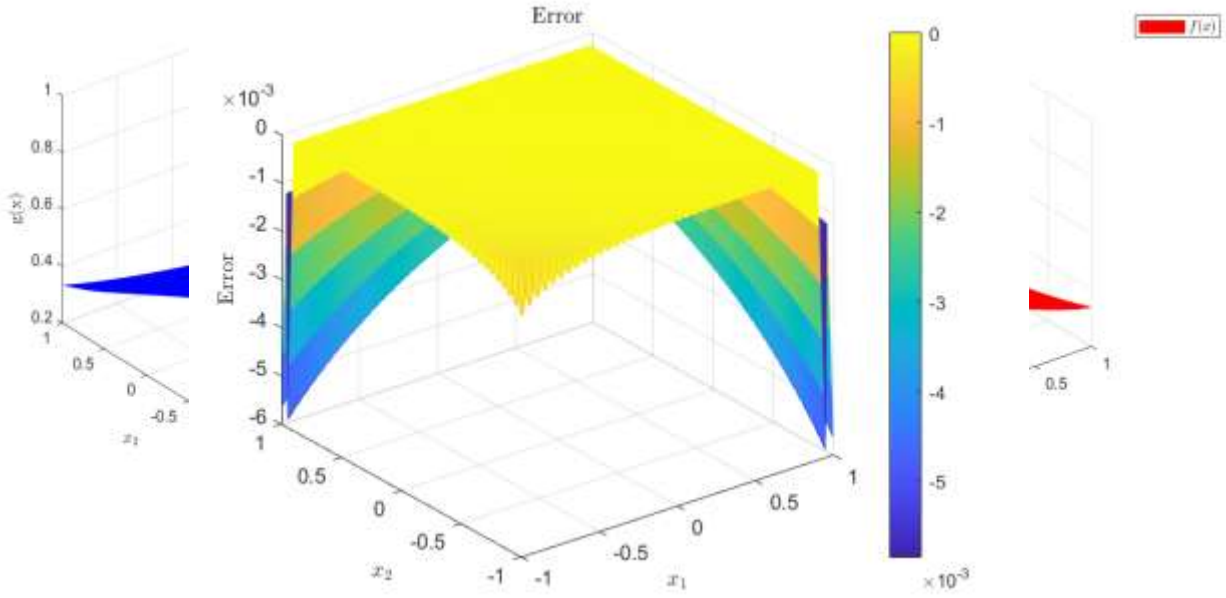
برای غیر فازی ساز میانگین و کران مرتبه اول :

```
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



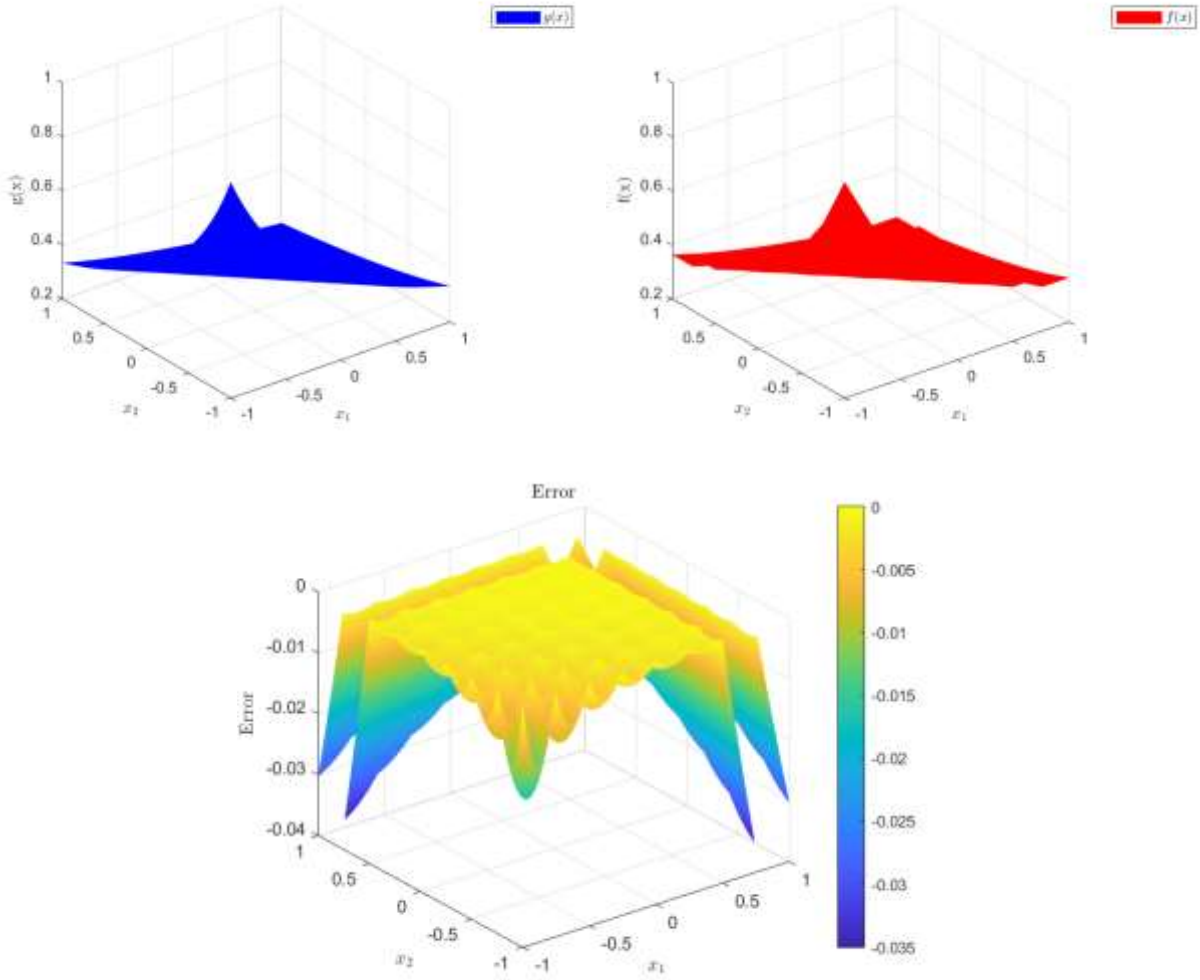
برای غیر فازی ساز میانگین و کران مرتبه دوم :

```
%% 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 :

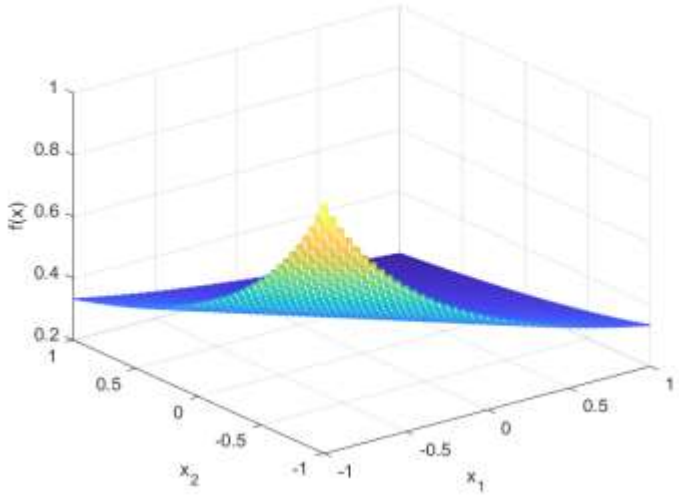
مقادیر h و N که همان مقادیری هستند که در بخش قبل بدست آمده است.

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 :



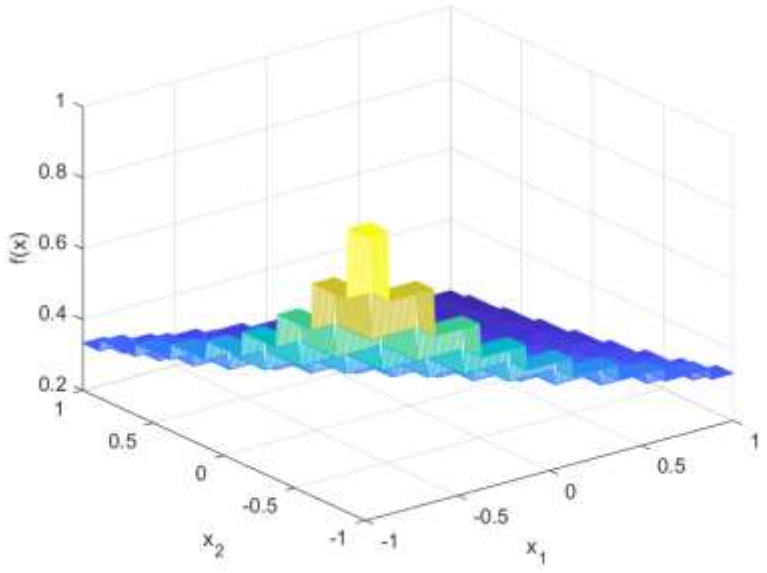
کران مرتبه دوم :

```
%% ?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 1 :

```
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;

figurel = 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)]);
```

۱- **تولید داده:**

- دنباله‌ای زمانی به نام Mackey-Glass ایجاد می‌شود که از طریق یک فرمول خاص به صورت بازخوردی به دست می‌آید.

۲- **طراحی سیستم منطق فازی:**

- دو حالت برای تعریف توابع عضویت برای ورودی‌ها در نظر گرفته شده است:
  - در حالت اول، ۷ تابع عضویت برای هر ورودی در نظر گرفته شده است.
  - در حالت دوم، ۱۵ تابع عضویت برای هر ورودی اعمال می‌شود.
  - پارامترهای توابع عضویت به صورت دستی تعیین 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 از سیستم حذف می‌شوند تا سادگی و کارآمدی سیستم فازی تضمین شود.

۵- **ایجاد سیستم منطق فازی:**

- با استفاده از قوانین فازی محاسبه شده، یک سیستم منطق فازی Mamdani طراحی می‌شود. ورودی‌ها و خروجی‌های این سیستم با دقت تعریف می‌شوند و توابع عضویت برای ورودی‌ها و خروجی‌ها به دقت تعیین می‌گردند.

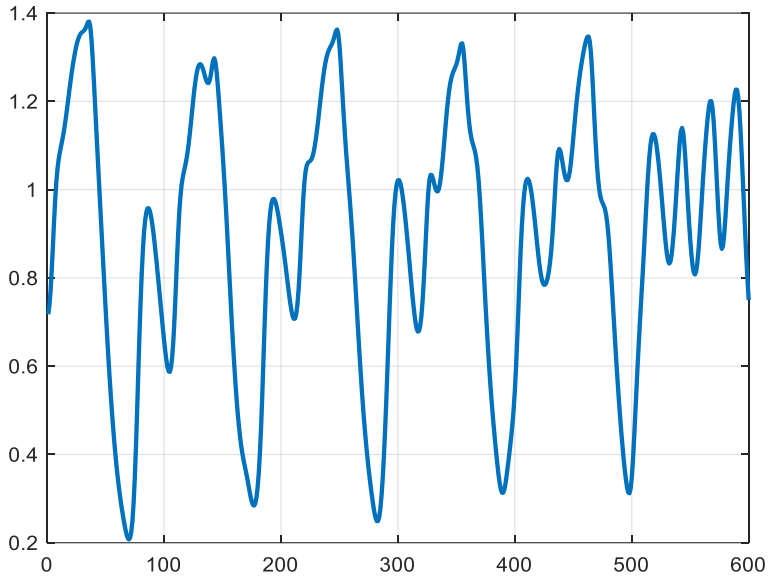
۶- **پیش‌بینی مقادیر:**

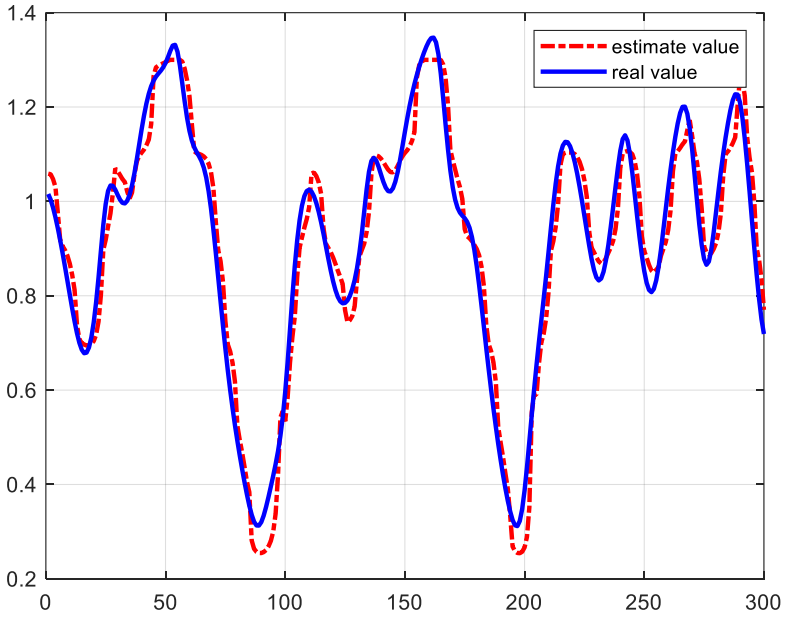
- از طریق سیستم منطق فازی ساخته شده، ۳۰۰ نقطه آینده از مجموعه داده پیش‌بینی می‌شوند. نتایج پیش‌بینی با نتایج واقعی مقایسه می‌شوند و در یک نمودار به تصویر کشیده می‌شوند.

۷- **نمایش توابع عضویت:**

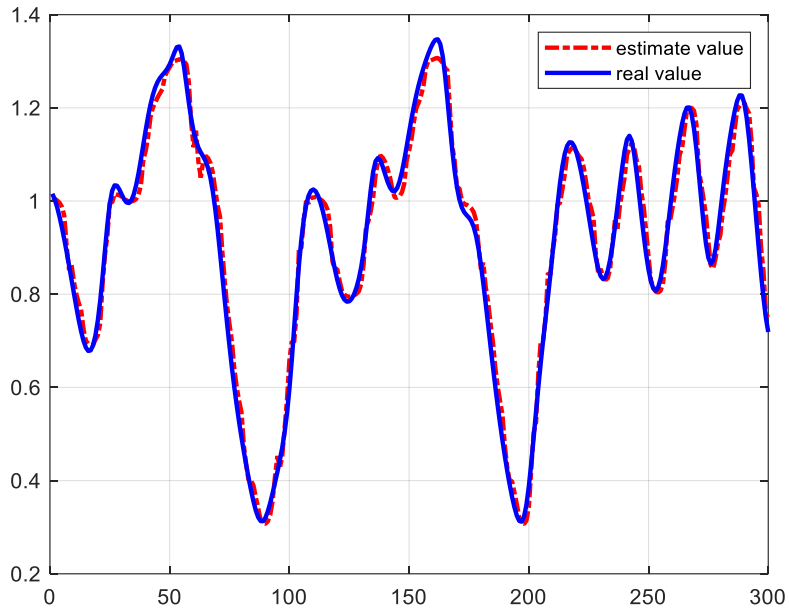
- این بخش امکان نمایش توابع عضویت برای یکی از ورودی‌ها را فراهم می‌کند. این توابع عضویت نشان می‌دهند که ورودی‌ها به چه اندازه در هر مرحله به هر قاعده فازی تعلق دارند. به طور کلی، این کد یک سیستم منطق فازی را ساخته که از آن برای پیش‌بینی مقادیر یک سری زمانی استفاده می‌شود.

شکل نتایج :

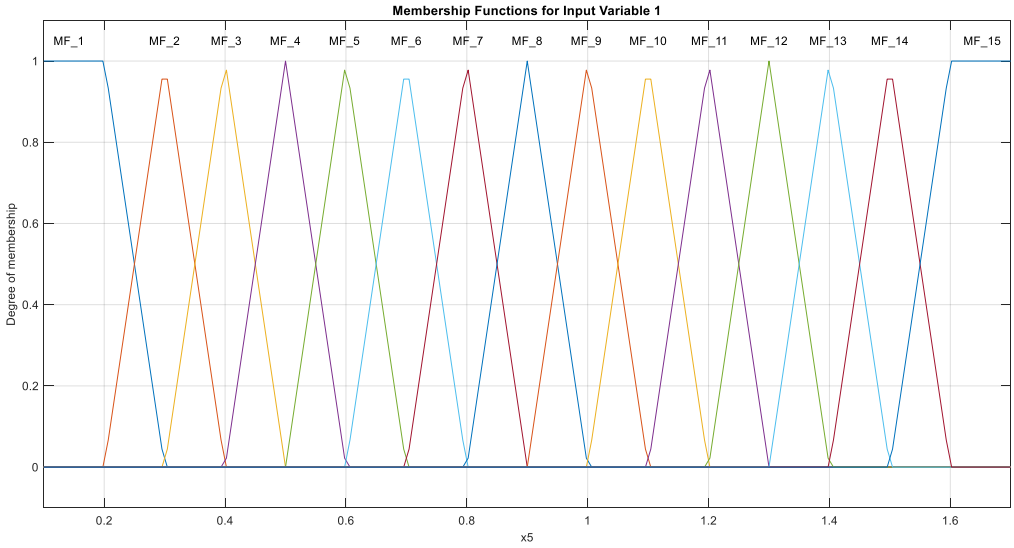




شکل 1) پیش بینی مدل سری زمانی



شکل 2) 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.

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

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

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

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.

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

کد بخش برگرفته از مرجع 2 :

```
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]);
```

۱- **متغیرها و پارامترها:**

- تعداد M توابع عضویت مربوط به سیستم فازی.

- تعداد نمونه‌های آموزشی برای مدل (num_training)

- تعداد کل نمونه‌ها (آموزش و تست) (total_num)

- مقدار ثابت گام آموزش learning rate با نام lambda که بر اساس گرادیان نزولی مشخص می‌شود.

۲- **پیش‌پردازش و مقداردهی اولیه:**

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

- مقداردهی اولیه برای ورودی‌ها و خروجی‌ها انجام می‌شود.

۳- **مقداردهی اولیه بر اساس توابع عضویت:**

- مقادیر اولیه برای توابع عضویت ورودی‌ها براساس توزیع یکنواخت در بازه [-1، 1] 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. در این مرحله، توابع عضویت، میانگین خروجی مدل، و ویژگی‌های مرتبط با توابع عضویت به‌روزرسانی می‌شوند.

۵- **آزمون مدل:**

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

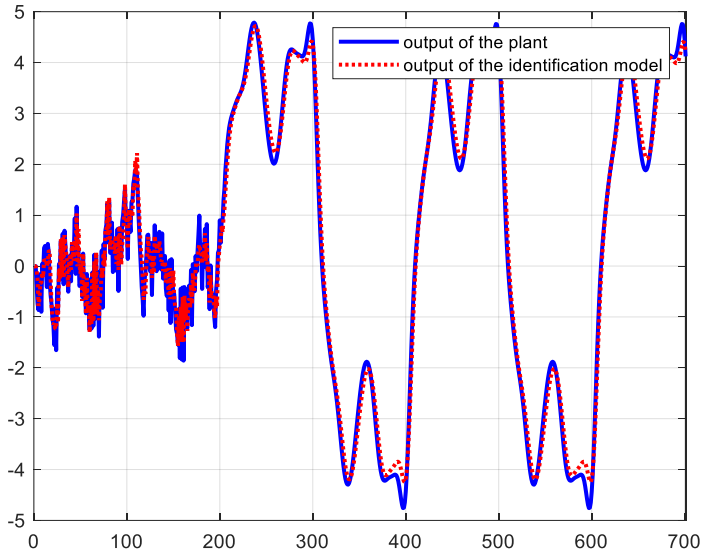
۶- **نمودار توابع عضویت اولیه:**

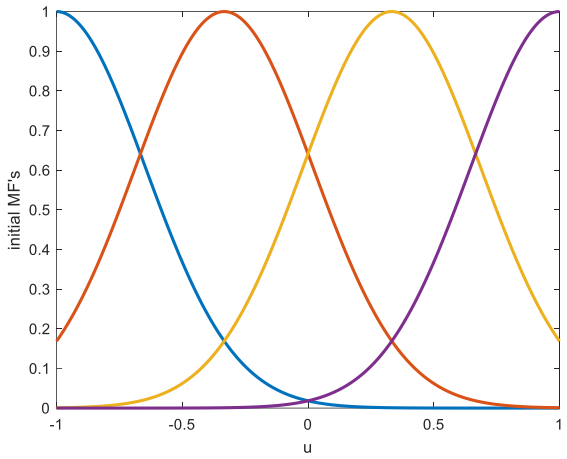
- توابع عضویت اولیه برای ورودی‌ها در یک نمودار نمایش داده می‌شوند.

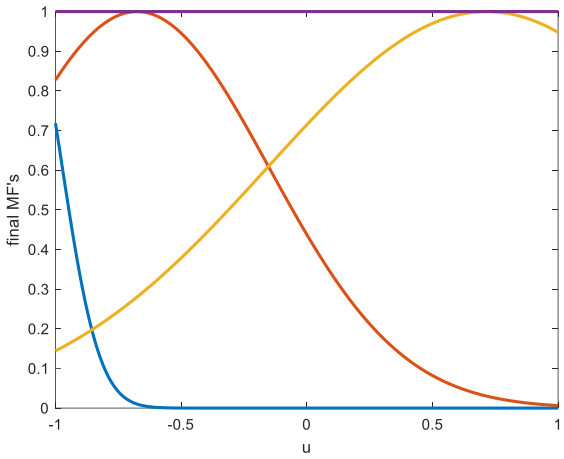
۷- **نمودار توابع عضویت نهایی:**

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

در کل، این کد یک مدل تطبیقی ایجاد می‌کند که با استفاده از توابع عضویت فازی، 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:

درخت تصمیم یک الگوریتم یادگیری ماشین است که به وسیله‌ی تصمیم‌ها و شرایط گسترده، داده‌ها را به دسته‌های مختلف تقسیم می‌کند. این الگوریتم از ساختار درختی برای نمایش تصمیم‌ها و جریان فرایند تصمیم‌گیری استفاده می‌کند. درخت تصمیم از سه عنصر اصلی تشکیل شده است:

1. **گره‌ها (نودها):**

- گره‌ها نقاط تصمیم‌گیری در درخت هستند. هر گره بر اساس یک ویژگی از داده‌ها، شرایط تصمیمی اعمال می‌کند و به یکی از شاخه‌های درخت می‌رود.

2. **شاخه‌ها:**

- شاخه‌ها به واحد اصلی اجزای درخت تصمیم اطلاق می‌شوند. هر شاخه نشان‌دهنده‌ی یک مسیر خاص از گره‌ها و تصمیمات است که به یک گره دستیابی می‌کند.

3. **برگ‌ها:**

- برگ‌ها در انتهای هر شاخه قرار دارند و نشان‌دهنده‌ی یک تصمیم نهایی یا گروه داده‌ها هستند. هر برگ می‌تواند به یک دسته یا مقدار خاص اشاره کند.

عملکرد اصلی درخت تصمیم در تصمیم‌گیری بر اساس ویژگی‌ها و شرایط مختلف داده‌ها می‌باشد. با پیروی از شاخه‌ها و گره‌ها، الگوریتم می‌تواند داده‌ها را به دسته‌ها مختلف تقسیم کرده و تصمیماتی بر اساس ویژگی‌های مختلف اعمال کند.3- **ریشه:**

- ریشه نقطه شروع درخت تصمیم است که از آن تمام جریان تصمیم‌گیری آغاز می‌شود.

- این گره به سوالات مرتبط با شرایط اولیه داده‌ها می‌پردازد.

- **گره‌های برگ:**

- گره‌های برگ پیش‌بینی یا دسته‌بندی نهایی را انجام می‌دهند.

- در این گره‌ها، تصمیمات نهایی بر اساس شرایط ایجاد شده در گره‌های داخلی گرفته می‌شود.

- **شرایط و سوالات:**

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

- برای مثال، ممکن است پرسیده شود: "آیا مقدار ویژگی X بزرگتر از یک حد مشخص است؟".

- **آموزش:**

- مدل درخت تصمیم با استفاده از مجموعه‌آموزشی آموزش می‌بینند.

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

- **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.

- **افزایش تفسیرپذیری:**

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

- این خاصیت از طریق تحلیل شاخص‌ها و شرایط درخت، تصمیم‌گیری مدل را قابل فهم‌تر و قابل تفسیرتر نموده و به کاربران امکان می‌دهد فرآیند تصمیم‌گیری را به صورت واضحتر درک کنند.

سوال 1)

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 می بینیم که ویژگی ستون 2 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
```

کد ارائه شده یک کلاس به نام Node ایجاد می‌کند که از آن برای ساختاردهی درخت تصمیم استفاده می‌شود. در این کد، تابع 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:

1. **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.

2. **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")})})})
```

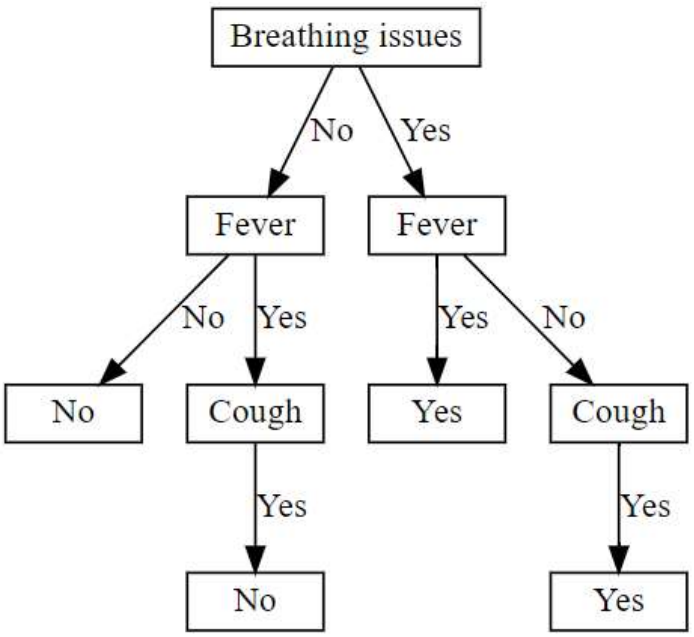
مشکل درون دیتاست می باشد. وقتی فیچر breating_issue به عنوان گره روت انتخاب می‌شه، برای مقادیر فیچر Yes و No هر دو Fever بیش‌ترین مقدار information gain رو دارد. همینجوری نظری هم به دیتاست نگاه کنیم، مشخص هست که فیچر Cough نقشی در ماجرا ندارد.

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)
```



با توجه به جدول مذکور، زمانی که فیچر breating_issue به عنوان گره روت انتخاب می‌شود، در هر دو حالت Yes و No برای فیچر Fever میزان بیشترین Information Gain را دارد. در نتیجه، اگر ورودی breating_issue در حالت No باشد و فیچر Fever در حالت Yes باشد، به فیچر Cough متصل می‌شویم. در این حالت، اگر Fever نیز در حالت No باشد، نتیجه می‌گیریم که شخص مبتلا به کرونا نیست. از سوی دیگر، اگر breating_issue در حالت Yes باشد و Fever هم در همین حالت باشد، مشخص است که فرد کرونایی است. در صورتی که breating_issue No باشد ولی Fever و Cough هر دو در حالت Yes باشند، این موضوع به تشخیص قطعی بیماری اشاره نمی‌کند.

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 2)

```
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 ساخت و نمایش درخت تصمیم بر روی داده‌های سرطان پستان به کار گرفته می‌شود. ابتدا، 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. این عمل به توجه به اطلاعات موجود در درخت تصمیم و روش تصمیم‌گیری در هر گره کمک می‌کند.

1. مدل با عمق 5:

- درخت تصمیم با عمق 5 به صورت گسترده‌تر و کلی‌تر اطلاعات را در اختیار می‌گیرد. این ممکن است به دلیل عمق کمتر باشد که از برخی اطلاعات خاص و ارتباطات محلی چشم‌پوشی می‌کند.

- با توجه به مقدار عمق 5، این درخت احتمالاً به دنبال اطلاعات مهم و کلان در مورد سرطان پستان است.

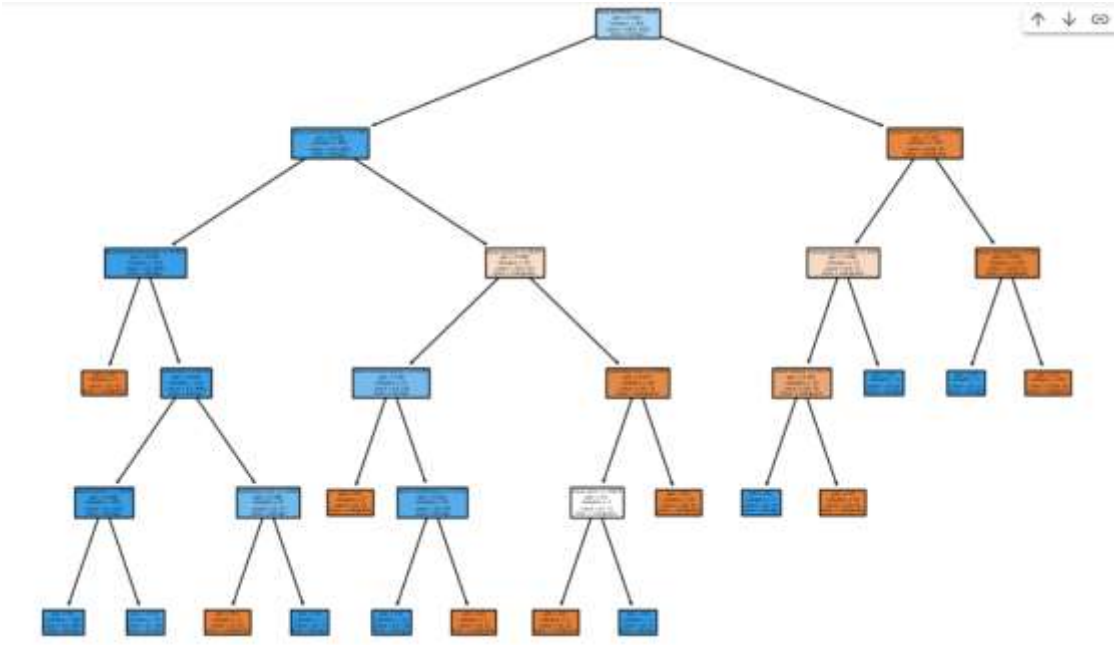
2. مدل با عمق 10:

- درخت با عمق بیشتر (10)، 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.

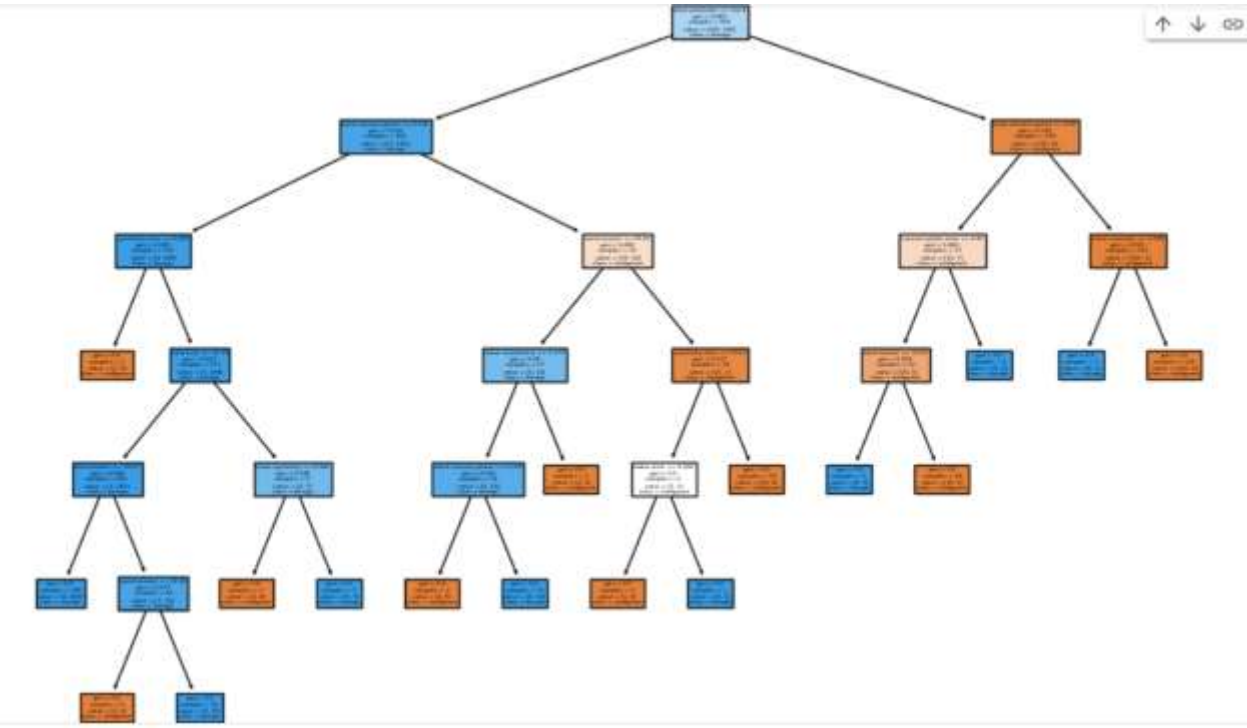
- این عمق بیشتر ممکن است منجر به یادگیری و حفظ جزئیات کمتری در مورد داده‌ها شود و ممکن است باعث افزایش دقت در دسته‌بندی شود.

نمایش نمودار ها :

درخت با عمق 5 :



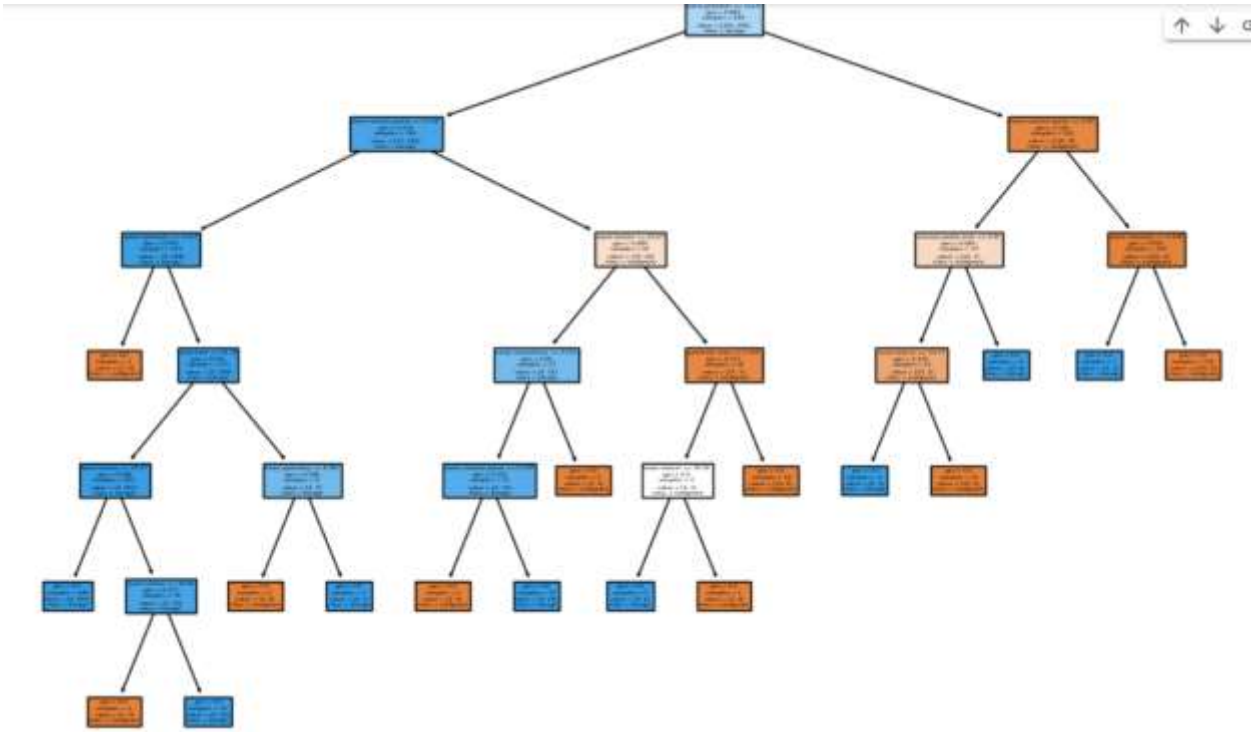
درخت با عمق 10 :



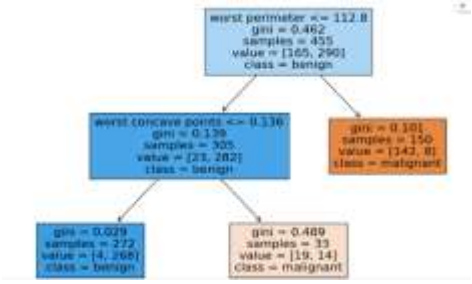
تغییر پارامتر هرس کردن :

```
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
```

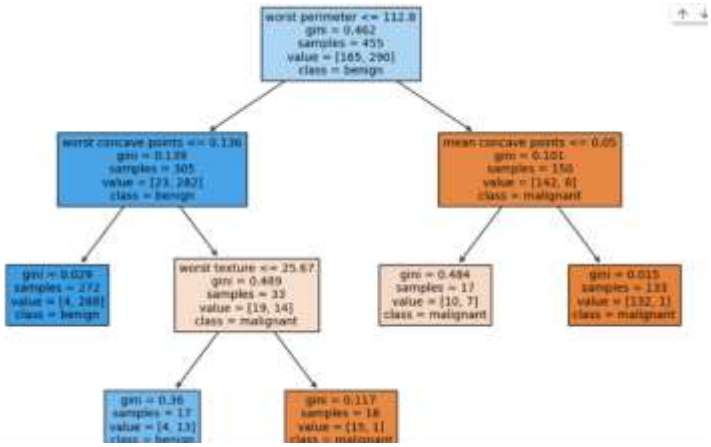
```
ccp_alpha_values = [0.0, 0.01, 0.02]  # Replace with your desired values
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

    # 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 3) ccp_alpha=0.01



شکل 4) ccp_alpha=0

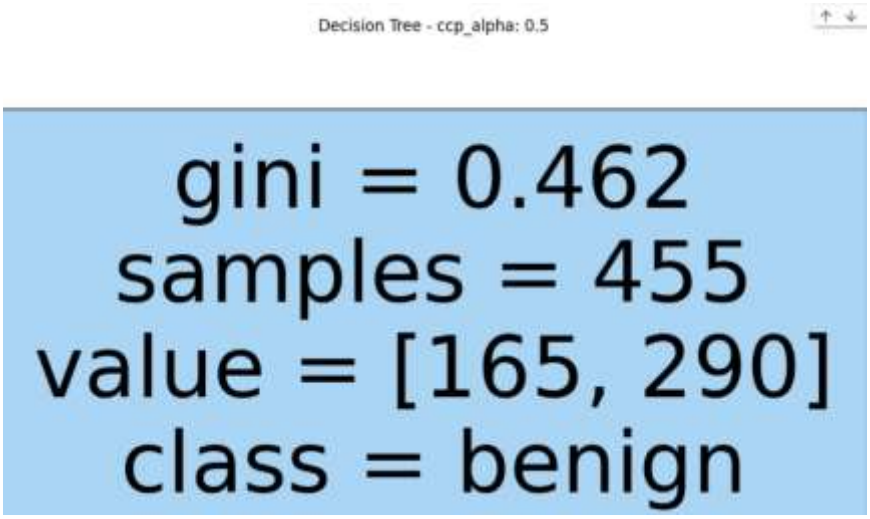


شکل 4) *ccp_alpha=0.02*

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 به این معناست که فاکتورهای پراورفیت افزایش می‌یابند و در نتیجه اورفیت (Overfitting) کاهش می‌یابد. هنگامی که ccp_alpha را افزایش می‌دهید، مدل ملزم می‌شود بیشترین تلاش را برای انطباق دقیق با داده‌های آموزشی نکند و سعی کند تا یک مدل ساده‌تر و کلان‌تر ایجاد کند.

تأثیر افزایش ccp_alpha شامل موارد زیر است:

1. **کاهش اورفیتینگ**: افزایش ccp_alpha باعث می‌شود که 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.

2. **ساختار ساده‌تر درخت**: افزایش ccp_alpha باعث می‌شود تا 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.

3. **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.

4. **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 3) 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 2000 تا 2015 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.