

به نام خدا

نام و نام خانوادگی:

محمد حسین بیاتی

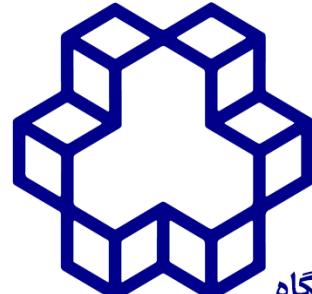
شماره دانشجویی:

۹۹۲۴۶۹۳

گزارش کار مینی پروژه شماره سه

استاد درس:

دکتر مهدی علیاری



دانشگاه
خواجہ نصیرالدین طوسی

K. N. Toosi University
of Technology



فهرست مطالب

۳	سوال ۱
۲۰	سوال ۲
۲۷	سوال ۳
۳۳	سوال ۴
۳۳	بخش ۱
۴۰	بخش ۲
۵۰	بخش ۳ (اختیاری)
۶۲	سوال ۵

سوال ۱

با استفاده از کران مرتبه اول (رابطه ۱۱-۴ در [۲]) و کران مرتبه دوم (رابطه ۱۰-۱۱ در [۲]), دو سیستم فازی با غیر فازی ساز میانگین و ماکریم طراحی کنید که تابع $g(x_1, x_2) = \frac{1}{3+x_1+x_2} \times [-1, 1]$ روی $U = [-1, 1] \times [-1, 1]$ را به شکل یکنواخت و با دقت $\epsilon = 0.1$ تقریب بزنند. سیستم های فازی طراحی شده را رسم کرده و با هم مقایسه کنید.

در هر کدام از روش ها توابع تعلق را به صورت مثلثی در نظر می گیریم و به کمک روش مطرح شده در صورت سوال تعداد توابع تعلق را تعیین می کنیم وسپس به کمک غیر فازی ساز خواسته شده سیستم فازی را طراحی می کنیم.

استفاده از کران مرتبه اول:

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

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

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

طبق این رابطه تنها کافیست نرم بی نهایت مشتق جزئی تابع g بر حسب x_1 و x_2 را حساب کنیم. طبق تعریف نرم بی نهایت داریم:

همانطور که مشخص است چونکه صورت ثابت است، حداقل مقدار زمانی رخ می دهد که مخرج کمترین مقدار را داشته باشد، که این زمانی اتفاق می افتد که $x_1 = x_2 = -1$

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

$$= \left| \frac{-1}{(3 - 1 - 1)^2} \right| = 1$$

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

$$= \left| \frac{-1}{(3 - 1 - 1)^2} \right| = 1$$

بنابراین طبق روش کران مرتبه اول اگر در نظر بگیریم $h = h_1 = h_2$ داریم:

$$h \leq \frac{\epsilon}{1+1} = \frac{0.1}{2} = 0.05 \rightarrow h = 0.05$$

همچنین با توجه به کران تعریف شده برای تابع داریم $\alpha = -1$, $\beta = 1$. پس داریم:

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

بنابراین برای این روش کافیست ۴۱ تابع تعلق در نظر بگیریم.

روش غیر فازی ساز میانگین:

چون سوال اطلاعاتی راجع به این موضوع نداده است این توابع تعلق را مثلثی در نظر می گیریم. بنابراین این توابع تعلق را به صورت زیر تعیین می کنیم:

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

$$\mu_{A^j}(x) = \mu_{A^j}(x; a_j, b_j, c_j) = \mu_{A^j}(x; e^{j-1}, e^j, e^{j+1}), \quad \begin{cases} j = 2, \dots, 40 \\ e^j = \alpha + h(j-1) = -1 + 0.05(j-1) \end{cases}$$

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

غیر فازی ساز:

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

کد متلب و نتایج:

توضیحات کد و همچنین انواع نمونه های آن در فایل های حل تمرین به طور کامل توضیح داده شده است، پس از آوردن توضیحات اضافی صرف نظر می کنیم.

```
%% First order
% Mean Defuzzification

clc
clear
close all

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;

% triangular fuzzy membership function
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(1)
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)', 'Interpreter', 'latex')
grid on

```



```

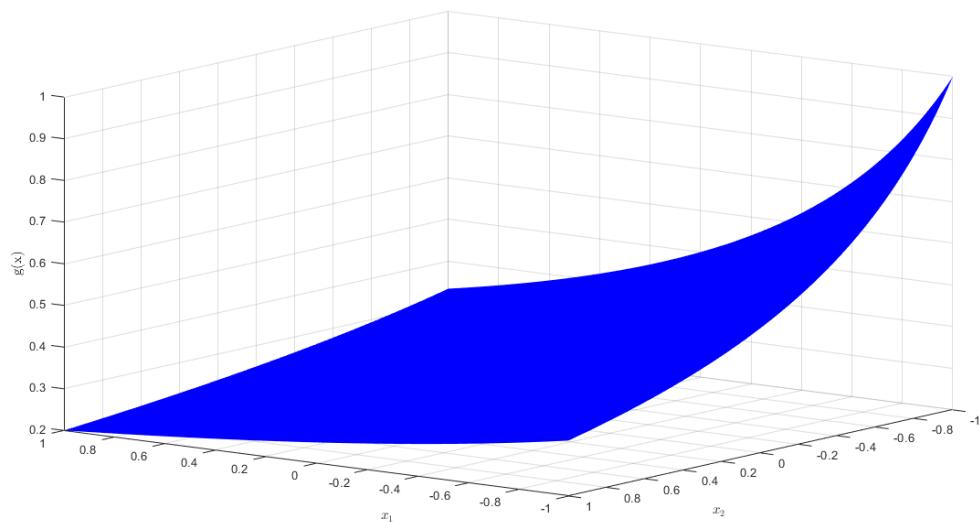
figure(2)
surf(x1, x2, f_x, 'FaceColor', 'r', 'EdgeColor', 'none')
xlabel('$x_1$', 'Interpreter', 'latex')
ylabel('$x_2$', 'Interpreter', 'latex')
zlabel('f(x)', 'Interpreter', 'latex')
legend('$f(x)$', 'Interpreter', 'latex')
grid on

figure(3)
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

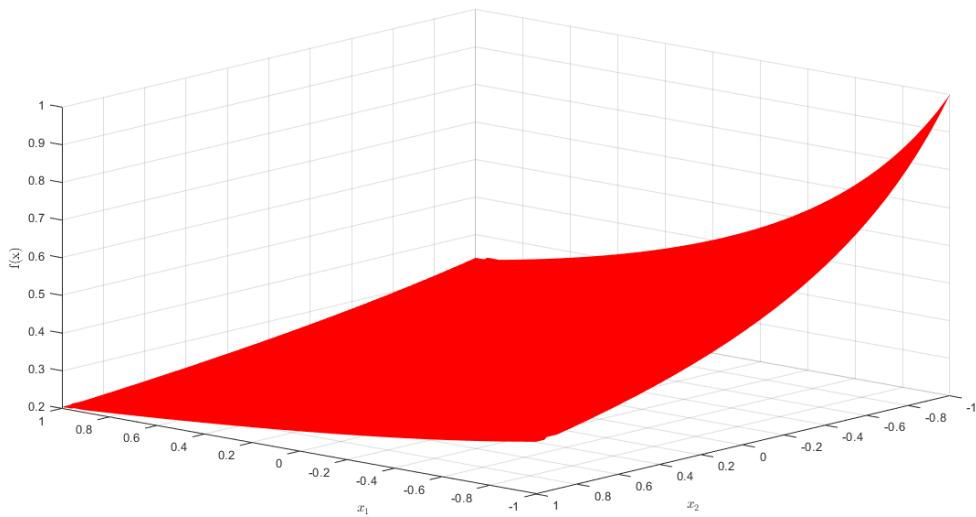
```

نتائج متلب:

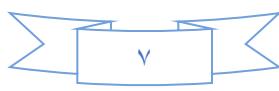
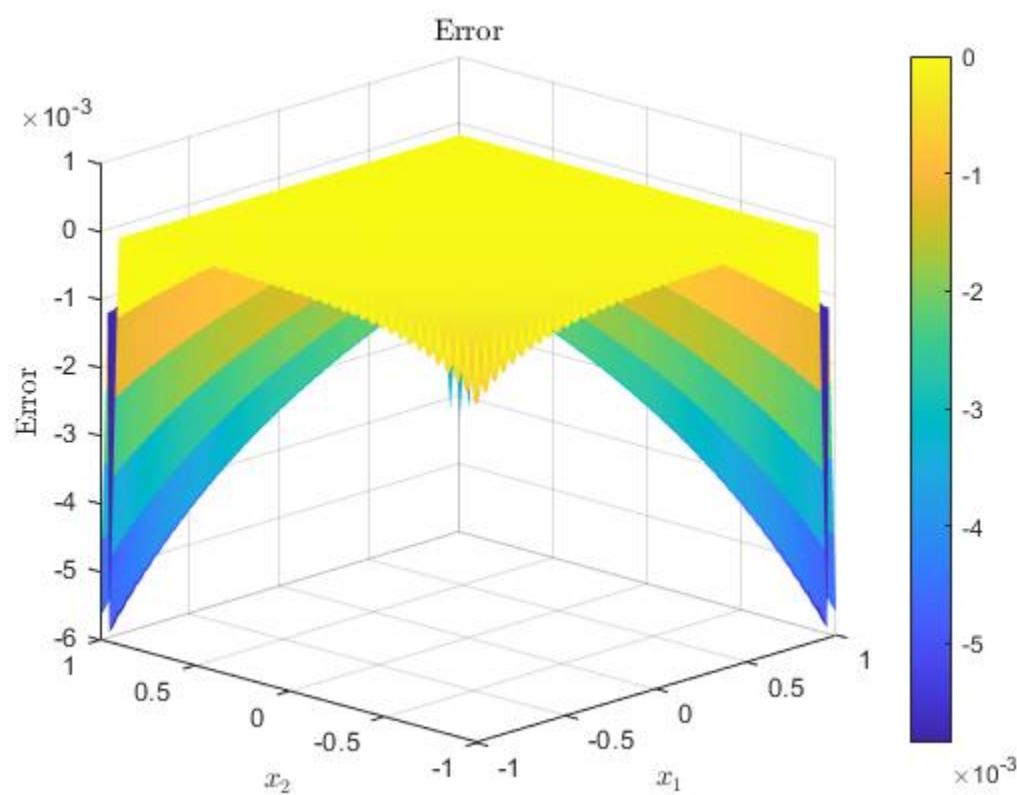
تابع اصلی($g(x)$):



تابع تقریب زده به کمک سیستم فازی ($f(x)$)



خطا:



همانطور که مشخص است حداقل خطا تقریب ما 0.006 بوده است که همانطور که انتظار داشتیم کمتر از 0.1 است.

روش غیر فازی ساز ماکزیمم:

کد مطلب و نتایج:

توضیحات کد و همچنین انواع نمونه های آن در فایل های حل تمرین به طور کامل توضیح داده شده است، پس از آوردن توضیحات اضافی صرف نظر می کنیم.

```
%% First order
% Max Defuzzification

clc
clear
close all

alfa = -1;
beta = 1;
h = 0.05;
N = 41;

x1 = alfa:0.01:beta;
x2 = alfa:0.01:beta;

[~,n1] = size(x1);
[~,n2] = size(x2);
e1 = beta*ones(1,N+1);
e2 = beta*ones(1,N+1);

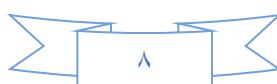
for j = 1:N
    e1(j) = alfa+h*(j-1);
    e2(j) = alfa+h*(j-1);
end

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 = 1;
            q = 1;
        else
            p = 0;
            q = 1;
        end
    end
end
```



```

        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

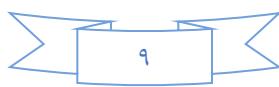
```

[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)')

g_x = 1 ./ (3 + x1 + x2);

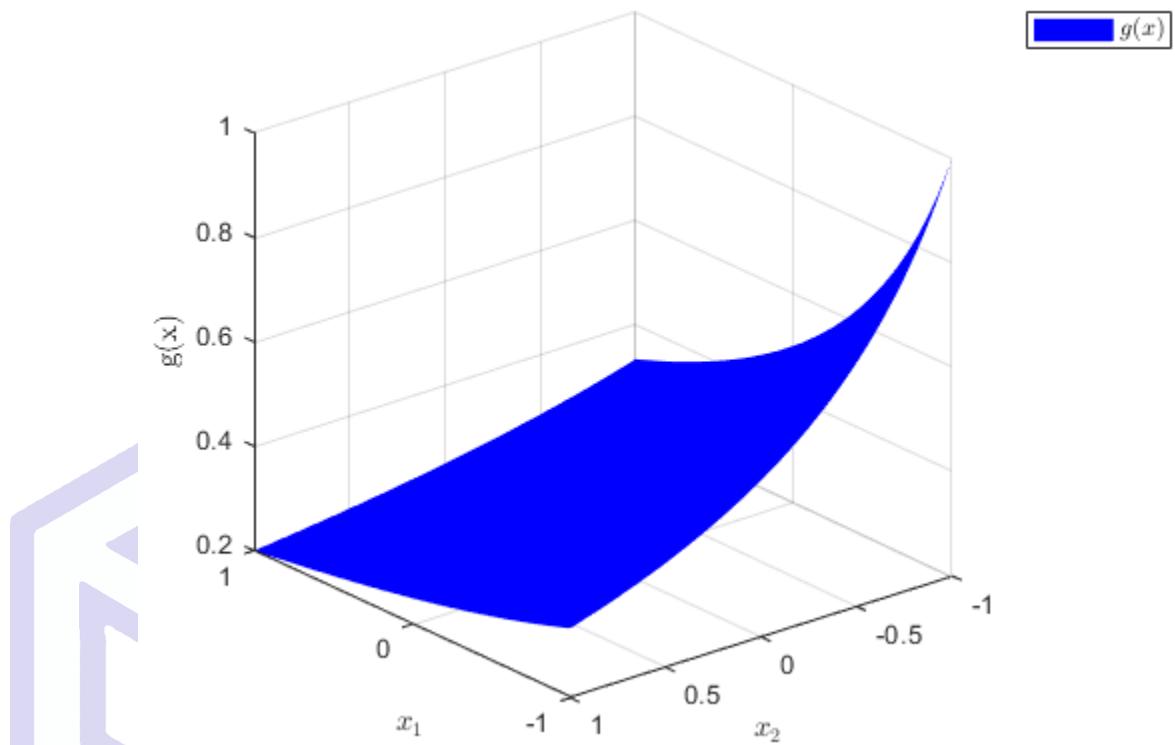
figure(2)
surf(x1, x2, g_x, 'FaceColor', 'b', 'EdgeColor', 'none')
xlabel('\$x_1\$', 'Interpreter', 'latex')
ylabel('\$x_2\$', 'Interpreter', 'latex')
zlabel('g(x)', 'Interpreter', 'latex')
legend('\$g(x)\$', 'Interpreter', 'latex')
grid on

figure(3)
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

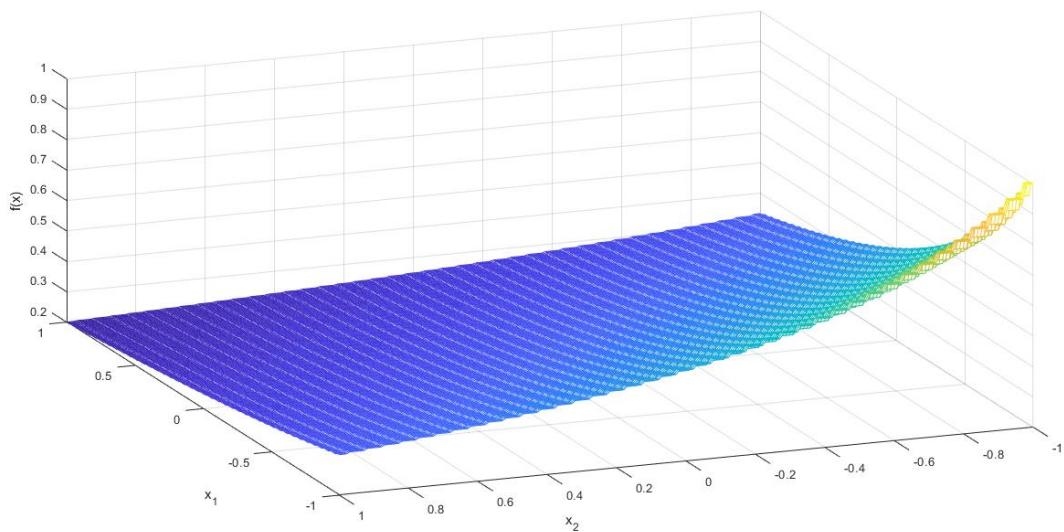


نتایج مطلب:

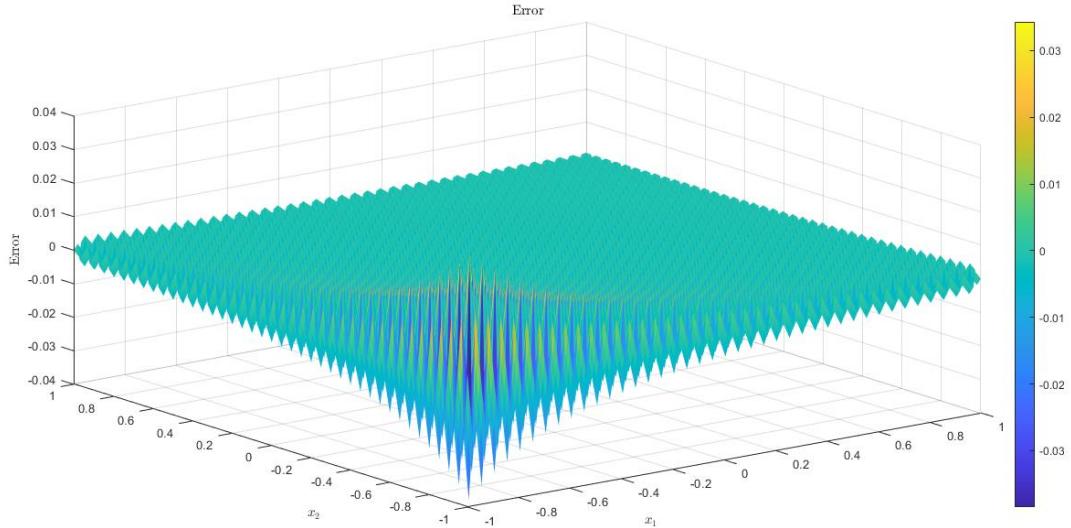
تابع اصلی($g(x)$):



تابع تقریب زده شده به کمک سیستم فازی($f(x)$):



خطا:



همانطور که مشخص است حداقل خطای تقریب ما 0.04 بوده است که همانطور که انتظار داشتیم کمتر از 0.1 است.

استفاده از کران مرتبه دوم:

برای کران مرتبه دوم از قضیه آن استفاده می‌کنیم، که به صورت زیر است:

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

از آنجا که دقت تقریب $\epsilon = 0.01$ ذکر شده و فرض شده که $h_1 = h_2 = h$ می‌نویسیم:

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

بنابراین نیاز است تا مقدار نرم بی نهایت مشتق‌های مرتبه دوم بر حسب x_1 و x_2 حساب کنیم:

$$\left\| \frac{\partial^2 g}{\partial x_1^2} \right\| = \left\| \frac{\partial^2 g}{\partial x_2^2} \right\| = \sup \left| \frac{\partial^2 g}{\partial x_1^2} \right| = \sup \left| \frac{\partial^2 g}{\partial x_2^2} \right| = \sup \left| \frac{2}{(x_1 + x_2 + 3)^3} \right| = 2$$

همانطور که مشخص است چونکه صورت ثابت است، حداکثر مقدار زمانی رخ می دهد که مخرج کمترین مقدار را داشته باشد، که این زمانی اتفاق می افتد که $-1 = x_1 = x_2$. و حداکثر مقدار برابر ۲ خواهد بود، پس می توان n و h را به صورت زیر حساب کرد.

$$h < \sqrt{\frac{\Lambda \varepsilon}{\left\| \frac{\partial^r g}{\partial x_1^r} \right\|_\infty + \left\| \frac{\partial^r g}{\partial x_2^r} \right\|_\infty}} = \sqrt{\frac{\Lambda \times 0/1}{2+2}} = 0.4472 \quad (18)$$

حال با محاسبه حدود h برای صحیح به دست آمدن n آن را معادل $0/25$ در نظر می گیریم، و تعداد توابع تعلق را محاسبه خواهیم کرد. بنابراین داریم:

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

$$N = n + 1 = 9$$

بنابراین برای این روش کافیست ۹ تابع تعلق در نظر بگیریم. همانطور که مشخص است در روش کران مرتبه ۲ توانسته ایم با تعداد توابع تعلق کمتر به دقت دلخواه مان بررسیم، که این یکی از مزیت های مهم روش کران مرتبه ۲ نسبت به روش کران مرتبه اول است، که این مزیت خودش را در سیستم های فازی با توابع تعلق بسیار زیاد نمایان می کند.

روش غیر فازی ساز میانگین:

چون سوال اطلاعاتی راجع به این موضوع نداده است این توابع تعلق را مثلثی در نظر می گیریم. بنابراین این توابع تعلق را به صورت زیر تعیین می کنیم:

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

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

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

غیر فازی ساز میانگین:

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

توضیحات کد و همچنین انواع نمونه های آن در فایل های حل تمرین به طور کامل توضیح داده شده است، پس از آوردن توضیحات اضافی صرف نظر می کنیم.

```
%% Second order

% Mean Defuzzification

clc
clear
close all

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;

% triangular fuzzy membership function
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
        num = num + mu_A_x1.*mu_A_x2;
        den = den + mu_A_x2;
        k = k + 1;
    end
end

g_bar = num/den;
```

```

g_bar(k,1) = 1 / (3 + e_i1(il-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(1)
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)', 'Interpreter', 'latex')
grid on

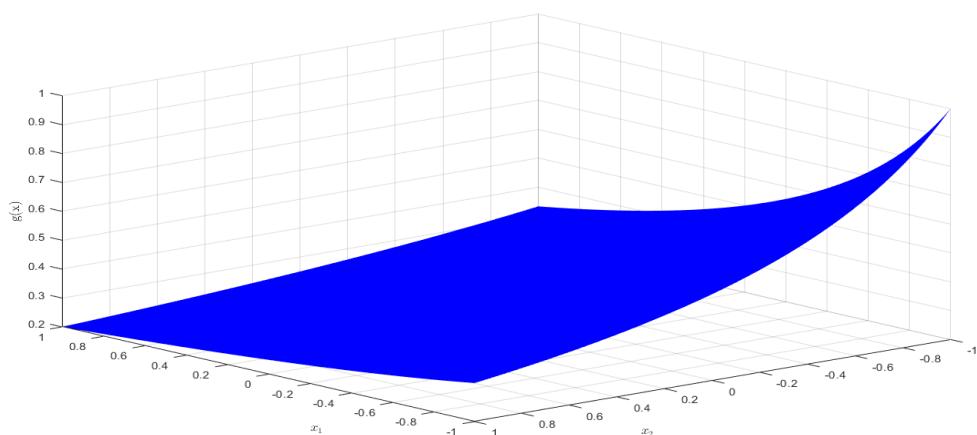
figure(2)
surf(x1, x2, f_x, 'FaceColor', 'r', 'EdgeColor', 'none')
xlabel('$x_1$', 'Interpreter', 'latex')
ylabel('$x_2$', 'Interpreter', 'latex')
zlabel('f(x)', 'Interpreter', 'latex')
legend('f(x)', 'Interpreter', 'latex')
grid on

figure(3)
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

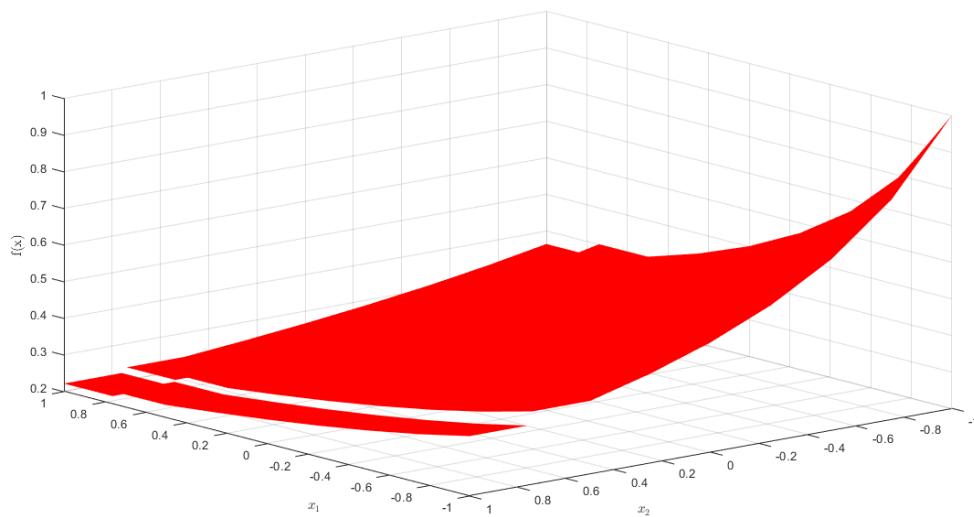
```

نتائج متلب:

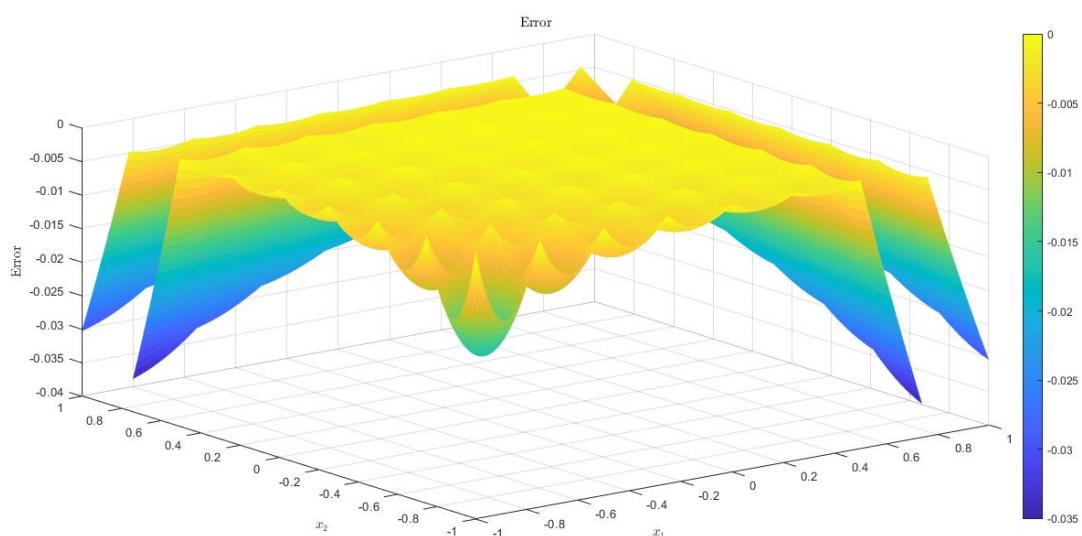
:تابع اصلی($g(x)$)



تابع تقریب زده شده به کمک سیستم فازی ($f(x)$)



خطا:



همانطور که مشخص است حداقل خطای تقریب ما 0.04 بوده است که همانطور که انتظار داشتیم کمتر از 0.1 است.

روش غیر فازی ساز ماکزیمم:

کد مطلب و نتایج:

توضیحات کد و همچنین انواع نمونه های آن در فایل های حل تمرین به طور کامل توضیح داده شده است، پس از آوردن توضیحات اضافی صرف نظر می کنیم.

```
%% Second order
% Max Defuzzification

clc
clear
close all

alfa = -1;
beta = 1;
h = 0.25;
N = 9;

x1 = alfa:0.01:beta;
x2 = alfa:0.01:beta;

[~,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;
        else
            p = 0.5;
            q = 0.5;
        end
    end
end
```

```

    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)')

g_x = 1 ./ (3 + x1 + x2);

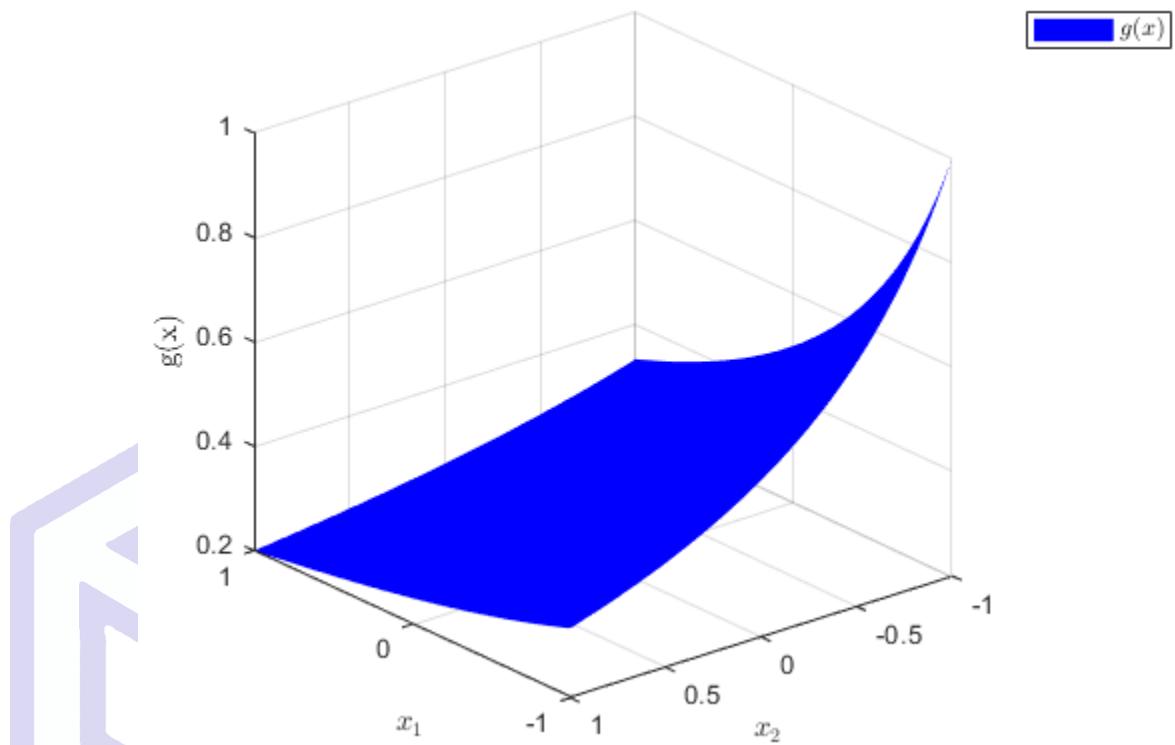
figure(2)
surf(x1, x2, g_x, 'FaceColor', 'b', 'EdgeColor', 'none')
xlabel('$x_1$', 'Interpreter', 'latex')
ylabel('$x_2$', 'Interpreter', 'latex')
zlabel('g(x)', 'Interpreter', 'latex')
legend('$g(x)$', 'Interpreter', 'latex')
grid on

figure(3)
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

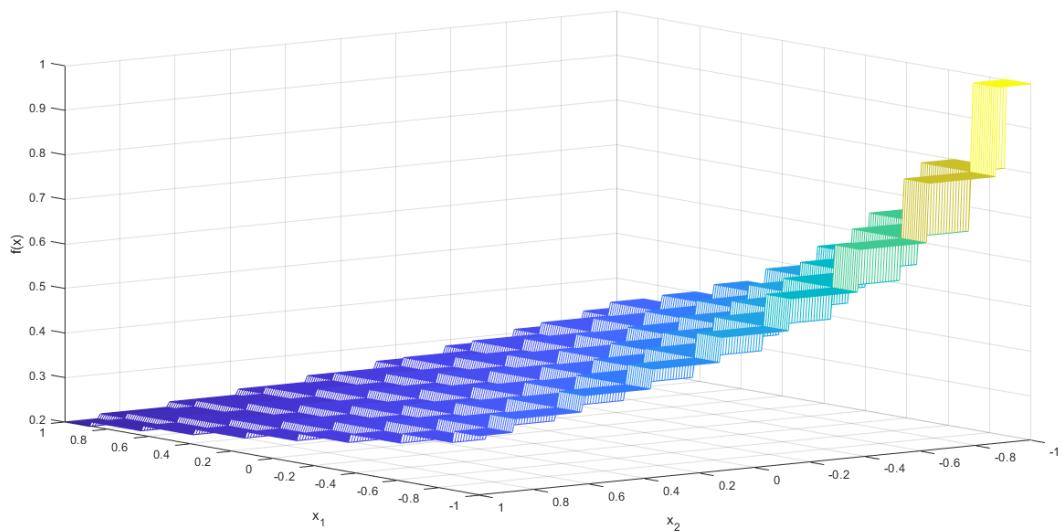
```

نتایج مطلب:

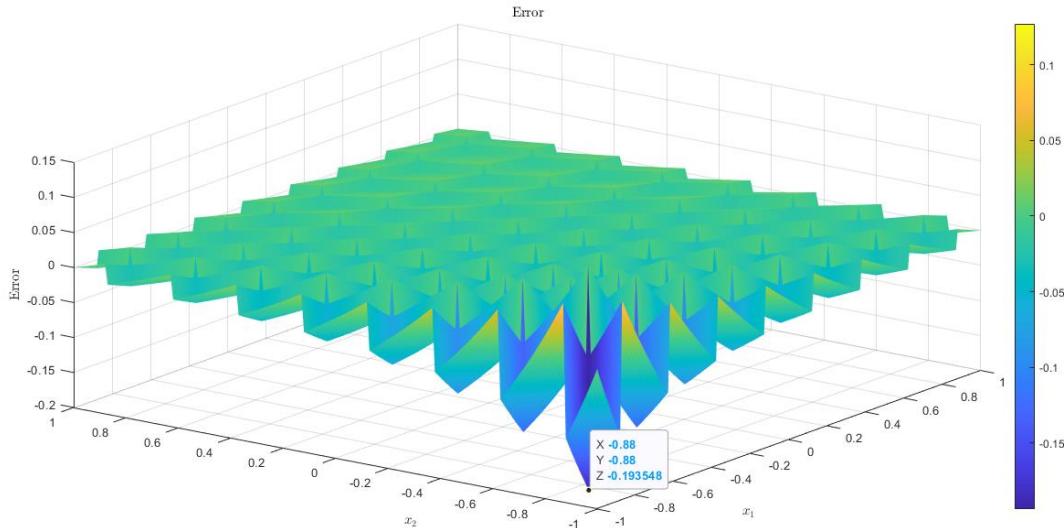
تابع اصلی($g(x)$):



تابع تقریب زده به کمک سیستم فازی($f(x)$):



خطا:



همانطور که مشخص است حداقل خطا تقریب ما $0.2 \cdot 0.1$ بوده است که بیشتر از 0.1 است، پس زمانی که از غیر فازی ساز ماکزیمم استفاده کنیم نمی توانیم به دقت مد نظر برسیم.

راجع به تفاوت روش کران مرتبه اول و کران مرتبه دوم در بالا صحبت کردیم.

مقایسه غیر فازی ساز میانگین و ماکزیمم:

- کاربرد: انتخاب بین غیر فازی سازهای میانگین و ماکزیمم به میزان تأثیر بخشی هر کدام در مسئله مورد بررسی وابسته است. اگر تأثیر میانگین مهم باشد، غیر فازی ساز میانگین مناسب است. اگر تأثیر ماکزیمم مهم باشد، غیر فازی ساز ماکزیمم بهتر است.

- حساسیت به اطلاعات پرت: غیر فازی ساز میانگین حساسیت کمتری به اطلاعات پرت دارد، زیرا میانگین تأثیر کمتری از ارتفاع مقادیر ناهمگن دارد. اما غیر فازی ساز ماکزیمم حساسیت زیادی به اطلاعات پرت دارد، زیرا یک مقدار پرت می‌تواند بر تصمیم نهایی تأثیر زیادی بگذارد.

- تعداد ورودی‌ها: در مواردی که تعداد زیادی از ورودی‌ها در نظر گرفته می‌شود، ممکن است محاسبه میانگین محاسباتی زیادی را نیاز داشته باشد، در حالی که محاسبه ماکزیمم نسبت به تعداد بیشتر ورودی‌ها مقرن به صرفه‌تر باشد.

سوال ۲

یک برنامه کامپیوترا برای پیاده سازی روش جستجو بنویسید. برای کامل و همه منظوره بودن برنامه، می توانید روش پر کردن خانه های خالی جدول جستجو را هم ذذ آن در نظر بگیرید. برنامه خود را برای مساله پیشگویی سری زمانی Mackey-Glass که در بخش ۳.۱۲ مرجع [۲] آورده شده است را به کار گرفته و اجرا کنید. نتایج را به شکل مناسب نشان دهید.

کد مربوطه در مرجع ۱ موجود است که در اینجا از آن استفاده شده است، در ادامه توضیحات آن داده خواهد شد.

```
clc
clear
close all

%% Data generation by Mackey-Glass chaotic time series

% Total number of sampling
n = 900;

% 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
            [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
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)]);

```

توضیحات کد:

این کد متلب در واقع یک سیستم استنتاج فازی (FIS) بر اساس داده‌های سری زمانی (Mackey-Glass) ایجاد می‌کند. در واقع این FIS برای پیش‌بینی مقدار بعدی در سری زمانی با استفاده از منطق فازی طراحی شده است.

روند کد به شکل زیر می‌باشد:

❖ تولید داده :

یک سری زمانی فراز و نشیب Mackey-Glass ایجاد شده و در متغیر x ذخیره می‌شود.

یک مجموعه داده (dataset_1) با استفاده از یک پنجره از سری زمانی Mackey-Glass ایجاد می‌شود.

❖ طراحی سیستم فازی :

دو حالت در نظر گرفته شده‌اند، هرکدام با تعداد مختلفی از توابع عضویت برای ورودی‌ها. توابع عضویت برای هر ورودی تعریف شده و قوانین بر اساس مجموعه داده ایجاد می‌شوند. درجه عضویت برای هر قاعده با استفاده از توابع عضویت مثلثی یا مربعی محاسبه می‌شود.

قوانین اضافی حذف شده و یک مجموعه نهایی از قوانین (Rules_total) به دست می‌آید.

یک سیستم استنتاج فازی از نوع (fis) Mamdani با ورودی‌ها، خروجی‌ها، توابع عضویت و قوانین ایجاد می‌شود.

❖ استنتاج فازی :

از سیستم استنتاج فازی برای پیش‌بینی ۳۰۰ نقطه بعدی مجموعه داده استفاده می‌شود.

پیش‌بینی‌ها (jadval_prediction) با مقادیر واقعی مقایسه می‌شوند و نتایج به نمودار کشیده می‌شوند.

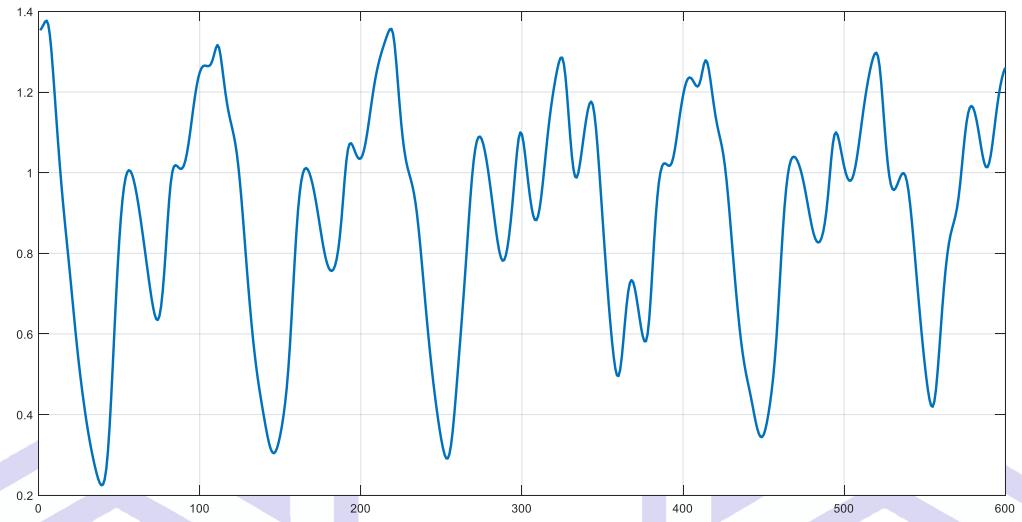
❖ تصویرسازی توابع عضویت :

توابع عضویت برای ورودی اول به منظور تصویرسازی نمایش داده می‌شوند.

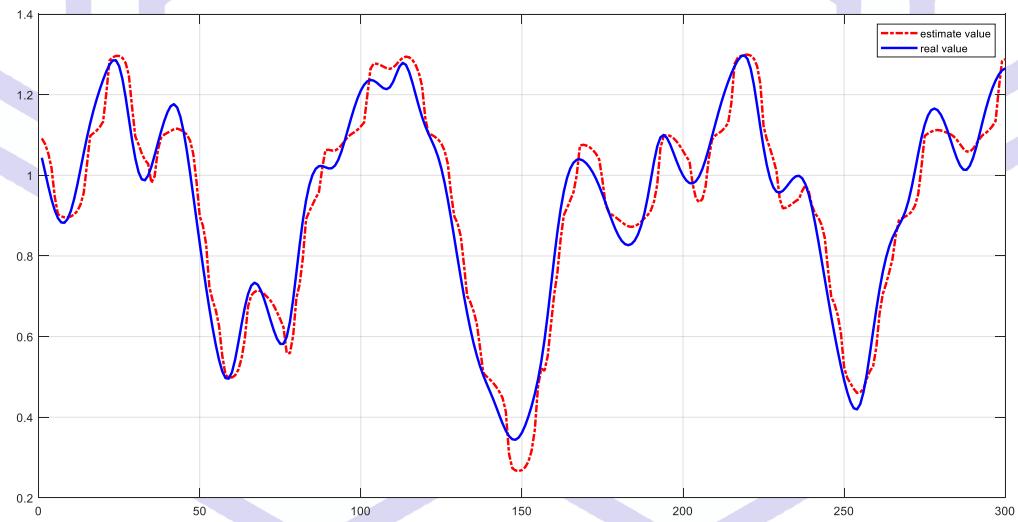
در واقع این کد از منطق فازی برای مدل‌سازی و پیش‌بینی سری زمانی Mackey-Glass استفاده می‌کند.

نتایج مطلب:

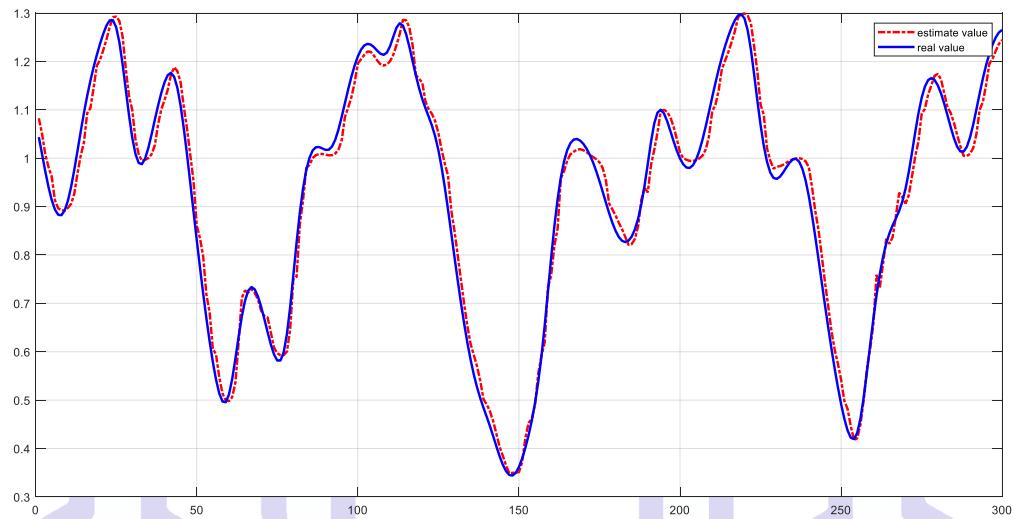
نمایش مجموعه داده های ایجاد شده:



از این مجموعه داده قرار است ۳۰۰ داده دوم را پیش بینی کنیم، نتایج پیش بینی اول ما به صورت زیر است:



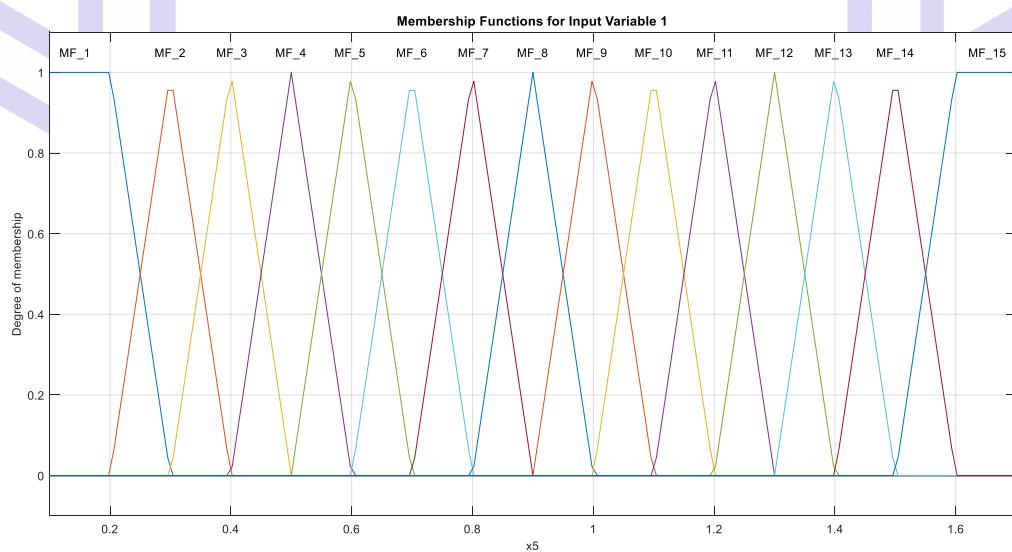
نتایج پیش بینی دوم که دقیق تر است (بالا به آن اشاره کردیم) به صورت زیر می باشد:



تعداد نهایی قوانین نیز برای هر کدام از متغیر های ورودی به شکل زیر است:

Final rules for 15 membership functions for each input variables

تابع تعلق مثلثی ما به شکل زیر می باشند:



همانطور که انتظار داشتیم ۱۵ تابع تعلق (۱۵ قانون) به ازای هر ورودی داریم.

سوال ۳

فرض کنید یک سیستم با معادله دیفرانسیل آورده شده در **معادله ۱** دارید که قرار است توسط یک شناساگر فازی شناسایی شود.

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

که در آنتابع نامعلوم $[g[u(k)]$ براساس **معادله ۲** تعریف می شود.

$$g(u) = 0.6 \sin(\pi u) + 0.3 \sin(3\pi u) + 0.1 \sin(5\pi u) \quad (2)$$

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

$$f(x) = \frac{\sum_{l=1}^M \bar{y}^l \left[\prod_{i=1}^n \exp \left(- \left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}{\sum_{l=1}^M \left[\prod_{i=1}^n \exp \left(- \left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]} \quad (3)$$

کد مربوطه در مرجع ۲ موجود است که در اینجا از آن استفاده شده است، در ادامه توضیحات آن داده خواهد شد.

```

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
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 (1,1))).^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 مطلب یک مدل تطبیقی (Adaptive) فازی برای تقریب توابع ناشناخته از داده‌های ورودی-خروجی ایجاد می‌کند. روند کد به شکل زیر می‌باشد:

❖ تعیین پارامترها :

M : تعداد توابع عضویت برای ورودی‌ها.

num_training : تعداد داده‌های آموزش

متغیرها مانند x ، y ، g_{bar} و σ برای ذخیره اطلاعات مربوط به توابع عضویت ورودی‌ها و پارامترهای مدل.

x ، y ، u ، u_{hat} و f_{hat} برای ذخیره خروجی‌ها و تخمین‌های مدل.

همچنین انتخاب یک نقطه تصادفی برای $u(1)$ انتخاب شده و $u(1)$ محاسبه می‌شود.

❖ طراحی اولیه سیستم فازی :

در این مرحله محاسبه اطلاعات اولیه مربوط به توابع عضویت ورودی‌ها براساس یک اندازه‌گیری اولیه از داده‌ها صورت می‌گیرد. همچنین تخمین مقادیر اولیه برای توابع عضویت، مراکز و انحراف‌های استاندارد انجام می‌شود.

❖ آموزش مدل تطبیقی فازی :

در این مرحله آموزش مدل تطبیقی فازی بر اساس داده‌های ورودی-خروجی و همچنین محاسبه f_{hat} براساس مدل فازی و بهروزرسانی توابع عضویت و پارامترهای مدل بر اساس معیارهای یادگیری انجام می‌شود.

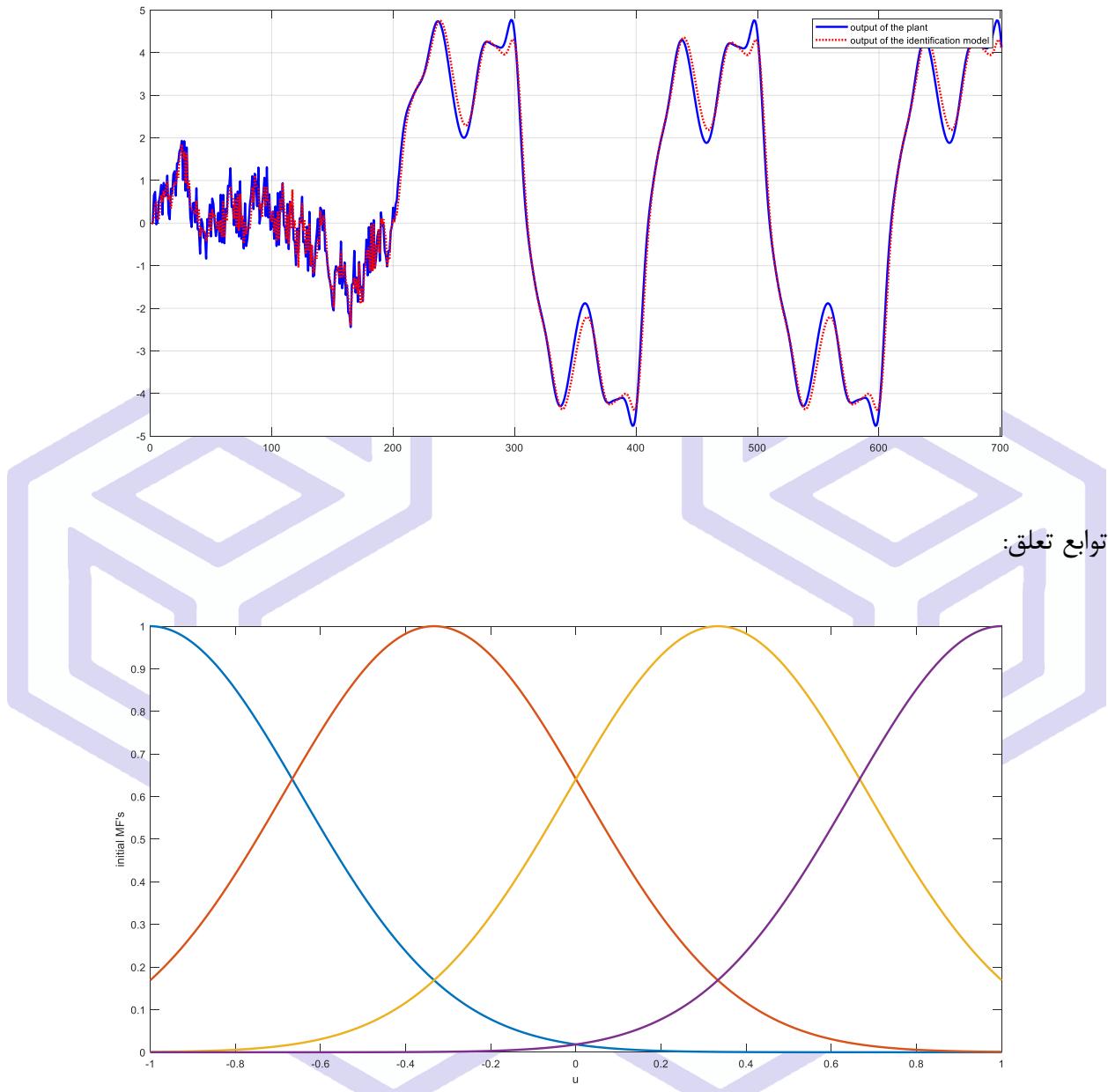
❖ استفاده از مدل برای تقریب توابع ناشناخته :

در این مرحله از مدل آموزش دیده برای تخمین خروجی‌ها بر اساس ورودی‌های جدید استفاده شود.

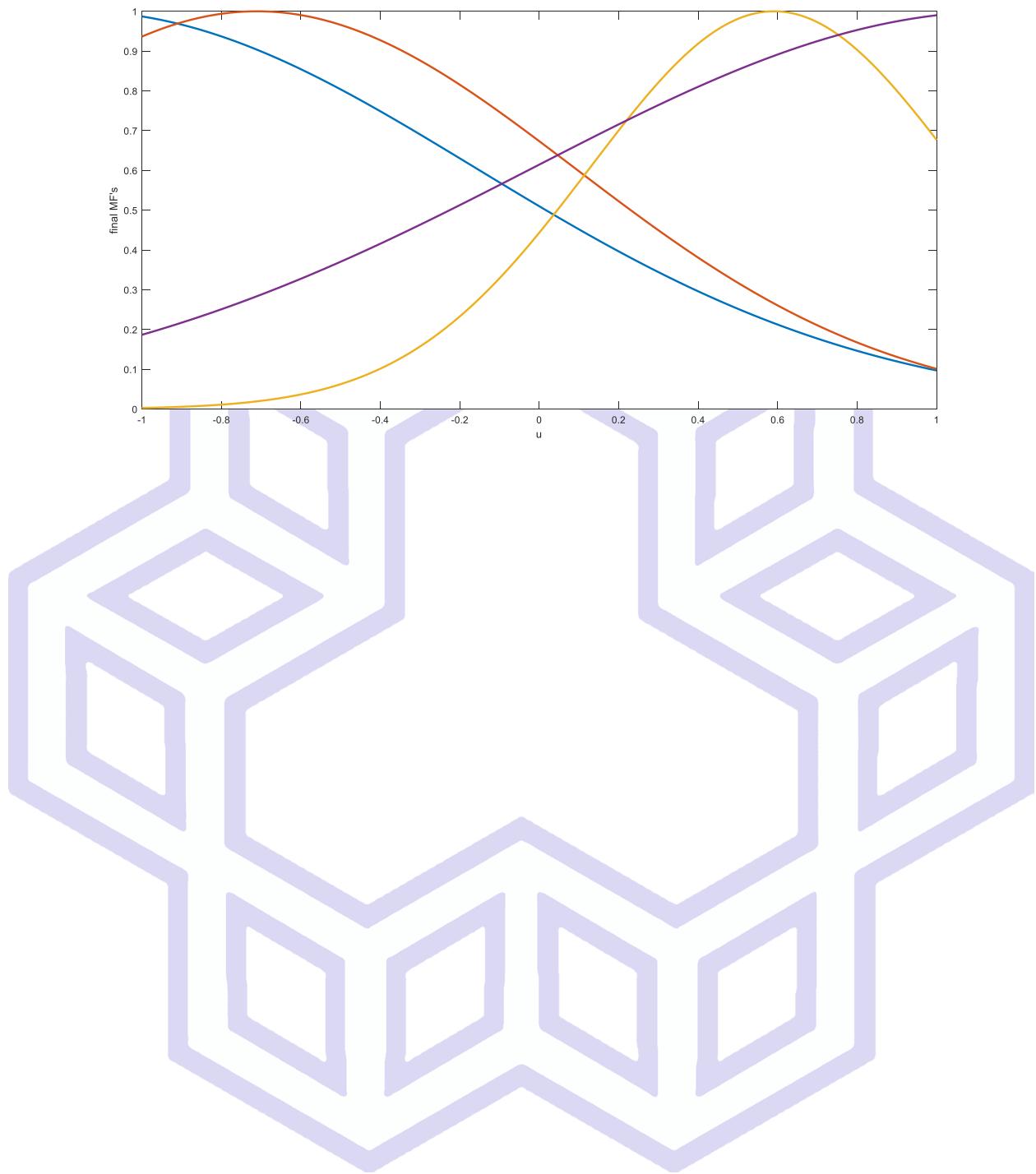
همچنین در انتهای نمایش خروجی مدل و مقادیر واقعی در یک نمودار رسم شده و همینطور نمایش توابع عضویت اولیه و نهایی برای ورودی‌ها انجام می‌شود.

نتایج مطلب:

مقایسه خروجی شناسایی سیستم و خروجی واقعی سیستم:



تابع تعلق:



سوال ۴

به سوالات زیر از مبحث درخت تصمیم پاسخ دهید:

بخش ۱

با بهره گیری از آموزش ارائه شده در خصوص کدنویسی درخت تصمیم از ابتدا، بدون استفاده از کتابخانه سایکیت لرن دستوراتی بنویسید که درخت تصمیم یک مجموعه داده مربوط به بیماری کرونا که در این پیوند موجود است را خروجی دهد. اگر می توانید این کار را به صورتی انجام دهید که اطلاعات بیشتری را در خروجی درخت تصمیم خود دریافت کنید. لازم است که تحلیل منطقی از نتیجه درخت تصمیم خود ارائه کنید. می توانید این کار را با الگوگرفتن از موارد گفته شده در ویدیوهای کلاس و این پیوند انجام دهید.

از آنجا که گفته شده از کتابخانه های سایکیت لرن استفاده نشود، پس با تعریف کردن آنتروپی و مشابه آنچه در کلاس حل تمرین گفته شد می توان درخت تصمیم را ایجاد کرد.

ابتدا کتابخانه های مورد نیاز را فراخوانی می کنیم:

```
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
```

سپس دیتاست داده شده را فراخوانی می کنیم و آنرا نمایش می دهیم:

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1zjjjYSr3qXYZg2_vykwhmjBF6NOUmI3jl
#
https://drive.google.com/file/d/1zjjjYSr3qXYZg2_vykwhmjBF6NOUmI3jl/view?usp=sharing
```

```
data = pd.read_csv('/content/decision_tree')
data
```

نتایج:

	Fever	Cough	Breathing issues	Infected
0	No	No		No
1	Yes	Yes		Yes
2	Yes	Yes		No
3	Yes	No		Yes
4	Yes	Yes		Yes
5	No	Yes		No
6	Yes	No		Yes
7	Yes	No		Yes
8	No	Yes		Yes
9	Yes	Yes		No
10	No	Yes		No
11	No	Yes		Yes
12	No	Yes		No
13	Yes	Yes		No

سپس با استفاده از فرمول آنتروپی ، مقدار گین ویژگی های مختلف را برای مشخص کردن root node بدهست آورده ایم.

```
labels = data['Infected']
len(labels), labels.unique(), labels.value_counts()
p = labels.value_counts() / len(labels)
-sum(p * np.log2(p))
```

تعریف آنتروپی به صورت تابع:

```
def entropy(labels):
    p = labels.value_counts() / len(labels)
    return -sum(p * np.log2(p))

data['Infected'].value_counts()
```

: محاسبه آنروپی child

```
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
0.9460794641311808

```

: محاسبه آنتروپی parent :

```
entropy(data['Infected'])
```

: نتایج :

```
0.9852281360342515
```

: information gain تعریف کردن

```

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)

```

تعریف کردن information gain به صورت تابع:

```

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

```

```

✓ 0s   information_gain(data, 'Fever', 'Infected')
      0.12808527889139443

✓ 0s [14] information_gain(data, 'Cough', 'Infected')
      0.0391486719030707

✓ 0s [15] information_gain(data, 'Breathing issues', 'Infected')
      0.39603884492804464

✓ 0s [16] data.iloc[:, :-1].columns
      Index(['Fever', 'Cough', 'Breathing issues'], dtype='object')

✓ 0s [17] [information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns]
      [0.12808527889139443, 0.0391486719030707, 0.39603884492804464]

✓ 0s [18] np.argmax([information_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns])
      2

```

در بالا مقدار information gain را برای ویژگی های مختلف به دست آورده ایم، ویژگی که بیشترین information gain را داشته باشد به عنوان root node انتخاب می کنیم که همانطور که مشخص است ویژگی ستون ۲ یعنی ویژگی مشکل تنفسی دارای بیشترین گین می باشد.

حال باید درخت تصمیم را همانطور که در کلاس حل تمرین نیز به آن اشاره شده به صورت تابع تشکیل دهیم:

تعریف node به صورت کلاس:

```

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}")'

```

در واقع این کد یک کلاس به نام Node ایجاد می کند که از آن برای ساختار داده درخت تصمیم استفاده می شود. سپس یک تابع به نام make_tree نیز تعریف شده است که از این کلاس Node برای ساخت درخت تصمیم

با توجه به Information Gain در هر ویژگی استفاده می‌کند. این کلاس ویژگی‌ها را به صورت ورودی دریافت می‌کند و تابع `repr` برای نمایش متنی مناسب گره‌ها است.

تعريف درخت به صورت تابع:

```
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 feature
    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 تابع یک درخت تصمیم را با استفاده از رویکرد یازگشته، 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 target ایجاد می‌شود. اگر موارد بالا نقصانی ایجاد نکنند، برای هر ویژگی محاسبه می‌شود. با استفاده از یک رویکرد حریصانه (greedy)، ویژگی Information Gain 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.

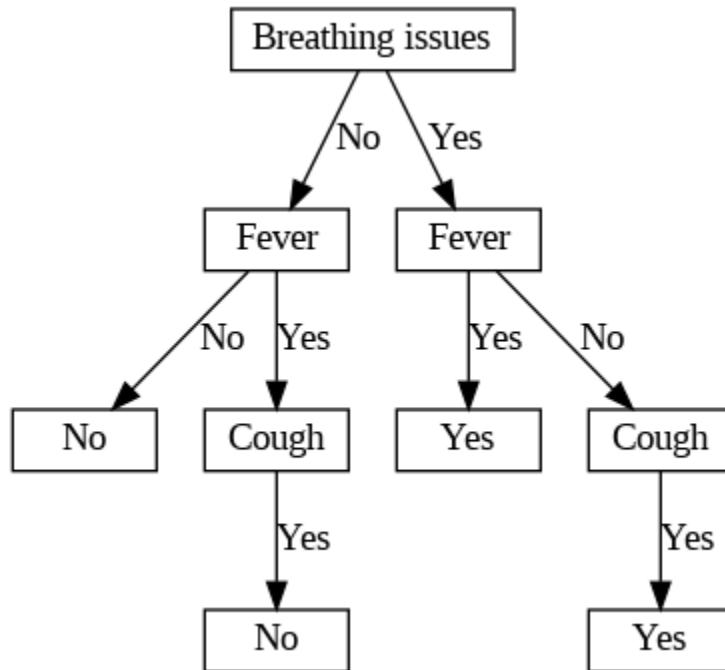
ساخت درخت تصمیم برای دیتاست مذکور:

```
[21] 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="Yes")})})
[22] tree.feature
      'Breathing issues'
```

مشکلی که وجود دارد این است که وقتی breathing_issue به عنوان root node انتخاب می شود، برای مقادیر فیچر Yes و No هر دو information gain بیشترین مقدار Fever را دارد، که اگر به دیتاست نگاه کنیم معلوم است که Cough در این حالت تاثیر ندارد.

برای رسم نمودار درختی به شغل زیر عمل 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)
```



همانطور که مشخص است وقتی **breathing_issue** به عنوان root node انتخاب می‌شود، برای Yes و No هر دو **Fever** بیشترین مقدار information gain را دارد. بنابراین چه در صورت yes و در صورت no بودن **Cough** و **fever** می‌بینیم. در صورت No بودن **breathing_issue** و **fever** و **cough** بررسی می‌شود و در این وضعیت در صورت **No** بودن **fever** شخص کرونایی نیست. از طرفی دیگر نیز در صورت **breathing_issue** بودن **yes** و در این وضعیت قطعاً شخص دارای کرونا می‌باشد. در غیر این صورت ویژگی **cough** بررسی می‌شود و در صورت **yes** بودن آن شخص کرونایی تشخیص داده می‌شود. اما در حالتی که **fever** و **breathing_issue** باشد ولی **cough** و **no** باشند، دلیلی بر تشخیص بیماری نمی‌باشد.

بخش ۲

به انتخاب خود یکی از دو مجموعه داده Drugs و cancer_breast_load را انتخاب کنید و کار طبقه بندی با درخت تصمیم را با استفاده از دستوراتی که آموزش دیده اید (کدنویسی از ابتدا و یا کدنویسی با کمک کتابخانه سایکیت لرن) انجام دهید. لازم است که توضیحات مختصری از مجموعه داده و منطق درخت تصمیم تولیدشده بنویسید. منطق معیاری که استفاده می کنید و نتایج آن در قسمت های مختلف را به صورت کامل تحلیل کنید. هم چنین، مسیر مربوط به دو نمونه از داده های مجموعه آزمون را نشان داده و تحلیل کنید. اگر از فرآپارامتر خاصی مانند فرآپارامترهای مخصوص هرس کردن استفاده می کنید لازم است که حداقل دو مقدار بزرگ و کوچک برای آن در نظر بگیرید و تحلیل خود از تأثیر آن روی نتیجه نهایی را بنویسید.

از مجموعه داده سرطان سینه استفاده می کنیم.

مشابه روندی که در کلاس حل تمرین برای نوشتن به کمک کتابخانه سایکیت لرن گفته شد عمل می کنیم:

```
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=93)

# Create a decision tree classifier
max_depth_values = [4 ,12]
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}' )
```



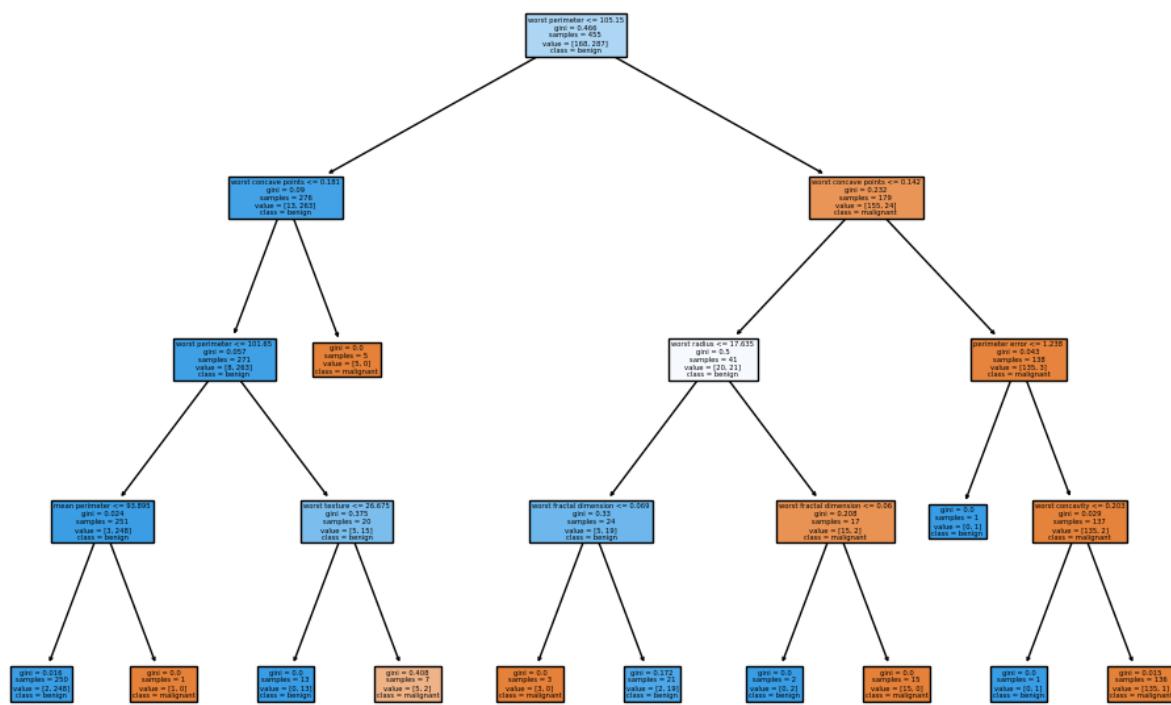
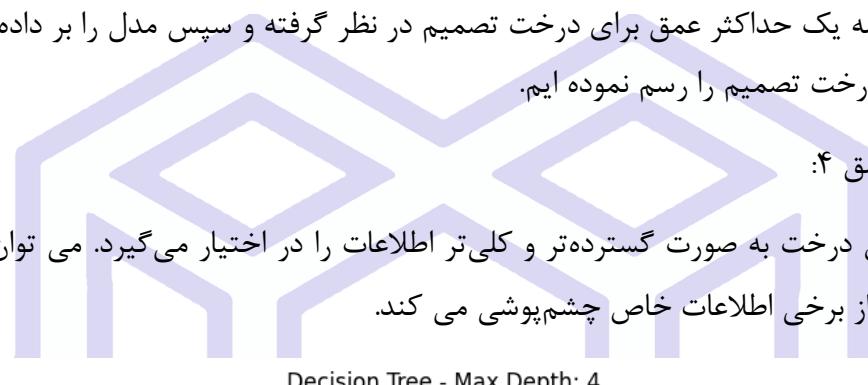
```
plt.savefig(f'decision_tree_max_depth_{max_depth}.png')
plt.show()
```

در این بخش اثر حداکثر عمق را بررسی نموده ایم، یکبار حداکثر عمق را برابر ۴ و یکبار برابر ۱۲ قرار داده ایم.

ابتدا ویژگی ها و هدف را از دیتاست مشخص کرده و داده ها را به نسبت ۱ به ۵ برای تست انتخاب نموده ایم، سپس در هر مرحله یک حداکثر عمق برای درخت تصمیم در نظر گرفته و سپس مدل را بر داده ها فیت کرده و در نهایت نمودار درخت تصمیم را رسم نموده ایم.

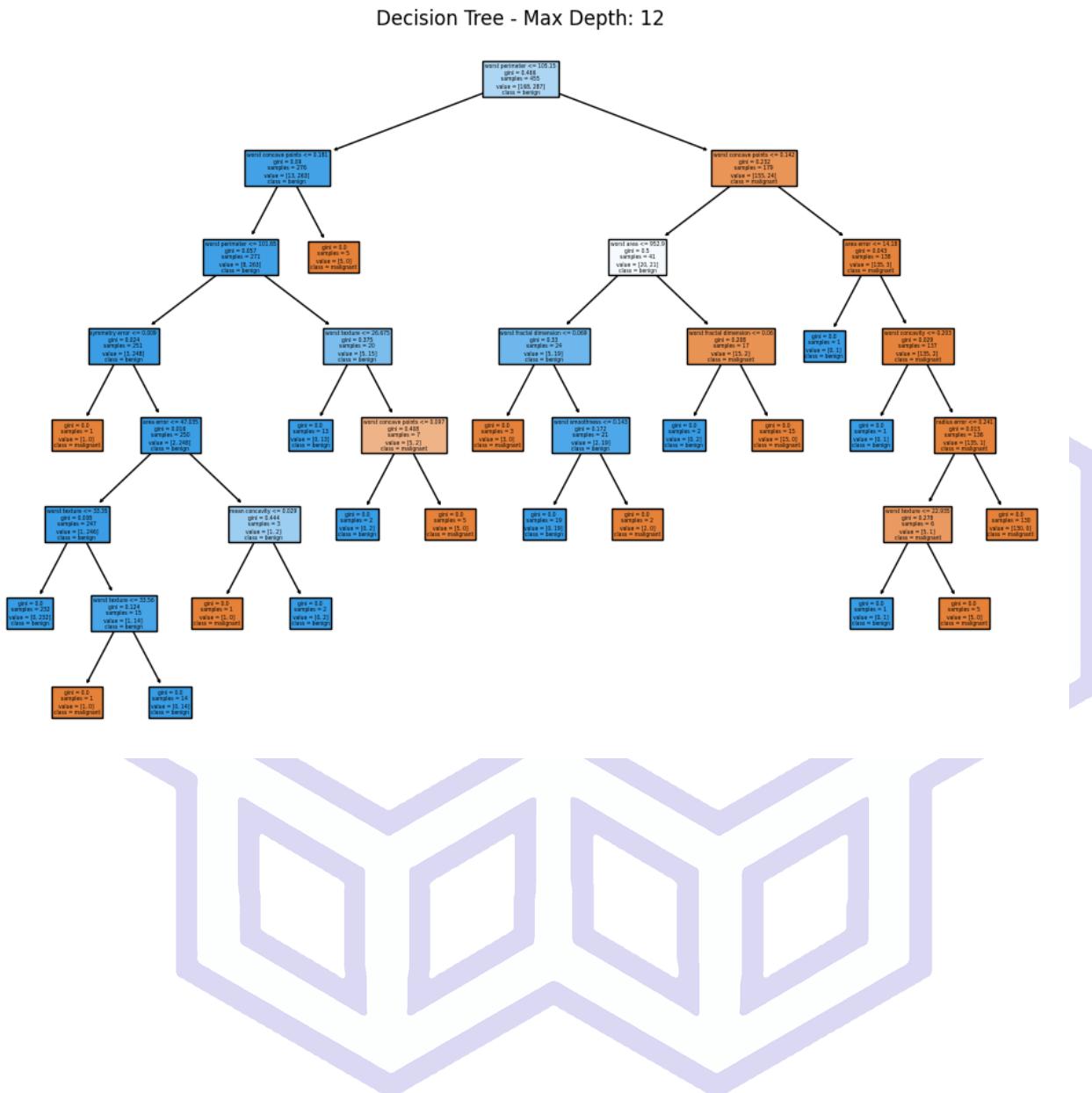
مدل با حداکثر عمق ۴:

می توان گفت این درخت به صورت گستره‌تر و کلی تر اطلاعات را در اختیار می‌گیرد. می توان گفت به خاطر کمتر بودن عمق، از برخی اطلاعات خاص چشم‌پوشی می‌کند.



مدل با حداکثر عمق ۱۲:

این درخت نسبت به درخت با حداکثر عمق ۴ اطلاعات دقیق‌تری را در اختیار می‌گیرد. این عمق بیشتر ممکن است منجر به یادگیری و حفظ جزئیات کمتری در مورد داده‌ها شود و منجر به overfitting شود و یا ممکن است باعث افزایش دقت در دسته‌بندی شود.



: `(ccp_alpha)` تغییر پارامتر هرس کردن

همانند بخش قبل عمل کرده فقط اینبار پارامتر `ccp_alpha` را به جای `Max_depth` تغییر می دهیم:

```
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=93)

# Create a decision tree classifier
ccp_alpha_values = [0.0, 0.01, 0.02]
for ccp_alpha in ccp_alpha_values:
    clf = DecisionTreeClassifier(ccp_alpha=ccp_alpha)

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

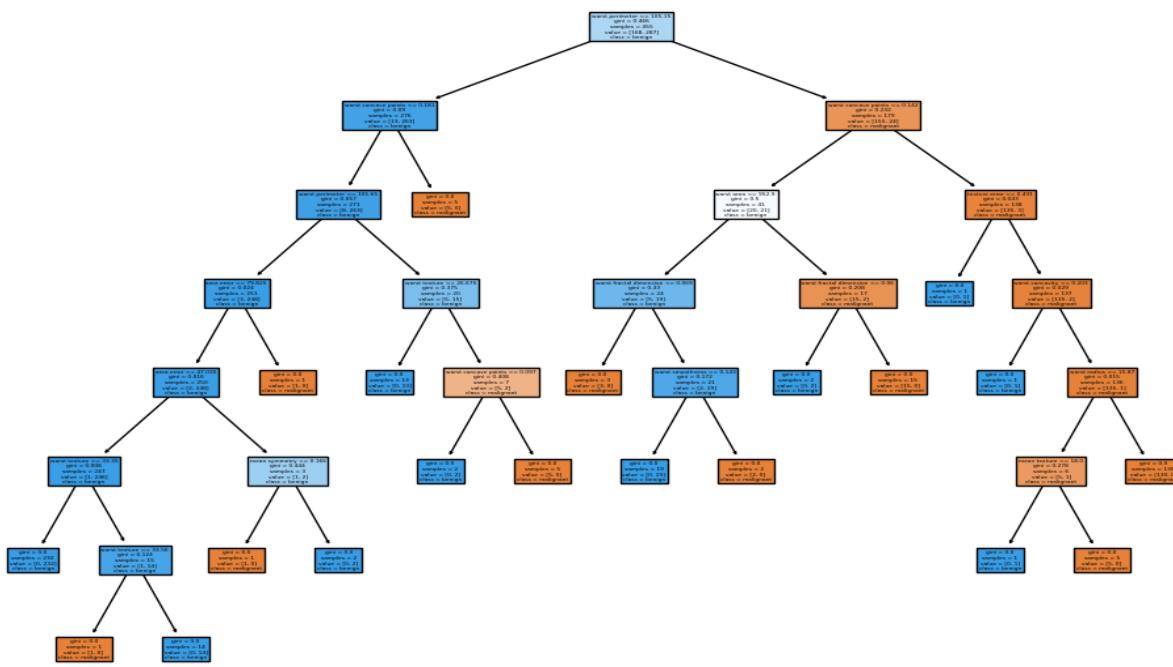
    # Plot the decision tree
    plt.figure(figsize=(12, 8))
    plot_tree(clf, filled=True, feature_names=data.feature_names,
    class_names=data.target_names)
    plt.title(f'Decision Tree - ccp_alpha: {ccp_alpha}')
    plt.savefig(f'decision_tree_ccp_alpha_{ccp_alpha}.png')
    plt.show()
```

پارامتر هرس کردن به شرط انتخاب مناسب سبب می شود که عمق مناسبی برای درخت تصمیم در نظر گرفته شود و سرعت ما افزایش یابد و همچنین می تواند از overfitting جلوگیری کند.

در زیر نمودار سه درخت تصمیم یکی بدون استفاده از پارامتر هرس کردن، یکی با پارامتر هرس کردن ۰.۰۱ و یکی با پارامتر هرس کردن ۰.۰۲ را نمایش داده ایم.

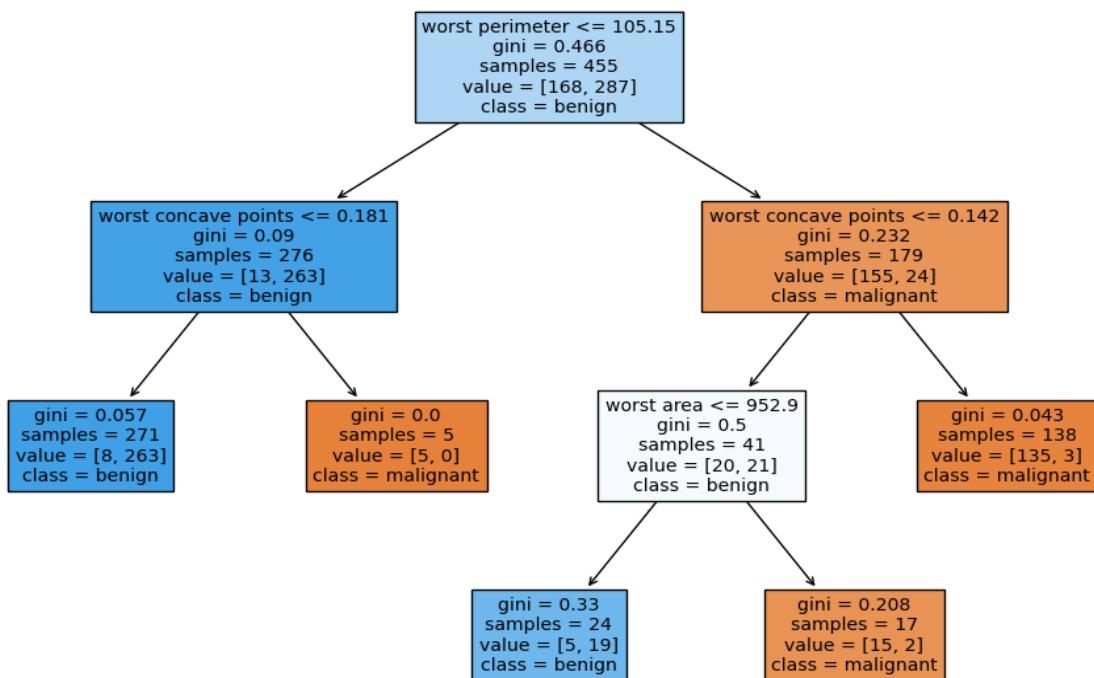
`Ccp_alpha=0:`

Decision Tree - ccp_alpha: 0.0



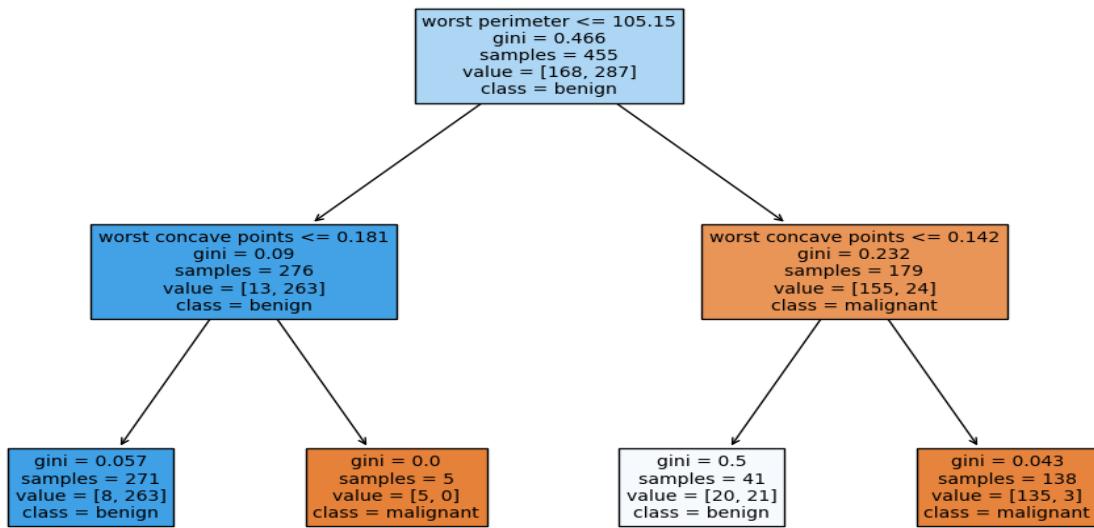
Ccp_alpha=0.01:

Decision Tree - ccp_alpha: 0.01



Ccp_alpha=0.02:

Decision Tree - ccp_alpha: 0.02



همانطور که مشخص است انتخاب درست پارامتر `ccp_alpha` می تواند عمق درخت را به خوبی کاهش دهد و دقیق‌تر را افزایش داده و از overfitting جلوگیری کند. اما انتخاب نادرست آن می تواند به کلی ساختار درخت تصمیم‌گیری را به هم بربزد، این پارامتر معمولاً یک مقدار کم حدود ۰.۰۱ در نظر گرفته می شود، انتخاب زیاد سبب از بین رفتن کامل عمق درخت تصمیم می شود. مثلاً به مقاله زیر توجه کنید:

Ccp_alpha=0.5:

Decision Tree - ccp_alpha: 0.5

```

gini = 0.466
samples = 455
value = [168, 287]
class = benign
  
```

: ccp_alpha اثرات بررسی

برای درک اثر `ccp_alpha`, باید مفهوم پرواز هزینه (Cost-Complexity Pruning) را بفهمید. این روش با افزودن یک جریمه به هر تقسیم درخت، ترجیح می‌دهد تا از تقسیم‌هایی که باعث افزایش در هزینه و کم شدن در پیش‌بینی‌های درست می‌شوند، اجتناب کند. `ccp_alpha` همان جریمه‌ای است که به هر تقسیم افزوده می‌شود.

: `ccp_alpha` اثرات ممکن

❖ کاهش اندازه درخت:

`ccp_alpha` بزرگتر موجب می‌شود که الگوریتم ترجیح دهد تا از تقسیم‌های غیرضروری خودداری کند. این می‌تواند منجر به ایجاد یک درخت با تعداد گره کمتر و ساختار ساده‌تر شود.

❖ کاهش بیش‌برازش:

با افزایش `ccp_alpha`, الگوریتم تمایل دارد تا از تقسیم‌هایی که ممکن است به ساختار غیرضروری و بیش‌برازش منجر شوند، اجتناب کند. این کاهش بیش‌برازش بهبود قابلیت تعمیم‌پذیری مدل را ارتقاء می‌دهد.

❖ تعادل بین دقت و سادگی:

با تنظیم مناسب `ccp_alpha`, می‌توانید تعادلی بین دقت مدل و سادگی آن را برقرار کنید.

❖ تاثیر بر هزینه زمانی:

در برخی مواقع، استفاده از `ccp_alpha` می‌تواند منجر به ساخت درخت‌های با هزینه زمانی کمتر شود، زیرا درخت‌های ساده‌تر و کوچک‌تر سریع‌تر تولید می‌شوند.

برای انتخاب بهترین مقدار `ccp_alpha`, می‌توانید از روش‌های ارزیابی مانند متقابل اجتماعی (cross validation) استفاده کنید. این روش به شما کمک می‌کند تا مقدار بهینه برای `ccp_alpha` را پیدا کنید که باعث بهبود عملکرد مدل شود.

پیش‌بینی مسیر درخت تصمیم برای ۲ داده از داده‌های تست:

توضیح کد:

دو نمونه از مجموعه آزمون با استفاده از `X_test` تعریف شده‌اند و در `sample1` و `sample2` ذخیره شده‌اند.
مدل با استفاده از `predict`, برچسب‌های پیش‌بینی شده برای دو نمونه را محاسبه کرده و در `prediction1` و `prediction2` ذخیره کرده است.

یک عنوان با اطلاعات پارامتر `max_depth` فراهم شده و سپس اطلاعات مربوط به هر نمونه به صورت جداگانه نمایش داده شده است.

```
# 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)
```

نتایج:

```

→ Analysis for Decision Tree with max_depth=12:

Sample 1:
Features: [1.026e+01 1.658e+01 6.585e+01 3.208e+02 8.877e-02 8.066e-02 4.358e-02
2.438e-02 1.669e-01 6.714e-02 1.144e-01 1.023e+00 9.887e-01 7.326e+00
1.027e-02 3.084e-02 2.613e-02 1.097e-02 2.277e-02 5.890e-03 1.083e+01
2.204e+01 7.108e+01 3.574e+02 1.461e-01 2.246e-01 1.783e-01 8.333e-02
2.691e-01 9.479e-02]
True Label: 1
Predicted Label: 1

Sample 2:
Features: [1.230e+01 1.902e+01 7.788e+01 4.644e+02 8.313e-02 4.202e-02 7.756e-03
8.535e-03 1.539e-01 5.945e-02 1.840e-01 1.532e+00 1.199e+00 1.324e+01
7.881e-03 8.432e-03 7.004e-03 6.522e-03 1.939e-02 2.222e-03 1.335e+01
2.846e+01 8.453e+01 5.443e+02 1.222e-01 9.052e-02 3.619e-02 3.983e-02
2.554e-01 7.207e-02]
True Label: 1
Predicted Label: 1

```

محاسبه دقت برای درخت تصمیم مذکور :

```

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

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

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

```

نتایج:

```

Analysis for Decision Tree with max_depth=12:
Accuracy: 0.6140350877192983

```

همانطور که مشخص است دقت به دست آمده کم است، با تغییر حد اکثر عمق و استفاده از هرس کردن سعی می کنیم دقت را بهبود دهیم:

حد اکثر عمق ۵ و پارامتر هرس کردن ۱

```

# Set hyper parameters
max_depth = 5
ccp_alpha = 0.01

# Create a decision tree classifier

```

```
clf = DecisionTreeClassifier(max_depth=max_depth, ccp_alpha=ccp_alpha)

# 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)
```

نتایج:

```
Analysis for Decision Tree with max_depth=5:
Accuracy: 0.9298245614035088
```

همانطور که مشخص است با استفاده درست از این دو پارامتر توانستیم دقیق را تا حد خوبی زیاد کنیم.

بخش ۳ (اختیاری)

مجموعه داده 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 مجموعه آزمون به خوبی انجام داد.

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 محركی 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 ۲۰۱۵ تا ۲۰۰۰ برای تمام کشورها. واکسیناسیون‌های مهم مانند هپاتیت B، پلیو و دیفتريا 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 پژوهه بر اطمینان از دقیقت داده‌ها بنا شده است. مخزن داده‌های سازمان جهانی بهداشت (WHO) 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 پژوهه، برای تحلیل بیشتر، از داده‌ها از سال ۲۰۱۵ تا ۲۰۰۰ 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 شناخته شده 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 پیش‌بینی کننده بود. تمام متغیرهای پیش‌بینی کننده 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 داده‌هایی به شکل string داریم و هم در بخشی از قسمت‌های دیتاست می‌بینیم که برخی از داده‌ها NaN می‌باشند. برای همین موضوع به صورت زیر عمل می‌کنیم :

```
# 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 کتابخانه‌های مختلف را در کولب فراخوانی می‌کنیم. سپس با استفاده از دستورات مربوط به درخت تصمیم به سراغ رگرسیون داده‌ها و پیش‌بینی داده‌های آزمون می‌رویم و سپس آن را با داده‌های واقعی مقایسه می‌کنیم و میزان خطا را مشخص می‌کنیم :

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1USBbkCOMXy8dUENFtCrE-c2bq7UhfXES
# https://drive.google.com/file/d/1USBbkCOMXy8dUENFtCrE-c2bq7UhfXES/view?usp=sharing
data = pd.read_csv('/content/Life Expectancy Data.csv')
data.head()
```

نتایج:

	Country	Year	Status	Life expectancy	Adult Mortality	Infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 19 years	thinness 5-9 years	Income composition of resources	Schooling
0	Afghanistan	2015	Developing	65.0	283.0	62	0.01	71.210824	65.0	1154	...	8.0	8.16	65.0	0.1	584.250210	33730494.0	17.2	17.3	0.470	10.1
1	Afghanistan	2014	Developing	59.8	271.0	64	0.01	73.923582	62.0	492	...	58.0	8.18	62.0	0.1	612.000514	327582.0	17.5	17.5	0.470	10.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744078	31731688.0	17.7	17.7	0.470	9.9
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	87.0	2787	...	87.0	8.52	87.0	0.1	669.050000	3696658.0	17.9	18.0	0.483	9.8
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.537231	2978599.0	18.2	18.2	0.454	9.5

```
# 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 می کند تا نام ستون های جدول داده های این را تمیز کنید. با استفاده از `str.strip()` بر روی نام ستون ها، هر فضای خالی در ابتدا یا انتهای نام ستون حذف می شود. این کار بهبود خوانایی و دسترسی به داده ها را فراهم می کند، زیرا احتمال دارد که در نام ستون ها فاصله های اضافی وجود داشته باشد که ممکن است باعث اشتباه در استفاده از آنها شود.

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

نتایج:

```
(2938, 22)
```

سپس به کمک دستور `info` اطلاعات سریعی در مورد داده های این ارائه می دهد. این شامل تعداد ردیفها، تعداد و نوع داده های هر ستون، میزان حافظه مصرفی و اطلاعات مربوط به داده های `null` می شود. این اطلاعات می تواند به شما کمک کند تا داده های این را بهتر فهمیده و نقاط ضعف یا نقاط قوت ممکن در داده ها را شناسایی کنید.

نتایج:

```
<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 2984 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
```

به کمک دستور `data.isnull().sum()` برسی میکنیم که هر ستون از داده‌ها دارای چه تعداد مقدار null است. این اطلاعات به ما اجازه می‌دهند تا برسی کنیم 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.

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

نتایج:

```
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
```

به کمک دستور `duplicated().any()` بررسی می‌کند که آیا در داده‌ها ردیف تکراری وجود دارد یا خیر. اگر خروجی True باشد، 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.

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

نتایج:

False

به کمک دستور `describe()` اطلاعات آماری خلاصه‌ای در مورد ستون‌های عددی داده‌هایتان ارائه می‌شود. 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.

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

نتایج:

	Year	Life expectancy	Adult Mortality	Infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	GME	under-five deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 5-19 years	thinness 5-9 years	Income composition of resources	Schooling
count	2938.000000	2938.000000	2928.000000	2988.000000	2744.000000	2938.000000	2988.000000	2904.000000	2938.000000	2919.000000	2938.000000	2713.000000	2919.000000	2938.000000	2490.000000	2,890,000,000+03	2,890,000,000	2,890,000,000	2,777,000,000	
mean	2007.516720	69.224892	164.79648	30.39348	4.602861	738.251295	80.940461	2419.952240	38.321547	42.038739	82.505188	5.93819	82.324684	1.742103	7483.18469	1,275,538,047+07	4.839704	4.870317	6,627,551 11,992,789	
std	4.615841	9.523867	124.202079	117.858601	4.020413	1987.91458	25.070216	11487.272489	20.044034	160.445648	23.428546	2.49832	23.718912	5.077785	14270.188342	6,912,125,047+07	4.420195	4.308882	6,210,904 3,358,892	
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.37000	2.000000	0.100000	1,681,350	3,402,000,000+01	0.100000	0.100000	0,000,000 0,000,000	
25%	2004.000000	63.100000	74.000000	0.000000	0.877600	4.685343	77.000000	0.000000	19.300000	0.000000	4.26000	78.000000	0.100000	463.93626	1,957,932,047+06	1,800,000	1,700,000	6,483,000 10,150,000		
50%	2008.000000	72.700000	144.000000	3.000000	3.756000	64.919206	92.000000	17.000000	43.500000	4.000000	9.000000	9.75000	93.000000	0.100000	1766.047695	3,186,042,047+06	3,300,000	3,300,000	6,877,000 12,350,000	
75%	2011.000000	78.700000	228.000000	22.000000	7.702500	441.034144	97.000000	360.260000	56.200000	28.00000	7.46200	97.000000	0.800000	5910.80335	7,423,036,047+06	7,300,000	7,300,000	6,775,000 14,350,000		
max	2015.000000	88.000000	723.000000	1800.000000	17.470000	19470.911610	99.000000	21183.000000	87.300000	390.000000	99.000000	17.80000	99.000000	50.000000	119172.741900	1,293,050,047+08	27,700,000	28,800,000	6,946,000 20,750,000	

این دستور `data.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 می‌کند تا توزیع و ویژگی‌های مهم داده‌های دسته‌ای را درک کنید.

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

نتایج:

	Country	status
count	2938	2938
unique	193	2
top	Afghanistan	Developing
freq	16	2426
2015	0.000000	0.000000

```
# splitting data to train and test
train, test = train_test_split(data, test_size = 0.2, random_state = 93)
```

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 کننده حجم داده‌های آزمون است که در اینجا برابر با 0.20% مجموع داده‌ها است. پارامتر `random_state` تعیین کننده 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_test_with_median` و `fill_train_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.

این دستور `train.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.

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

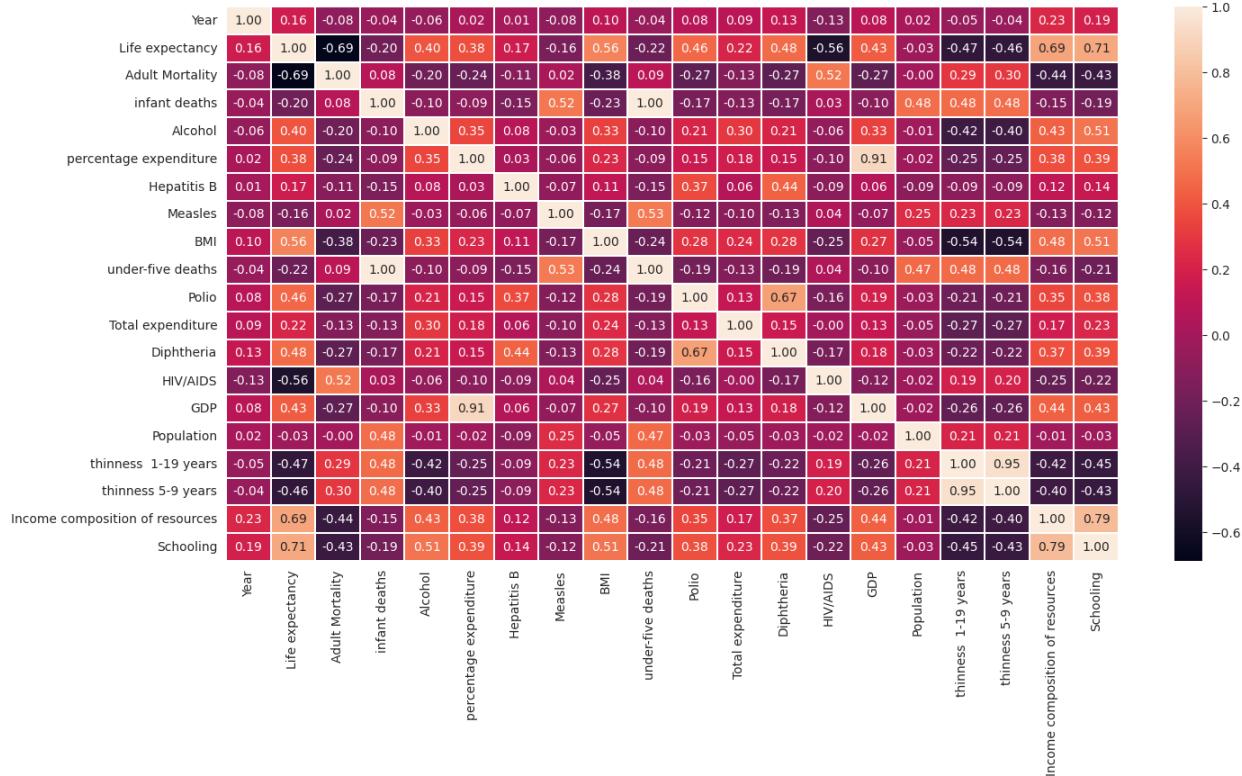
نتایج:

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()
```

ایجاد heatmap از ماتریس همبستگی بین ستون‌های عددی داده‌های آموزش train 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 دهنده ارتباطات آماری بین ستون‌های عددی است.

نتایج:



```
# 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_{test} حاوی 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 `MinMaxScaler` مقیاس‌پذیر شده‌اند. این مقیاس‌دهی به ازای هر ستون، مقادیر را به یک بازه استاندارد (معمولًاً $[0, 1]$) تبدیل می‌کند. این کار معمولاً برای بهبود عملکرد الگوریتم‌های یادگیری ماشین که به مقیاس داده حساس هستند، مفید است.

y train نتایج:

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

یک مدل رگرسیون درخت تصمیم با استفاده از کلاس `DecisionTreeRegressor` ایجاد شده است. این مدل به منظور پیش‌بینی یک مقدار عددی، به عنوان مثال، پیش‌بینی امیدراه زندگی، استفاده می‌شود.

```
tree_model = tree.DecisionTreeRegressor(random_state=93)
# 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 سپس، با استفاده از مدل آموزش دیده بر روی داده‌های آزمون، پیش‌بینی‌ها انجام شده است.

```

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 متريک‌های معمول برای ارزیابی عملکرد مدل استفاده شده است.

`mean_squared_error` و `mean_absolute_error` به ترتیب معیارهای خطای میانگین مطلق و خطای میانگین مربع را بر حسب `y_test` و `y_pred` محاسبه می‌کنند. سپس، مقادیر اين معیارها چاپ می‌شوند.

برای محاسبه ضریب توافق `r2score` بر حسب `y_test` و `y_pred` استفاده شده و نتیجه آن نیز چاپ می‌شود. `R-squared` نشان دهندهٔ تطابق مدل با داده‌هاست که در بازه [۰, ۱] قرار دارد، که مقادیر بالاتر نشان‌دهندهٔ 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 و برچسب‌های واقعی محاسبه شده‌اند.

خروجی داده‌های 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:

❖ `max_depth` (حداکثر عمق درخت)

افزایش عمق ممکن است باعث برآذش بهتر داده‌های آموزش شود، اما اگر به افزایش بیش از حد برود، ممکن است باعث برآذش بیش از حد و برداشت از داده‌ها شود (*overfitting*)

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

❖ `min_samples_split` حداقل تعداد نمونه‌ها برای تقسیم یک گره

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

❖ `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 برآذش کند یا خیر.

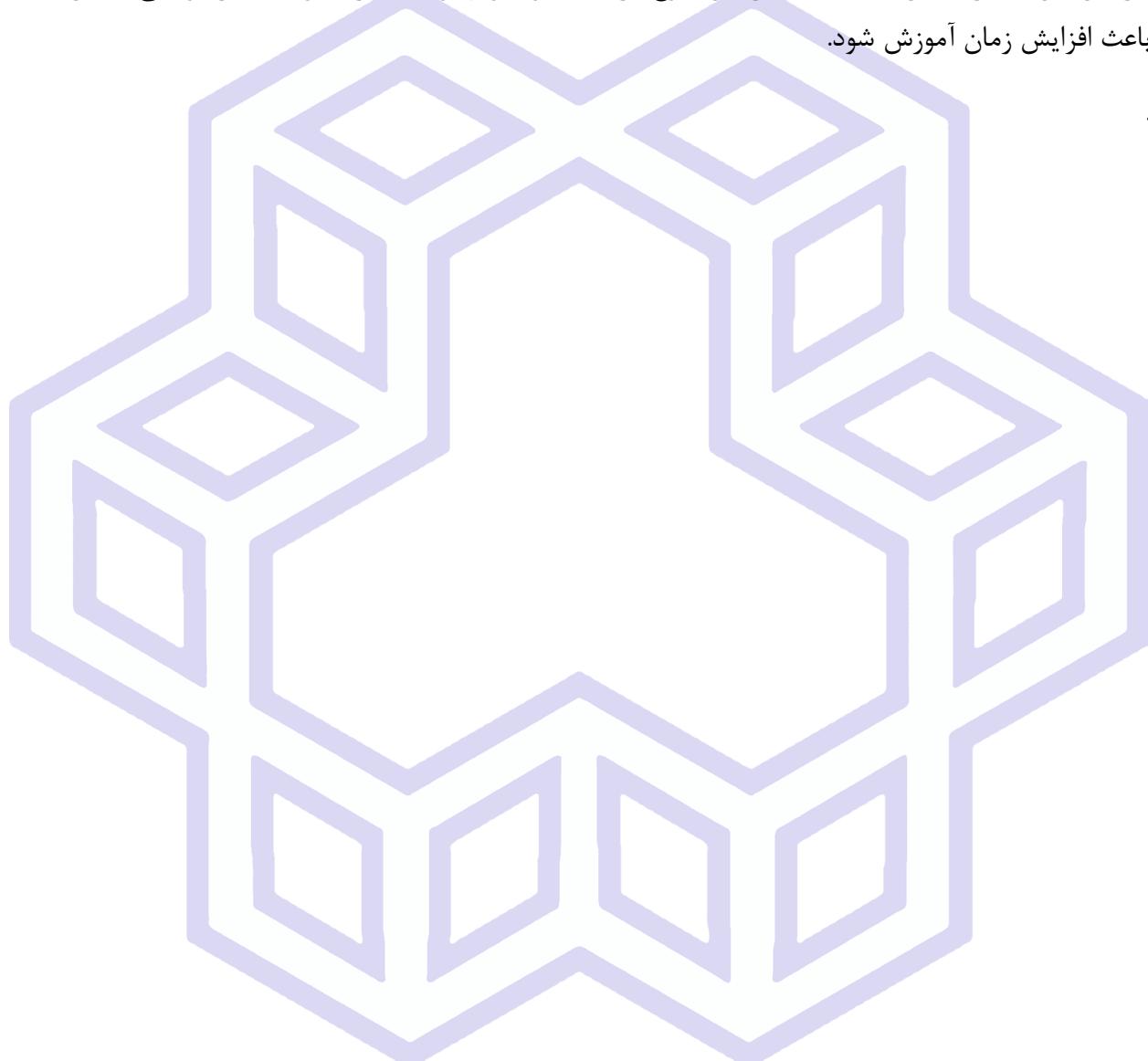
افزایش این مقدار ممکن است باعث جلوگیری از برازش بیش از حد شود.

❖ max_features حداکثر تعداد ویژگی‌ها برای جستجو در هر تقسیم

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

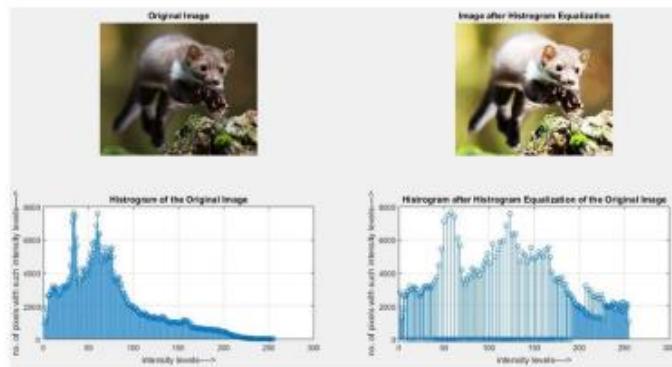
افزایش این مقدار ممکن است باعث افزایش تنوع در جستجوها و بهبود عملکرد شود، اما در موقعی ممکن است

باعث افزایش زمان آموزش شود.



سوال ۵

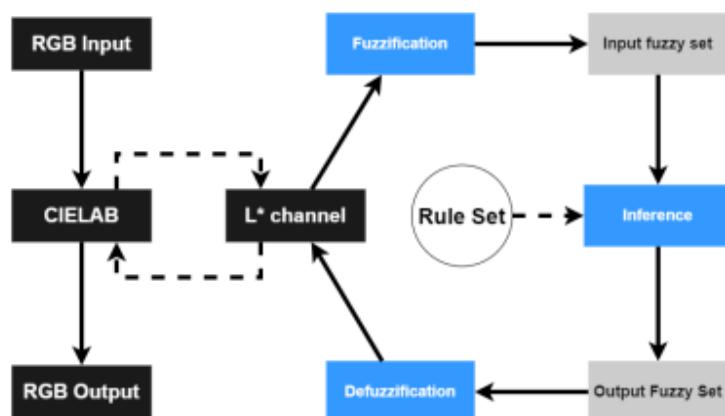
بهبود تصویر^۳ یک روش معمول در پردازش تصویر است و بهبود کنتراست^۴ یک جنبه اصلی آن است. روش‌های سنتی بهبود تصاویر مانند Histogram Equalization ممکن است موجب افزایش یا کاهش بیش از حد کنتراست تصاویر شوند، به خصوص در تصاویر باوضوح کم. هدف این پروژه، توسعه یک سیستم استدلال فازی جدید برای بهبود کنتراست تصاویر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



شکل ۱: نمایی از فرآیند بهبود تصویر.

برای پیاده‌سازی سیستم فازی مورد نیاز این پروژه می‌توانید از مقالات آورده شده در این پوشه گوگل درایو استفاده کنید و یا از این دفترچه کد گوگل کولب (**شکل ۲**) استفاده کرده و آن را تکمیل کنید. لازم است که در انتهای کار، عملکرد سیستم فازی مقایسه کنید. برای سهولت انجام پیاده‌سازی‌ها تنها کافی است که کدهای آورده شده در این دفترچه کد گوگل کولب را تکمیل کنید.

طراحی شده‌تان را با حداقل یک الگوریتم سنتی در این زمینه در دو شاخصه زمان و کیفیت عمل کرد (مانند شاخصه PSNR^۵) مقایسه کنید. برای سهولت انجام پیاده‌سازی‌ها تنها کافی است که کدهای آورده شده در این دفترچه کد گوگل کولب را تکمیل کنید.



شکل ۲: نمایی از سیستم فازی مدنظر برای بهبود تصویر در دفترچه کد گوگل کولب.

برای این منظور به صورت زیر عمل می کنیم:

ابتدا کتابخانه های مورد نیاز را فراخوانی می کنیم:

```
import math  
  
import cv2  
import matplotlib.pyplot as plt  
import numpy as np  
from IPython.display import display, Markdown  
from glob2 import glob  
  
PATH = '../Data'
```

حال باید توابع مورد نیاز برای fuzzification و توزیع پیکسل ها تعریف کرده و توابع تعلق را مشخص کنیم:

```
# Gaussian Function:  
def G(x, mean, std):  
    return np.exp(-0.5*np.square((x-mean)/std))  
  
# Membership Functions:  
def ExtremelyDark(x, M):  
    return G(x, -50, M/6)  
  
def VeryDark(x, M):  
    return G(x, 0, M/6)  
  
def Dark(x, M):  
    return G(x, M/2, M/6)  
  
def SlightlyDark(x, M):  
    return G(x, 5*M/6, M/6)  
  
def SlightlyBright(x, M):  
    return G(x, M+(255-M)/6, (255-M)/6)  
  
def Bright(x, M):  
    return G(x, M+(255-M)/2, (255-M)/6)  
  
def VeryBright(x, M):  
    return G(x, 255, (255-M)/6)  
  
def ExtremelyBright(x, M):  
    return G(x, 305, (255-M)/6)
```

```

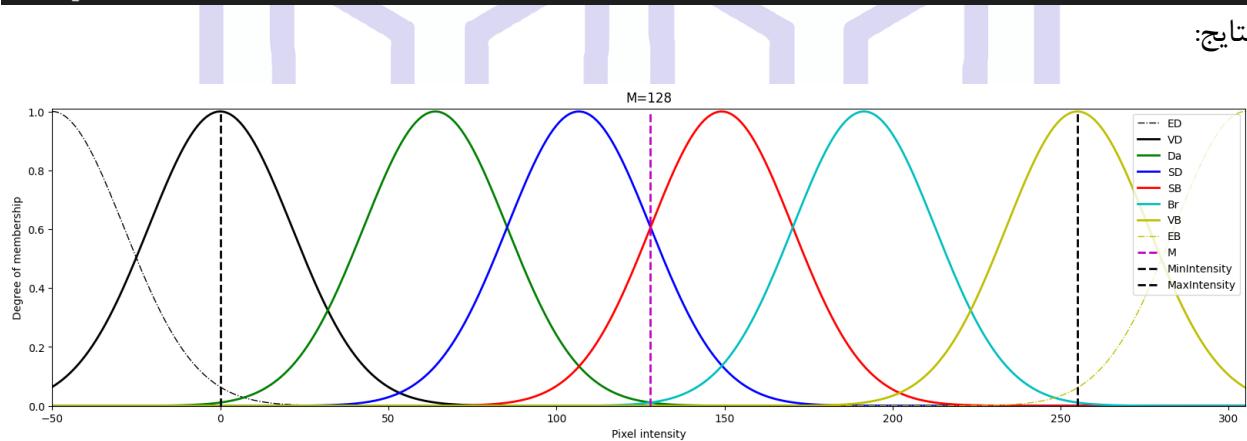
for M in (128, 64, 192):
    x = np.arange(-50, 306)

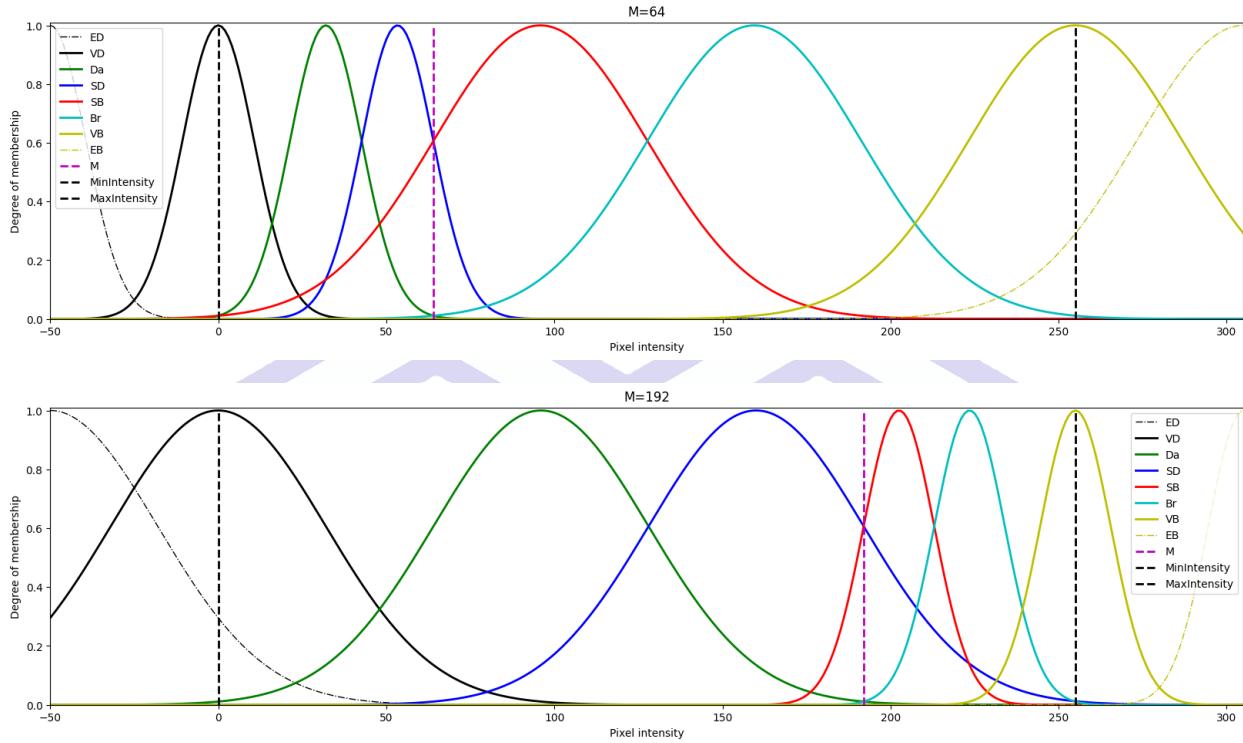
    ED = ExtremelyDark(x, M)
    VD = VeryDark(x, M)
    Da = Dark(x, M)
    SD = SlightlyDark(x, M)
    SB = SlightlyBright(x, M)
    Br = Bright(x, M)
    VB = VeryBright(x, M)
    EB = ExtremelyBright(x, M)

    plt.figure(figsize=(20,5))
    plt.plot(x, ED, 'k-.', label='ED', linewidth=1)
    plt.plot(x, VD, 'k-', label='VD', linewidth=2)
    plt.plot(x, Da, 'g-', label='Da', linewidth=2)
    plt.plot(x, SD, 'b-', label='SD', linewidth=2)
    plt.plot(x, SB, 'r-', label='SB', linewidth=2)
    plt.plot(x, Br, 'c-', label='Br', linewidth=2)
    plt.plot(x, VB, 'y-', label='VB', linewidth=2)
    plt.plot(x, EB, 'y-.', label='EB', linewidth=1)
    plt.plot((M, M), (0, 1), 'm--', label='M', linewidth=2)
    plt.plot((0, 0), (0, 1), 'k--', label='MinIntensity', linewidth=2)
    plt.plot((255, 255), (0, 1), 'k--', label='MaxIntensity', linewidth=2)
    plt.legend()
    plt.xlim(-50, 305)
    plt.ylim(0.0, 1.01)
    plt.xlabel('Pixel intensity')
    plt.ylabel('Degree of membership')
    plt.title(f'M={M}')
    plt.show()

```

نتائج:





حال باید روش defuzzification و همچنین روش مد نظرمان که mamdani است را تعریف کیم:

```

def OutputFuzzySet(x, f, M, thres):
    x = np.array(x)
    result = f(x, M)
    result[result > thres] = thres
    return result

def AggregateFuzzySets(fuzzy_sets):
    return np.max(np.stack(fuzzy_sets), axis=0)

def Infer(i, M, get_fuzzy_set=False):
    # Calculate degree of membership for each class
    VD = VeryDark(i, M)
    Da = Dark(i, M)
    SD = SlightlyDark(i, M)
    SB = SlightlyBright(i, M)
    Br = Bright(i, M)
    VB = VeryBright(i, M)

    # Fuzzy Inference:
    x = np.arange(-50, 306)
    Inferences = (
        OutputFuzzySet(x, ExtremelyDark, M, VD),
        OutputFuzzySet(x, VeryDark, M, Da),

```

```

        OutputFuzzySet(x, Dark, M, SD),
        OutputFuzzySet(x, Bright, M, SB),
        OutputFuzzySet(x, VeryBright, M, Br),
        OutputFuzzySet(x, ExtremelyBright, M, VB)
    )

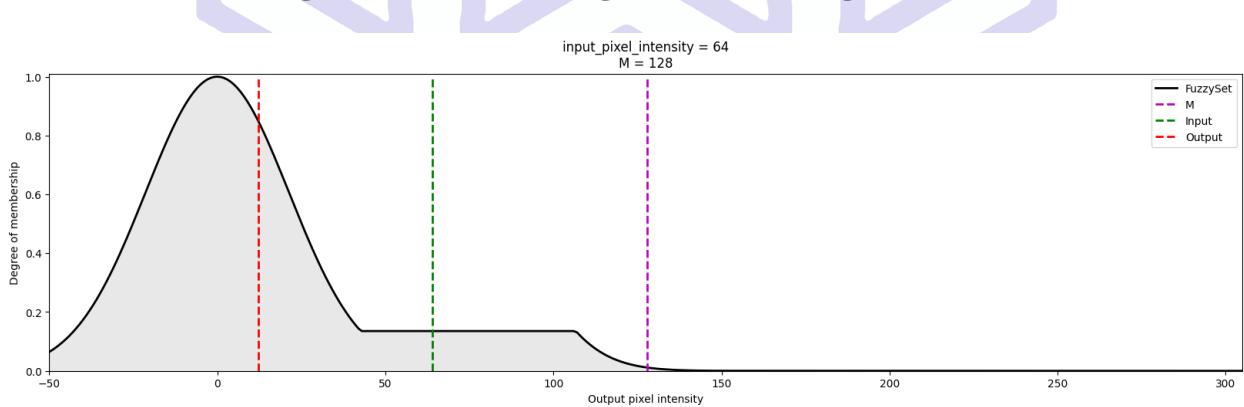
# Calculate AggregatedFuzzySet:
fuzzy_output = AggregateFuzzySets(Inferences)

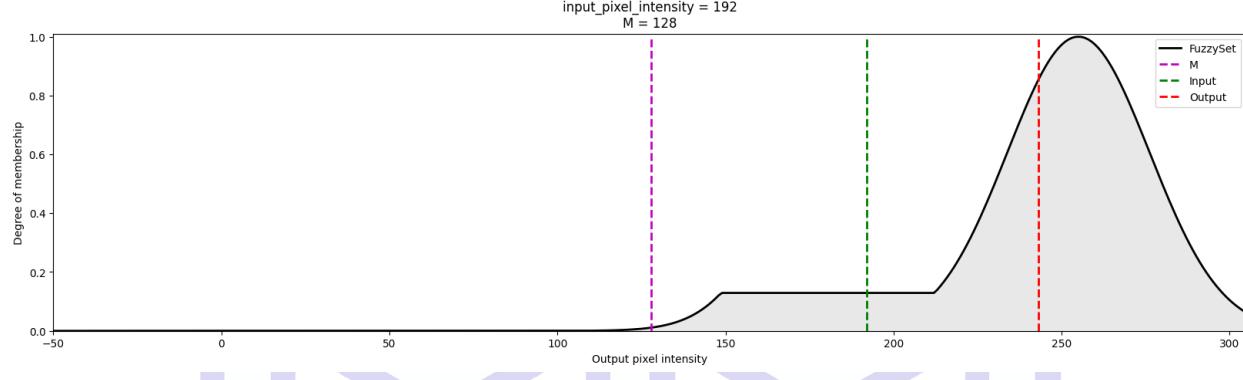
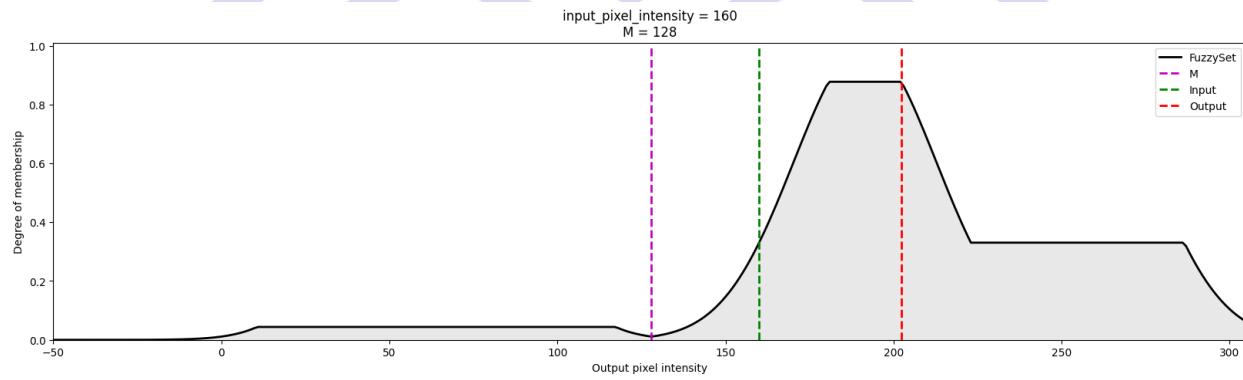
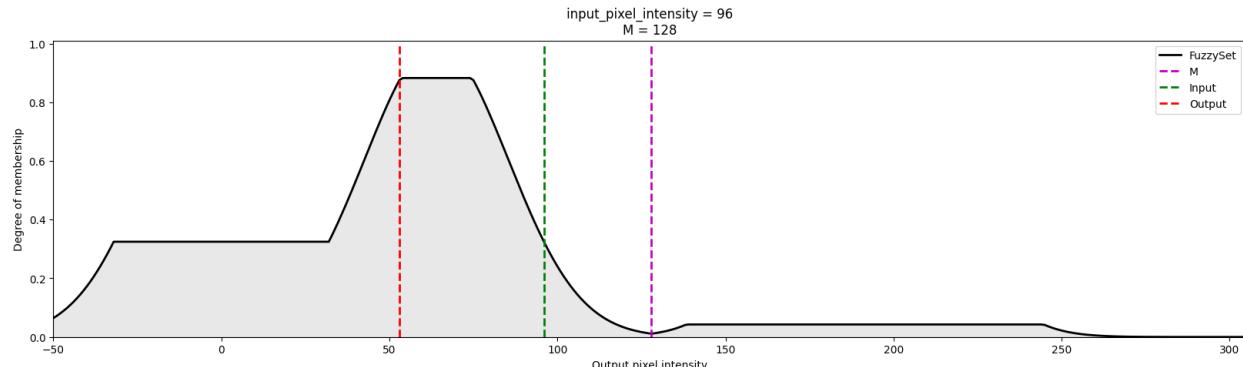
# Calculate crisp value of centroid
if get_fuzzy_set:
    return np.average(x, weights=fuzzy_output), fuzzy_output
return np.average(x, weights=fuzzy_output)

for pixel in (64, 96, 160, 192):
    M = 128
    x = np.arange(-50, 306)
    centroid, output_fuzzy_set = Infer(np.array([pixel]), M,
get_fuzzy_set=True)
    plt.figure(figsize=(20,5))
    plt.plot(x, output_fuzzy_set, 'k-', label='FuzzySet', linewidth=2)
    plt.plot((M, M), (0, 1), 'm--', label='M', linewidth=2)
    plt.plot((pixel, pixel), (0, 1), 'g--', label='Input', linewidth=2)
    plt.plot((centroid, centroid), (0, 1), 'r--', label='Output',
linewidth=2)
    plt.fill_between(x, np.zeros(356), output_fuzzy_set, color=(.9, .9,
.9))
    plt.legend()
    plt.xlim(-50, 305)
    plt.ylim(0.0, 1.01)
    plt.xlabel('Output pixel intensity')
    plt.ylabel('Degree of membership')
    plt.title(f'input_pixel_intensity = {pixel}\nM = {M}')
    plt.show()

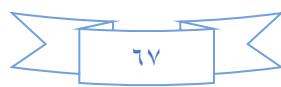
```

میزان درجه عضویت بر حسب چگالی پیکسل های خروجی به صورت زیر به دست می آید:





```
means = (64, 96, 128, 160, 192)
plt.figure(figsize=(25,5))
for i in range(len(means)):
    M = means[i]
    x = np.arange(256)
    %time y = np.array([Infer(np.array([i]), M) for i in x])
    plt.subplot(1, len(means), i+1)
    plt.plot(x, y, 'r-', label='IO mapping')
    plt.xlim(0, 256)
    plt.ylim(-50, 355)
    plt.xlabel('Input Intensity')
```

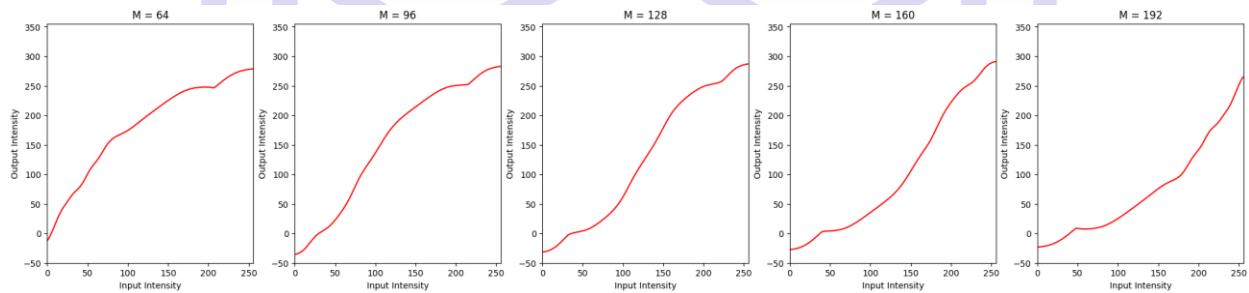


```

    plt.ylabel('Output Intensity')
    plt.title(f'M = {M}')
plt.show()
CPU times: user 52.9 ms, sys: 407 µs, total: 53.3 ms
Wall time: 53.6 ms
CPU times: user 49.8 ms, sys: 0 ns, total: 49.8 ms
Wall time: 50.4 ms
CPU times: user 65.3 ms, sys: 0 ns, total: 65.3 ms
Wall time: 65.6 ms
CPU times: user 54.4 ms, sys: 0 ns, total: 54.4 ms
Wall time: 54.3 ms
CPU times: user 51 ms, sys: 0 ns, total: 51 ms
Wall time: 51 ms

```

نتایج:



برای نمایش نتایج به صورت زیر عمل می کنیم:

```

# Proposed fuzzy method
def FuzzyContrastEnhance(rgb):
    # Convert RGB to LAB
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)

    # Get L channel
    l = lab[:, :, 0]

    # Calculate M value
    M = np.mean(l)
    if M < 128:
        M = 127 - (127 - M) / 2
    else:
        M = 128 + M / 2

    # Precompute the fuzzy transform
    x = list(range(-50, 306))
    FuzzyTransform = dict(zip(x, [Infer(np.array([i]), M) for i in x]))

    # Apply the transform to l channel

```

```

        u, inv = np.unique(l, return_inverse = True)
        l = np.array([FuzzyTransform[i] for i in u])[inv].reshape(l.shape)

        # Min-max scale the output L channel to fit (0, 255):
        Min = np.min(l)
        Max = np.max(l)
        lab[:, :, 0] = (l - Min) / (Max - Min) * 255

        # Convert LAB to RGB
        return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)

# Traditional method of histogram equalization
def HE(rgb):
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)
    lab[:, :, 0] = cv2.equalizeHist(lab[:, :, 0])
    return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)

# Contrast Limited Adaptive Histogram Equalization
def CLAHE(rgb):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)
    lab[:, :, 0] = clahe.apply(lab[:, :, 0])
    return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)

```

برای لود کردن نمونه تصاویر که می خواهیم بهبود تصویر بر روی آنها صورت بگیرد به صورت زیر عمل می کنیم:

```

data = np.array([cv2.cvtColor(cv2.imread(p), cv2.COLOR_BGR2RGB) for p in
glob(f'{PATH}/*')])
data.shape

```

نتایج:

(21)

همانطور که مشخص است ۲۱ تصویر نمونه داریم.

برای نمایش تصاویر اصلی و تصاویر بهبود یافته به صورت زیر عمل می کنیم:

```

for i in range(data.shape[0]):
    img = data[i]
    fce = FuzzyContrastEnhance(img)
    he = HE(img)
    clahe = CLAHE(img)
    display(Markdown(f'### <p style="text-align: center;">Sample Photo
{i+1}</p>'))
    plt.figure(figsize=(15, 10))
    plt.subplot(2, 2, 1)
    plt.imshow(data[i])

```

```
plt.title('Original Image')

plt.subplot(2, 2, 2)
plt.imshow(fce)
plt.title('Fuzzy Contrast Enhance')

plt.subplot(2, 2, 3)
plt.imshow(he)
plt.title('Traditional HE')

plt.subplot(2, 2, 4)
plt.imshow(clahe)
plt.title('CLAHE')

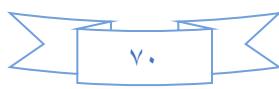
plt.show()
```

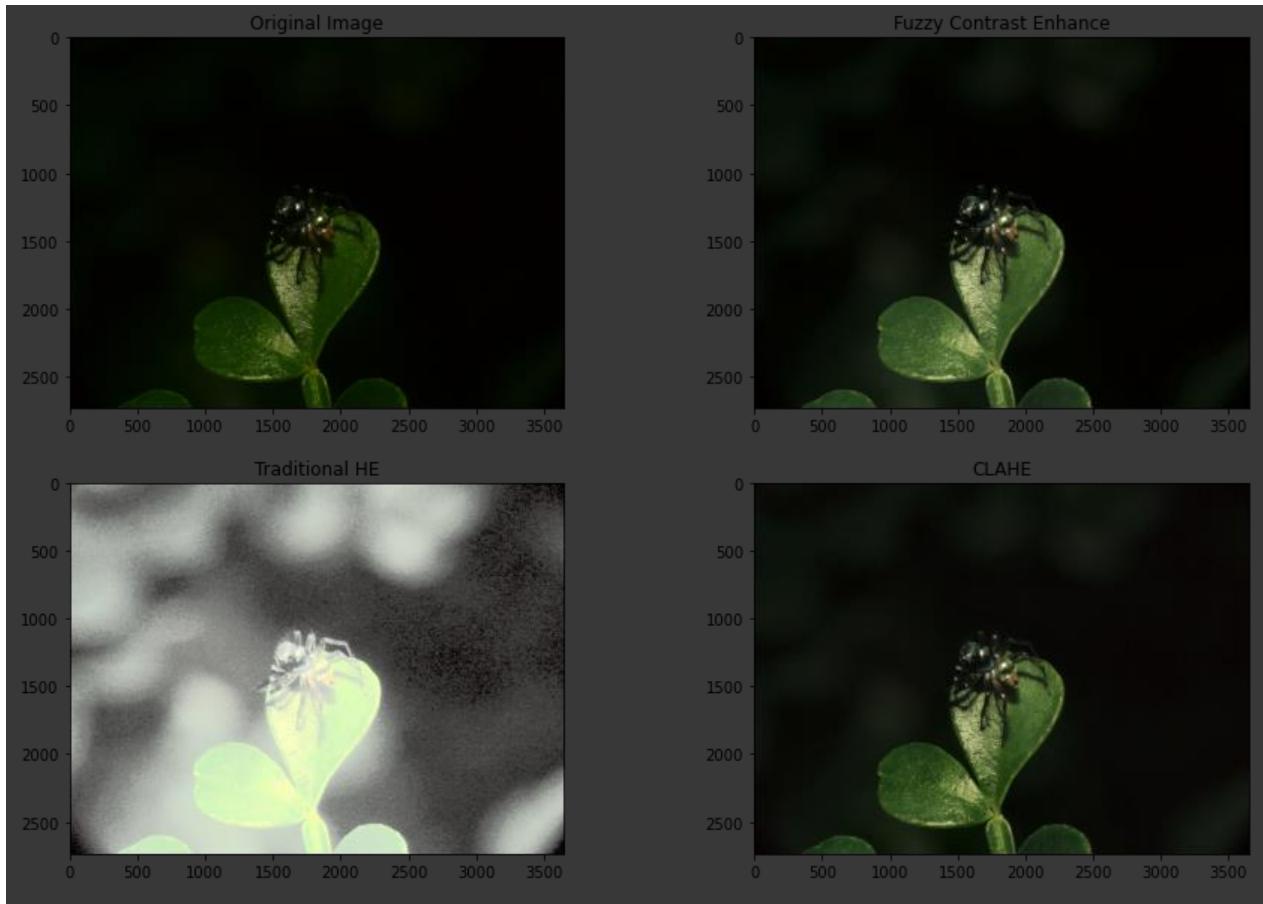
نتائج:

Sample Photo 1

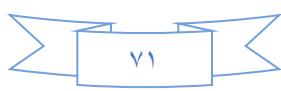
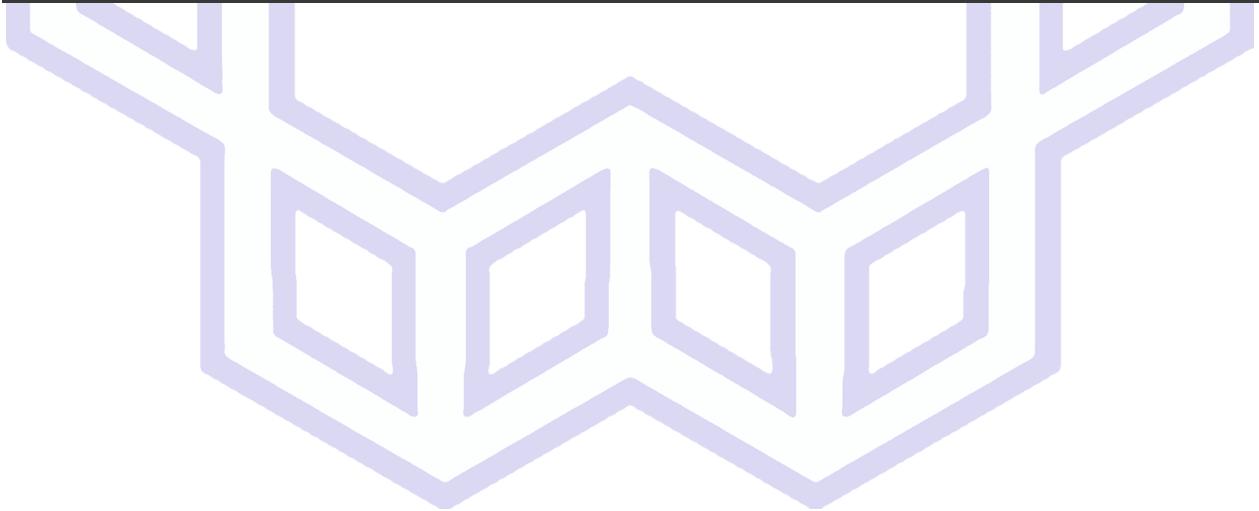


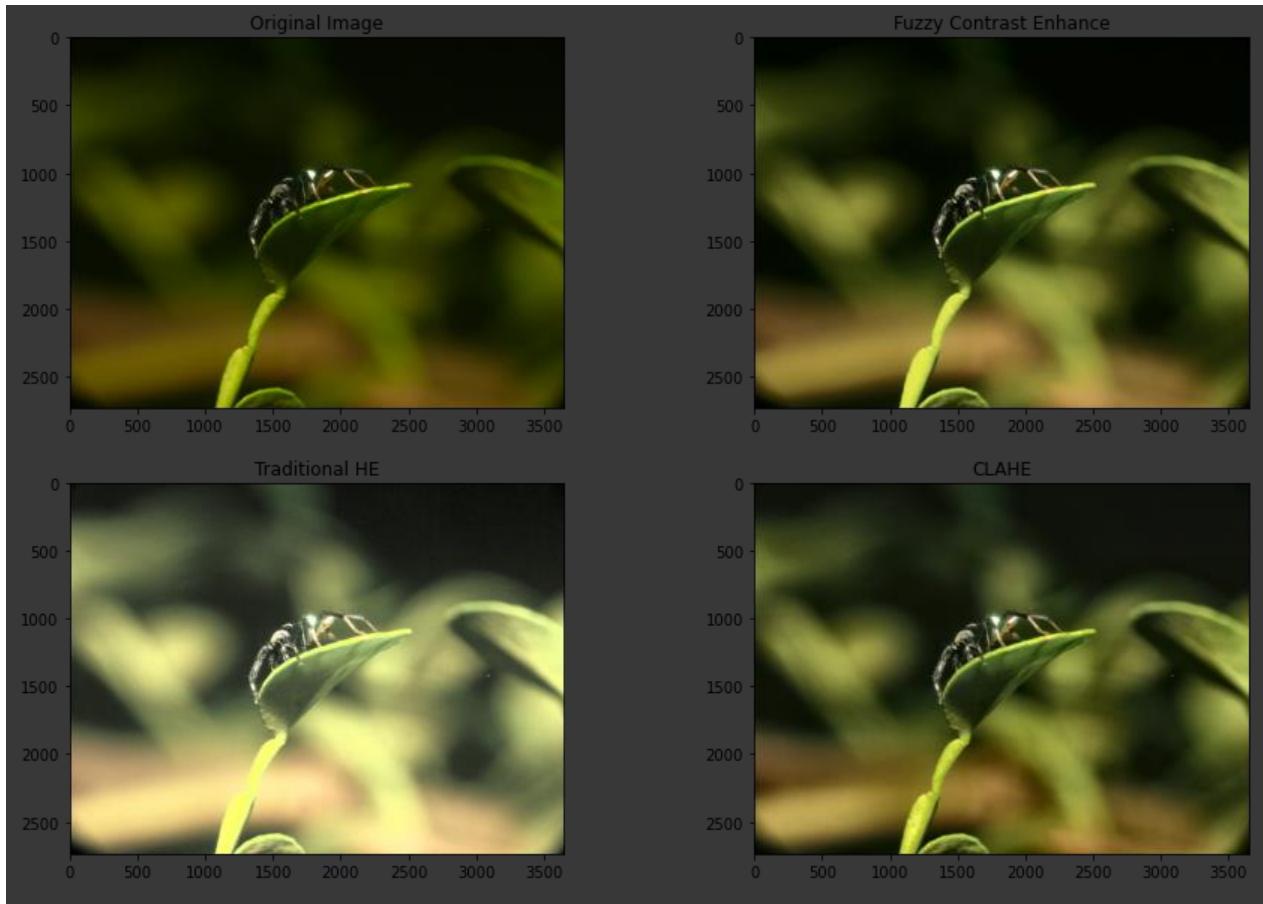
Sample Photo 2



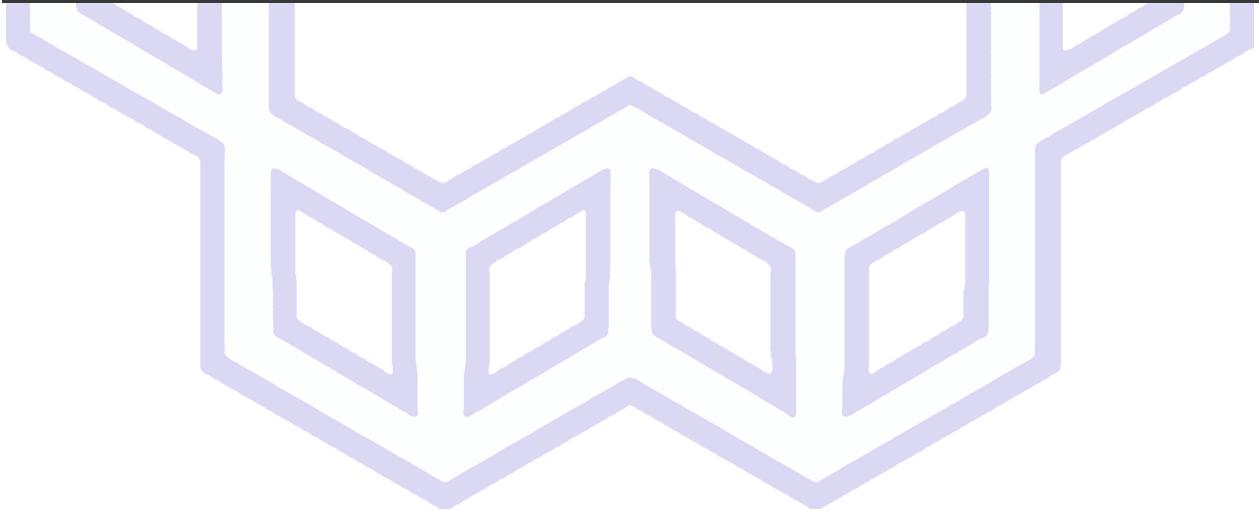


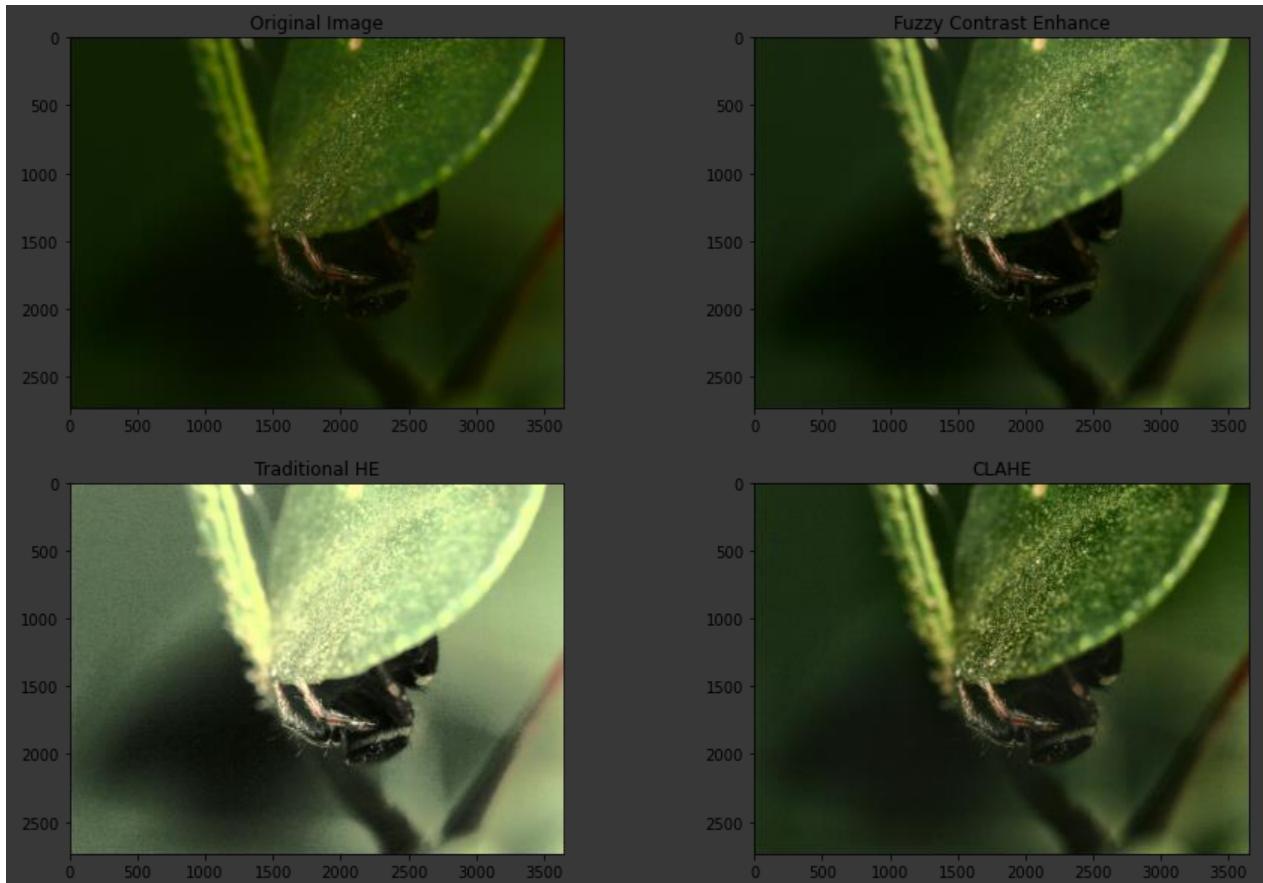
Sample Photo 3



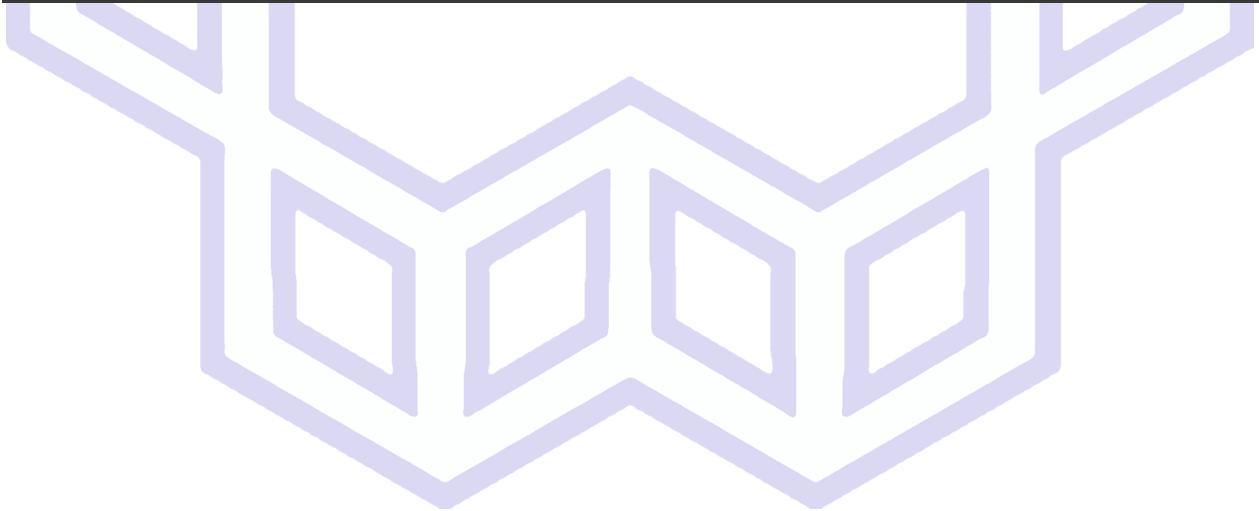


Sample Photo 4



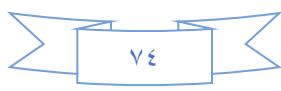
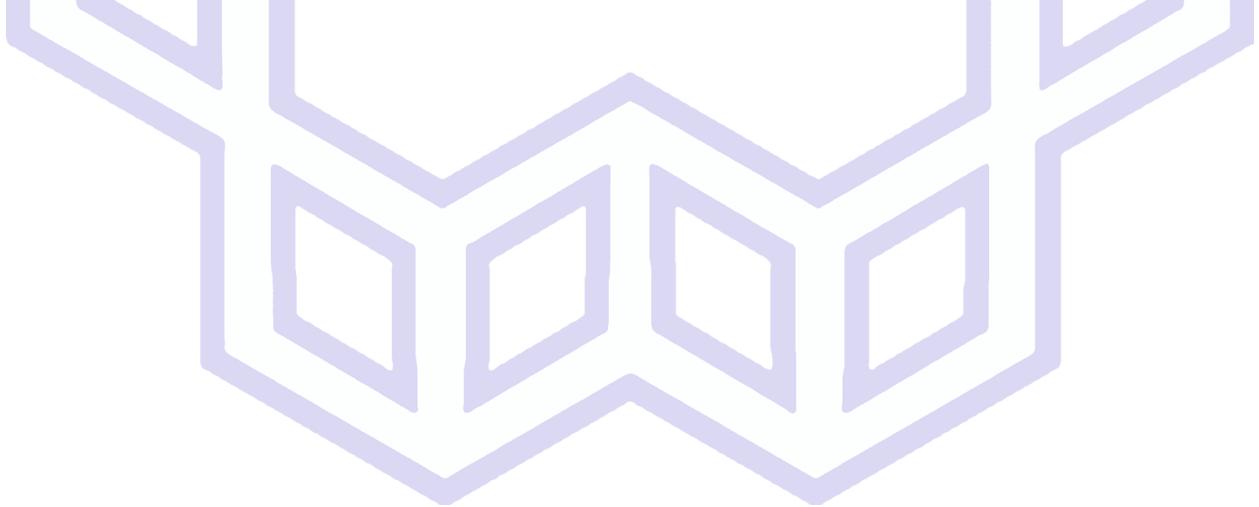


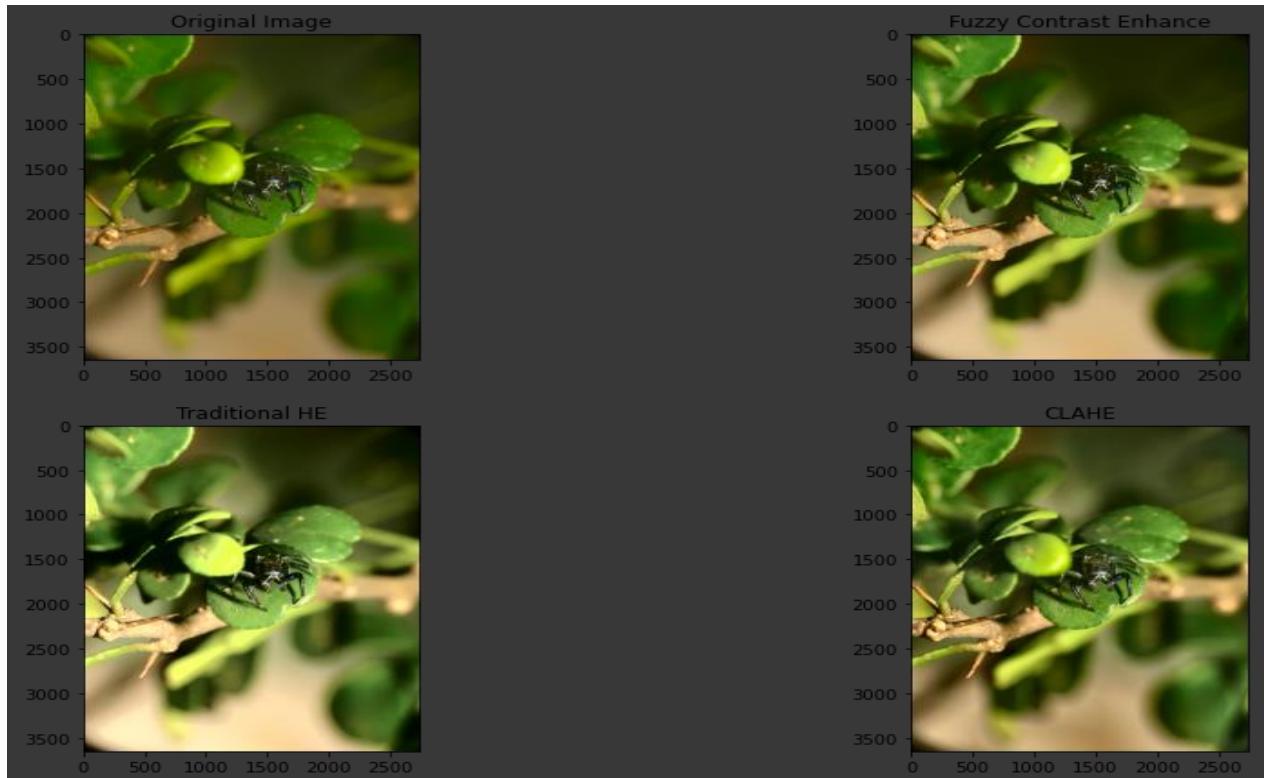
Sample Photo 5



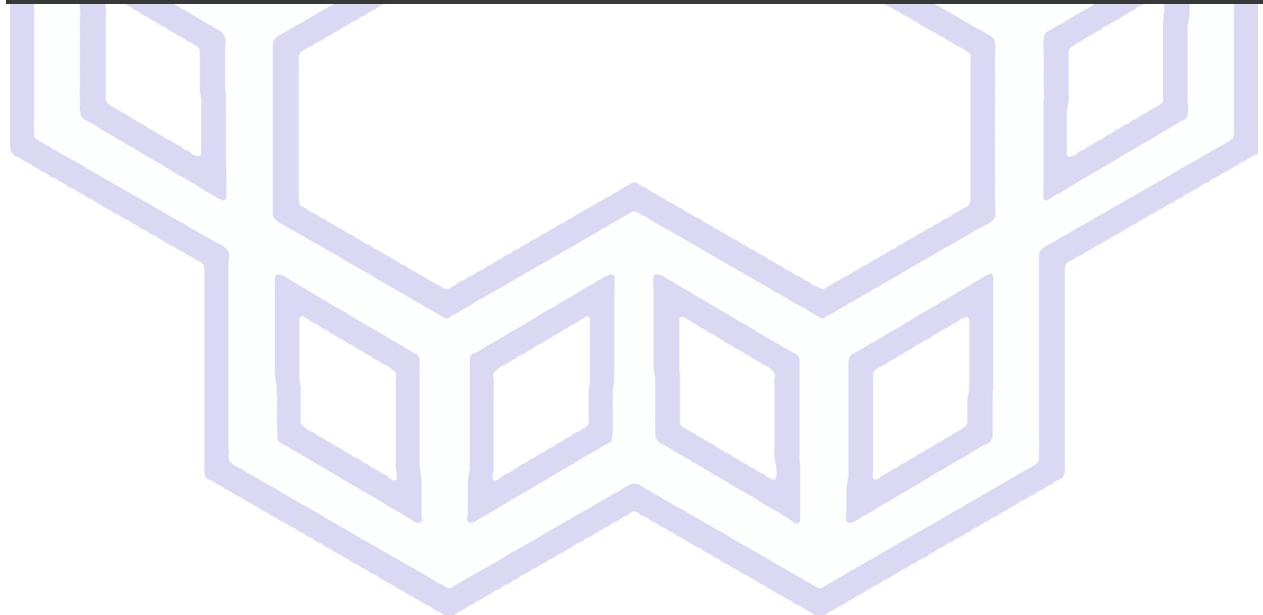


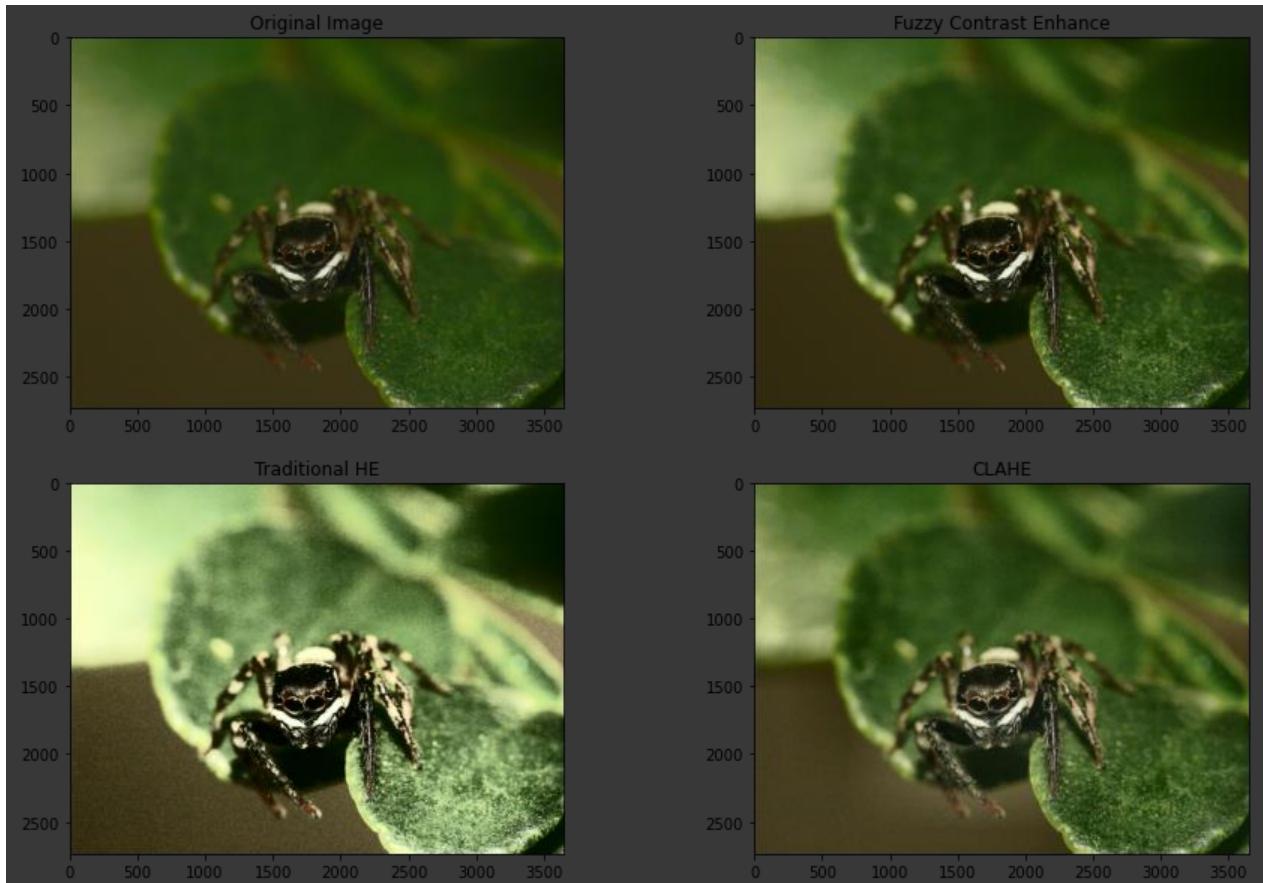
Sample Photo 6





Sample Photo 7





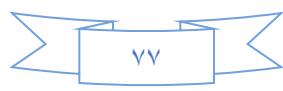
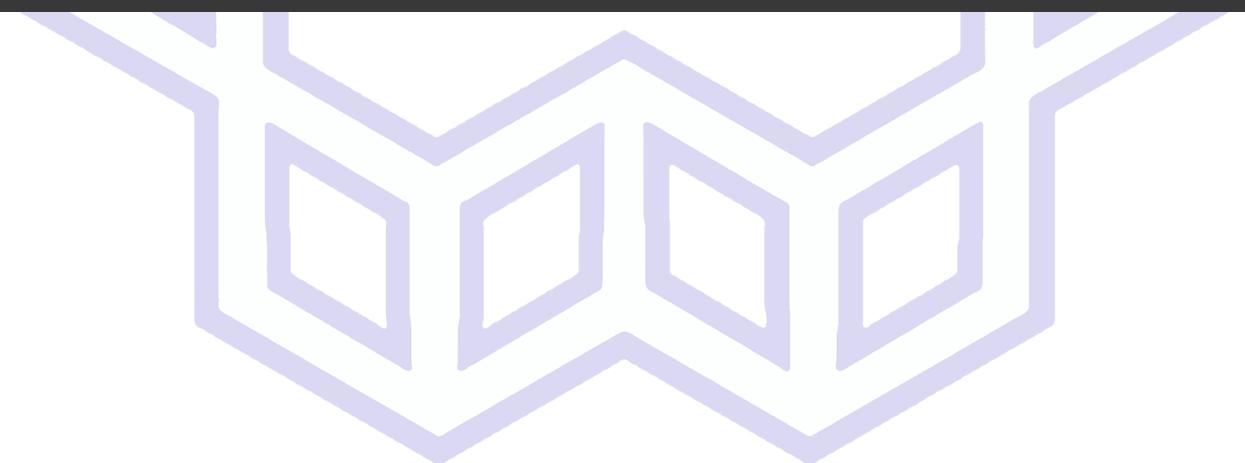
Sample Photo 8



Sample Photo 9

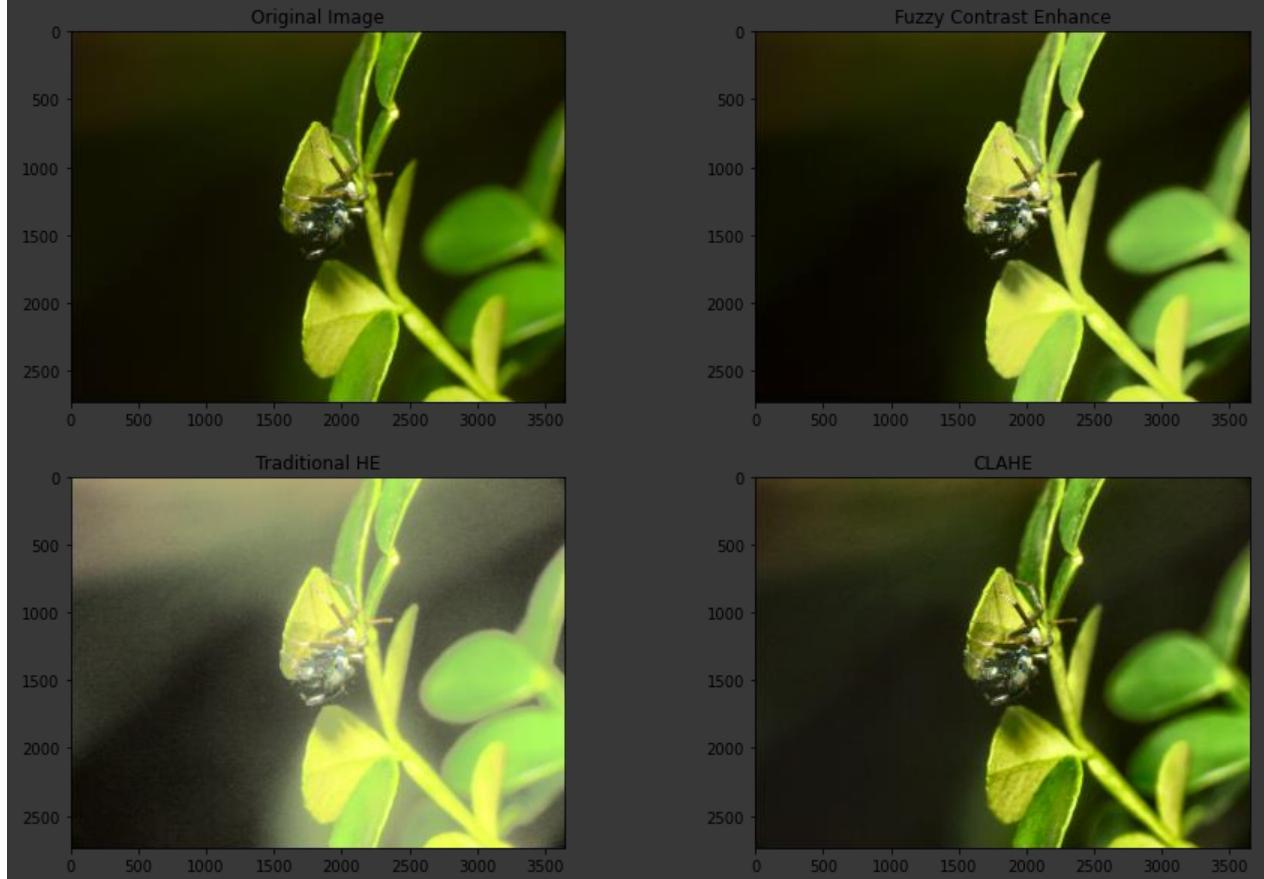


Sample Photo 10

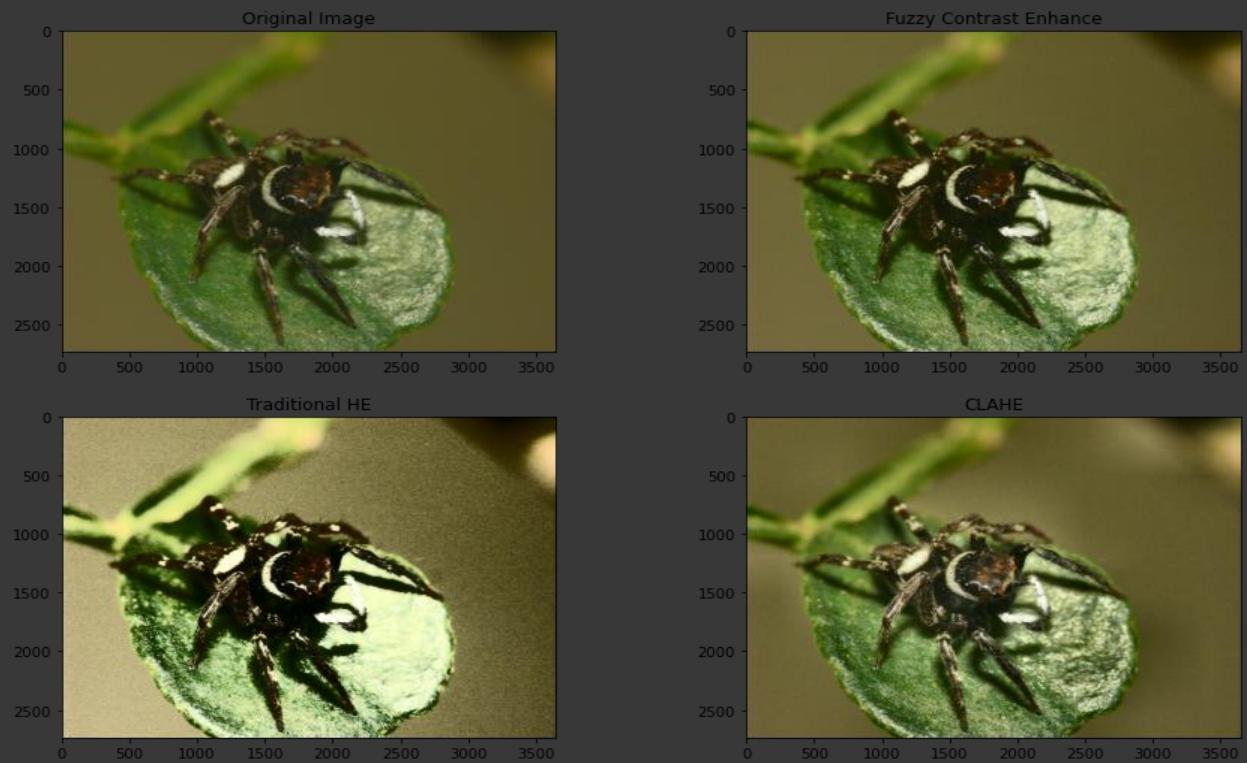




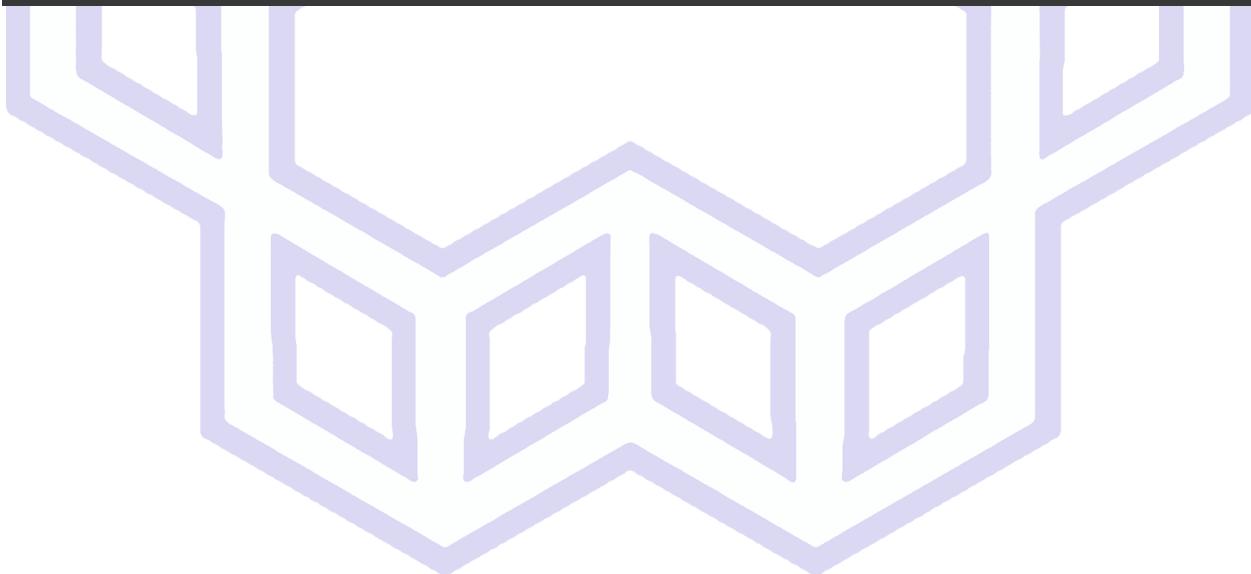
Sample Photo 11

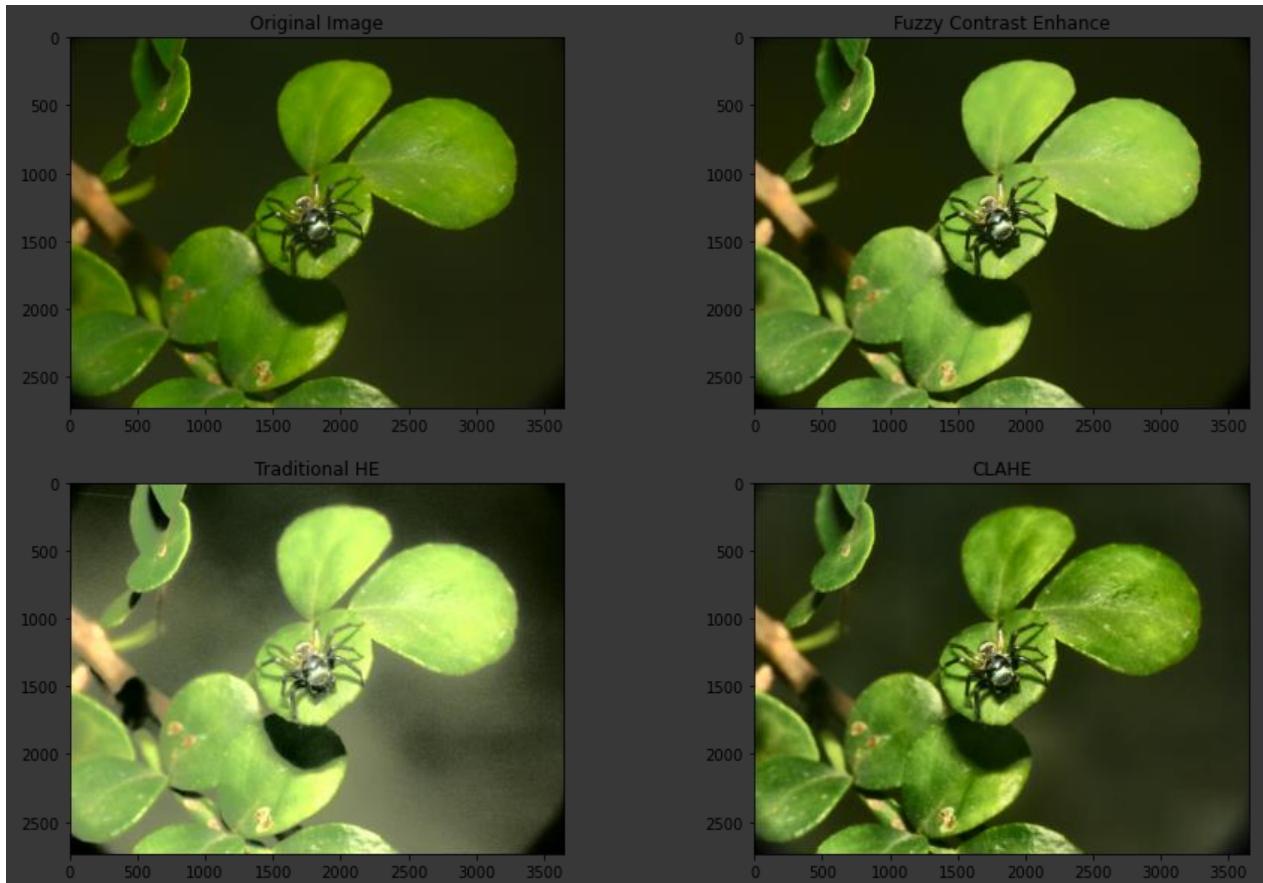


Sample Photo 12

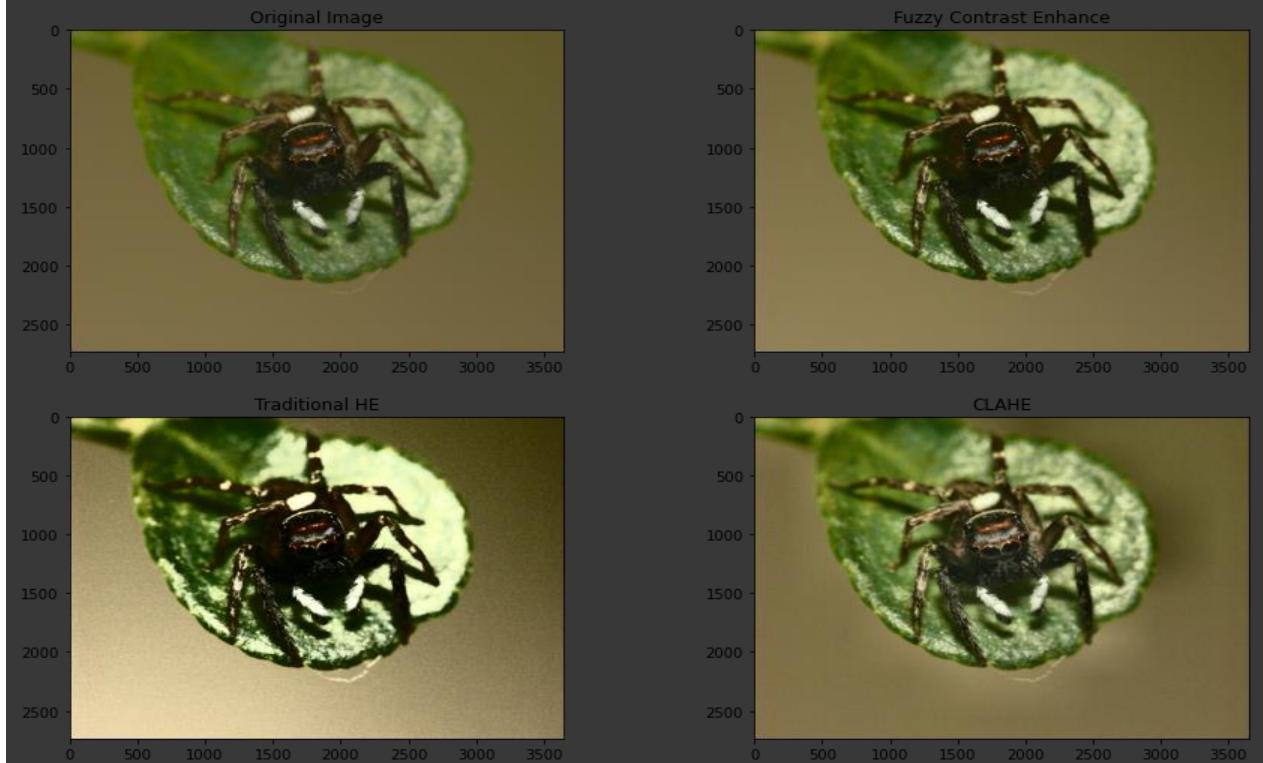


Sample Photo 13





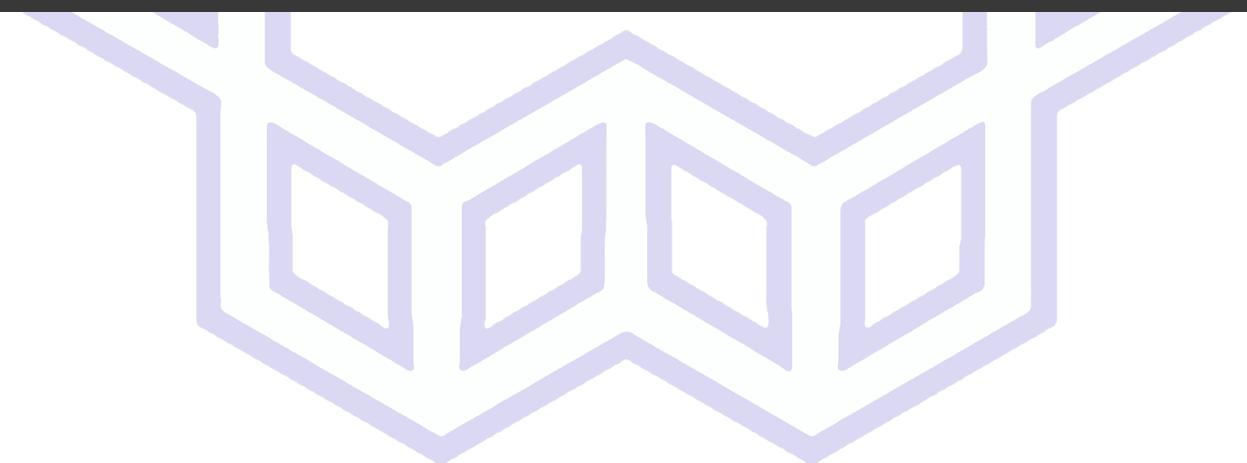
Sample Photo 14

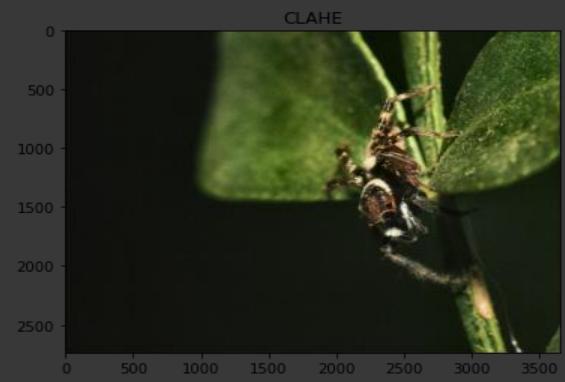
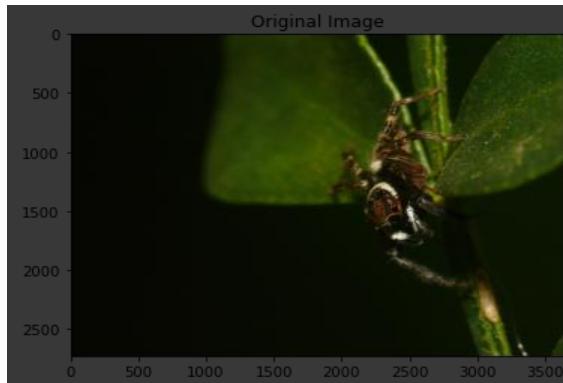


Sample Photo 15

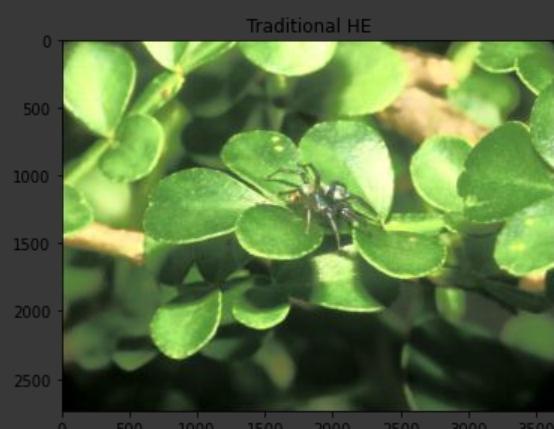


Sample Photo 16

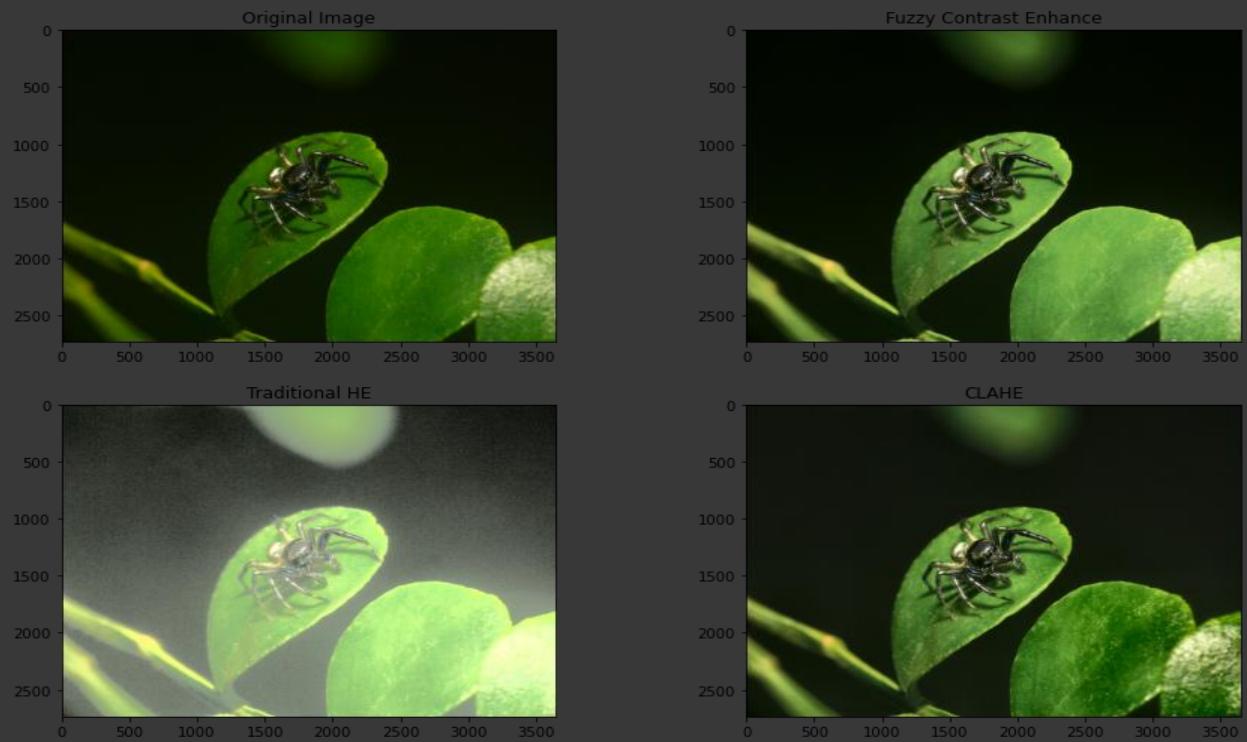




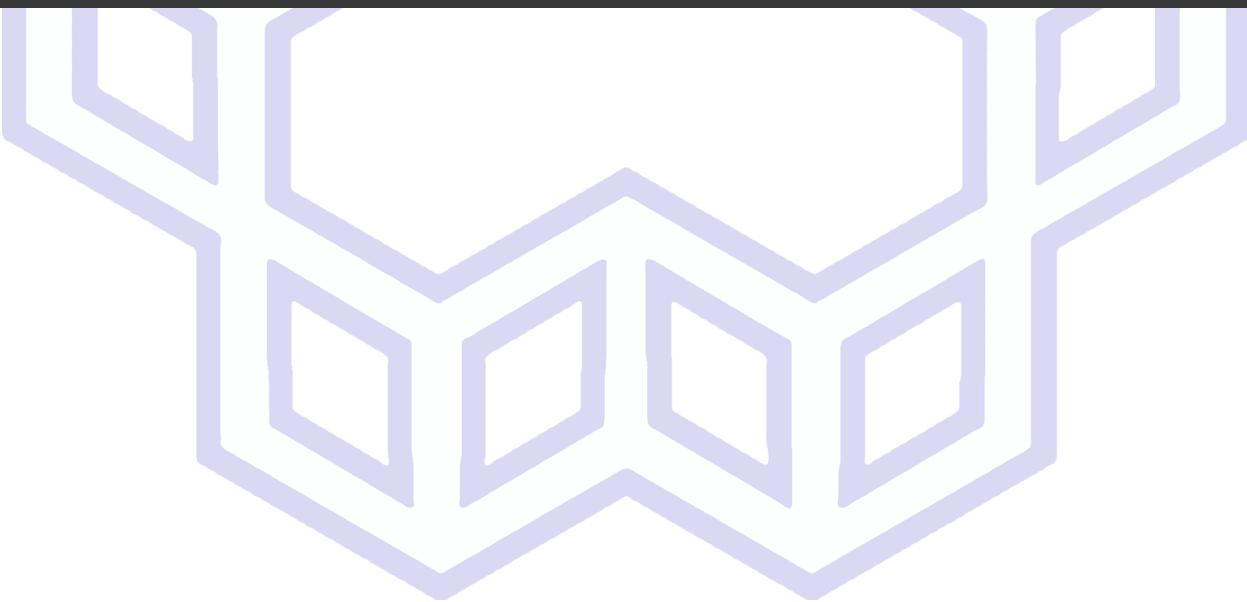
Sample Photo 17

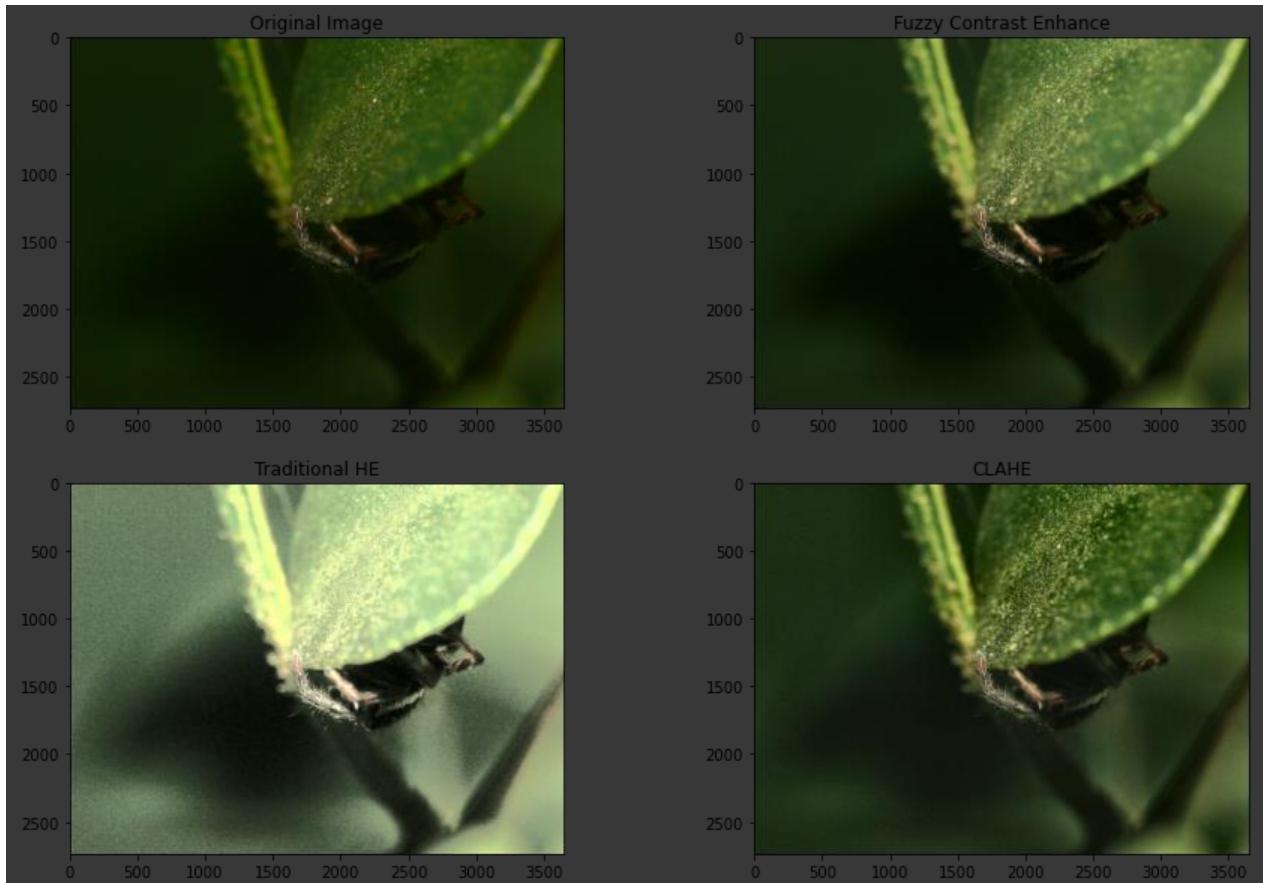


Sample Photo 18

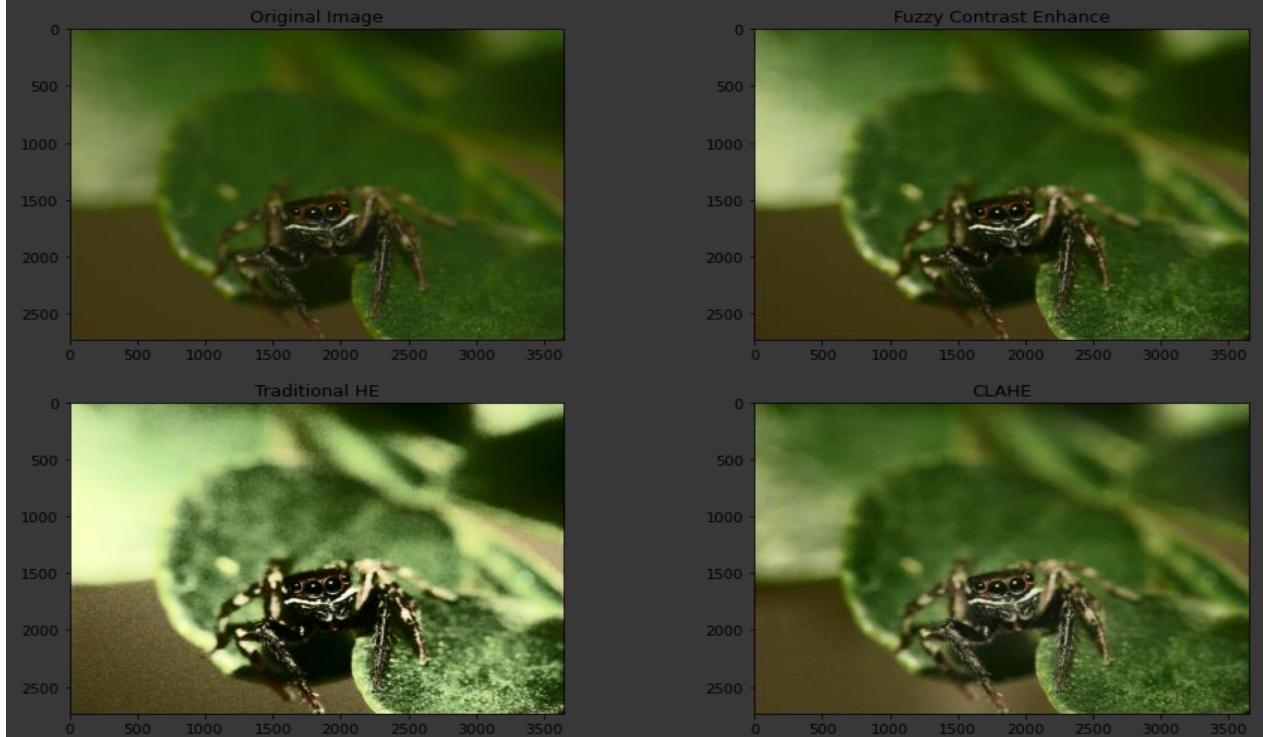


Sample Photo 19





Sample Photo 20



Sample Photo 21



همچنین شاخصه زمان عملکرد به صورت زیر است:

```
%%time
for i in range(10):
    FuzzyContrastEnhance(data[i])
```

نتایج:

```
CPU times: user 11.9 s, sys: 2.78 ms, total: 11.9 s
Wall time: 9.61 s
```

```
%%time
for i in range(10):
    HE(data[i])
```

نتایج:

```
CPU times: user 3.53 s, sys: 3.96 ms, total: 3.53 s
Wall time: 957 ms
```

```
%%time
for i in range(10):
    CLAHE(data[i])
```

نتایج:

```
CPU times: user 4.67 s, sys: 794 µs, total: 4.68 s
Wall time: 1.24 s
```

شاخصه کیفیت عملکرد PSNR :

```
def MSE(img1, img2):
    return np.mean(np.square(img1 - img2))

def PSNR(Max, MSE):
    return 10*math.log10(Max**2/MSE)

display(Markdown(f'FCE: {PSNR(255*255, np.mean([MSE(org,
FuzzyContrastEnhance(org)) for org in data])))}')))
display(Markdown(f'HE: {PSNR(255*255, np.mean([MSE(org, HE(org)) for org
in data])))}'))
display(Markdown(f'CLAHE: {PSNR(255*255, np.mean([MSE(org, CLAHE(org)) for
org in data])))}'))
```

نتایج:

```
FCE: 78.36331918083445
HE: 76.13619116041234
CLAHE: 76.67745784638485
```