 به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

**مبانی سیستم های هوشمند**

**گزارش مینی پروژه شماره سوم**

**[ ایمان فکری اسکی]**

**[9929083]**

**استاد : آقای دکتر مهدی علیاری**

بهمن ماه 1402

##### فهرست مطالب

|  |  |
| --- | --- |
| عنوان | شماره صفحه |

[بخش۱: چکیده](#_Toc128223055) 3

[مقدمه](#_Toc128223056) 4

[سوال اول](#_Toc128223057) 5

[سوال دوم](#_Toc128223064) 17

[سوال سوم](#_Toc128223064) 25

[سوال چهارم](#_Toc128223065) 31

[سوال پنجم-اختیاری](#_Toc128223064) 59

[مراجع](#_Toc128223066) 60

## چکیده :

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

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

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

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

## مقدمه :

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

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

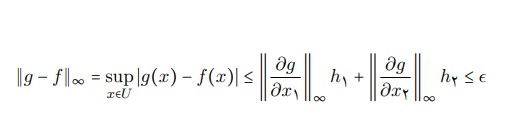
با ارائه یک تحلیل جامع از این دو رویکرد، ما نقشه‌ای دقیق از اینکه چگونه می‌توانند در مواجهه با چالش‌های مختلف، از جمله پیش‌بینی داده‌ها، کنترل سیستم‌ها، و تصمیم‌گیری در شرایط ناپایدار مورد استفاده قرار گیرند، ارائه خواهیم داد.

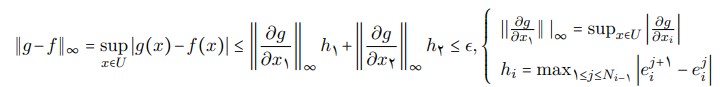
در طول این گزارش، به بررسی پیشرفت‌های اخیر و کاربردهای عملی این دو تکنیک در حل مسائل واقعی پرداخته و اهمیت آنها در توسعه‌ی راهکارهای هوشمند برای جامعه را مورد بحرانی قرار خواهیم داد.

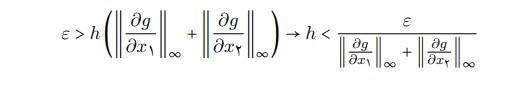
## سوال اول :

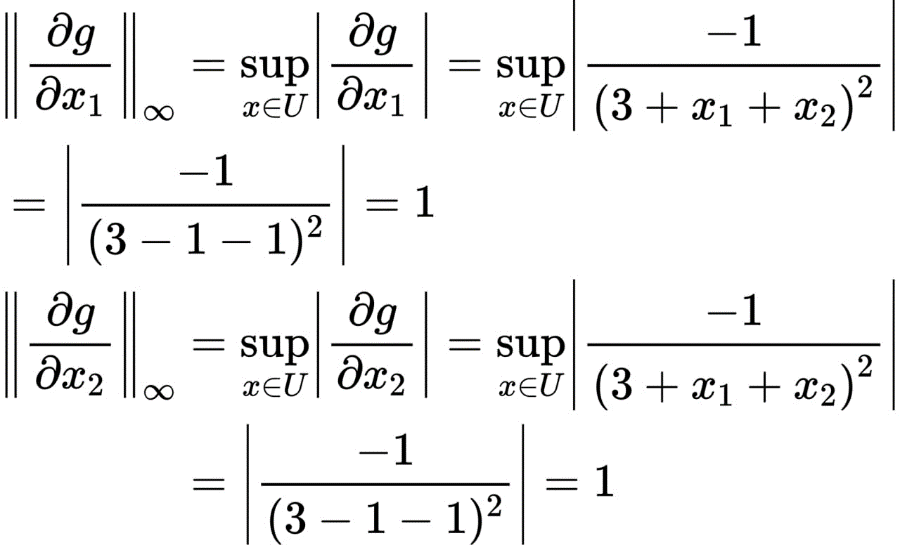
در ابتدا در سوالی که به ما داده شده است باید تعداد توابع تعلق را محاسبه کنیم. در این قسمت دو حالت متفاوت برای این موضوع خواهیم داشت که باید یک بار برای کران اول و بار دیگر برای کران دوم ، مشخصات مربوطه محاسبه شود. به صورت زیر عمل می کنیم :

برای کران مرتبه اول :





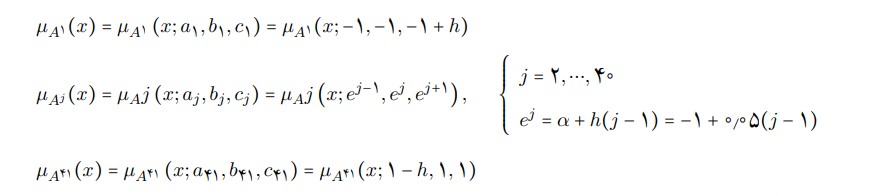


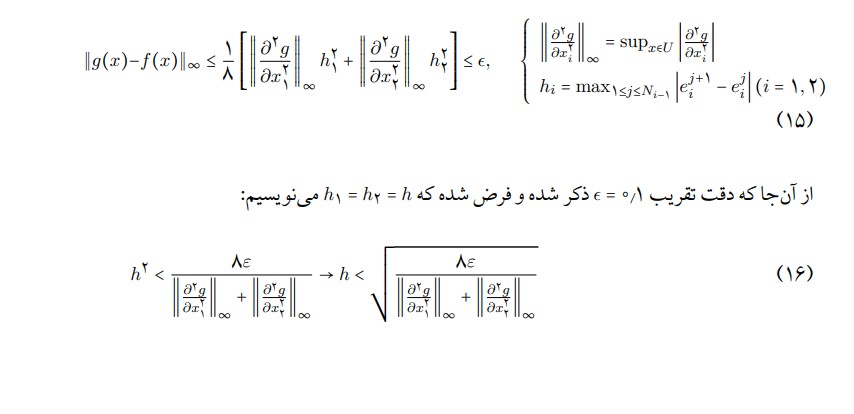


مقدار ماکسیمم تابع به ازای x1=-1 و x2=-1 محاسبه شده است :

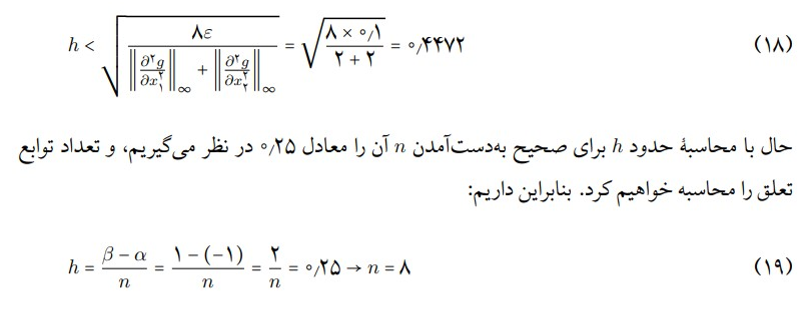
در نهایت می توانیم با جایگذاری مقادیر بدست آمده در رابطه بالا ، مقدار h و تعداد توابع تعلق را محاسبه کنیم.

بنابراین ما برای این سیستم فازی (کران مرتبه اول )، 41 تابع تعلق مثلثی خواهیم داشت.

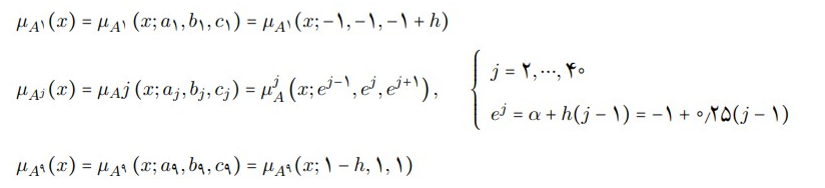
توابع تعلق به صورت زیر خواهند بود :

برای کران مرتبه دوم :

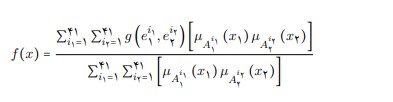
مقدار ماکسیمم تابع به ازای x1=-1 و x2=-1 محاسبه شده است :

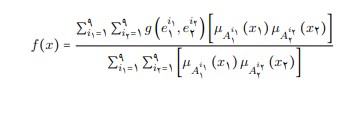


بنابراین ما برای این سیستم فازی ( کران مرتبه دوم ) ، 9 تابع تعلق مثلثی خواهیم داشت.

توابع تعلق به صورت زیر خواهند بود :

برای غیر فازی ساز میانگین داریم :

کران مرتبه اول

کران مرتبه دوم

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

برای غیر فازی ساز میانگین و کران مرتبه اول :

clc;

clear;

close all;

%% First order limit

alpha = -1;

beta = 1;

h = 0.05;

N = 41;

x1 = alpha:0.01:beta;

x2 = alpha:0.01:beta;

[x1, x2] = meshgrid(x1, x2);

g\_bar = zeros(N\*N, 1);

e\_i1 = zeros(N, 1);

e\_i2 = zeros(N, 1);

num = 0;

den = 0;

k = 1;

trimf = @(x, abc) max(min((x - abc(1)) / (abc(2) - abc(1)), (abc(3) - x) / (abc(3) - abc(2))), 0);

for i1 = 2:N

for i2 = 2:N

e\_i1(i1-1,1) = -1 + h\*(i1-2);

e\_i2(i2-1,1) = -1 + h\*(i2-2);

if i1 == 2

mu\_A\_x1 = trimf(x1, [-1, -1, -1+h]);

elseif i1 == N

mu\_A\_x1 = trimf(x1, [1-h, 1, 1]);

else

mu\_A\_x1 = trimf(x1, [-1+h\*(i1-3), -1+h\*(i1-2), -1+h\*(i1-1)]);

end

if i2 == 2

mu\_A\_x2 = trimf(x2, [-1, -1, -1+h]);

elseif i2 == N

mu\_A\_x2 = trimf(x2, [1-h, 1, 1]);

else

mu\_A\_x2 = trimf(x2, [-1+h\*(i2-3), -1+h\*(i2-2), -1+h\*(i2-1)]);

end

g\_bar(k,1) = 1 / (3 + e\_i1(i1-1,1) + e\_i2(i2-1,1));

num = num + g\_bar(k,1) \* mu\_A\_x1 .\* mu\_A\_x2;

den = den + mu\_A\_x1 .\* mu\_A\_x2;

k = k + 1;

end

end

f\_x = num ./ den;

g\_x = 1 ./ (3 + x1 + x2);

figure

surf(x1, x2, g\_x, 'FaceColor', 'b', 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

ylabel('$x\_2$', 'Interpreter', 'latex');

zlabel('g(x)', 'Interpreter', 'latex');

legend('$g(x)$', '$f(x)$', 'Interpreter', 'latex');

grid on;

figure

surf(x1, x2, f\_x, 'FaceColor', 'r', 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

ylabel('$x\_2$', 'Interpreter', 'latex');

zlabel('f(x)', 'Interpreter', 'latex');

legend('$f(x)$', 'Interpreter', 'latex');

grid on;

figure

surf(x1, x2, g\_x - f\_x, 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

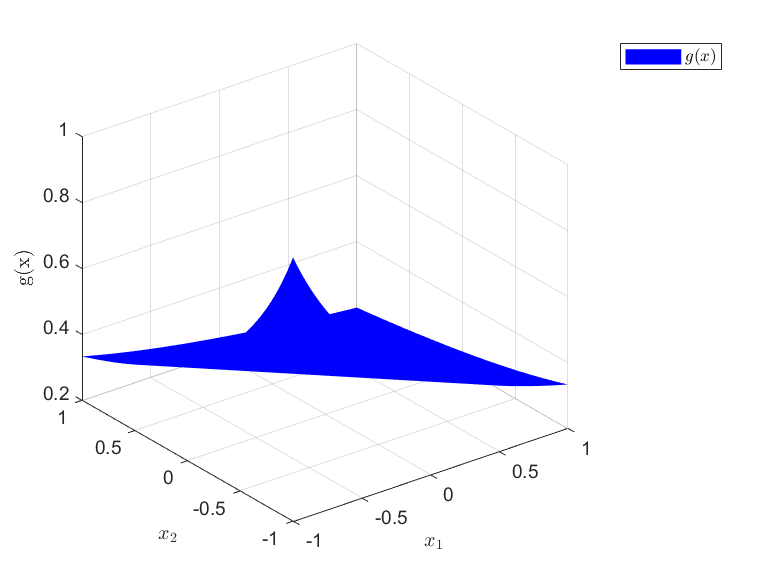
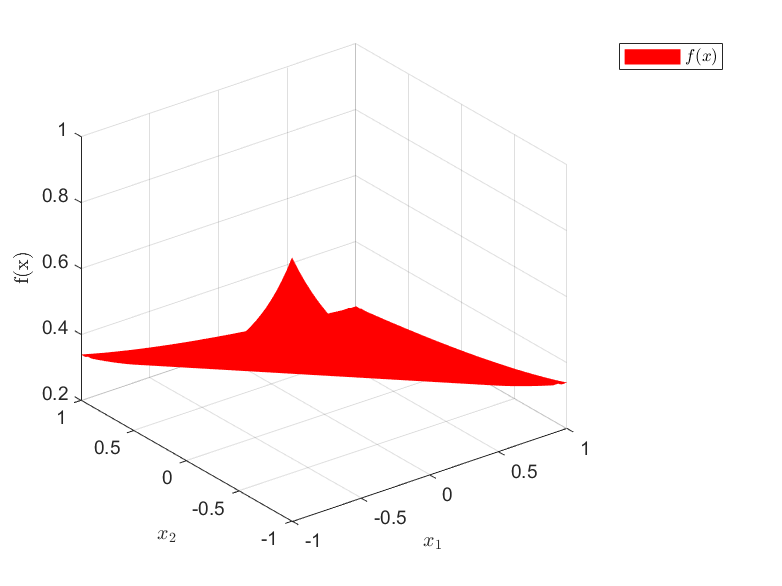
ylabel('$x\_2$', 'Interpreter', 'latex');

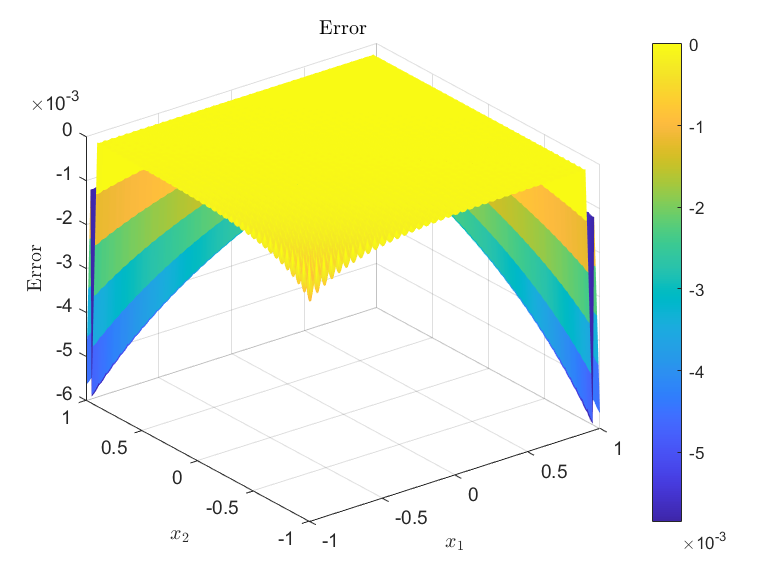
zlabel('Error', 'Interpreter', 'latex');

title('Error', 'Interpreter', 'latex');

colorbar;

grid on;

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



برای غیر فازی ساز میانگین و کران مرتبه دوم :

%% Second order limit

alpha = -1;

beta = 1;

h = 0.25;

N = 9;

x1 = alpha:0.01:beta;

x2 = alpha:0.01:beta;

[x1, x2] = meshgrid(x1, x2);

g\_bar = zeros(N\*N, 1);

e\_i1 = zeros(N, 1);

e\_i2 = zeros(N, 1);

num = 0;

den = 0;

k = 1;

% Define trimf function

trimf = @(x, abc) max(min((x - abc(1)) / (abc(2) - abc(1)), (abc(3) - x) / (abc(3) - abc(2))), 0);

% Loop to calculate memberships and g\_bar

for i1 = 2:N

for i2 = 2:N

e\_i1(i1-1,1) = -1 + h\*(i1-2);

e\_i2(i2-1,1) = -1 + h\*(i2-2);

if i1 == 2

mu\_A\_x1 = trimf(x1, [-1, -1, -1+h]);

elseif i1 == N

mu\_A\_x1 = trimf(x1, [1-h, 1, 1]);

else

mu\_A\_x1 = trimf(x1, [-1+h\*(i1-3), -1+h\*(i1-2), -1+h\*(i1-1)]);

end

if i2 == 2

mu\_A\_x2 = trimf(x2, [-1, -1, -1+h]);

elseif i2 == N

mu\_A\_x2 = trimf(x2, [1-h, 1, 1]);

else

mu\_A\_x2 = trimf(x2, [-1+h\*(i2-3), -1+h\*(i2-2), -1+h\*(i2-1)]);

end

g\_bar(k,1) = 1 / (3 + e\_i1(i1-1,1) + e\_i2(i2-1,1));

num = num + g\_bar(k,1) \* mu\_A\_x1 .\* mu\_A\_x2;

den = den + mu\_A\_x1 .\* mu\_A\_x2;

k = k + 1;

end

end

% Calculate f\_x and g\_x

f\_x = num ./ den;

g\_x = 1 ./ (3 + x1 + x2);

figure

surf(x1, x2, g\_x, 'FaceColor', 'b', 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

ylabel('$x\_2$', 'Interpreter', 'latex');

zlabel('g(x)', 'Interpreter', 'latex');

legend('$g(x)$', '$f(x)$', 'Interpreter', 'latex');

grid on;

figure

surf(x1, x2, f\_x, 'FaceColor', 'r', 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

ylabel('$x\_2$', 'Interpreter', 'latex');

zlabel('f(x)', 'Interpreter', 'latex');

legend('$f(x)$', 'Interpreter', 'latex');

grid on;

figure

surf(x1, x2, g\_x - f\_x, 'EdgeColor', 'none');

xlabel('$x\_1$', 'Interpreter', 'latex');

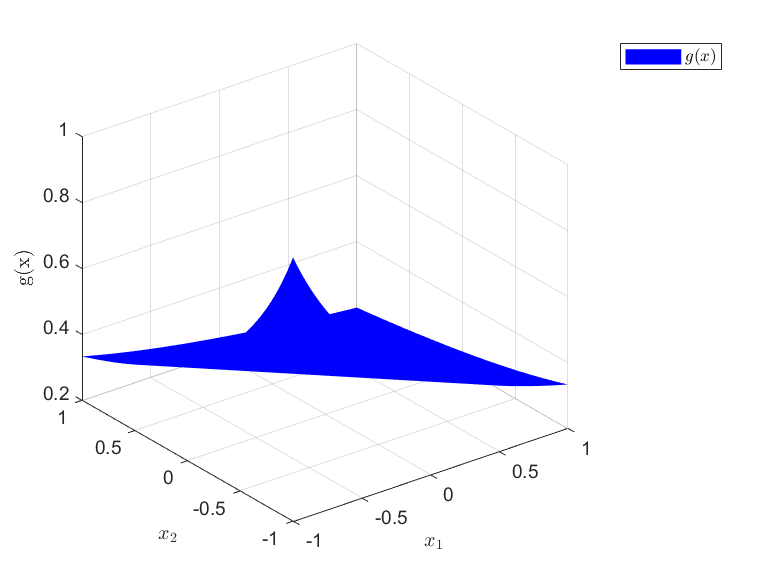
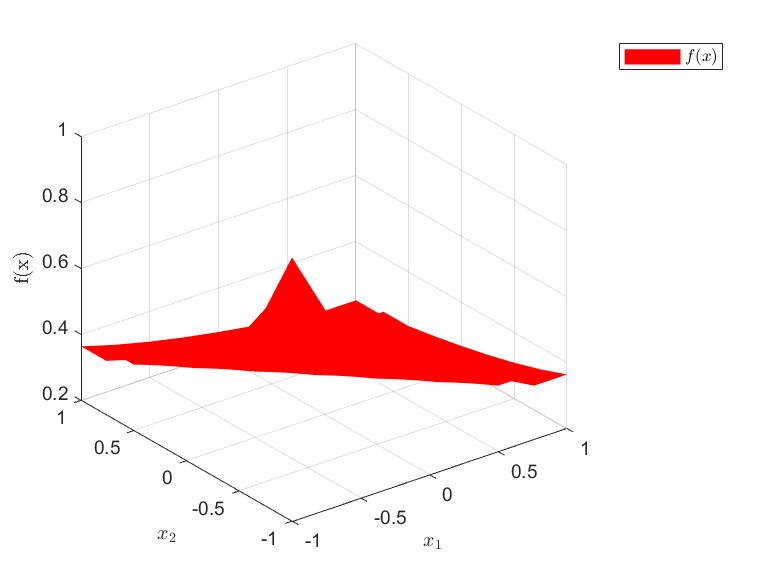
ylabel('$x\_2$', 'Interpreter', 'latex');

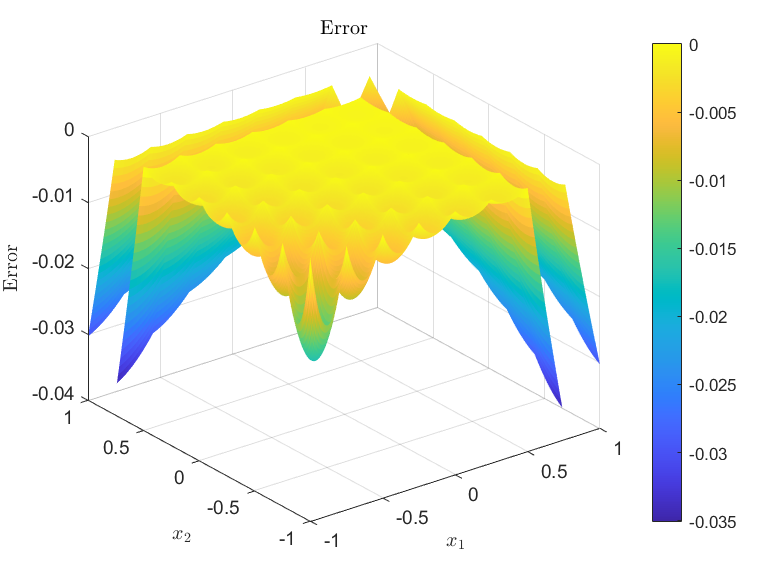
zlabel('Error', 'Interpreter', 'latex');

title('Error', 'Interpreter', 'latex');

colorbar;

grid on;

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



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

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

کران مرتبه اول :

%% First order limit (max)

alfa=-1;

beta=1;

h=0.05;

N=41;

x1=alfa:0.01:beta;

x2=x1;

[~,n1]=size(x1);

[~,n2]=size(x2);

e1=beta\*ones(1,N+1);

e2=beta\*ones(1,N+1);

for j=1:N

e1(j)=alfa+h\*(j-1);

e2(j)=alfa+h\*(j-1);

end

f\_x=zeros(n1,n2);

for k1=1:n1

for k2=1:n2

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

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

if x1(1,k1)>=e1(1,i1)&& x1(1,k1)<=.5\*(e1(1,i1)+e1(1,1+i1))&& x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=.5\*(e2(1,i2)+e2(1,1+i2))

p=0;

q=0;

elseif x1(1,k1)>=e1(1,i1)&& x1(1,k1)<=.5\*(e1(1,i1)+e1(1,1+i1))&& x2(1,k2)>=0.5\*(e2(1,i2)+e2(1,1+i2))&& x2(1,k2)<=e2(1,1+i2)

p=0;

q=1;

elseif x1(1,k1)>=.5\*(e1(1,i1)+e1(1,1+i1))&& x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=0.5\*(e2(1,i2)+e2(1,1+i2))

p=1;

q=0;

elseif x1(1,k1)>=.5\*(e1(1,i1)+e1(1,1+i1))&& x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=0.5\*(e2(1,i2)+e2(1,1+i2))&& x2(1,k2)<=e2(1,1+i2)

p=1;

q=1;

end

f\_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));

end

end

[x1,x2]=meshgrid(x1,x2);

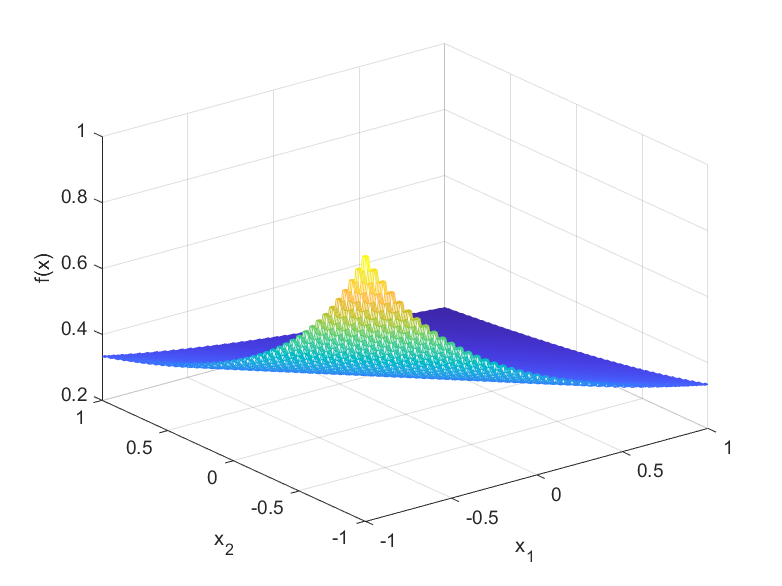
figure1 = figure('Color',[1 1 1]);

mesh(x1,x2,transpose(f\_x));

xlabel('x\_1')

ylabel('x\_2')

zlabel('f(x)')

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

کران مرتبه دوم :

%% ?Second order limit (max)

alfa=-1;

beta=1;

h=0.25;

N=9;

x1=alfa:0.01:beta;

x2=x1;

[~,n1]=size(x1);

[~,n2]=size(x2);

e1=beta\*ones(1,N+1);

e2=beta\*ones(1,N+1);

for j=1:N

e1(j)=alfa+h\*(j-1);

e2(j)=alfa+h\*(j-1);

end

f\_x=zeros(n1,n2);

for k1=1:n1

for k2=1:n2

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

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

if x1(1,k1)>=e1(1,i1)&& x1(1,k1)<=.5\*(e1(1,i1)+e1(1,1+i1))&& x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=.5\*(e2(1,i2)+e2(1,1+i2))

p=0;

q=0;

elseif x1(1,k1)>=e1(1,i1)&& x1(1,k1)<=.5\*(e1(1,i1)+e1(1,1+i1))&& x2(1,k2)>=0.5\*(e2(1,i2)+e2(1,1+i2))&& x2(1,k2)<=e2(1,1+i2)

p=0;

q=1;

elseif x1(1,k1)>=.5\*(e1(1,i1)+e1(1,1+i1))&& x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=e2(1,i2)&& x2(1,k2)<=0.5\*(e2(1,i2)+e2(1,1+i2))

p=1;

q=0;

elseif x1(1,k1)>=.5\*(e1(1,i1)+e1(1,1+i1))&& x1(1,k1)<=e1(1,1+i1)&& x2(1,k2)>=0.5\*(e2(1,i2)+e2(1,1+i2))&& x2(1,k2)<=e2(1,1+i2)

p=1;

q=1;

end

f\_x(k1,k2)=1/(3+e1(1,i1+p)+e2(1,i2+q));

end

end

[x1,x2]=meshgrid(x1,x2);

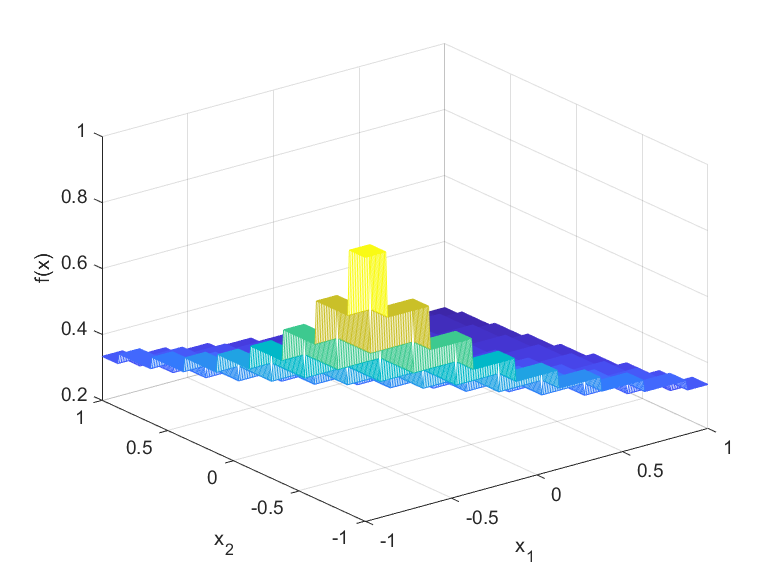
figure1 = figure('Color',[1 1 1]);

mesh(x1,x2,transpose(f\_x));

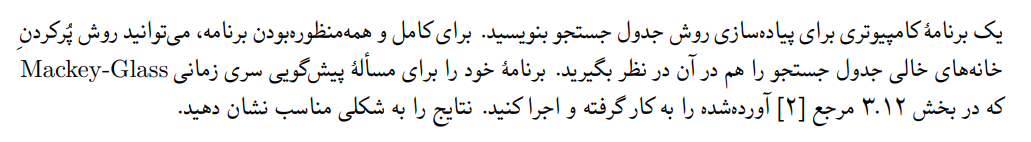
xlabel('x\_1')

ylabel('x\_2')

zlabel('f(x)')

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

## سوال دوم :



کد بخش برگرفته از مرجع 1 :

clc;

clear;

close all;

%% Data generation by Mackey-Glass chaotic time series

n=900; % Total number of sampling

% Preallocations

x=zeros (1, n);

dataset\_1=zeros (n, 7);

x(1,1:31)=1.3+0.2\*rand;

for k=31:n-1

x (1, k+1)=0.2\* ((x(1, k-30))/ (1+x (1, k-30)^10))+0.9\*x(1, k);

dataset\_1 (k, 2:6)= [x(1, k-3) x(1, k-2) x(1, k-1) x(1, k) x(1, k+1)];

end

dataset (1:600, 2:6)=dataset\_1 (201: 800, 2:6);

t=1:600;

figurel = figure ('Color', [1 1 1]); plot (t,x (201:800), 'Linewidth', 2)

grid on

[Number\_training, ~]=size (dataset);

Rul=zeros (Number\_training/2,6);

Rules\_total=zeros (Number\_training/2, 6);

%% designing fuzzy system considering two cases:

% (assigning 7 membership functions for each input variables)

% s=1 ;

% (assigning 15 membership functions for each input variables)

% s=2 ;?

for s=1:2

switch s

case 1

num\_membership\_functions=7; c=linspace (0.5, 1.3,5);

h=0.2;

membership\_functions=cell(num\_membership\_functions, 2);

for k=1:num\_membership\_functions

if k==1

membership\_functions {k, 1}= [0, 0, 0.3, 0.5];

membership\_functions {k, 2}='trapmf';

elseif k==num\_membership\_functions

membership\_functions{k, 1}=[1.3, 1.5, 1.8, 1.8];

membership\_functions {k, 2}='trapmf';

else

membership\_functions {k, 1}=[c(k-1)-h, c(k-1), c(k-1)+h];

membership\_functions {k, 2}='trimf';

end

end

case 2

num\_membership\_functions=15;

c=linspace(0.3,1.5, 13);

h=0.1;

membership\_functions=cell(num\_membership\_functions, 2);

for k=1:num\_membership\_functions

if k==1

membership\_functions{k, 1}=[0, 0, 0.2, 0.3];

membership\_functions{k, 2}='trapmf';

elseif k==num\_membership\_functions

membership\_functions{k, 1}=[1.5, 1.6, 1.8, 1.8];

membership\_functions{k,2}='trapmf';

else

membership\_functions{k, 1}=[c(k-1)-h, c(k-1), c(k-1)+h];

membership\_functions{k,2}='trimf';

end

end

end

%% Assign degree to each rule

vec\_x=zeros (1, num\_membership\_functions);

vec=zeros (1,5);

for t=1: Number\_training

dataset(t, 1)=t;

for i=2:6

x=dataset(t, i);

for j=1:num\_membership\_functions

if j==1

vec\_x (1, j) = trapmf(x, membership\_functions {1,1});

elseif j==num\_membership\_functions

vec\_x (1, j)=trapmf (x, membership\_functions{num\_membership\_functions, 1});

else

vec\_x (1, j) = trimf (x, membership\_functions {j,1});

end

end

[valu\_x, column\_x]=max(vec\_x);

vec (1, i-1)=max (vec\_x);

Rules(t, i-1)=column\_x;

Rules(t, 6) =prod(vec);

dataset (t,7) =prod(vec);

end

end

%% Delete extra rules

Rules\_total(1, 1:6)=Rules(1,1:6);

i=1;

for t=2:Number\_training

m=zeros (1,1);

for j=1:i

m(1, j)=isequal(Rules(t, 1:4), Rules\_total(j, 1:4));

if m(1,j)==1 && Rules(t, 6)>=Rules\_total (j,6)

Rules\_total(j, 1:6)=Rules (t, 1:6);

end

end

if sum (m)==0

Rules\_total(i+1, 1:6)=Rules(t, 1:6);

i=i+1;

end

end

%%

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

disp(['Final rules for ', num2str(num\_membership\_functions),' membership functions for each input variables'])

final\_Rules=Rules\_total(1:1, :);

%% Create Fuzzy Inference System

Fisname='Prediction controller';

Fistype='mamdani';

Andmethod='prod';

Ormethod='max';

Impmethod='prod';

Aggmethod='max';

Defuzzmethod='centroid';

fis=newfis(Fisname, Fistype, Andmethod, Ormethod, Impmethod, Aggmethod, Defuzzmethod);

%% Add Variables

for num\_input = 1:4

fis = addInput(fis, [0.1 1.7], "Name", ['x', num2str(num\_input)]);

end

fis = addOutput(fis,[0.1, 1.7], 'Name', 'x5');

%% Add Membership functions

for num\_input = 1:4

for input\_Rul = 1:num\_membership\_functions

fis = addMF(fis, ['x', num2str(num\_input)], membership\_functions{input\_Rul,2},membership\_functions{input\_Rul,1}, 'Name', ['A', num2str(input\_Rul)]);

end

end

for input\_Rul = 1:num\_membership\_functions

fis = addMF(fis, 'x5',membership\_functions{input\_Rul, 2}, membership\_functions{input\_Rul, 1}, 'Name', ['MF\_', num2str(input\_Rul)]);

end

%% Add Rules

non\_zero\_rows = any(Rules\_total(:, 1:5), 2); % Find rows with non-zero rules

fis\_Rules = ones(sum(non\_zero\_rows), 7);

fis\_Rules(:, 1:6) = Rules\_total(non\_zero\_rows, 1:6);

fis = addrule(fis, fis\_Rules);

%% Prediction of 300 points of chosen dataset

jadval\_prediction=zeros(300,2);

f=1;

for i=301:600

input=dataset(i, 2:6);

output1=dataset(i, 6);

x5=evalfis([input(1, 1); input(1, 2); input(1,3); input(1,4)], fis);

jadval\_prediction(f, :)= [f, x5];

f=f+1;

end

figure;

plot(jadval\_prediction(:,1),jadval\_prediction(:,2), 'r-.', 'Linewidth', 2);

hold on;

grid on

plot(jadval\_prediction(:,1),dataset(301: 600, 6), 'b', 'Linewidth', 2);

legend('estimate value', 'real value')

grid on

end

% Assuming 'fis' is your fuzzy inference system

inputVariableIndex = 1; % Change this to the index of the input variable you're interested in

% Plot the membership functions for the specified input variable

figure;

plotmf(fis, 'output', inputVariableIndex);

grid on

title(['Membership Functions for Input Variable ', num2str(inputVariableIndex)]);

توضیح روند عملکرد :

1-تولید داده

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

2-طراحی سیستم منطق فازی

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

- حالت اول با 7 تابع عضویت برای هر ورودی.

- حالت دوم با 15 تابع عضویت برای هر ورودی.

پارامترهای توابع عضویت به صورت دستی تعریف شده‌اند.

3-تخصیص درجه به هر قانون

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

4- حذف قوانین اضافی

قوانین اضافی حذف می‌شوند تا سیستم فازی ساده‌تر شود.

5-ایجاد سیستم منطق فازی

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

6-پیش‌بینی مقادیر

با استفاده از سیستم منطق فازی ایجاد شده، 300 نقطه آینده از مجموعه داده پیش‌بینی می‌شوند. نتایج پیش‌بینی با نتایج واقعی مقایسه شده و در یک نمودار نمایش داده می‌شوند.

7- نمایش توابع عضویت

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

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

شکل نتایج :



شکل 1) پیش بینی مدل سری زمانی



شکل 2) پیش بینی دقیق تر مدل سری زمانی



## سوال سوم :

کد بخش برگرفته از مرجع 2 :

clc;

clear;

close all;

%% Initializing

M=4; %Number of membership functions (Based on 1st step of fuzzy system design)

num\_training=200; % Number of training

total\_num=700;

landa=0.1; % A constant stepsize

% Preallocation

x\_bar=zeros (num\_training, M);

g\_bar=zeros (num\_training, M);

sigma=zeros (num\_training, M);

y=zeros(total\_num, 1);

u=zeros(total\_num, 1);

x=zeros(total\_num, 1);

y\_hat=zeros(total\_num, 1);

f\_hat=zeros(total\_num, 1);

z=zeros(total\_num, 1);

g\_u=zeros(total\_num, 1);

u(1)=-1+2\*rand;

y(1)=0;

g\_u(1)=0.6\*sin(pi\*u(1))+0.3\*sin(3\*pi\*u(1))+0.1\*sin(5\*pi\*u(1));

f\_hat(1)=g\_u(1);

%% Based on the 1st step of fuzzy system design

u\_min=-1;

u\_max=1;

h=(u\_max-u\_min)/(M-1);

for k=1:M

x\_bar(1, k)=-1+h\*(k-1);

u(1,k) =x\_bar(1, k);

g\_bar(1,k)=0.6\*sin(pi\*u(1,k))+0.3\*sin(3\*pi\*u(1,k))+0.1\*sin(5\*pi\*u(1,k));

end

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

x\_bar(2,:)=x\_bar(1, :);

g\_bar(2,:)=g\_bar(1, :);

sigma(2, :)=sigma(1,:);

x\_bar\_initial=x\_bar(1, :);

sigma\_initial=sigma(1, :);

y\_bar\_initial=g\_bar(1,:);

%% Based on the 2nd and 3rd step of fuzzy system design for q=2: num\_training

for q=2:num\_training

b=0;a=0;

x(q)=-1+2\*rand;

u(q)=x(q);

g\_u(q)=0.6\*sin(pi\*u(q))+0.3\*sin(3\*pi\*u(q))+0.1\*sin(5\*pi\*u(q));

for l=1:M

z(l)=exp(-((x(q)-x\_bar(q,l))/sigma(q, l))^2);

b=b+z(l);

a=a+g\_bar(q, l)\*z(l);

end

f\_hat (q)=a/b;

y(q+1)=0.3\*y(q)+0.6\*y(q-1)+g\_u(q);

y\_hat(q+1)=0.3\*y(q)+0.6\*y(q-1)+f\_hat(q);

for l=1:M

g\_bar(q+1,l)=g\_bar(q,l)-landa\*(f\_hat(q)-g\_u(q))\*z(l)/b;

x\_bar(q+1,l)=x\_bar(q,l)-landa\*((f\_hat(q)-g\_u(q))/b)\*(g\_bar(q,l)-f\_hat(q))\*z(l)\*2\*(x(q)-x\_bar(q,l))/(sigma(q,l)^2);

sigma (q+1,l)=sigma(q, l)-landa\*((f\_hat(q)-g\_u(q))/b)\*(g\_bar(q,l)-f\_hat(q))\*z(l)\*2\*(x(l)-x\_bar(q,l))^2/(sigma(q,l)^3);

end

end

x\_bar\_final=x\_bar(num\_training,:);

sigma\_final=sigma(num\_training,:);

g\_bar\_final=g\_bar(num\_training,:);

for q=num\_training:700

b=0;

a=0;

x(q)=sin(2\*q\*pi/200);

u(q)=x(q);

g\_u(q)=0.6\*sin(pi\*u(q))+0.3\*sin(3\*pi\*u(q))+0.1\*sin(5\*pi\*u(q));

for l=1: M

z(l)=exp(-((x(q)-x\_bar(num\_training,l))/sigma(num\_training, l))^2);

b=b+z(l);

a=a+g\_bar(num\_training, l)\*z(l);

end

f\_hat(q)=a/b;

y(q+1)=0.3\*y(q)+0.6\*y(q-1)+g\_u(q);

y\_hat(q+1)=0.3\*y(q)+0.6\*y(q-1)+f\_hat(q);

end

%% Plots and Figures

figure1=figure('Color', [1 1 1]);

plot(1:701, y, 'b', 1:701, y\_hat, 'r:', 'Linewidth', 2);

legend('output of the plant', 'output of the identification model')

axis([0 701 -5 5]);

grid on

figure2=figure('Color', [1 1 1]);

xp=-2:0.001:2;

for l=1:M

miu\_x=exp(-((xp-x\_bar(1, l))./(sigma (1,l))).^2);

plot(xp, miu\_x, 'Linewidth', 2);

hold on

end

xlabel('u');

ylabel('initial MF''s');

axis([-1 1 0 1]);

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

for l=1:M

miu\_x=exp(-((xp-x\_bar(num\_training, l))./ (sigma (num\_training, l))).^2);

plot (xp, miu\_x, 'Linewidth', 2);

hold on

end

xlabel('u');

ylabel('final MF''s');

axis ([-1 1 0 1]);

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

1- متغیرها و پارامترها

: Mتعداد توابع عضویت مربوط به سیستم فازی.

num\_training :تعداد نمونه‌های استفاده شده برای آموزش مدل.

:total\_numتعداد کل نمونه‌ها (آموزش و تست).

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

2-پیش‌پردازش و مقداردهی اولیه

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

مقادیر اولیه برای ورودی‌ها و خروجی‌ها تعیین می‌شوند.

3-مقداردهی اولیه بر اساس توابع عضویت

مقادیر اولیه برای توابع عضویت ورودی‌ها براساس توزیع یکسان در بازه [-1,1] محاسبه می‌شوند.

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

4- آموزش مدل

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

5-آزمون مدل

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

6- نمودار توابع عضویت اولیه

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

7- نمودار توابع عضویت نهایی

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

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

نمودار نتایج :



## سوال چهارم :

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

1-گره‌ها

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

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

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

2- شاخه‌ها

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

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

3- ریشه

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

- ریشه به سوالی مرتبط با شرایط اولیه داده‌ها می‌پردازد.

4- گره‌های برگ

- گره‌های برگ پیش‌بینی یا دسته‌بندی نهایی را انجام می‌دهند.

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

5- شرایط و سوالات

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

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

6- آموزش

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

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

7- پیش‌بینی

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

- نمونه از ریشه تا گره‌های برگ پیش‌بینی می‌شود.

8- افزایش تفسیرپذیری

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

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

سوال 1 )

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

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from graphviz import Digraph

!pip install --upgrade --no-cache-dir gdown

!gdown 1UCDlb6gatarImiHiLnrDKDVrUqLQq6oW

data = pd.read\_csv('/content/covid.csv')

data

labels = data['Infected']

len(labels), labels.unique(), labels.value\_counts()

p = labels.value\_counts() / len(labels)

-sum(p \* np.log2(p))

def entropy(labels):

    p = labels.value\_counts() / len(labels)

    return -sum(p \* np.log2(p))

data['Infected'].value\_counts()

 entropy\_child = 0

 for value in data['Cough'].unique():

        subset = data[data['Cough'] == value]

        print(subset)

        wi = len(subset) / len(data)

        entropy\_child += wi \* entropy(subset['Infected'])

entropy\_child

def entropy(labels):

    p = labels.value\_counts() / len(labels)

    return -sum(p \* np.log2(p))

target = 'Infected'

entropy\_parent = entropy(data[target])

entropy\_parent

entropy\_child = 0

feature = 'Fever'

for value in data[feature].unique():

    subset = data[data[feature] == value]

    display(subset)

    wi = len(subset) / len(data)

    entropy\_child += wi \* entropy(subset[target])

information\_gain = entropy\_parent - entropy\_child

print(information\_gain)

def information\_gain(data, feature, target):

    # Entropy of parent

    entropy\_parent = entropy(data[target])

    # Entropy of child

    entropy\_child = 0

    for value in data[feature].unique():

        subset = data[data[feature] == value]

        #display(subset)

        wi = len(subset) / len(data)

        entropy\_child += wi \* entropy(subset[target])

    return entropy\_parent - entropy\_child

arg=[information\_gain(data, feature, 'Infected') for feature in data.iloc[:, :-1].columns]

def information\_gain(data, feature, target):

    # Entropy of parent

    entropy\_parent = entropy(data[target])

    # Entropy of child

    entropy\_child = 0

    for value in data[feature].unique():

        subset = data[data[feature] == value]

        wi = len(subset) / len(data)

        entropy\_child += wi \* entropy(subset[target])

    return entropy\_parent - entropy\_child



تحلیل کد :

در کد بالا پس از ایمپورت کردن اطلاعات در کولب ، داده ها را تقسیم بندی کرده ایم و سپس با استفاده از فرمول آنتروپی ، مقدار گین ویژگی های مختلف را برای مشخص کردن root node بدست آورده ایم. با توجه به کد بالا می بینیم که ویژگی ستون 2 یعنی ویژگی مشکل تنفسی دارای بیشترین گین می باشد. پس به سراغ تشکیل درخت تصمیم با استفاده از تعریف تابع می رویم :

class Node:

  def \_\_init\_\_(self, feature=None, label=None):

    self.feature = feature

    self.label = label

    self.children = {}

  def \_\_repr\_\_(self):

    if self.feature is not None:

      return f'DecisionNode(feature="{self.feature}", children={self.children})'

    else:

      return f'LeafNode(label="{self.label}")'

def make\_tree(data, target):

  # leaf node?

  if len(data[target].unique()) == 1:

    return Node(label=data[target].iloc[0])

  features = data.drop(target, axis=1).columns

  if len(features) == 0 or len(data) == 0:

    return Node(label=data[target].mode()[0])

  # calculate information gain

  gains = [information\_gain(data, feature, target) for feature in features]

  # greedy search to find best fearure

  max\_gains\_idx = np.argmax(gains)

  best\_features = features[max\_gains\_idx]

  # make a node

  node = Node(feature=best\_features)

  # loop over the best feature

  for value in data[best\_features].unique():

    subset = data[data[best\_features] == value].drop(best\_features, axis=1)

    # display(subset)

    node.children[value] = make\_tree(subset, target)

  return node

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

تحلیل کوتاه کد:

1. کلاس Node

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

- تابع repr برای نمایش متنی مناسب گره‌ها است.

2. تابع make\_tree

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

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

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

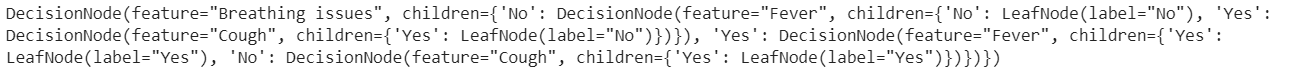
- اگر موارد بالا نقصانی ایجاد نکنند، اطلاعات گنجانده شده (Information Gain) برای هر ویژگی محاسبه می‌شود.

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

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

tree = make\_tree(data, 'Infected')

tree



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

در نهایت نمودار درختی مربوطه را رسم می کنیم :

def visualize\_tree(tree, parent=None, node\_id=None):

    if node\_id is None:

        node\_id = '0'

        g = Digraph(node\_attr={'shape': 'record', 'height':'.1'})

        g.node(node\_id, label=tree.feature)

    else:

        g = parent

        g.node(node\_id, label=tree.feature)

    if len(tree.children) == 0:

        g.node(node\_id, label=tree.label)

        return g

    for i, (value, child) in enumerate(tree.children.items()):

        child\_id = f'{node\_id}\_{i+1}'

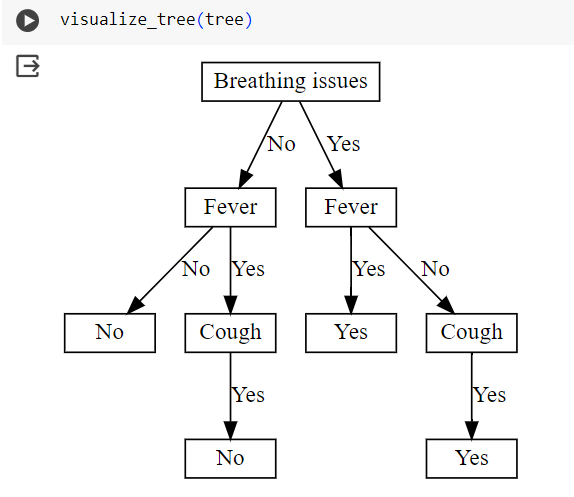
        visualize\_tree(child, g, child\_id)

        g.edge(node\_id, child\_id, label=value)

    return g

g = visualize\_tree(tree)

g.render('decision\_tree', format='png', view=True)



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

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

سوال 2 )

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

from sklearn import metrics

# Load breast cancer dataset

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=83)

# Create a decision tree classifier

# You can experiment with different hyperparameters, including pruning-related ones

# Example with max\_depth as a pruning parameter

max\_depth\_values = [5 ,10]  # Replace with your desired values

for max\_depth in max\_depth\_values:

    clf = DecisionTreeClassifier(max\_depth=max\_depth)

    # Train the model

    clf.fit(X\_train, y\_train)

    # Plot the decision tree

    plt.figure(figsize=(12, 8))

    plot\_tree(clf, filled=True, feature\_names=data.feature\_names, class\_names=data.target\_names)

    plt.title(f'Decision Tree - Max Depth: {max\_depth}')

    plt.savefig(f'decision\_tree\_max\_depth\_{max\_depth}.png')

    plt.show()

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

1. مدل با عمق 5

- این درخت تصمیم با عمق 5 به صورت گسترده‌تر و کلی‌تر اطلاعات را در اختیار می‌گیرد. این می‌تواند به خاطر کمتر بودن عمق، از برخی اطلاعات خاص و ارتباطات محلی چشم‌پوشی کند.

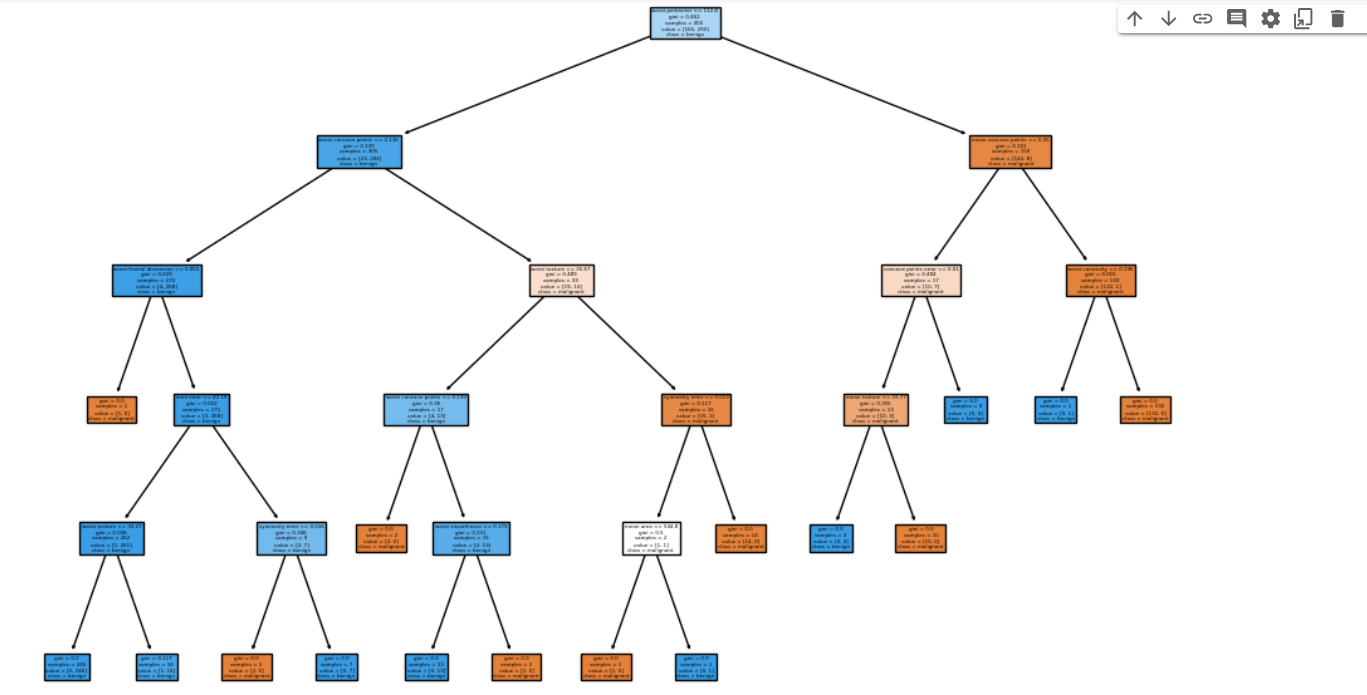
- با توجه به مقدار عمق 5، درخت احتمالاً به دنبال اطلاعات مهم و کلان در مورد سرطان پستان است.

2. مدل با عمق 10

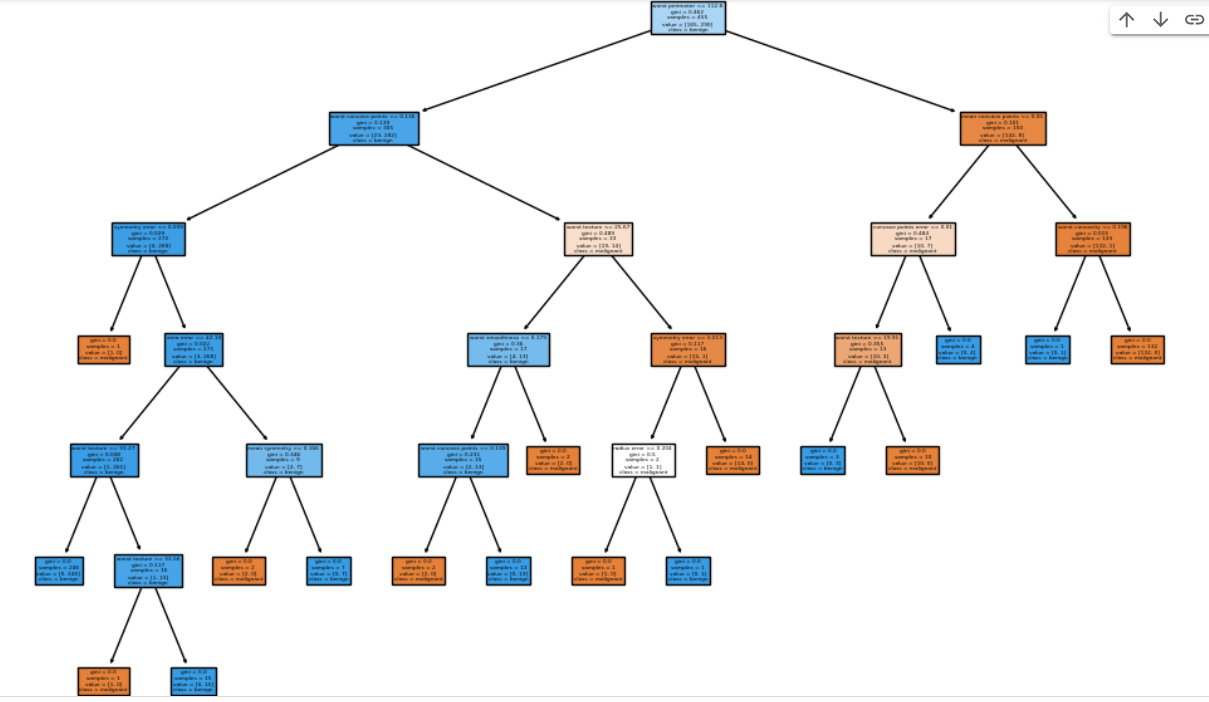
- این درخت با عمق بیشتر (10)، اطلاعات دقیق‌تر و خاص‌تری را در اختیار می‌گیرد. این ممکن است به دلیل این باشد که در این حالت، درخت قادر به درک اطلاعات محلی و تفاوت‌های کوچک‌تر در داده‌ها می‌شود.

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

نمایش نمودار ها :

درخت با عمق 5 :

درخت با عمق 10 :



تغییر پارامتر هرس کردن :

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

from sklearn import metrics

# Load breast cancer dataset

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=83)

# Create a decision tree classifier

# You can experiment with different hyperparameters, including pruning-related ones

# Example with ccp\_alpha as a pruning parameter

ccp\_alpha\_values = [0.0, 0.01, 0.02]  # Replace with your desired values

for ccp\_alpha in ccp\_alpha\_values:

    clf = DecisionTreeClassifier(ccp\_alpha=ccp\_alpha)

    # Train the model

    clf.fit(X\_train, y\_train)

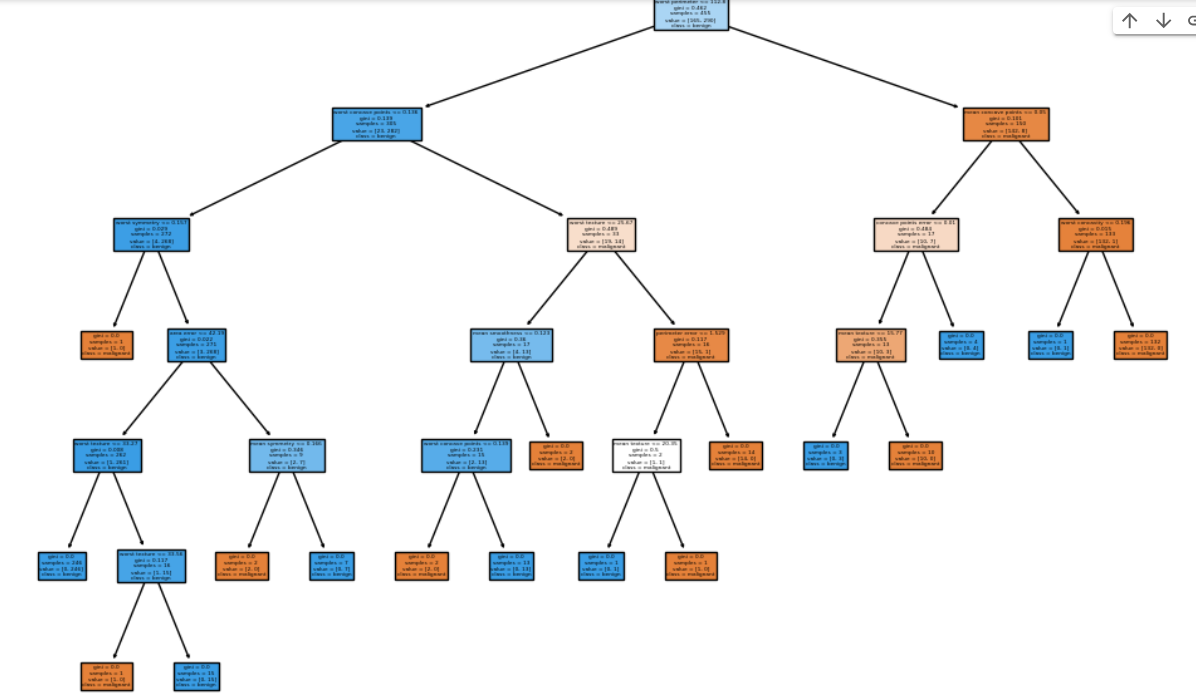
    # Plot the decision tree

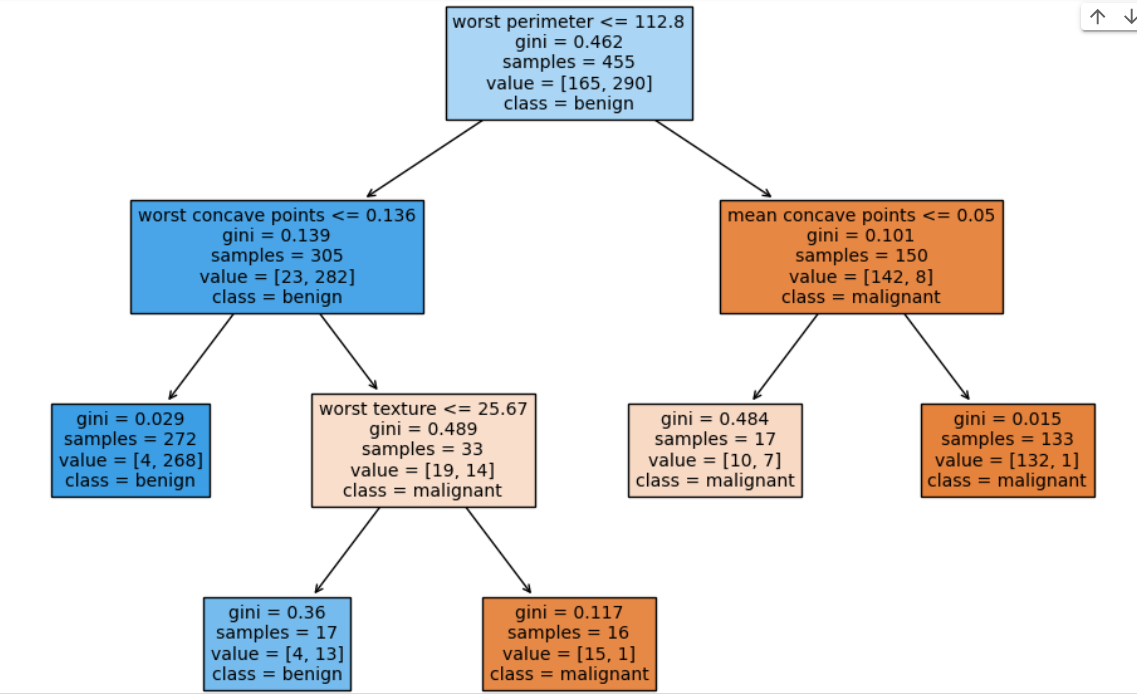
    plt.figure(figsize=(12, 8))

    plot\_tree(clf, filled=True, feature\_names=data.feature\_names, class\_names=data.target\_names)

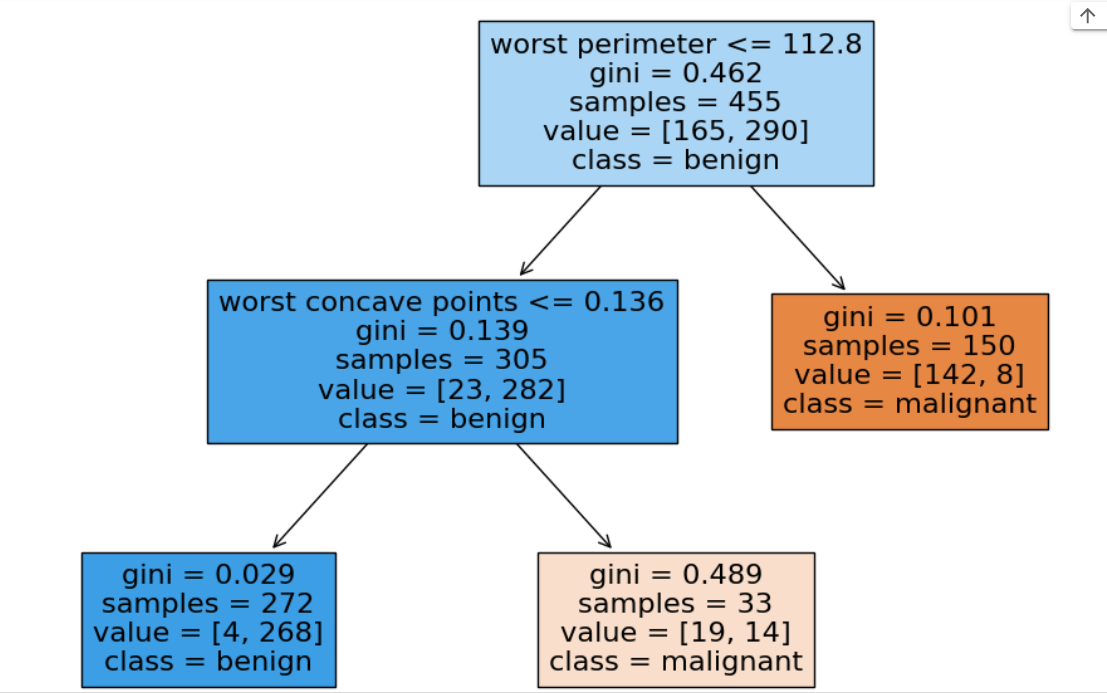
    plt.title(f'Decision Tree - ccp\_alpha: {ccp\_alpha}')

    plt.savefig(f'decision\_tree\_ccp\_alpha\_{ccp\_alpha}.png')

    plt.show()

شکل 3) ccp\_alpha = 0

شکل 4)ccp\_alpha=0.01



شکل 5) ccp\_alpha=0.02

حال مقدار ccp\_alpha را بزرگ در نظر می گیریم :

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

from sklearn import metrics

# Load breast cancer dataset

data = load\_breast\_cancer()

X = data.data

y = data.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=83)

# Create a decision tree classifier

# You can experiment with different hyperparameters, including pruning-related ones

# Example with ccp\_alpha as a pruning parameter

ccp\_alpha\_values = [0.5]  # Replace with your desired values

for ccp\_alpha in ccp\_alpha\_values:

    clf = DecisionTreeClassifier(ccp\_alpha=ccp\_alpha)

    # Train the model

    clf.fit(X\_train, y\_train)

    # Plot the decision tree

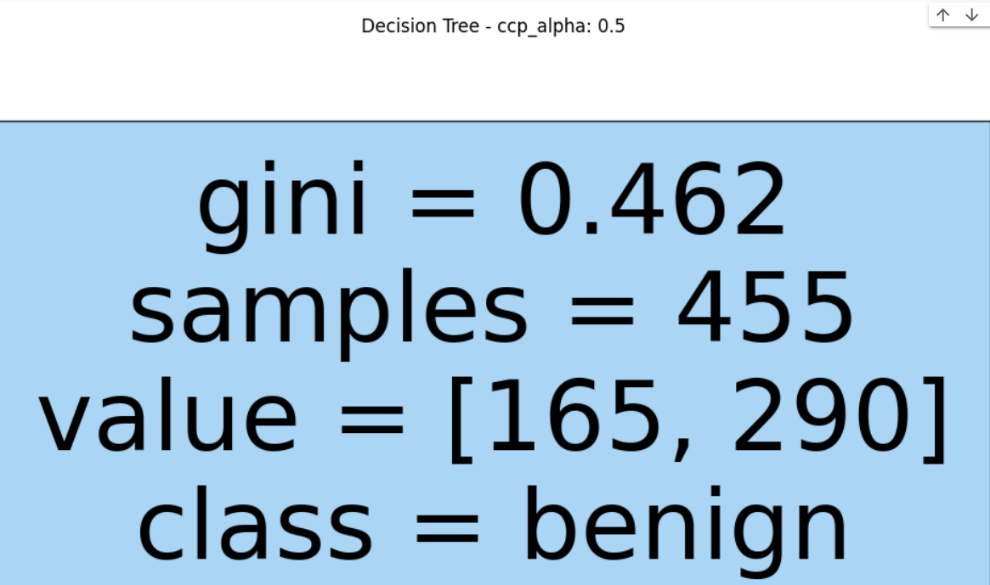
    plt.figure(figsize=(12, 8))

    plot\_tree(clf, filled=True, feature\_names=data.feature\_names, class\_names=data.target\_names)

    plt.title(f'Decision Tree - ccp\_alpha: {ccp\_alpha}')

    plt.savefig(f'decision\_tree\_ccp\_alpha\_{ccp\_alpha}.png')

    plt.show()



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

تأثیر افزایش ccp\_alpha شامل موارد زیر می‌شود:

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

2. ساختار ساده‌تر درخت : افزایش ccp\_alphaباعث می‌شود که درخت ساخته شده ساده‌تر باشد. بخش‌های درخت که با افزایش ccp\_alphaاضافه نمی‌شوند، حذف می‌شوند و این باعث می‌شود که درخت کلی ساده‌تر و قابل فهم‌تر باشد.

3. کاهش دقت در داده‌های آموزشی : افزایش ccp\_alphaممکن است باعث کاهش دقت مدل بر روی داده‌های آموزشی شود، زیرا مدل کمتر به داده‌های آموزشی نزدیک می‌شود.

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

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

    # Analyze two samples from the test set

    sample1 = X\_test[0]

    sample2 = X\_test[1]

    # Make predictions for the samples

    prediction1 = clf.predict([sample1])[0]

    prediction2 = clf.predict([sample2])[0]

    # Display the results

    print(f"\nAnalysis for Decision Tree with max\_depth={max\_depth}:\n")

    # Sample 1

    print("Sample 1:")

    print("Features:", sample1)

    print("True Label:", y\_test[0])

    print("Predicted Label:", prediction1)

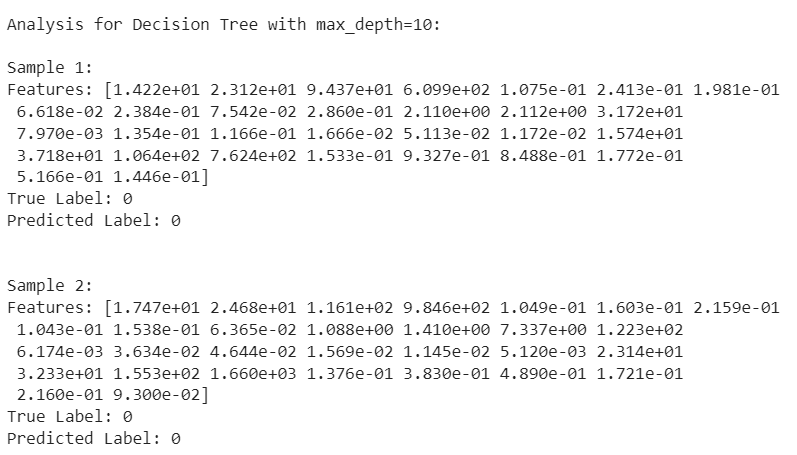
    print("\n")

    # Sample 2

    print("Sample 2:")

    print("Features:", sample2)

    print("True Label:", y\_test[1])

    print("Predicted Label:", prediction2)

در این کد، دو نمونه از مجموعه داده آزمون X\_testبا استفاده از یک مدل درخت تصمیم clf با پارامتر max\_depthمشخص شده، تحلیل شده‌اند. این دو نمونه به ترتیب با نام‌های `sample1` و `sample2` شناخته می‌شوند.

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

- برای `sample1`:

- ویژگی‌ها: مقادیر ویژگی‌های این نمونه.

- برچسب واقعی: برچسب واقعی متناظر با این نمونه از مجموعه داده آزمون (`y\_test`).

- برچسب پیش‌بینی شده: پیش‌بینی مدل برای این نمونه (`prediction1`).

- برای `sample2`:

- ویژگی‌ها: مقادیر ویژگی‌های این نمونه.

- برچسب واقعی: برچسب واقعی متناظر با این نمونه از مجموعه داده آزمون (`y\_test1`).

- برچسب پیش‌بینی شده: پیش‌بینی مدل برای این نمونه (`prediction2`).

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

محاسبه دقت برای دو نمونه درخت تصمیم ارائه شده :

    # Make predictions on the test set

    y\_pred = clf.predict(X\_test)

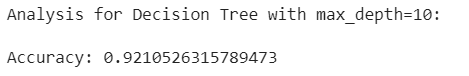
    # Calculate accuracy

    accuracy = metrics.accuracy\_score(y\_test, y\_pred)

    # Display the results

    print(f"\nAnalysis for Decision Tree with max\_depth={max\_depth}:\n")

    print("Accuracy:", accuracy)



# Set max\_depth to 5

max\_depth = 5

# Create a decision tree classifier

clf = DecisionTreeClassifier(max\_depth=max\_depth)

# Train the model

clf.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = clf.predict(X\_test)

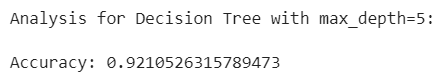
# Calculate accuracy

accuracy = metrics.accuracy\_score(y\_test, y\_pred)

# Display the results

print(f"\nAnalysis for Decision Tree with max\_depth={max\_depth}:\n")

print("Accuracy:", accuracy)



سواال 3 ) اختیاری

توضیحات مربوط به سوال :

اگرچه در گذشته تحقیقات زیادی درباره عوامل مؤثر بر امید زندگی با در نظر گرفتن متغیرهای جمعیتی، ترکیب درآمد و نرخ مرگ و میر انجام شده است، اما مشاهده شده است که تأثیر واکسیناسیون و شاخص توسعه انسانی در گذشته به درستی مورد توجه قرار نگرفته است. همچنین، برخی از تحقیقات گذشته با استفاده از مدل‌های رگرسیون خطی چندگانه براساس داده‌های یک ساله برای تمام کشورها انجام شده‌اند. بنابراین، این موضوع محرکی است برای حل هر دو عامل ذکر شده با فراهم آوردن یک مدل رگرسیون بر پایه مدل اثرات ترکیبی و رگرسیون خطی چندگانه در نظر گرفتن داده‌ها از سال 2000 تا 2015 برای تمام کشورها. واکسیناسیون‌های مهم مانند هپاتیت B، پلیو و دیفتریا نیز در نظر گرفته خواهند شد. به طور خلاصه، این مطالعه بر فاکتورهای واکسیناسیون، فاکتورهای مرگ و میر، فاکتورهای اقتصادی، فاکتورهای اجتماعی و سایر فاکتورهای مرتبط با سلامت تمرکز خواهد داشت. از آنجا که مشاهدات این مجموعه داده بر اساس کشورهای مختلف است، برای یک کشور بهتر است تا عامل پیش‌بینی‌کننده‌ای که به کاهش امید زندگی منجر می‌شود را تشخیص دهد. این به کشور کمک می‌کند تا بفهمد کدام حوزه باید با اهمیت بیشتری مورد توجه قرار گیرد تا به بهبود بهره‌وری امید زندگی جمعیت خود بپردازد.

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

شرح کد مربوطه :

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

# import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.model\_selection import train\_test\_split

# set style of visualization

sns.set\_style("whitegrid")

sns.set\_palette("RdBu")

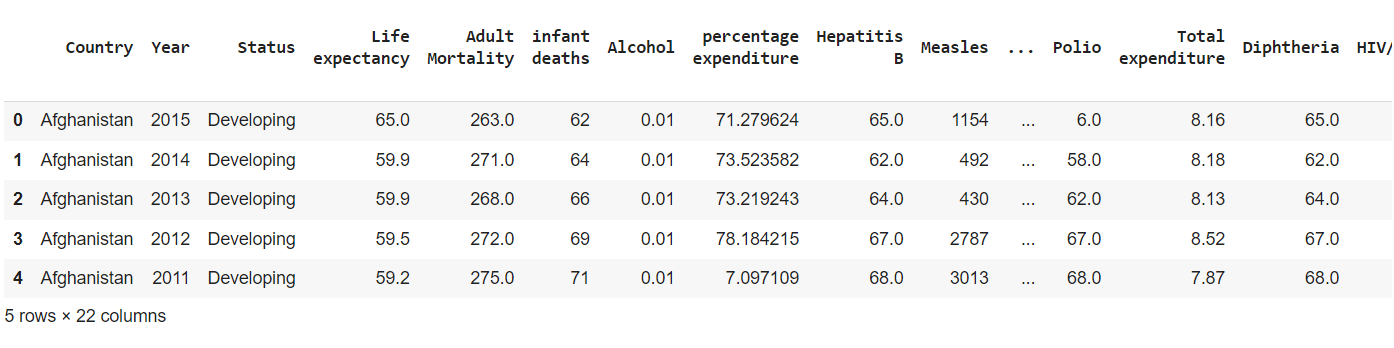
در ابتدا که کتابخانه های مختلف را در کولب ایمپورت می کنیم. سپس با استفاده از دستورات مربوط به درخت تصمیم به سراغ رگرسیون داده ها و پیش بینی داده های آزمون می رویم و سپس آن را با داده های واقعی مقایسه می کنیم و میزان خطا را مشخص می کنیم :

!pip install --upgrade --no-cache-dir gdown

!gdown 13UXkURa\_S\_QaHNsBO0m1UzbrVH1cakJK

data = pd.read\_csv('/content/Life Expectancy Data.csv')

data.head()



# first i see some column name with empty space i will fixed it to ease of use

data.columns = data.columns.str.strip()

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

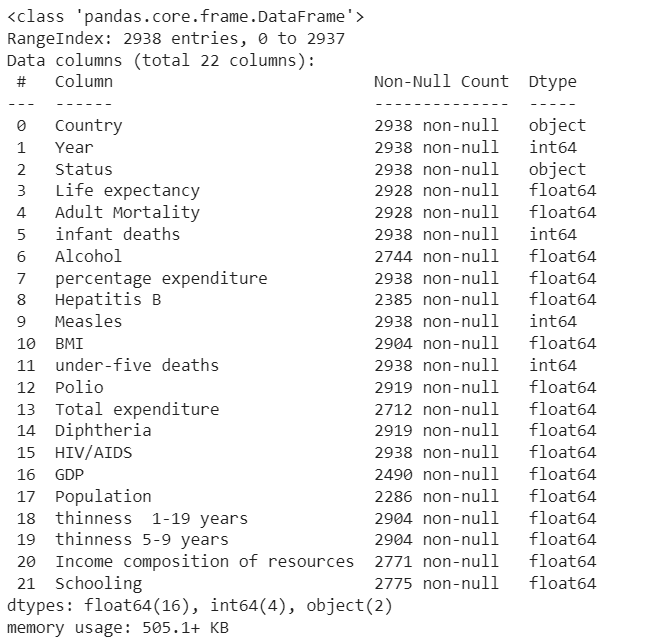
# Size of the data

data.shape



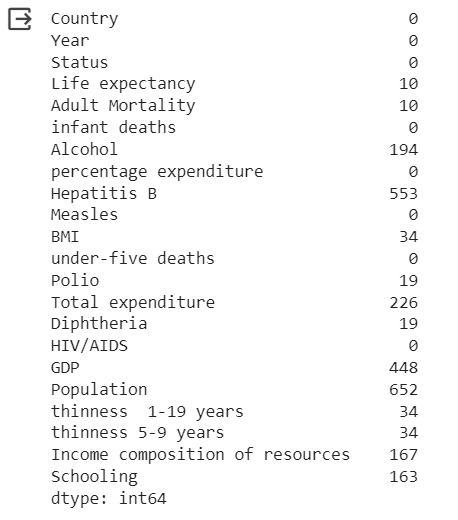
# A Quick Information about the Data

data.info()

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

# Checking for Null Values

data.isnull().sum()

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

# check if duplicated in data

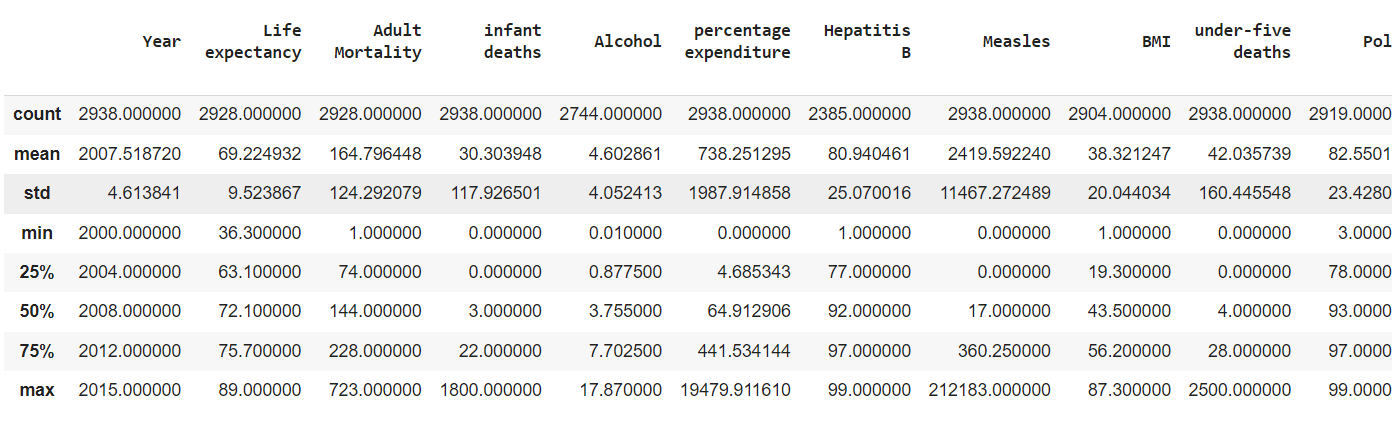
data.duplicated().any()

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

# see quick info of numeric values

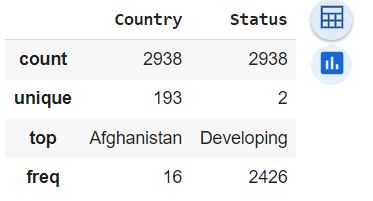
data.describe()

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



# see quick info of category values

data.describe(include = object)

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

# spliting data to train and test

train, test =  train\_test\_split(data, test\_size = 0.2, random\_state = 83)

در این دستور، داده‌ها به دو بخش آموزش train و آزمون test تقسیم می‌شوند. این کار با استفاده از train\_test\_split انجام می‌شود. پارامتر test\_size نسبت تعیین کننده‌ی حجم داده‌های آزمون است که در اینجا برابر با 0.2 یا 20% مجموع داده‌ها است. پارامتر random\_state تعیین‌کننده‌ی یک seed برای اطمینان از قابل تکرار بودن تقسیم داده‌ها است، به این ترتیب هر بار که این دستور اجرا می‌شود، نتایج یکسانی به دست می‌آید.

def fill\_train\_with\_median():

    return train.fillna(train.median(numeric\_only = True))

def fill\_test\_with\_median():

    return test.fillna(test.median(numeric\_only = True))

# Apply the function to data

train = fill\_train\_with\_median()

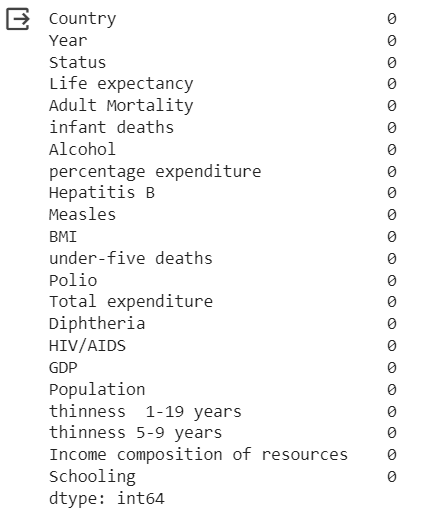
test = fill\_test\_with\_median()

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

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

train.isna().sum()

این دستور isna().sumتعداد 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.

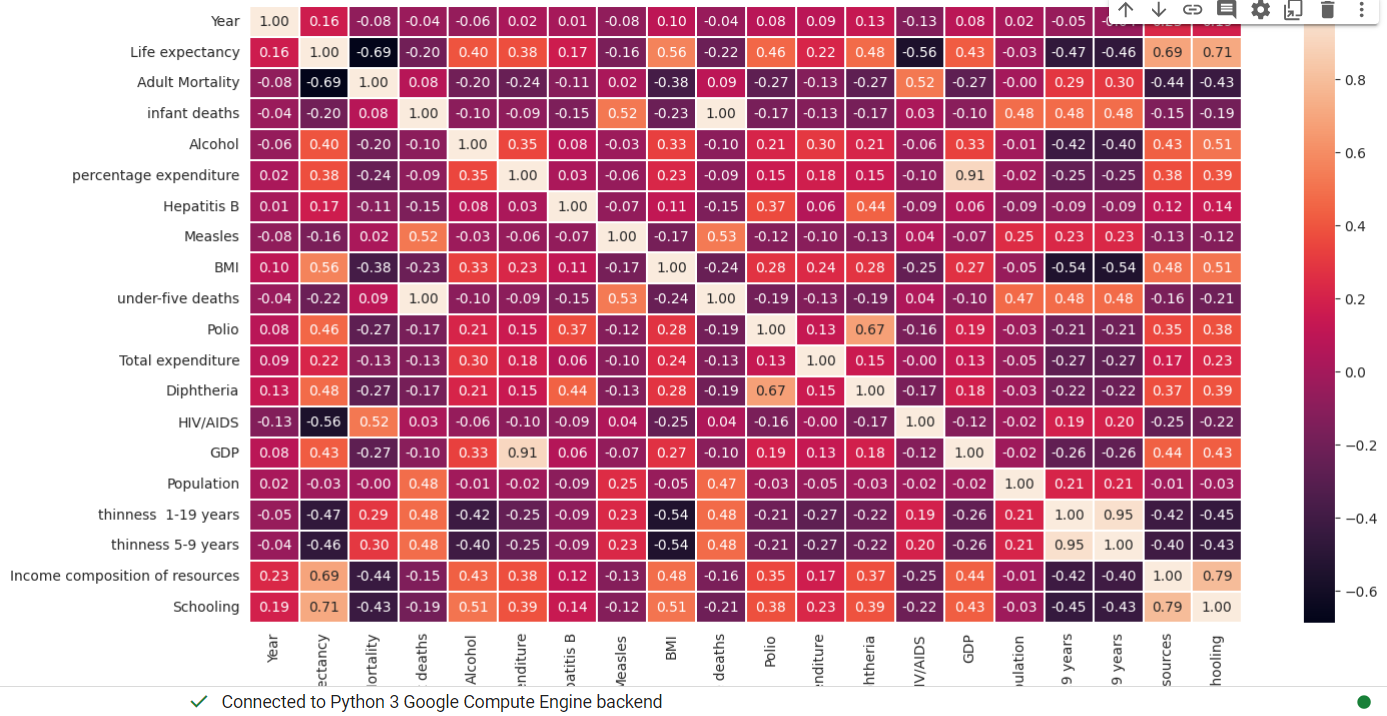


plt.figure(figsize = (16,8))

sns.heatmap(train.select\_dtypes(exclude = object).corr(), annot = True, fmt = ".2f", linewidths = 0.2)

plt.show()

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



# create object from labelencoder

encoder = LabelEncoder()

for column in ["Country", "Status"]:

    train[column] = encoder.fit\_transform(train[column])

    test[column] = encoder.fit\_transform(test[column])

در این بخش از کد، یک شیء از کلاس LabelEncoderایجاد می‌شود و سپس برای ستون‌های Country و Statusداده‌های آموزش train و آزمون testاین LabelEncoderبه کار گرفته می‌شود.

عملکرد LabelEncoderاین 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\_trainحاوی ویژگی‌ها برای آموزش، y\_trainحاوی مقدار متغیر وابسته برای آموزش، X\_testحاوی ویژگی‌ها برای آزمون، و y\_testحاوی مقدار متغیر وابسته برای آزمون است.

# Scaling train data using min max scaler

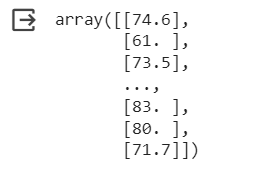
scaler = MinMaxScaler()

X\_train= scaler.fit\_transform(X\_train)

X\_test= scaler.transform(X\_test)

در این بخش از کد، داده‌های ویژگی X\_train و X\_test برای آموزش و آزمون با استفاده از MinMaxScalerمقیاس‌پذیر شده‌اند. این مقیاس‌دهی به ازای هر ستون، مقادیر را به یک بازه استاندارد (معمولاً [0, 1]) تبدیل می‌کند. این کار معمولاً برای بهبود عملکرد الگوریتم‌های یادگیری ماشین که به مقیاس داده حساس هستند، مفید است.

y\_train



نتایج داده ای پیش بینی شده و مقایسه ان ها :

tree\_model = tree.DecisionTreeRegressor(random\_state=83)

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

# Fit the regressor to the training data

tree\_model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = tree\_model.predict(X\_test)

در این بخش از کد:

- مدل رگرسیون درخت تصمیم با داده‌های آموزش X\_trainو y\_trainآموزش داده شده است، از طریق (tree\_model.fit(X\_train, y\_train

- سپس، با استفاده از مدل آموزش دیده بر روی داده‌های آزمون (`X\_test`): `y\_pred = tree\_model.predict(X\_test) ، پیش‌بینی‌ها انجام شده و در y\_predذخیره شده‌اند.

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from sklearn.metrics import r2\_score

# Evaluate the model using metrics

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

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

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

# Calculate R-squared

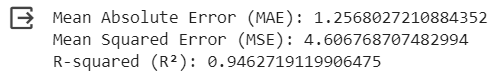
r2 = r2\_score(y\_test, y\_pred)

print(f'R-squared (R²): {r2}')

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

- mean\_absolute\_errorو mean\_squared\_errorبه ترتیب معیارهای خطای میانگین مطلق و خطای میانگین مربع را بر حسب y\_testو y\_predمحاسبه می‌کنند.

- سپس، مقادیر این معیارها چاپ می‌شوند.

- r2scoreبرای محاسبه ضریب توافق R-squaredبر حسب y\_testو y\_predاستفاده شده و نتیجه آن نیز چاپ می‌شود. R-squared نشان دهنده‌ی تطابق مدل با داده‌هاست که در بازه [0, 1] قرار دارد، که مقادیر بالاتر نشان‌دهنده‌ی عملکرد بهتر مدل است.

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")

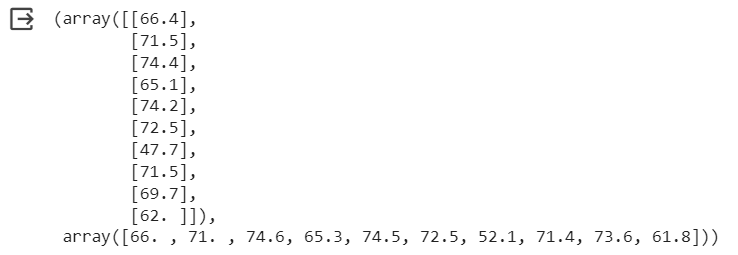
# label\_test2 , y\_hat2 = arrays\_with\_names = np.vstack(("prediction", label\_test2)), np.vstack(("label", y\_hat2))

# array = np.hstack((label\_test2 , y\_hat2))

# print(array)

در این بخش از کد، برای تست مدل روی یک زیرمجموعه از داده‌های آزمون X\_testواحد‌های تصادفی انتخاب شده‌اند. سپس بر روی این زیرمجموعه از داده‌ها، پیش‌بینی‌های مدل (`y\_hat2`) و برچسب‌های واقعی (`label\_test2`) محاسبه شده‌اند.

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



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:

1. max\_depth (حداکثر عمق درخت)

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

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

2. min\_samples\_split حداقل تعداد نمونه‌ها برای تقسیم یک گره

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

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

3. min\_samples\_leaf حداقل تعداد نمونه‌ها در یک برگ

- این پارامتر نیز به مانند min\_samples\_splitکنترل می‌کند که آیا مدل تا چه حد از جزئیات داده‌ها برازش کند یا خیر.

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

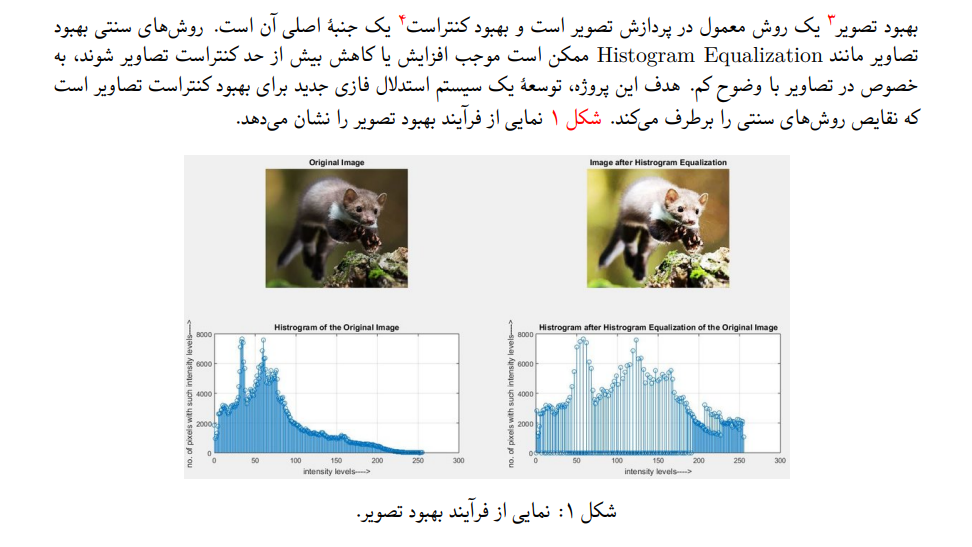
4. max\_features حداکثر تعداد ویژگی‌ها برای جستجو در هر تقسیم

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

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

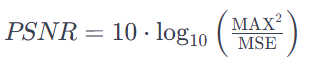
همان طور که مشاهده هم شد با تغییر پارامتر های درخت به تخمین بهتری از داده های خروجی آزمون رسیده ایم و از طرفی درصد خطا نیز کاهش یافته است.

## سوال پنجم :



توضیحات کلی در مورد سوال :

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

فرمول PSNR به صورت زیر است:

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

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

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

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



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

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