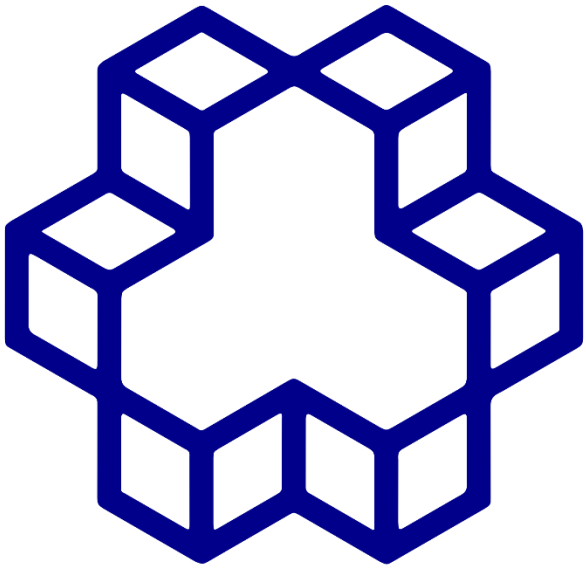


دانشگاه صنعتی خواجه نصیرالدین طوسی

مینی پروژه سوم

علی خدارحمی ۹۸۲۱۵۲۳

آدرس کدها در گیت هاب:

https://github.com/AliKhodarahmy/machinelearning2023/tree/main/miniproject/3

سوال ۱:

محاسبات لازم را برای پارامتر های هر حالت انجام میدهیم:

کران مرتبه اول: $\|f-g\|_\infty < \|\frac{\partial g}{\partial x_1}\| h_1 + \|\frac{\partial g}{\partial x_2}\| h_2$

$g = \frac{1}{2+x_1+x_2} \rightarrow \frac{\partial g}{x_i} = \frac{-1}{(2+x_1+x_2)^2} \rightarrow \|\frac{\partial g}{\partial x_i}\|_\infty = 1$

$x_1 = -1$, $x_2 = -1$

$\varepsilon = 0.1 \Rightarrow h_1 + h_2 \leq 0.1 \rightarrow h_1 = h_2 = 0.05$

$\Rightarrow \frac{1-(-1)}{0.05} = 40 \quad 40^2 + 1$ → باقی‌مانده → Rules = 1681

$N = 40, h = 0.05$

کران مرتبه دوم: $\|f-g\|_\infty \leq \frac{1}{8}\left[\|\frac{\partial g^2}{\partial x_1^2}\| h_1^2 + \|\frac{\partial g^2}{\partial x_2^2}\| h_2^2\right]$

$\frac{\partial g^2}{\partial x_i^2} = \frac{\partial (1/(2+x_1+x_2))^2}{\partial x_i^2} = \frac{+2 \times (2+x_1+x_2)}{(2+x_1+x_2)^4} \rightarrow \|\frac{\partial g^2}{\partial x_1^2}\| = 2$

$\varepsilon = 0.1 \Rightarrow \frac{1}{8}\left[2 \times h_1^2 + 2 \times h_2^2\right] \leq 0.1 \xrightarrow{h_1 = h_2} h^2 \leq 0.2$

$h = 0.447 \rightarrow h = 0.4 = h_1 = h_2$

$\frac{2}{0.4} = 5 \rightarrow$ Rules $= 5 \times 5 = 25$

$N = 5, h = 0.4$

از کد متلب زیر استفاده کرده و پارامترهای محاسبه شده و فرمول را جایگذاری میکنیم:

کران مرتبه اول:

```
clc
clear all
close all
tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.05;
N = 40;

g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

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

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
        e_i1(i1,1) = -1 + h*(i1-1);
        e_i2(i2,1) = -1 + h*(i2-1);

        if i1==1
            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*(i1-2), -1+h*(i1-1), -1+h*(i1)]);
        end

        if i2==1
            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*(i2-2), -1+h*(i2-1), -1+h*(i2)]);
        end

        g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
        num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
        den = den + mu_A_x1.*mu_A_x2;
        k = k + 1;
    end
end

f_x = reshape(num./den, size(x1)); % Reshape f_x to match the size of x1 and x2
```

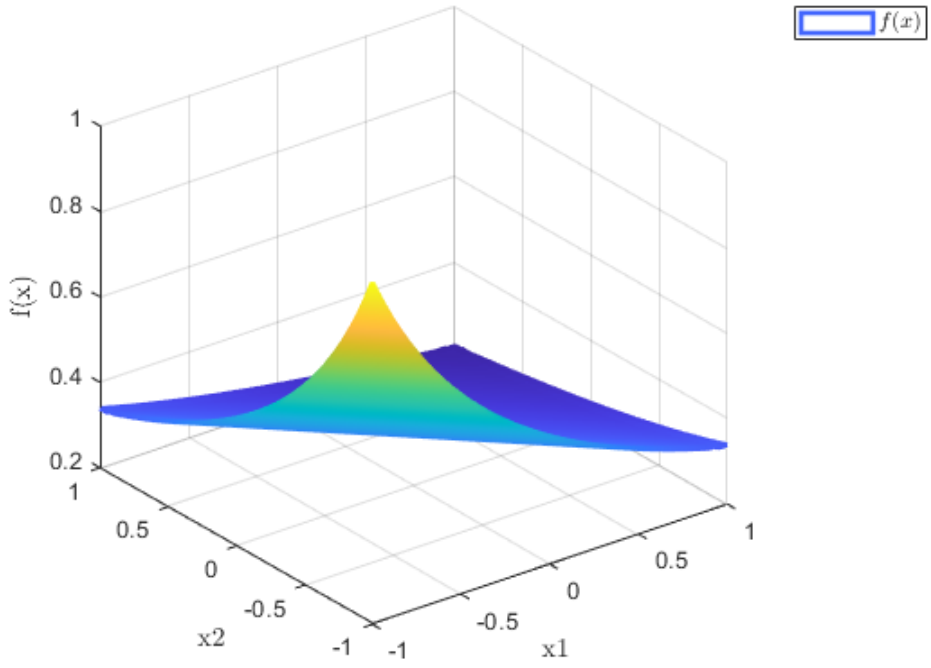
```
g_x = 1./(3+x1+x2);

figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on

figure2 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

toc
```

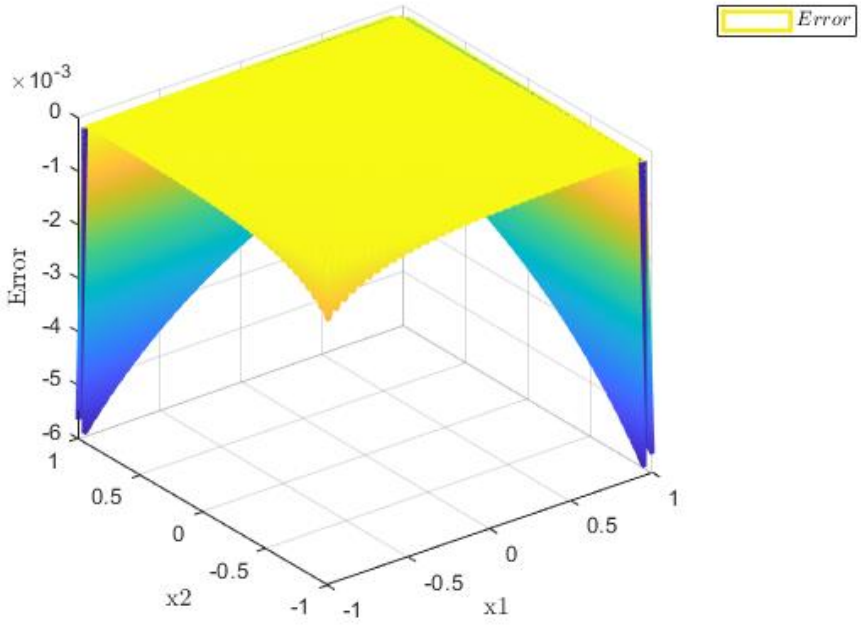
کران مرتبه دوم:

```
clc
clear all
close all
tic

alpha = -1;
beta = 1;
x1 = alpha:0.001:beta;
x2 = alpha:0.001:beta;
h = 0.4;
N = 5;

g_bar = zeros(N*N,1);
e_i1 = zeros(N,1);
e_i2 = zeros(N,1);

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

num = 0;
den = 0;
k = 0;

for i1=1:N
    for i2=1:N
        e_i1(i1,1) = -1 + h*(i1-1);
        e_i2(i2,1) = -1 + h*(i2-1);

        if i1==1
            mu_A_x1 = trimf(x1(:), [-1,-1,-1+h]);
        elseif i1==N
```

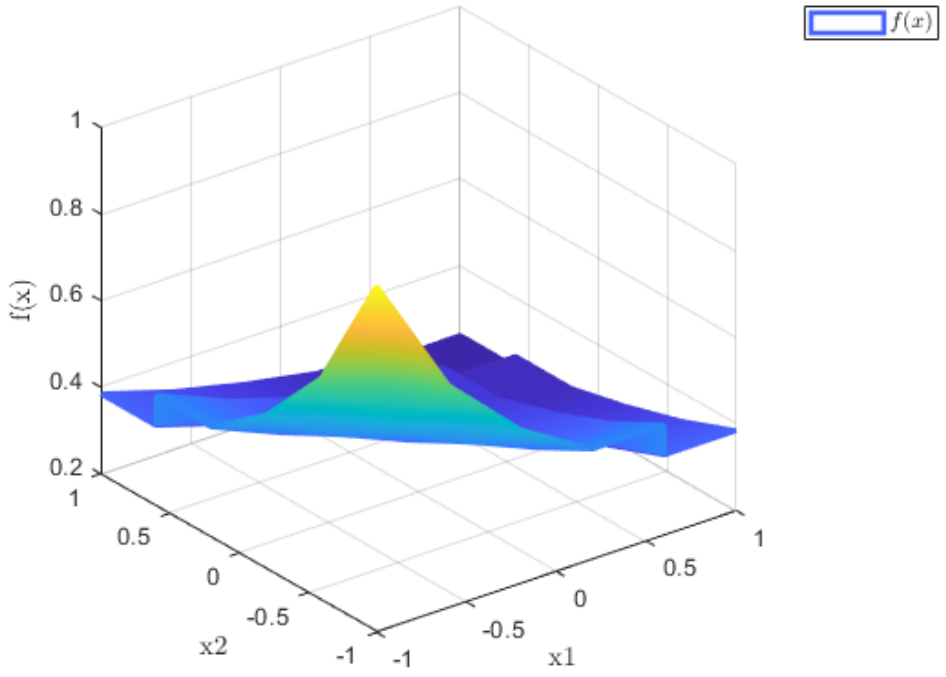
```
            mu_A_x1 = trimf(x1(:), [1-h, 1, 1]);
        else
            mu_A_x1 = trimf(x1(:), [-1+h*(i1-2), -1+h*(i1-1), -1+h*(i1)]);
        end

        if i2==1
            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*(i2-2), -1+h*(i2-1), -1+h*(i2)]);
        end

        g_bar(k+1,1) = 1./(3+e_i1(i1,1)+e_i2(i2,1));
        num = num + g_bar(k+1,1).*mu_A_x1.*mu_A_x2;
        den = den + mu_A_x1.*mu_A_x2;
        k = k + 1;
    end
end

f_x = reshape(num./den, size(x1)); % Reshape f_x to match the size of x1 and x2

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

figure1 = figure('Color',[1 1 1]);
mesh(x1,x2,f_x,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('f(x)','Interpreter','latex');
legend('$f(x)$','Interpreter','latex')
grid on

figure2 = figure('Color',[1 1 1]);
E = g_x - f_x;
mesh(x1,x2,E,'Linewidth',2);
xlabel('x1','Interpreter','latex');
ylabel('x2','Interpreter','latex');
zlabel('Error','Interpreter','latex');
legend('$Error$','Interpreter','latex')
grid on

toc
```

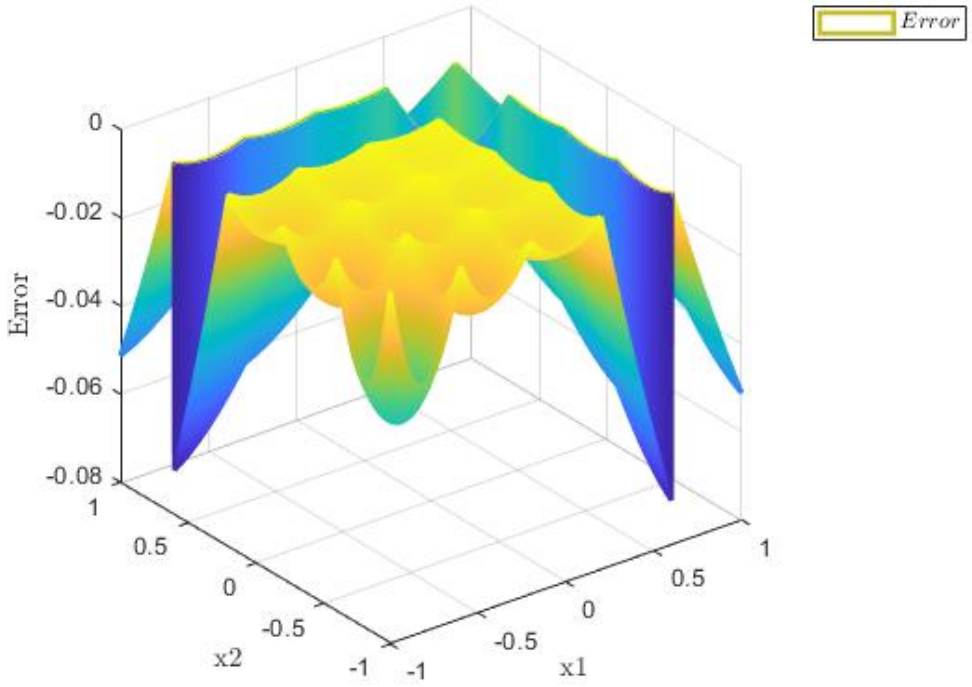
خروجی کران مرتبه اول:





خروجی کران مرتبه دوم:





تحلیل:

با توجه به شـــکل نمودار هر تابع تقریب زده شـــده و نیز شـــکل خطای هر یک میتوان گفت که کران مرتبه اول تقریب بهتری از تابع هدف انجام داده است که دلیل این امر تعداد قوانین بیشـتر و انعطاف پذیری بالاتر سـیسـتم

اســت. البته با افزایش قوانین زمان اجرای کد نیز افزایش میابد که بســته به نوع اســتفاده (خطا اهمیت بیشــتری دارد یا زمان اجرا) میتوان هر یک را انتخاب و استفاده کرد.

سوال ۲:

کد زیر را برای پیش گویی سری زمانی Mackey-Glass مینویسیم و در ادامه به توضیح و خروجی کد میپردازیم:

Data Generation

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

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

figure1 = figure ('Color', [1 1 1]); plot (t,x (201:800), 'Linewidth', 2)
```

این بخش یک سری زمانی آشفته Mackey-Glass (x) ایجاد می کند و یک مجموعه داده (مجموعه داده) را با استفاده از مقادیر آن ایجاد می کند.

Rule Generation

```
[Number_training, ~]=size (dataset);
Rul=zeros (Number_training/2,6);
Rules_total=zeros (Number_training/2, 6);

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
```

این قسمت مجموعه های مختلفی از توابع عضویت را بر اساس مقدار s مقداردهی اولیه و تعریف می کند. از توابع عضویت ذوزنقه ای ('trapmf') و مثلثی ('trimf') استفاده می کند. پارامترهای این توابع با توجه به شرایط مشخص شده تعریف می شوند.

برای s=1 ، 7تابع عضویت با اشکال ذوزنقه ای برای اولین و آخرین توابع و اشکال مثلثی برای بقیه ایجاد می کند.

برای s=2 ، ۱۵ تابع عضویت با اشکال ذوزنقه ای برای اولین و آخرین توابع و اشکال مثلثی برای بقیه ایجاد می کند.

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
```

این بخش درجات عضویت را برای هر قانون بر اساس توابع عضویت تعریف شده محاسبه می کند. حداکثر درجه عضویت را برای هر متغیر ورودی محاسبه می کند و آن را به قانون مربوطه اختصاص می دهد. حاصل ضرب این درجات در Rules(t, 6) و dataset(t, 6) ذخیره می شود.

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
```

این قسمت از کد قوانین تکراری یا قوانین با اهمیت کمتر را حذف می کند. قوانین را بر اساس شرایط ورودی آنها مقایسه می کند و قانون را با ارزش محصول بالاتر نگه می دارد.

Fuzzy Inference System (FIS) Creation

```
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);
```

در این قسمت یک سیستم استنتاج فازی (FIS) از نوع ممدانی با استفاده از تابع newfis ایجاد می شود . FIS "کنترل کننده پیش بینی" نامیده می شود و از روش های استنتاج محصول ("prod") و حداکثر تجمع ("max")استفاده می کند. خروجی با استفاده از روش سانتروئید غیرفازی می شود.

```
%% 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');
```

این بخش متغیرهای ورودی و خروجی را به FIS اضافه می کند. متغیرهای ورودی، با نام های 'x1' تا 'x4' دارای محدوده [۰.۱، ۱.۷] و متغیر خروجی 'x5' نیز دارای محدوده [۰.۱، ۱.۷] است.

```
%% 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
```

در اینجا، توابع عضویت به متغیرهای ورودی 'x1') تا ('x4' و متغیر خروجی ('x5') در FIS اضافه می شوند. پارامترهای توابع عضویت از آرایه سلولی Membership_functions گرفته شده است که قبلاً بر اساس مقدار Sتعریف شده بود.

```
%% 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);
```

این بخش قوانین فازی را به FIS اضافه می کند. این سطرها را در Rules_total با قوانین غیر صفر شناسایی می کند، یک ماتریس جدید fis_Rules با آن ردیف ها ایجاد می کند و سپس این قوانین را با استفاده از تابع addruleبه FIS اضافه می کند.

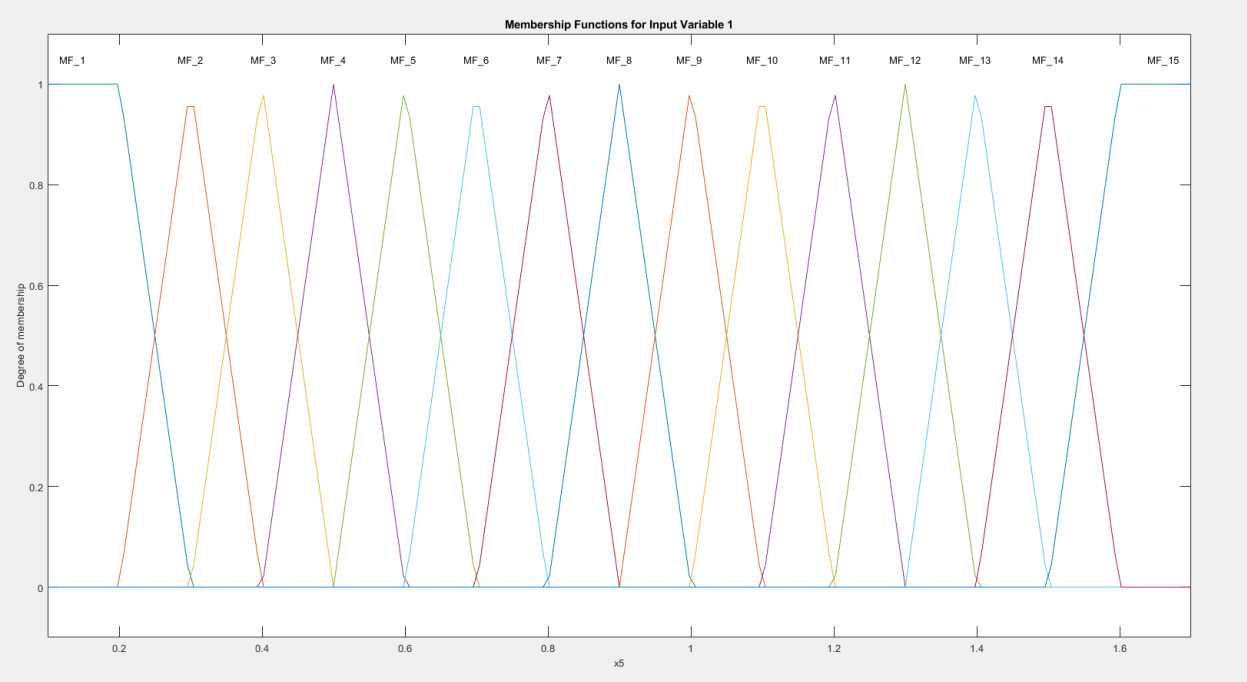
```
%% Prediction of 300 points of chosen dataset
jadval_prediction=zeros(300,2);
f=1;
for i=301:600
    input=dataset(i, 2:6);
    output1=dataset(i, 6);
    x5=evalfis([input(1, 1); input(1, 2); input(1,3); input(1,4)], fis);
    jadval_prediction(f, :)= [f, x5];
    f=f+1;
end
```

این بخش ۳۰۰ نقطه را با استفاده از FIS آموزش دیده پیش بینی می کند. از طریق مجموعه داده از ردیف ۳۰۱ تا ۶۰۰ تکرار می شود، خروجی (x5) را با استفاده از FIS محاسبه می کند و نتایج را در ماتریس jadval_predictionذخیره می کند.

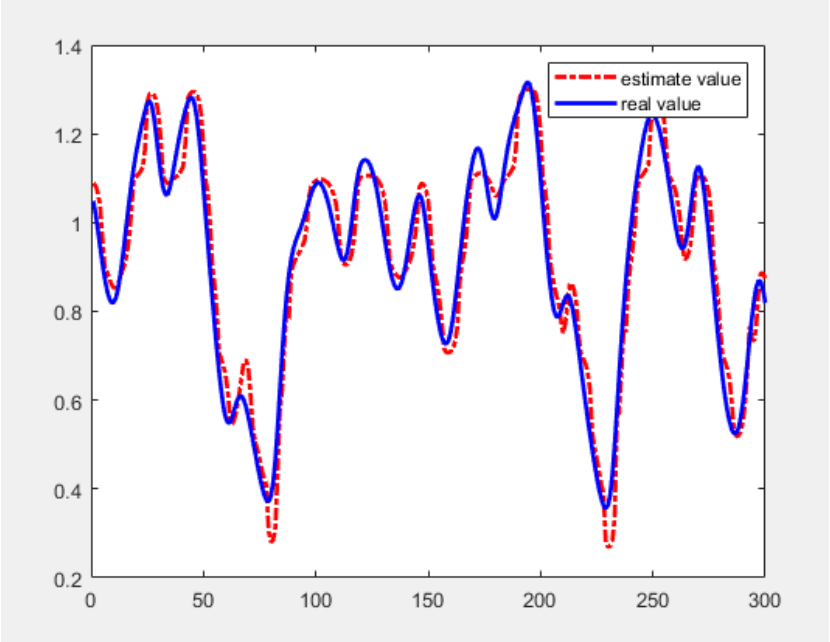
```
figure;
plot(jadval_prediction(:,1),jadval_prediction(:,2), 'r-.', 'Linewidth', 2);
hold on;
plot(jadval_prediction(:,1),dataset(301: 600, 6), 'b', 'Linewidth', 2);
legend('estimate value', 'real value')
end
% Assuming 'fis' is your fuzzy inference system
inputVariableIndex = 1;  % Change this to the index of the input variable you're
interested in

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

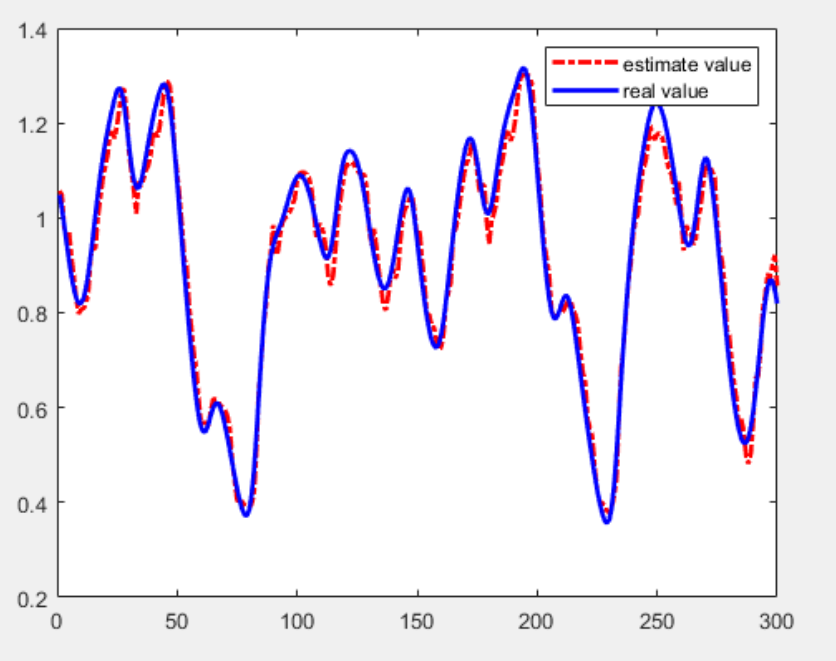
در نهایت، این بخش یک شکل ایجاد می کند و مقادیر برآورد شده (jadval_prediction(:,2)) را در برابر مقادیر واقعی (dataset(301: 600, 6)) از ردیف ۳۰۱ تا ۶۰۰ ترسیم می کند. این شامل افسانه ای برای تمایز بین مقادیر تخمینی و واقعی است. به نظر می رسد فیلمنامه برای مدیریت دو سناریو بر اساس مقدار S طراحی شده است.

توابع عضویت:



نمایش:




estimate value
real value
1.4
1.2
1
0.8
0.6
0.4
0.2
0
50
100
150
200
250
300

سوال ۳:

کد زیر که شناسایی معادله درجه دوم و تقریب آن به همراه گرادیان نزولی است را نوشته و هر بخش را توضیح میدهیم:

```
clc
clear all
close all
%% GRADIEN DECENT TRAINING ALGORITHEM.
disp(' GRADIEN DECENT TRAINING ALGORITHEM.');
disp('  This programm develops as a course project for Fuzzy systems course')
disp('   by Habibollah Naeimi.')
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-
.-*-.-*-----------------------------------------------------')
disp(' ');
%% 1st Part: Parameter Setting.
%%  Parameters Initiating.
disp(' Parameters Initiating...');

M = 30;
Alpha = 0.3;
Q = 100;
epsilon = 0;
InpuNum = 1;
y_Bar = zeros(M,1);
x_Bar = zeros(M,InpuNum);
Sigma = zeros(M,InpuNum);
```

این قسمت پارامترهای مورد نیاز برای الگوریتم آموزش نزول گرادیان را تنظیم می کند. تعداد قوانین(M) ، نسبت آموزش (آلفا)، تعداد تکرارها(Q) ، خطای نهایی برای تمرین تکرار نقطه(epsilon) ، و تعداد متغیرهای ورودی (InpuNum)را تعریف می‌کند. همچنین ماتریس های y_Bar ، x_Bar و Sigma را با صفر مقداردهی می کند.

```
disp(' Part 1: DONE!');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-
.-*-.-*-----------------------------------------------------')
disp(' ');

%% 2rd Part: Sampling.
%%  Calculating Samples.
Data_Pairs_Num = 300;

SAMPLES_Num = 500;
a1 = 0.3;
a2 = 0.6;

y_aprx = zeros(1,SAMPLES_Num);
SAMPLES = zeros(SAMPLES_Num,InpuNum+1);
iNSamples = 0.2:0.01:0.21;
for k=2:SAMPLES_Num+33+InpuNum
```

```
    iNSamples(k + 1) = a1*iNSamples(k) + a2*iNSamples(k - 1) + 0.6*sin(pi*k) +
0.3*sin(3*pi*k) + 0.1*sin(5*pi*k)
end

iNSamples = iNSamples(2:end);

for i=1:SAMPLES_Num
    SAMPLES(i,:) = iNSamples(i:i+InpuNum);
end

Pairs = SAMPLES(1:Data_Pairs_Num,:);
```

این بخش یک داده سری زمانی (iNSamples) تولید می کند و با گرفتن بخش های متوالی از سری زمانی جفت های داده (SAMPLES) ایجاد می کند. سپس یک زیر مجموعه (جفت) را برای پردازش بیشتر انتخاب می کند.

```
disp(' Complete Sampling.');
%%  Primary Parameter Fixing Using Online initial Parameter Choosing.

x_Bar = Pairs(1:M,1:InpuNum);
y_Bar = Pairs(1:M,end);
Sigma = repmat(((max(x_Bar)-min(x_Bar))/M),M,1);
```

در اینجا، پارامترهای اولیهx_Bar ، y_Barو Sigma بر اساس جفت داده های انتخاب شده (Pairs) تنظیم می شوند.

```
disp(' Initial Parameters are Reasdy!');
disp(' ');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-
.-*-.-*-------------------------------------------------------')
disp(' ');
%% 3rd Part: Parameters Updating.
%%  Updating.

z = zeros(M,1);
iN_z = zeros(1,InpuNum);

 for p=1:size(Pairs,1)

     for l=1:M
         for i=1:InpuNum
             iN_z(i) = exp(-(((Pairs(p,i)-x_Bar(l,i))/Sigma(l,i))^2))
         end
         z(l) = prod(iN_z);
     end

     b = sum(z);
     a = sum(y_Bar.*z);
     f = a/b;

            for q=1:Q
```

```
                for l=1:M
                    y_Bar(l) = y_Bar(l)-Alpha*(f-Pairs(p,end))/b*z(l);
                    for i=1:InpuNum
                        x_Bar(l,i) = x_Bar(l,i)-Alpha*(f-Pairs(p,end))/b*(y_Bar(l)-
f)*z(l)*(2*(Pairs(p,i)-x_Bar(l,i))/(Sigma(l,i)^2));
                        Sigma(l,i) = Sigma(l,i)-Alpha*(f-Pairs(p,end))/b*(y_Bar(l)-
f)*z(l)*(2*((Pairs(p,i)-x_Bar(l,i))^2)/(Sigma(l,i)^3));
                    end
                end

                if (f-Pairs(p,end))<epsilon
                    break;
                end
            end
 end

disp(' Parameters Updated.');
disp('*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-.-*-
.-*-.-*-------------------------------------------------------')
disp(' ');
```

این بخش پارامترها را با استفاده از رویکرد نزولی گرادیان به روز می کند. از طریق جفت داده ها تکرار می شود، خروجی f را محاسبه می کند و پارامترهایx_Bar ، y_Barو Sigma را به روز می کند تا خطا را به حداقل برساند.

```
%% 4th Part: Results.
%%  Results.

f = zeros(1,SAMPLES_Num);
f(1:2) = SAMPLES(1:2,end);

for k=3:SAMPLES_Num

    for l=1:M
        for i=1:InpuNum
            iN_z(i) = exp(-(((SAMPLES(k,i)-x_Bar(l,i))/Sigma(l,i))^2));
        end
        z(l) = prod(iN_z);
    end

    b = sum(z);
    a = sum(y_Bar.*z);
    f(k) = a/b;
    y_aprx(k) = f(k);

end
```

این قسمت خروجی پیش بینی شده f را بر اساس پارامترهای به روز شده محاسبه می کند.

```
%%  Plotting.

figure;
plot(SAMPLES(:,end));
```

```
hold on
plot(y_aprx,'r');
legend('Real Value','Predicted Value');
```

این بخش نموداری را ایجاد می کند که مقادیر واقعی (SAMPLES(:,end)) را با مقادیر پیش بینی شده (y_aprx)مقایسه می کند.

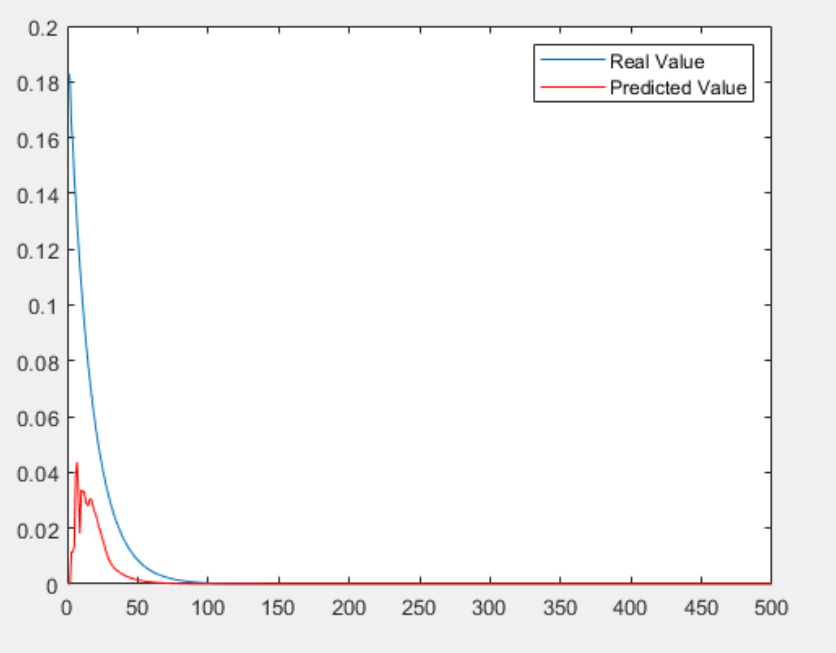
```
%%  Errors.
Error = SAMPLES(:,end)-y_aprx';
disp('Mean Square Error is:');
MSE = mse(Error)
disp('Mean Absolute Error is:');
MAE = mae(Error)
```

در نهایت، این قسمت میانگین مربع خطا (MSE) و میانگین خطای مطلق (MAE) را بین مقادیر واقعی و پیش بینی شده محاسبه و چاپ می کند.

خروجی خطاها :

```
MSE = 0.00040599
MAE =  0.0047
```

رسم:



سوال ۴:

کد زیر که شناسایی معادله درجه دوم و تقریب آن به همراه گرادیان نزولی است را نوشته و هر بخش را توضیح میدهیم:

بخش ۱: بیماری کرونا

```
!pip install --upgrade --no-cache-dir gdown
!gdown 140-3CYiGdnKfgDXQMiXDNH1eMtn3PWe1
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

در این قسمت فراخوانی دیتاست از درایو و آماده سازی کتابخانه های مورد نیاز را انجام میدهیم.

```
data_trainset, data_testset = train_test_split(data, test_size=0.2,
random_state=7)
```

تقسیم بندی دیتا به ۸۰ درصد آموزش و ۲۰ درصد داده ی تست

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

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

0.9456603046006401

محاسبه آنتروپی را به شکل یک تابع نوشته و آنتروپی گره اول را حساب میکنیم.

```
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

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

[0.2741310366085653, 0.02368671823006807, 0.61751117056093]

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

2

تابع IG را مینویسیم و برای هر یک از ویژگی ها حساب میکنیم و در نهایت ویژگی شماره ۲ یعنی Breathing issues انتخاب میشود.

```
class Node:

    def __init__(self, feature=None, label=None):
        self.feature = feature
        self.label = label
        self.children = {}

    def __repr__(self):#  این تابع کمک میکنه که خیلی تر و تمیز خروجی بده و
بهمون یه سری عدد (که آدرسن) نده
        if self.feature is not None:
            return f'DecisionNode(feature="{self.feature}",
children={self.children})'
            #وقتی دیسیژن نود داشته باشیم حتما چیلدرن نود هم داریم
        else:
            return f'LeafNode(label="{self.label}")'

Node(label='Cough')
```

کلاس گره ها را تشکیل میدهیم. تابع اول آن برای معرفی متغیر های داخل کلاس و رجیستر شدن است و تابع دوم نیز کمک میکند تا به جای خروجی گرفتن یک سری عدد که نامفهوم هستند، نمایش بهتری داشته باشیم و در نهایت گره ها تعریف شده اند. در بازگرداندن این تابع، اگر ویژگی ای به کلاس نداده باشیم یعنی گره برگ (کرونا دارد یا ندارد) است و در غیر این صورت گره تصمیم گیری می باشد.

```
def make_tree(data, target):
    # leaf node?
    if len(data[target].unique()) == 1:# برای زمانی که به برگ رسیدیم و
دیگه همه یک وضعیت دارند
```

```
        return Node(label=data[target].iloc[0])#  همه وضعیت ها مشابه همند
و یکی شونو برداریم و به کلاس نود بدیم کافیه

    features = data.drop(target, axis=1).columns#  ازش خود تارگت رو حذف
کنیم
    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_gain_idx = np.argmax(gains)# میخواهیم اندیشو بهمون بده تا بریم در
دیتا با اون کار کنیم
    best_feature = features[max_gain_idx]

    # make a node
    node = Node(feature=best_feature)

    # loop over the best feature
    for value in data[best_feature].unique():
        subset = data[data[best_feature] == value].drop(best_feature,
axis=1)#در ساب ست دیگه به خود اون ستون نیازی نداریم
        display(subset)

        node.children[value] = make_tree(subset, target)#  فرزندان را ثبت
کنیم همونی که داخل کلاس نود خالی گذاشته بودیمش

    return node
```

این تابع دو آرگومان می گیرد: داده، که فرض می شود یک DataFrame پاندا حاوی مجموعه داده است، و هدف، که نام ستونی است که متغیر هدف را نشان می دهد (متغیری که باید پیش بینی شود) .اولین دستور if بررسی می کند که آیا فقط یک مقدار منحصر به فرد در متغیر هدف وجود دارد یا خیر. اگر درست باشد، یک گره برگ ایجاد می کند که برچسب آن مقدار منحصر به فرد است .دستور دوم if بررسی می کند که آیا هیچ ویژگی باقی نمانده است یا اینکه مجموعه داده خالی است. اگر درست باشد، یک گره برگ ایجاد می کند که برچسب آن حالت (متداول ترین) مقدار متغیر هدف است .سپس با استفاده از تابعی به نام information_gain (که در قطعه کد ارائه نشده است) بهره اطلاعات را برای هر ویژگی محاسبه می کند. افزایش اطلاعات معیاری است که در الگوریتم‌های درخت تصمیم برای تصمیم‌گیری در مورد کدام ویژگی به کار می‌رود .این یک جستجوی حریصانه برای یافتن ویژگی با حداکثر کسب اطلاعات انجام می دهد و یک گره برای آن ویژگی ایجاد می کند .سپس به

صورت بازگشتی خود را برای هر مقدار منحصر به فرد بهترین ویژگی فراخوانی می کند و گره های فرزند را برای هر مقدار ایجاد می کند.

در نهایت درخت ساخته شده را برمی گرداند.

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

	Fever	Cough	Infected
5	No	Yes	No
0	No	No	No
13	Yes	Yes	No
10	No	Yes	No
9	Yes	Yes	Yes

	Cough	Infected
5	Yes	No
0	No	No
10	Yes	No

	Cough	Infected
13	Yes	No
9	Yes	Yes

	Infected
13	No
9	Yes

	Fever	Cough	Infected
1	Yes	Yes	Yes
8	No	Yes	Yes
7	Yes	No	Yes

	Fever	Cough	Infected
3	Yes	No	Yes
6	Yes	No	Yes
4	Yes	Yes	Yes

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

درخت بالا رفته رفته داده ها را تقسیم بنده کرده است و در هر گره تعدادی باقی مانده و برای آنها تصمیم گیری شده است. برای نمایش بهتر جلوتر از نمایش گرافیکی آن استفاده میکنیم.

فرایند ارزیابی

داده های تست به شرح زیر اند:

	Fever	Cough	Breathing issues	Infected
12	No	Yes	Yes	No
11	No	Yes	Yes	Yes
2	Yes	Yes	No	No

```
def predict(node, sample):
    if node.feature is None:
        return node.label

    feature_value = sample[node.feature]

    if feature_value in node.children:
        return predict(node.children[feature_value], sample)
    else:
        return node.label

MyPredict = [predict(tree, sample) for _, sample in
data_testset.iterrows()]
MyPredict
```

کد بالا سه داده تست را به درخت تصمیم داده و پاسخ نهایی آنها را برمیگرداند:

['Yes', 'Yes', 'No']

مشاهده میکنیم که داده اول را اشتباه طبقه بندی کرده ولی دو داده دیگر به درستی انجام گرفته اند.

```
from sklearn.metrics import accuracy_score

# Calculate accuracy
accuracy = accuracy_score(infected_list, MyPredict)

# Print the accuracy
print(f"Accuracy: {accuracy * 100:.2f}%")
```

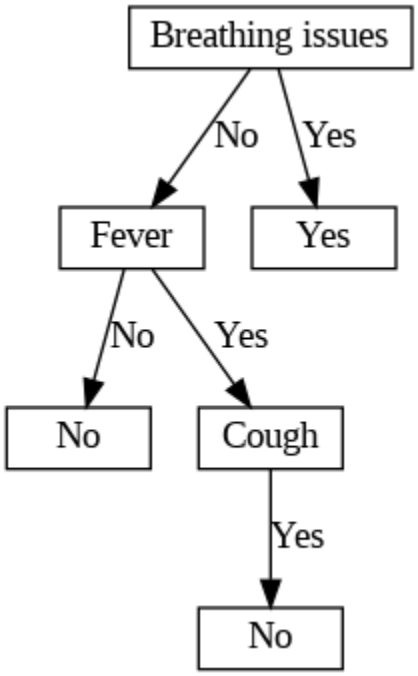
```
Accuracy: 66.67%
```

دقت برابر با ۶۶ درصد حاصل میشود.

```
from graphviz import Digraph, nohtml
```

```
g = Digraph('g', filename='decision-tree.gv', node_attr={'shape':
'record', 'height': '.1'})

def plot_tree(tree, g):
    root_node = tree.feature
    if root_node is None:
        return g
    g.node(root_node, nohtml(root_node))
    child_nodes = tree.children.keys()
    for i, child in enumerate(child_nodes):
        node = tree.children[child]
        name = node.feature if node.feature is not None else
child+node.label
        label = node.feature if node.feature is not None else node.label
        g.node(name, nohtml(label))
        g.edge(root_node, name, label=child)
        plot_tree(node, g)
    return g

g = plot_tree(tree, g)
g.render('decision_tree', format='png', view=True)
```

کد بالا برای رسم گرافیکی درخت تصمیم است که خروجی با فرمت png به صورت زیر به دست می آید:



بخش 2: دیتاست Drugs انتخاب شده است.

```
from sklearn import tree
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder

data = pd.read_csv('/content/drug200.csv')
```

دیتاست دراگ را در کولب وارد میکنیم و کتابخانه های لازم را فراخوانی میکنیم.

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

این دیتاست دارای ۶ ویژگی است که هر کدام دارای ۲۰۰ سمپل هستند و لازم است تعدادی از آنها از جمله ویژگی دراگ به عدد تبدیل شوند. پس کد زیر را نوشته و اجرا میکنیم:

```
l1 = LabelEncoder()
for i in data.columns:
    if data[i].dtype == 'object':
        data[i] = l1.fit_transform(data[i])
```

سمپل های فرمت object به عدد تبدیل شدند.

```
data.describe()
```

حال به کمک کد بالا یک سری اطلاعات اولیه در مورد ویژگی ها از جمله بیشترین مقدار و کمترین مقدار در هر یک را نمایش میدهیم.

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	44.315000	0.520000	0.910000	0.485000	16.084485	2.870000
std	16.544315	0.500854	0.821752	0.501029	7.223956	1.372047

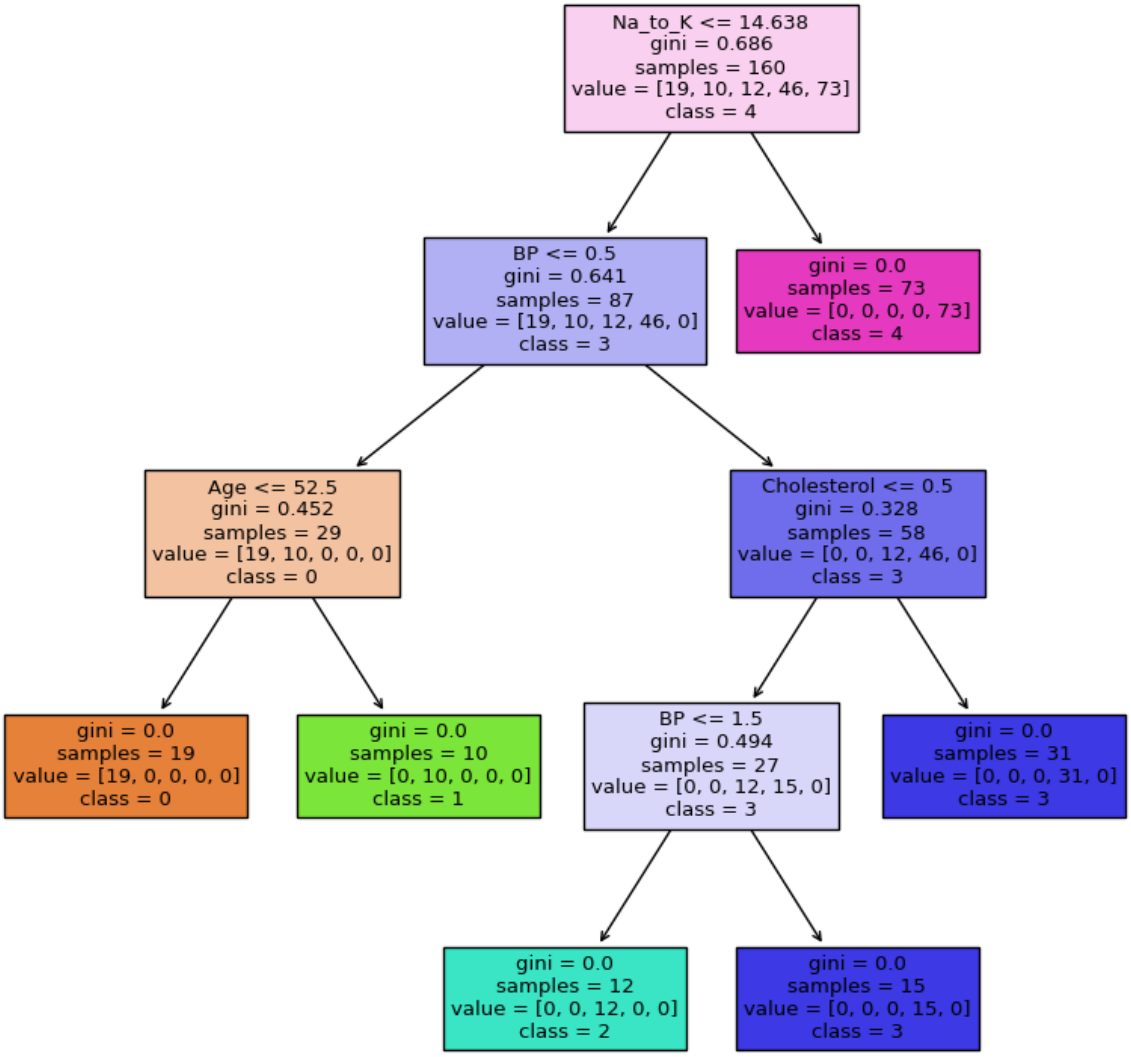
	Age	Sex	BP	Cholesterol	Na_to_K	Drug
min	15.000000	0.000000	0.000000	0.000000	6.269000	0.000000
25%	31.000000	0.000000	0.000000	0.000000	10.445500	2.000000
50%	45.000000	1.000000	1.000000	0.000000	13.936500	3.000000
75%	58.000000	1.000000	2.000000	1.000000	19.380000	4.000000
max	74.000000	1.000000	2.000000	1.000000	38.247000	4.000000

برای مثال در مورد ویژگی age : بیشترین مقدار برابر با ۷۴ سال و کمترین ۱۵ ساله است و میانگین سن افراد ۴۴ سال می باشد.

درخت تصمیم

```
import matplotlib.pyplot as plt
X = data.drop('Drug', axis=1)
y = data['Drug']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)
clf = tree.DecisionTreeClassifier(criterion ='gini' ,max_depth=4,
random_state=42)
clf.fit(X_train, y_train)

# Convert class names to strings
class_names = list(map(str, clf.classes_))
plt.figure(figsize=(10, 10))
tree.plot_tree(clf, filled=True, feature_names=X.columns,
class_names=class_names)
plt.show()
```

درخت را به با عمق ۴ تشکیل میدهیم. علت این عمق این است که در عمق ۳ دقت کمتری دارد و از ۴ به بالا دقت افزایش نمیابد. همچنین نیازی به هرس کردن ندارد چرا که هم دقت خوبی (کدهای بعدی = ۹۵ درصد دقت) به دست آمده است هم اینکه تاثیری (تا جایی که من تلاش کردم) روی دقت نداشتند. همچنین معیار gini دقت بالاتری نسبت به معیار entropy (حدودا ۸۸ درصد) به ما میدهد. همچنین کد list(map(str مروبوط به یک مشکل هنگام ساخت درخت بود که لازم بود کلاس های درخت از عدد به رشته تبدیل شوند. در نهایت خروجی گرافیکی زیر را خواهیم داشت:



توضیح کلی درمورد درخت ساخته شده: اولین ویژگی ای که دارای IG بالاتری بوده و در گره صفر قرار گرفته است ویژگی سهمیه سدیم به پتاسیم است که تعداد ۷۳ سمپل در همان مرحله در گره برگ قرار گرفته اند. سپس در سمت چپ درخت و در گره شماره یک، سطح فشار خون دارای بالاترین IG برای سمپل های باقی مانده است و همین طور تا آخر ادامه یافته و تصمیم گیری شده. همان طور که میبینیم رفته رفته ناخالصی کمتر شده و در نهایت هر ۵ کلاس به درستی جدا شده اند و gini به صفر رسیده است.

محاسبه دقت:

```
from sklearn.metrics import accuracy_score
y_hat = clf.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_hat)
# Print the accuracy
print(f"Accuracy: {accuracy * 100:.2f}%")
```

۴۰ داده تست را به درخت ایجاد شده میدهیم و نتیجه ۹۵ درصد دقت شده است.

مشاهده مسیر ۲ داده رندوم

داده اول:

```
i = 3
print (X_test.iloc[[i]])
decision_path = clf.decision_path(X_test.iloc[[i]])
decision_path.toarray()
```

خروجی:

```
output
    Age  Sex  BP  Cholesterol  Na_to_K
11   34    0   0            1   19.199
array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])
```

داده شماره ۱۱ از مجموعه تست را به درخت تصمیم داده ایم و از گره های صفر و ده عبور کرده است. این داده مربوط به کلاس ۴ بوده است.

داده دوم:

```
i = 15
print (X_test.iloc[[i]])
decision_path = clf.decision_path(X_test.iloc[[i]])
decision_path.toarray()
```

خروجی:

```
    Age  Sex  BP  Cholesterol  Na_to_K
15    5   49   1            1        0   10.537
array([[1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0]])
```

داده ۱۵ام از مجموعه داده تست، ابتدا از گره صفر عبور کرده و مقدار سهمیه سدیم به پتاسیم کمتر از ۱۴.۶۳۷ بوده است سپس به گره یک رفته و سطح فشار خون آن از ۰.۵ بیشتر بوده و به گره تصمیم گیری شماره ۵ رفته است و به ترتیب کلسترل و سپس سطح فشار خون آن هر دو از ۰.۵ کمتر بوده و در نهایت در کلاس ۲ قرار گرفته است.

بخش ۳: دیتاست میزان امید به زندگی

داده را در کولب لود کرده و اطلاعاتی از مجموعه داده مشاهده میکنیم:

```
from sklearn import tree
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder

data = pd.read_csv('/content/Life Expectancy Data.csv')
data.info()
```

خروجی:

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

این دیتاست دارای ۲۲ ویژگی است که Life expectancy را به عنوان خروجی در ادامه کد قرار میدهیم. داده های نامتعادل هستند و لازم است که تعداد سمپل ها برابر شوند پس کمترین مقدار را مرجع قرار داده و دیگر سطرهای ویژگی های دیگر را حذف میکنیم. همین طور لازم است که دو ویژگی به عدد تبدیل شوند. در ادامه این دو عمل را انجام میدهیم.

متعادل کردن داده ها:

```
data = data.dropna(how='any')
```

تبدیل داده های object به عددی:

```
l1 = LabelEncoder()
for i in data.columns:
    if data[i].dtype == 'object':
        data[i] = l1.fit_transform(data[i])
```

حال مجدد اطلاعات دیتاست را مشاهده میکنیم:

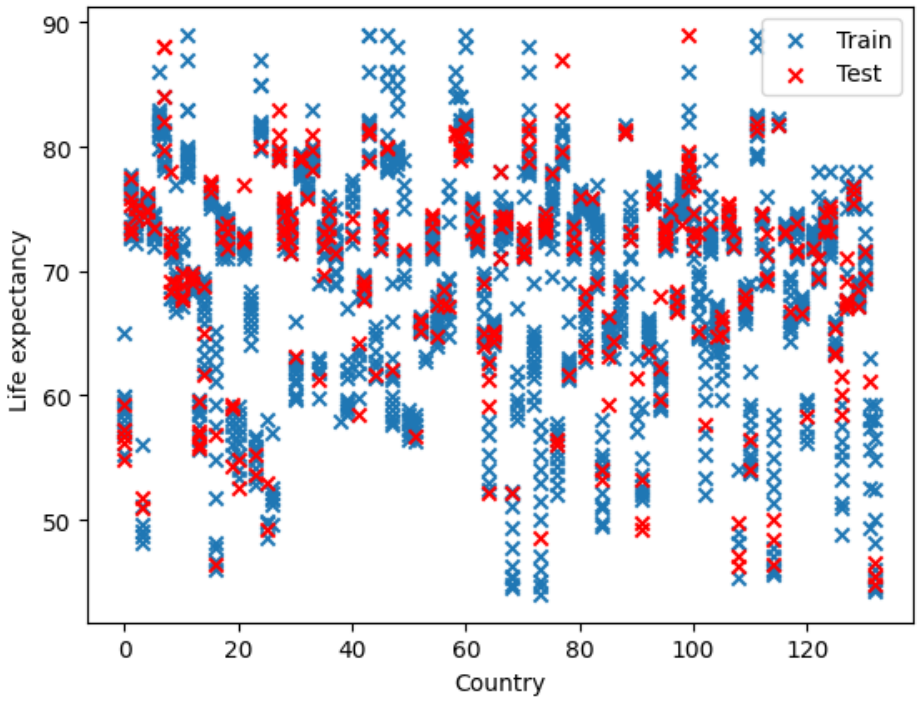
```
data.info()
```

```
Int64Index: 1649 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          1649 non-null   int64
 1   Year                             1649 non-null   int64
 2   Status                           1649 non-null   int64
 3   Life expectancy                  1649 non-null   float64
 4   Adult Mortality                  1649 non-null   float64
 5   infant deaths                    1649 non-null   int64
 6   Alcohol                          1649 non-null   float64
 7   percentage expenditure           1649 non-null   float64
 8   Hepatitis B                      1649 non-null   float64
 9   Measles                          1649 non-null   int64
 10   BMI                             1649 non-null   float64
 11  under-five deaths                1649 non-null   int64
 12  Polio                            1649 non-null   float64
 13  Total expenditure                1649 non-null   float64
 14  Diphtheria                       1649 non-null   float64
 15   HIV/AIDS                        1649 non-null   float64
 16  GDP                              1649 non-null   float64
 17  Population                       1649 non-null   float64
 18   thinness  1-19 years            1649 non-null   float64
 19   thinness 5-9 years             1649 non-null   float64
 20  Income composition of resources  1649 non-null   float64
 21  Schooling                        1649 non-null   float64
dtypes: float64(16), int64(6)
```

هر دو مشکل رفع شدند.

داده های آموزش و تست را رسم میکنیم:

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Selecting the first feature for the scatter plot
feature_for_plot = X.columns[0]

x_train, x_test, y_train, y_test = train_test_split(X[feature_for_plot],
y, test_size=0.2, random_state=7)

plt.scatter(x_train, y_train, marker='x')
plt.scatter(x_test, y_test, c='r', marker='x')

plt.xlabel(feature_for_plot)
plt.ylabel('Life expectancy ')
plt.legend(['Train', 'Test'])
plt.show()
```



خط feature_for_plot = X.columns[0] مربوط به هم سایز کردن x,y است.

حال به دنبال پیدا کردن best_ccp_alpha و best_max_depth هستیم و معیارمان R2_score است.

برای این کار از دو حلقه در ترکیب با هم استفاده میکنیم:

```
from sklearn.model_selection import GridSearchCV
```

```
# Assuming 'data' is your DataFrame
X = data.drop('Life expectancy ', axis=1)
y = data['Life expectancy ']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)
```

مجدد تقسیم بندی میکنیم تا در ادامه به مشکل برنخوریم.

```
best_score = -np.inf
# Define the parameter grids
ccp_alpha_values = np.linspace(0, 0.005, num=100)
max_depth_values = range(1, 30)
```

مقدار ccp را از ۰ تا ۰.۰۰۵ تغییر میدهیم همچنین عمق را از ۱ تا ۳۰ پیمایش میکنیم.

```
for ccp_alpha in ccp_alpha_values:
    for max_depth in max_depth_values:
        reg = DecisionTreeRegressor(ccp_alpha=ccp_alpha,
max_depth=max_depth, random_state=7)
        reg.fit(x_train, y_train)
        score = reg.score(x_test, y_test)

        if score > best_score:
            best_score = score
            best_ccp_alpha = ccp_alpha
            best_max_depth = max_depth
```

بهترین مقادیر را بر اساس R2Score ذخیره میکنیم .

```
reg = DecisionTreeRegressor(ccp_alpha=best_ccp_alpha,
max_depth=best_max_depth, random_state=7)
reg.fit(x_train, y_train)
```

با پارامترهای به دست آمده آموزش میدهیم.

```
# Generate data points for plotting
xp = np.linspace(X[feature_of_interest].min(),
X[feature_of_interest].max(), 100)  # Use linspace for a 1D array
xp = xp.reshape(-1, 1)  # Reshape to a column vector
yp = reg.predict(np.column_stack([xp] * X.shape[1]))  # Repeat the array
to match the number of features
```

```
# Flatten x_train and x_test for scatter plots
plt.scatter(x_train[feature_of_interest].values.flatten(), y_train,
marker='x')
plt.scatter(x_test[feature_of_interest].values.flatten(), y_test, c='r',
marker='x')
plt.plot(xp, yp, color='gold', linewidth=3)

plt.legend(['Train', 'Test', 'DT'])
plt.xlabel(feature_of_interest)
plt.ylabel('Life expectancy')
plt.title('Decision Tree Regression')
plt.show()
```

رسم میکنیم و نتیجه به شکل زیر به دست می آید:

بهترین مقدار R2Score :

```
reg.score(x_train, y_train), reg.score(x_test, y_test)
```

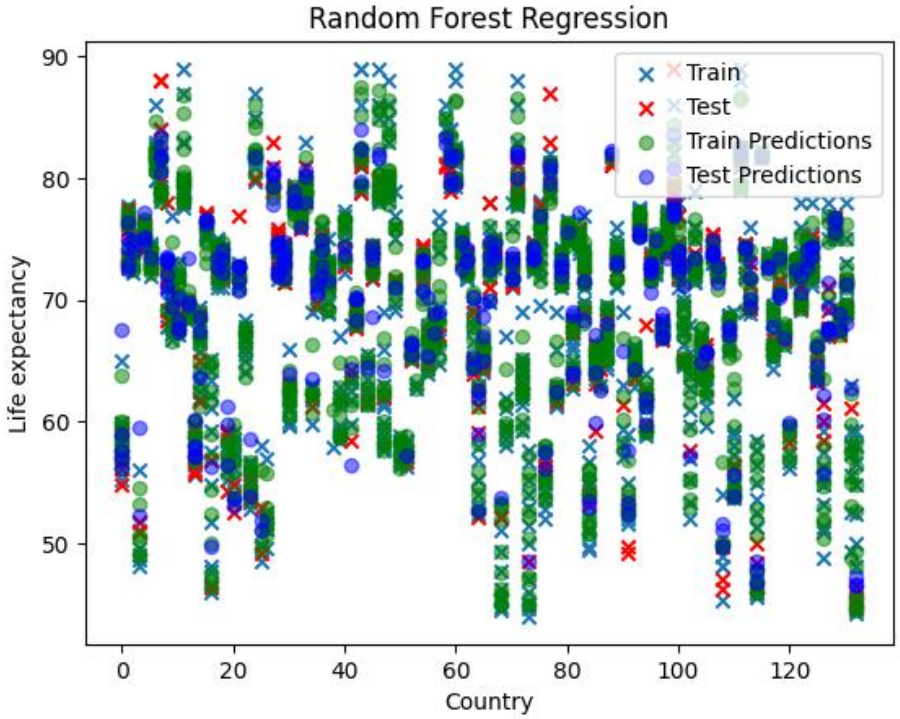
(0.9903073081925604, 0.9264743641874817)

یک راه دیگر استفاده از random forest است که با استفاده از چند درخت تصمیم نتیجه بهتری از روش قبل به ما میدهد:

```
from sklearn.ensemble import RandomForestRegressor

# Create the Random Forest Regressor
rf_reg = RandomForestRegressor(n_estimators=100, random_state=7)

# Train the model
rf_reg.fit(x_train, y_train)

# Predictions
y_train_pred = rf_reg.predict(x_train)
y_test_pred = rf_reg.predict(x_test)

# Plotting
plt.scatter(x_train[feature_of_interest].values.flatten(), y_train,
marker='x')
plt.scatter(x_test[feature_of_interest].values.flatten(), y_test, c='r',
marker='x')
plt.scatter(x_train[feature_of_interest].values.flatten(), y_train_pred,
marker='o', c='green', alpha=0.5)
plt.scatter(x_test[feature_of_interest].values.flatten(), y_test_pred,
marker='o', c='blue', alpha=0.5)

plt.legend(['Train', 'Test', 'Train Predictions', 'Test Predictions'])
plt.xlabel(feature_of_interest)
plt.ylabel('Life expectancy')
plt.title('Random Forest Regression')
plt.show()
```

در زمینه RandomForestRegressor در scikit-learn ، n_estimators یک ابرپارامتر است که تعداد درخت‌های تصمیم را در جنگل تصادفی نشان می‌دهد. الگوریتم جنگل تصادفی درخت های تصمیم گیری متعددی را در طول آموزش ایجاد می کند و پیش بینی ها با میانگین گیری از پیش بینی های هر درخت برای وظایف رگرسیون انجام می شود.

خروجی نهایی:



ارزیابی:

```
rf_reg.score(x_train, y_train), rf_reg.score(x_test, y_test)
output
(0.9941582145255462, 0.9592518276515615)
```

که برای داده های آموزش در حد چند دهم درصد بیشتر و برای داده های تست حدود ۳ درصد بهتر شده است.

سوال ۵:

توضیح اولیه: سه تصویر مختلف گل از اینترنت دانلود کردم ولی هر چه تلاش کردم موفق نشدم از درایو داخل کولب پوشه عکس ها را وارد کنم برای همین از داخل کامپیوتر عکس ها را آپلود کردم و فایل سه تصویر در فایل زیپ ارائه شده قرار داده شده است.

```
import math

import cv2
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Markdown
from glob2 import glob

PATH = '/content/data'
```

'/content/data' همان مسیر پوشه سه عکس است. این پوشه را خودم دستی در کولب درست کردم.

جاهای خواسته شده را تکمیل میکنیم:

2. Fuzzification of Pixel Intensity

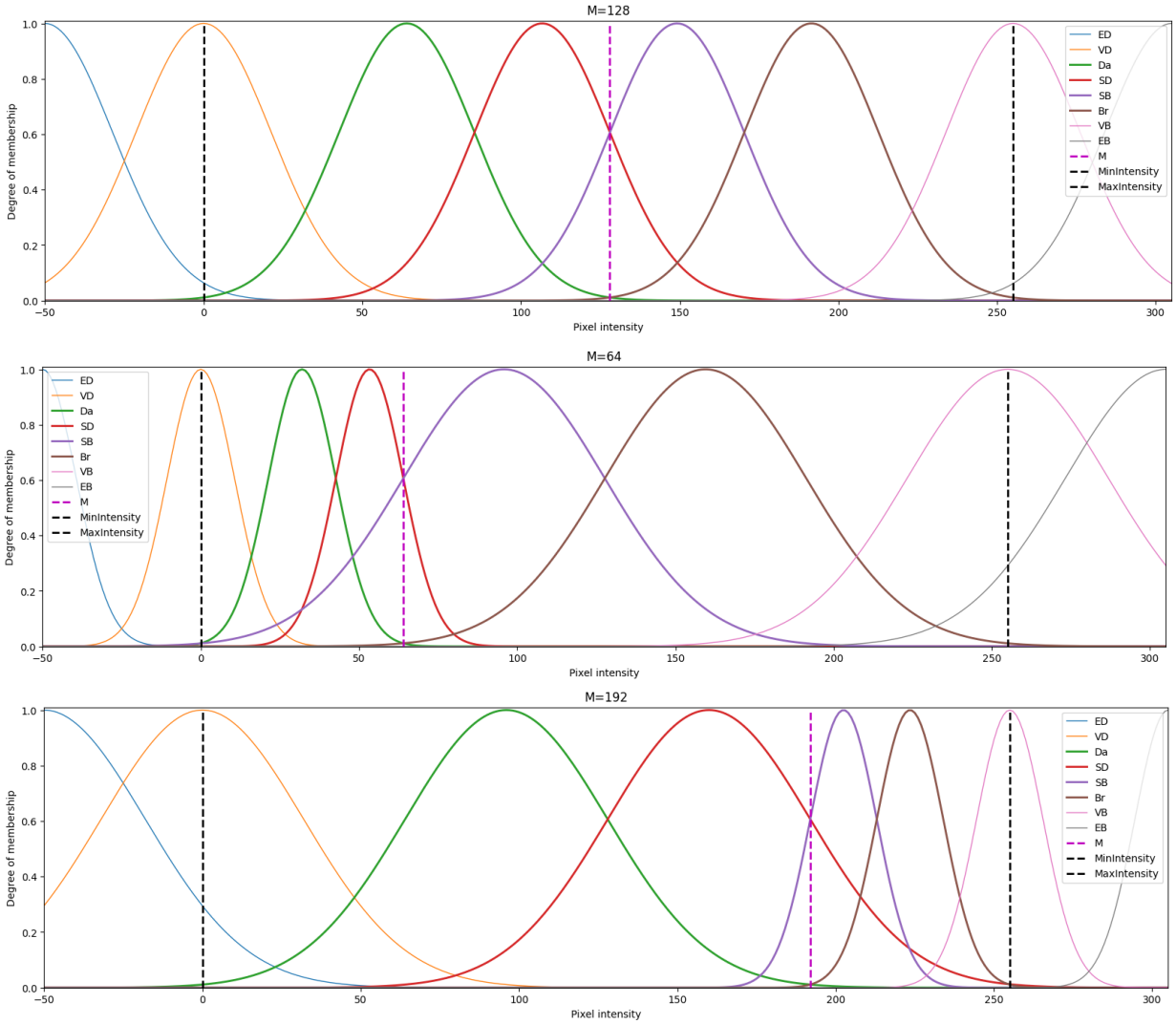
```
import numpy as np
import matplotlib.pyplot as plt

# Gaussian Function:
def G(x, mean, std):
    return np.exp(-0.5 * np.square((x - mean) / std))

def membership_functions(x, M):
    functions = {
        'ED': G(x, -50, M/6),
        'VD': G(x, 0, M/6),
        'Da': G(x, M/2, M/6),
        'SD': G(x, 5*M/6, M/6),
        'SB': G(x, M + (255-M)/6, (255-M)/6),
        'Br': G(x, M + (255-M)/2, (255-M)/6),
        'VB': G(x, 255, (255-M)/6),
        'EB': G(x, 305, (255-M)/6),
    }
    return functions
```

این تابع مجموعه ای از توابع عضویت گاوسی را با ابزارهای مختلف 'ED' برای بسیار تاریک، 'VD'برای خیلی تاریک و غیره بر اساس مقدار M مشخص شده تولید می کند. یک فرهنگ لغت را برمی گرداند که در آن کلیدها برچسب هایی برای توابع مختلف عضویت هستند و مقادیر آرایه هایی هستند که حاوی مقادیر تابع عضویت برای مقادیر x متناظر هستند.

```
def plot_membership_functions(x, functions, M):
    plt.figure(figsize=(20,5))
    for label, values in functions.items():
        plt.plot(x, values, label=label, linewidth=1 if
label.startswith('E') or label.startswith('V') else 2)

    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()

# Values of M
M_values = [128, 64, 192]

# Plot membership functions for each M
for M in M_values:
    x = np.arange(-50, 306)
    functions = membership_functions(x, M)
    plot_membership_functions(x, functions, M)
```

این تابع توابع عضویت تولید شده توسط Membership_functions را ترسیم می کند. از Matplotlib برای ایجاد یک شکل استفاده می کند و سپس هر تابع عضویت را رسم می کند. همچنین خطوط عمودی را برایM ، MinIntensityو MaxIntensity برای مرجع اضافه می کند. این تابع ظاهر طرح و برچسب ها را برای وضوح تنظیم می کند. نمودار حاصل توابع عضویت برای برچسب های زبانی مختلف را بر اساس مقدار M مشخص شده نشان می دهد.

خروجی حاصل:



4. Inference and Defuzzication (Mamdani's method)

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

ورودی ها :x مقادیر ورودی (به عنوان مثال، شدت پیکسل تابع عضویت) به عنوان مثال، ExtremelyDark، VeryDark، و غیره .M پارامتر تابع عضویت- thres آستانه برای محدود کردن مقادیر عضویت خروجی مقادیر عضویت را با استفاده از تابع عضویت مشخص شده f با پارامتر M محاسبه می کند .مقادیر عضویت را در آستانه‌های مشخصی قرار می‌دهد.

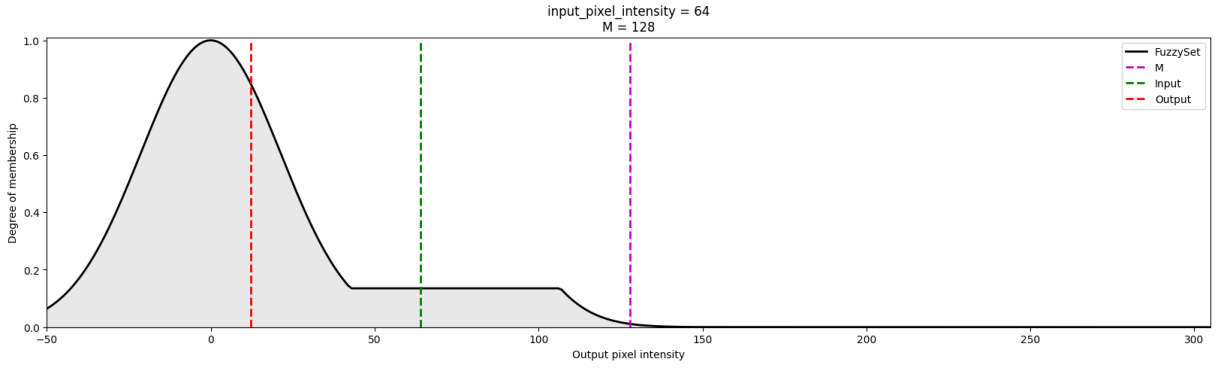
```
def AggregateFuzzySets(fuzzy_sets):
```

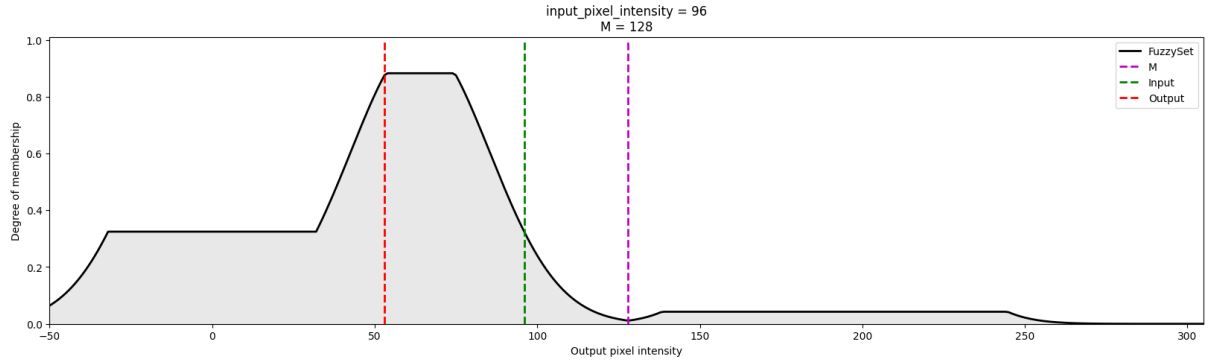
```
    return np.max(np.stack(fuzzy_sets), axis=0)

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)
```

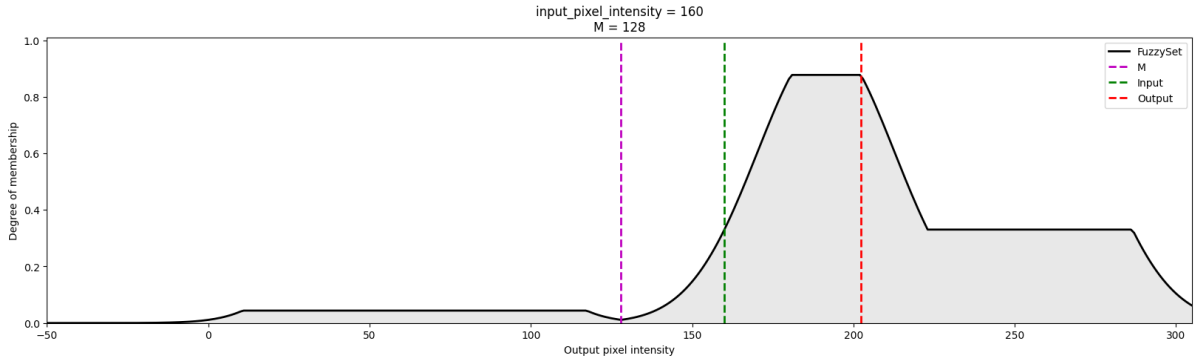
ورودی ها :i مقادیر ورودی (به عنوان مثال، شدت پیکسل) Mپارامتر برای توابع عضویت get_fuzzy_setبرای نشان دادن اینکه آیا مجموعه فازی را به همراه مرکز مرکزی باید برگردانید یا خیر، پرچم گذاری کنید.
خروجی :درجه عضویت برای کلاس های مختلفVD ، Da، SD، SB، Br، VBرا با استفاده از M مشخص شده محاسبه می کند .استنتاج فازی را انجام می دهد، مجموعه های فازی را جمع می کند و مرکز (مقدار واضح) مجموعه فازی حاصل را محاسبه می کند .اگر get_fuzzy_set روی True تنظیم شده باشد، به صورت اختیاری، مجموعه فازی را برمی‌گرداند.

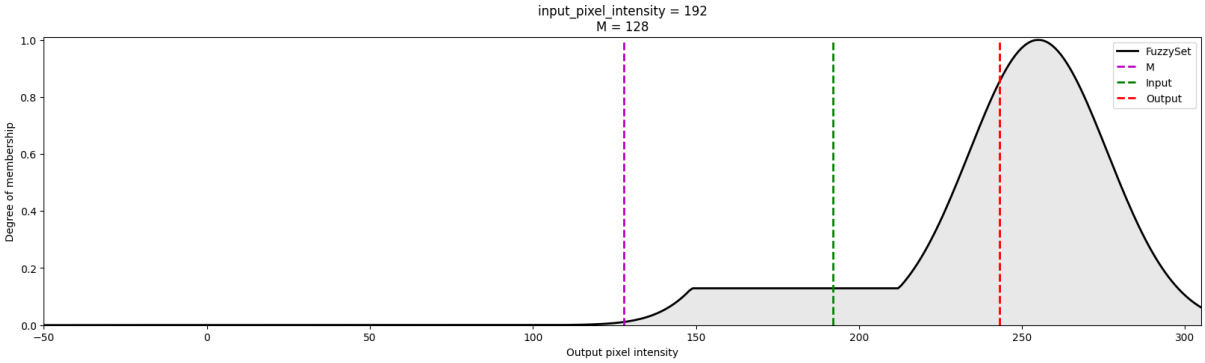
```
def plot_inference_result(pixel, M, centroid, output_fuzzy_set):
    x = np.arange(-50, 306)
    plt.figure(figsize=(20, 5))
    plt.plot(x, output_fuzzy_set, 'k-', label='FuzzySet', linewidth=2)
    plt.plot((M, M), (0, 1), 'm--', label='M', linewidth=2)
```

```
    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, .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()

def infer_and_plot_for_pixels(pixels, M):
    for pixel in pixels:
        centroid, output_fuzzy_set = Infer(np.array([pixel]), M,
get_fuzzy_set=True)
        plot_inference_result(pixel, M, centroid, output_fuzzy_set)

# Values of M
M = 128

# Pixels to infer and plot
pixels_to_infer = [64, 96, 160, 192]

# Infer and plot for each pixel
infer_and_plot_for_pixels(pixels_to_infer, M)
```





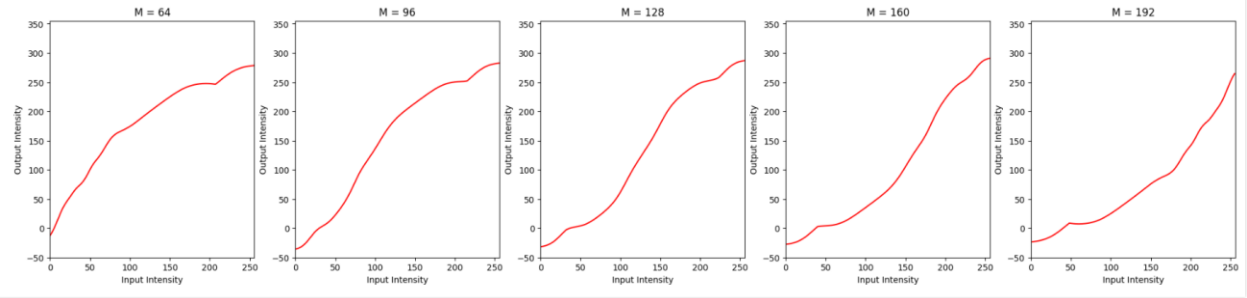





```
def infer_for_mean(M):
    x = np.arange(256)
    y = np.array([Infer(np.array([i]), M) for i in x])
    return x, y

means = (64, 96, 128, 160, 192)
plt.figure(figsize=(25, 5))

for i, M in enumerate(means):
    x, y = infer_for_mean(M)
    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()
```

کد بالا، برای هر مقدار M در آرایه میانگین یک شکل با نمودارهای فرعی ایجاد می کند infer_for_mean . را برای هر M برای بدست آوردن مقادیر ورودی (x) و خروجی (y) فراخوانی می کند .نگاشت های ورودی-خروجی را در نمودارهای فرعی جداگانه ترسیم می کند .محدودیت ها، برچسب ها و عناوین محور را برای وضوح تنظیم می کند .کل شکل را نشان می دهد. خروجی در نهایت به شکل زیر است:



همین طور دو بخش زیر را تکمیل میکنیم:

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

1. Load sample photos

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

(3,)

همان سه عکس گفته شده است که در ادامه قابل مشاهده اند.

Apply FCE on sample photos

```
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()
```

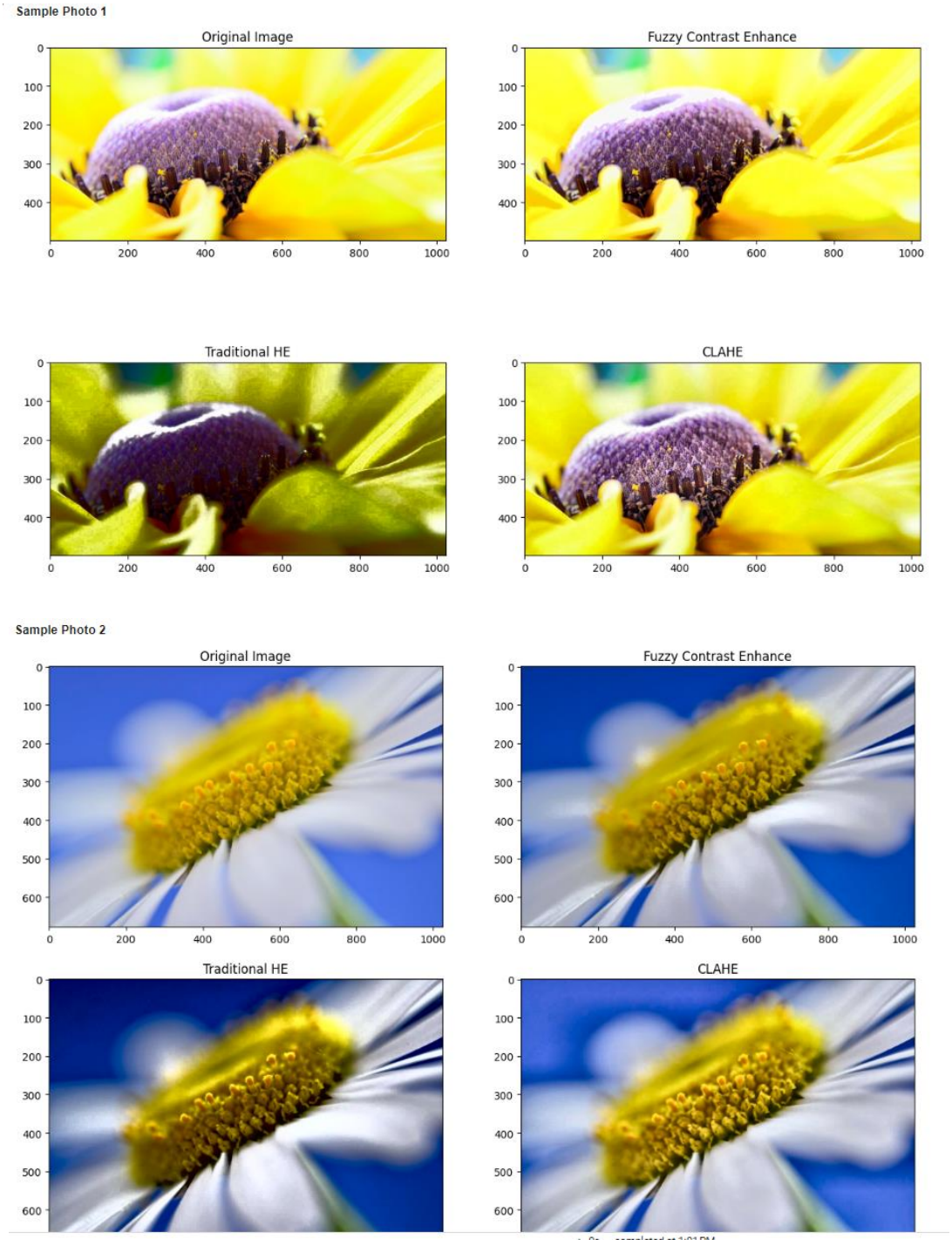
برای هر تصویر (img) در حلقه، سه تکنیک مختلف پردازش تصویر اعمال میشود:

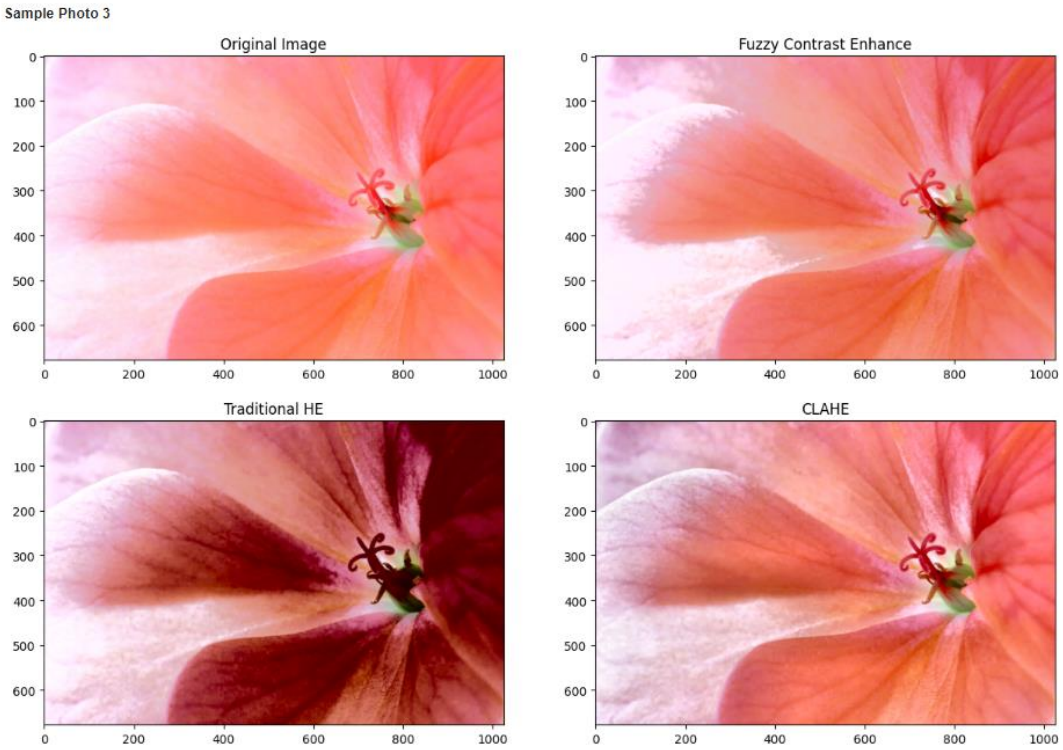
FuzzyContrastEnhance : یک روش افزایش کنتراست مبتنی بر منطق فازی

HE Histogram Equalization : یک روش سنتی برای افزایش کنتراست

CLAHE .تعادل هیستوگرام تطبیقی با کنتراست: روش دیگری برای افزایش کنتراست که کنتراست را در مناطق محلی محدود می کند.

خروجی:





بررسی عملکرد

شاخص زمان:

Fuzzy Contrast Enchance

```
%%time
for i in range(3):
    FuzzyContrastEnhance(data[i])
```

```
CPU times: user 617 ms, sys: 18.7 ms, total: 636 ms
Wall time: 673 ms
```

Traditional HE

```
%%time
for i in range(3):
    HE(data[i])
```

output

CPU times: user 70.9 ms, sys: 1.88 ms, total: 72.7 ms
Wall time: 75.9 ms

CLAHE

```
%%time
for i in range(3):
    CLAHE(data[i])
```

```
CPU times: user 85.9 ms, sys: 1.56 ms, total: 87.4 ms
Wall time: 81.1 ms
```

تحلیل شاخص زمانی:

CPU Time (User Component):
Fuzzy Contrast Enhance: 617 ms
Traditional HE: 70.9 ms
CLAHE: 85.9 ms

CPU Time (Sys Component):
Fuzzy Contrast Enhance: 18.7 ms
Traditional HE: 1.88 ms
CLAHE: 1.56 ms

Total CPU Time:
Fuzzy Contrast Enhance: 636 ms
Traditional HE: 72.7 ms
CLAHE: 87.4 ms

Wall Time:
Fuzzy Contrast Enhance: 673 ms
Traditional HE: 75.9 ms
CLAHE: 81.1 ms

افزایش کنتراست فازی در مقایسه با Traditional HE وCLAHE ، هم از نظر اجزای کاربر و هم از نظر سیستم، زمان بسیار بیشتری را صرف می‌کند.

HE سنتی در بین سه روش از نظر CPU و زمان دیوار سریعترین است.

CLAHE کمی بیشتر از Traditional HE زمان می برد، اما همچنان سریعتر از Fuzzy Contrast Enhanceاست.

زمان دیواری زمان واقعی سپری شده را منعکس می کند، از جمله زمان انتظار ناشی از عوامل مرتبط با سیستم، و با روند مشاهده شده در زمان CPU مطابقت دارد.

در نهایت:

Best CPU Time (User Component):
Traditional HE: 70.9 ms
Best CPU Time (Sys Component):
Traditional HE: 1.88 ms
Best Total CPU Time:
Traditional HE: 72.7 ms
Best Wall Time:
Traditional HE: 75.9 ms

شاخص 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]))}'))
```

خروجی:

FCE: 77.13221055054723

HE: 76.39168991402406

CLAHE: 77.26701861954572

تحلیل: مقادیر PSNR همگی نسبتاً بالا هستند، که نشان می دهد تصاویر بازسازی شده از نظر کیفیت به تصاویر اصلی نزدیک هستند .در بین سه روش، CLAHE (77.27)دارای بالاترین مقدار PSNR است که نشان می دهد به طور متوسط کیفیت تصویر کمی بهتر در مقایسه با FCE (77.13) و HE (76.39) ارائه می دهد . FCEو CLAHE مقادیر PSNR مشابهی دارند و تفاوت ها نسبتاً کوچک است .به طور خلاصه، بر اساس متریکPSNR ، به نظر می رسد هر سه روش افزایش کنتراستFCE ، HE، CLAHEعملکرد خوبی دارند، با CLAHE به طور متوسط از نظر کیفیت تصویر در مقایسه با دو روش دیگر عملکرد کمی بهتر از خود نشان می دهد. با این حال، ذکر این نکته ضروری است که PSNR تنها یک معیار است و درک بصری کیفیت تصویر ممکن است بسته به ویژگی‌های خاص تصاویر و زمینه برنامه متفاوت باشد.

آدرس کدها در گیت هاب:

https://github.com/AliKhodarahmy/machinelearning2023/tree/main/miniproject/3