به نام خدا

# گزارش مینی پروژه سوم مبانی سیستم های هوشمند


رضا آقاجری ۹۸۱۹۵۸۳

زمستان ۱۴۰۲

## سوال اول:

حل دستی:

به نام خدا

رضا آقاوری ۹۸۱۹۵۸۳

گران مرتبه اول:

$$\|f-g\|_\infty < \|\partial g/\partial x_1\|_\infty \times h_1 + \|\partial g/\partial x_2\|_\infty \times h_2 + \cdots$$

$$\partial g/\partial x_1 = \frac{-1}{(3+x_1+x_2)^2}$$

$$\|\partial g/\partial x_1\|_\infty = 1 \quad , \quad \|\partial g/\partial x_2\|_\infty = 1$$

تعدی مرکز

$$\varepsilon = 0.1 \Longrightarrow h_1+h_2 < 0.1 \xrightarrow{h_1=h_2} 2h_1 < 0.1 \Rightarrow h_1 < 0.05 \Rightarrow h_1=h_2=0.04$$

$$n = \frac{1-(-1)}{0.04} = 50 \longrightarrow \text{تعداد نقاط} = 50\times 50+1 = 2501$$

گران مرتبه دوم:

$$\|f-g\|_\infty < \frac{1}{8}\left[\|\partial^2 g/\partial x_1^2\|_\infty \times h_1^2 + \|\partial^2 g/\partial x_2^2\|_\infty \times h_2^2 + \cdots\right]$$

$$\partial^2 g/\partial x_1^2 = \frac{-2(3+x_1+x_2)(-1)}{(3+x_1+x_2)^4} = \frac{2}{(3+x_1+x_2)^3} \Rightarrow \|\partial^2 g/\partial x_1^2\|_\infty = 2 \text{ و } \|\partial^2 g/\partial x_2^2\|_\infty = 2$$

$$\varepsilon = 0.1 \Rightarrow \frac{1}{8}\left[2h_1^2+2h_2^2\right] < 0.1 \xrightarrow{h_1=h_2} 4h_1^2 < 0.8 \Rightarrow h_1^2 < 0.2$$

تعدی مرکز

$$\Rightarrow h_1 < 0.45 \Rightarrow h_1=h_2=0.4 \longrightarrow n = \frac{1-(-1)}{0.4} = 5 \xrightarrow{\text{تعداد نقاط}} = 26$$

با توجه به پارامتر های بدست آمده سیستم فازی را به شکل زیر پیاده سازی میکنیم.

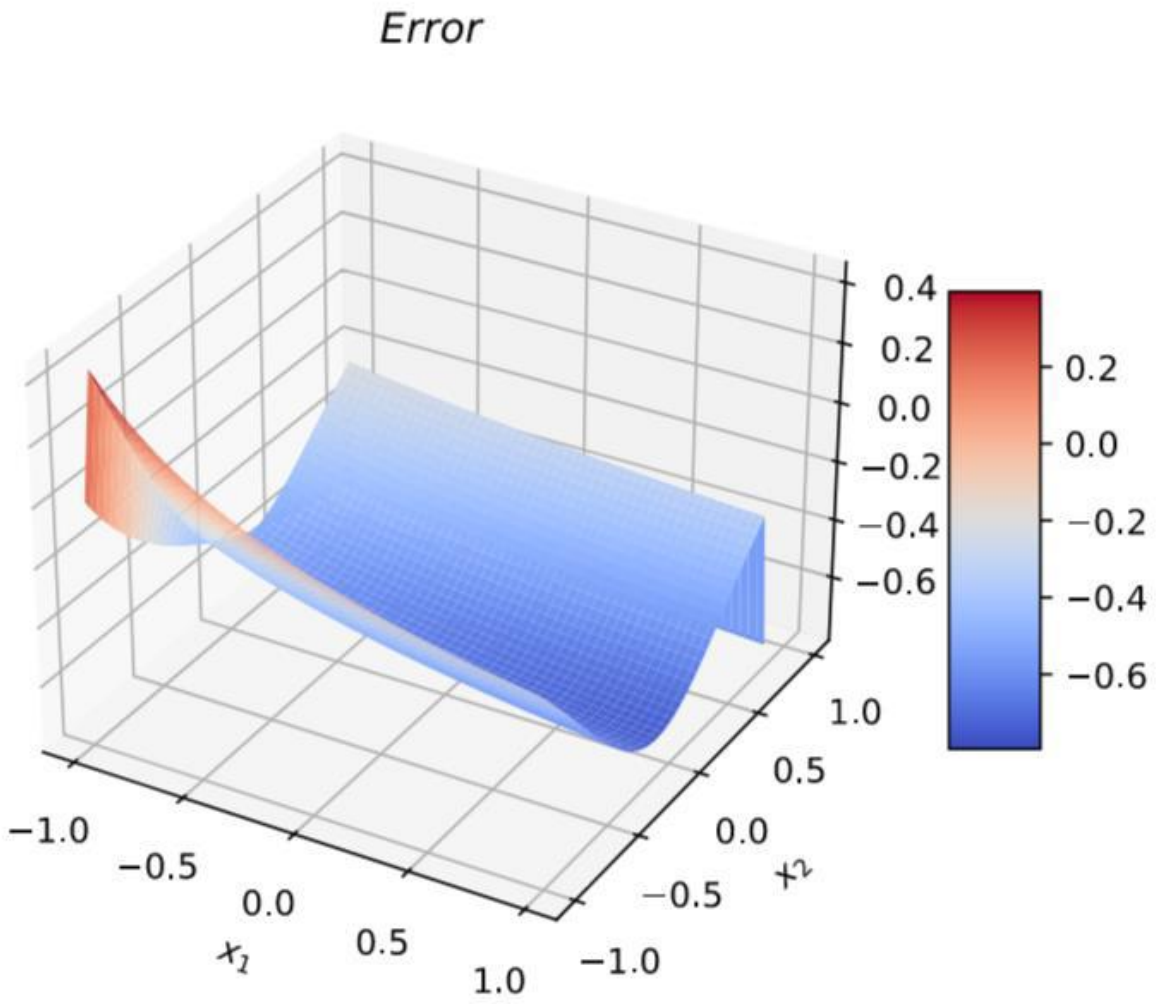
کران مرتبه اول:

با توجه به محاسبات انجام شده h=0.04 و n=50 را اعمال می نماییم.

```
import time
start_time = time.time()
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import warnings
warnings.filterwarnings('ignore')

alpha = -1
beta = 1
h = 0.04
N = 50

x1 = np.arange(alpha, beta, 0.01)
x2 = np.arange(alpha, beta, 0.01)
x1, x2 = np.meshgrid(x1, x2)

g_bar = np.zeros((N*N, 1))
e_i1 = np.zeros((N, 1))
e_i2 = np.zeros((N, 1))

num = 0
den = 0
k = 0
```

```
def trimf(x, abc):
    return np.fmax(np.fmin((x-abc[0])/(abc[1]-abc[0]), (abc[2]-x)/(abc[2]-abc[1])), 0)

for i1 in range(1,N):
    for i2 in range(1,N):
        e_i1[i1-1,0] = -1 + h*(i1-1)
        e_i2[i2-1,0] = -1 + h*(i2-1)
        if i1==1:
            mu_A_x1 = trimf(x1, [-1,-1,-1+h])
        elif 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)])

        if i2==1:
            mu_A_x2 = trimf(x2, [-1,-1,-1+h])
        elif 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)])

        g_bar[k,0]= 1/(1+e_i1[i1, 0]**2+e_i2[i2, 0]**2)
        num = num + g_bar[k,0]*mu_A_x1*mu_A_x2
        den=den+mu_A_x1*mu_A_x2
        k=k+1

f_x = num/den
g_x = 1/(3 + x1 + x2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
E = g_x - f_x
surf = ax.plot_surface(x1, x2, E, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$Error$')
ax.set_title('$Error$')
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.savefig('fuzzy1.svg')
plt.show()
```

تابع error که همان اختلاف بین f(x) و g(x) می باشد به صورت زیر حاصل می‌شود.



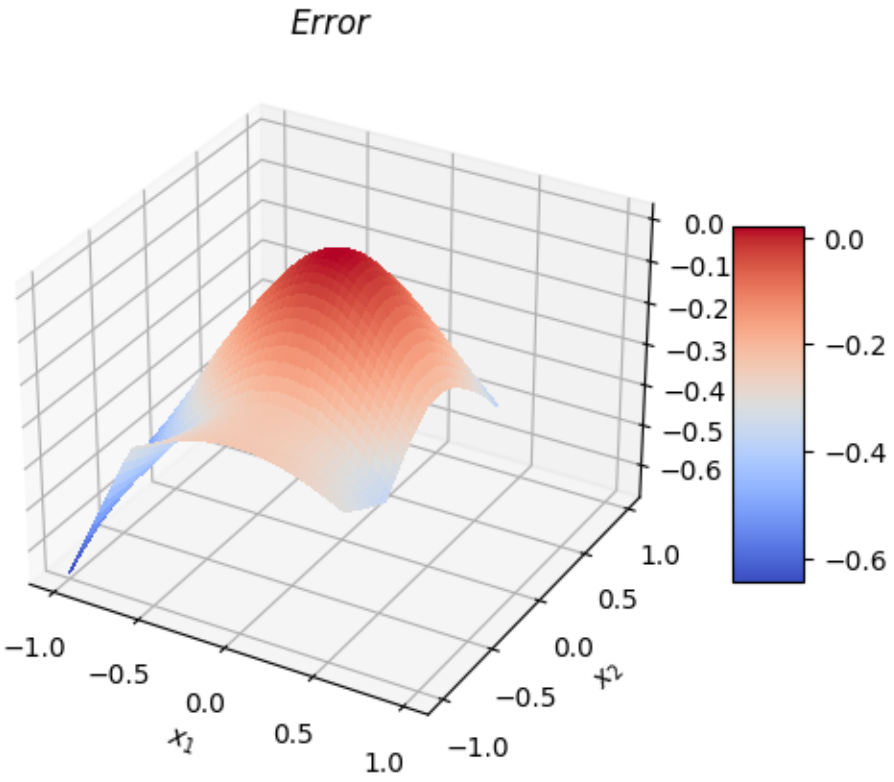
کران مرتبه دوم:

با توجه به محاسبات انجام شده h=0. 4 و n=5 را اعمال می نماییم.

```
import time
start_time = time.time()
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import warnings
warnings.filterwarnings('ignore')

alpha = -1
beta = 1
h = 0.4
N = 5

x1 = np.arange(alpha, beta, 0.01)
x2 = np.arange(alpha, beta, 0.01)
x1, x2 = np.meshgrid(x1, x2)

g_bar = np.zeros((N*N, 1))
e_i1 = np.zeros((N, 1))
e_i2 = np.zeros((N, 1))

num = 0
den = 0
k = 0
```

```
def trimf(x, abc):
    return np.fmax(np.fmin((x-abc[0])/(abc[1]-abc[0]), (abc[2]-x)/(abc[2]-abc[1])), 0

for i1 in range(1,N):
    for i2 in range(1,N):
        e_i1[i1-1,0] = -1 + h*(i1-1)
        e_i2[i2-1,0] = -1 + h*(i2-1)
        if i1==1:
            mu_A_x1 = trimf(x1, [-1,-1,-1+h])
        elif 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)])

        if i2==1:
            mu_A_x2 = trimf(x2, [-1,-1,-1+h])
        elif 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)])

        g_bar[k,0]= 1/(1+e_i1[i1, 0]**2+e_i2[i2, 0]**2)
        num = num + g_bar[k,0]*mu_A_x1*mu_A_x2
        den=den+mu_A_x1*mu_A_x2
        k=k+1

f_x = num/den
g_x = 1/(1+x1**2+x2**2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
E = g_x - f_x
surf = ax.plot_surface(x1, x2, E, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$Error$')
ax.set_title('$Error$')
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.savefig('fuzzy2.svg')
plt.show()

# Print time of execution
print("--- %s seconds ---" % (time.time() - start_time))
```

تابع error که همان اختلاف بین f(x) و g(x) می باشد به صورت زیر حاصل می‌شود.




درحالت دوم طبق محاسبات در حل دستی، با برآورده کردن دقت مورد نظر ۰.۱، تعداد توابع تعلق به وضوح کمتری در مقایسه با حالت اول نیاز است.

## سوال دوم:

در ابتدا سری زمانی مکی گلاس را با ۹۰۰ داده تولید می‌کنیم. x یک بردار برای مکی‌گلاس در نظر می‌گیرد و dataset-۱ یک ماتریس برای دیتاست تولید می‌کند. با لوپ سری زمانی مکی گلاس را تولید می‌کنیم و سری زمانی را رسم می‌کنیم.

```
n=900;
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)
```

S۱ و S۲ را برای شمارهی توابع تعلق مختلف تعریف میکنیم

```
[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
```

به هر قانون درجه اختصاص می دهیم. با استفاده از توابع عضویت، برای هر ویژگی ورودی در داده‌های آموزش، درجه عضویت در هر توابع عضویت محاسبه می‌شود. سپس با استفاده از این درجه‌ها، قانون مناسب برای هر ویژگی ورودی انتخاب و درجه قانون و خروجی محاسبه می‌شود.

```
    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
```

قوانین اضافه را حذف میکنیم

```
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
```

سپس سیستم استنتاج فازی را تشکیل می دهیم.

```
disp('******************************')
disp(['Final rules for ', num2str(num_membership_functions),'
membership functions for each input variables'])
final_Rules=Rules_total(1:1, :);

Fisname='Prediction controller';
Fistype='mamdani';
Andmethod='prod';
Ormethod='max';
Impmethod='prod';
Aggmethod='max';
Defuzzmethod='centroid';
fis=newfis(Fisname, Fistype, Andmethod, Ormethod, Impmethod,
Aggmethod, Defuzzmethod);
```

متغیر های ورودی و خروجی و توابع تعلق را اضافه میکنیم.

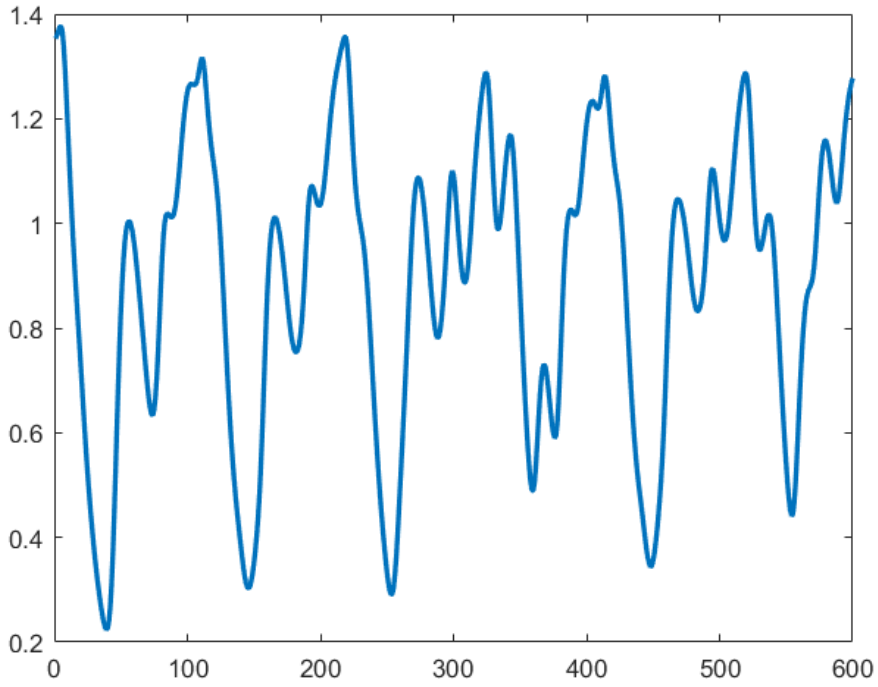
```
for num_input = 1:4
    fis = addInput(fis, [0.1 1.7], "Name", ['x', num2str(num_input)]);
end
fis = addOutput(fis,[0.1, 1.7],  'Name', 'x5');

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
```

قوانین غیر صفر را به سیستم استنتاج اضافه میکنیم.

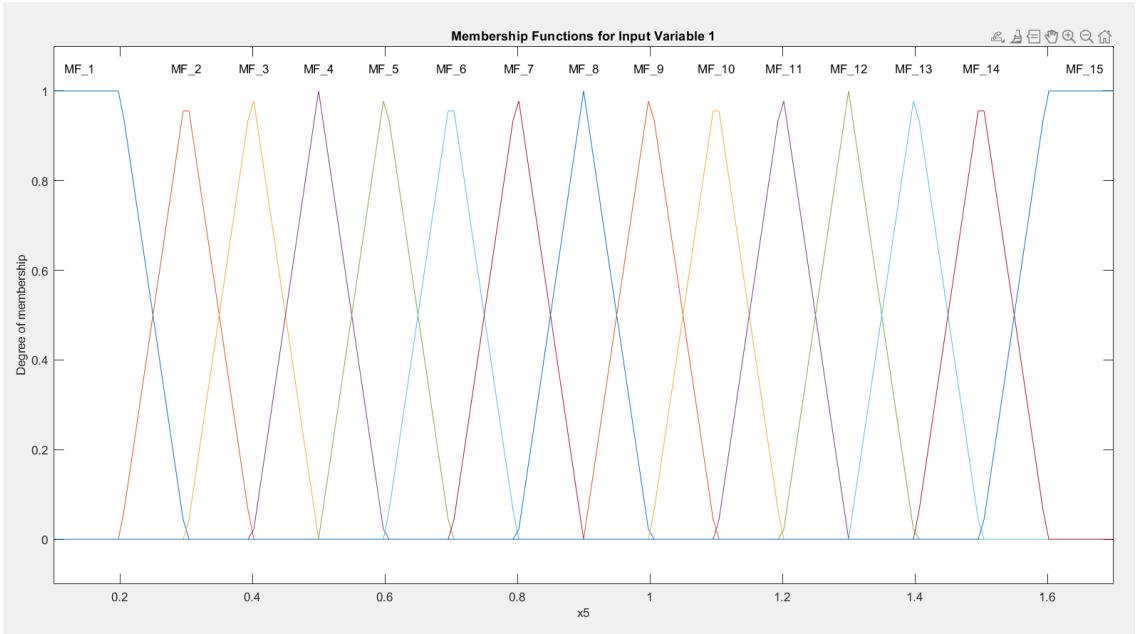
```
non_zero_rows = any(Rules_total(:, 1:5), 2);
fis_Rules = ones(sum(non_zero_rows), 7);
fis_Rules(:, 1:6) = Rules_total(non_zero_rows, 1:6);
fis = addrule(fis, fis_Rules);
```

۳۰۰ نقطه سری زمانی را با این سیستم استنتاج فازی درست شده، پیش‌بینی می‌کنیم و مقادیر به دست آمده و واقعی را رسم می‌کنیم. همچنین در نهایت توابع تعلق را نیز رسم می‌کنیم.

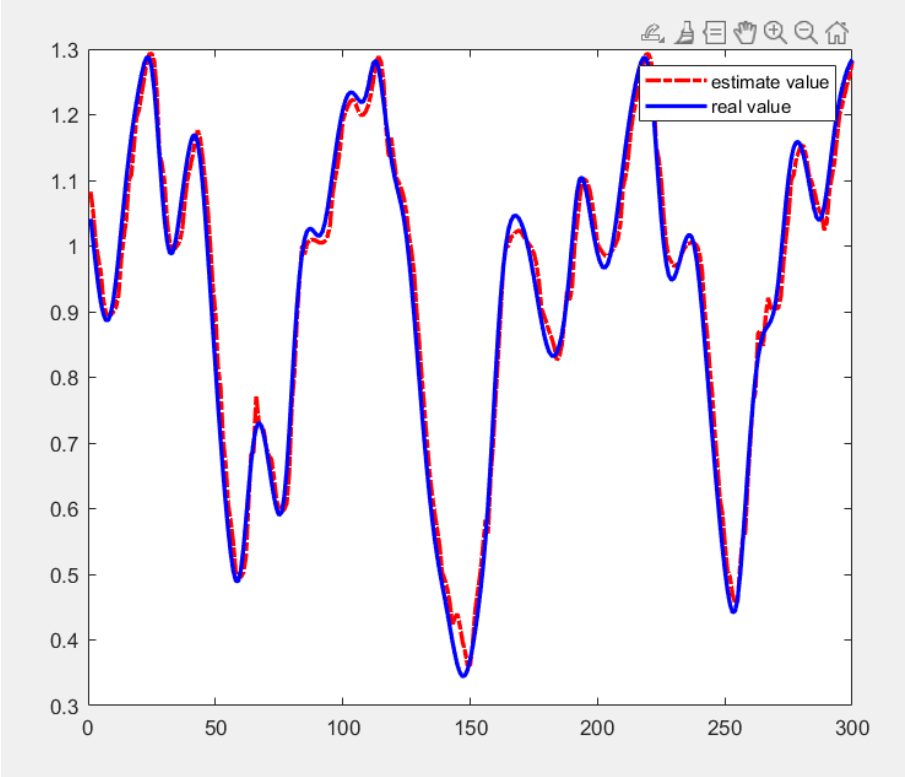
```
jadval_prediction=zeros(300,2);
f=1;
for i=301:600
    input=dataset(i, 2:6);
    output1=dataset(i, 6);
    x5=evalfis([input(1, 1); input(1, 2); input(1,3); input(1,4)],
fis);
    jadval_prediction(f, :)= [f, x5];
    f=f+1;
end
figure;
plot(jadval_prediction(:,1),jadval_prediction(:,2), 'r-.',
'Linewidth', 2);
hold on;
plot(jadval_prediction(:,1),dataset(301: 600, 6), 'b', 'Linewidth',
2);
legend('estimate value', 'real value')
end

inputVariableIndex = 1;

figure;
plotmf(fis, 'output', inputVariableIndex);
title(['Membership Functions for Input Variable ',
num2str(inputVariableIndex)]);
```

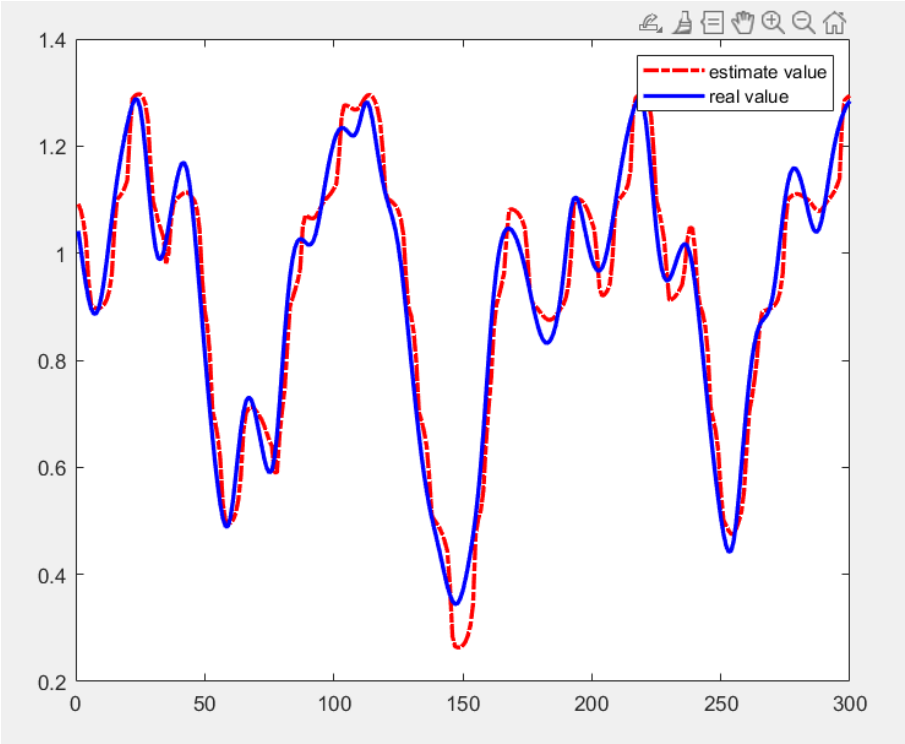



سری زمانی مکی گلاس




توابع تعلق




**اختلاف تابع اصلی و پیشبینی**

**سوال سوم:**

ابتدا پارامتر ها را وارد میکنیم.

```
ParamM = 4;
TrainNum = 200;
TotalNum = 700;
Lambda = 0.1;
```

به منظور پیش‌تخصیص حافظه (Preallocation) برای آرایه‌ها ایجاد میکنیم این کار به بهینه‌سازی عملکرد و سرعت اجرای کد کمک می‌کند.

```
xBar = zeros(TrainNum, ParamM);
gBar = zeros(TrainNum, ParamM);
SigmaVal = zeros(TrainNum, ParamM);
yVal = zeros(TotalNum, 1);
uVal = zeros(TotalNum, 1);
xVal = zeros(TotalNum, 1);
yHat = zeros(TotalNum, 1);
fHat = zeros(TotalNum, 1);
zVal = zeros(TotalNum, 1);
gUVal = zeros(TotalNum, 1);
```

شرایط اولیه را وارد میکنیم

```
uVal(1) = -1 + 2 * rand;
yVal(1) = 0;
gUVal(1) = 0.6 * sin(pi * uVal(1)) + 0.3 * sin(3 * pi * uVal(1)) + 0.1
* sin(5 * pi * uVal(1));
fHat(1) = gUVal(1);
```

سیستم فازی را طراحی میکنیم

```
uMin = -1;
uMax = 1;
stepH = (uMax - uMin) / (ParamM - 1);
for i = 1:ParamM
    xBar(1, i) = uMin + stepH * (i - 1);
    uVal(1, i) = xBar(1, i);
    gBar(1, i) = 0.6 * sin(pi * uVal(1, i)) + 0.3 * sin(3 * pi *
uVal(1, i)) + 0.1 * sin(5 * pi * uVal(1, i));
end

SigmaVal(1, :) = (max(uVal(1, :)) - min(uVal(1, :))) / ParamM;


xBarInitial = xBar(1, :);
SigmaInitial = SigmaVal(1, :);
gBarInitial = gBar(1, :);

for iter = 2:TrainNum
    sumA = 0;
    sumB = 0;
    xVal(iter) = -1 + 2 * rand;
    uVal(iter) = xVal(iter);

    gUVal(iter) = 0.6 * sin(pi * uVal(iter)) + 0.3 * sin(3 * pi *
uVal(iter)) + 0.1 * sin(5 * pi * uVal(iter));
    for j = 1:ParamM
        zVal(j) = exp(-((xVal(iter) - xBar(iter, j)) / SigmaVal(iter,
j))^2);
        sumA = sumA + zVal(j);
        sumB = sumB + gBar(iter, j) * zVal(j);
    end

    fHat(iter) = sumB / sumA;
    yVal(iter + 1) = 0.3 * yVal(iter) + 0.6 * yVal(iter - 1) +
gUVal(iter);
    yHat(iter + 1) = 0.3 * yVal(iter) + 0.6 * yVal(iter - 1) +
fHat(iter);

    for j = 1:ParamM
        gBar(iter + 1, j) = gBar(iter, j) - Lambda * (fHat(iter) -
gUVal(iter)) * zVal(j) / sumA;
```

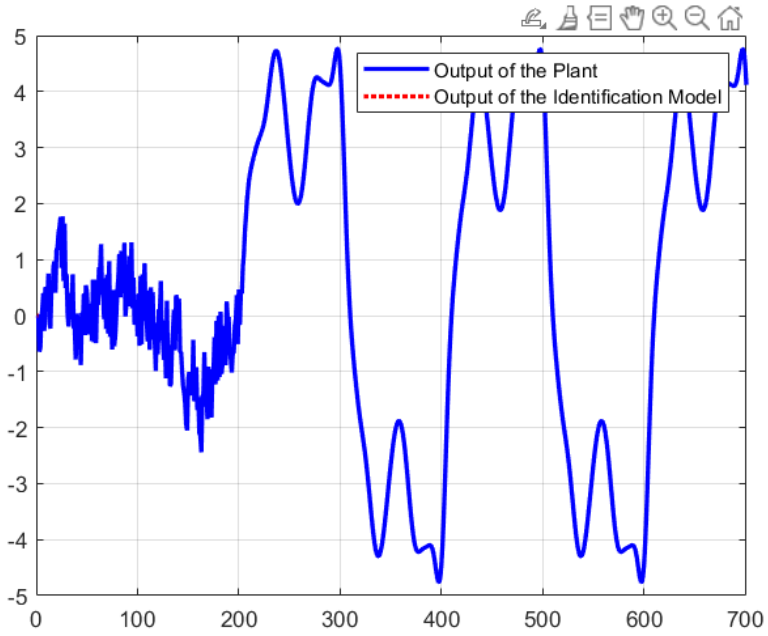
```
        xBar(iter + 1, j) = xBar(iter, j) - Lambda * ((fHat(iter) -
gUVal(iter)) / sumA) * (gBar(iter, j) - fHat(iter)) * zVal(j) * 2 *
(xVal(iter) - xBar(iter, j)) / (SigmaVal(iter, j)^2);
        SigmaVal(iter + 1, j) = SigmaVal(iter, j) - Lambda *
((fHat(iter) - gUVal(iter)) / sumA) * (gBar(iter, j) - fHat(iter)) *
zVal(j) * 2 * (xVal(iter) - xBar(iter, j)) / (SigmaVal(iter, j)^3);
    end
end

xBarFinal = xBar(TrainNum, :);
SigmaFinal = SigmaVal(TrainNum, :);
gBarFinal = gBar(TrainNum, :);
```

این کد داده‌های باقی‌مانده را با استفاده از یک حلقه پردازش می‌کند تا xVal، uVal، gUVal را به‌روزرسانی کرده و fHat را محاسبه کرده، سپس با استفاده از مقادیر قبلی، yVal و yHat را به‌روزرسانی می‌کند.

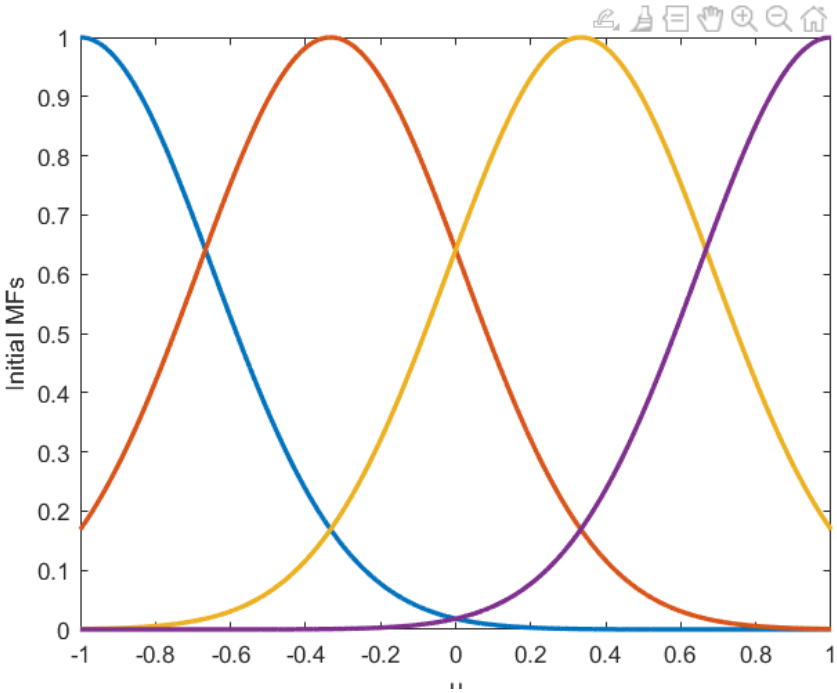
```
for iter = TrainNum:TotalNum
    sumA = 0;
    sumB = 0;
    xVal(iter) = sin(2 * iter * pi / 200);
    uVal(iter) = xVal(iter);

    gUVal(iter) = 0.6 * sin(pi * uVal(iter)) + 0.3 * sin(3 * pi *
uVal(iter)) + 0.1 * sin(5 * pi * uVal(iter));
    for j = 1:ParamM
        zVal(j) = exp(-((xVal(iter) - xBarFinal(j)) /
SigmaFinal(j))^2);
        sumA = sumA + zVal(j);
        sumB = sumB + gBarFinal(j) * zVal(j);
    end
    fHat(iter) = sumB / sumA;
    yVal(iter + 1) = 0.3 * yVal(iter) + 0.6 * yVal(iter - 1) +
gUVal(iter);
    yHat(iter + 1) = 0.3 * yVal(iter) + 0.6 * yVal(iter - 1) +
fHat(iter);
end
```

نتایج را پلات میکنیم

```
figure('Color', [1 1 1]);
plot(1:701, yVal, 'b', 1:701, yHat, 'r:', 'LineWidth', 2);
legend('Output of the Plant', 'Output of the Identification Model');
axis([0 701 -5 5]);
grid on;

plotRange = -2:0.001:2;

figure('Color', [1 1 1]);
for j = 1:ParamM
    mu_x = exp(-((plotRange - xBarInitial(j)) / SigmaInitial(j)).^2);
    plot(plotRange, mu_x, 'LineWidth', 2);
    hold on;
end
xlabel('u');
ylabel('Initial MFs');
axis([-1 1 0 1]);

figure('Color', [1 1 1]);
for j = 1:ParamM
    mu_x = exp(-((plotRange - xBarFinal(j)) / SigmaFinal(j)).^2);
    plot(plotRange, mu_x, 'LineWidth', 2);
    hold on;
end
xlabel('u');
ylabel('Final MFs');
axis([-1 1 0 1]);
```




مقایسه تابع اصلی و پیشبینی




توابع تعلق

## سوال چهارم:

### بخش اول:

کتابخانه های مورد نظر را import میکنیم و دیتا ست را لود میکنیم

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

from sklearn.model_selection import train_test_split
```

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1CQAOCneG14yRdGot0fYyuMzNi0Yux4MJ
```

دیتا ست را نمایش میدهیم

```
df = pd.read_csv('/content/covid.csv')
df
```

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

توابع آنتروپی و information gain را تعریف میکنیم و ig هر ویژگی را نسبت به target که همان infected می باشد محاسبه میکنیم.

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


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
```

```
a = information_gain(df, 'Fever', 'Infected')
b = information_gain(df, 'Cough', 'Infected')
c = information_gain(df, 'Breathing issues', 'Infected')

print(f"IG_Fever: {a}")
print(f"IG_Cough: {b}")
print(f"IG_Breathing_Issues: {c}")
```

```
IG_Fever: 0.12808527889139443
IG_Cough: 0.0391486719030707
IG_Breathing_Issues: 0.396038844928044464
```

براساس محتوای آموزشی، در حال کدنویسی درخت هستیم. یک کلاس با نام `Node` برای گره‌های تصمیم درخت ایجاد کرده و ویژگی‌های آن شامل `feature` و `label` را تعریف می‌کنیم. همچنین یک متد به نام `children` را برای افزودن گره‌های فرزند در همین کلاس ایجاد می‌کنیم. با استفاده از تابع `repr` نحوه نمایش اطلاعات خروجی درخت را به شکل مطلوب تعریف می‌کنیم، به طوری که هر گره از درخت شامل ویژگی مورد استفاده از آن و سایر گره‌های زیرمجموعه‌اش نمایش داده شود. در نهایت، با استفاده از تابع بازگشتی `make_tree`، ساختار درخت را تکمیل می‌کنیم. در هر مرحله از اجرای درخت، بررسی می‌شود که آیا درخت حاوی یک گره برگ (`Leaf`) شده است یا خیر؛ زیرا اگر یکی از این دو شرط رخ دهد، به انتهای شاخه‌ی آن رسیده‌ایم و درخت توانایی تصمیم‌گیری دارد. این بررسی با کد زیر آغاز می‌شود.

سپس، در صورتی که شرط رخ نداده باشد، گره‌های تصمیم‌گیری را تشکیل می‌دهیم. برای این منظور، از تابع بهره اطلاعات و آنتروپی به عنوان معیارهای انتخاب ویژگی استفاده می‌کنیم تا گره‌ها بر اساس این معیارها ایجاد شوند. پس از محاسبه بهره اطلاعات (IG)، با استفاده از روش "Search Greedy"، ویژگی با بیشترین بهره اطلاعات را انتخاب کرده و آن ویژگی را برای گره انتخاب می‌کنیم. در ادامه، داخل این گره، همان‌طور که گفته شد، یک زیرمجموعه ایجاد می‌شود و تصمیم‌گیری با استفاده از ویژگی‌های باقی‌مانده به همین شکل انجام می‌شود تا به گره پیش‌بینی برسیم. نکته مهم در این تابع این است که ابتدا هر شاخه درخت را به قدر ادامه می‌دهیم تا به گره پیش‌بینی برسیم و سپس به مرحله قبلی بازمی‌گردیم و همین عمل را تکرار می‌کنیم تا

در نهایت درخت کامل شود. در انتها، درخت را با ورودی `df` به عنوان دیتافریم شامل داده‌ها و `Infected` به عنوان خروجی و برچسب‌های داده‌ها تشکیل می‌دهیم.

```
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}")'
```

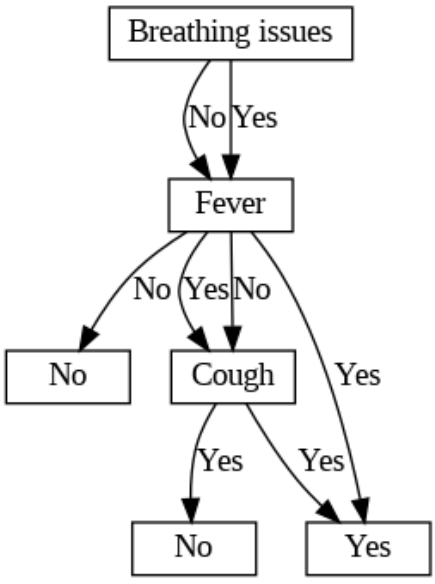
```
def make_tree(data, target):

  if (len(data[target].unique()) == 1 or len(data.columns) == 1):
    return Node(label = data[target].iloc[0])

  # Calculate IG
  features = data.drop(target, axis=1).columns
  gains = [information_gain(data, feature, target) for feature in features]

  # Greedy Search for findinf Best Feature
  max_gain_idx = np.argmax(gains)
  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)
    display(subset)

    node.children[value] = make_tree(subset, target)

  return node
```

خروجی به شکل زیر میباشد:

```
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")})})})
```

برای رسم درخت تصمیم از کد زیر استفاده میکنیم:

```python
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)
```



**تحلیل درخت تصمیم:**

با توجه به بالاترین information gain گره تصمیم اول breathing issues انتخاب می‌شود.

| | Fever | Cough | Infected |
|---|---|---|---|
| 0 | No | No | No |
| 2 | Yes | Yes | No |
| 5 | No | Yes | No |
| 9 | Yes | Yes | Yes |
| 10 | No | Yes | No |
| 13 | Yes | Yes | No |

سپس ویژگی fever برای گره تصمیم انتخاب میشود. در صورت no بودن این ویژگی درخت به یک leaf node می رسد.

| | Cough | Infected |
|---|---|---|
| 0 | No | No |
| 5 | Yes | No |
| 10 | Yes | No |

در صورت yes بودن descision node به صورت زیر و با ویژگی cough ادامه می یابد.

| | Cough | Infected |
|---|---|---|
| 2 | Yes | No |
| 9 | Yes | Yes |
| 13 | Yes | No |

| | Infected |
|---|---|
| 2 | No |
| 9 | Yes |
| 13 | No |

در صورت yes بودن گره ابتدایی درخت به صورت زیر نمایش داده می شود.

| | Fever | Cough | Infected |
|---|---|---|---|
| 1 | Yes | Yes | Yes |
| 3 | Yes | No | Yes |
| 4 | Yes | Yes | Yes |
| 6 | Yes | No | Yes |
| 7 | Yes | No | Yes |
| 8 | No | Yes | Yes |
| 11 | No | Yes | Yes |
| 12 | No | Yes | No |

ویژگی بعد fever است که در صورت yes بودن درخت به شکل زیر در می اید.

| | Cough | Infected |
|---|---|---|
| 1 | Yes | Yes |
| 3 | No | Yes |
| 4 | Yes | Yes |
| 6 | No | Yes |
| 7 | No | Yes |

برای ویژگی cough نیز ادامه درخت به شکل زیر میشود.

| | Cough | Infected |
|---|---|---|
| 1 | Yes | Yes |
| 3 | No | Yes |
| 4 | Yes | Yes |
| 6 | No | Yes |
| 7 | No | Yes |

| | Infected |
|---|---|
| 8 | Yes |
| 11 | Yes |
| 12 | No |

**بخش دوم:**

در این بخش دیتاست drugs را انتخاب میکنیم.

کتابخانه های مورد نظر را import میکنیم و دیتا ست را لود میکنیم

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

from sklearn.model_selection import train_test_split
from sklearn import tree
```

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1reEl49aLUDMe6cjKzbpJ1sxuE_ySw6DS
```

دیتاست به شکل زیر نمایش داده می شود.

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

| | Age | Sex | BP | Cholesterol | Na_to_K | Drug |
|---|---|---|---|---|---|---|
| 0 | 23 | F | HIGH | HIGH | 25.355 | drugY |
| 1 | 47 | M | LOW | HIGH | 13.093 | drugC |
| 2 | 47 | M | LOW | HIGH | 10.114 | drugC |
| 3 | 28 | F | NORMAL | HIGH | 7.798 | drugX |
| 4 | 61 | F | LOW | HIGH | 18.043 | drugY |
| ... | ... | ... | ... | ... | ... | ... |
| 195 | 56 | F | LOW | HIGH | 11.567 | drugC |
| 196 | 16 | M | LOW | HIGH | 12.006 | drugC |
| 197 | 52 | M | NORMAL | HIGH | 9.894 | drugX |
| 198 | 23 | M | NORMAL | NORMAL | 14.020 | drugX |
| 199 | 40 | F | LOW | NORMAL | 11.349 | drugX |

برای خروجی Drug باید به هر دارو یک عدد نسبت دهیم چون این ستون تنها شامل خروجی است.

برای این کار از متد replace() استفاده می کنیم و به ترتیب از ۰ تا ۴ به DrugA تا DrugYنسبت می دهیم.

سپس با متد pop(). ابتدا ستون Drug را حذف کرده و سپس با insert(). آن را به عنوان آخرین ستون به دیتافریم df2 اضافه می کنیم و در ادامه با آن کار می کنیم.

```
df1 = pd.get_dummies(df, columns=['Sex', 'BP', 'Cholesterol'], drop_first = True)
df2 = df1.replace({'Drug' : {'drugA': 0, 'drugB': 1, 'drugC': 2, 'drugX': 3, 'drugY':4 }})

col = df2.pop('Drug')
df2.insert(len(df2.columns), 'Drug', col)
df2
```

| | Age | Na_to_K | Sex_M | BP_LOW | BP_NORMAL | Cholesterol_NORMAL | Drug |
|---|---|---|---|---|---|---|---|
| 0 | 23 | 25.355 | 0 | 0 | 0 | 0 | 4 |
| 1 | 47 | 13.093 | 1 | 1 | 0 | 0 | 2 |
| 2 | 47 | 10.114 | 1 | 1 | 0 | 0 | 2 |
| 3 | 28 | 7.798 | 0 | 0 | 1 | 0 | 3 |
| 4 | 61 | 18.043 | 0 | 1 | 0 | 0 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 195 | 56 | 11.567 | 0 | 1 | 0 | 0 | 2 |
| 196 | 16 | 12.006 | 1 | 1 | 0 | 0 | 2 |
| 197 | 52 | 9.894 | 1 | 0 | 1 | 0 | 3 |
| 198 | 23 | 14.020 | 1 | 0 | 1 | 1 | 3 |
| 199 | 40 | 11.349 | 0 | 1 | 0 | 1 | 3 |

در ابتدا، با استفاده از متد `.iloc()`، داده‌ها را به `X` اختصاص می‌دهیم و داده‌های ستون آخر را به عنوان خروجی در `Y` قرار می‌دهیم. سپس با بهره‌گیری از `train_test_split` و نسبت ۸۵ به ۱۵ درصد، داده‌ها را به دو دسته آموزشی و ارزیابی تقسیم می‌کنیم.

برای تعریف مدل از دستور آماده `tree.DecisionTreeClassifier` استفاده می‌کنیم. در اینجا، `max_depth` به ۳ تنظیم شده و پارامتر مرتبط با هرس کردن یعنی `ccp_alpha` ابتدا با مقدار صفر مقداردهی می‌شود. این پارامتر با حذف گره‌هایی که تا دقت خوبی داده‌ها را جدا کرده‌اند، از افراز مدل بر روی داده‌های آموزشی جلوگیری می‌کند و همچنین با کاهش مقدار محاسبات، کارایی مدل بهبود می‌یابد. برای ایجاد یک درخت ثابت در هر اجرا، `random_state=83` قرار می‌دهیم. سپس مدل را بر داده‌های آموزشی fit می‌کنیم و با استفاده از `plot_tree`، ساختار درخت را نمایش می‌دهیم. برای داده‌های ارزیابی، می‌توان با متد `.predict` خروجی را دریافت کرد و با استفاده از `.score`، دقت مدل را بر روی داده‌های آموزشی و ارزیابی اندازه‌گیری نمود.

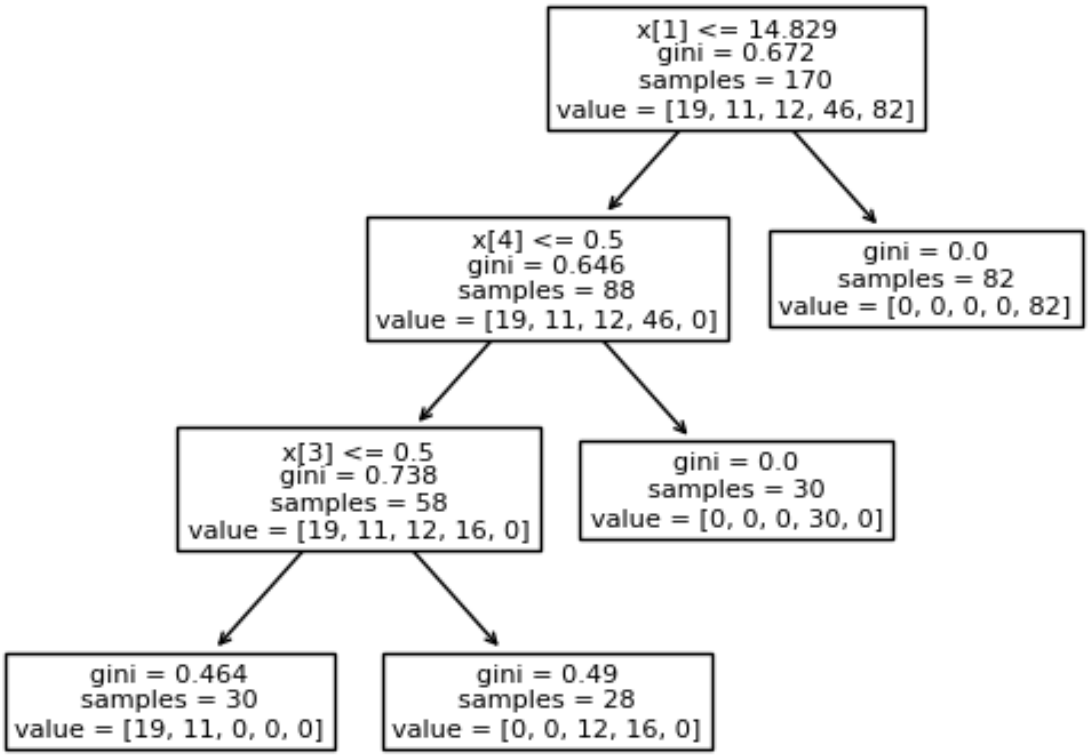
نتیجتاً با توجه به دقت‌های به دست آمده، ضروری است تا پارامترهای مدل را بهینه‌سازی کنیم تا به دقت بهتری دست یابیم.

```
X = df2.iloc[:,:-1]
y = df2.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.15, shuffle = True, random_state=83)
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

```
(170, 6) (170,) (30, 6) (30,)
```

```
model = tree.DecisionTreeClassifier(random_state = 83, max_depth = 3)
model.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=83)
```




دقت مدل به صورت زیر محاسبه می شود

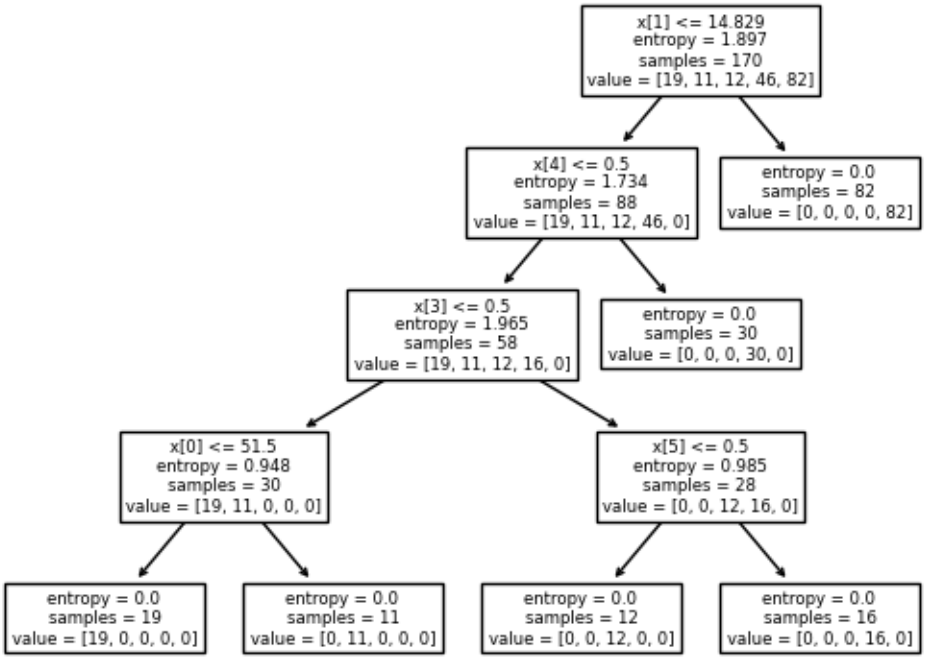
```
model.predict(x_test)
s1 = model.score(x_train, y_train)
s2 = model.score(x_test, y_test)

print(f"Train Accuracy: {s1}")
print(f"Test Accuracy: {s2}")
```

```
Train Accuracy: 0.8647058823529412
Test Accuracy: 0.7
```

حال متد را به entropy و depth را به ۴ تغییر میدهیم و ccp alpha را روی ۰.۰۰۱ قرار میدهیم

```
model2 = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth = 4, ccp_alpha = 0.1, random_state = 83)
model2.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.1, criterion='entropy', max_depth=4,
                       random_state=83)
```




```
s1 = model2.score(x_train, y_train)
s2 = model2.score(x_test, y_test)

print(f"Train Accuracy: {s1}")
print(f"Test Accuracy: {s2}")
```

```
Train Accuracy: 1.0
Test Accuracy: 0.9666666666666667
```

با تغییر پارامتر هرس مشاهده می شود مقادیر پایین تر از ۰.۱۶ اثری بر درخت نمیگذارند. ولی با تعیین ccp alpha برابر با ۰.۱۷ دقت به صورت قابل توجهی کاهش می یابد.

```
s1 = model2.score(x_train, y_train)
s2 = model2.score(x_test, y_test)

print(f"Train Accuracy: {s1}")
print(f"Test Accuracy: {s2}")
```

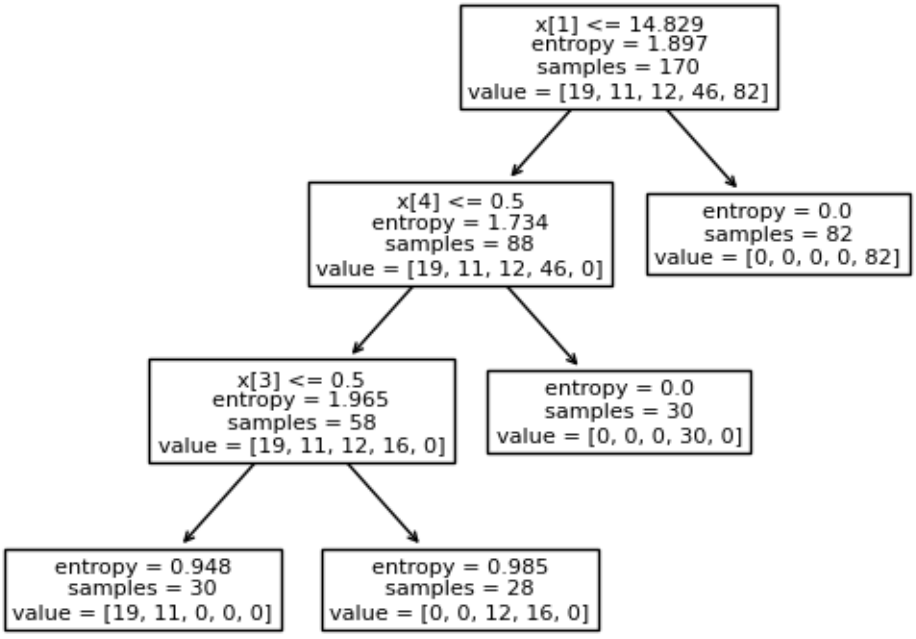
```
Train Accuracy: 0.8647058823529412
Test Accuracy: 0.7
```

تحلیل درخت:

مدل ۲ را بررسی می نماییم

```
txt = tree.export_text(model2)
txt
```

```
|--- feature_1 <= 14.83\n|   |--- feature_4 <= 0.50\n|   |   |--- feature_3
<= 0.50\n|   |   |   |--- feature_0 <= 51.50\n|   |   |   |   |--- class:
0\n|   |   |   |--- feature_0 >  51.50\n|   |   |   |   |--- class: 1\n|   |
|--- feature_3 >  0.50\n|   |   |   |--- feature_5 <= 0.50\n|   |   |   |
|--- class: 2\n|   |   |   |--- feature_5 >  0.50\n|   |   |   |   |---
class: 3\n|   |--- feature_4 >  0.50\n|   |   |--- class: 3\n|--- feature_1 >
14.83\n|   |--- class: 4\n
```




بر اساس ساختار این درخت تصمیم، ابتدا تصمیم‌گیری با توجه به feature 1 انجام شده است و مقدار عددی مربوط به آن ویژگی با توجه به محاسبات در داخل درخت به دست آمده است. در متن، مسیر یک شاخه به صورت کامل پیمایش شده و سپس به شاخه دیگر منتقل شده است. ابتدا شرطی بر اساس کمترین مقدار feature 1 بررسی شده است و در این حالت، به feature 3 رسیده‌ایم. با اعمال شرطی که بر اساس کمترین مقدار این ویژگی انجام شده است، از feature 0 استفاده می‌شود تا دو کلاس با مقادیر ۰ و ۱ پیش‌بینی شوند. سپس در متن، به گره قبلی بازگشته و شرطی بر اساس بیشترین مقدار feature 3 را با استفاده از feature5 بررسی کرده و کلاس‌های ۲ و ۳ را پیش‌بینی کرده‌اند. شرط بیشترین مقدار برای feature 4 ، ادامه کلاس ۳ را تعیین می‌کند و شرط بیشترین مقدار برای گره ابتدایی، کلاس ۱ را مشخص می‌سازد.

مسیر دو نمونه از دیتاست:

ابتدا کلاس حقیقی و سپس کلاس پیش بینی شده داده تست را نمایش میدهیم

و در ادامه با استفاده از متد گفته شده، مسیر گره ها را مشخص می کنیم.

```
#First sample
i = 5
print(f"x_example: {x_test.iloc[i]}")
print(f"\ny_example: {y_test.iloc[i]}")

p1 = model2.predict(x_test.iloc[[i]])
print(f"\nPrediction: {p1}")

path = model2.decision_path(x_test.iloc[[i]])
print(f"Path: {path.toarray()}")
```

```
x_example: Age                      59.000
Na_to_K                 10.444
Sex_M                    0.000
BP_LOW                   1.000
BP_NORMAL                0.000
Cholesterol_NORMAL       0.000
Name: 158, dtype: float64

y_example: 2

Prediction: [3]
Path: [[1 1 1 0 1 0 0]]
```

```python
#Second sample
i = 10
print(f"x_example: {x_test.iloc[i]}")
print(f"\ny_example: {y_test.iloc[i]}")

p1 = model2.predict(x_test.iloc[[i]])
print(f"\nPrediction: {p1}")

path = model2.decision_path(x_test.iloc[[i]])
print(f"Path: {path.toarray()}")
```

```
x_example: Age                    37.000
Na_to_K                23.091
Sex_M                   0.000
BP_LOW                  0.000
BP_NORMAL               0.000
Cholesterol_NORMAL      1.000
Name: 88, dtype: float64

y_example: 4

Prediction: [4]
Path: [[1 0 0 0 0 0 1]]
```

**بخش سوم:**

اگرچه تاکنون تحقیقات زیادی در زمینه عوامل موثر بر امید به 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 اندازه کافی مورد توجه قرار نگرفته‌اند. همچنین، برخی از تحقیقات گذشته بر اساس رگرسیون خطی چندگانه بر روی مجموعه داده یک ساله برای تمام کشورها انجام شده است. بنابراین، این امر انگیزه می‌دهد تا با فرموله کردن یک مدل رگرسیون بر اساس مدل اثرات مختلط و رگرسیون خطی چندگانه، هر دو عامل مذکور را حل کنیم، در حالی که از داده‌ها در بازه زمانی ۲۰۰۰ تا ۲۰۱۵ برای تمام کشورها استفاده می‌کنیم.
واکسیناسیون‌های مهم مانند هپاتیت B، پلیو و دیفتریا نیز در نظر گرفته می‌شوند. به طور خلاصه، این مطالعه بر عوامل واکسیناسیون، عوامل مرگ و میر، عوامل اقتصادی، عوامل اجتماعی و سایر عوامل مرتبط با سلامت تمرکز خواهد داشت. از آنجایی که مشاهدات این مجموعه داده مربوط به کشورهای مختلف هستند، برای یک کشور بهتر است که عامل پیش‌بینی‌کننده که به کاهش امید به زندگی کمک می‌کند را تعیین کند. این به کشور کمک می‌کند تا به بهبود بهره‌وری امید به زندگی جمعیت خود بپردازد.

این پروژه بر اطلاعات دقیق داده‌ها تکیه دارد. مخزن داده‌های Global Health Observatory (GHO) تحت مجموعه داده جهانی سازمان بهداشت جهانی (WHO)، وضعیت سلامت و عوامل مرتبط دیگر برای تمام کشورها را پایش می‌کند. این مجموعه داده‌ها به عنوان اطلاعات بهداشت به عموم افراز می‌شوند تا برای تحلیل داده‌های بهداشت مورد استفاده قرار گیرند. مجموعه داده مربوط به امید به زندگی و

عوامل بهداشتی برای ۱۹۳ کشور از همین وب‌سایت مخزن داده WHO جمع‌آوری شده و داده اقتصادی مرتبط آن از وب‌سایت سازمان ملل متحد جمع‌آوری شده است. از بین همه دسته‌های عوامل مرتبط با سلامت، فقط آن عوامل حیاتی انتخاب شده‌اند که نمایان‌تر هستند. مشاهده شده است که در ۱۵ سال گذشته، توسعه زیادی در بخش بهداشت رخ داده است که منجر به بهبود نرخ مرگ و میر انسانی، به ویژه در کشورهای در حال توسعه نسبت به ۳۰ سال گذشته شده است. بنابراین، در این پروژه، ما از داده‌ها از سال ۲۰۰۰ تا ۲۰۱۵ برای ۱۹۳ کشور برای تحلیل بیشتر استفاده کرده‌ایم. فایل‌های داده فردی با یکدیگر به یک مجموعه داده ترکیب شده‌اند. در بازدید اولیه از داده‌ها، برخی از مقادیر ناقص مشاهده شد. زیرا داده‌ها از WHO بودند، هیچ خطای آشکاری یافت نشد. داده‌های ناقص در نرم‌افزار R با استفاده از دستور Missmap پردازش شدند. نتیجه نشان داد که بیشتر داده‌های ناقص مربوط به جمعیت، هپاتیت B و GDP بود. این داده‌های ناقص از کشورهای کمتر شناخته شده مانند وانواتو، تونگا، توگو، کیپ ورد و غیره بودند. پیدا کردن تمام داده‌ها برای این کشورها دشوار بود و بنابراین تصمیم گرفته شد که این کشورها را از مجموعه داده نهایی حذف کنیم. فایل ترکیب شده نهایی (مجموعه داده نهایی) شامل ۲۲ ستون و ۲۹۳۸ ردیف بود که به این معناست که ۲۰ متغیر پیش‌بینی‌کننده وجود دارد. تمام متغیرهای پیش‌بینی‌کننده سپس به چندین دسته گسترده تقسیم شدند: عوامل واکسیناسیون مرتبط، عوامل مرگ و میر، عوامل اقتصادی و عوامل اجتماعی.

```
[64] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn import tree
```

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1VEyd3_LKC7vb4wuT5BQwFVprTgsZ8lLn
```

تعداد داده های nan را پیدا کرده و سپس با dropna() آن ها را حذف میکنیم.

```
df = pd.read_csv('/content/Life Expectancy Data.csv')
df.isnull().sum()
```

```
df1 = df.dropna()
print(df1.isnull().sum())
df1
```

سپس ابتدا با متد drop(). ستون مربوط به امید به زندگی را از df1 حذف کرده و باقی داده ها را داخل df2 قرار می دهیم. برای حذف داده های توصیفی مشابه قبل از get_dummies. استفاده می کنیم تا با one-hot encoding آن ها را به صورت اعداد درآوریم. سپس تمام داده های df2 را داخل X ریخته و برای Y از دیتافریم df1 تنها ستون مربوط به امید به زندگی ستون 3 را داخل آن قرار میدهیم.

به کمک train_test_split داده ها را با نسبت 80 به 20 به آموزش و تست تقسیم می کنیم.

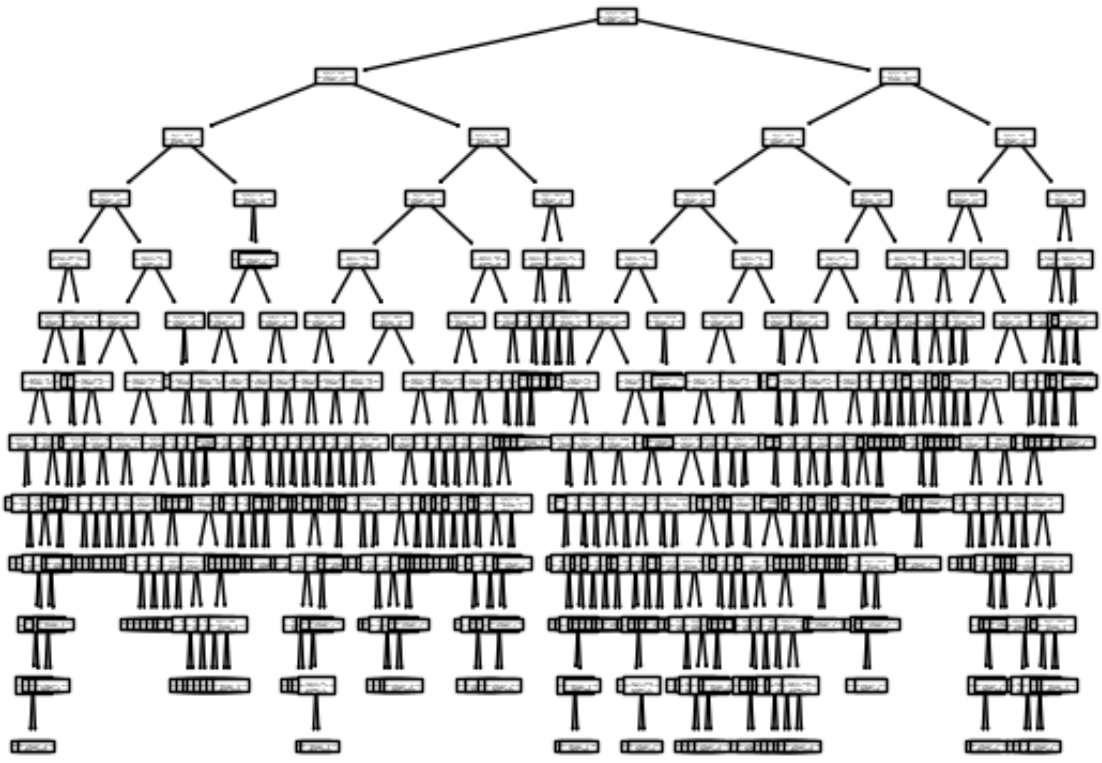
```
df2 = df1.drop( labels = 'Life expectancy ', axis = 1)
df2 = pd.get_dummies(df2, columns=['Country', 'Status'], drop_first = True)
X = df2.iloc[: , :]

Y = df1.iloc[:,3]

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state=83)
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

```
(1319, 152) (1319,) (330, 152) (330,)
```

سپس درخت تصمیم را ایجاد کرده و هایپرپارامتر های آن را تنظیم میکنیم تا دقت مطلوب حاصل شود.

```
model1 = tree.DecisionTreeRegressor(max_depth = 12, ccp_alpha=0.001, random_state=83)
model1.fit(x_train, y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.001, max_depth=12, random_state=83)
```



دقت به صورت زیر نمایش داده می شود:

```
s1 = model1.score(x_train, y_train)
s2 = model1.score(x_test, y_test)

print(f"Train Accuracy: {s1}")
print(f"Test Accuracy: {s2}")
```

```
Train Accuracy: 0.9912685853458939
Test Accuracy: 0.9278626919259654
```

**سوال پنجم:**

PSNR به عنوان اختصار "نسبت سیگنال به نویز بیشینه" شناخته می‌شود و یک معیار است که برای اندازه‌گیری کیفیت تصاویر و ویدئوها مورد استفاده قرار می‌گیرد. این معیار عمدتاً در زمینه‌های پردازش تصویر، فشرده‌سازی تصویر و انتقال تصاویر به کار می‌رود.

فرمول PSNR به صورت زیر است:

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

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

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

۲. ناپیوستگی PSNR: این معیار نمی‌تواند تغییراتی که به صورت غیرخطی در تصویر رخ می‌دهند را به خوبی ارزیابی کند؛ به عبارت دیگر، 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 شباهت ساختاری (SSIM) یا PSNR-HVS (نسبت سیگنال به نویز بیشینه با در نظر گرفتن سیستم بینایی انسان) باشد.