تمرین شماره 2 درس سیستم های هوشمند

پوریا ازادی مقدم

810193331

سوال 1

الف)

همان طور که از دیتاست داده شده مشخص میباشد ,این داده ها مربوط به حروف الفبا می باشد در نتیجه دارای 26 کلاس خواهیم بود.

همچنین 16 ویژگی هر داده دارد که این داده ها مقادیری از 0 تا 15 به خود اختصاص داده اند.

در نتیجه برای ساخت درخت چندین راه وجود دارد از جمله:

روش اول انکه برای هر نود 16 بچه بالقوه در نظر بگیریم زیرا مقدار هر ویژگی مقادیر گسسته از 0 تا 15 می باشد.

روش دوم انکه با استفاده از روش هایی چون میانگین گیری , برای هر ویژگی ترشهولد ای قرار دهیم و هر نود دیگر 16 فرزند نداشته باشد بلکه 2 فرزند داشته باشد به این نحوه که مقدار ویژگی بزرگتر از ترشهولد است یا کوچک تر.

رئش دوم دارای دقت بسیار کمتری از روش اول خواهد بود اما باعث کوچکی درخت خواهد شد .

در مجموع ما از روش اول برای ساخت درخت ها استفاده نمودیم.

همچنین با توجه به این که 26 کلاس داریم عملا من از روش O vs ALL استفاده کردم به این صورت که 26 درخت که هر کدام ادعا میکند که ایا داده تست عضو ان هست یا نه.

در اکثر مواقع تنها یک درخت همچین ادعایی خواهد داشت در مواردی که دو یا چند درخت همچین ادعایی داشته باشند درختی که با جمعیت بیشتری راجب ان دیتا تصمیم گیری کرده باشد یعنی تعداد جمعیت موجود در ان نود لیبل بیشتر باشد , به عنوان کلاس داده انتخاب خواهد شد.

همان طور که مشخص است و از ویژگی های درخت نیز محسوب میشود, هر زیر مجموعه نود های درخت , خود نیز درختی میباشد, این عامل مارا به سمت استفاده از تابعی بازگشتی جهت ساخت درخت سوق میدهد:

```
function [outs]=make_tree(attributes,datas,class,level,column)
```

ارگومان های ورودی این تابع به ترتیب عبارتند از:

Attributes که ویژگی های باقی مانده تا این لحظه برای نود میباشند که ان نود باید بهترین را انتخاب نماید.

Datas دیتا های ترین ورودی به نود مباشد.

Class کلاسی که درخت را برای ان میسازیم.

Level لایه که نود در ان قرار دارد.

Column که ستونی که این نود قرار دارد در حالتی که درخت کامل بود را مشخص خواهد کرد.

یکی از دغدغه های ساخت درخت,نحوه نگه داری نود های ان میباشد, بدین منظور هر نود اطلاعاتی که از بچه های خود گرفته است را به ترتیب به هم می چسباند و اطلاعات خود را نیز در ابتدای ان قرار خواهد داد و در ادامه به نود پدر خود ان ها را پاس خواهد داد.

این روند ادامه پیدا میکند تا تمامی اطلاعات به هم پیوسته به بالاترین نود خواهد رسید.

منظور از اطلاعات هر نود در بالا این پکت زیر شامل 4 عدد زیر خواهد بود:

Level----column-----attribute or label-----number of data in node

از به هم پیوستن پکت های 4 تایی بالا , درخت مورد نظر ساخته خواهد شد.

شرط توقف این الگوریتم را آن قرار دادم که لیبل تمام داده های ورودی به نود true یا false باشند. در این صورت مشخص است که به برگ رسیده ایم وفرزندی دیگر وجود ندارد:

```
if(t==0)
    f=f;
    outs=[level,column,-1,f];
    return
end
if(f==0)
    t=t;
    outs=[level,column,-2,t];
    return
end
```

در این قطعه کد, t و f نشان دهنده تعداد داده های ورودی با لیبل true و false می باشند.

برای تشخیص انکه این برگ برای کلاس مورد نظر درست است یا غلط از 1- یا 2- استفاده کردم.

در این مرحله از Information gain استفاده خواهیم کرد که عبارت است از:

$$E(S)-\sum_{i}\frac{|S_i|}{|S|}\cdot E(S_i)$$

یعنی هر نود باید بر اساس بزرگترین IG بهترین ویژگی را برای پرسش انتخاب نماید.

در قطعه کد زیر محاسبه جمعیت , تعداد داده ها با لیبل true و لیبل false در فرزندان نود به ازای انتخاب هر کدام از ویژگی ها محاسبه خواهد شد:

```
 %calculating the population of each child in the case of choosing each atribute
 %also calculating the true and false labels in each child
for atts=1:1:length(attributes)
   for possibles=1:1:16
       for data=1:1:length(datas(:,1))
           if(datas(data,atts)==(possibles-1))
               number_of_childs(atts, possibles)=number_of_childs(atts, possibles)+1;
               if(datas(data,17)==class)
                   ff=trues(atts, possibles)+1;
                   trues(atts, possibles)=ff;
               end
               if(datas(data,17)~=class)
                   ff=falses(atts, possibles)+1;
                   falses(atts, possibles)=ff;
               end
           end
       end
   end
end
```

در این بخش براساس یافته های بالا IG به ازای هر نود محاسبه خواهد شد:

```
%calculating information gain for each attribute
gain(1)=1;
for atts=1:1:length(attributes)
    c=0.0;
   for possibles=1:1:16
       if((trues(atts,possibles)+falses(atts,possibles)) ~= 0)
           i1=number_of_childs(atts, possibles)/(t+f);
           i2=(trues(atts,possibles))/(trues(atts,possibles)+falses(atts,possibles));
           i3=(falses(atts,possibles))/(trues(atts,possibles)+falses(atts,possibles));
           if(i2==0 || i3==0 )
               c=c;
           else
            c=c+i1*(-1*i2*(log2(i2))+(-1*i3)*log2(i3));
           end
       end
   end
   gain(atts)=Et-c;
end
```

در ادامه انتخاب ویژگی با بزرگترین attributes و همچنین ساختن بردار ویژگی ها بعد از حذف این ویژگی برای پاس دادن ب نود های بعدی را مشاهده می نمایید:

```
%choose the attribute with biggest IG
[~,max_index]=max(gain);

%update attributes to pass children
if(max_index==1)
    new_attributes=attributes(2:(length(attributes)));
elseif(max_index==length(attributes))
    new_attributes=attributes(1:(length(attributes)-1));
else
    new_attributes=attributes([1:(max_index-1),(max_index+1):(length(attributes))]);
end
```

در این بخش جدا کردن داده های هر فرزند جهت پاس دادن به ان ها را مشاهده می نمایید:

```
%now we call function for each child
next=[];
for possibles=1:1:16
    c=1;
    clear childs_data;
    for data=1:1:length(datas(:,1))
        if(datas(data,attributes(max_index))==possibles-1)
                childs_data(c,:)=datas(data,:);
                c=c+1;
        end
    end
```

در نهایت تمامی این داده ها به نود بالایی پاس داده خواهند شد:

```
    %in this part we paste outs of each child to each other
    if(c~=1)
        a=next;
        next=[a,make_tree(new_attributes,childs_data,class,level+1,16*(column-1)+possibles)]
    end
end
o=[level,column,attributes(max_index),-10];
outs=[o,next];
```

لیبل های لایه ۱ و دو درخت های ۱ تا ۶:

توجه شود ۱- به معنای رسیدن به برگی است که لیبل ان false برای ان کلاس می باشد:

Class 1:

11

1- 1- 1- 5 8 9 5 7 7 9 9 9 6

1- 1- 1-

Class 2:

15

14 13 8 8 13 9 12 12 12 2 12 1- 1-

1- 1- 1-

Class 3:

12

7 15 10 2 8 8 8 1- 1- 1- 1- 1- 1-

16 16 16

Class 4:

12

1- 10 2 14 14 14 8 7 7 15 7 1- 1-

1- 1- 1-

Class 5:

12

2 16 16 10 10 5 1- 1- 1- 1- 1- 1- 1-

16 16 14

Class 6:

7

15 11 10 13 13 15 15 1- 1- 1- 1- 1- 1-

6 6 6

Confusion matrix به همان نحوی که تی ای مربوطه در کلاس توجیهی توضیح دادند در زیر اورده شده است:

ماتریس مربوطه را در فایل های ارسالی به اسم confusion _part1 میتوانیدمشاهده نمایید.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	125	0	0	0	0	0	0	0	0	0	0	1	0	3	0	0	2	1	0	0	0	0	0	0	0	0
2	0	84	0	3	3	2	0	1	0	2	1	0	1	0	2	1	0	7	4	1	1	1	0	3	1	2
3	0	0	102	0	1	0	1	1	1	0	2	1	0	0	2	0	1	0	0	0	0	0	0	0	0	0
4	0	1	0	98	0	2	0	1	0	0	1	0	1	1	1	3	2	4	3	0	0	0	0	2	0	1
5	1	0	4	0	99	0	0	1	1	0	4	0	2	0	1	0	1	0	2	1	0	0	0	2	0	3
6	0	0	0	0	0	88	1	1	0	0	0	1	0	0	1	10	0	2	2	1	0	3	0	0	1	1
7	2	2	2	0	0	1	87	0	0	0	1	0	0	0	1	1	3	2	0	1	2	2	2	0	0	1
8	0	2	0	6	1	1	2	85	0	0	4	1	0	2	0	3	2	6	1	0	0	2	0	0	2	1
9	0	0	0	0	1	1	0	0	136	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	2	0	0	1	2	1	0	0	4	115	0	0	0	0	0	0	2	1	1	0	0	0	0	0	0	0
11	0	1	2	0	4	2	1	2	0	1	89	0	0	1	0	0	1	6	1	0	0	1	0	2	1	1
12	3	1	0	0	0	0	1	2	0	0	0	118	1	1	2	0	1	0	3	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	1	0	0	1	0	112	1	0	0	0	0	0	0	1	1	2	0	1	0
14	1	2	0	1	0	2	0	4	0	0	1	1	3	105	2	1	0	1	0	0	2	0	4	0	1	0
15	0	0	1	3	1	0	4	4	0	0	0	0	1	0	80	0	16	1	1	0	4	1	0	0	0	2
16	0	1	0	0	1	3	0	0	2	0	0	0	0	1	1	95	1	0	0	2	0	1	0	0	3	0
17	1	3	2	0	2	0	2	1	0	2	1	2	0	0	5	0	80	1	2	1	6	2	0	8	1	3
18	0	2	0	3	3	0	1	3	0	4	6	1	0	1	0	0	0	92	2	0	0	0	0	2	0	1
19	2	0	1	2	1	0	3	0	0	0	1	0	0	0	2	1	0	1	99	0	0	0	0	2	0	1
20	1	1	0	0	1	5	0	0	0	2	1	1	0	1	0	1	0	0	1	91	3	3	0	0	3	2
21	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	1	1	0	0	1	117	0	0	2	0	0
22	0	1	1	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	2	0	98	1	0	1	0
23	0	0	0	0	0	2	2	1	0	0	0	0	3	2	0	0	1	0	0	0	0	0	114	0	0	0
24	0	1	0	1	1	2	1	1	2	1	2	0	0	4	0	2	1	1	3	1	0	0	0	105	0	2
25	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	2	0	0	7	0	1	2	0	100	0
26	0	0	0	0	4	0	0	0	0	2	0	2	0	0	0	1	4	0	0	0	0	0	0	4	2	110

همچنین دقا این روش برابر است با :

```
accuracy =

    0.6563
```

سرعت اجرا این برنامه :

```
>> toc
Elapsed time is 36.464372 seconds.
```

ب)

در این بخش از معیار gini جهت ساخت درخت استفاده میکنیم.

این معیار به شکل زیر تعریف خواهد شد:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

در نتیجه به تغییرات اندکی در کد ساخت درخت خود نیاز داریم که در شکل زیر مشهود است:

```
for atts=1:1:length(attributes)
    c=0.0;
   for possibles=1:1:16
       if((trues(atts,possibles)+falses(atts,possibles)) ~= 0)
           i1=number_of_childs(atts, possibles)/(t+f);
           i2=(trues(atts,possibles))/(trues(atts,possibles)+falses(atts,possibles));
           i3=(falses(atts,possibles))/(trues(atts,possibles)+falses(atts,possibles));
            c=c+i1*(1-i2^2-i3^2);

       end
   end
   gain(atts)=c;
end
```

Class 1:

4

1- 1- 1- 1 11 11 11 11 11 1 1 8 1

1- 1- 1-

Class 2:

2

1 11 9 11 9 11 11 8 8 8 1 1 8

1 1 1

Class 3:

1

1- 1- 2 2 2 16 8 8 11 11 11 2 2

1- 1- 1-

Class 4:

2

1 11 6 11 11 11 11 8 10 8 1 1 8

1 1 1

Class 5:

2

1 6 12 14 8 12 12 8 8 8 1 1 8

1 1 1

Class 6:

4

1- 1- 1 1 9 9 9 9 12 1 1 8 1

1- 1- 1-

Confusion matrix در این جالت طبق گفته تی ای برابر است با:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	83	0	1	2	2	2	0	0	2	2	1	17	0	0	0	0	0	4	0	1	0	0	0	3	1	2
2	2	48	0	6	5	4	3	0	0	1	3	1	3	0	3	3	0	8	5	0	1	0	0	0	1	2
3	1	1	51	1	3	1	5	0	2	1	2	1	0	1	2	1	2	0	5	3	4	0	0	1	0	1
4	2	5	2	47	2	0	2	8	0	3	4	2	2	2	6	3	2	9	3	0	2	0	0	9	0	3
5	3	3	2	1	53	1	3	2	0	0	10	8	1	4	3	4	1	3	3	0	1	0	0	2	1	2
6	1	1	0	0	0	35	1	1	1	4	3	2	0	3	0	9	1	2	2	1	2	2	0	0	2	0
7	2	0	5	2	1	0	51	1	0	1	1	1	3	1	3	0	2	2	3	3	5	0	2	0	1	1
8	0	3	0	3	0	2	1	40	0	0	2	0	6	4	4	0	2	3	1	0	4	0	4	2	0	3
9	1	0	0	0	0	0	1	1	110	1	0	1	0	0	1	0	1	0	2	2	0	0	1	0	0	3
10	0	0	1	0	2	4	1	0	2	74	1	2	0	0	0	2	1	3	1	0	0	1	0	1	1	0
11	0	0	5	2	2	1	6	8	0	0	36	0	3	1	1	1	1	5	1	2	0	0	2	4	0	0
12	0	2	1	3	2	3	0	0	1	2	1	44	0	0	3	2	3	0	0	1	1	2	0	0	1	0
13	1	1	0	2	0	1	0	8	0	0	2	2	43	4	2	1	1	3	0	0	1	0	6	2	1	1
14	0	0	0	2	5	2	6	4	0	0	2	0	1	60	2	2	0	3	1	0	0	1	2	1	0	1
15	0	1	2	6	0	0	6	1	0	2	0	0	1	0	51	0	1	1	5	1	1	0	0	1	0	0
16	0	1	1	4	1	13	0	0	1	0	2	2	0	3	1	39	0	1	1	1	1	0	1	3	3	1
17	1	0	1	2	2	0	3	0	0	0	2	1	0	0	0	0	81	0	2	0	1	0	0	0	2	3
18	0	3	0	5	1	1	3	8	0	0	5	1	1	2	1	4	0	47	1	0	1	1	1	0	0	0
19	1	0	0	5	1	0	1	1	1	1	1	2	0	3	0	2	0	0	44	3	1	1	4	0	0	4
20	1	0	1	1	2	3	1	1	1	1	0	2	0	0	0	3	1	2	2	48	1	4	0	0	4	2
21	1	0	0	1	2	3	1	0	0	0	0	2	2	1	0	3	2	1	0	3	57	1	0	2	2	0
22	1	0	1	1	0	2	0	0	0	0	0	1	0	0	0	0	0	1	1	5	6	62	1	1	9	0
23	0	0	0	1	2	1	4	3	0	1	2	0	6	1	0	0	3	0	0	0	2	0	51	2	1	2
24	2	1	0	1	2	0	0	1	0	2	2	3	2	2	2	1	2	1	3	0	1	1	1	54	0	5
25	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	5	0	0	1	5	3	6	1	0	53	3
26	2	1	3	2	5	0	0	0	1	0	0	0	0	4	0	0	0	0	2	0	1	2	0	1	0	57

این ماتریس را در فایل های ارسالی به اسم confusion_part2 میتوانید مشاهده نمایید.

همان طور که میبینید دقت این روش برابر است با :

```
accuracy =

    0.3663
```

زمان اجرا این الگوریتم:

```
>> toc
Elapsed time is 122.605902 seconds.
```

این زمان اجرا بسیار بیشتر از روش قبلی است مه نشان دهنده توزیع بد و نامتقارن است.

نتیجه گیری:

اولین نکته ان که همان طور که به نظر میرسد دقت روش IG بسیار بهتر از دقت روش GINI خواهد بود.

همچنین باید توجه کرد که در بررسی ها این روش نسبت به روش قبل باعث افزایش عمق درخت میشود یعنی در حالت IG داده ها تجمع کمتری میابند و در نتیجه در لایه های بالاتر به برگ خواهد رسید.

همچنین این روش به علت عمق زیاد خود باعث اورفیت شدن خواهد شد.

ج)

در این بخش با قرار دادن شرط زیر:

```
if(length(attributes)==16)
    gain(max_index)=0;
    [~,max_index]=max(gain);
end
```

با قرار دادن این قطعه کد دومین ماکس را به جای اولین ماکس انتخاب خواهیم کرد در این صورت انتخاب ویژگی نود اول ما به ترتیب به شکل زیر خواهد بود:

14 11 13 6 8 8 15 10 15 9 6 12 7

11 13 9 14 13 11 10 14 8 14 15 15

15

با این کار \یش بینی ما ان است که شاهد کاهش دقت خواهیم بود زیرا در اولین انتخاب بر اساس بیشترین توزیع عمل نکرده ایم در نتیجه دقت کاهش خواهد یافت:

```
accuracy =

    0.6388
```

شکل بالا نیز نشان میدهد دقت کاهش یافته است امل زمان اجرا تفادت اندکی خواهد داشت:

```
>> toc
Elapsed time is 37.592039 seconds.
```

د)

در این مرحله از دو روش جهت جلوگیری از overfit شدن استفاده خواهد شد که توئمان استفاده میگردند.

این دو شرط را در تابع make_tree_haras میتوانید بیابید.

اولین مرحله ان است که شرطی به شکل زیر قرار خواهیم داد:

```
if(length(attributes)==11)
    if(t>=0.5*f)
        outs=[level,column,-2,t+f];
    else
        outs=[level,column,-1,t+f];
    end
    return
end
```

یعنی انکه تعداد لایه ها را محدود خواهیم کرد و بنای نتیجه گیری را تا این مرحله قرار خواهیم داد.

در دومین مرحله تعداد بچه هایی که جواب true میدهند و تعداد انهایی که جواب false را می شماریم و با شرط زیر خرجی را چک می نماییم:

```
if(tt>9)
    outs=[level,column,-2,t+f];
elseif(ff>12)
    outs=[level,column,-1,t+f];
else
    o=[level,column,attributes(max_index),-10];
    outs=[o,next];
end
```

در مورد سرعت اجرا و دقت داریم که:

```
Elapsed time is 28.591329 seconds.
>> truth=xx/4000

truth =

    0.6535
```

همان طور که میبینید سرعت اجرا حدود 2/3 برابر شده است اما دقت حدود 1 الی 2 درصد کاهش یافته است.

همان طور که انتظار داشتیم هرس کردن و شرط توقف اثر چندانی در دقت ما نداشتند زیرا از روش one vs all استفاده شده است در نتیجه با داشتن 26 درخت نسبت به نتیجه خود مقاوم شده و هرس کردن اثر چندانی ندارد.

هم

ه)

در این مرحله از روش Random Forest استفاده خواهیم کرد.

در این روش داده های ترین را به k قسمت تقسیم میکنیم, در ادامه برای هر داده تست k نتیجه مختلف خواهیم گرفت که به وسیله ان ها از طریق majority voting بهنتیجه گیری راجب کلاس داده تست خواهیم رسید.

از انجایی که ما ویژگی ها را تقسیم نکرده و تماما به داخل درخت فرستادیم , روش bagging را پیاده سازی کرده ایم.همچنین روش با تقسیم کردن ویژگی ها به دسته ریز تر نیز پیاده ساطی شده است اما جواب روش bagging بسیار بهتر بود. به نتایج روش عادی در انتها نیز اشاره شده است.

برای k مقادیر 2 و 4 و 8 مورد بررسی قرار خواهد گرفت:

K=2:

```
e1=[train_data(1:8000,:),train_labels(1:8000,:)];
e2=[train_data(8001:16000,:),train_labels(8001:16000,:)];
attributes=1:16;
```

تعداد نتایج صحیح:

```
xx =

        1999
```

دقت در این حالت برابر است با :

```
truth =

    0.4998
```

```
Elapsed time is 58.705229 seconds.
```

K=4:

```
e1=[train_data(1:4000,:),train_labels(1:4000,:)];
e2=[train_data(4001:8000,:),train_labels(4001:8000,:)];
e3=[train_data(8001:12000,:),train_labels(8001:12000,:)];
e4=[train_data(12001:16000,:),train_labels(12001:16000,:)];
attributes=1:16;
```

تعداد نتایج صحیح:

```
xx =

        2012
```

دقت در این حالت برابر است با :

```
truth =

    0.5030
```

K=8:

```
%deviding data and attributes to k part
e=[train_data(1:n_train,:),train_labels(1:n_train,:)];
e1=[train_data(1:2000,:),train_labels(1:2000,:)];
e2=[train_data(2001:4000,:),train_labels(2001:4000,:)];
e3=[train_data(4001:6000,:),train_labels(4001:6000,:)];
e4=[train_data(6001:8000,:),train_labels(6001:8000,:)];
e5=[train_data(8001:10000,:),train_labels(8001:10000,:)];
e6=[train_data(10001:12000,:),train_labels(10001:12000,:)];
e7=[train_data(12001:14000,:),train_labels(12001:14000,:)];
e8=[train_data(14001:16000,:),train_labels(14001:16000,:)];
attributes=1:16;
```

تعداد نتایج صحیح:

```
xx =

        1880
```

دقت در این حالت برابر است با :

```
truth =

    0.4700
```

نتیجه گیری:

همان طور که از اعداد بالا مشخص می باشد بهترین نتیجه گرفته شده برابر با 4 است هرچند 2 و 4 بسیار به یکدیگر نزدیک میباشند.

این نتیجه گیری با تعوری نیز همخوانی کامل دارد زیرا بر اساس تعوری بهترین گزینه برای k برابر با ریشه تعداد کلاس ها می بتشد که اینجا برابر با 5 است, پس نزدیک ترین عدد به این مقدار برابر با 4 است.

روش random forest با تقسیم کردن ویژگی ها:

در این حالت برای k=2 دقتی بسیار بیشتر از k=4بدست امد:

K=2

```
>> truth=xx/4000

truth =

    0.3300
```

X=4

```
>> truth=xx/4000

truth =

    0.1925
```

همان طور که مشاهده می شود روش bagging بسیار بهتر از روش عادی برای random forestجواب میدهد.

سوال 2

این سوال به قسمت 6 folded cross validation و 3 folded cross validation تقسیم خواهد شد, در ابتدا توضیحات بخش ابتدایی را مشاهده می نمایید:

الف)

از انجایی که به ازای k های برابر با 3 5 7 9 الگوریتم knn را اجرا باید کرد, در ابتدا برداری به اسم k جهت تکرار الگوریتم به ازا همه این مقادیر میسازیم.

در ابتدا برای این روش باید داده ها را به 6 دسته تصادفی تقسیم نماییم, جهت این کار برداری تصادفی از اعداد 1 تا 150 می سازیم و در فواصل 25 تایی , 6 بردار موردنظر را میسازیم.

توجه شود که باید به ازای هر k 6 بار تست اجرا خواهد شد 125 داده به عنوان دادهtrain و 25 داده باقی مانده را برای تست استفاده مینماییم.

در قطعه کد زیر 6 ایجاد 6 دسته بردار تست 25 تایی و بردار های train متناظر هر کدام را مشاهده می نمایید:

```
shuffled=randperm(150);
e=[data,labels];
test(1,1:25,:)=e(shuffled(1:25),:);
test(2,1:25,:)=e(shuffled(26:50),:);
test(3,1:25,:)=e(shuffled(51:75),:);
test(4,1:25,:)=e(shuffled(76:100),:);
test(5,1:25,:)=e(shuffled(101:125),:);
test(6,1:25,:)=e(shuffled(126:150),:);
train(1,1:125,:)=e(shuffled(26:150),:);
train(2,1:125,:)=e([shuffled(1:25),shuffled(51:150)],:);
train(3,1:125,:)=e([shuffled(1:50),shuffled(76:150)],:);
train(4,1:125,:)=e([shuffled(1:75),shuffled(101:150)],:);
train(5,1:125,:)=e([shuffled(1:100),shuffled(126:150)],:);
train(6,1:125,:)=e(shuffled(1:125),:);
```

روند این الگوریتم و نحوه پیاده سازی ان در ادامه قابل مشاهده میباشد:

- نحوه اجرا الگوریتم به این شکل است که در ابتدا فاصله داده های تِرین از تک تک اعضای تست پیدا خواهد شد که در این بخش از فاصله اقلیدسی برای این کار استفاده نمودیم.

تعریف فاصله اقلیدسی در زیر اورده شده است:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

در قطعه کد زیر پیاده سازی این بخش مشخص میباشد:

```
for i=1:1:25
    d=zeros(1,125);
    for j=1:1:125
        for index=1:1:4
            d(j)=((test(test_number,i,index)-train(test_number,j,index))).^2+d(j);
        end
        d(j)=d(j)^.5;
    end
```

- حال در ادامه k عضو داده های ترین که نزدیک ترین فاصله را دارند را پیدا نمودیم:

```
u=1:1:125;
r=[d',u'];
sorted_distance=sortrows(r,1);
k_nearest(1:k(c),:)=sorted_distance(1:k(c),:);
```

- در ادامه نیز کلاس غالب این نزدیک ترین داده ها را به عنوان کلاس غالب برای داده تست در نظر گرفته شد:

```
for t=1:1:k(c)
    if(train(test_number,k_nearest(t,2),5)==1)
        class1(i)=class1(i)+1;
    end
    if(train(test_number,k_nearest(t,2),5)==2)
        class2(i)=class2(i)+1;
    end
    if(train(test_number,k_nearest(t,2),5)==3)
        class3(i)=class3(i)+1;
    end
end
if(class1(i)>=class2(i) & class1(i)>=class3(i))
    choosed_class(i+25*(test_number-1))=1;
end
if(class2(i)>=class1(i) & class2(i)>=class3(i))
    choosed_class(i+25*(test_number-1))=2;
end
if(class3(i)>=class2(i) & class3(i)>=class1(i))
    choosed_class(i+25*(test_number-1))=3;
end
```

- تمامی مراحل بالا را 6 بار تکرار میکنیم کلاس تشخیص داده شده 150 داده مورد نظر را ذخیره کرده و با کلاس اصلی ان ها مقایسه خواهیم کرد:

```
for i=1:1:150
    if(choosed_class(i)==e(shuffled(i),5))
        g(c)=g(c)+1;
    end
end
```

توجه مهم: با توجه به انکه انتخاب 6 گروه داده ها به صورت تصادفی انجام میشود جهت نتیجه گیری صحیح در مورد بهترین مقدار برای k الگوریتم بالا را چندین بار تکرار کرده و جهت نتیجه گیری, میانگین تعداد صحیح تشخیص داده شده به ازای هر مرحله را گزارش خواهیم کرد:

```
for c=1:1:4
    truth(c)=g(c)/n_mean
end
```

حال نتیجه اجرا به ازای مقادیر 3 5 7 9 برای k را خواهیم داشت:

```
truth =

  144.1070  144.6400  144.8440  144.9180
```

همان طور که میبینید برای k برابر با 7 و 9 بهترین نتیجه بدست می اید که این دو مقدار به شدت به یکدیگر نزدیک خواهند بود, تصمیم گیری بین این دو مقدار با اندکی دشواری رو به رو می باشد.

ب)

در این بخش به ازای kبرابر با 9 به انجام الگوریتم ذکر شده در بالا پرداختیم اما از فاصله کسینوسی استفاده نمودیم.

فاصله کسینوسی به صورت زیر تعریف خواهد شد:

$$cosine\ distance\ (a,b) = 1 - \frac{a.b}{norm(a) * norm(b)}$$

حال با توجه به فرمول بالا مقدار فاصله توسط قطعه کد زیر محاسبه میشود:

```
for i=1:1:50
    d=zeros(1,100);
    first(i,:)=test(test_number,i,1:4);
    for j=1:1:100
        for index=1:1:4
            d(j)=((test(test_number,i,index)*train(test_number,j,index)))+d(j);
        end

        second(j,:)=train(test_number,j,1:4);
        d(j)=1-d(j)/(norm(first(i,:))*norm(second(j,:)));
    end
```

در نتیجه اجرا 100 بار این الگوریتم برای k برابر با 100 خواهیم داشت که کیانگین صحیح های پیدا شده عبارت است از:

```
truth =

  145.1100
```

همان طور که میبینید در حالت استفاده از فاصله کسینوسی مقدار خطا کاهش یافته است هرچند در مجموع این دو متد دارای دقت مناسب و نزدیک هم میباشند

سوال 1:

روش k-means clustering از جمله روش های خوشه بندی میباشد که از دسته بدون نظارت میباشد.این روش سعی دارد تا فاصله اعضای هر خوشه تا مرکز ان را مینیموم نماید.

در این روش از تخصیص لیبل به خوشه ها خود داری می نماییم. در اصل هدف ان است که در دنیای واقعی که اطلاعاتی راجب تعداد خوشه ها موجود نمی باشد بتوان بهترین تقسیم بندی برای داده هارا بدست اورد.

الف)

150 بار به صورت رندم داده ها را به صورت رندم یکی یکی انتخاب کرده و و با توجه به تعداد خوشه های پیش فرض که یکی از مقادیر 3 5 7 9 می باشد, این داده ها را در این خوشه ها قرار خواهیم داد.

در این قطعه کد زیر شاخه ها و مرکز های اولیه را مشاهده می نمایید:

```
branch(1:k(c),1,1:5)=-1;
center(1:k(c),:)=e(shuffled(1:k(c)),:);
branch(1:k(c),1,1:5)=center(1:k(c),:);
```

فاصله هر کدام از داده هایی که به صورت رندم انتخاب کردیم را از همه خوشه ها خواهیم یافت:

```
%choose each data one by one randomly and put it in best branch based
%on distances from each center
for i=k(c)+1:1:150
    d=zeros(1,k(c));
    for j=1:1:k(c)
        for index=1:1:4
            d(j)=(e(shuffled(i),index)-center(j,index))^2+d(j);
        end
        d(j)=sqrt(d(j));
    end
```

سپس بر اساس این فاصله ان را در نزدیک ترین خوشه قرار خواهیم داد:

```
%choose the nearest center to data
u=1:1:k(c);
r=[d',u'];
sorted_distance=sortrows(r,1);
choosed=sorted_distance(1,2);

%updating centers and datas
center(choosed,:)=(center(choosed,:)*b_length(choosed)+e(shuffled(i),:))/(b_length(choosed)+1);
b_length(choosed)=b_length(choosed)+1;
branch(choosed,b_length(choosed),:)=e(shuffled(i),:);
```

برای انتخاب بهترین تعداد خوشه بندی به معیار های فاصله داخلی و فاصله خارجی نیاز مندیم.

معیرا فاصله داخلی را به صورت زیر تعریف نمودیم:

$W(C)$ can be rewritten as:

$$W(C) = \sum_{k=1}^{K} |C_k| \sum_{i \in C_k} \|x_i - \bar{x}_k\|^2$$

(obtained by rewriting $(x_i - x_j) = (x_i - \bar{x}_k) - (x_j - \bar{x}_k)$)

where

$\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$ is the mean vector of cluster C_k

$|C_k|$ is the number of points in cluster C_k

در مورد این فاصله باید گفت که فاصله تمام اعضای هر خوشه را از مرکط ان در نظظر گرفته و در تعداد اعضای ان خوشه ضرب خواهد کرد.

نکته مثبت تعریف به این شکل ان است که اثر تعداد اعضای خوشه ها را نیز در فاصله اثر میدهیم .

نتیجه بخش مشخ شده با رنگ قرمز را در شکل زیر می توانید مشاهده نمایید:

	1	2	3	4	5	6	7	8	9
1	207.6545	348.2199	296.2780	0	0	0	0	0	0
2	44.0336	102.7647	16.2296	171.9594	196.7986	0	0	0	0
3	44.1321	34.7853	22.1693	299.9575	3.3317	69.4628	15.7016	0	0
4	13.5860	53.7042	37.1018	21.0855	61.9106	87.0614	33.7310	10.0150	1.6971

این عملیات توسط قطعه کد زیر پیاده شده است:

```
%in this part we find inner distances of branches.
for i=1:1:k(c)
    in_distance(c,i)=0;
    for j=1:1:b_length(i)

        for index=1:1:4
            in_distance(c,i)=(branch(i,j,index)-center(i,index))^2+in_distance(c,i);
        end
    end
    in_distance(c,i)=in_distance(c,i)^.5;
    in_distance(c,i)=in_distance(c,i)*b_length(i);
end
```

فاصله خارجی را نیز به شکل زیر تعریف خواهیم کرد:

مجموع فاصله مرکز هر خوشه از مرکز های خوشه های دیگر

در این مورد توجه شود که فاصله بیرونی را میتوان فاصله هر داده از تمام داده های دیگر تعریف کرد که این مورد \ردازش سنگینی را طلب خواهد کرد.به همین دلیل از تعریف ذکر شده استفاده کردیم.

قطعه کد مربوطه عبارت است از:

```
%in this part we find out distances of branches.
for i=1:1:k(c)
    out_distance(c,i)=0;
    for j=1:1:k(c)
        for index=1:1:4
            out_distance(c,i)=(center(i,index)-center(j,index))^2+out_distance(c,i);
        end
    end
    out_distance(c,i)=out_distance(c,i)^.5;
    %decision_distance(c,i)=out_distance(c,i)/in_distance(c,i)
end
```

و فواصل خروجی عبارتند از:

	1	2	3	4	5	6	7	8	9
1	5.1541	3.6816	5.8411	0	0	0	0	0	0
2	4.8999	6.9047	6.8171	7.7094	5.5306	0	0	0	0
3	8.4251	8.6019	11.3194	7.0587	9.2609	9.1721	6.5554	0	0
4	5.5561	5.7787	10.8485	7.1912	10.2575	8.9996	6.3303	5.9978	5.9169

حال با قطعه کد زیر به معیار مورد نظر جهت تصمیم گیری برای بهترین K خواهیم رسید:

```
sum(out_distance,2)./sum(in_distance,2)
```

نتیجه حاصل:

```
ans =

    0.0172
    0.0599
    0.1234
    0.2091
```

توجه: به این نکته توجه کنید که فاصله بیرونی به درونی به عنوان معیار گرفته شده است در نتیجه هر چه این معیار بزرگ تر باشد نشان دهنده عملکرد بهتر می باشد زیرا ب معنای دور تر بودن خوشه ها از هم و همچنین نزدیکی اعضای یک خوشه به یکدیگر یعنی شباهت بیشتر ان ها خواهد بود.

یعنی به ازای K برابر با ۹: به طور نسبی از یک سو داده های هر یک از شاخه ها با شاخه های دیگر فاصله بیشتر دارند ولی داده های هر خوشه به یکدیگر شبیه تر میباشند.

توجه مهم : با توجه به ماهیت انتخابی تصادفب مربوط به این روش , گاهی به ازای k برابر با 7 نیز معیار عملکرد بهتری نسبت به k برابر 9 از خود نشان میدهد , به طور مثال میتوانید در شکل زیر همچین نمونه این را مشاهده نمایید

```
    0.0194    0.0577    0.1321    0.1044
```

همچنین یکی از دلایل این اتفاق بسیار نزدیک بودن عملکرد الگوریتم به ازای k برابر با 7 و 9 است که اثر تصادفی بودن خود را بر ان می گذارد.

مقایسه knn و k clustering :

Kmeans از جمله روش های unsupervised learning technique میباشد که برای خوشه بندی از ان استفاده خواهد شد, حال انکه که الگوریتم knn در دسته روش های supervised استفاده خواهد شد.

در روش kmeans , k تعداد خوشه هایی است که الگوزیتم تلاش میکند تا از داده ها شناسایی نماید. همچنین این خوشه ها و تعداد انها اغلب ناشناخته هستند زیرا این روشی غیر نظارتی است.

از k means برای عملیات های چون درک جمعیت, تقسیم بندی بازار و رسانه های اجتماعی استفاده می شود اما روش knn برای رگرسیون و یا طبقه بندی کاربرد دارد.

در knn ,k تعداد نزدیک ترین همسایه ها می باشد که باعث طبقه بندی یک داده یا در رگرسیون یک داده ب کار میرود.

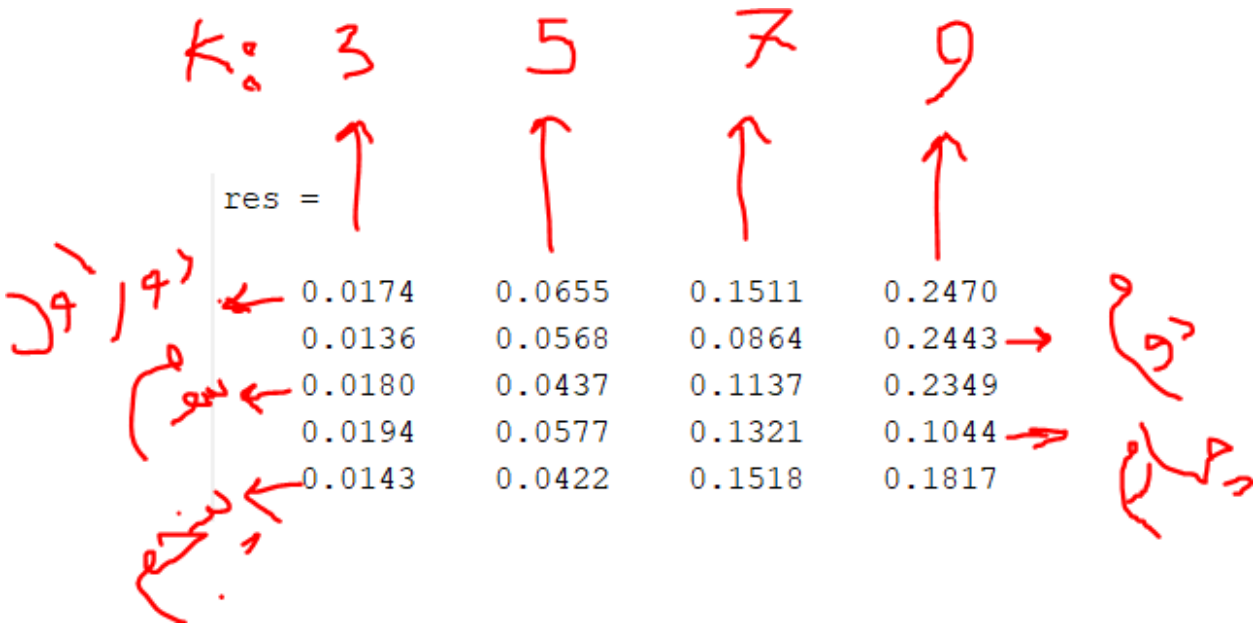
در الگوریتم kmeans , k به صورت خود سرانه انتخاب می شوند و مرکز هایی به تعداد k در ان انخاب میشوند از این رو به عنوان centroids شناخته میشوند. در نتیجه هر نقطه در فضای n بعدی به نزدیک ترین خوشه تخصیص داده میشود.و ان مرکز نیز اپدیت خواهد شد.

در الگوریتم knn لیبل ها را داریم و در نتیجه طبقه بندی یا رگرسیون داده ها یه وسیله داده هایی انجام میگیرد که ویژگی و لیبل ان هارا میدانیم.

یکی از تفاوت های عمده این دو روش ان است که روش knn مرحله train کردن ندارد و طبقه بندی براس k داده نزدیک انجام خواهد شد.

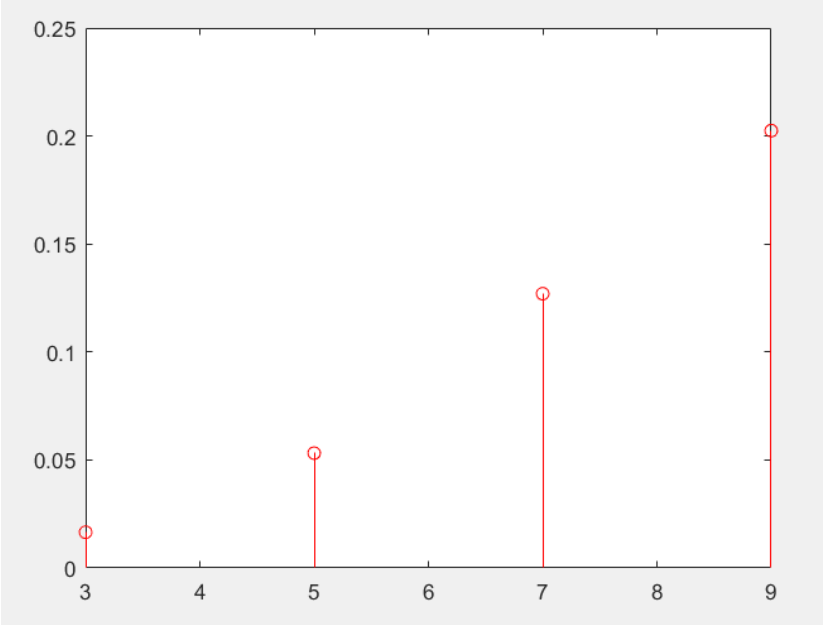
ب)

همان طور که صورت پروژه خواسته است 5 بار الگوریتم بالا را اجراخواهیم کرد:

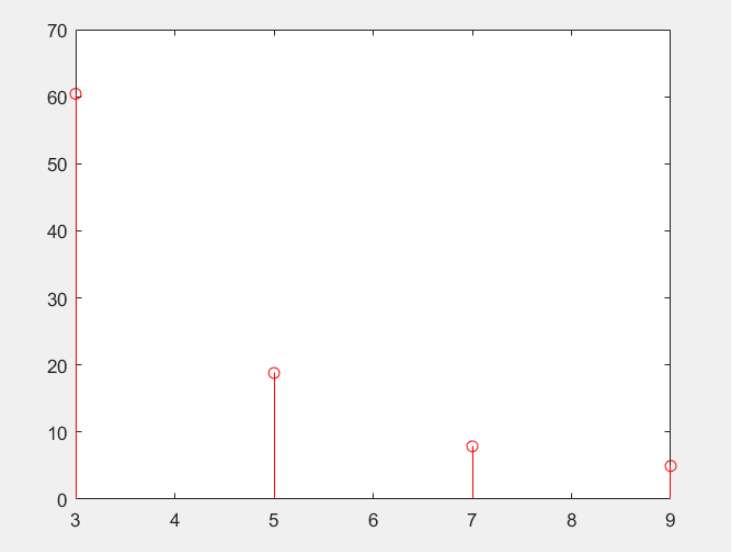


توجه نمایید که تابع های رسم شده در شکل های زیر در ابتدا در حالت معیار نسبت فاصله خروجی به فاصله درونی و سپس در حالت فاصله درونی به بیرونی رسم شده است, طبیعتا در جایی که میانگین بیشتر است در حالت عکس مانگین کمتر است و در جایی که واریانس بیشتر است در حالت معکوس وارینس کمتر خواهد بود.

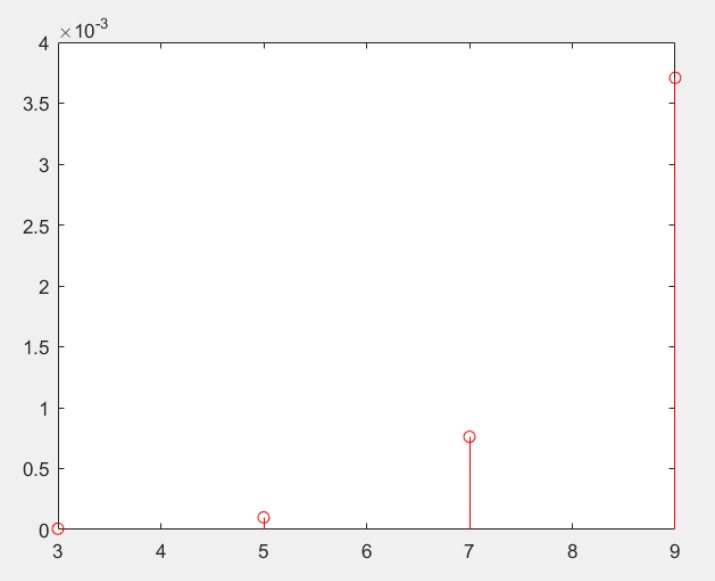
Mean of $\frac{out\ distance}{in\ distance}$ for k :3 5 7 9:



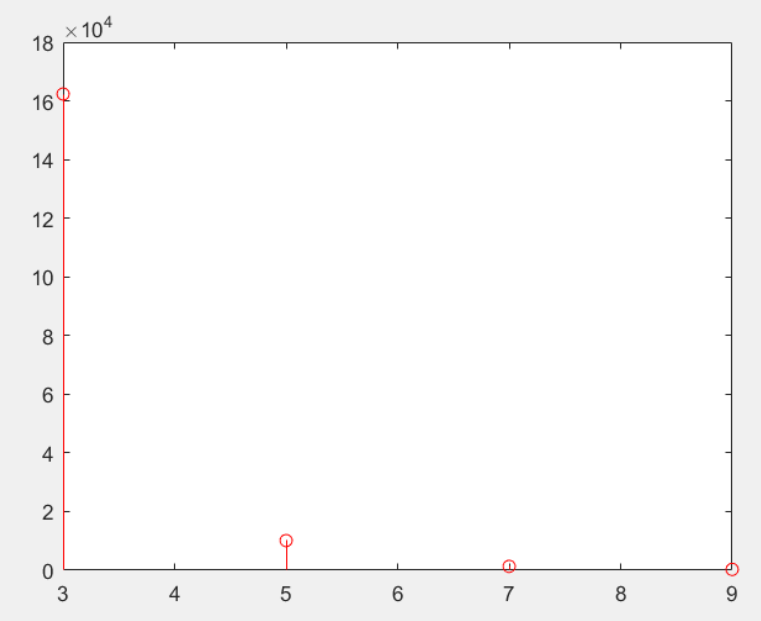
Mean of $\frac{in\ distance}{out\ distance}$ for k :3 5 7 9:



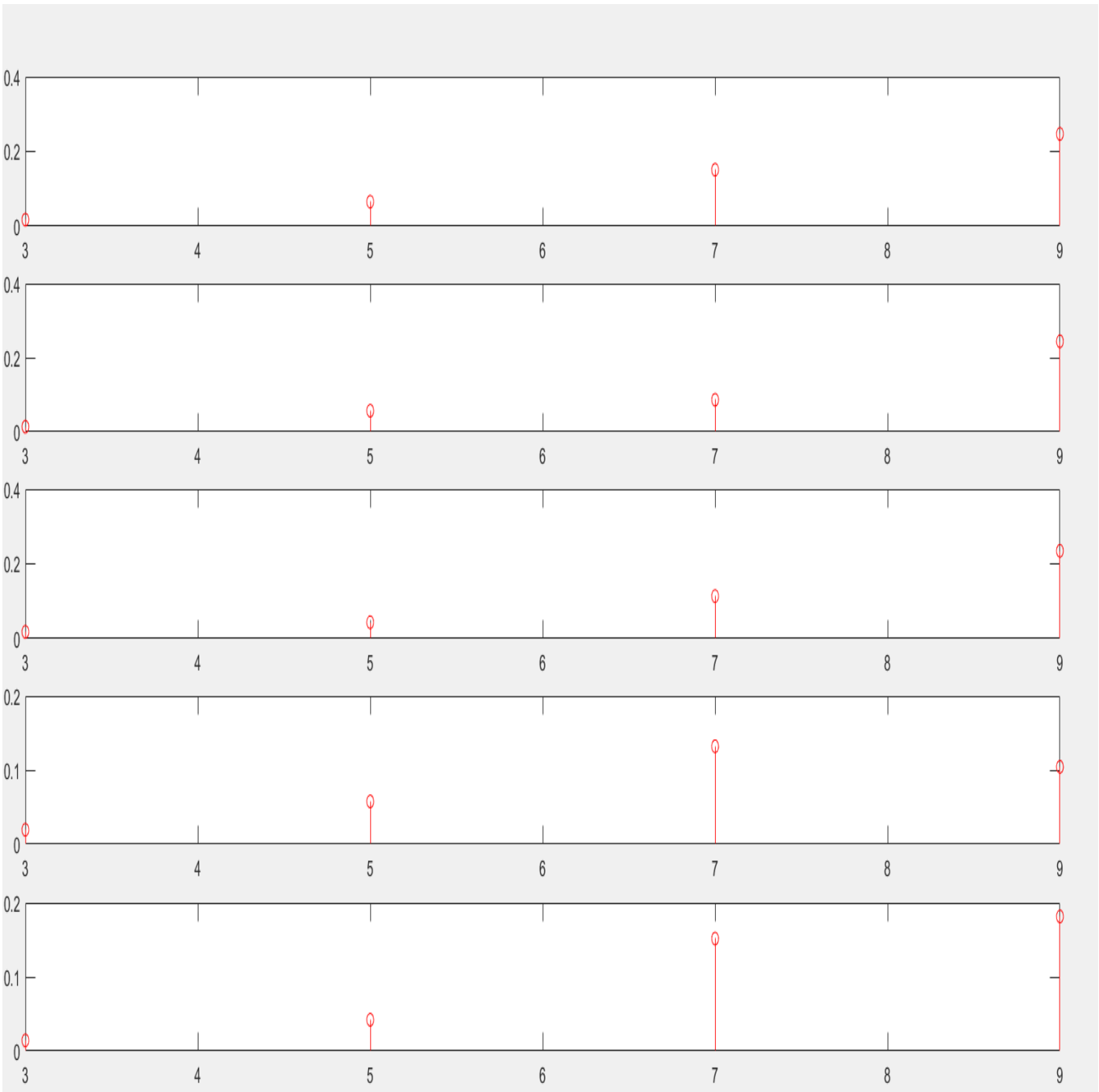
Variance of $\frac{out\ distance}{in\ distance}$ for k :3 5 7 9:



Variance of $\frac{in\ distance}{out\ distance}$ for k :3 5 7 9:



تابع معیار در هر دور اجرا:


0.4
0.2
0
3
4
5
6
7
8
9
0.4
0.2
0
3
4
5
6
7
8
9
0.4
0.2
0
3
4
5
6
7
8
9
0.2
0.1
0
3
4
5
6
7
8
9
0.2
0.1
0
3
4
5
6
7
8
9