Assignment 1

Intelligent Data Analysis

Q1)

Add the two scores for each student and write the sum in a new column. Discretize the sum column into five groups using equal width partitioning and assign one of the five grades (A, B, C,D, and F) to students in the five groups (the highest scores get A and the lowest scores get F).Show the grades assigned to each student in a list sorted according to student id. List the counts of each letter grade awarded.

Explanation:

Given data in excel file is read into Table(T), converted table into Cell(C) using in-built **table2cell** function and added physics,maths scores by user-defined **AddingTwoColumns** function and placed the sum in other column. **EqualWidthPattern** function is used to discretize the total scores in equal width partitions and naming them with A to F grades. Resultant cell is converted back to table using in built **cell2table** function and provided variables to display in the table header.And the frequency of each grade is displayed by identifying the counts using **hist** function and printed using **disp()**

Program:

```
C=table2cell(T);

C=AddingTwoColumns(C,2,3,5);

%Calling equal width partioning for new score and passing grades ordered vector

[C,GradeMat]=EqualWidthPartn(C,5,5,{'F', 'D', 'C', 'B', 'A'},6);

C=sortrows(C,1);

 TResult=cell2table(C,'VariableNames',{'StudentId' 'Maths' 'Physics'...

                        'C__Grade' 'Total' ‘TotalGrade_EWP'}); disp(TResult);

GradeResult=CountGrades(GradeMat,'hist');

Result=cell2table(GradeResult,...

            'VariableNames',{'Grade' 'Count'});

 disp(Result);
```

AddingTwoColumns function:

```
function Cell = AddingTwoColumns(Cell,Col1,Col2,newIdxPos)
%Adds Col1 and Col2 and places new result in newIdPos (new index position)
for idx=1:length(Cell)
   Cell{idx,newIdxPos}=Cell{idx,Col1}+Cell{idx,Col2};
end
end
```

EqualWidthPartn function:

```
function [Cell,MAT] = EqualWidthPartn(Cell,partnCIdx,no_Partn,orderGrade,newCIdx)
%Equal width partitioning will be applied on matrix B and partioned into
%no_partn( no. of partitions, orderGrade is ordered grading vector
B=cell2mat(Cell(:,5));
min_value=min(B);max_value=max(B);
width=(max_value-min_value)/no_Partn;
B=sort(B);
edges = min_value:width:max_value;
MAT = discretize(B,edges,'categorical',...
   orderGrade);
Cell=sortrows(Cell,partnCIdx);
%inserting grades into Cell Array
for index=1:length(MAT)
   Cell{index,newCIdx}=MAT(index,:);
end
end
```

CountGrades function:

```
function [Result] =CountGrades(GradeMat,type)
 GradeMat_Unique = unique(GradeMat);
 if strcmp(type,'histc')
 var2 = histc(GradeMat,GradeMat_Unique);
 elseif strcmp(type,'hist')
 var2 = hist(GradeMat,GradeMat_Unique);
 end
 for index=1:length(GradeMat_Unique)
      Result{index,1}=GradeMat_Unique(index,1);
     if strcmp(type,'histc')
        Result{index,2}=var2(index,1);
     elseif strcmp(type,'hist')
        Result{index,2}=var2(1,index);
      end
 end
  Result=sortrows(Result,2);
 end
```

Output:

```
 StudentId    Maths    Physics    C__Grade    Total    TotalGrade_EWP
   _________    _____    _______    ________    _____    ______________


    1         65       82       'A'       147      B
    2         78       48       'B'       126      B
    3         37       45       'C'        82     D
    4         75       87       'A'       162      A
```

5	55	15	'C'	70	D
6	54	60	'B'	114	C
7	55	19	'C'	74	D
8	63	32	'C'	95	C
9	95	65	'A'	160	A
10	87	48	'B'	135	B
11	46	14	'D'	60	D
12	59	43	'C'	102	C
13	67	61	'B'	128	B
14	91	100	'A'	191	A
15	58	30	'C'	88	C
16	57	45	'C'	102	C
17	58	19	'C'	77	D
18	74	45	'B'	119	C
19	75	37	'C'	112	C
20	74	10	'C'	84	D
21	66	26	'C'	92	C
22	47	2	'D'	49	F
23	67	14	'C'	81	D
24	79	65	'B'	144	B
25	64	43	'B'	107	C
26	70	23	'C'	93	C
27	67	49	'B'	116	C
28	56	21	'C'	77	D
29	62	54	'B'	116	C
30	52	62	'B'	114	C
31	68	69	'B'	137	B
32	48	34	'C'	82	D

```
33        98      91      'A'       189     A
34        51      14      'C'        65     D
35        13       4     'D'        17     F
36        74       8     'C'        82     D
37        63      68      'B'       131     B
38        52      43      'C'        95     C
39        99      88      'A'       187     A
40        24       0     'D'        24     F

Grade    Count
_____    _____
F        3
A        5
B        7
D        11
C        14
```

Q2)

Repeat problem #1 above with the difference that this time use equal frequency partitioning to discretize the sum of points obtained. List the counts of each letter grade awarded. List student ids of those students who would be happier with equal width binning and also of those students' ids who would be happier with equal frequency binning

Explanation: Followed the same procedure explained in Ans1. Used discretize function for equal frequency binning. Compared the grades generated in Ans1 and Ans2 , displayed the happier graders using **CompareHappierGraders** function.\

Program:

```
T = readtable('/DataHW1A.xlsx');
```

```
C=table2cell(T);
C=AddingTwoColumns(C,2,3,5);
C1=C;
%Calling equal frequency partitioning for
% new score and passing grades ordered
%vector
[C,GradeMat]=EqualFrqncyPartn(C,5,5,{'F', 'D', 'C', 'B', 'A'},6);
 %%sorting by studentId
C=sortrows(C,1);
CellDisplay=cell2table(C,...
                    'VariableNames',{'StudentId' 'Maths' 'Physics'...
                                  'C__Grade' 'Total' 'TotalGrade_EFP'});
disp(CellDisplay);
 GradeResult=CountGrades(GradeMat,'histc');
 Result=cell2table(GradeResult,...
                 'VariableNames',{'Grade' 'Count'});
 disp(Result);
[C1,GradeMat]=EqualWidthPartn(C1,5,5,{'F', 'D', 'C', 'B', 'A'},6);
 %sorting by studentId
 C1=sortrows(C1,1);
[student_happy_epw,student_happy_efw]=CompareHappierGraders(C,C1,6,1);
 format short;
  fprintf('\nstudent happy with Equal Width Partitioning\n')
  Tstudent_happy_epw=array2table(student_happy_epw,...
    'VariableNames',{'StudentId'});
   disp(Tstudent_happy_epw);
 fprintf('\nstudent happy with Equal Frequency Partitioning\n')
   Tstudent_happy_efw=array2table(student_happy_efw,...
   'VariableNames',{'StudentId'});
 disp(Tstudent_happy_efw);
```

EqualFrqncyPartn function

```
function [Cell,MAT] = EqualFrqncyPartn(Cell,partnCIdx,no_Partn,orderGrade,newCIdx)
%Equal width partitioning will be applied on matrix B and partioned into
%no_partn( no. of partitions, orderGrade is ordered grading vector
B=cell2mat(Cell(:,5));
B=sort(B);
width=length(B)/no_Partn;
edges(1,1)=B(1,1);
 for i=1:no_Partn;
     edges(1,1+i)=B(i*width,1);
 end
% K= histcounts(B,edges);
temp = discretize(B,edges,'IncludedEdge','right');
for idx=1:length(temp);
      switch(temp(idx,1))
   case 1
      MAT(idx,1)='F';
   case 2
      MAT(idx,1)='D';
   case 3
      MAT(idx,1)='C';
   case 4
      MAT(idx,1)='B';
```

```
  case 5
    MAT(idx,1)='A';
  otherwise
     fprintf('Invalid grade\n' );
    end
  end
 Cell=sortrows(Cell,partnCIdx);
% % %inserting grades into Cell Array
 for index=1:length(MAT)
    Cell{index,newCIdx}=MAT(index,:);
 end
end
```

CompareHappierGraders function

```
function [Stud1_Happier,Stud2_Happier] = CompareHappierGraders(Cell1,Cell2,...
    idxCompare,idxReturn)
% %Comparing students happier in equal width and equal frequency
 j=1;k=1;
 for idx=1:length(Cell1)
    if (char(Cell1{idx,idxCompare})<char(Cell2{idx,idxCompare}))
      temp = cell2mat(Cell1(idx,idxReturn));
      Stud1_Happier{j,1}=temp(1,1);
      j=j+1;
    elseif (char(Cell1{idx,idxCompare})>char(Cell2{idx,idxCompare}))
         temp = cell2mat(Cell2(idx,idxReturn));
        Stud2_Happier{k,1}=temp(1,1);
      k=k+1;
    end
 end
 Stud1_Happier=cell2mat(Stud1_Happier(:,1));
 Stud2_Happier=cell2mat(Stud2_Happier(:,1));
end
```

Output:

```
  StudentId    Maths    Physics    C__Grade    Total    TotalGrade_EFP
    _________    ______    ________    _________    ______    _______________


    1        65      82      'A'       147     'A'
    2        78      48      'B'       126     'B'
    3        37      45      'C'        82     'D'
    4        75      87      'A'       162     'A'
    5        55      15      'C'        70     'F'
    6        54      60      'B'       114     'C'
    7        55      19      'C'        74     'F'
    8        63      32      'C'        95     'C'
    9        95      65      'A'       160     'A'
    10        87      48      'B'       135     'B'
    11        46      14      'D'        60     'F'
    12        59      43      'C'       102     'C'
    13        67      61      'B'       128     'B'
    14        91      100      'A'       191     'A'
    15        58      30      'C'        88     'D'
    16        57      45      'C'       102     'C'
    17        58      19      'C'        77     'F'
    18        74      45      'B'       119     'B'
    19        75      37      'C'       112     'C'
```

```
20      74    10    'C'    84   'D'
21      66    26    'C'    92   'D'
22      47     2   'D'    49   'F'
23      67    14    'C'    81   'D'
24      79    65    'B'   144    'A'
25      64    43    'B'   107    'C'
26      70    23    'C'    93   'C'
27      67    49    'B'   116    'B'
28      56    21    'C'    77   'F'
29      62    54    'B'   116    'B'
30      52    62    'B'   114    'C'
31      68    69    'B'   137    'A'
32      48    34    'C'    82   'D'
33      98    91    'A'   189    'A'
34      51    14    'C'    65   'F'
35      13     4   'D'    17   'F'
36      74     8   'C'    82   'D'
37      63    68    'B'   131    'B'
38      52    43    'C'    95   'C'
39      99    88    'A'   187    'A'
40      24     0   'D'    24   'F'

Grade    Count
_____    _____

'B'     7
'D'     7
'A'     8
```

```
'C'     9
'F'     9
```

student happy with Equal Width Partitioning

```
StudentId
_________

 1
18
24
27
29
31
```

student happy with Equal Frequency Partitioning

```
StudentId
_________

 5
 7
11
15
17
21
28
34
```

Q3)

Compare the grades assigned in problem #1 and #2 above. Make a list of those student ids whose grades changed when the method of binning changed.

Explanation: Equal width partitioning and equal frequency partition calculated in Ans1 and Ans2 produces list of student id who are happier with grades.These stundents id comparison gives the students with changing grades when binning is changed

Program:

```
T = readtable('/DataHW1A.xlsx');
C=table2cell(T);
C=AddingTwoColumns(C,2,3,5);
C1=C;
%Calling equal frequency partioning for
% new score and passing grades ordered
%vector
[C,GradeMat]=EqualFrqncyPartn(C,5,5,{'F', 'D', 'C', 'B', 'A'},6);
 %%sorting by studentId
 C=sortrows(C,1);
 [C1,GradeMat]=EqualWidthPartn(C1,5,5,{'F', 'D', 'C', 'B', 'A'},6);
 %sorting by studentId
 C1=sortrows(C1,1);
 [student_happy_epw,student_happy_efw]=CompareHappierGraders(C,C1,...
                              6,1);
ChangedGradeStudents=[student_happy_epw;student_happy_efw];
fprintf('\n Students with changed grades when binning changed\n');
ChangedGradeStudents=sortrows(ChangedGradeStudents);
 TResult=array2table(ChangedGradeStudents,...
    'VariableNames',{'StudentId'});
```

```
disp(TResult);
```

Output:

```
Students with changed grades when binning changed

    StudentId
    _________

        1
        5
        7
       11
       15
       17
       18
       21
       24
       27
       28
       29
       31
       34
```

Q4)

Convert the Physics and Maths points to their equivalent z-scores in each column. Sum the two z-scores for each student and use equal frequency binning to create five bins. Assign grades to the students and show them in a list sorted by student ids.

Explanation: Zscore of physics and maths calculated using Zscore() and applied equal frequency binning using EqualFrqncyPartn()

Program:

```
T = readtable('/DataHW1A.xlsx');
```

```
Cell=table2cell(T);
% C=AddingTwoColumns(C,2,3,5);
M=cell2mat(Cell(:,2));
P=cell2mat(Cell(:,3));
format bank;
ZM=zscore(M);
ZP=zscore(P);
for idx=1:length(Cell)
   Cell{idx,2}=ZM(idx,1);
   Cell{idx,3}=ZP(idx,1);
end
Cell=AddingTwoColumns(Cell,2,3,5);
[Cell,GradeMat]=EqualFrqncyPartn(Cell,5,5,{'F', 'D', 'C', 'B', 'A'},6);
Cell=sortrows(Cell,1);
CellDisplay=cell2table(Cell,...
                    'VariableNames',{'StudentId' 'Maths' 'Physics'...
                                  'C__Grade' 'Total' ‘TotalGrade_EFP'});
disp(CellDisplay);
```

EqualFrqncyPartn function

```
function [Cell,MAT] = EqualFrqncyPartn(Cell,partnCIdx,no_Partn,orderGrade,newCIdx)
%Equal width partitioning will be applied on matrix B and partioned into
%no_partn( no. of partitions, orderGrade is ordered grading vector
B=cell2mat(Cell(:,5));
B=sort(B);
width=length(B)/no_Partn;
edges(1,1)=B(1,1);
```

```
 for i=1:no_Partn;
     edges(1,1+i)=B(i*width,1);
 end
 % K= histcounts(B,edges);
temp = discretize(B,edges,'IncludedEdge','right');
for idx=1:length(temp);
     switch(temp(idx,1))
   case 1
      MAT(idx,1)='F';
   case 2
      MAT(idx,1)='D';
   case 3
      MAT(idx,1)='C';
   case 4
      MAT(idx,1)='B';
   case 5
     MAT(idx,1)='A';
       otherwise
      fprintf('Invalid grade\n' );
     end
   end
 Cell=sortrows(Cell,partnCIdx);
% % %inserting grades into Cell Array
 for index=1:length(MAT)
    Cell{index,newCIdx}=MAT(index,:);
 end
end
```

Output

```
StudentId    Maths    Physics    C__Grade    Total    TotalGrade_EFP

    __________    ______    ________    _________    ______    _______________

     1.00         0.08     1.49      'A'         1.57    'A'

     2.00         0.81     0.22      'B'         1.03    'B'

     3.00        -1.49     0.11      'C'        -1.38    'F'

     4.00         0.64     1.68      'A'         2.32    'A'

     5.00        -0.48    -1.01      'C'        -1.49    'F'

     6.00        -0.54     0.67      'B'         0.13    'C'

     7.00        -0.48    -0.86      'C'        -1.34    'F'

     8.00        -0.03    -0.38      'C'        -0.41    'C'

     9.00         1.76     0.86      'A'         2.62    'A'

    10.00         1.32     0.22      'B'         1.54    'A'

    11.00        -0.99    -1.05      'D'        -2.04    'F'

    12.00        -0.26     0.03      'C'        -0.22    'C'

    13.00         0.19     0.71      'B'         0.90    'B'

    14.00         1.54     2.17      'A'         3.71    'A'

    15.00        -0.31    -0.45      'C'        -0.76    'D'

    16.00        -0.37     0.11      'C'        -0.26    'C'

    17.00        -0.31    -0.86      'C'        -1.18    'D'

    18.00         0.59     0.11      'B'         0.69    'B'

    19.00         0.64    -0.19      'C'         0.45    'B'

    20.00         0.59    -1.20      'C'        -0.61    'D'

    21.00         0.14    -0.60      'C'        -0.47    'C'

    22.00        -0.93    -1.50      'D'        -2.43    'F'

    23.00         0.19    -1.05      'C'        -0.86    'D'

    24.00         0.87     0.86      'B'         1.72    'A'

    25.00         0.02     0.03      'B'         0.06    'C'
```

```
   26.00     0.36   -0.71    'C'     -0.35   'C'
   27.00     0.19    0.26    'B'      0.45   'B'
   28.00    -0.43   -0.79    'C'     -1.21   'D'
   29.00    -0.09    0.45    'B'      0.36   'B'
   30.00    -0.65    0.75    'B'      0.10   'C'
   31.00     0.25    1.01    'B'      1.26   'B'
   32.00    -0.87   -0.30    'C'     -1.18   'D'
   33.00     1.93    1.83    'A'      3.76   'A'
   34.00    -0.71   -1.05    'C'     -1.76   'F'
   35.00    -2.84   -1.42    'D'     -4.26   'F'
   36.00     0.59   -1.27    'C'     -0.69   'D'
   37.00    -0.03    0.97    'B'      0.94   'B'
   38.00    -0.65    0.03    'C'     -0.62   'D'
   39.00     1.99    1.72    'A'      3.71   'A'
   40.00    -2.22   -1.57    'D'     -3.80   'F'
```

Q5)

Compare the grades obtained in #4 and in #2 above. Make a list of students who would be happier with the method in #2 and also a list of those who would be happier with the method in #4.

Explanation: compared results generated in Ans2 and Ans4. compared two data sets and printed the student ids who are happier with 2 and 4 individually.

Program:

```
T = readtable('/DataHW1A.xlsx');

Cell=table2cell(T);

Cell2=Cell;

% C=AddingTwoColumns(C,2,3,5);

M=cell2mat(Cell(:,2));

P=cell2mat(Cell(:,3));

ZM=zscore(M);
```

```
ZP=zscore(P);
for idx=1:length(Cell)
    Cell{idx,2}=ZM(idx,1);
    Cell{idx,3}=ZP(idx,1);
end
Cell=AddingTwoColumns(Cell,2,3,5);
[Cell,GradeMat]=EqualFrqncyPartn(Cell,5,5,{'F', 'D', 'C', 'B', 'A'},6);
Cell=sortrows(Cell,1);
%Calling equal frequency partioning for
% new score and passing grades ordered
%vector
Cell2=AddingTwoColumns(Cell2,2,3,5);
[Cell2,GradeMat]=EqualFrqncyPartn(Cell2,5,5,{'F', 'D', 'C', 'B', 'A'},6);
 %%sorting by studentId
Cell2=sortrows(Cell2,1);

[student_happy_epw,student_happy_efw]=CompareHappierGraders(Cell,Cell2,6,1);
 format short;
 fprintf('\nstudent happy with ZScore Equal Frequency Partitioning\n')
  Tstudent_happy_epw=array2table(student_happy_epw,...
    'VariableNames',{'StudentId'});
 disp(Tstudent_happy_epw);
  fprintf('\nstudent happy with Equal Frequency Partitioning\n')
   Tstudent_happy_efw=array2table(student_happy_efw,...
    'VariableNames',{'StudentId'});
 disp(Tstudent_happy_efw);
```

CompareHappierGraders function

```
function [Stud1_Happier,Stud2_Happier] = CompareHappierGraders(Cell1,Cell2,...
   idxCompare,idxReturn)
% %Comparing students happier in equal width and equal frequency
 j=1;k=1;
 for idx=1:length(Cell1)
   if (char(Cell1{idx,idxCompare})<char(Cell2{idx,idxCompare}))
     temp = cell2mat(Cell1(idx,idxReturn));
     Stud1_Happier{j,1}=temp(1,1);
     j=j+1;
   elseif (char(Cell1{idx,idxCompare})>char(Cell2{idx,idxCompare}))
        temp = cell2mat(Cell2(idx,idxReturn));
       Stud2_Happier{k,1}=temp(1,1);
     k=k+1;
   end
 end
 Stud1_Happier=cell2mat(Stud1_Happier(:,1));
 Stud2_Happier=cell2mat(Stud2_Happier(:,1));
end
```

Output:-

```
student happy with ZScore Equal Frequency Partitioning
    StudentId
    _________

    10
    17
    19
    21
    28
student happy with Equal Frequency Partitioning
```

StudentId

____

3

31

38

Q6)

Consider a student who is happier with the method in #4 compared to the method in #2.Briefly explain why his being happier is justified.

Ans

ZScore gives more accuracy by partitioning the scores that we will no more have the same values at the edges. So all the bins will be getting equal no of student ids.

Let us consider the question why a certain student is happy with zscore frequency binning rather than the normal frequency binning.

For stunted id 28, In physics he got 56 and in Maths he got 21. Just by visualizing at the scores we will think that he is poor student in mathematics but an average student in physics.

But that is not true. When we look at the math scores of all the students only 5 students got above 80 but remaining all the stuck be low 60 which shows us that they are only few bright students in the class and remaining all are normal students with respect to maths. And student 28 comes as an average student not a failure.

But in physics, all the students are distributed equally and his score is an average score.

When we apply equal frequency binning without zscore, it just pushed him into 'F' grade without considering the whole class avg. But in zscore frequency binning, it pushed him back into 'D' grade by considering the average of the class.

Q7)

For Physics and Maths scores individually, perform the following. Use z-scores to assign label "Low" to those students whose z-score is strictly below -0.3, the label "Mid" to those whose z-score is between -0.3 and +0.3 (both values inclusive), and the label "High" to those students whose z-score is strictly greater than 0.3. Show the data table sorted according to student ids.

Explanation: The data tuples are partitioned with split point condition -0.3<A, -0.3<=A<=0, A>0.3. Named the groups Low, Mid and High and displayed the table.

Program:

```
T = readtable('/DataHW1A.xlsx');
Cell=table2cell(T);
Cell=Calculatezscores(Cell);
Cell=sortrows(Cell,1);
CellDisplay=cell2table(Cell,...
                    'VariableNames',{'StudentId' 'Physics_ZScore' 'Maths_ZScore'...
                                  'C__Grade' 'Physics_Label' 'Maths_Label'});
disp(CellDisplay);
```

Calculatezscores Function:

```
%C=AddingTwoColumns(C,2,3,5);
function [Cell] =Calculatezscores(Cell)
P=cell2mat(Cell(:,2));
M=cell2mat(Cell(:,3));
ZM=zscore(M);
ZP=zscore(P);
for idx=1:length(Cell)
    Cell{idx,2}=ZP(idx,1);
    Cell{idx,3}=ZM(idx,1);
     if (ZM(idx,1)<-0.3)
         Cell{idx,6}='Low';
      elseif(-0.3<=ZM(idx,1) && ZM(idx,1)<=0.3)
         Cell{idx,6}='Mid';
      else
         Cell{idx,6}='High';
      end
       if (ZP(idx,1)<-0.3)
```

```
        Cell{idx,5}='Low';
      elseif(-0.3<=ZP(idx,1) && ZP(idx,1)<=0.3)
        Cell{idx,5}='Mid';
      else
        Cell{idx,5}='High';
      end
   end
end
```

Output:-

```
StudentId    Physics_ZScore    Maths_ZScore    C__Grade    Physics_Label    Maths_Label
    _________    _______________    _____________    ________    ______________    ____________


     1         0.080001         1.4936       'A'       'Mid'         'High'
     2          0.80984        0.22165       'B'       'High'        'Mid'
     3           -1.492        0.10942       'C'       'Low'         'Mid'
     4          0.64141         1.6806       'A'       'High'        'High'
     5         -0.48141        -1.0129       'C'       'Low'         'Low'
     6         -0.53755        0.67058       'B'       'Low'         'High'
     7         -0.48141       -0.86324       'C'       'Low'         'Low'
     8        -0.032281       -0.37691       'C'       'Mid'         'Low'
     9           1.7642        0.85763       'A'       'High'        'High'
    10           1.3151        0.22165       'B'       'High'        'Mid'
    11         -0.98668        -1.0503       'D'       'Low'         'Low'
    12         -0.25685       0.034604       'C'       'Mid'         'Mid'
    13          0.19228        0.70799       'B'       'Mid'         'High'
    14           1.5397          2.167       'A'       'High'        'High'
    15         -0.31299       -0.45173       'C'       'Low'         'Low'
```

16	-0.36913	0.10942	'C'	'Low'	'Mid'
17	-0.31299	-0.86324	'C'	'Low'	'Low'
18	0.58527	0.10942	'B'	'High'	'Mid'
19	0.64141	-0.18986	'C'	'High'	'Mid'
20	0.58527	-1.1999	'C'	'High'	'Low'
21	0.13614	-0.60137	'C'	'Mid'	'Low'
22	-0.93054	-1.4992	'D'	'Low'	'Low'
23	0.19228	-1.0503	'C'	'Mid'	'Low'
24	0.86598	0.85763	'B'	'High'	'High'
25	0.02386	0.034604	'B'	'Mid'	'Mid'
26	0.36071	-0.7136	'C'	'High'	'Low'
27	0.19228	0.25906	'B'	'Mid'	'Mid'
28	-0.42527	-0.78842	'C'	'Low'	'Low'
29	-0.088422	0.44612	'B'	'Mid'	'High'
30	-0.64983	0.7454	'B'	'Low'	'High'
31	0.24842	1.0073	'B'	'Mid'	'High'
32	-0.8744	-0.30209	'C'	'Low'	'Low'
33	1.9327	1.8303	'A'	'High'	'High'
34	-0.70597	-1.0503	'C'	'Low'	'Low'
35	-2.8393	-1.4244	'D'	'Low'	'Low'
36	0.58527	-1.2747	'C'	'High'	'Low'
37	-0.032281	0.96986	'B'	'Mid'	'High'
38	-0.64983	0.034604	'C'	'Low'	'Mid'
39	1.9888	1.7181	'A'	'High'	'High'
40	-2.2218	-1.574	'D'	'Low'	'Low'

Q8)

We want to use the labels assigned in #7 above for Physics and Maths scores to predict the letter grade a student would obtain in his/her C++ class. Use the entropy method to determine which course’s label (Physics or Maths) is a better predictor of a student’s grade in C++ class.Show all your work to arrive at your answer.

Explanation:

Physics and maths class labels are classified into High Low and Mid. Applied weighted entropy formula

minimum info requirement =(D1/D)*Entropy(D1)+(D2/D)*Entropy(D2)+(D3/D)*Entropy(D3)+

Entropy = -sigma(pilogpi) where i=1 to n

Calculated minimum info requirement for physics and maths separately.

The min value of(maths,physics) predicts C++ grade well.

Program:

```
% InfoA(D) = |D1|Entropy(D1)+ |D2|Entropy(D2),

T = readtable('/DataHW1A.xlsx');

Cell=table2cell(T);

Cell=Calculatezscores(Cell);

Cell=sortrows(Cell,5);

PhysicsWeightedEntropy=WeightedEntropy(Cell,5);

Cell=sortrows(Cell,6);

MathsWeightedEntropy=WeightedEntropy(Cell,6);

fprintf('\nWeighted Entropy of Maths:%f\n',MathsWeightedEntropy);

fprintf('\nWeighted Entropy of Physics:%f\n’,PhysicsWeightedEntropy);
```

Calculatezscores Function:-

```
function [Cell] =Calculatezscores(Cell)

P=cell2mat(Cell(:,2));

M=cell2mat(Cell(:,3));

ZM=zscore(M);

ZP=zscore(P);

for idx=1:length(Cell)
```

```
   Cell{idx,2}=ZP(idx,1);
   Cell{idx,3}=ZM(idx,1);
    if (ZM(idx,1)<-0.3)
        Cell{idx,6}='Low';
    elseif(-0.3<=ZM(idx,1) && ZM(idx,1)<=0.3)
        Cell{idx,6}='Mid';
    else
        Cell{idx,6}='High';
    end
    if (ZP(idx,1)<-0.3)
        Cell{idx,5}='Low';
    elseif(-0.3<=ZP(idx,1) && ZP(idx,1)<=0.3)
        Cell{idx,5}='Mid';
    else
        Cell{idx,5}='High';
    end
end
end
```

WeightedEntropy function

```
 function [InfoA] = WeightedEntropy(Cell,ClassIndexValue)
GradeMat = Cell(:, ClassIndexValue);
GradeMat=cellstr(GradeMat);
[C,ia,ic] = unique(GradeMat,'last','legacy');
startIndex=0;
InfoA=0;
for idx=1:length(C)
     InfoA=InfoA+(nnz(ic==idx)/length(ic))*CalculateEntropy(Cell,4,startIndex,ia(idx,1));
     startIndex=ia(idx,1);
```

```
end

end
```

CalculateEntropy function

```
function [Entropy] = CalculateEntropy(Cell,EntrpCIndeX,startIndex,endIndex)

     startIndex=startIndex+1;

  entropyMat = Cell(startIndex:endIndex,EntrpCIndeX);

    entropyMat=sortrows(entropyMat,1);

   [idEnt,iaEnt,icEnt] = unique(entropyMat,'last','legacy');

    Entropy=0;

   for idx=1:length(idEnt)

      p= (nnz(icEnt==idx)/length(icEnt));

       Entropy=Entropy-p*log2(p);

   end

   end
```

Output:

```
Weighted Entropy of Maths:0.908140

Weighted Entropy of Physics:1.395519
```

Weighted entropy of maths is less compared to physics. That means maths requires less information requirement. So, Maths is better predictor for C++ grades