IDS24_HW01

April 17, 2024

1 Question #1

Please complete the code blocks marked with "TODO" from T1 to T8.

1.1 Import packages and datasets

The file 'students.csv' contains information about the students, including their student ID, major, academic year, and email address. * **sid**: student ID * **major**: major * **year**: academic year * **email**: email address

The files 'classA.csv', 'classB.csv', and 'classC.csv' contain the student IDs registered in each classroom, as well as their total scores and the number of hours they spent studying the subject.

- **sid**: student ID
- **study**: study time in hours

[ ]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

path_data = "https://raw.githubusercontent.com/mlee-pnu/IDS/main/HW2/"

students = pd.read_csv(path_data + 'students.csv')
scores_A = pd.read_csv(path_data + 'classA.csv')
scores_B = pd.read_csv(path_data + 'classB.csv')
scores_C = pd.read_csv(path_data + 'classC.csv')
```

[ ]:
```
students
```

[ ]:
```
           sid major  year                  email
0    202060001   CSE     2    202060001@naver.com
1    202260002   CSE     2    202260002@naver.com
2    202260003   CSE     2    202260003@gmail.com
3    202160004   CSE     2    202160004@naver.com
4    202260005   CSE     2    202260005@naver.com
..         …   …   …                    …
134  201762033   ECE     4    201762033@naver.com
```

```
135  201962034   ECE     3    201962034@naver.com
136  201862035   ECE     4    201862035@naver.com
137  201862036   ECE     3    201862036@naver.com
138  201962037   ECE     3  201962037@pusan.ac.kr

[139 rows x 4 columns]
```

[ ]:
```
scores_A.head()
```

[ ]:
```
         sid      score  study
0  202060001  46.274451   9.70
1  202260002  39.669772  12.40
2  202260003  47.743211  14.82
3  202160004  27.737961   6.64
4  202260005  10.780187   6.56
```

Initially, you want to count the number of students for each major.

[ ]:
```
pd.unique(students['major'])
```

[ ]:
```
array(['CSE', 'ARCH', 'BIO', 'NAOE', 'ECE', 'GS', 'GEO', 'CBE', 'BIZ',
       'PHYS'], dtype=object)
```

[ ]:
```
majors = students.groupby('major').size().reset_index(name = 'count')
majors
```

[ ]:
```
  major  count
0  ARCH      1
1   BIO      1
2   BIZ      1
3   CBE      1
4   CSE     91
5   ECE     38
6   GEO      1
7    GS      1
8  NAOE      3
9  PHYS      1
```

1.2 T1. Define a function and applying to a column

You are planning to implement a function that will be applied to each element of the 'major' column. This function will convert the major name to 'Other' if the major is not CSE or ECE.

[ ]:
```
# TODO: complete the following code block
def renameMajor(_major):
  if _major == "CSE":
    return "CSE"
  elif _major == "ECE":
```

```
    return "ECE"
  else:
    return "Other"
```

[ ]:
```
# TODO: apply the function to the column 'major'
students['major'] = students['major'].apply(renameMajor)
students.groupby('major').size().reset_index(name = 'count')
```

[ ]:

	major	count
0	CSE	91
1	ECE	38
2	Other	10

1.3 T2. Applying arithmetic to a column

Oops, a teaching assistant from class B reported that they forgot to include the attendance score. Fortunately, all the students in class B received a score of 5 for attendance, which is the maximum score. Please revise the scores for class B.

[ ]:
```
# TODO: add 5 points to socres of class B
scores_B["score"] += 5
scores_B
```

[ ]:

	sid	score	study
0	202061001	64.339677	12.16
1	202261002	27.650547	17.80
2	202061003	41.275622	20.92
3	202261004	57.095365	11.30
4	202061005	42.025605	15.34
5	201661006	78.281203	24.94
6	202261007	27.127357	17.38
7	202061008	39.150454	9.88
8	202261009	28.662163	4.84
9	202061010	96.260009	21.40
10	202061011	6.945845	3.18
11	202161012	35.939875	17.08
12	202261013	29.136651	5.08
13	202061014	71.651618	19.42
14	202261015	43.435342	19.38
15	202061016	68.644525	13.72
16	202061017	48.362379	19.72
17	202261018	49.353219	12.32
18	202261019	72.890577	15.44
19	202261020	47.485410	22.32
20	202261021	64.328125	16.78
21	201861022	45.851487	10.32
22	202261023	6.056184	10.34
23	202061024	33.776093	8.30

```
24  202261025  70.784248  15.86
25  201861026  13.181276   8.52
26  202261027  46.032495  10.24
27  202261028  43.153127  14.30
28  202261029  52.011593  15.94
29  202061030  40.543979  12.80
30  202261031  46.996757  11.88
31  202261032  30.862428  10.54
32  201761033  48.925224  21.94
33  201961034  36.956914  14.70
34  201861035  21.784545  13.30
35  201861036  13.696730  10.98
36  201761037  53.597309  15.04
37  201761038  54.556528  18.36
38  201961039  11.751692   6.42
39  201961040  67.238499  13.58
40  201761041  49.758011  22.16
41  201861042  68.850835  26.22
42  201861043  31.928239  10.88
43  201861044  36.186316  14.10
44  201761045  57.407820  14.76
45  201861046  32.479875  18.02
46  201961047  46.702518  21.06
47  202161048  40.962385   9.68
48  202161049  69.344486  21.24
```

1.4 T3. Joining Tables by Columns

First, you need to concatenate the scores of all the classes.

[ ]:
```
# add an additional column to indicate the class division
scores_A['div'] = 'Class_A'
scores_B['div'] = 'Class_B'
scores_C['div'] = 'Class_C'

# concatenate three tables. Note that those tables consist of the same columns
scores = pd.concat([scores_A, scores_B, scores_C])
scores
```

[ ]:
```
          sid      score  study      div
0   202060001  46.274451   9.70  Class_A
1   202260002  39.669772  12.40  Class_A
2   202260003  47.743211  14.82  Class_A
3   202160004  27.737961   6.64  Class_A
4   202260005  10.780187   6.56  Class_A
..        …        …    …       …
32  201762033  78.310276  19.64  Class_C
```

```
33  201962034  58.429588  17.94  Class_C
34  201862035  50.774370  20.08  Class_C
35  201862036  71.981646  15.06  Class_C
36  201962037  71.600689  22.66  Class_C

[139 rows x 4 columns]
```

Now, you want to join the two tables using the shared column 'sid'.

[ ]:
```
# TODO: Join the two tables, scores and students
total = pd.merge(scores, students, on = 'sid')
total
```

[ ]:
```
           sid      score  study      div major  year                  email
0    202060001  46.274451   9.70  Class_A   CSE     2    202060001@naver.com
1    202260002  39.669772  12.40  Class_A   CSE     2    202260002@naver.com
2    202260003  47.743211  14.82  Class_A   CSE     2    202260003@gmail.com
3    202160004  27.737961   6.64  Class_A   CSE     2    202160004@naver.com
4    202260005  10.780187   6.56  Class_A   CSE     2    202260005@naver.com
..         …        …    …       …   …   …                    …
134  201762033  78.310276  19.64  Class_C   ECE     4    201762033@naver.com
135  201962034  58.429588  17.94  Class_C   ECE     3    201962034@naver.com
136  201862035  50.774370  20.08  Class_C   ECE     4    201862035@naver.com
137  201862036  71.981646  15.06  Class_C   ECE     3    201862036@naver.com
138  201962037  71.600689  22.66  Class_C   ECE     3  201962037@pusan.ac.kr

[139 rows x 7 columns]
```

1.5 T4. Draw a histogram

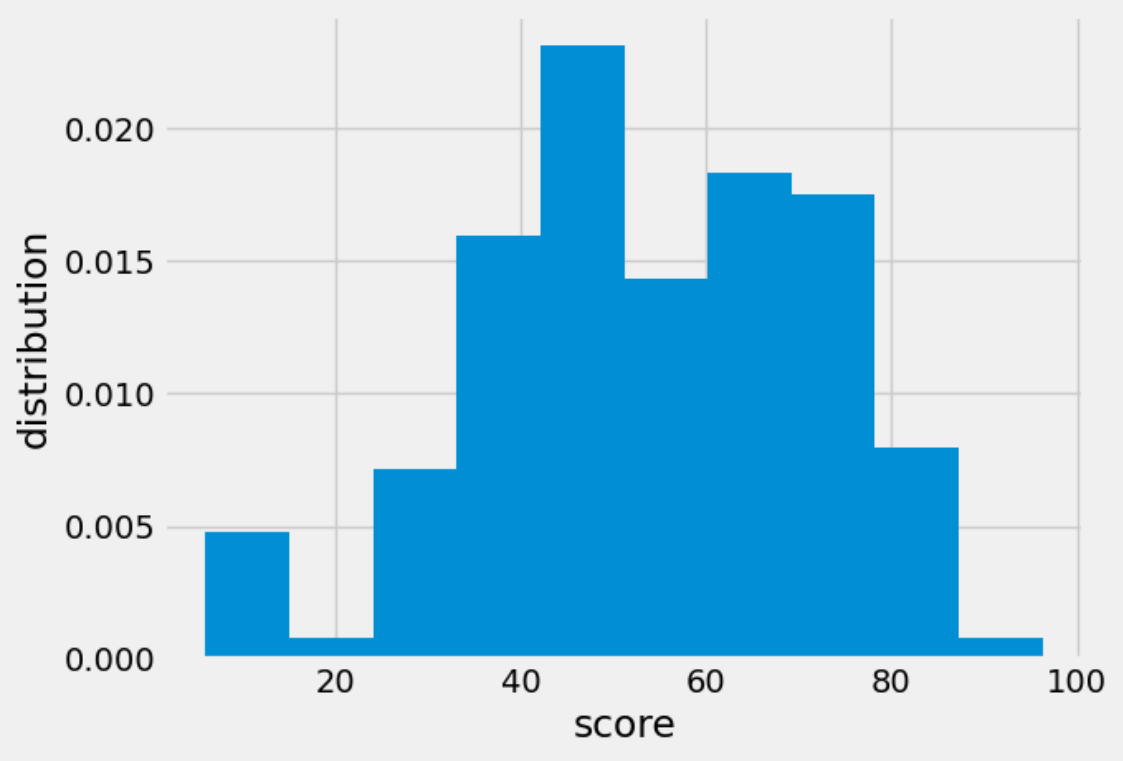
[ ]:
```
# Check the range
max(total['score']), min(total['score'])
```

[ ]:
```
(96.26000935, 6.056183983)
```

To visualize the distribution of the scores, you have decided to create a histogram with 10 equally-sized bins ranging from 0 to 100 points.

[ ]:
```
# TODO: draw a histogram with density scale
total["score"].plot.hist(xlabel="score", ylabel="distribution", density=True,␣
↪bins=10)
```

[ ]:
```
<Axes: xlabel='score', ylabel='distribution'>
```



Can you tell how many students fall into the bin ranging [50, 60)?

[ ]:
```
0.015 * 10 * len(total['score'])
```

[ ]:
```
20.849999999999998
```

[ ]:
```
len(total[ (50<=total['score']) & (total['score']<60)])
```

[ ]:
```
21
```

1.6 T5. Adding a grade column

[ ]:
```
# Check the cutoff scores for the top 25%, 50%, 75% of scores.
np.quantile(total['score'], [0.75, 0.5, 0.25])
```

[ ]:
```
array([68.73141863, 52.32727968, 41.65061315])
```

Based on the scores obtained, you have decided to assign grades generously as follows:

- A: [68, 100]
- B: [52, 68)
- C: [41, 52)
- D: [0, 41)

[ ]:
```
# TODO: Please add a new column named 'grade' and assign the grades based on the criteria provided above.

def assignGrade(_score):
  if _score >= 68:
    return "A"
  elif _score >= 52:
    return "B"
  elif _score >= 41:
    return "C"
  else:
    return "F"

total['grade'] = total['score'].apply(assignGrade)
```

[ ]:
```
total
```

[ ]:

	sid	score	study	div	major	year	email	\
0	202060001	46.274451	9.70	Class_A	CSE	2	202060001@naver.com	
1	202260002	39.669772	12.40	Class_A	CSE	2	202260002@naver.com	
2	202260003	47.743211	14.82	Class_A	CSE	2	202260003@gmail.com	
3	202160004	27.737961	6.64	Class_A	CSE	2	202160004@naver.com	
4	202260005	10.780187	6.56	Class_A	CSE	2	202260005@naver.com	
..	…	…	…	…	…	…	…	
134	201762033	78.310276	19.64	Class_C	ECE	4	201762033@naver.com	
135	201962034	58.429588	17.94	Class_C	ECE	3	201962034@naver.com	
136	201862035	50.774370	20.08	Class_C	ECE	4	201862035@naver.com	
137	201862036	71.981646	15.06	Class_C	ECE	3	201862036@naver.com	
138	201962037	71.600689	22.66	Class_C	ECE	3	201962037@pusan.ac.kr	

	grade
0	C
1	F
2	C
3	F
4	F
..	…
134	A
135	B
136	C
137	A
138	A

[139 rows x 8 columns]

Now you want to check the distribution of grades.

[ ]:
```
grades = total.groupby('grade').size().reset_index(name = 'count')
grades.sort_values('grade', ascending=False).reset_index(drop=True)
```

[ ]:

	grade	count
0	F	33
1	C	35
2	B	35
3	A	36

1.7 T6. The mean score for each academic year

Calculate the mean score for each academic year using the 'groupby' method.

[ ]:
```
# TODO:

total.groupby("year").agg({"score": "mean"}).reset_index()

# total
```

[ ]:

	year	score
0	1	55.662441
1	2	54.612547
2	3	54.483818
3	4	50.842452

1.8 T7. Generate a pivot table

Now you want to see the distribution of the grades by class division.

[ ]:
```
# select only required
division_and_grade = total[['sid','div','grade']]
division_and_grade.head()
```

[ ]:

	sid	div	grade
0	202060001	Class_A	C
1	202260002	Class_A	F
2	202260003	Class_A	C
3	202160004	Class_A	F
4	202260005	Class_A	F

[ ]:
```
# TODO: generate a table using pd.pivot_table
# Use 'grade' as index and 'div' as column

my_pivot = pd.pivot_table(division_and_grade, index='grade', columns='div',␣
 ↪aggfunc=len, fill_value=0)

# TODO: sort the values of 'grade' in descending order

```

```
my_pivot.sort_values('grade', ascending=False, inplace=True)
my_pivot
```

[ ]:
```
          sid
div   Class_A Class_B Class_C
grade
F          12      20       1
C          19      13       3
B          13       8      14
A           9       8      19
```

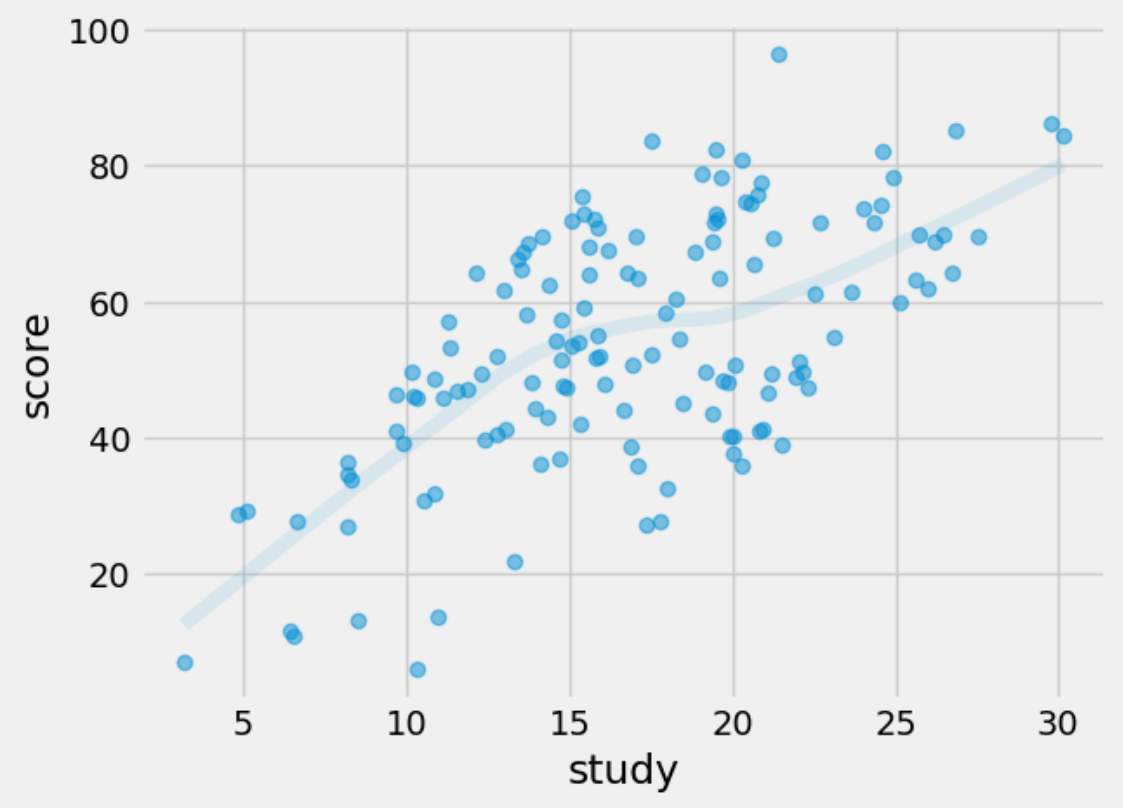
1.9 T8. Drawing a scatter plot

Our dataset, 'total', contains study hours along with the score. Draw a scatter plot to see an association between the two variables.

[ ]:
```
# TODO: Draw a scatter plot to see the relation between study time and scores
import seaborn as sns

# sns.scatterplot(data=total, x='study', y='score', alpha=.5)
sns.regplot(data=total, x='study', y='score', lowess=True, scatter_kws={'alpha':
 ↪0.5}, line_kws={'alpha':0.1})
```

[ ]:
```
<Axes: xlabel='study', ylabel='score'>
```



Let's assume that the scatter plot shows a moderate positive correlation between study time and score.

Does this mean that spending more time studying the subject will result in a higher score?

Ask yourself whether the data we've explored so far were collected from a randomized controlled experiment.