ds-exercise1-b1-ui22cs03

August 9, 2024

1 Excercise 1 by UI22CS03

1.1 Objectives

- Import Libraries
- Lab Exercises
  - Identifying duplicates
  - Plotting Scatterplots

1.2 Import Libraries

Import the libraries we need

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the csv file

[3]:
```
data = pd.read_csv("TeachingRatings_ui22cs03.csv")
print(data)
```

```
     rownames minority  age  gender credits    beauty  eval division native  \
0           1      yes   36  female    more  0.289916   4.3    upper    yes
1           2       no   59    male    more -0.737732   4.5    upper    yes
2           3       no   51    male    more -0.571984   3.7    upper    yes
3           4       no   40  female    more -0.677963   4.3    upper    yes
4           5       no   31  female    more  1.509794   4.4    upper    yes
..        …        …   …     …       …         …   …      …     …
458       459       no   32    male    more  1.231394   3.2    lower    yes
459       460       no   32    male    more  1.231394   4.3    upper    yes
460       461      yes   42  female    more  0.420400   3.3    upper     no
461       462      yes   42  female    more  0.420400   3.2    upper     no
462       463      yes   42  female  single  0.420400   4.1    lower     no

    tenure  students  allstudents  prof
0      yes        24           43     1
```

```
1       yes         17            20     2
2       yes         55            55     3
3       yes         40            46     4
4       yes         42            48     5
..      …        …          …    …
458     yes          9            21    93
459     yes         52            86    93
460     yes         52            67    94
461     yes         54            66    94
462     yes         28            35    94

[463 rows x 13 columns]
```

1.3 Display information about the dataset

1. Structure of the dataframe
2. Describe the dataset
3. Number of rows and columns

[4]:
```python
# 1. Structure of the dataframe
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   rownames     463 non-null    int64
 1   minority     463 non-null    object
 2   age          463 non-null    int64
 3   gender       463 non-null    object
 4   credits      463 non-null    object
 5   beauty       463 non-null    float64
 6   eval         463 non-null    float64
 7   division     463 non-null    object
 8   native       463 non-null    object
 9   tenure       463 non-null    object
 10  students     463 non-null    int64
 11  allstudents  463 non-null    int64
 12  prof         463 non-null    int64
dtypes: float64(2), int64(5), object(6)
memory usage: 47.1+ KB
None
         rownames         age        beauty        eval    students  \
count  463.000000  463.000000  4.630000e+02  463.000000  463.000000
mean   232.000000   48.365011  6.263499e-08    3.998272   36.624190
std    133.800847    9.802742  7.886477e-01    0.554866   45.018481
min      1.000000   29.000000 -1.450494e+00    2.100000    5.000000
```

```
25%    116.500000   42.000000 -6.562689e-01    3.600000   15.000000
50%    232.000000   48.000000 -6.801430e-02    4.000000   23.000000
75%    347.500000   57.000000  5.456024e-01    4.400000   40.000000
max    463.000000   73.000000  1.970023e+00    5.000000  380.000000

       allstudents        prof
count   463.000000  463.000000
mean     55.177106   45.434125
std      75.072800   27.508902
min       8.000000    1.000000
25%      19.000000   20.000000
50%      29.000000   44.000000
75%      60.000000   70.500000
max     581.000000   94.000000
Number of rows and columns: (463, 13)
```

[5]:
```
# 2. Describe the dataset
print(data.describe())
```

```
         rownames         age        beauty        eval    students  \
count  463.000000  463.000000  4.630000e+02  463.000000  463.000000
mean   232.000000   48.365011  6.263499e-08    3.998272   36.624190
std    133.800847    9.802742  7.886477e-01    0.554866   45.018481
min      1.000000   29.000000 -1.450494e+00    2.100000    5.000000
25%    116.500000   42.000000 -6.562689e-01    3.600000   15.000000
50%    232.000000   48.000000 -6.801430e-02    4.000000   23.000000
75%    347.500000   57.000000  5.456024e-01    4.400000   40.000000
max    463.000000   73.000000  1.970023e+00    5.000000  380.000000

       allstudents        prof
count   463.000000  463.000000
mean     55.177106   45.434125
std      75.072800   27.508902
min       8.000000    1.000000
25%      19.000000   20.000000
50%      29.000000   44.000000
75%      60.000000   70.500000
max     581.000000   94.000000
```

[6]:
```
# 3. Number of rows and columns
print("Number of rows and columns:", data.shape)
```

```
Number of rows and columns: (463, 13)
```

[7]:
```
#Head Preview of data : Head means the first 5 datas
data.head()
```

[7]:
```
   rownames minority  age  gender credits    beauty  eval division native  \
0         1      yes   36  female    more  0.289916   4.3    upper    yes
1         2       no   59    male    more -0.737732   4.5    upper    yes
2         3       no   51    male    more -0.571984   3.7    upper    yes
3         4       no   40  female    more -0.677963   4.3    upper    yes
4         5       no   31  female    more  1.509794   4.4    upper    yes

  tenure  students  allstudents  prof
0    yes        24           43     1
1    yes        17           20     2
2    yes        55           55     3
3    yes        40           46     4
4    yes        42           48     5
```

[8]:
```python
#Tail Preview of data : Tail means the last 5 datas
data.tail()
```

[8]:
```
     rownames minority  age  gender credits    beauty  eval division native  \
458       459       no   32    male    more  1.231394   3.2    lower    yes
459       460       no   32    male    more  1.231394   4.3    upper    yes
460       461      yes   42  female    more  0.420400   3.3    upper     no
461       462      yes   42  female    more  0.420400   3.2    upper     no
462       463      yes   42  female  single  0.420400   4.1    lower     no

    tenure  students  allstudents  prof
458    yes         9           21    93
459    yes        52           86    93
460    yes        52           67    94
461    yes        54           66    94
462    yes        28           35    94
```

1.3.1 Identify all duplicate cases using prof variable - find the unique values of the prof variables

[9]:
```python
unique_prof = data['prof'].unique()
print(unique_prof)
```

```
[ 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 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94]
```

1.3.2 Print out the number of unique values in the prof variable

[11]:
```python
print(len(unique_prof))
```

```
94
```

1.3.3 Using all observations, Find the average and standard deviation for age

[12]:
```python
# Using all observations, Find the average and standard deviation for age
print("Average age:", data['age'].mean())
print("Standard deviation of age:", data['age'].std())
```

```
Average age: 48.365010799136066
Standard deviation of age: 9.802742037864817
```

1.3.4 Repeat the analysis by first filtering the data set to include one observation for each instructor with a total number of observations restricted to 94.

[21]:
```python
# Filter the dataset to include one observation per instructor
filtered_data = data.drop_duplicates(subset='prof')

# Restrict the number of observations to 94
filtered_data = filtered_data.head(94)

filtered_data.head()
```

[21]:
```
   rownames minority  age  gender credits    beauty  eval division native  \
0         1      yes   36  female    more  0.289916   4.3    upper    yes
1         2       no   59    male    more -0.737732   4.5    upper    yes
2         3       no   51    male    more -0.571984   3.7    upper    yes
3         4       no   40  female    more -0.677963   4.3    upper    yes
4         5       no   31  female    more  1.509794   4.4    upper    yes

  tenure  students  allstudents  prof
0    yes        24           43     1
1    yes        17           20     2
2    yes        55           55     3
3    yes        40           46     4
4    yes        42           48     5
```

[20]:
```python
filtered_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 94 entries, 0 to 93
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   rownames     94 non-null     int64
 1   minority     94 non-null     object
 2   age          94 non-null     int64
 3   gender       94 non-null     object
 4   credits      94 non-null     object
 5   beauty       94 non-null     float64
 6   eval         94 non-null     float64
```

```
 7   division     94 non-null     object
 8   native       94 non-null     object
 9   tenure       94 non-null     object
 10  students     94 non-null     int64
 11  allstudents  94 non-null     int64
 12  prof         94 non-null     int64
dtypes: float64(2), int64(5), object(6)
memory usage: 10.3+ KB
```

Use the new dataset to get the mean and SD of age

[14]:
```
# Calculate the average and standard deviation of age for the filtered dataset
print("Average age (filtered):", filtered_data['age'].mean())
print("Standard deviation of age (filtered):", filtered_data['age'].std())
```

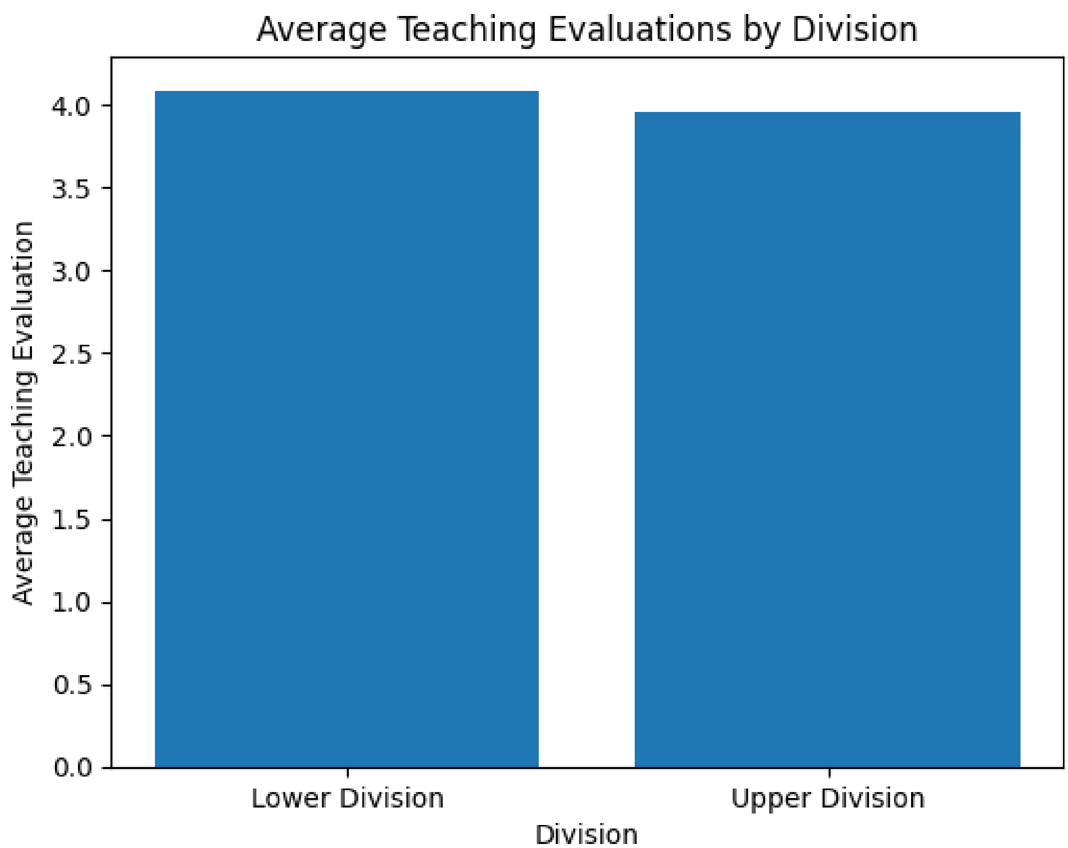
```
Average age (filtered): 47.5531914893617
Standard deviation of age (filtered): 10.25651329515495
```

1.3.5 Using a bar chart, demonstrate if instructors teaching lower-division courses receive higher average teaching evaluations.

Find the average teaching evaluation in both groups of upper and lower-division

Plot the barplot

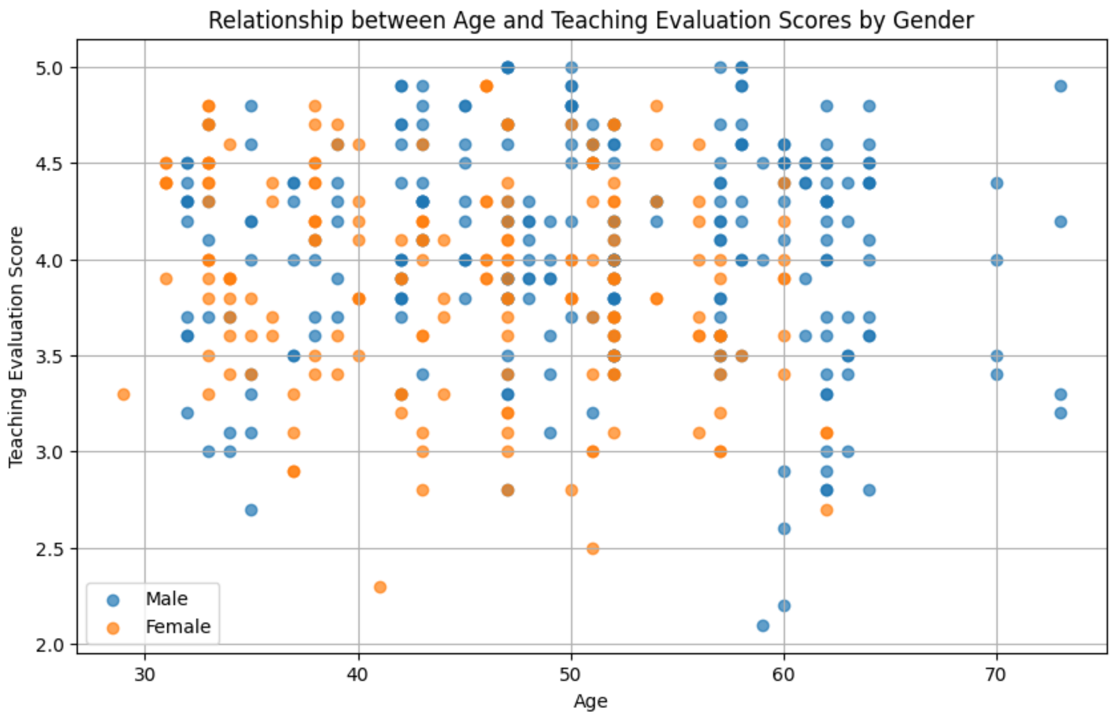
[22]:
```
# Find the average teaching evaluation in both groups of upper and␣
↪lower-division

# Calculate the average teaching evaluation for lower-division courses
lower_division_mean = data[data['division'] == 'lower']['eval'].mean()

# Calculate the average teaching evaluation for upper-division courses
upper_division_mean = data[data['division'] == 'upper']['eval'].mean()

# Create a bar chart
plt.bar(['Lower Division', 'Upper Division'], [lower_division_mean,␣
↪upper_division_mean])
plt.xlabel('Division')
plt.ylabel('Average Teaching Evaluation')
plt.title('Average Teaching Evaluations by Division')
plt.show()
```



1.3.6 Using gender-differentiated scatter plots, plot the relationship between age and teaching evaluation scores.

[23]:
```
# Create separate DataFrames for male and female instructors
male_instructors = data[data['gender'] == 'male']
female_instructors = data[data['gender'] == 'female']

# Create scatter plots for male and female instructors
plt.figure(figsize=(10, 6))

plt.scatter(male_instructors['age'], male_instructors['eval'], label='Male',␣
 ↪alpha=0.7)
plt.scatter(female_instructors['age'], female_instructors['eval'],␣
 ↪label='Female', alpha=0.7)

plt.xlabel('Age')
plt.ylabel('Teaching Evaluation Score')
plt.title('Relationship between Age and Teaching Evaluation Scores by Gender')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



1.3.7 What is the number of courses taught by gender?

[26]:
```
# Group the data by gender and count the number of courses
courses_by_gender = data.groupby('gender')['division'].count()

# Print the results
print(courses_by_gender)
```

```
gender
female    195
male      268
Name: division, dtype: int64
```

1.3.8 Create a group histogram of taught by gender and tenure

[27]:
```
# Group the data by gender and tenure, then count the number of courses
courses_by_gender_tenure = data.groupby(['gender', 'tenure'])['division'].
 ↪count().unstack()

# Create a grouped histogram
courses_by_gender_tenure.plot(kind='bar', figsize=(10, 6))
```

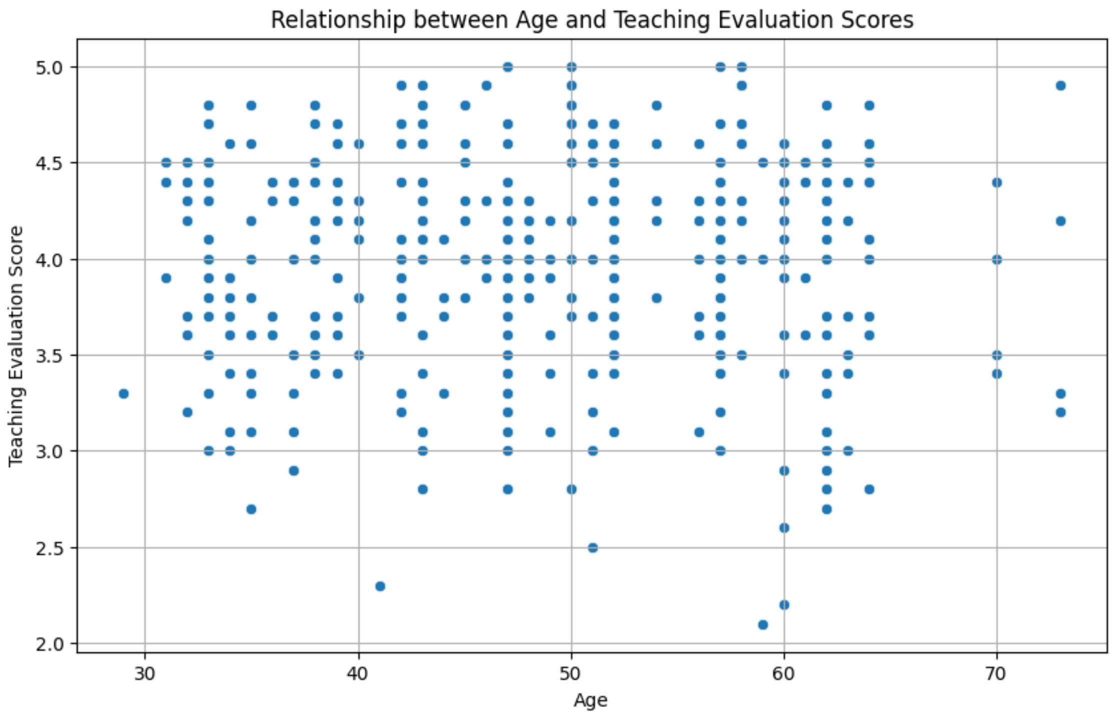
```
plt.xlabel('Gender')
plt.ylabel('Number of Courses')
plt.title('Number of Courses Taught by Gender and Tenure')
plt.legend(title='Tenure')
plt.show()
```

1.3.9 Does age affect teaching evaluation scores?

[28]:
```
# Calculate the correlation coefficient between age and teaching evaluation␣
↪scores
correlation = data['age'].corr(data['eval'])

print("Correlation between age and teaching evaluation scores:", correlation)

# Create a scatter plot to visualize the relationship
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='eval', data=data)
plt.xlabel('Age')
plt.ylabel('Teaching Evaluation Score')
plt.title('Relationship between Age and Teaching Evaluation Scores')
plt.grid(True)
plt.show()
```

```python
# Perform a simple linear regression to model the relationship
import statsmodels.api as sm

X = data['age']
y = data['eval']
X = sm.add_constant(X)  # Add a constant to the model

model = sm.OLS(y, X).fit()
print(model.summary())
```

```
Correlation between age and teaching evaluation scores: -0.051696190877182864
```




```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   eval   R-squared:                       0.003
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.235
Date:                Fri, 09 Aug 2024   Prob (F-statistic):              0.267
Time:                        07:21:20   Log-Likelihood:                -383.13
No. Observations:                 463   AIC:                             770.3
Df Residuals:                     461   BIC:                             778.5
Df Model:                           1
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.1398      0.130     31.865      0.000       3.884       4.395
age           -0.0029      0.003     -1.111      0.267      -0.008       0.002
==============================================================================
Omnibus:                       14.980   Durbin-Watson:                   1.344
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               15.952
Skew:                          -0.449   Prob(JB):                     0.000344
Kurtosis:                       2.857   Cond. No.                         249.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Calculate the percentage of visible minorities are tenure professors. Will you say that tenure status differed if teacher was a visible minority?

[29]:
```
# Calculate the percentage of visible minorities who are tenure professors
visible_minority_tenure = data[(data['minority'] == 'yes') & (data['tenure'] ==␣
 ↪'yes')].shape[0]
visible_minority_total = data[data['minority'] == 'yes'].shape[0]
percentage_visible_minority_tenure = (visible_minority_tenure /␣
 ↪visible_minority_total) * 100

print("Percentage of visible minorities who are tenure professors:",␣
 ↪percentage_visible_minority_tenure)

# Calculate the percentage of non-visible minorities who are tenure professors
non_visible_minority_tenure = data[(data['minority'] == 'no') & (data['tenure']␣
 ↪== 'yes')].shape[0]
non_visible_minority_total = data[data['minority'] == 'no'].shape[0]
percentage_non_visible_minority_tenure = (non_visible_minority_tenure /␣
 ↪non_visible_minority_total) * 100

print("Percentage of non-visible minorities who are tenure professors:",␣
 ↪percentage_non_visible_minority_tenure)

# Compare the percentages
if percentage_visible_minority_tenure != percentage_non_visible_minority_tenure:
  print("Tenure status appears to differ based on whether the teacher is a␣
 ↪visible minority.")
else:
  print("Tenure status does not appear to differ based on whether the teacher␣
 ↪is a visible minority.")
```

```
Percentage of visible minorities who are tenure professors: 84.375
```

```
Percentage of non-visible minorities who are tenure professors:
76.94235588972431
Tenure status appears to differ based on whether the teacher is a visible
minority.
```

Does average age differ by tenure? Produce the means and standard deviations for both tenured and untenured professors.

[30]:
```python
# Calculate the average and standard deviation of age for tenured professors
tenured_age_mean = data[data['tenure'] == 'yes']['age'].mean()
tenured_age_std = data[data['tenure'] == 'yes']['age'].std()

# Calculate the average and standard deviation of age for untenured professors
untenured_age_mean = data[data['tenure'] == 'no']['age'].mean()
untenured_age_std = data[data['tenure'] == 'no']['age'].std()

# Print the results
print("Average age of tenured professors:", tenured_age_mean)
print("Standard deviation of age for tenured professors:", tenured_age_std)

print("\nAverage age of untenured professors:", untenured_age_mean)
print("Standard deviation of age for untenured professors:", untenured_age_std)
```

```
Average age of tenured professors: 47.850415512465375
Standard deviation of age for tenured professors: 10.420055773692855

Average age of untenured professors: 50.18627450980392
Standard deviation of age for untenured professors: 6.946372216733221
```

2 CODE BY ADITYA KUMAR (UI22CS03)