DSBDA Practical No.03

May 19, 2023

[4]:
```
import pandas as pd
```

[5]:
```
import numpy as np
```

[6]:
```
student = pd.read_csv("/Users/shreyaspeherkar/Desktop/Dataset/
 ↪StudentsPerformance.csv")
```

[7]:
```
student.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test_preparation_course      1000 non-null   object
 5   math_score                   991 non-null    float64
 6   reading_score                995 non-null    float64
 7   writing_score                994 non-null    float64
dtypes: float64(3), object(5)
memory usage: 62.6+ KB
```

[8]:
```
student.describe()
```

[8]:
```
       math_score  reading_score  writing_score
count  991.000000     995.000000     994.000000
mean    66.116044      69.223116      68.113682
std     15.217867      14.577775      15.182945
min      0.000000      17.000000      10.000000
25%     57.000000      59.000000      58.000000
50%     66.000000      70.000000      69.000000
75%     77.000000      79.000000      79.000000
max    100.000000     100.000000     100.000000
```

[9]:
```
student.head()
```

[9]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test_preparation_course  math_score  reading_score  writing_score
0                    none        72.0           72.0           74.0
1               completed        69.0           90.0           88.0
2                    none        90.0           95.0           93.0
3                    none        47.0           57.0           44.0
4                    none        76.0           78.0           75.0
```

[10]:
```python
male_female = student.groupby('gender')['gender'].count()
print(male_female)
```

```
gender
female    518
male      482
Name: gender, dtype: int64
```

[11]:
```python
student.test_preparation_course.unique()
```

[11]:
```
array(['none', 'completed'], dtype=object)
```

[12]:
```python
mean_math = student.groupby('gender').math_score.mean()
```

[13]:
```python
print(mean_math)
```

```
gender
female    63.654902
male      68.725572
Name: math_score, dtype: float64
```

[14]:
```python
mean_math_test_preparation = student.
  ↪groupby(['gender','test_preparation_course']).math_score.mean()
```

[15]:
```python
print(mean_math_test_preparation)
```

```
gender  test_preparation_course
female  completed                  67.331492
        none                       61.632219
male    completed                  72.339080
        none                       66.677524
Name: math_score, dtype: float64
```

[16]:
```
mean_math_test_preparation = student.groupby(['gender','race/ethnicity']).
↪math_score.mean()
```

[17]:
```
print(mean_math_test_preparation)
```

```
gender  race/ethnicity
female  group A           58.514286
        group B           61.450980
        group C           61.988764
        group D           65.236220
        group E           71.014706
male    group A           63.735849
        group B           65.882353
        group C           67.611511
        group D           69.413534
        group E           76.746479
Name: math_score, dtype: float64
```

[18]:
```
student.math_score.unique()
```

[18]:
```
array([ 72.,  69.,  90.,  47.,  76.,  71.,  88.,  40.,  64.,  38.,  58.,
        nan,  78.,  50.,  18.,  46.,  54.,  66.,  65.,  44.,  74.,  73.,
        70.,  62.,  63.,  56.,  97.,  81.,  75.,  57.,  55.,  53.,  59.,
        82.,  77.,  33.,  52.,   0.,  79.,  39.,  67.,  45.,  60.,  61.,
        41.,  49.,  30.,  80.,  42.,  27.,  43.,  68.,  85.,  98.,  87.,
        51.,  99.,  84.,  91.,  83.,  89.,  22., 100.,  96.,  94.,  48.,
        35.,  34.,  86.,  92.,  37.,  28.,  24.,  26.,  95.,  36.,  29.,
        32.,  93.,  19.,  23.,   8.])
```

[19]:
```
#Group by of a Single Column and Apply the describe() Method on a Single Column
```

[20]:
```
print(student.groupby('gender').math_score.describe())
```

```
        count       mean        std   min   25%   50%   75%    max
gender
female  510.0  63.654902  15.593640   0.0  54.0  65.0  74.0  100.0
male    481.0  68.725572  14.371106  27.0  59.0  69.0  79.0  100.0
```

[21]:
```
groups = pd.cut(student['math_score'],bins=4)
groups
```

[21]:
```
0       (50.0, 75.0]
1       (50.0, 75.0]
2      (75.0, 100.0]
3       (25.0, 50.0]
4      (75.0, 100.0]
           …
```

```
995    (75.0, 100.0]
996     (50.0, 75.0]
997     (50.0, 75.0]
998     (50.0, 75.0]
999    (75.0, 100.0]
Name: math_score, Length: 1000, dtype: category
Categories (4, interval[float64, right]): [(-0.1, 25.0] < (25.0, 50.0] < (50.0,
75.0] < (75.0, 100.0]]
```

[22]:
```
groups = pd.cut(student['math_score'],bins=5)
groups
```

[22]:
```
0       (60.0, 80.0]
1       (60.0, 80.0]
2      (80.0, 100.0]
3       (40.0, 60.0]
4       (60.0, 80.0]
           …
995    (80.0, 100.0]
996     (60.0, 80.0]
997     (40.0, 60.0]
998     (60.0, 80.0]
999     (60.0, 80.0]
Name: math_score, Length: 1000, dtype: category
Categories (5, interval[float64, right]): [(-0.1, 20.0] < (20.0, 40.0] < (40.0,
60.0] < (60.0, 80.0] < (80.0, 100.0]]
```

[23]:
```
student.groupby(groups)['math_score'].count()
```

[23]:
```
math_score
(-0.1, 20.0]       4
(20.0, 40.0]      46
(40.0, 60.0]     285
(60.0, 80.0]     480
(80.0, 100.0]    176
Name: math_score, dtype: int64
```

[24]:
```
pd.crosstab(groups, student['gender'])
```

[24]:
```
gender         female  male
math_score
(-0.1, 20.0]        4     0
(20.0, 40.0]       32    14
(40.0, 60.0]      165   120
(60.0, 80.0]      241   239
(80.0, 100.0]      68   108
[25]: #Write a Python program to display some basic statistical details like␣
       ↪percentile, mean, standard deviation etc.
      #of the species of ‘Iris-setosa’, ‘Iris- versicolor’ and 'Iris-versicolor’ of␣
       ↪iris.csv dataset.
```

```
[26]: import statistics as st
```

```
[27]: data = [1,2,3,4,5,6]
```

```
[28]: st.mean(data)
```

```
[28]: 3.5
```

```
[29]: st.median(data)
```

```
[29]: 3.5
```

```
[30]: #Will show error as data is having no unique modal value
      st.mode(data)
```

```
[30]: 1
```

```
[31]: data1 = [1,2,7,5,4,7,8,2,1,7]
      st.mode(data1)
```

```
[31]: 7
```

```
[32]: #Variance
      st.variance(data1)
```

```
[32]: 7.6
```

```
[33]: import pandas as pd
      df = pd.DataFrame(data1)
```

```
[34]: df.mean()
```

```
[34]: 0    4.4
      dtype: float64
```

```
[35]: df.mode()
```

```
[35]:    0
      0  7
```

```
[36]: df.median()
```

[36]:
```
0    4.5
dtype: float64
```

[37]:
```
#using California housing csv file
df1 = pd.read_csv("/Users/shreyaspeherkar/Desktop/Dataset/housing.csv")
df1
```

[37]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           190.0
3        -122.25     37.85                52.0       1274.0           235.0
4        -122.25     37.85                52.0       1627.0           280.0
…            …         …                   …           …               …
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income  median_house_value  \
0           322.0       126.0         8.3252            452600.0
1          2401.0      1138.0         8.3014            358500.0
2           496.0       177.0         7.2574            352100.0
3           558.0       219.0         5.6431            341300.0
4           565.0       259.0         3.8462            342200.0
…             …           …              …                  …
20635       845.0       330.0         1.5603             78100.0
20636       356.0       114.0         2.5568             77100.0
20637      1007.0       433.0         1.7000             92300.0
20638       741.0       349.0         1.8672             84700.0
20639      1387.0       530.0         2.3886             89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…                 …
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

[38]:
```
df1.mean()
```

```
/var/folders/cs/hplqvnxd09bg_bgmf6zh8t3m0000gn/T/ipykernel_2105/2053335143.py:1:
FutureWarning: The default value of numeric_only in DataFrame.mean is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  df1.mean()
```

[38]:
```
longitude                -119.569704
latitude                   35.631861
housing_median_age         28.639486
total_rooms              2635.763081
total_bedrooms            537.870553
population               1425.476744
households                499.539680
median_income               3.870671
median_house_value     206855.816909
dtype: float64
```

[39]:
```
df1["households"].mean()
```

[39]:
```
499.5396802325581
```

[40]:
```
df1["households"].median()
```

[40]:
```
409.0
```

[41]:
```
df1["households"].mode()
```

[41]:
```
0    306.0
Name: households, dtype: float64
```

[42]:
```
df1["households"].var()
```

[42]:
```
146176.03990028054
```

[43]:
```
st.stdev(df1["households"])
```

[43]:
```
382.3297528316107
```

[44]:
```
#Descriptive Statistics on IRIS dataset
```

[45]:
```
import pandas as pd
data = pd.read_csv("/Users/shreyaspeherkar/Desktop/Dataset/iris.csv")
print('Iris-setosa')
```

```
Iris-setosa
```

[46]:
```
setosa = data['species'] == 'Iris-setosa'
print(data[setosa].describe())
```

```
       sepal_length  sepal_width  petal_length  petal_width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```

[47]:
```
print('\nIris-versicolor')
setosa = data['species'] == 'Iris-versicolor'
print(data[setosa].describe())
```

```

Iris-versicolor
       sepal_length  sepal_width  petal_length  petal_width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```

[48]:
```
print('\nIris-virginica')
setosa = data['species'] == 'Iris-virginica'
print(data[setosa].describe())
```

```

Iris-virginica
       sepal_length  sepal_width  petal_length  petal_width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```