KEY_Practice15_Basic_Stats_I_Averages

May 25, 2020

1 Practice: Basic Statistics I: Averages

For this practice, let's use the Boston dataset.

[118]:
```
# Import the numpy package so that we can use the method mean to calculate␣
↪averages
import numpy as np
```

[119]:
```
# Import the load_boston method
from sklearn.datasets import load_boston
```

[120]:
```
# Import pandas, so that we can work with the data frame version of the Boston␣
↪data
import pandas as pd
```

[121]:
```
# Load the Boston data
boston = load_boston()
```

[122]:
```
# This will provide the characteristics for the Boston dataset
print(boston.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000
sq.ft.
```

```
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by
town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at Carnegie
Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
…', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that
address regression
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential
Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-243,
University of Massachusetts, Amherst. Morgan Kaufmann.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

[123]:
```python
# Here, I'm including the prices of Boston's houses, which is boston['target'],␣
↪as a column with the other
# features in the Boston dataset.
boston_data = np.concatenate((boston['data'], pd.DataFrame(boston['target'])),␣
↪axis = 1)
```

[124]:
```python
# Convert the Boston data to a data frame format, so that it's easier to view␣
↪and process
boston_df = pd.DataFrame(boston_updated, columns = np.
↪concatenate((boston['feature_names'], 'MEDV'), axis = None))
boston_df
```

[124]:
```
         CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS   RAD    TAX  \
0     0.00632  18.0   2.31   0.0  0.538  6.575   65.2  4.0900   1.0  296.0
1     0.02731   0.0   7.07   0.0  0.469  6.421   78.9  4.9671   2.0  242.0
2     0.02729   0.0   7.07   0.0  0.469  7.185   61.1  4.9671   2.0  242.0
3     0.03237   0.0   2.18   0.0  0.458  6.998   45.8  6.0622   3.0  222.0
4     0.06905   0.0   2.18   0.0  0.458  7.147   54.2  6.0622   3.0  222.0
5     0.02985   0.0   2.18   0.0  0.458  6.430   58.7  6.0622   3.0  222.0
6     0.08829  12.5   7.87   0.0  0.524  6.012   66.6  5.5605   5.0  311.0
7     0.14455  12.5   7.87   0.0  0.524  6.172   96.1  5.9505   5.0  311.0
8     0.21124  12.5   7.87   0.0  0.524  5.631  100.0  6.0821   5.0  311.0
9     0.17004  12.5   7.87   0.0  0.524  6.004   85.9  6.5921   5.0  311.0
10    0.22489  12.5   7.87   0.0  0.524  6.377   94.3  6.3467   5.0  311.0
11    0.11747  12.5   7.87   0.0  0.524  6.009   82.9  6.2267   5.0  311.0
12    0.09378  12.5   7.87   0.0  0.524  5.889   39.0  5.4509   5.0  311.0
13    0.62976   0.0   8.14   0.0  0.538  5.949   61.8  4.7075   4.0  307.0
14    0.63796   0.0   8.14   0.0  0.538  6.096   84.5  4.4619   4.0  307.0
15    0.62739   0.0   8.14   0.0  0.538  5.834   56.5  4.4986   4.0  307.0
16    1.05393   0.0   8.14   0.0  0.538  5.935   29.3  4.4986   4.0  307.0
17    0.78420   0.0   8.14   0.0  0.538  5.990   81.7  4.2579   4.0  307.0
18    0.80271   0.0   8.14   0.0  0.538  5.456   36.6  3.7965   4.0  307.0
19    0.72580   0.0   8.14   0.0  0.538  5.727   69.5  3.7965   4.0  307.0
20    1.25179   0.0   8.14   0.0  0.538  5.570   98.1  3.7979   4.0  307.0
21    0.85204   0.0   8.14   0.0  0.538  5.965   89.2  4.0123   4.0  307.0
22    1.23247   0.0   8.14   0.0  0.538  6.142   91.7  3.9769   4.0  307.0
23    0.98843   0.0   8.14   0.0  0.538  5.813  100.0  4.0952   4.0  307.0
24    0.75026   0.0   8.14   0.0  0.538  5.924   94.1  4.3996   4.0  307.0
25    0.84054   0.0   8.14   0.0  0.538  5.599   85.7  4.4546   4.0  307.0
26    0.67191   0.0   8.14   0.0  0.538  5.813   90.3  4.6820   4.0  307.0
27    0.95577   0.0   8.14   0.0  0.538  6.047   88.8  4.4534   4.0  307.0
28    0.77299   0.0   8.14   0.0  0.538  6.495   94.4  4.4547   4.0  307.0
29    1.00245   0.0   8.14   0.0  0.538  6.674   87.3  4.2390   4.0  307.0
..        …    …     …     …      …      …      …       …     …      …
476   4.87141   0.0  18.10   0.0  0.614  6.484   93.6  2.3053  24.0  666.0
477  15.02340   0.0  18.10   0.0  0.614  5.304   97.3  2.1007  24.0  666.0
478  10.23300   0.0  18.10   0.0  0.614  6.185   96.7  2.1705  24.0  666.0
```

479	14.33370	0.0	18.10	0.0	0.614	6.229	88.0	1.9512	24.0	666.0
480	5.82401	0.0	18.10	0.0	0.532	6.242	64.7	3.4242	24.0	666.0
481	5.70818	0.0	18.10	0.0	0.532	6.750	74.9	3.3317	24.0	666.0
482	5.73116	0.0	18.10	0.0	0.532	7.061	77.0	3.4106	24.0	666.0
483	2.81838	0.0	18.10	0.0	0.532	5.762	40.3	4.0983	24.0	666.0
484	2.37857	0.0	18.10	0.0	0.583	5.871	41.9	3.7240	24.0	666.0
485	3.67367	0.0	18.10	0.0	0.583	6.312	51.9	3.9917	24.0	666.0
486	5.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0
487	4.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0
488	0.15086	0.0	27.74	0.0	0.609	5.454	92.7	1.8209	4.0	711.0
489	0.18337	0.0	27.74	0.0	0.609	5.414	98.3	1.7554	4.0	711.0
490	0.20746	0.0	27.74	0.0	0.609	5.093	98.0	1.8226	4.0	711.0
491	0.10574	0.0	27.74	0.0	0.609	5.983	98.8	1.8681	4.0	711.0
492	0.11132	0.0	27.74	0.0	0.609	5.983	83.5	2.1099	4.0	711.0
493	0.17331	0.0	9.69	0.0	0.585	5.707	54.0	2.3817	6.0	391.0
494	0.27957	0.0	9.69	0.0	0.585	5.926	42.6	2.3817	6.0	391.0
495	0.17899	0.0	9.69	0.0	0.585	5.670	28.8	2.7986	6.0	391.0
496	0.28960	0.0	9.69	0.0	0.585	5.390	72.9	2.7986	6.0	391.0
497	0.26838	0.0	9.69	0.0	0.585	5.794	70.6	2.8927	6.0	391.0
498	0.23912	0.0	9.69	0.0	0.585	6.019	65.3	2.4091	6.0	391.0
499	0.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6.0	391.0
500	0.22438	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6.0	391.0
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0

	PTRATIO	B	LSTAT	MEDV
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2
5	18.7	394.12	5.21	28.7
6	15.2	395.60	12.43	22.9
7	15.2	396.90	19.15	27.1
8	15.2	386.63	29.93	16.5
9	15.2	386.71	17.10	18.9
10	15.2	392.52	20.45	15.0
11	15.2	396.90	13.27	18.9
12	15.2	390.50	15.71	21.7
13	21.0	396.90	8.26	20.4
14	21.0	380.02	10.26	18.2
15	21.0	395.62	8.47	19.9
16	21.0	386.85	6.58	23.1
17	21.0	386.75	14.67	17.5

18	21.0	288.99	11.69	20.2
19	21.0	390.95	11.28	18.2
20	21.0	376.57	21.02	13.6
21	21.0	392.53	13.83	19.6
22	21.0	396.90	18.72	15.2
23	21.0	394.54	19.88	14.5
24	21.0	394.33	16.30	15.6
25	21.0	303.42	16.51	13.9
26	21.0	376.88	14.81	16.6
27	21.0	306.38	17.28	14.8
28	21.0	387.94	12.80	18.4
29	21.0	380.23	11.98	21.0
..	…	…	…	…
476	20.2	396.21	18.68	16.7
477	20.2	349.48	24.91	12.0
478	20.2	379.70	18.03	14.6
479	20.2	383.32	13.11	21.4
480	20.2	396.90	10.74	23.0
481	20.2	393.07	7.74	23.7
482	20.2	395.28	7.01	25.0
483	20.2	392.92	10.42	21.8
484	20.2	370.73	13.34	20.6
485	20.2	388.62	10.58	21.2
486	20.2	392.68	14.98	19.1
487	20.2	388.22	11.45	20.6
488	20.1	395.09	18.06	15.2
489	20.1	344.05	23.97	7.0
490	20.1	318.43	29.68	8.1
491	20.1	390.11	18.07	13.6
492	20.1	396.90	13.35	20.1
493	19.2	396.90	12.01	21.8
494	19.2	396.90	13.59	24.5
495	19.2	393.29	17.60	23.1
496	19.2	396.90	21.14	19.7
497	19.2	396.90	14.10	18.3
498	19.2	396.90	12.92	21.2
499	19.2	395.77	15.10	17.5
500	19.2	396.90	14.33	16.8
501	21.0	391.99	9.67	22.4
502	21.0	396.90	9.08	20.6
503	21.0	396.90	5.64	23.9
504	21.0	393.45	6.48	22.0
505	21.0	396.90	7.88	11.9

[506 rows x 14 columns]

[125]:
```python
# Determine the mean of each feature
averages_column = np.mean(boston_df, axis = 0)
print(averages_column)
```

```
CRIM         3.593761
ZN          11.363636
INDUS       11.136779
CHAS         0.069170
NOX          0.554695
RM           6.284634
AGE         68.574901
DIS          3.795043
RAD          9.549407
TAX        408.237154
PTRATIO     18.455534
B          356.674032
LSTAT       12.653063
MEDV        22.532806
dtype: float64
```

[126]:
```python
# Determine the mean of each row
averages_row = np.mean(boston_df, axis = 1)
print(averages_row)
```

```
0      59.635666
1      56.235315
2      55.298456
3      52.585755
4      53.731875
5      53.256432
6      61.520342
7      64.543646
8      64.077024
9      62.390724
10     63.379471
11     62.601226
12     59.316977
13     59.951733
14     60.346704
15     59.437714
16     56.999681
17     60.880721
18     50.586658
19     60.061236
20     61.470549
21     61.904810
22     62.467812
23     62.892474
```

```
24     62.291561
25     55.078724
26     60.745351
27     55.671012
28     61.852906
29     60.935961
          …
476    90.554622
477    88.216579
478    89.752321
479    89.804136
480    88.547301
481    88.859420
482    89.237483
483    86.210763
484    84.816184
485    86.797169
486    89.342475
487    86.874712
488    92.280340
489    88.865841
490    87.484433
491    92.284703
492    91.821659
493    65.674786
494    65.189448
495    64.136614
496    67.542371
497    66.809291
498    66.533016
499    66.892266
500    67.353899
501    57.842659
502    58.516841
503    59.581947
504    59.144678
505    58.111815
Length: 506, dtype: float64
```

So we can determine the averages by row, but should we do this? Why or why not?

Answer: It's very hard to interpret a these values, because taking an average across different features does not make sense.

Let's put together what you have learned about averages and subsetting to do the next problems.

We will determine the average price for houses along the Charles River and that for houses NOT along the river.

[130]:
```python
# Use the query method to define a subset of boston_df that only include houses␣
↪are along the river (CHAS = 1).
along_river = boston_df.query('CHAS == 1')
along_river
```

[130]:
```
        CRIM    ZN  INDUS  CHAS     NOX     RM    AGE     DIS   RAD    TAX  \
142  3.32105   0.0  19.58   1.0  0.8710  5.403  100.0  1.3216   5.0  403.0
152  1.12658   0.0  19.58   1.0  0.8710  5.012   88.0  1.6102   5.0  403.0
154  1.41385   0.0  19.58   1.0  0.8710  6.129   96.0  1.7494   5.0  403.0
155  3.53501   0.0  19.58   1.0  0.8710  6.152   82.6  1.7455   5.0  403.0
160  1.27346   0.0  19.58   1.0  0.6050  6.250   92.6  1.7984   5.0  403.0
162  1.83377   0.0  19.58   1.0  0.6050  7.802   98.2  2.0407   5.0  403.0
163  1.51902   0.0  19.58   1.0  0.6050  8.375   93.9  2.1620   5.0  403.0
208  0.13587   0.0  10.59   1.0  0.4890  6.064   59.1  4.2392   4.0  277.0
209  0.43571   0.0  10.59   1.0  0.4890  5.344  100.0  3.8750   4.0  277.0
210  0.17446   0.0  10.59   1.0  0.4890  5.960   92.1  3.8771   4.0  277.0
211  0.37578   0.0  10.59   1.0  0.4890  5.404   88.6  3.6650   4.0  277.0
212  0.21719   0.0  10.59   1.0  0.4890  5.807   53.8  3.6526   4.0  277.0
216  0.04560   0.0  13.89   1.0  0.5500  5.888   56.0  3.1121   5.0  276.0
218  0.11069   0.0  13.89   1.0  0.5500  5.951   93.8  2.8893   5.0  276.0
219  0.11425   0.0  13.89   1.0  0.5500  6.373   92.4  3.3633   5.0  276.0
220  0.35809   0.0   6.20   1.0  0.5070  6.951   88.5  2.8617   8.0  307.0
221  0.40771   0.0   6.20   1.0  0.5070  6.164   91.3  3.0480   8.0  307.0
222  0.62356   0.0   6.20   1.0  0.5070  6.879   77.7  3.2721   8.0  307.0
234  0.44791   0.0   6.20   1.0  0.5070  6.726   66.5  3.6519   8.0  307.0
236  0.52058   0.0   6.20   1.0  0.5070  6.631   76.5  4.1480   8.0  307.0
269  0.09065  20.0   6.96   1.0  0.4640  5.920   61.5  3.9175   3.0  223.0
273  0.22188  20.0   6.96   1.0  0.4640  7.691   51.8  4.3665   3.0  223.0
274  0.05644  40.0   6.41   1.0  0.4470  6.758   32.9  4.0776   4.0  254.0
276  0.10469  40.0   6.41   1.0  0.4470  7.267   49.0  4.7872   4.0  254.0
277  0.06127  40.0   6.41   1.0  0.4470  6.826   27.6  4.8628   4.0  254.0
282  0.06129  20.0   3.33   1.0  0.4429  7.645   49.7  5.2119   5.0  216.0
283  0.01501  90.0   1.21   1.0  0.4010  7.923   24.8  5.8850   1.0  198.0
356  8.98296   0.0  18.10   1.0  0.7700  6.212   97.4  2.1222  24.0  666.0
357  3.84970   0.0  18.10   1.0  0.7700  6.395   91.0  2.5052  24.0  666.0
358  5.20177   0.0  18.10   1.0  0.7700  6.127   83.4  2.7227  24.0  666.0
363  4.22239   0.0  18.10   1.0  0.7700  5.803   89.0  1.9047  24.0  666.0
364  3.47428   0.0  18.10   1.0  0.7180  8.780   82.9  1.9047  24.0  666.0
369  5.66998   0.0  18.10   1.0  0.6310  6.683   96.8  1.3567  24.0  666.0
370  6.53876   0.0  18.10   1.0  0.6310  7.016   97.5  1.2024  24.0  666.0
372  8.26725   0.0  18.10   1.0  0.6680  5.875   89.6  1.1296  24.0  666.0

     PTRATIO       B  LSTAT  MEDV
142     14.7  396.90  26.82  13.4
152     14.7  343.28  12.12  15.3
154     14.7  321.02  15.12  17.0
155     14.7   88.01  15.02  15.6
```

```
160      14.7  338.92   5.50  27.0
162      14.7  389.61   1.92  50.0
163      14.7  388.45   3.32  50.0
208      18.6  381.32  14.66  24.4
209      18.6  396.90  23.09  20.0
210      18.6  393.25  17.27  21.7
211      18.6  395.24  23.98  19.3
212      18.6  390.94  16.03  22.4
216      16.4  392.80  13.51  23.3
218      16.4  396.90  17.92  21.5
219      16.4  393.74  10.50  23.0
220      17.4  391.70   9.71  26.7
221      17.4  395.24  21.46  21.7
222      17.4  390.39   9.93  27.5
234      17.4  360.20   8.05  29.0
236      17.4  388.45   9.54  25.1
269      18.6  391.34  13.65  20.7
273      18.6  390.77   6.58  35.2
274      17.6  396.90   3.53  32.4
276      17.6  389.25   6.05  33.2
277      17.6  393.45   4.16  33.1
282      14.9  377.07   3.01  46.0
283      13.6  395.52   3.16  50.0
356      20.2  377.73  17.60  17.8
357      20.2  391.34  13.27  21.7
358      20.2  395.43  11.48  22.7
363      20.2  353.04  14.64  16.8
364      20.2  354.55   5.29  21.9
369      20.2  375.33   3.73  50.0
370      20.2  392.05   2.96  50.0
372      20.2  347.88   8.88  50.0
```

What do you notice about the CHAS column?

Answer: It's all 1.0! This means that we successfully subsetting all houses that are along the Charles River. Great work!

[128]:
```
# Now determine the average price for these houses. 'MEDV' is the column name␣
↪for the prices.
averages_price_along_river = np.mean(along_river['MEDV'])
averages_price_along_river
```

[128]:
```
28.44
```

Now try determining the average for houses NOT along the River.

[129]:
```
# Determine the average price for houses that are NOT along the Charles River␣
↪(when CHAS = 0).
```

```
not_along_river = boston_df.query('CHAS == 0')
averages_price_not_along_river = np.mean(not_along_river['MEDV'])
averages_price_not_along_river
```

[129]: 22.093842887473482

Good work! You're becoming an expert in subsetting and determining averages on subsetted data. This will be integral for your capstone projects and future careers as data scientists!