PandasLibrary

July 5, 2024

[1]:
```
#For data Processing and Analysis
#2 dimensional tabular data structure labeled axes(rows and columns)
```

[2]:
```
import pandas as pd
```

creating a Pandas DataFrame example boston_df = pd.DataFrame(boston_dataset.data, columns = boston_dataset.feature_names

#Assuming I was importing it from sklearn directly example below. I would have use the method above to convert it to a padas DataFrame

from sklearn.datasets import load_boston

#Load the Boston housing dataset boston = load_boston()

#Print the keys of the dataset print(boston.keys())

[3]:
```
boston_df = pd.read_csv('boston_house_prices.csv', header = 1) #header contain␣
 ↪extra data so I used header = 1 to remove extra header
boston_df.head()
```

[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	

	B	LSTAT	MEDV
0	396.90	4.98	24.0
1	396.90	9.14	21.6
2	392.83	4.03	34.7
3	394.63	2.94	33.4
4	396.90	5.33	36.2

[4]:
```
#shape
boston_df.shape
```

[4]: (506, 14)

[5]:
```
#type
type(boston_df)
```

[5]:
```
pandas.core.frame.DataFrame
```

[6]:
```
#Exporting a DataFrame to csv
boston_df.to_csv('boston.csv')
```

[7]:
```
#Exporting a DataFrame to excel
boston_df.to_excel('boston.xlsx')
```

[8]:
```
#Creating a DataFrame with random values

import numpy as np

random_df = pd.DataFrame(np.random.rand(20,10))
random_df.head()
```

[8]:

	0	1	2	3	4	5	6	7	8	9
0	0.353794	0.569348	0.335363	0.338580	0.723563	0.544879	0.932823	0.670619	0.389457	0.187429
1	0.765536	0.134124	0.529256	0.684574	0.814044	0.028395	0.267463	0.162250	0.552952	0.554127
2	0.261995	0.395016	0.956637	0.329251	0.134040	0.174550	0.405641	0.968517	0.580209	0.322305
3	0.163821	0.127225	0.401216	0.822183	0.238618	0.846591	0.759350	0.526902	0.690157	0.872856
4	0.637001	0.359534	0.282441	0.860696	0.038564	0.208109	0.065581	0.330574	0.268005	0.855705

[9]:
```
random_df.shape
```

[9]:
```
(20, 10)
```

[10]:
```
import pandas as pd
import numpy as np

# Create a DataFrame with 20 rows and 10 columns of random integers
random_whole = pd.DataFrame(np.random.randint(20, 100, size=(5, 5)))

# Display the first 5 rows of the DataFrame
print(random_whole.head())
```

```
    0   1   2   3   4
0  22  42  61  37  76
1  35  27  53  80  73
```

```
2  60  80  68  86  26
3  78  60  38  88  24
4  77  47  38  50  65
```

[11]:
```
#Number of  rows and columns
random_whole.shape
```

[11]:
```
(5, 5)
```

[12]:
```
#information about your data
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

[13]:
```
#findig the number of missing values

boston_df.isnull().sum()
```

[13]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
```

```
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

[14]:
```
diabetes_df = pd.read_csv('diabetes.csv')
diabetes_df.head()
```

[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[15]:
```
#Counting values based on the labels

diabetes_df.value_counts('Outcome')
```

[15]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

[16]:
```
#Grouping values base on labes
diabetes_df.groupby('Outcome').mean()
```

[16]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

Statistical Measures

[17]:
```
boston_df.count()
```

[17]:
```
CRIM       506
ZN         506
INDUS      506
CHAS       506
NOX        506
RM         506
AGE        506
DIS        506
RAD        506
TAX        506
PTRATIO    506
B          506
LSTAT      506
MEDV       506
dtype: int64
```

[18]:
```
boston_df.mean
```

[18]:
```
<bound method NDFrame._add_numeric_operations.<locals>.mean of         CRIM
ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31      0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07      0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07      0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18      0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18      0  0.458  7.147  54.2  6.0622    3  222
..       …   …    …   …    …    …    …    …  …  …
501  0.06263   0.0  11.93      0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93      0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93      0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93      0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93      0  0.573  6.030  80.8  2.5050    1  273

     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
..       …     …    …   …
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]>
```

[ ]:

[19]:
```
boston_df.std()
```

[19]:
```
CRIM         8.601545
ZN          23.322453
INDUS        6.860353
CHAS         0.253994
NOX          0.115878
RM           0.702617
AGE         28.148861
DIS          2.105710
RAD          8.707259
TAX        168.537116
PTRATIO      2.164946
B           91.294864
LSTAT        7.141062
MEDV         9.197104
dtype: float64
```

[20]:
```
boston_df.mean()
```

[20]:
```
CRIM         3.613524
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
```

[21]:
```
boston_df.max()
```

[21]:
```
CRIM        88.9762
ZN         100.0000
INDUS       27.7400
CHAS         1.0000
NOX          0.8710
RM           8.7800
AGE        100.0000
```

```
DIS         12.1265
RAD         24.0000
TAX        711.0000
PTRATIO     22.0000
B          396.9000
LSTAT       37.9700
MEDV        50.0000
dtype: float64
```

[22]:
```
#All the Satistical VAlue
boston_df.describe()
```

[22]:

	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

	AGE	DIS	RAD	TAX	PTRATIO	B
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000

	LSTAT	MEDV
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

[23]:
```
#Manipulating a DataFrame
# Check how to add column and row from a dataset

boston_df.head()
```

[23]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[24]:
```
boston_df.head()
```

[24]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[25]:
```
#To drop a row you mention the index

new_boston_df = boston_df.drop(index=0, axis=0)
```

[26]:
```
boston_df.head()
```

[26]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
```

```
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[27]:
```
boston_df.shape
```

[27]:
```
(506, 14)
```

[28]:
```
#To drop column

boston_df2 = boston_df.drop(columns=['ZN'])
```

To drop row but keep it in the main dataset

#Load the Boston housing dataset boston_df = pd.read_csv('boston_house_prices.csv')

#Drop the first row (index 0) new_boston_df = boston_df.drop(index=0, axis=0)

#Display the first 5 rows of the new DataFrame print(new_boston_df.head())

[34]:
```
boston_df.head()
```

[34]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[33]:
```
The first row was temporary removed because I gave the DataFrame a differen name
new_boston_df.head()
```

[33]:
```
      CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
1  0.02731  0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729  0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237  0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905  0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7
5  0.02985  0.0   2.18     0  0.458  6.430  58.7  6.0622    3  222     18.7

        B  LSTAT  MEDV
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
```

```
4  396.90   5.33  36.2
5  394.12   5.21  28.7
```

[45]:
```
boston_df.head()
```

[45]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[49]:
```
print(boston_df.iloc[:,0]) #first column
print(boston_df.iloc[:,1]) #second column
print(boston_df.iloc[:,2]) #third  column
print(boston_df.iloc[:,3]) #fouth column
print(boston_df.iloc[:,4]) #fifth column
print(boston_df.iloc[:,-1]) #last column
```

```
0      0.00632
1      0.02731
2      0.02729
3      0.03237
4      0.06905
        …
501    0.06263
502    0.04527
503    0.06076
504    0.10959
505    0.04741
Name: CRIM, Length: 506, dtype: float64
0      18.0
1       0.0
2       0.0
3       0.0
4       0.0
       …
501     0.0
502     0.0
503     0.0
504     0.0
```

```
505     0.0
Name: ZN, Length: 506, dtype: float64
0       2.31
1       7.07
2       7.07
3       2.18
4       2.18
       …
501    11.93
502    11.93
503    11.93
504    11.93
505    11.93
Name: INDUS, Length: 506, dtype: float64
0      0
1      0
2      0
3      0
4      0
      ..
501    0
502    0
503    0
504    0
505    0
Name: CHAS, Length: 506, dtype: int64
0      0.538
1      0.469
2      0.469
3      0.458
4      0.458
       …
501    0.573
502    0.573
503    0.573
504    0.573
505    0.573
Name: NOX, Length: 506, dtype: float64
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       …
501    22.4
502    20.6
503    23.9
504    22.0
```

```
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

CORRELATION Positive and negative correlation

[ ]:
```
And is the relationship between various columns.
Decrease in one value and increase in old value is negative correlation.
Increase in one value and increase in the other value is the positive value
```

[51]:
```
boston_df.corr()
```

[51]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339
MEDV	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955

	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.388305
ZN	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725
CHAS	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260
NOX	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321
RM	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.695360
AGE	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.376955
DIS	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.249929
RAD	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.381626
TAX	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.468536
PTRATIO	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.507787
B	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.333461
LSTAT	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.737663
MEDV	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.000000

[52]:
```
# Renaming a column in-place
boston_df.rename(columns={'MEDV': 'PRICE'}, inplace=True)
```

[53]:
```
boston_df.head()
```

[53]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	

	B	LSTAT	PRICE
0	396.90	4.98	24.0
1	396.90	9.14	21.6
2	392.83	4.03	34.7
3	394.63	2.94	33.4
4	396.90	5.33	36.2

[ ]: