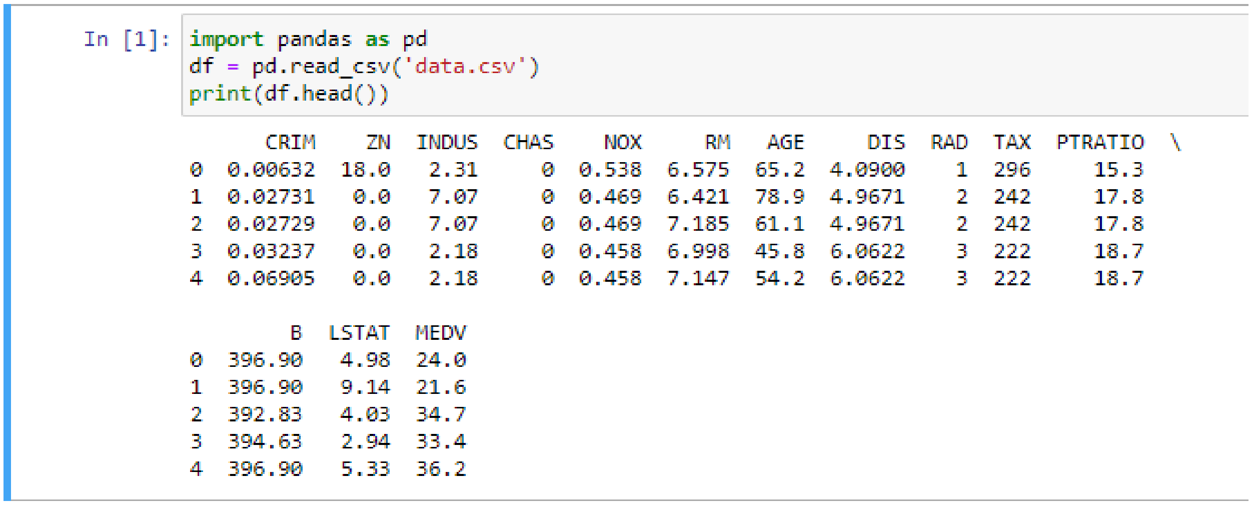
Rowena T. Sabellita

BSIT 4R7

LabAssignment04 – ML Pre-Processing of Data

I. Data Normalization and /Standardization Activity

1. Load the above dataset.

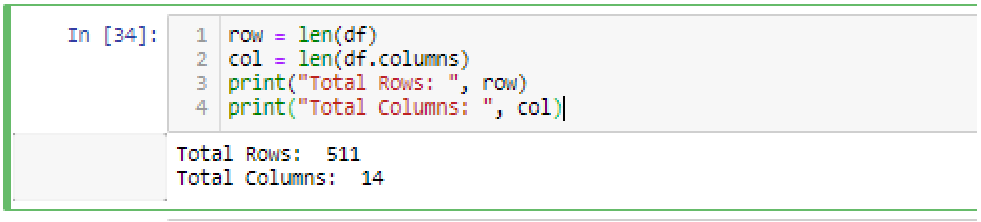


```
In [1]: import pandas as pd
        df = pd.read_csv('data.csv')
        print(df.head())
```

```
      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
```

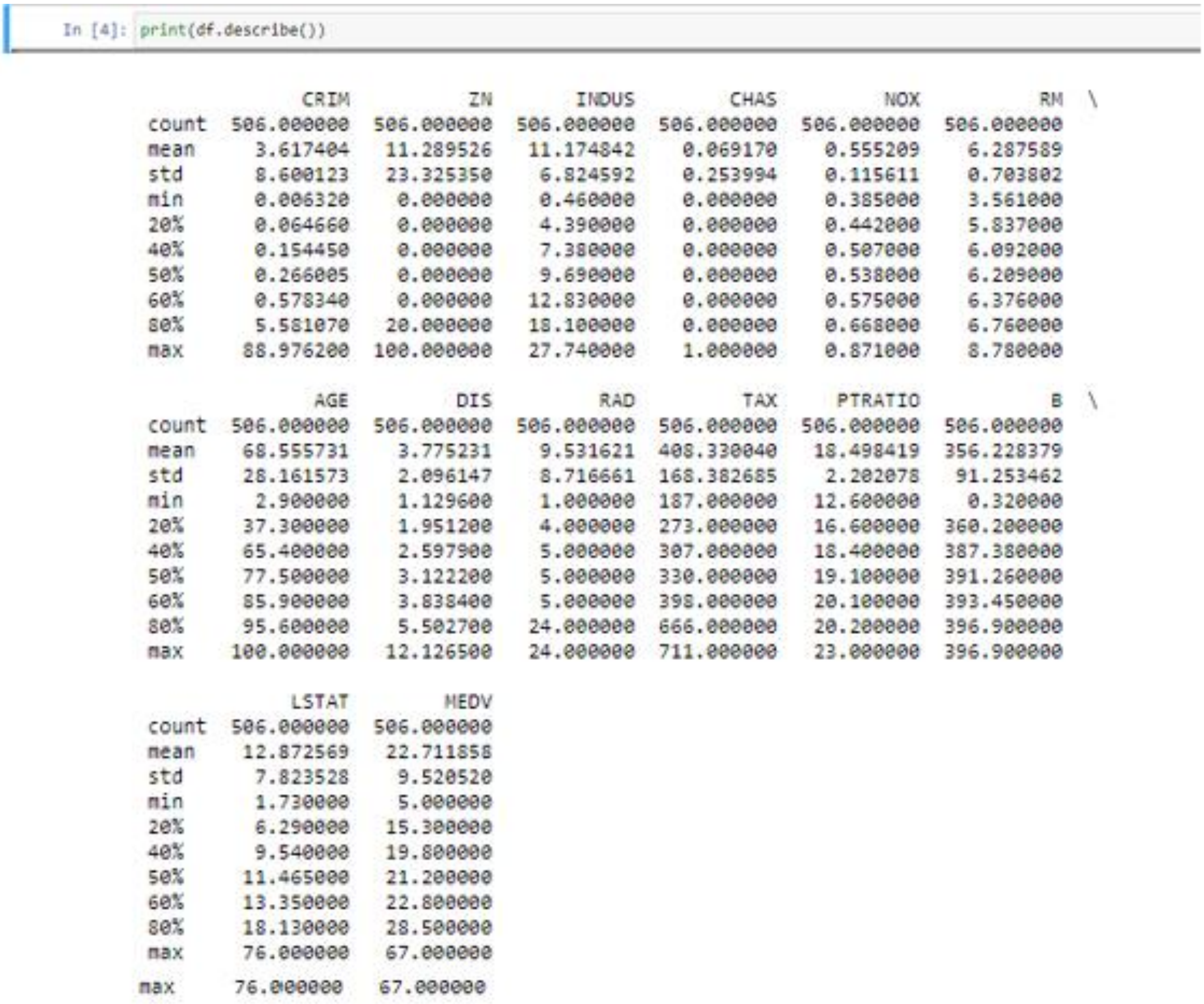
2. Get the number of columns and rows.



```
In [34]: row = len(df)
         col = len(df.columns)
         print("Total Rows: ", row)
         print("Total Columns: ", col)
```

```
Total Rows:  511
Total Columns:  14
```

3. Use describe function for the set.



```
In [4]: print(df.describe())
```

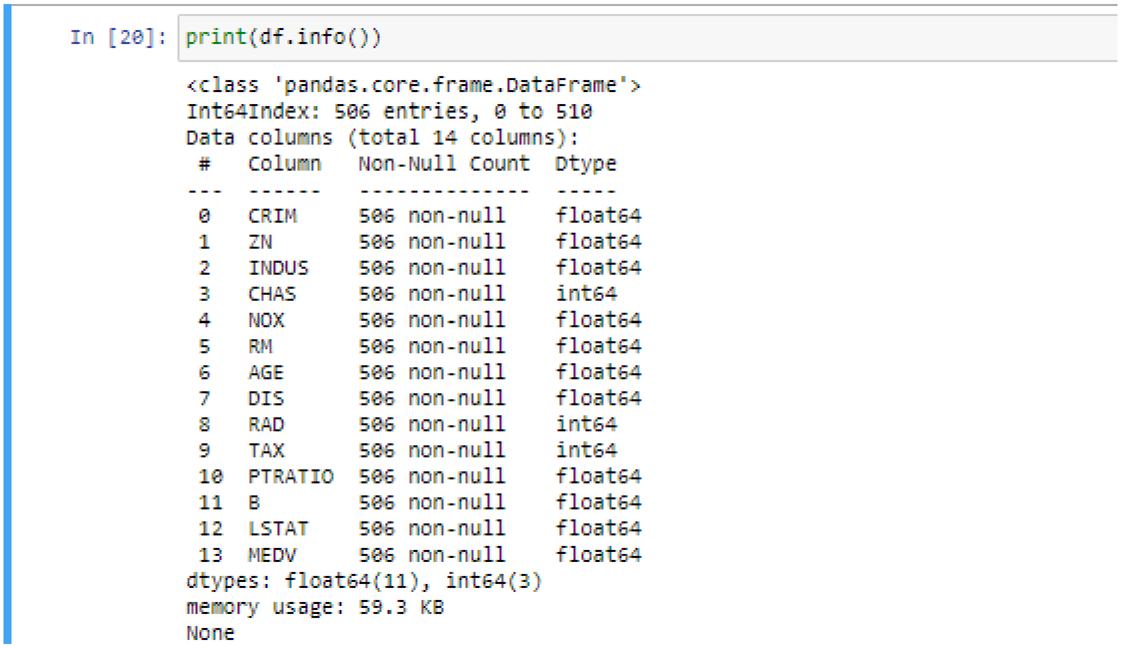
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.617404   11.289526   11.174842    0.069170    0.555209    6.287589
std      8.600123   23.325350    6.824592    0.253994    0.115611    0.703802
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
20%      0.064660    0.000000    4.390000    0.000000    0.442000    5.837000
40%      0.154450    0.000000    7.380000    0.000000    0.507000    6.092000
50%      0.266005    0.000000    9.690000    0.000000    0.538000    6.209000
60%      0.578340    0.000000   12.830000    0.000000    0.575000    6.376000
80%      5.581070   20.000000   18.100000    0.000000    0.668000    6.760000
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    68.555731    3.775231    9.531621  408.330040   18.498419  356.228379
std     28.161573    2.096147    8.716661  168.382685    2.202078   91.253462
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
20%     37.300000    1.951200    4.000000  273.000000   16.600000  360.200000
40%     65.400000    2.597900    5.000000  307.000000   18.400000  387.380000
50%     77.500000    3.122200    5.000000  330.000000   19.100000  391.260000
60%     85.900000    3.838400    5.000000  398.000000   20.100000  393.450000
80%     95.600000    5.502700   24.000000  666.000000   20.200000  396.900000
max    100.000000   12.126500   24.000000  711.000000   23.000000  396.900000

            LSTAT        MEDV
count  506.000000  506.000000
mean    12.872569   22.711858
std      7.823528    9.520520
min      1.730000    5.000000
20%      6.290000   15.300000
40%      9.540000   19.800000
50%     11.465000   21.200000
60%     13.350000   22.800000
80%     18.130000   28.500000
max     76.000000   67.000000
```

```
max     76.000000   67.000000
```

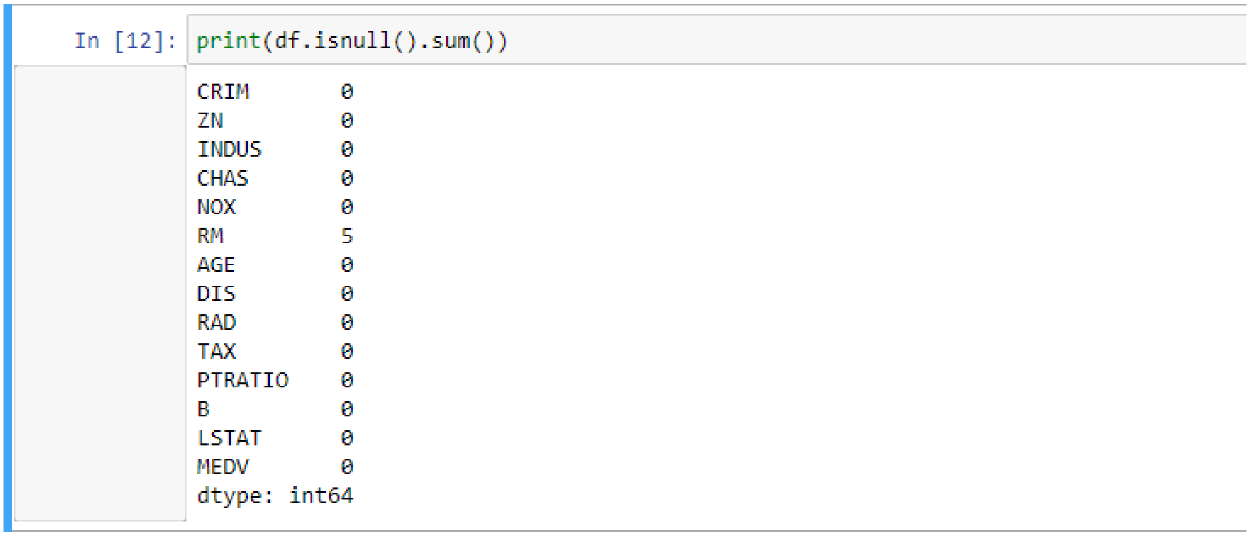
4. Use info function for the data.



```
In [20]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 506 entries, 0 to 510
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: 59.3 KB
None
```

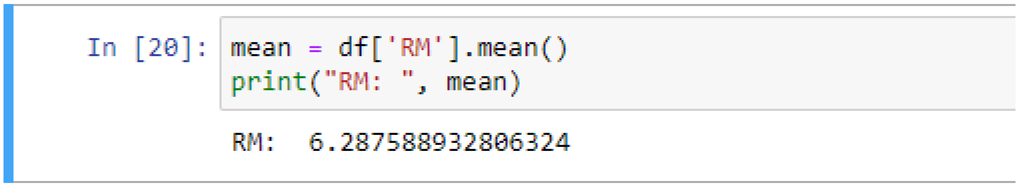
5. Count the number of missing values.



```
In [12]: print(df.isnull().sum())
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         5
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

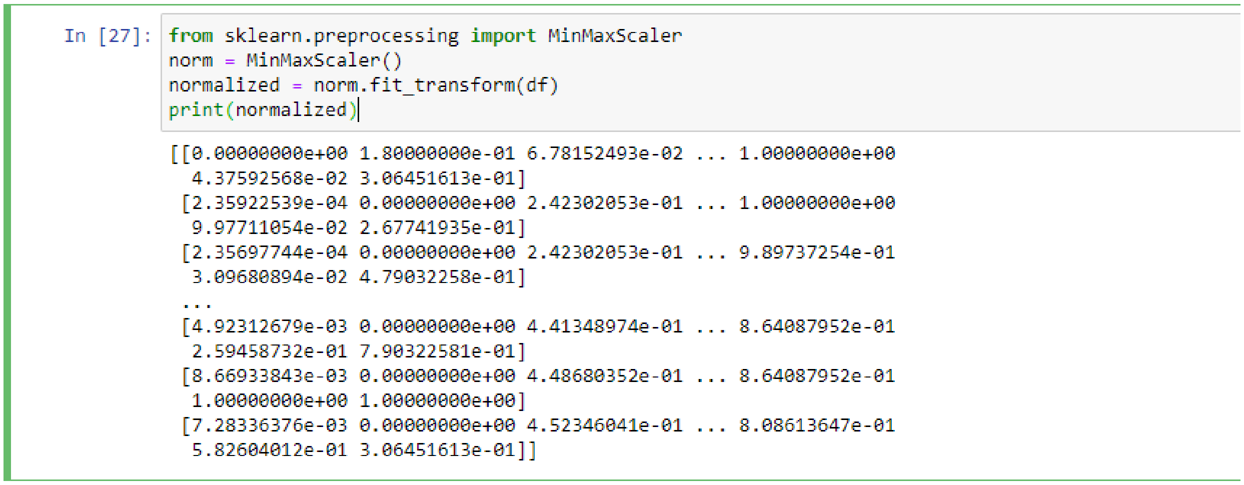
6. Replace the missing values with the average of non-null values.



```
In [20]: mean = df['RM'].mean()
         print("RM: ", mean)

         RM:  6.287588932806324
```

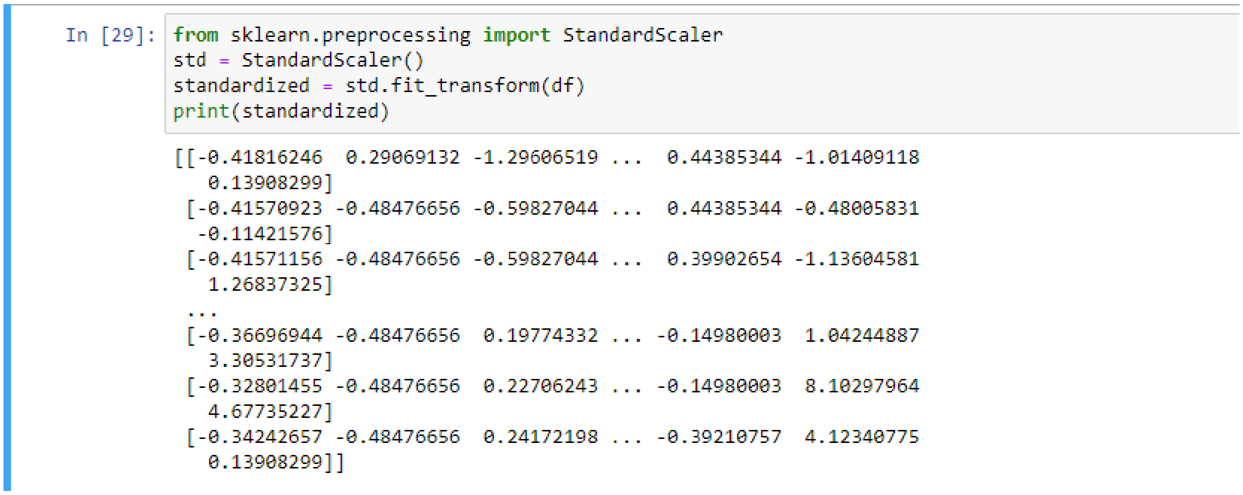
7. Perform data normalization for all the features.



```
In [27]: from sklearn.preprocessing import MinMaxScaler
         norm = MinMaxScaler()
         normalized = norm.fit_transform(df)
         print(normalized)

         [[0.00000000e+00 1.80000000e-01 6.78152493e-02 ... 1.00000000e+00
           4.37592568e-02 3.06451613e-01]
          [2.35922539e-04 0.00000000e+00 2.42302053e-01 ... 1.00000000e+00
           9.97711054e-02 2.67741935e-01]
          [2.35697744e-04 0.00000000e+00 2.42302053e-01 ... 9.89737254e-01
           3.09680894e-02 4.79032258e-01]
          ...
          [4.92312679e-03 0.00000000e+00 4.41348974e-01 ... 8.64087952e-01
           2.59458732e-01 7.90322581e-01]
          [8.66933843e-03 0.00000000e+00 4.48680352e-01 ... 8.64087952e-01
           1.00000000e+00 1.00000000e+00]
          [7.28336376e-03 0.00000000e+00 4.52346041e-01 ... 8.08613647e-01
           5.82604012e-01 3.06451613e-01]]
```

8. Perform data standardization to all the features.



```
In [29]: from sklearn.preprocessing import StandardScaler
         std = StandardScaler()
         standardized = std.fit_transform(df)
         print(standardized)

         [[-0.41816246  0.29069132 -1.29606519 ...  0.44385344 -1.01409118
            0.13908299]
          [-0.41570923 -0.48476656 -0.59827044 ...  0.44385344 -0.48005831
           -0.11421576]
          [-0.41571156 -0.48476656 -0.59827044 ...  0.39902654 -1.13604581
            1.26837325]
          ...
          [-0.36696944 -0.48476656  0.19774332 ... -0.14980003  1.04244887
            3.30531737]
          [-0.32801455 -0.48476656  0.22706243 ... -0.14980003  8.10297964
            4.67735227]
          [-0.34242657 -0.48476656  0.24172198 ... -0.39210757  4.12340775
            0.13908299]]
```