13-simple-linear-regression

May 17, 2023

1 Simple Linear Regression Model

[3]:
```
#Step 1: Import the necessary libraries

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model, metrics
```

[7]:
```
# Step 2: # load the boston dataset

#boston = datasets.load_boston(return_X_y=False)

boston = pd.read_csv('C:/Users/lenovo/Desktop/DS Lab Week Wise/
 ↪Week-5-Regression_model/Boston.csv')
```

[8]:
```
# Displaying the dataset

boston.head()
```

[8]:
```
   Unnamed: 0     crim    zn  indus  chas    nox     rm   age     dis  rad  \
0           1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1
1           2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2
2           3  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2
3           4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3
4           5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3

   tax  ptratio   black  lstat  medv
0  296     15.3  396.90   4.98  24.0
1  242     17.8  396.90   9.14  21.6
2  242     17.8  392.83   4.03  34.7
3  222     18.7  394.63   2.94  33.4
4  222     18.7  396.90   5.33  36.2
```

[11]:
```
# Step 3: Having the glance at independent and dependent variable

boston = boston.loc[:,['lstat','medv']]
```

[13]:

```
# Displaing the new dataset

boston.head(10)
```

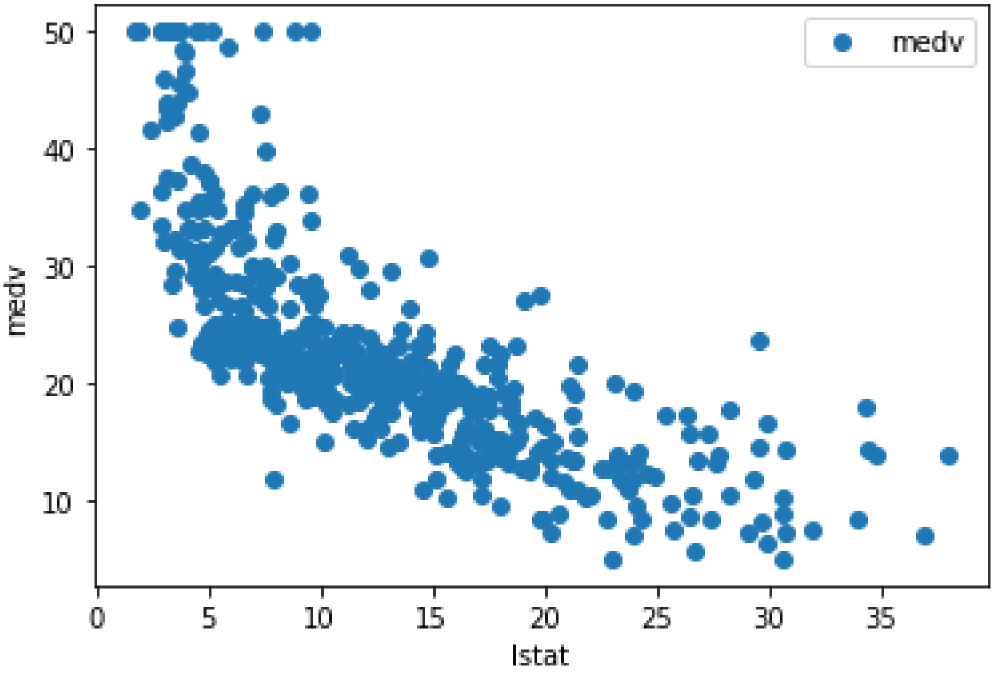
[13]:

```
   lstat  medv
0   4.98  24.0
1   9.14  21.6
2   4.03  34.7
3   2.94  33.4
4   5.33  36.2
5   5.21  28.7
6  12.43  22.9
7  19.15  27.1
8  29.93  16.5
9  17.10  18.9
```

[14]:

```
# Step 4: Visualizing the change in the variables

boston.plot(x = 'lstat', y = 'medv', style = 'o')
plt.xlabel('lstat')
plt.ylabel('medv')
plt.show()
```



[16]:

```
# Step 5: Dividing the dataset into dependent and independent variables

x = boston['lstat']
y = boston['medv']

x
```

[16]:

```
0      4.98
1      9.14
2      4.03
3      2.94
4      5.33
       …
501    9.67
502    9.08
503    5.64
504    6.48
505    7.88
Name: lstat, Length: 506, dtype: float64
```

[17]:

```
# Step 6: Spliting the data into train and test sets

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.
 ↪4,random_state=1)
```

[18]:

```
# Checking the shapes of the train and test sets

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(303,)
(203,)
(303,)
(203,)
```

[30]:

```
x_train = x_train[:, None]

x_test = x_test[:, None]
```

```
C:\Users\lenovo\AppData\Local\Temp/ipykernel_6060/296195278.py:3: FutureWarning:
Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and
will be removed in a future version.  Convert to a numpy array before indexing
instead.
  x_test = x_test[:, None]
```

[23]:
```
# Step 7: Training the algorithm

from sklearn.linear_model import LinearRegression

lregressor = LinearRegression()
lregressor.fit(x_train.reshape(-1, 1),y_train)
```

[23]:
```
LinearRegression()
```

[31]:
```
# Step 8: Predicting the y values from the built model

y_pred = lregressor.predict(x_test)
```

[24]:
```
# Step 9: Retrieving the intercept

print(lregressor.intercept_)
```

```
34.13076276013123
```

[25]:
```
# Step 10: Retrieving the slope

print(lregressor.coef_)
```

```
[-0.91573274]
```

[32]:
```
# Step 11: Printing the predicted values

y_pred
```

[32]:
```
array([27.23529525, 27.55580171, 16.91498731, 26.71332759, 24.79944617,
       23.94781473, 29.8268189 , 22.18960787, 17.72083211, 26.06315735,
       27.04299138, 29.90923484, 21.66764021, 24.75365953, 23.39837508,
       23.02292466, 12.84913395, 29.89092019, 27.32686852,  7.08917504,
       23.59983629, 18.88381269, 25.63276296, 28.52647841, 29.85429088,
       11.79604131, 15.50475889, 24.48809704, 27.48254309, 15.0377352 ,
       29.10339003, 17.22633644, 31.49345248, 19.07611657, 25.80675218,
       21.68595487, 17.83987737, 29.24990727, 12.75756068, 20.41308636,
       27.40928447, 27.94956679, 27.17119396, 12.0616038 , 17.62010151,
       13.30700032, 32.37255591, 19.15853251, 25.14742461, 24.39652377,
       23.45331905, 23.85624145, 29.3597952 , 23.96612938,  6.95181513,
       27.96788144,  6.68625263, 28.65468099, 20.69696351, 30.51361845,
       20.37645705, 28.15102799, 15.88936664, 17.97723728,  7.07086038,
       29.51546977, 31.8689029 , 26.20967459, 24.6071423 , 23.4716337 ,
       28.49900643,  8.23384096, 21.31050444, 23.16028457, 21.2280885 ,
       24.58882764, 31.42019386, 26.59428234, 27.36349783, 30.87075422,
       20.03763594, 24.69871557, 29.19496331, 12.82166197, 28.65468099,
       29.57957106, 16.44796361, 29.47884046, 21.21893117, 18.59993554,
       28.96603012, 29.44221115, 17.5834722 , 22.74820484, 20.83432342,
       21.97898934, 25.27562719, 26.65838363, 30.33962923, 22.22623718,
       19.76291612, 16.86920067, 28.81035556, 27.84883618,  5.91703714,
       22.83062079, 18.41678899, 29.09423271, 22.08887727, 27.10709267,
       31.23704731, 25.85253882, 14.42419426, 28.20597195, 26.99720474,
       31.50260981, 25.2115259 , 23.42584707, 32.31761194, 27.3177112 ,
       25.00090737, 16.99740325, 29.69861631, 21.13651522, 25.42214443,
       22.2994958 , 26.27377588, 25.81590951, 20.67864885, 29.28653658,
       21.17314453, 24.64377161, 28.44406246, 10.61474608, 27.5191724 ,
       27.78473489,  9.57081076, 16.72268343, 27.20782327, 18.37100236,
       29.03013142, 17.22633644, 17.04318989, 28.30670255, 31.08137275,
       25.74265089, 27.53748705, 14.60734081, 28.04114006, 17.71167479,
       25.34888581, 15.92599595, 26.56681035, 20.90758204, 14.68059943,
       30.17479734, 27.61074567, 27.0521487 , 20.70612084, 25.39467245,
       30.86159689, 29.28653658, 28.12355601, 19.75375879, 21.21893117,
       22.29033847, 28.30670255, 12.43705422, 18.38931701, 27.97703877,
       24.7078729 , 17.59262953, 18.30690107, 28.91108616, 20.61454756,
       27.22613792, 28.01366808, 23.11449793, 32.54654513, 26.081472  ,
       14.47913823, 26.62175432, 14.69891408, 26.2463039 , 11.64036674,
       20.56876093, 24.83607548, 28.10524135, 25.68770693, 20.97168333,
       19.74460146, 17.52852824, 22.22623718, 21.82331478, 26.91478879,
       20.66949153, 29.9275495 , 25.62360563, 30.05575208, 25.99905606,
       18.93875665,  2.65702859, 25.54118969])
```

[27]:

```
# Step 12: Printing the Actual values

y_test
```

[27]:

```
307    28.2
343    23.9
47     16.6
67     22.0
362    20.8
       …
186    50.0
372    50.0
442    18.4
412    17.9
213    28.1
Name: medv, Length: 203, dtype: float64
```

[33]:

```
# Step 13: Evaluating the algorithm

from sklearn import metrics
print('Mean Absolute Error: ', metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error: ', metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error: ', np.sqrt(metrics.
 ↪mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error:  4.85248058295052
Mean Squared Error:  45.23188062977258
Root Mean Squared Error:  6.7254650865031325
```

[ ]:

```
```