Linear Regression Boston Dataset

March 29, 2021

Linear Regression ML Boston Housing Price Dataset

Sohini Mukherjee

29.03.2021

Loading libraries

[75]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn import metrics
%matplotlib inline
```

Loading Boston Housing Price Dataset

[14]:
```
boston = load_boston()
```

Checking the data

[15]:
```
print(boston['DESCR'])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**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.
https://archive.ics.uci.edu/ml/machine-learning-databases/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.

.. topic:: 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.
```

[16]:

```
print(boston['feature_names'])
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

[19]:
```
X = boston.data
print(X.shape)
```

```
(506, 13)
```

[21]:
```
y = boston.target
print(y.shape)
```

```
(506,)
```

Converting into Dataframe

[23]:
```
boston = pd.DataFrame(boston.data, columns= boston.feature_names)
```

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

[24]:
```
      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

   PTRATIO       B  LSTAT
0     15.3  396.90   4.98
1     17.8  396.90   9.14
2     17.8  392.83   4.03
3     18.7  394.63   2.94
4     18.7  396.90   5.33
```

MEDV is missing from the dataset. Creating new column.

[27]:
```
boston['MEDV'] = y  #y contains the target values of boston dataset
```

Checking Missing Values

[32]:
```
boston.isnull().sum()
```

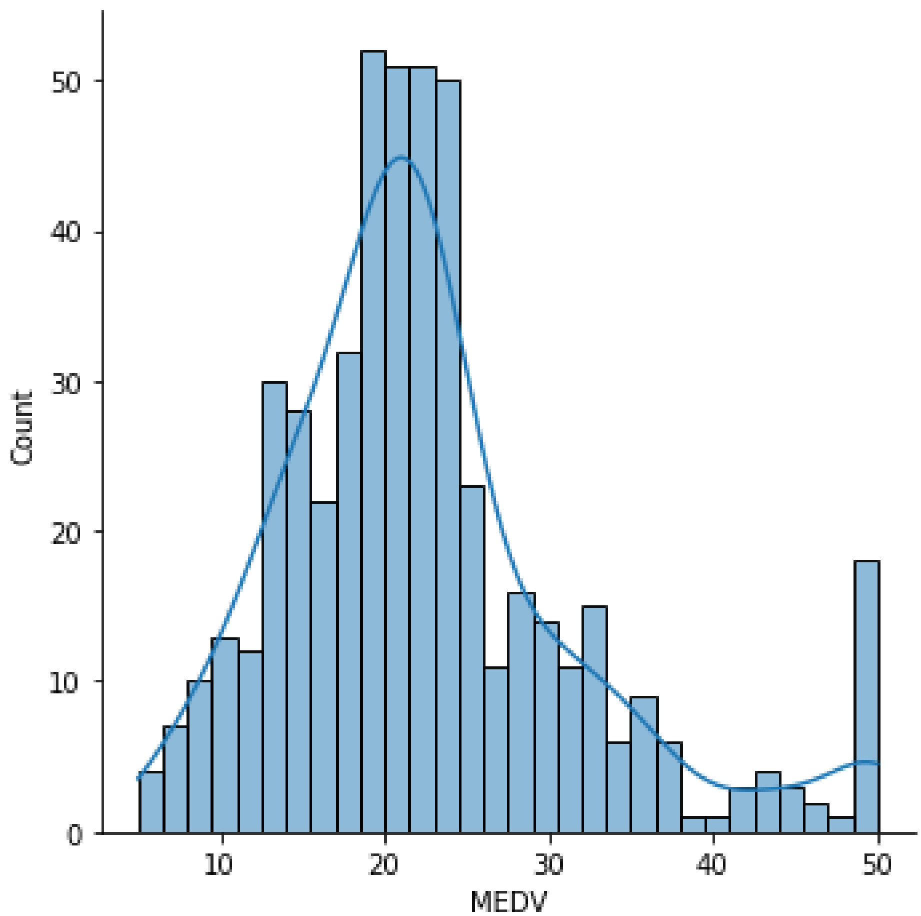
[32]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
```

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

Plotting a Distribution Plot

[31]:
```
plt.rcParams["patch.force_edgecolor"] = True
sns.displot(boston['MEDV'], kde= True, bins=30)
```

[31]: <seaborn.axisgrid.FacetGrid at 0x1d6fc0ef8b0>



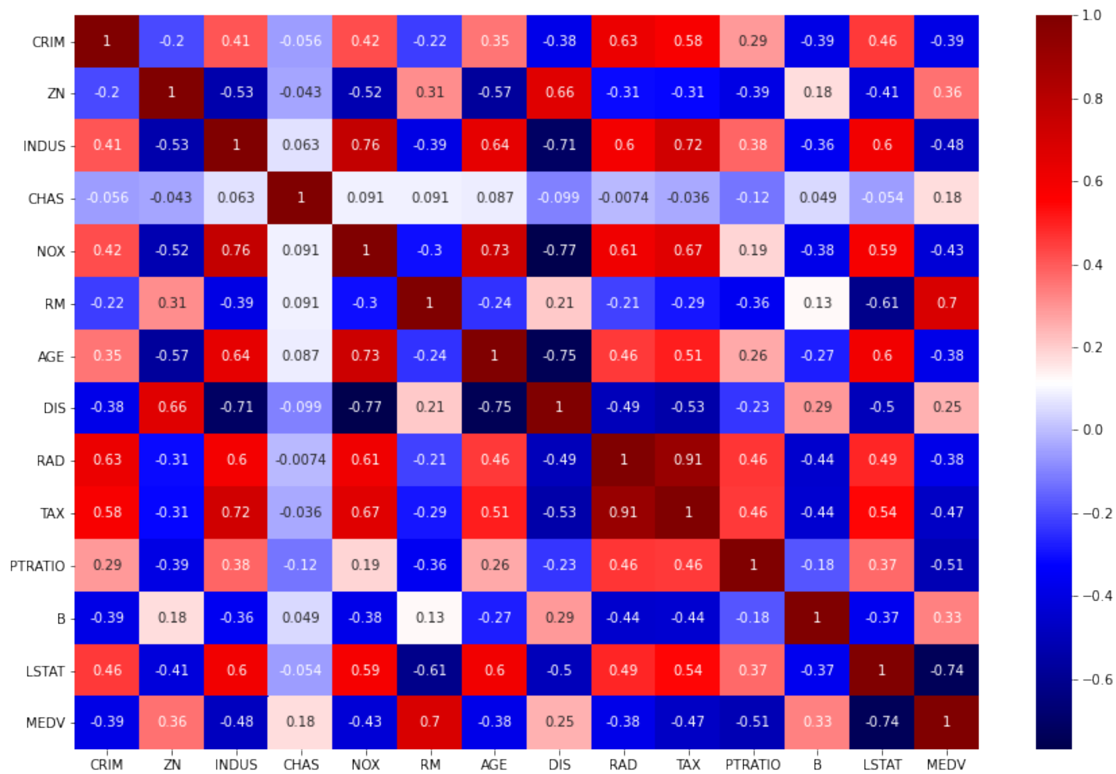
The MEDV data is normally distributed with a few outliers.

[35]:
```
boston.corr()
```

[35]:
```
            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
```

[42]:
```python
plt.figure(figsize=(15,10))
sns.heatmap(boston.corr(), annot=True, cmap= 'seismic')
plt.show()
```



To fit a Regression Model we select the feature that has high Correlation Value. From the correlation matrix we can see that RM has a strong positive correlation with MEDV (0.7), where as LSTAT has a high negative correlation with MEDV(-0.74).

While selecting features for a linear regression model we check for multi-co-linearity. The features RAD, TAX have a correlation of 0.91. These feature pairs are strongly correlated to each other. We should not select both these features together for training the model. Same goes for the features DIS and AGE which have a correlation of -0.75.

Based on the above observations we will RM and LSTAT as our features. Using a scatter plot let's see how these features vary with MEDV.

[44]:
```
#features to train
X = boston[['RM', 'LSTAT']]
```

[46]:
```
#feature to predict
y = boston['MEDV']
```

[65]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=100)
```

Using Linear Regression and training set

[66]:
```
lm= LinearRegression()
```

[67]:
```
lm.fit(X_train, y_train)
```

[67]:
```
LinearRegression()
```

Checking Intercepts and Coefficients

[68]:
```
lm.intercept_
```

[68]:
```
-0.6726621207092123
```

[69]:
```
lm.coef_
```

[69]:
```
array([ 4.87252709, -0.58585259])
```

Making a Dataframe with columns and Coefficients

[70]:
```
df = pd.DataFrame(lm.coef_, X.columns, columns=['Coefficient'])
```

[71]:
```
df.head()
```

[71]:

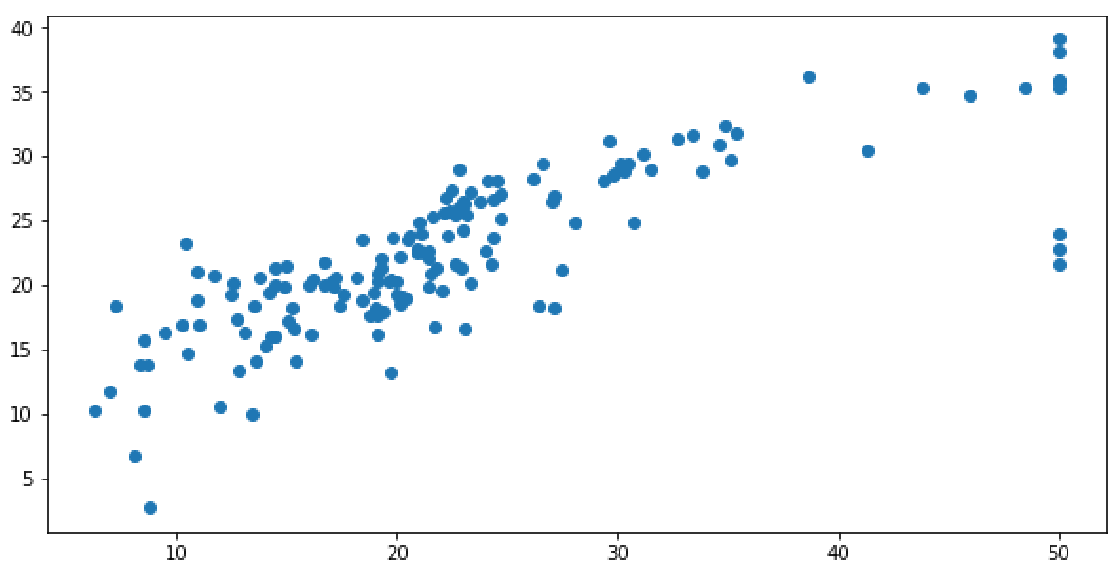
	Coefficient
RM	4.872527
LSTAT	-0.585853

Predictions

[122]:
```
predictions = lm.predict(X_test)
```

[123]:
```
plt.figure(figsize=(10,5))
plt.scatter(y_test, predictions)
```

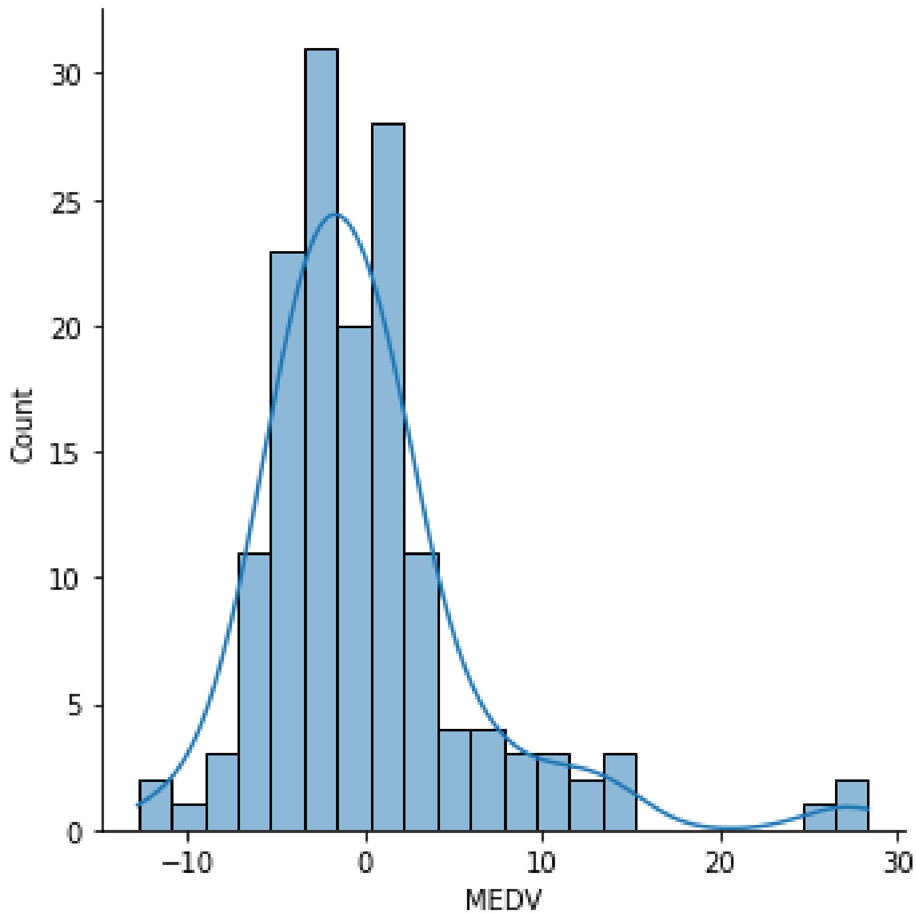
[123]:
```
<matplotlib.collections.PathCollection at 0x1d68abf4220>
```



Scatter plot resembles a Straight Line with a few Outliers

[74]:
```
#histogram of residuals
sns.displot((y_test-predictions),kde=True)
```

[74]: <seaborn.axisgrid.FacetGrid at 0x1d6846f15e0>



The residuals data is normally distributed with a few outliers. so the model choice is correct

[76]:
```
metrics.mean_absolute_error(y_test, predictions)
```

[76]: 4.220033478324288

[77]:
```
#root mean squared error
np.sqrt(metrics.mean_absolute_error(y_test, predictions))
```

[77]: 2.054272006897891

```
[ ]:
```