

|  |
| --- |
| Boston Housing Dataset |
|  |
| April 27  People-click Techno Solutions  Authored by: Biswajit Mohapatra |



# Boston Housing Price

|  |
| --- |
| **Predicting House Prices**[**¶**](https://www.kaggle.com/pallavie/housing-prices-prediction#Predicting-House-Prices) |
| This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from Kaggle(<https://www.kaggle.com/vikrishnan/boston-house-prices> ) The dataset is small in size with only 506 cases and 14 features. |
| **The Notebook contains –**  1.DATA Pre-Processing  2. Data Analysis  3. Model Building and Prediction - ML regression  4.Train, Test and Conclusion |

IMPORTING library

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib import \*

from sklearn import datasets

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error,r2\_score

LOADING THE DATA

#Data loading using scikit-learn library

boston\_dataset = datasets.load\_boston()

boston = pd.DataFrame(boston\_dataset.data, columns=boston\_dataset.feature\_names)

boston.head()

O/P:-

|  | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396.90 | 4.98 |
| 1 | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 |
| 2 | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 |
| 3 | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 |
| 4 | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396.90 | 5.33 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

\*We Effectively load the data\*

PRE-PROCESSING WORK

#Data preprocessing

boston['MEDV'] = boston\_dataset.target

boston.isnull().sum()

O/P:-

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

\*We can see there is no null value present in our dataset\*

DATA VISUALIZATION

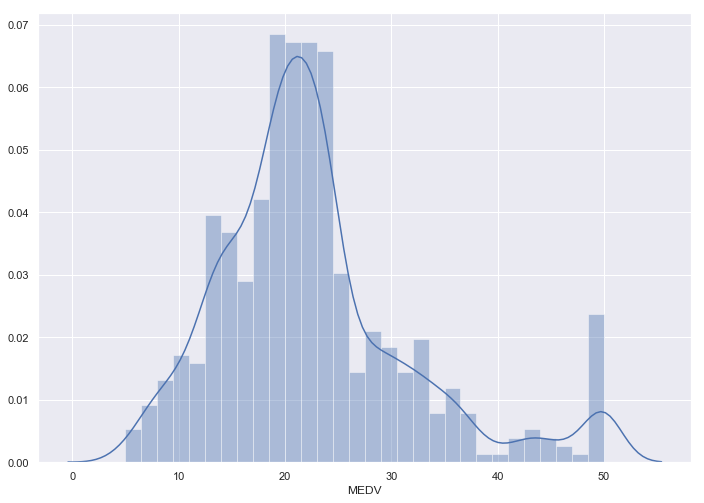
#Exploratory Data Analysis

sns.set(rc={'figure.figsize':(11.7,8.27)})

sns.distplot(boston['MEDV'], bins=30)

plt.show()

O/p:-



# annot = True to print the values inside the square

cor = boston.corr()

sns.heatmap(cor , annot=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x137ad9084e0>



Observations:

* To fit a linear regression model, we select those features which have a high correlation with our target variable MEDV. By looking at 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).
* An important point in selecting features for a linear regression model is to 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. Check [this](https://stats.stackexchange.com/a/1150) for an explanation. 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.

plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']

target = boston['MEDV']

for i, col in enumerate(features):

plt.subplot(1, len(features) , i+1)

x = boston[col]

y = target

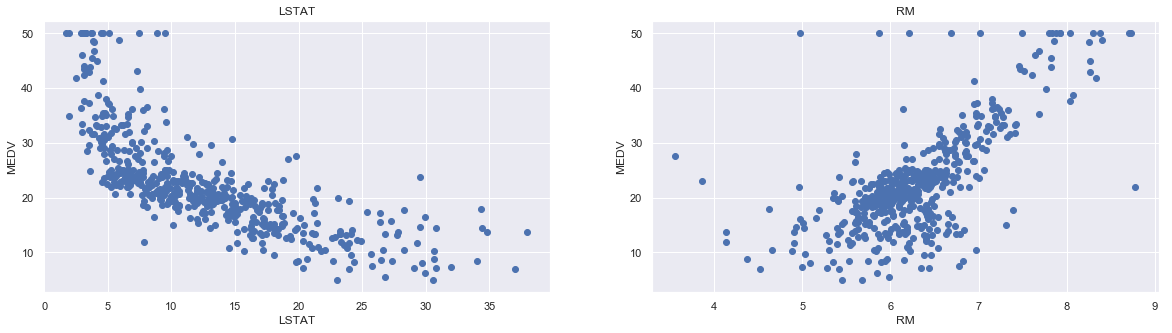
plt.scatter(x, y, marker='o')

plt.title(col)

plt.xlabel(col)

plt.ylabel('MEDV')

O/p:-



Time to Train the Model

#Preparing the data for training the model

X = pd.DataFrame(np.c\_[boston['LSTAT'], boston['RM']], columns = ['LSTAT','RM'])

Y = boston['MEDV']

#Splitting the data into training and testing sets

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state=5)

print(X\_train.shape)

print(X\_test.shape)

print(Y\_train.shape)

print(Y\_test.shape)

O/p:-

(404, 2) (102, 2) (404,) (102,)

#Training and testing the model

lin\_model = LinearRegression()

lin\_model.fit(X\_train, Y\_train)

o/p:-

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

#Model evaluation

#We will evaluate our model using RMSE and R2-score

y\_train\_predict = lin\_model.predict(X\_train)

rmse = (np.sqrt(mean\_squared\_error(Y\_train, y\_train\_predict)))

r2 = r2\_score(Y\_train, y\_train\_predict)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

o/p:-

The model performance for training set -------------------------------------- RMSE is 5.6371293350711955 R2 score is 0.6300745149331701

Testing is important

# model evaluation for testing set

y\_test\_predict = lin\_model.predict(X\_test)

rmse = (np.sqrt(mean\_squared\_error(Y\_test, y\_test\_predict)))

r2 = r2\_score(Y\_test, y\_test\_predict)

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

o/p:-

The model performance for testing set -------------------------------------- RMSE is 5.137400784702912 R2 score is 0.6628996975186952

Conclusion

#In this story, we applied the concepts of linear regression on the Boston housing dataset.

Predicting the housing price.

Model tested successfully. Ready for deployment.