**Data Analytics - 1 (Linear regression)**

**Problem Statement**

**Create a Linear Regression Model using Python/R to predict home prices using Boston Housing .Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.The objective is to predict the value of prices of the house using the given features.**

In [21]:

*# imports*

**import** numpy **as** np

**import** pandas **as** pd

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**from** sklearn.datasets **import** load\_boston

**import** warnings

warnings**.**filterwarnings('ignore')

**from** sklearn.preprocessing **import** StandardScaler

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

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error, mean\_absolute\_error

In [7]:

temp **=** load\_boston()

In [8]:

temp**.**keys()

Out[8]:

dict\_keys(['data', 'target', 'feature\_names', 'DESCR', 'filename', 'data\_module'])

In [10]:

data **=** pd**.**DataFrame(temp**.**data, columns**=**temp**.**feature\_names)

target **=** pd**.**DataFrame(temp**.**target, columns**=**['MEDV'])

In [13]:

data["target"] **=** target

data**.**sample(5)

Out[13]:

|  | **CRIM** | **ZN** | **INDUS** | **CHAS** | **NOX** | **RM** | **AGE** | **DIS** | **RAD** | **TAX** | **PTRATIO** | **B** | **LSTAT** | **target** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | 1.35472 | 0.0 | 8.14 | 0.0 | 0.5380 | 6.072 | 100.0 | 4.1750 | 4.0 | 307.0 | 21.0 | 376.73 | 13.04 | 14.5 |
| **351** | 0.07950 | 60.0 | 1.69 | 0.0 | 0.4110 | 6.579 | 35.9 | 10.7103 | 4.0 | 411.0 | 18.3 | 370.78 | 5.49 | 24.1 |
| **357** | 3.84970 | 0.0 | 18.10 | 1.0 | 0.7700 | 6.395 | 91.0 | 2.5052 | 24.0 | 666.0 | 20.2 | 391.34 | 13.27 | 21.7 |
| **204** | 0.02009 | 95.0 | 2.68 | 0.0 | 0.4161 | 8.034 | 31.9 | 5.1180 | 4.0 | 224.0 | 14.7 | 390.55 | 2.88 | 50.0 |
| **123** | 0.15038 | 0.0 | 25.65 | 0.0 | 0.5810 | 5.856 | 97.0 | 1.9444 | 2.0 | 188.0 | 19.1 | 370.31 | 25.41 | 17.3 |

In [17]:

data**.**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 float64

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 float64

9 TAX 506 non-null float64

10 PTRATIO 506 non-null float64

11 B 506 non-null float64

12 LSTAT 506 non-null float64

13 target 506 non-null float64

dtypes: float64(14)

memory usage: 55.5 KB

In [18]:

data**.**isna()**.**sum()

Out[18]:

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

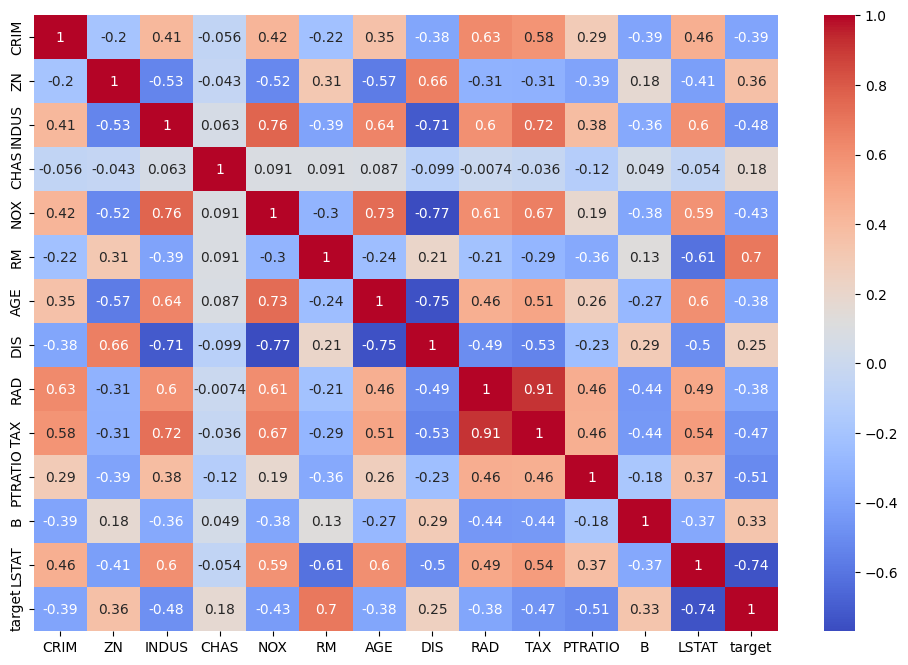
target 0

dtype: int64

In [16]:

plt**.**figure(figsize**=**(12,8))

sns**.**heatmap(data**.**corr(), annot**=** **True**, cmap**=** 'coolwarm');



Features that affect target (High value for -ve or +ve corelation): 'LSTAT', 'RM', 'TAX', 'INDUS'

In [19]:

train\_df**=** data[['LSTAT', 'RM', 'TAX','INDUS','target']]

In [20]:

x **=** train\_df[['LSTAT', 'RM', 'TAX', 'INDUS']]

y **=** train\_df['target']

In [22]:

scaler **=** StandardScaler()

x **=** scaler**.**fit\_transform(x)

In [23]:

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.2, shuffle**=True**)

**Training**

In [25]:

model **=** LinearRegression()

model**.**fit(x\_train, y\_train)

Out[25]:

LinearRegression()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

**Evaluation**

In [26]:

y\_pred **=** model**.**predict(x\_test)

In [29]:

y\_pred[:5]

Out[29]:

array([17.93709135, 26.83168283, 28.0011609 , 26.92079525, 38.14498029])

In [35]:

y\_test[:5]

Out[35]:

465 19.9

314 23.8

272 24.4

82 24.8

364 21.9

Name: target, dtype: float64

In [32]:

mse **=** mean\_squared\_error(y\_test, y\_pred)

mse

Out[32]:

27.519027800477765

In [51]:

sns**.**regplot(x **=** y\_test, y **=** y\_pred, ci**=** 95);

