In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
from sklearn.datasets import load_boston
boston_datasets=load_boston()
```

In [4]:

```
print(boston_datasets.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [5]:

```
boston_datasets.DESCR
```

Out[5]:

```
".. _boston_dataset:\n\nBoston house prices dataset\n---------------------
------\n\n**Data Set Characteristics:**  \n\n    :Number of Instances: 506 
\n\n    :Number of Attributes: 13 numeric/categorical predictive. Median V
alue (attribute 14) is usually the target.\n\n    :Attribute Information 
(in order):\n        - CRIM     per capita crime rate by town\n        - Z
N       proportion of residential land zoned for lots over 25,000 sq.ft.\n
- INDUS    proportion of non-retail business acres per town\n        - CHA
S      Charles River dummy variable (= 1 if tract bounds river; 0 otherwis
e)\n        - NOX      nitric oxides concentration (parts per 10 million)
\n        - RM       average number of rooms per dwelling\n        - AGE 
proportion of owner-occupied units built prior to 1940\n        - DIS 
weighted distances to five Boston employment centres\n        - RAD      i
ndex of accessibility to radial highways\n        - TAX      full-value pr
operty-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by tow
n\n        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blac
ks by town\n        - LSTAT    % lower status of the population\n        -
MEDV     Median value of owner-occupied homes in $1000's\n\n    :Missing A
ttribute Values: None\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n
\nThis is a copy of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/m
l/machine-learning-databases/housing/\n\n\nThis dataset was taken from the
StatLib library which is maintained at Carnegie Mellon University.\n\nThe
Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\npric
es and the demand for clean air', J. Environ. Economics & Management,\nvo
l.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostic
s\n...', Wiley, 1980.   N.B. Various transformations are used in the table
on\npages 244-261 of the latter.\n\nThe Boston house-price data has been u
sed in many machine learning papers that address regression\nproblems.   
\n     \n.. topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression 
diagnostics: Identifying Influential Data and Sources of Collinearity', Wi
ley, 1980. 244-261.\n   - 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.\n"
```

In [8]:

```
boston=pd.DataFrame(boston_datasets.data,columns=boston_datasets.feature_names)
boston
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	..
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 13 columns

In [11]:

```
boston['MEDV']=boston_datasets.target
boston
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	..
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 14 columns

In [12]:

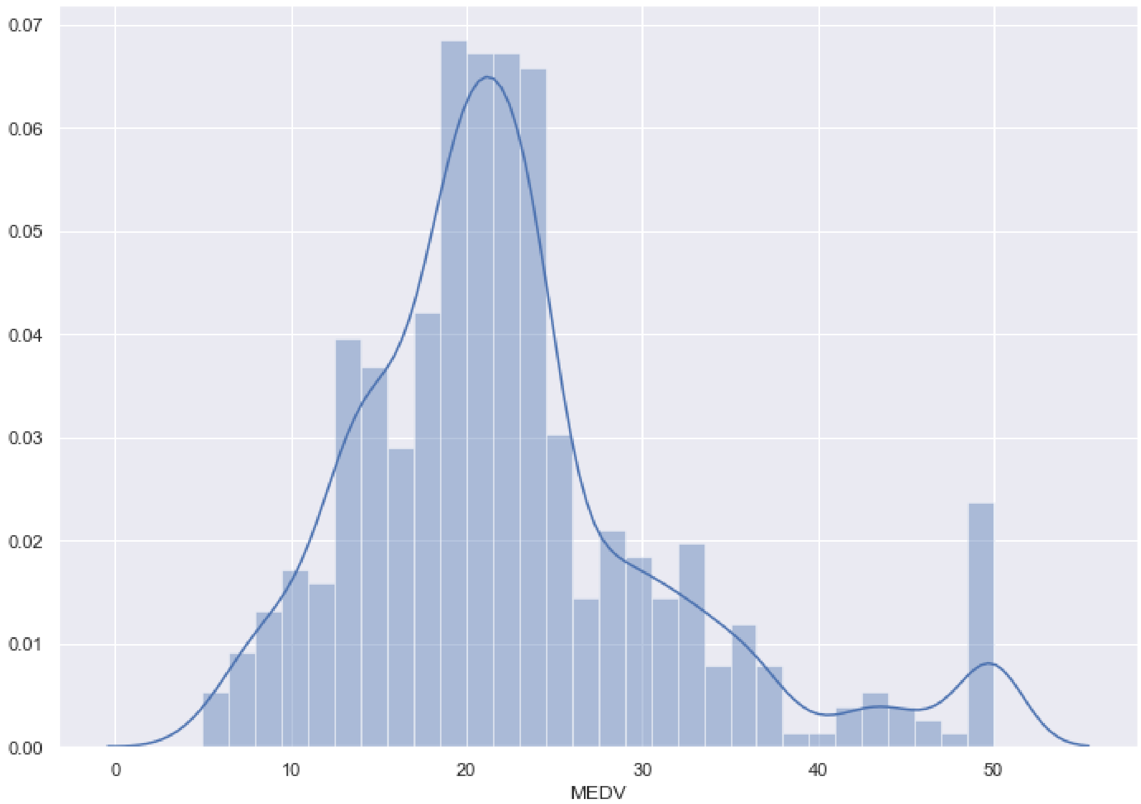
```
boston.isnull().sum()
```

Out[12]:

```
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
```

In [14]:

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston['MEDV'],bins=30)
plt.show()
```

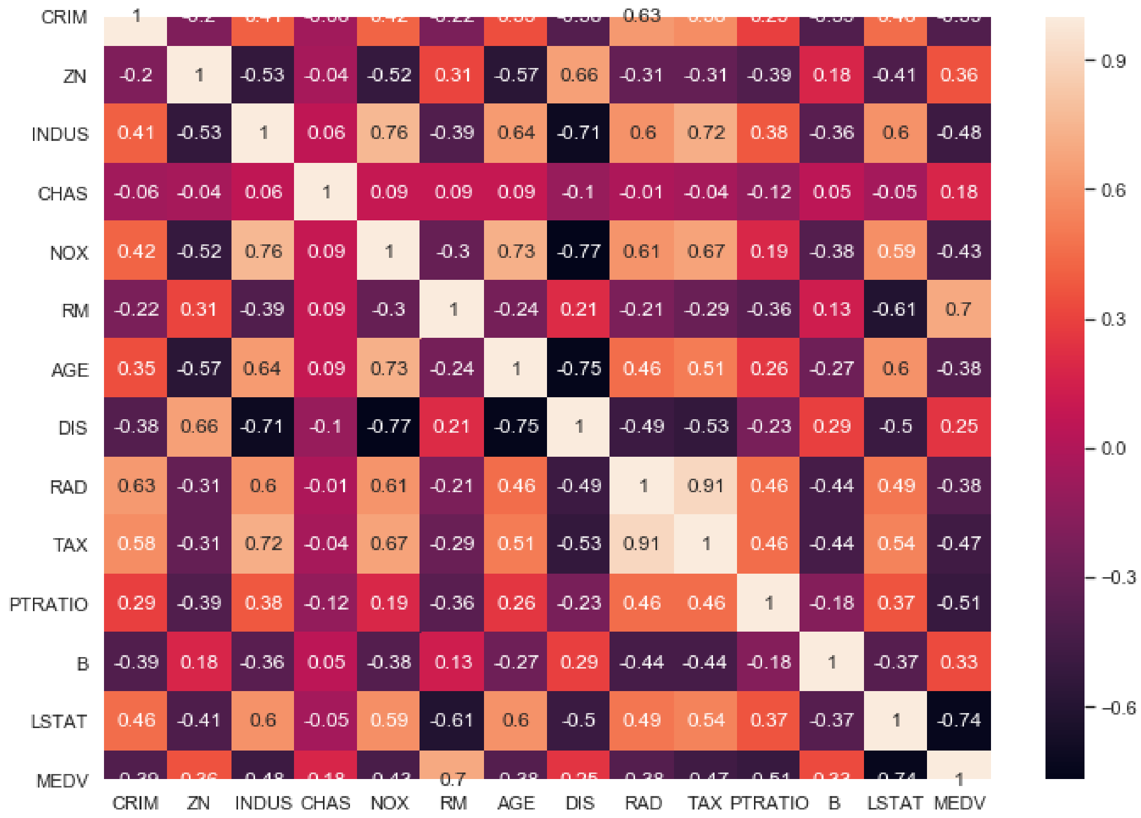


In [15]:

```
correlation_matrix=boston.corr().round(2)
sns.heatmap(data=correlation_matrix,annot=True)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20b3046a408>
```



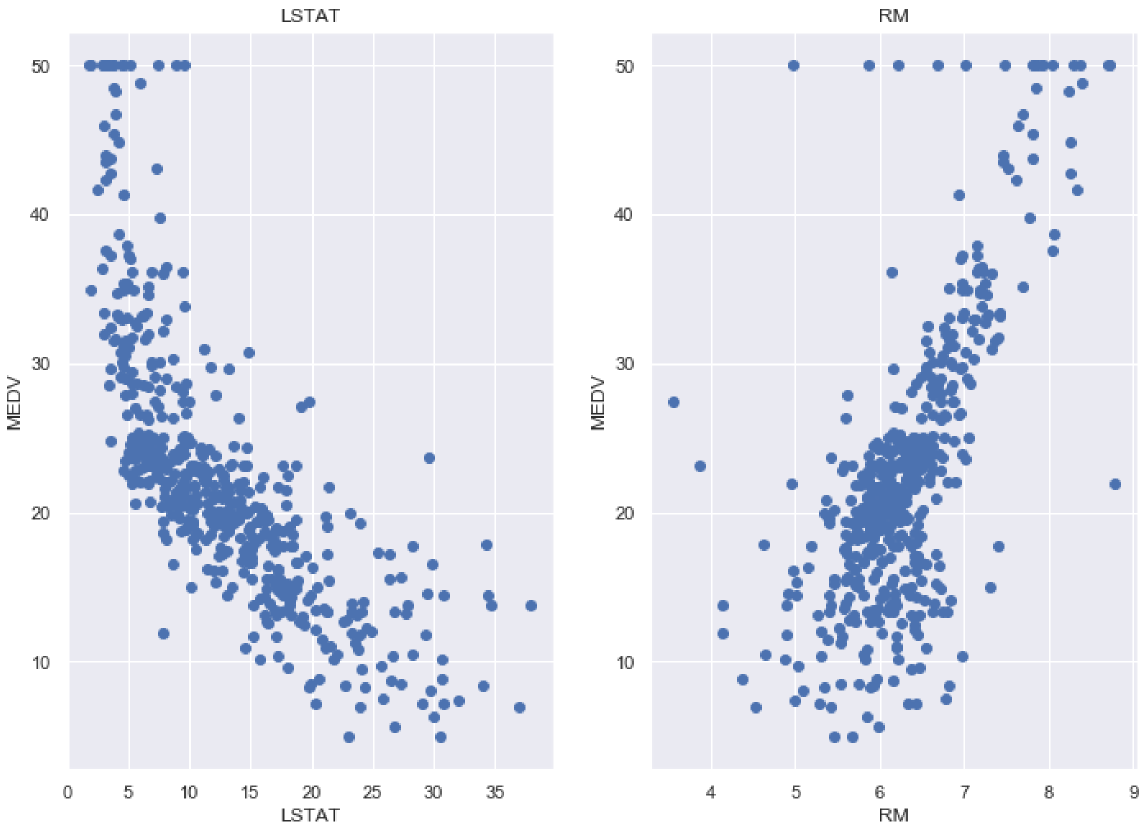
In [16]:

```
plt.figure(figsize=(20,5))
features=['LSTAT','RM']
target=boston['MEDV']
```

```
<Figure size 1440x360 with 0 Axes>
```

In [17]:

```
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')
```



In [21]:

```
X=pd.DataFrame(np.c_[boston['LSTAT'],boston['RM']],columns=['LSTAT','RM'])
Y=boston['MEDV']
```

In [22]:

```
from sklearn.model_selection import train_test_split
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)
```

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

In [28]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
lin_model=LinearRegression()
lin_model.fit(X_train,Y_train)
```

Out[28]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [37]:

```
from sklearn.metrics import 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('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")
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('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")
```

```
the model performance for training set
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701


the model performance for testing set
RMSE is 5.137400784702911
R2 score is 0.6628996975186953
```

In [ ]:

```

```