```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
```

```
csv_url="/content/housing.csv"
data=pd.read_csv(csv_url)
data
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0
...	...	...	...	...
484	6.593	9.67	21.0	470400.0
485	6.120	9.08	21.0	432600.0
486	6.976	5.64	21.0	501900.0
487	6.794	6.48	21.0	462000.0
488	6.030	7.88	21.0	249900.0

489 rows × 4 columns

```
dataset=pd.DataFrame(data)
dataset
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0
...	...	...	...	...
484	6.593	9.67	21.0	470400.0
485	6.120	9.08	21.0	432600.0
486	6.976	5.64	21.0	501900.0
487	6.794	6.48	21.0	462000.0
488	6.030	7.88	21.0	249900.0

489 rows × 4 columns

```
dataset.head(n=5)
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

Data Preprocessing

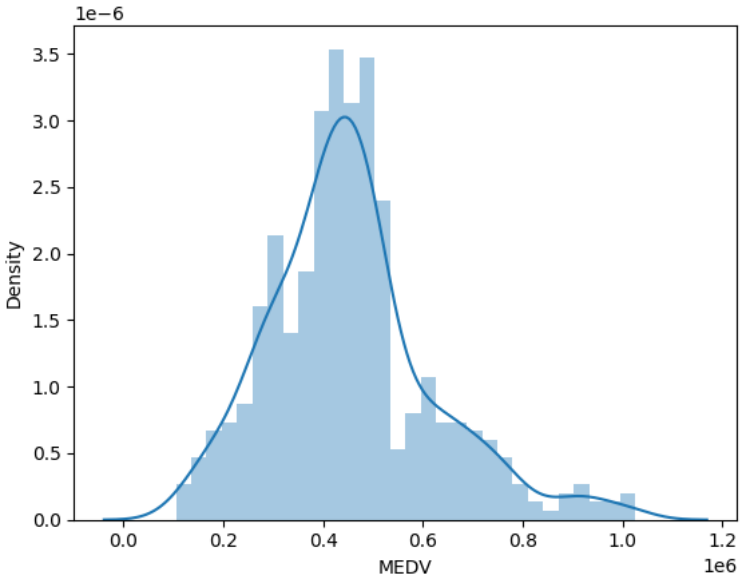
```
data.isnull().sum()
```

```
RM         0
LSTAT      0
PTRATIO    0
MEDV       0
dtype: int64
```

Exploratory Data Analysis

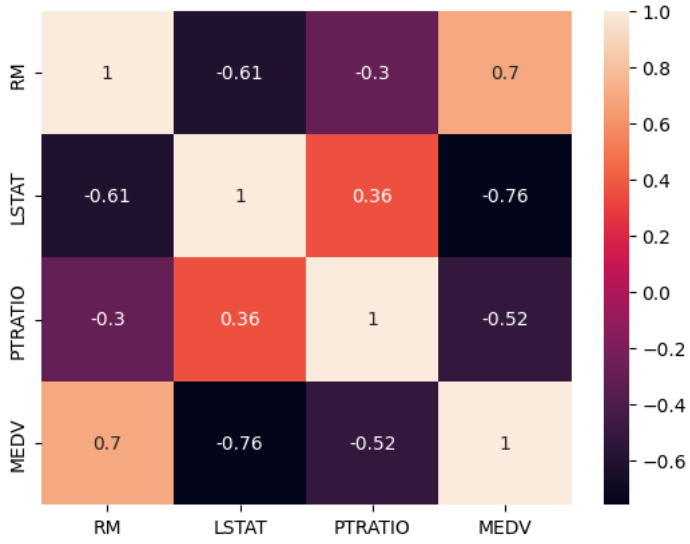
```
sns.set9rc={'figure.figsize':(11.7,8.27)}
sns.distplot(data['MEDV'],bins=30)
plt.show()
```

```
<ipython-input-13-72b58c8d7cac>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

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




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

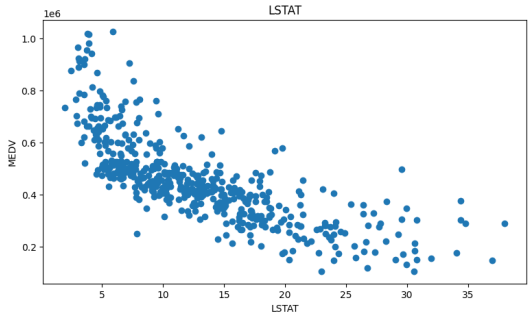
```
<Axes: >
```

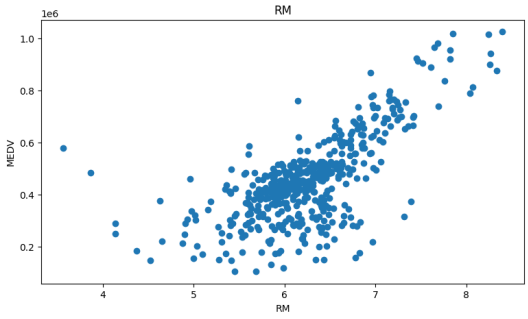



```
plt.figure(figsize=(20,5))
```

```
features=['LSTAT','RM']
```

```
target=data['MEDV']

for i, col in enumerate(features):
  plt.subplot(1,len(features),i+1)
  x=data[col]
  y=target
  plt.scatter(x,y,marker='o')
  plt.title(col)
  plt.xlabel(col)
  plt.ylabel('MEDV')
```





Splitting the data into training and testing

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


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)
```

```
(391, 2)
(98, 2)
(391,)
(98,)
```

Training and testing the model

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

```
LinearRegression
LinearRegression()
```

Model evaluation

```
from sklearn.metrics import r2_score
# model evaluation for training set
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")

# 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))
```

```
The model performance for training set
--------------------------------------
RMSE is 95873.37016704663
R2 score is 0.65444093465046


The model performance for testing set
--------------------------------------
RMSE is 95722.95711092254
R2 score is 0.6944867306092937
```