In [1]:
```python
###### DATA DESCRIPTION
```

In [2]:
```python
import numpy as np
import pandas as pd
```

In [3]:
```python
#reading the first five values of dataset
df = pd.read_csv(r'file:///C:\Users\Kapil%20Chaudhary\Desktop\analysis\Boston_Train.csv')
df.head()
```

Out[3]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [4]:
```python
###### EDA
```

In [5]:
```python
#finding the shape of the dataset
np.shape(df)
```

Out[5]:
```
(351, 15)
```

In [6]:
```python
df.isnull().sum()
```

Out[6]:
```
Unnamed: 0    0
crim          0
zn            0
indus         0
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
medv          0
dtype: int64
```

In [7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 351 entries, 0 to 350
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  351 non-null    int64  
 1   crim        351 non-null    float64
 2   zn          351 non-null    float64
 3   indus       351 non-null    float64
 4   chas        351 non-null    int64  
 5   nox         351 non-null    float64
 6   rm          351 non-null    float64
 7   age         351 non-null    float64
 8   dis         351 non-null    float64
 9   rad         351 non-null    int64  
 10  tax         351 non-null    int64  
 11  ptratio     351 non-null    float64
 12  black       351 non-null    float64
 13  lstat       351 non-null    float64
 14  medv        351 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 41.3 KB
```

In [8]:
```python
df.describe()
```

Out[8]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
count	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000	351.000000
mean	175.000000	0.401659	15.327635	8.435670	0.076923	0.510737	6.403900	60.817949	4.420862	4.472934	310.344729	17.707692	380.489174	10.407692	25.062678
std	101.469207	0.641716	25.605040	6.088947	0.266850	0.102256	0.676424	28.393094	1.968666	1.615543	67.577707	2.198252	40.455904	5.888639	8.461918
min	0.000000	0.006320	0.000000	0.460000	0.000000	0.385000	4.903000	2.900000	1.321600	1.000000	188.000000	12.600000	70.800000	1.730000	11.800000
25%	87.500000	0.057845	0.000000	4.025000	0.000000	0.437450	5.949500	36.150000	2.768500	4.000000	264.000000	16.100000	383.670000	6.095000	19.500000
50%	175.000000	0.132620	0.000000	6.200000	0.000000	0.493000	6.266000	62.000000	4.095200	4.000000	304.000000	17.900000	392.690000	9.380000	22.900000
75%	262.500000	0.404865	22.000000	10.010000	0.000000	0.544000	6.733000	88.450000	5.871800	5.000000	358.000000	19.100000	396.220000	13.210000	28.700000
max	350.000000	4.097400	100.000000	25.650000	1.000000	0.871000	8.725000	100.000000	9.222900	8.000000	469.000000	21.200000	396.900000	34.410000	50.000000

In [9]:
```python
corr = df.corr()
corr.shape
```

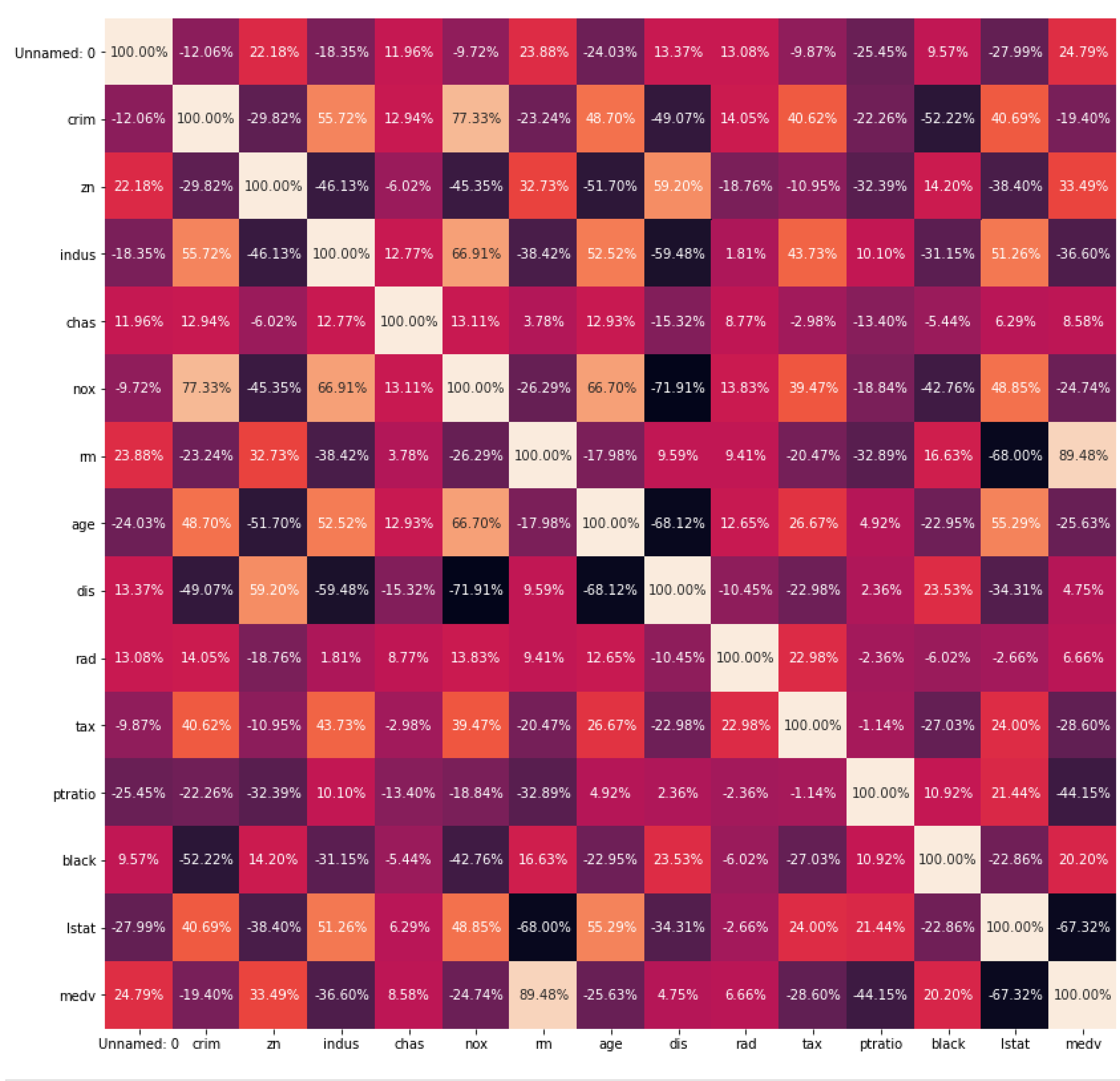
Out[9]:
```
(15, 15)
```

In [10]:
```python
####### DATA VISUALIZATION
```

In [11]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

In [12]:
```python
plt.figure(figsize=(14,14))
sns.heatmap(corr, cbar=False, square= True, fmt='.2%', annot=True,)
```

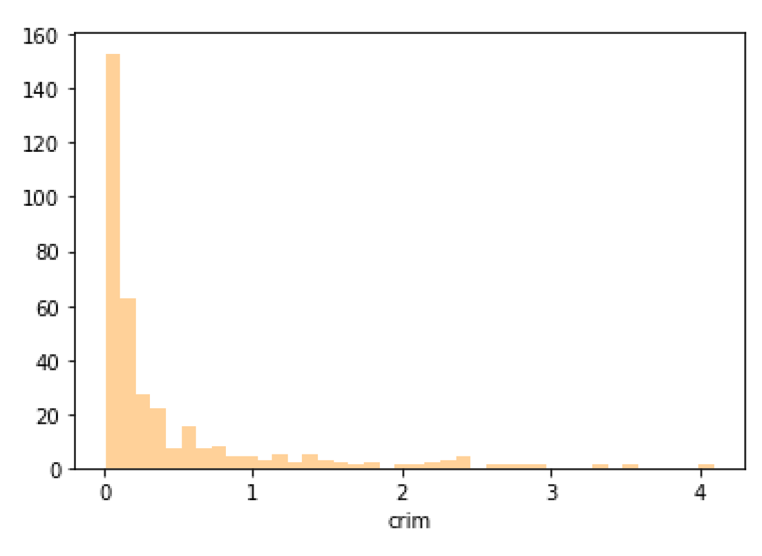
Out[12]:
```
<AxesSubplot:>
```



In [13]:
```python
sns.distplot(df['crim'].dropna(),kde=False,color='darkorange',bins=40)
```

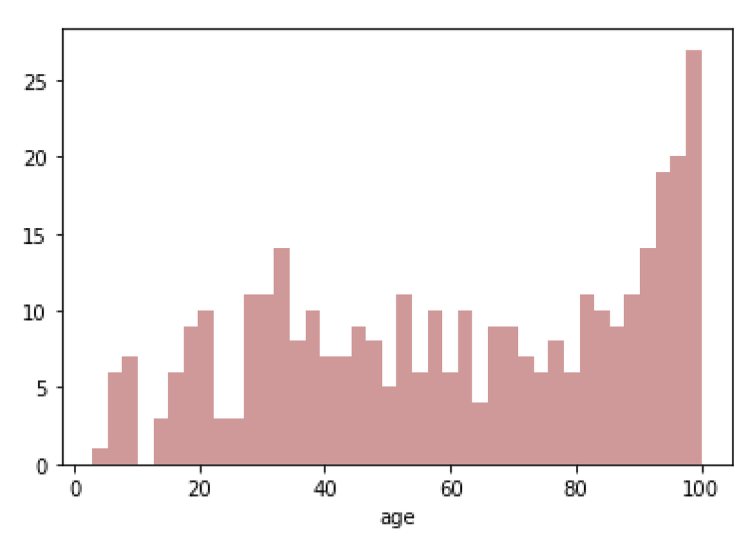
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[13]:
```
<AxesSubplot:xlabel='crim'>
```



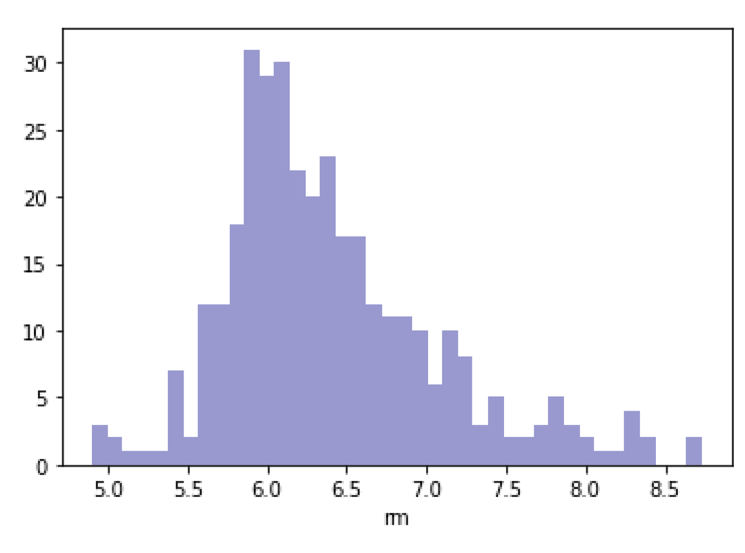
In [14]:
```python
sns.distplot(df['age'].dropna(),kde=False,color='darkred',bins=40)
```

Out[14]:
```
<AxesSubplot:xlabel='age'>
```



In [15]:
```python
sns.distplot(df['rm'].dropna(),kde=False,color='darkblue',bins=40)
```

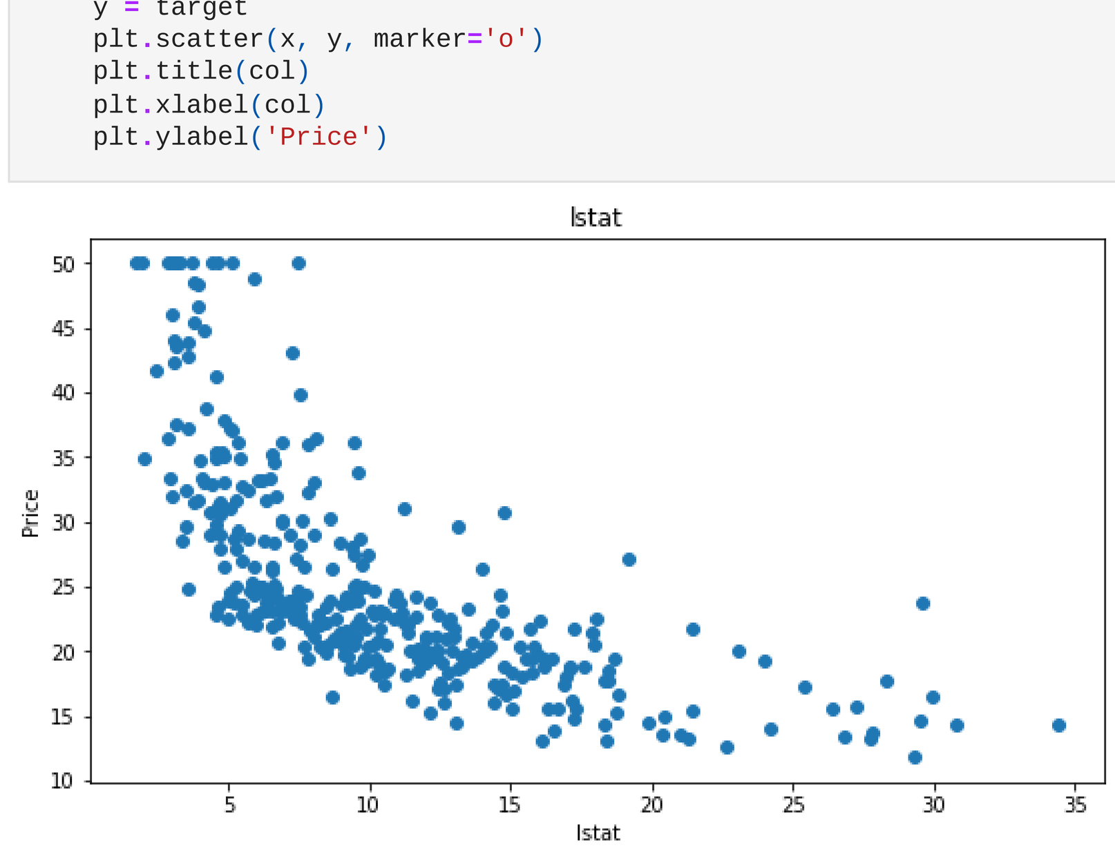
Out[15]:
```
<AxesSubplot:xlabel='rm'>
```

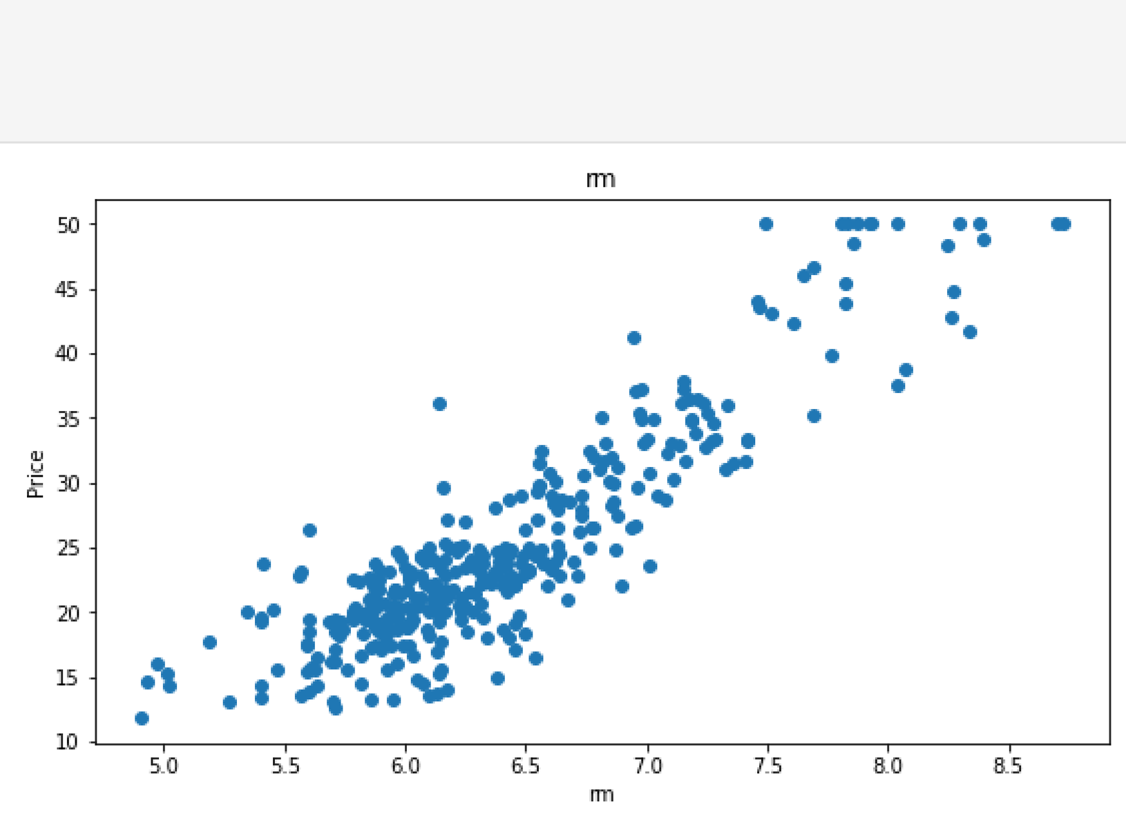


In [16]:
```python
plt.figure(figsize=(20, 5))

features = ['lstat', 'rm']
target = df['medv']

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





In [17]:
```python
###### DATA PRE-PROCESSING
```

In [18]:
```python
l = []
for c in df['medv']:
 if c == "M":
    l.append(0)
 else:
    l.append(1)
df['medv'] = 1
df
```

Out[18]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	1
1	1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	1
2	2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	1
3	3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	1
4	4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
346	346	0.06162	0.0	4.39	0	0.442	5.898	52.3	8.0136	3	352	18.8	364.61	12.67	1
347	347	0.01870	85.0	4.15	0	0.429	6.516	27.7	8.5353	4	351	17.9	392.43	6.36	1
348	348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.99	1
349	349	0.02899	40.0	1.25	0	0.429	6.939	34.5	8.7921	1	335	19.7	389.85	5.89	1
350	350	0.06211	40.0	1.25	0	0.429	6.490	44.4	8.7921	1	335	19.7	396.90	5.98	1

351 rows × 15 columns

In [19]:
```python
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [20]:
```python
le.fit_transform(df['medv'])
```

Out[20]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [21]:
```python
###### LINEAR REGRESSION MODEL
```

In [22]:
```python
#defining training set
df_train = pd.read_csv(r'file:///C:\Users\Kapil%20Chaudhary\Desktop\analysis\Boston_Train.csv')
```

In [23]:
```python
#defining testing set
df_test = pd.read_csv(r'file:///C:\Users\Kapil%20Chaudhary\Desktop\analysis\Boston_Test.csv')
```

In [24]:
```python
x_train = df_train['rm']
y_train = df_train['medv']
x_test = df_train['rm']
y_test = df_train['medv']
```

In [25]:
```python
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [26]:
```python
x_train = np.array(x_train).reshape(-1,1)
x_test = np.array(x_test).reshape(-1,1)
```

In [27]:
```python
model = reg.fit(x_train, y_train)
```

In [28]:
```python
model.coef_
```

Out[28]:
```
array([11.1942171])
```

In [29]:
```python
model.intercept_
```

Out[29]:
```
-46.62397204400658
```

In [30]:
```python
y_pred = model.predict(x_test)
```

In [31]:
```python
plt.scatter(x_test,y_pred, color='r',label ='Predicted')
plt.scatter(x_test,y_test, color='b',label ='Actual')
plt.xlabel('rm')
plt.ylabel('medv')
plt.legend()
plt.show()
```