DSBDAL PRACTICAL :- 4

In [1]:

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

In [5]:

```
BostonTrain = pd.read_csv('train.csv')
```

In [6]:

```
BostonTrain.head()
```

Out[6]:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	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	2	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	4	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
3	5	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
4	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9

In [7]:

```
BostonTrain.info()
BostonTrain.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   ID       333 non-null    int64
 1   crim     333 non-null    float64
 2   zn       333 non-null    float64
 3   indus    333 non-null    float64
 4   chas     333 non-null    int64
 5   nox      333 non-null    float64
 6   rm       333 non-null    float64
 7   age      333 non-null    float64
 8   dis      333 non-null    float64
 9   rad      333 non-null    int64
 10  tax      333 non-null    int64
 11  ptratio  333 non-null    float64
 12  black    333 non-null    float64
 13  lstat    333 non-null    float64
 14  medv     333 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 39.1 KB
```

Out[7]:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad
count	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000
mean	250.951952	3.360341	10.689189	11.293483	0.060060	0.557144	6.265619	68.226426	3.709934	9.633634
std	147.859438	7.352272	22.674762	6.998123	0.237956	0.114955	0.703952	28.133344	1.981123	8.742174
min	1.000000	0.006320	0.000000	0.740000	0.000000	0.385000	3.561000	6.000000	1.129600	1.000000
25%	123.000000	0.078960	0.000000	5.130000	0.000000	0.453000	5.884000	45.400000	2.122400	4.000000
50%	244.000000	0.261690	0.000000	9.900000	0.000000	0.538000	6.202000	76.700000	3.092300	5.000000
75%	377.000000	3.678220	12.500000	18.100000	0.000000	0.631000	6.595000	93.800000	5.116700	24.000000
max	506.000000	73.534100	100.000000	27.740000	1.000000	0.871000	8.725000	100.000000	10.710300	24.000000

In [8]:

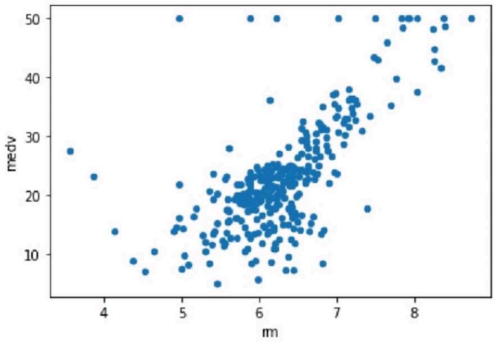
```
BostonTrain.drop('ID', axis = 1, inplace=True)
```

In [9]:

```
BostonTrain.plot.scatter('rm', 'medv')
```

Out[9]:

```
<AxesSubplot:xlabel='rm', ylabel='medv'>
```

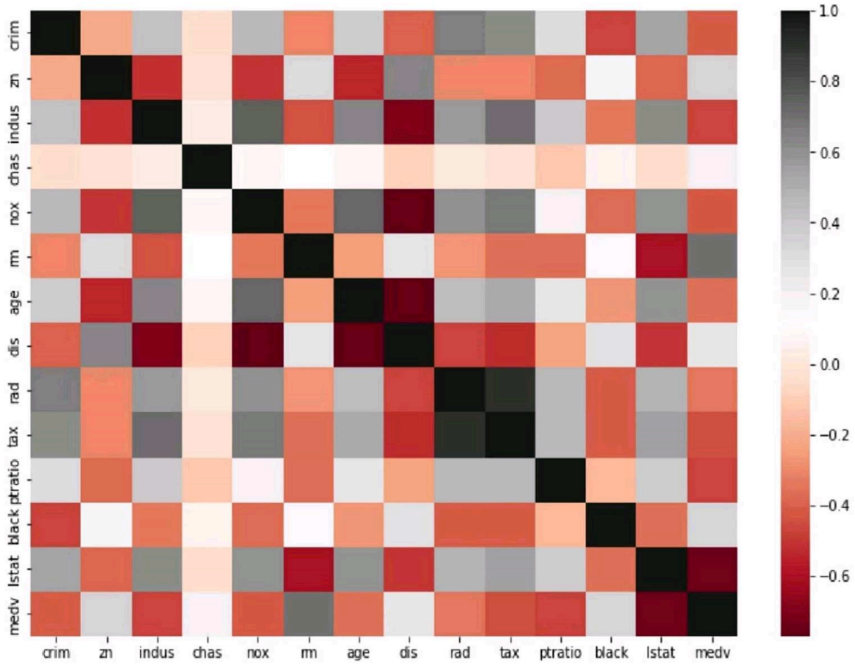


In [10]:

```
plt.subplots(figsize=(12,8))
sns.heatmap(BostonTrain.corr(), cmap = 'RdGy')
```

Out[10]:

```
<AxesSubplot:>
```



In [14]:

```
X = BostonTrain[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = BostonTrain['medv']
```

In [17]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [18]:

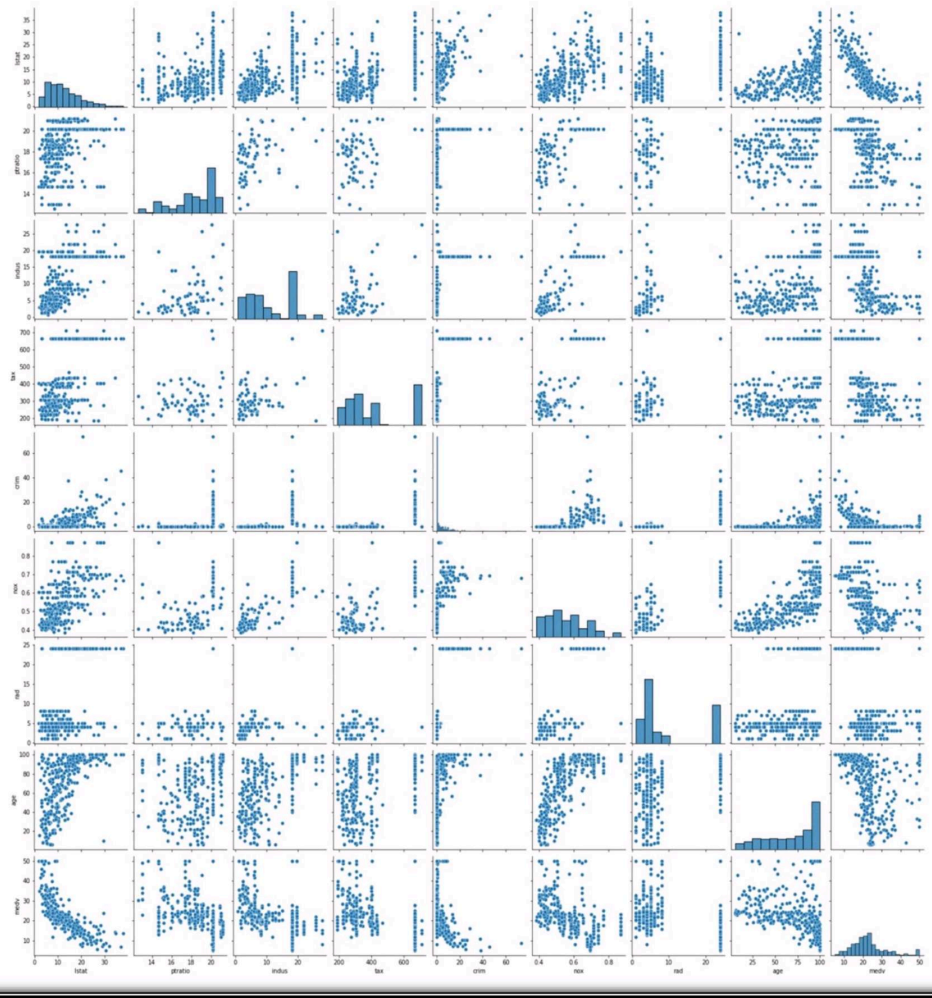
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

In [11]:

```
sns.pairplot(BostonTrain, vars = ['lstat', 'ptratio', 'indus', 'tax', 'crim', 'nox', 'rad', 'age', 'medv'])
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7f0600342f10>
```

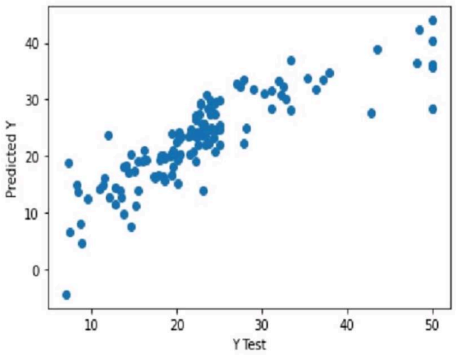


In [21]:

```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[21]:

```
Text(0, 0.5, 'Predicted Y')
```

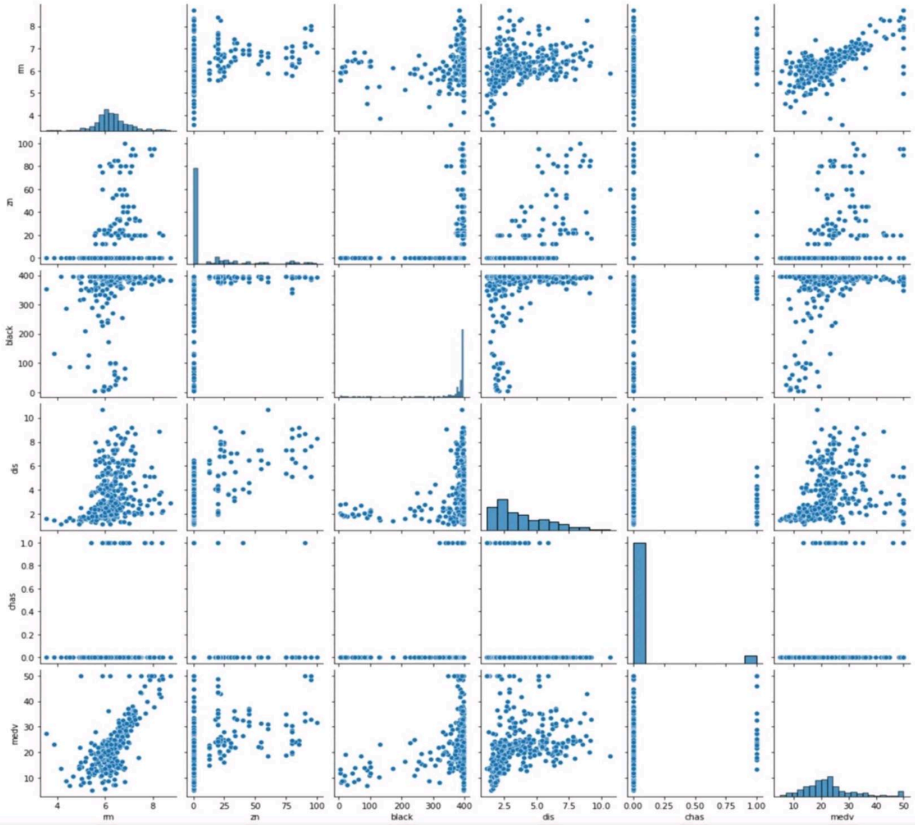


In [12]:

```
sns.pairplot(BostonTrain, vars = ['rm', 'zn', 'black', 'dis', 'chas','medv'])
```

Out[12]:

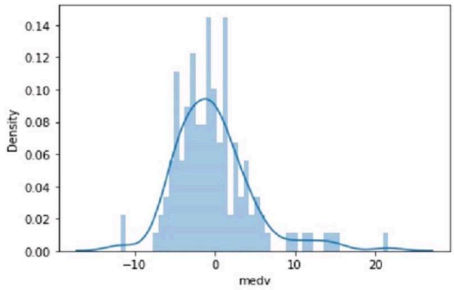
```
<seaborn.axisgrid.PairGrid at 0x7f05be72b910>
```



In [23]:

```
sns.distplot((y_test-predictions),bins=50);
```

```
/home/admin1/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `hist
plot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [25]:

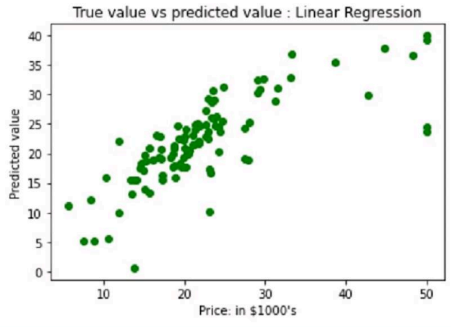
```
coefficients = pd.DataFrame(lm.coef_,X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[25]:

	coefficients
crim	-0.113416
zn	0.066909
indus	0.000361
chas	2.572606
nox	-18.318718
rm	2.973366
age	0.011659
dis	-1.807005
rad	0.415772
tax	-0.016038
ptratio	-0.755570
black	0.012601
lstat	-0.615696

In [43]:

```
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```



In [28]:

```
boston.data.shape
```

Out[28]:

```
(506, 13)
```

In [29]:

```
boston.feature_names
```

Out[29]:

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [30]:

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

In [31]:

```
data.head(10)
```

Out[31]:

	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
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

In [32]:

```
boston.target.shape
```

Out[32]:

```
(506,)
```

In [33]:

```
data['Price'] = boston.target
data.head()
```

Out[33]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
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	24.0
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	21.6
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	34.7
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	33.4
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	36.2

In [34]:

```
data.describe()
```

Out[34]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000