In [6]:
```
#Linear regression by using Deep Neural network: Implement Boston housing price
#prediction problem by Linear regression using Deep Neural network. Use Boston House price
#predictiondataset.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [8]:
```
# Importing DataSet and take a look at Data
BostonTrain = pd.read_csv("boston_train.csv")
```

In [9]:
```
BostonTrain.head()
```

Out[9]:

| | 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 [10]:
```
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
```

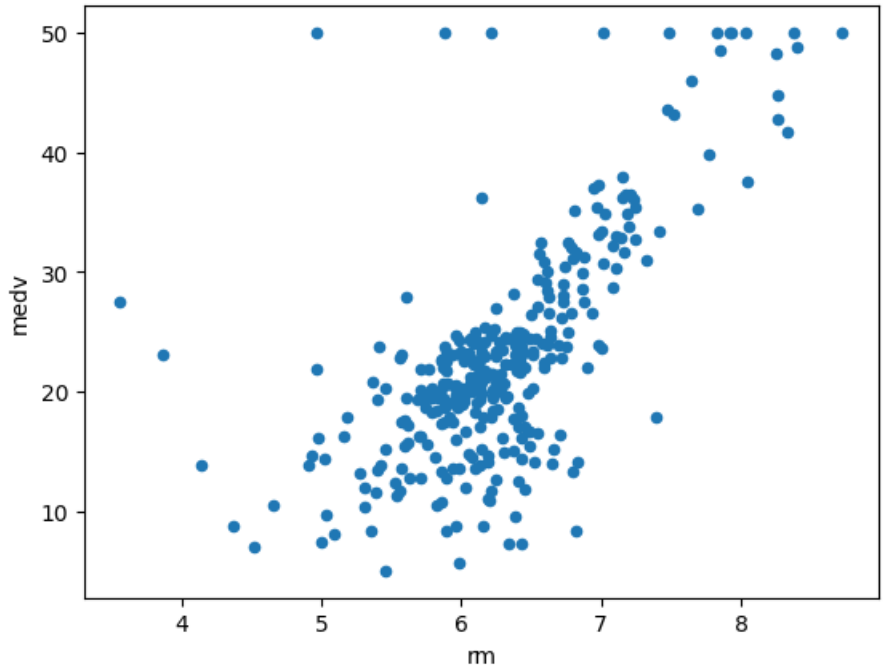
Out[10]:

| | ID | crim | zn | indus | chas | nox | rm | age | dis | |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333 |
| **mean** | 250.951952 | 3.360341 | 10.689189 | 11.293483 | 0.060060 | 0.557144 | 6.265619 | 68.226426 | 3.709934 | 9 |
| **std** | 147.859438 | 7.352272 | 22.674762 | 6.998123 | 0.237956 | 0.114955 | 0.703952 | 28.133344 | 1.981123 | 8 |
| **min** | 1.000000 | 0.006320 | 0.000000 | 0.740000 | 0.000000 | 0.385000 | 3.561000 | 6.000000 | 1.129600 | 1 |
| **25%** | 123.000000 | 0.078960 | 0.000000 | 5.130000 | 0.000000 | 0.453000 | 5.884000 | 45.400000 | 2.122400 | 4 |
| **50%** | 244.000000 | 0.261690 | 0.000000 | 9.900000 | 0.000000 | 0.538000 | 6.202000 | 76.700000 | 3.092300 | 5 |
| **75%** | 377.000000 | 3.678220 | 12.500000 | 18.100000 | 0.000000 | 0.631000 | 6.595000 | 93.800000 | 5.116700 | 24 |
| **max** | 506.000000 | 73.534100 | 100.000000 | 27.740000 | 1.000000 | 0.871000 | 8.725000 | 100.000000 | 10.710300 | 24 |

In [11]:
```
#ID columns does not relevant for our analysis.
BostonTrain.drop('ID', axis = 1, inplace=True)
```

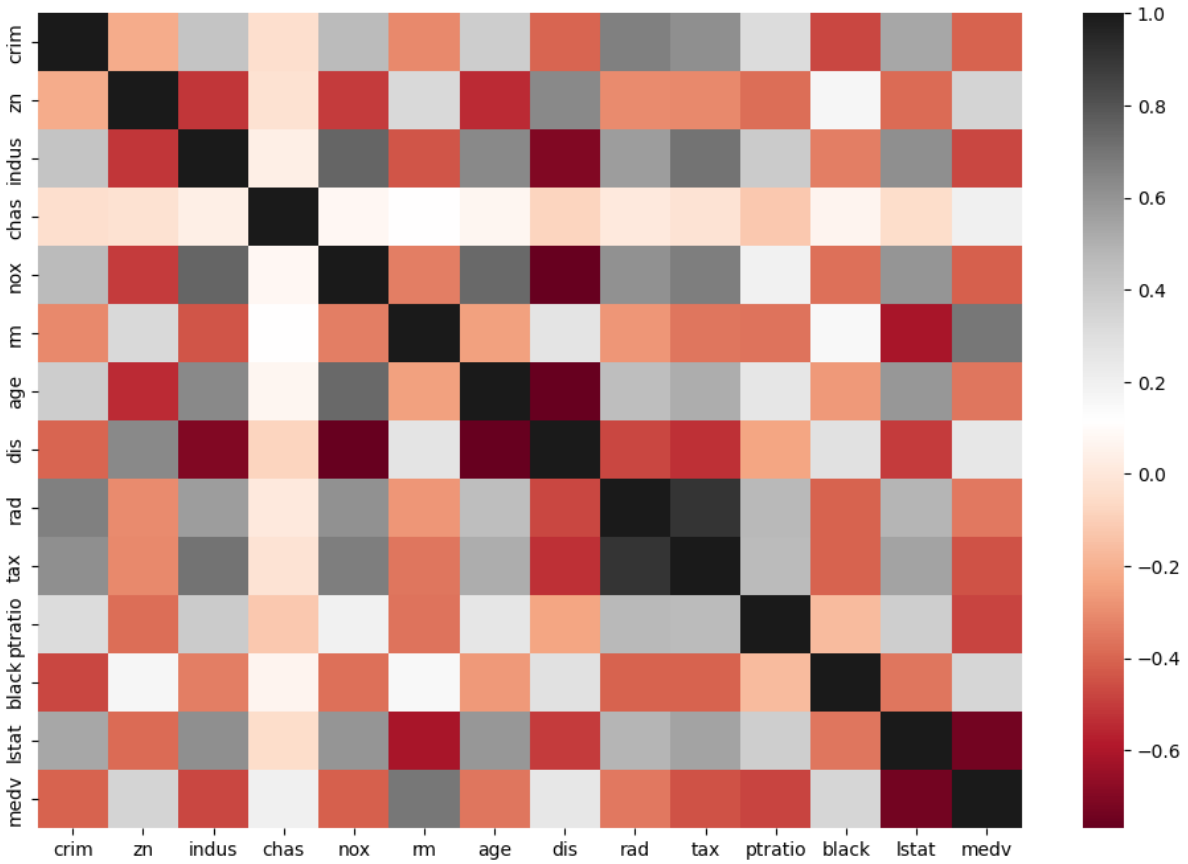
In [12]:
```
BostonTrain.plot.scatter('rm', 'medv')
```

Out[12]:
```
<Axes: xlabel='rm', ylabel='medv'>
```



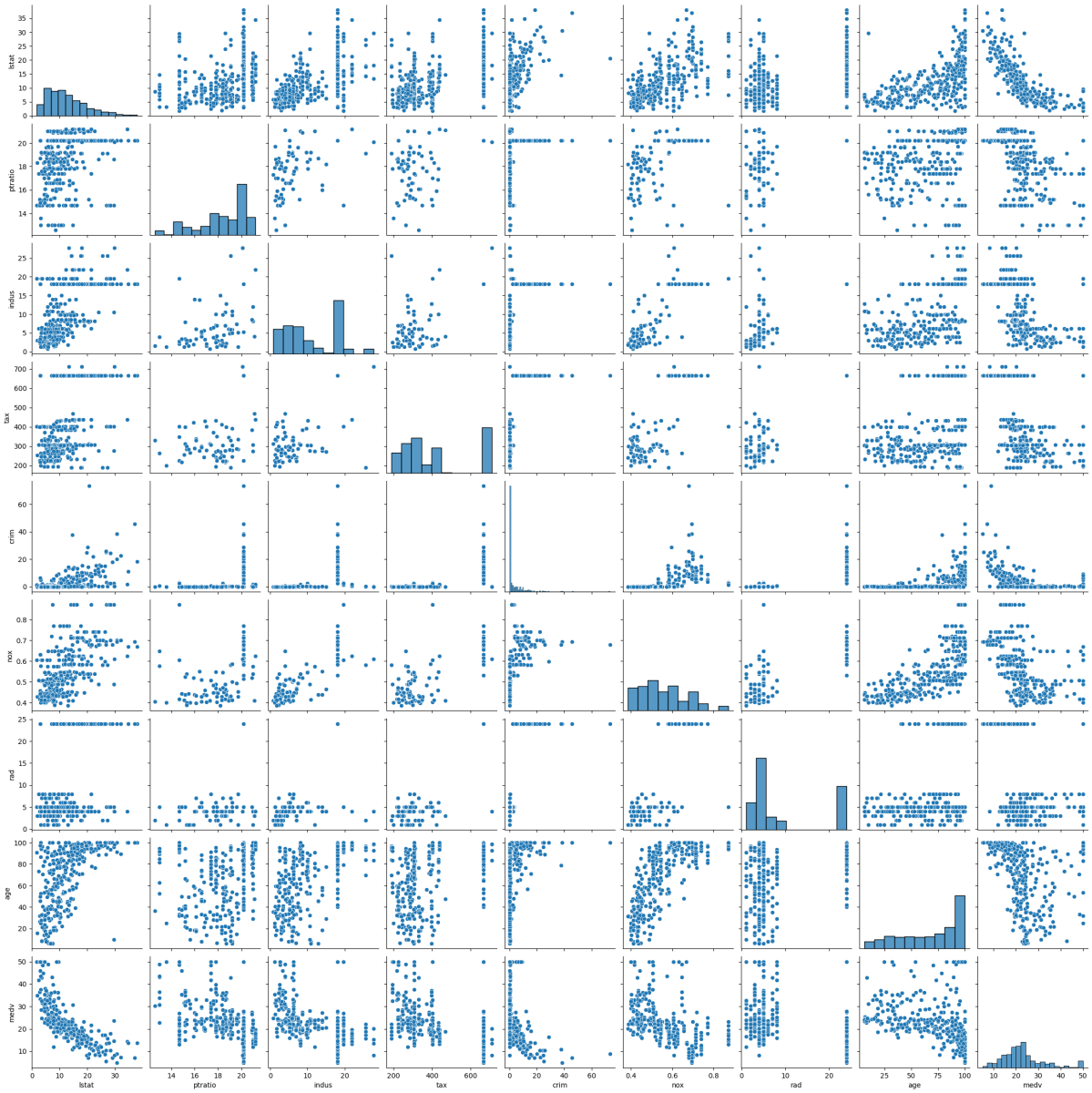

In [13]:
```
plt.subplots(figsize=(12,8))
sns.heatmap(BostonTrain.corr(), cmap = 'RdGy')
```

Out[13]: <Axes: >



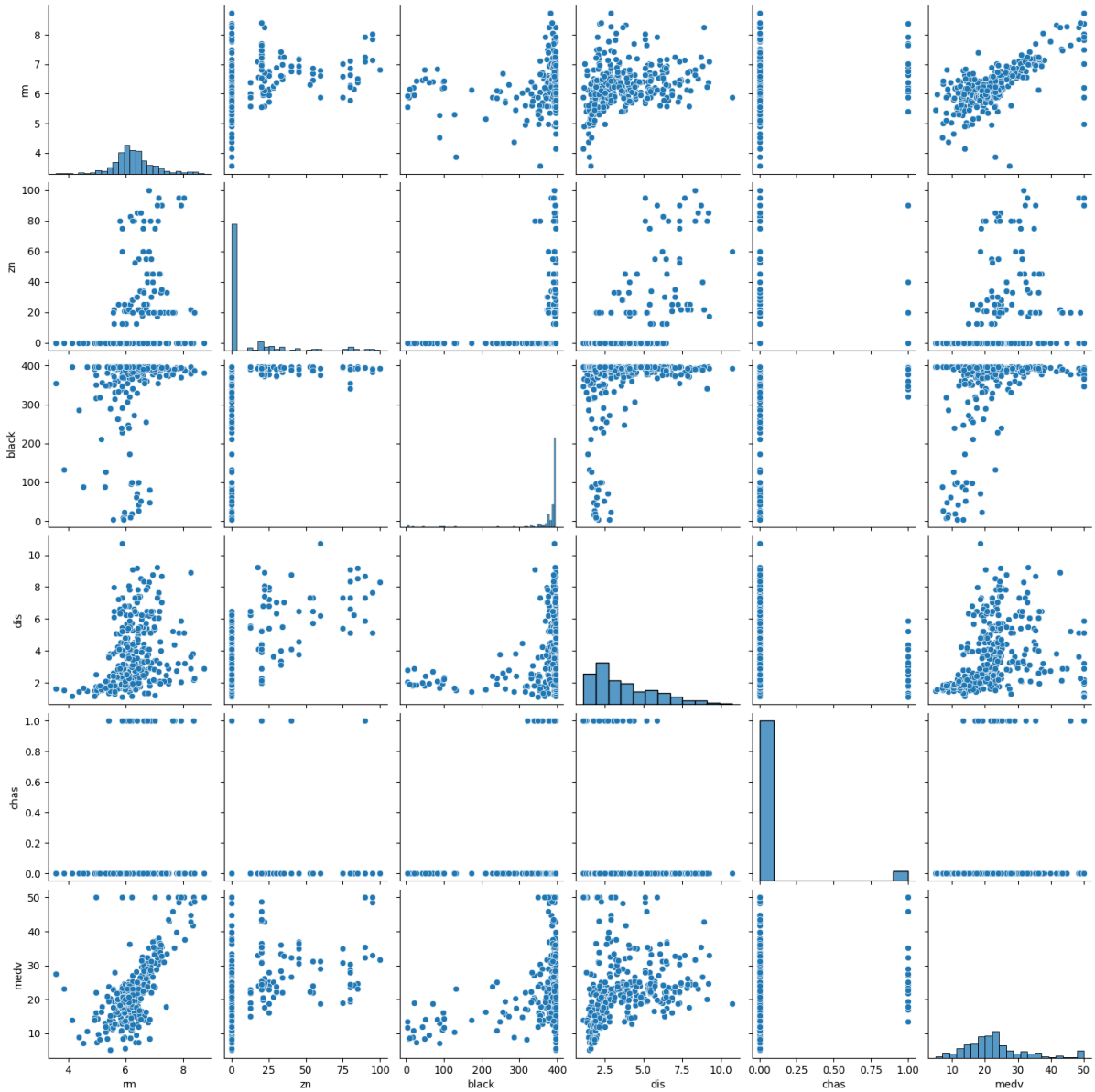

In [14]:
```
sns.pairplot(BostonTrain, vars = ['lstat', 'ptratio', 'indus', 'tax', 'crim', 'nox', 'rad', 'ag
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x7f2b7c831e40>




In [15]:
```
sns.pairplot(BostonTrain, vars = ['rm', 'zn', 'black', 'dis', 'chas','medv'])
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x7f2b6f675b10>
```




In [16]:
```
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)
```

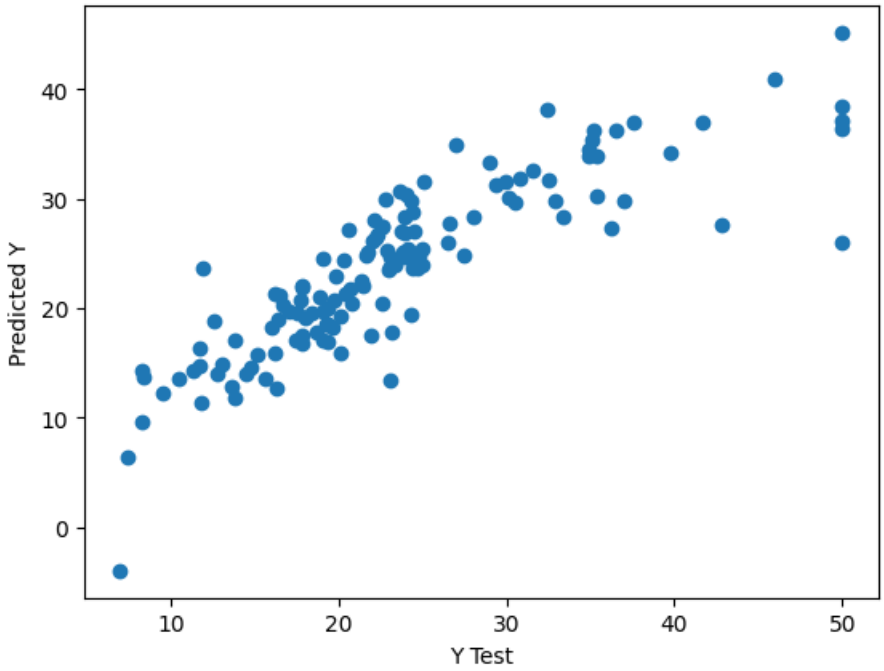
In [19]:
```
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression
LinearRegression()
```

In [20]:
```
predictions = lm.predict(X_test)
```

In [21]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[21]:
```
Text(0, 0.5, 'Predicted Y')
```




In [22]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.300709932401183
MSE: 22.97276220573061
RMSE: 4.792990945717571
```

In [23]:
```
sns.distplot((y_test-predictions),bins=50);
```

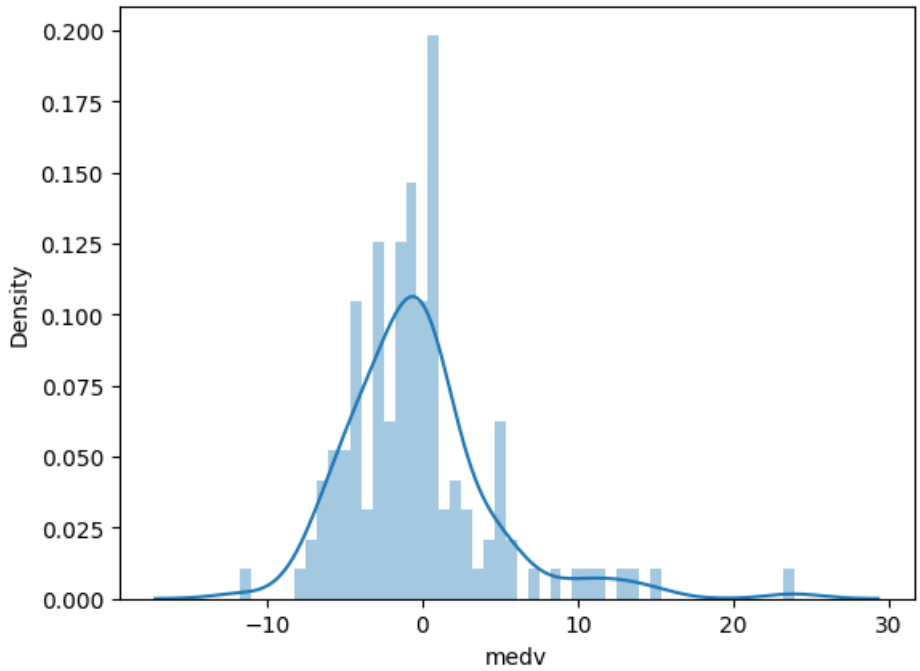
```
/tmp/ipykernel_4418/1326397652.py:1: UserWarning:

`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((y_test-predictions),bins=50);
```



In [24]:
```
coefficients = pd.DataFrame(lm.coef_,X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[24]:

| | coefficients |
|---|---|
| **crim** | -0.102833 |
| **zn** | 0.053870 |
| **indus** | -0.005014 |
| **chas** | 4.278293 |
| **nox** | -13.918157 |
| **rm** | 2.877124 |
| **age** | 0.000062 |
| **dis** | -1.724588 |
| **rad** | 0.327951 |
| **tax** | -0.014207 |
| **ptratio** | -0.861155 |
| **black** | 0.010960 |
| **lstat** | -0.606912 |

In [ ]:
```
```