Boston Housing with Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
BostonTrain = pd.read_csv("boston_train.csv")
```

In [3]:
```
BostonTrain.head()
```

Out[3]:

	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 [4]:
```
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[4]:

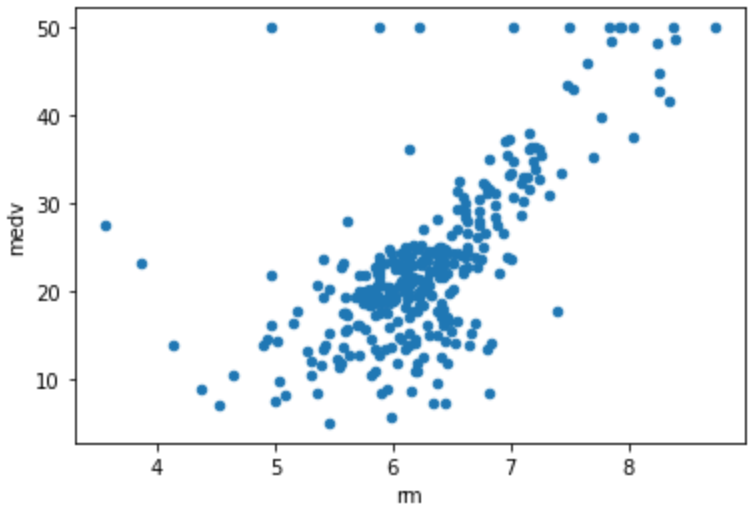
	ID	crim	zn	indus	chas	nox	rm	age	dis
count	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000
mean	250.951952	3.360341	10.689189	11.293483	0.060060	0.557144	6.265619	68.226426	3.709934
std	147.859438	7.352272	22.674762	6.998123	0.237956	0.114955	0.703952	28.133344	1.981123
min	1.000000	0.006320	0.000000	0.740000	0.000000	0.385000	3.561000	6.000000	1.129600
25%	123.000000	0.078960	0.000000	5.130000	0.000000	0.453000	5.884000	45.400000	2.122400
50%	244.000000	0.261690	0.000000	9.900000	0.000000	0.538000	6.202000	76.700000	3.092300

75%	377.000000	3.678220	12.500000	18.100000	0.000000	0.631000	6.595000	93.800000	5.116700
max	506.000000	73.534100	100.000000	27.740000	1.000000	0.871000	8.725000	100.000000	10.710300

In [5]:
```
BostonTrain.drop('ID', axis = 1, inplace=True)
```

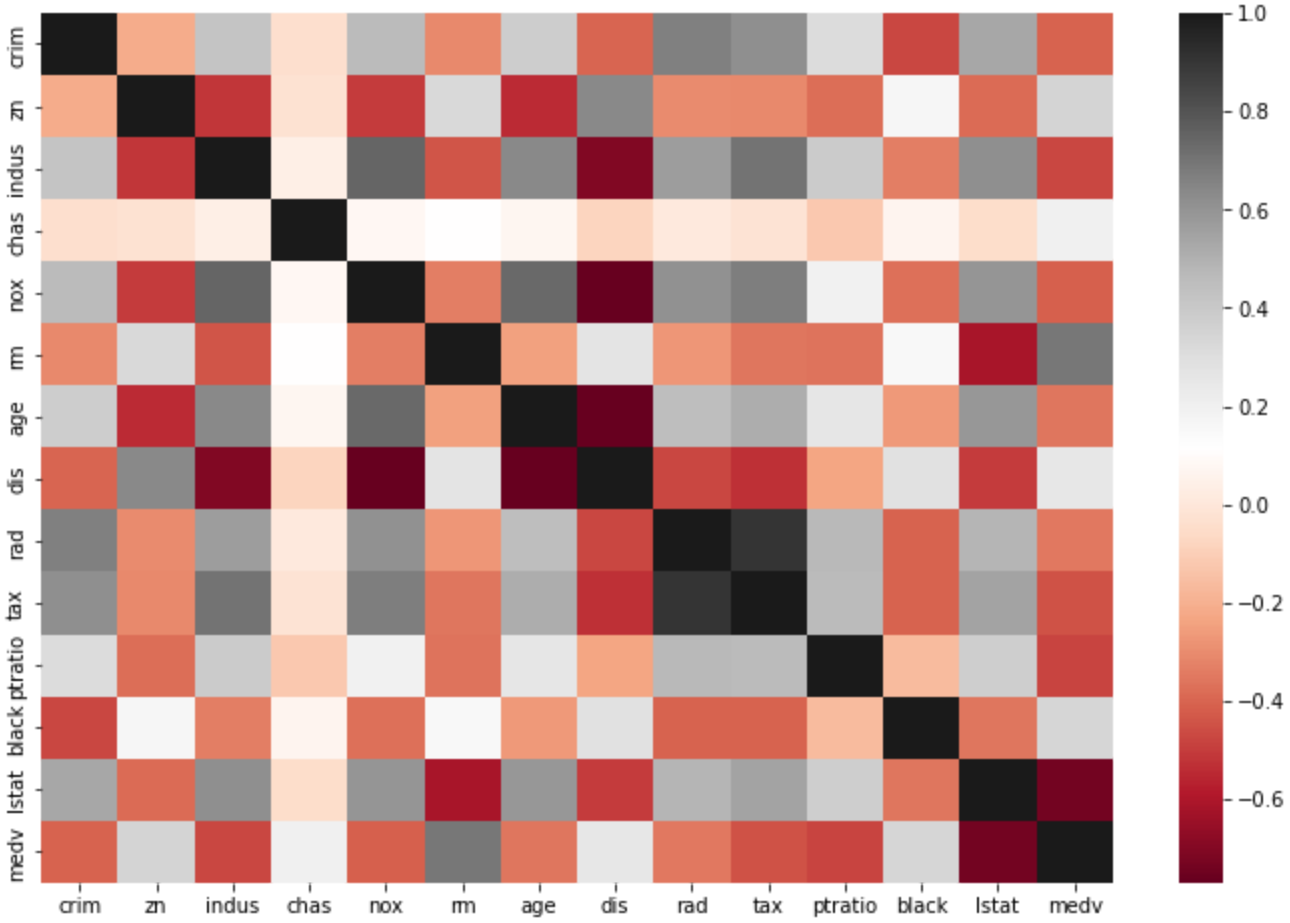
In [6]:
```
BostonTrain.plot.scatter('rm', 'medv')
```

Out[6]:
```
<AxesSubplot:xlabel='rm', ylabel='medv'>
```



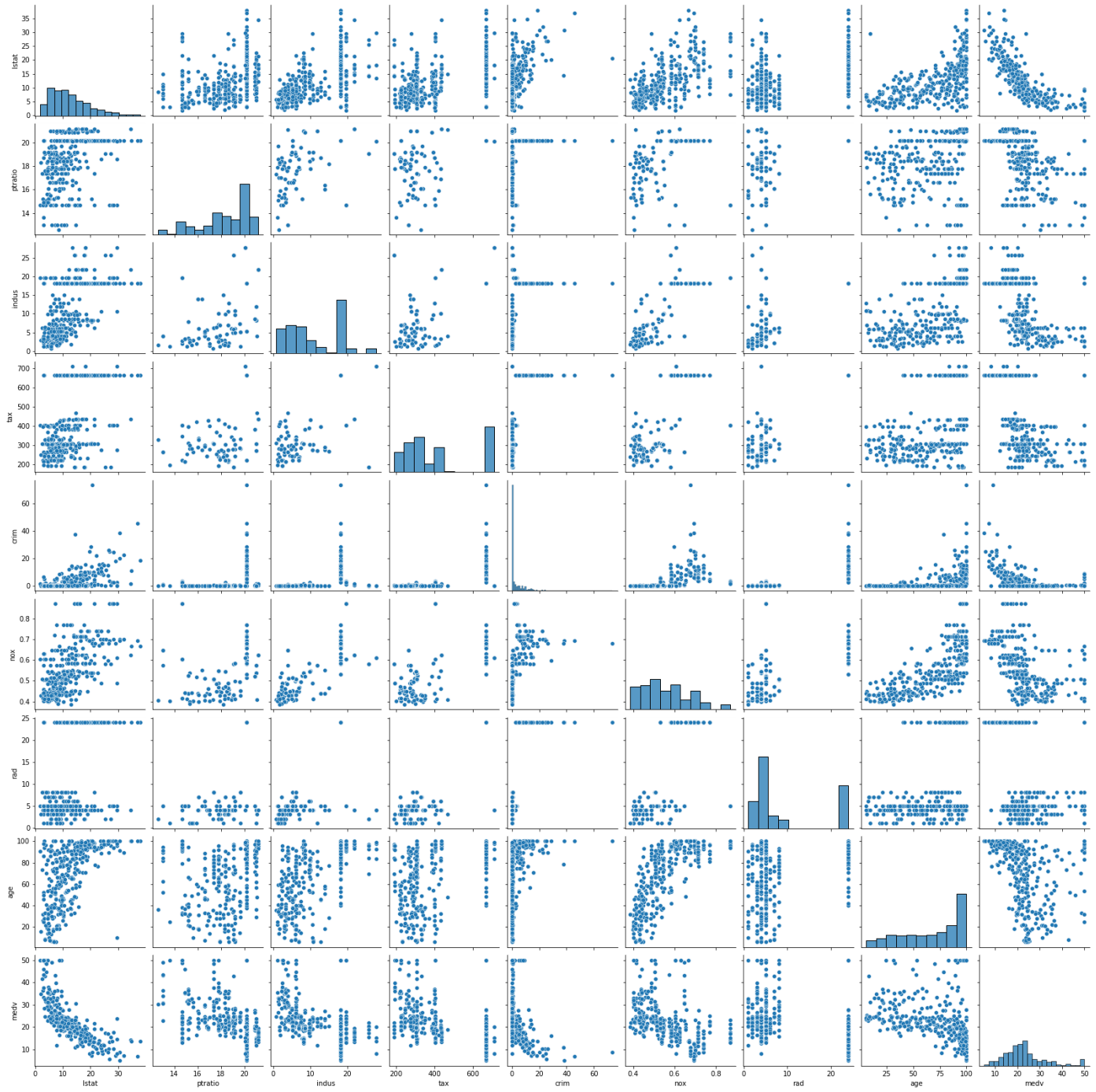
In [7]:
```
plt.subplots(figsize=(12,8))
sns.heatmap(BostonTrain.corr(), cmap = 'RdGy')
```

Out[7]:
```
<AxesSubplot:>
```



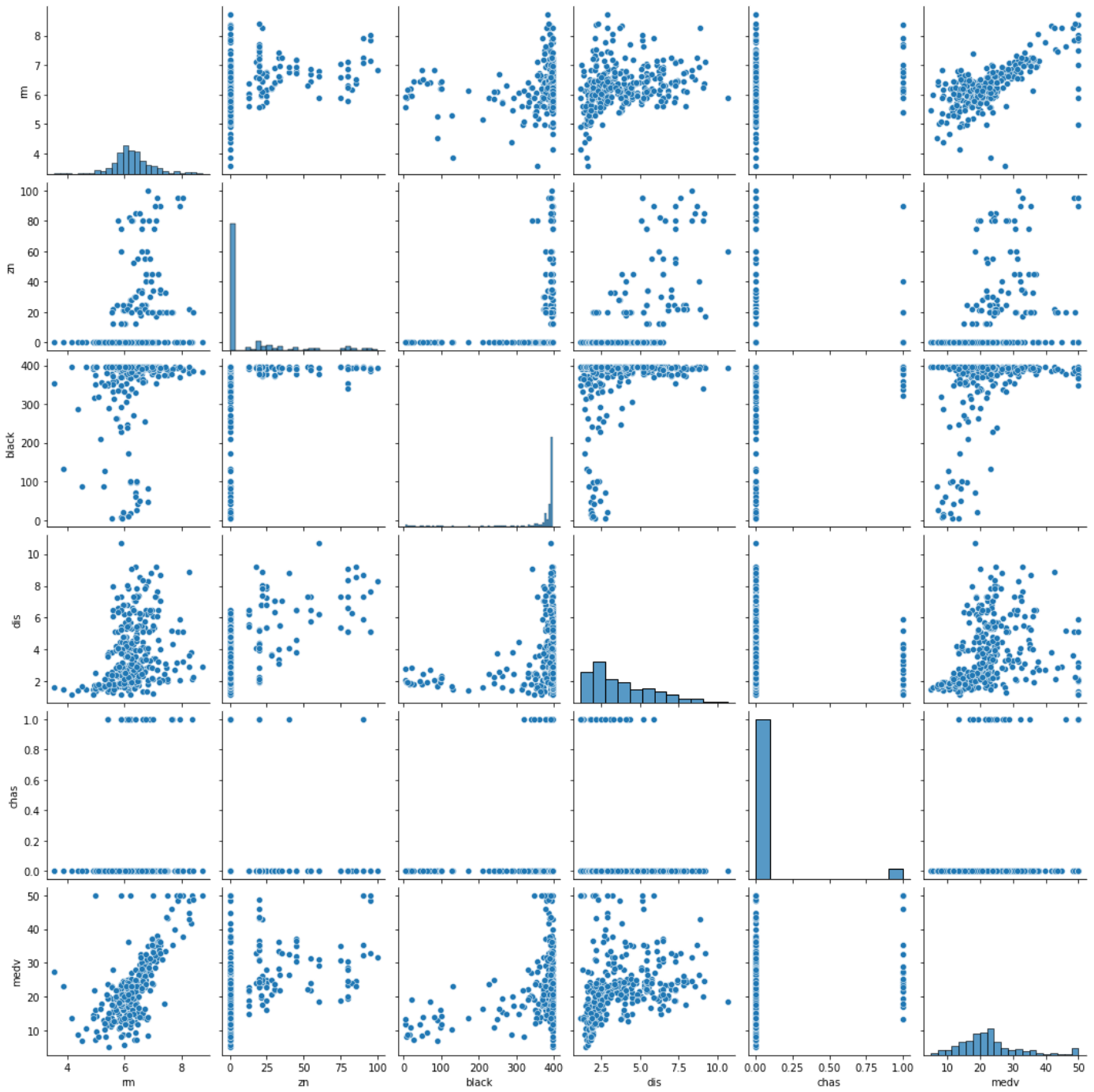
In [8]:
```
sns.pairplot(BostonTrain, vars = ['lstat', 'ptratio', 'indus', 'tax', 'crim', 'nox', 'ra
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x1cd076c2d90>



In [9]:
```
sns.pairplot(BostonTrain, vars = ['rm', 'zn', 'black', 'dis', 'chas','medv'])
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x1cd0b2e2880>



In [10]:
```
X = BostonTrain[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = BostonTrain['medv']
```

In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

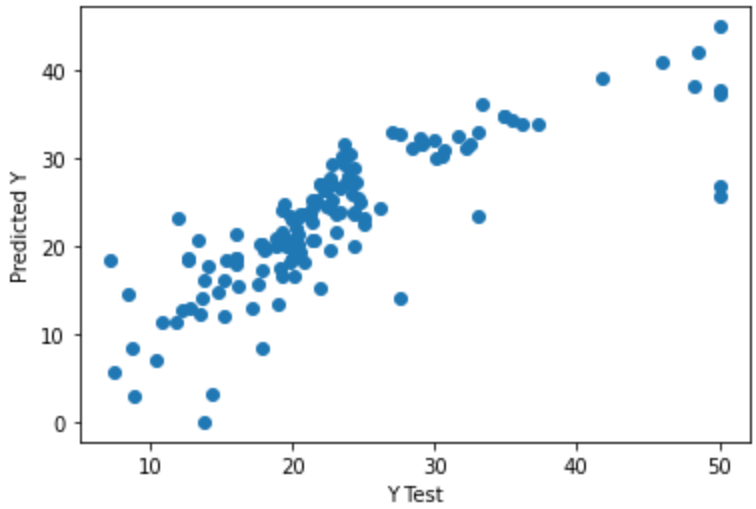
In [13]:
```
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
predictions = lm.predict(X_test)
```

In [15]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[15]:
```
Text(0, 0.5, 'Predicted Y')
```



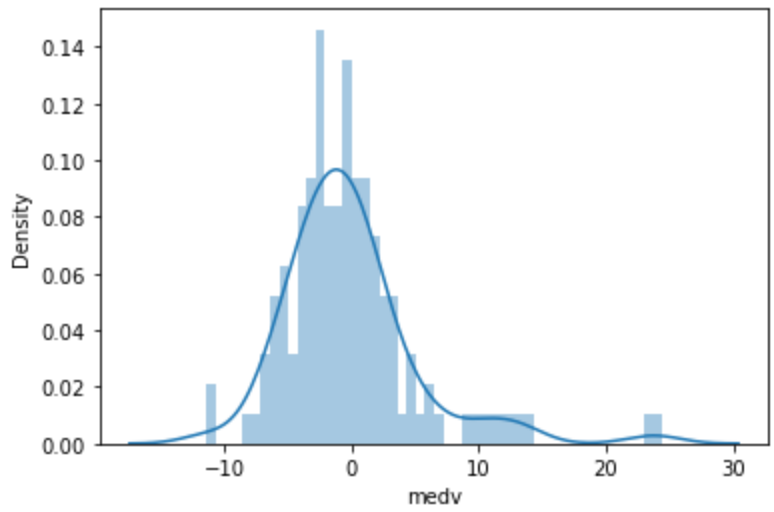
In [16]:
```
from sklearn import metrics

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: 3.6343448898432333
MSE: 28.33134403210523
RMSE: 5.322719608631027
```

In [17]:
```
sns.distplot((y_test-predictions),bins=50);
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [18]:
```
coefficients = pd.DataFrame(lm.coef_,X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[18]:

	coefficients
crim	-0.042923
zn	0.039557
indus	-0.013613

chas	2.994593
nox	-16.931949
rm	4.128603
age	-0.003357
dis	-1.589776
rad	0.255408
tax	-0.010798
ptratio	-0.803241
black	0.011224
lstat	-0.577529