In [1]:

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

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
from scipy.stats import skew
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.preprocessing import PolynomialFeatures
```

In [3]:

```
df = pd.read_csv("BostonHousing.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	med
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.
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.
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.
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	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
crim       506 non-null float64
zn         506 non-null float64
indus      506 non-null float64
chas       506 non-null int64
nox        506 non-null float64
rm         506 non-null float64
age        506 non-null float64
dis        506 non-null float64
rad        506 non-null int64
tax        506 non-null int64
ptratio    506 non-null float64
b          506 non-null float64
lstat      506 non-null float64
medv       506 non-null float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

In [65]:

```
cor = df.corr()
```

In [64]:

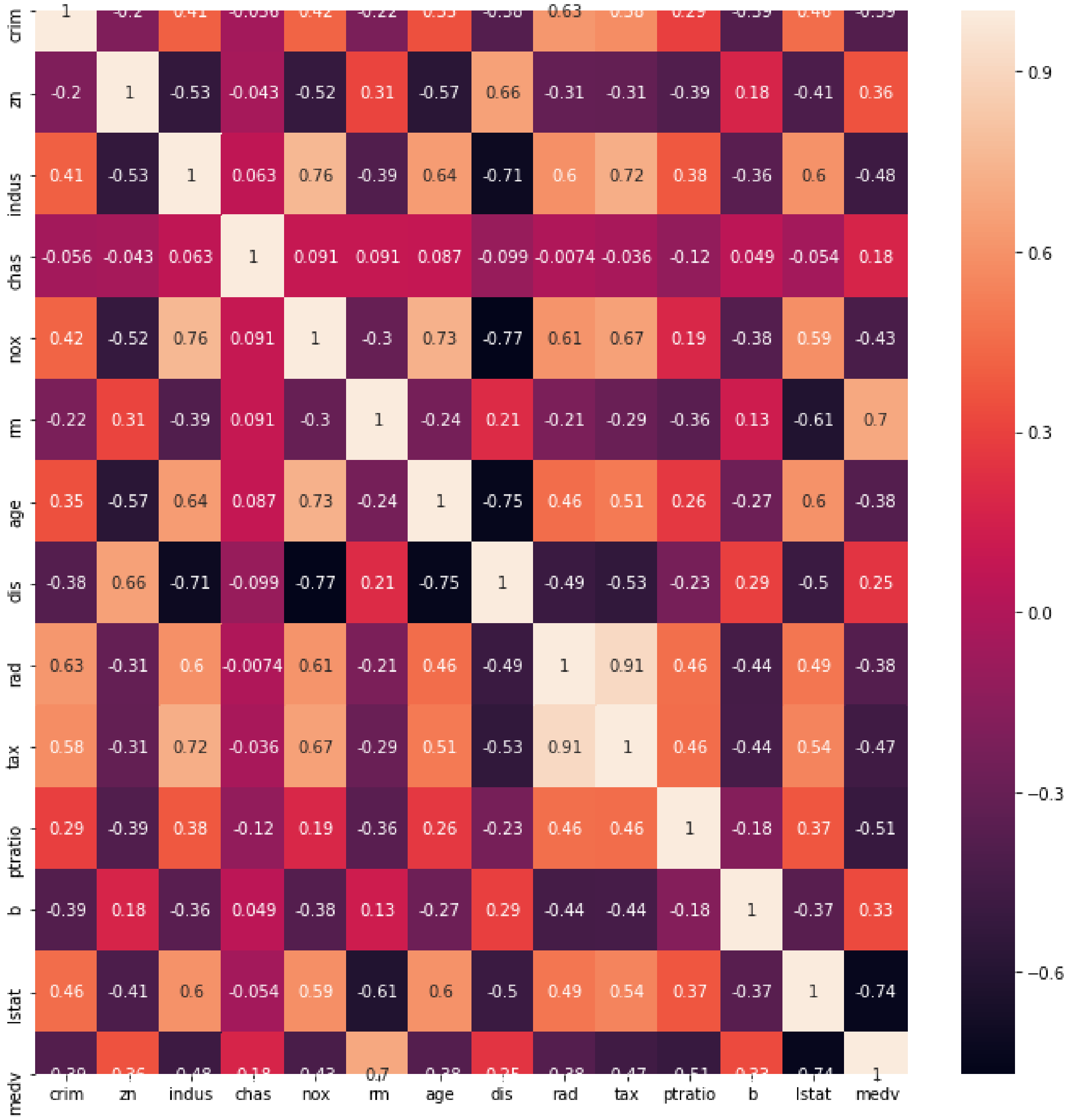
```
cor
```

Out[64]:

	crim	zn	indus	chas	nox	rm	age	dis	
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0
b	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0
medv	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0

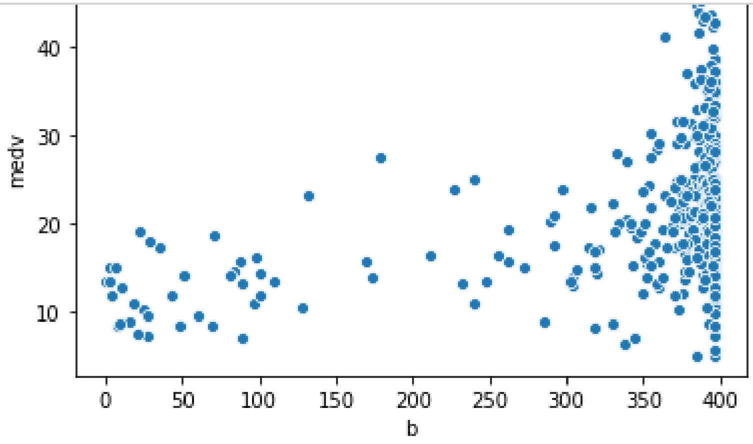
In [15]:

```
plt.figure(figsize=(12,12))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [9]:

```
for col in df:
    print("Column:",col)
    plt.figure()
    sns.scatterplot(data=df,x=col,y="medv")
    plt.show()
    print("-------------------------------------------------------------------------")
```



```
-------------------------------------------------------------------------
Column: lstat
```

In [21]:

```
df.columns
```

Out[21]:

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'ta
x',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

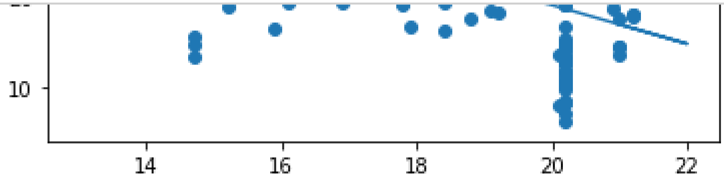
In [10]:

```
def lr_model(X,y):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=1)
    model = LinearRegression()
    model.fit(X_train,y_train)

    print("Modeling with ",X.columns[0])
    print("intercept: ",model.intercept_)
    print("Coef: ",model.coef_)

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test,y_pred)
    rmse = np.sqrt(mse)

    r2 = r2_score(y_test,y_pred)

    print("mse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))

    # Plot the model
    plt.figure()
    plt.scatter(X_test,y_test)
    plt.plot(X_test,y_pred)
    plt.show()
    print("---------------------------------------------------------------------")
```

Builiding each separate model

In [11]:

```
for col in df.iloc[:,:-1]:
    lr_model(df[[col]],df["medv"])
```



```
---------------------------------------------------------------------
Modeling with  b
intercept:  10.801659976646253
Coef:  [0.03230252]
mse: 80.69014287483378,
rmse: 8.982769220837959,
r2: 0.11962692251813789
```

Building combine model

In [60]:

```
X = df.drop(["medv"],axis=1)
y = df["medv"]
```

In [117]:

```
def mlr_model(x,y):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=1)
    model = LinearRegression()
    model.fit(X_train,y_train)

    print("intercept: ",model.intercept_)
    c = -1
    for col in X:
        c = c + 1
        print(f"Coef of {col}:",model.coef_[c])

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test,y_pred)
    rmse = np.sqrt(mse)

    r2 = r2_score(y_test,y_pred)

    print("\nmse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

In [118]:

```
mlr_model(X,y)
```

```
intercept:  38.74669344410249
Coef of chas: 2.8500826816191687
Coef of nox: -17.50458442807624
Coef of rm: 3.365163538385576
Coef of dis: -1.0665911118873166
Coef of ptratio: -1.009338915104558
Coef of b: 0.005751015708256091
Coef of lstat: -0.5647744696441257

mse: 20.80394418616361,
rmse: 4.561134089912684,
r2: 0.7730177229287565
```

In [84]:

```
## Coefficient single and combine
## crim  :  -0.42    -0.09    remove
## zn    :   0.14     0.06    remove
## indus :  -0.62     0.05    remove
## chase :   5.94     2.43    keep
## nox   :  -34.60   -21.46   keep
## rm    :   8.46     2.79    keep
## age   :  -0.13     0.00    remove
## dis   :   1.24    -1.51    keep
## rad   :  -0.38     0.30    remove
## tax   :  -0.02    -0.01    remove
## ptratio :-2.13    -1.00    keep
## b       : 0.03     0.00    keep
## lstat : -0.91     -0.56    keep
```

In [66]:

```
cor
```

Out[66]:

	crim	zn	indus	chas	nox	rm	age	dis	
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0
b	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0
medv	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0

Taking care of multicolinearity

In [119]:

```
X = df[['chas','nox','rm','dis','ptratio','b','lstat']]
y = df["medv"]
```

In [120]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=1)
model = LinearRegression()
model.fit(X_train,y_train)

print("intercept: ",model.intercept_)
c = -1
for col in X:
    c = c + 1
    print(f"Coef of {col}:",model.coef_[c])

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)

r2 = r2_score(y_test,y_pred)

print("\nmse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

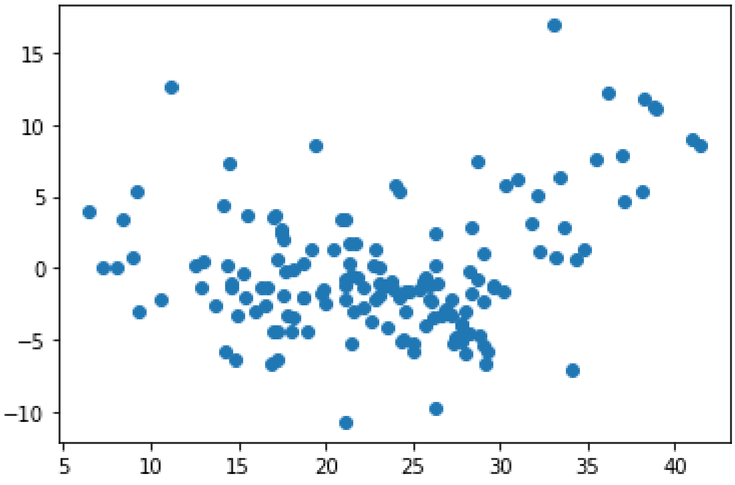
```
intercept:  38.74669344410249
Coef of chas: 2.8500826816191687
Coef of nox: -17.50458442807624
Coef of rm: 3.365163538385576
Coef of dis: -1.0665911118873166
Coef of ptratio: -1.009338915104558
Coef of b: 0.005751015708256091
Coef of lstat: -0.5647744696441257

mse: 20.80394418616361,
rmse: 4.561134089912684,
r2: 0.7730177229287565
```

In [121]:

```
residuals = y_test - y_pred
```

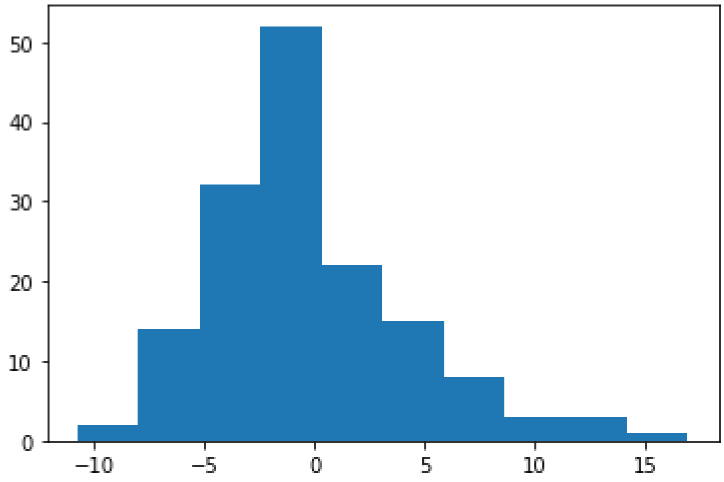
In [122]:

```
plt.figure()
plt.scatter(y_pred,residuals)
plt.show()
```



In [123]:

```
plt.figure()
plt.hist(residuals)
plt.show()
# Positively skewed
```



In [85]:

```
# This show clearly no linear relationship
# but can see a curve.
```

Polynomial Regression

In [127]:

```
X = df[['chas','nox','rm','dis','ptratio','b','lstat']]
y = df["medv"]
```

In [128]:

```
poly = PolynomialFeatures(2)
```

In [129]:

```
X_poly = poly.fit_transform(X)
```

In [130]:

```
mlr_model(X_poly,y)
```

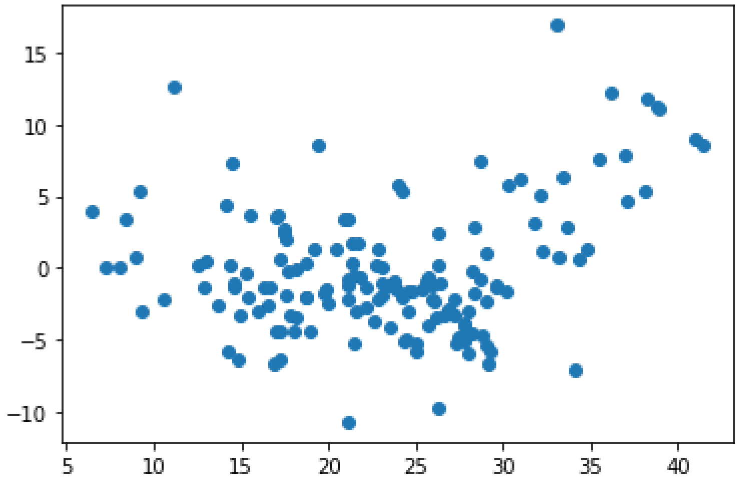
```
intercept:  38.74669344410249
Coef of chas: 2.8500826816191687
Coef of nox: -17.50458442807624
Coef of rm: 3.365163538385576
Coef of dis: -1.0665911118873166
Coef of ptratio: -1.009338915104558
Coef of b: 0.005751015708256091
Coef of lstat: -0.5647744696441257

mse: 20.80394418616361,
rmse: 4.561134089912684,
r2: 0.7730177229287565
```

In [131]:

```
residuals = y_test - y_pred
```

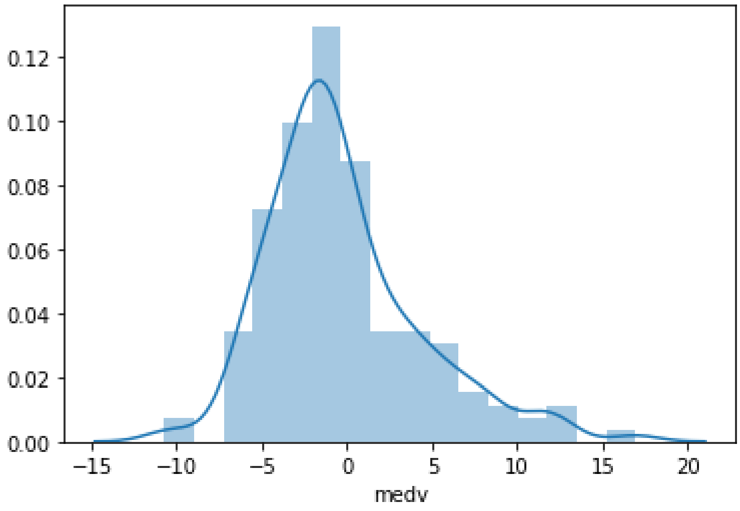
In [132]:

```
plt.figure()
# sns.scatterplot(y_pred,residuals)
plt.scatter(y_pred,residuals)
plt.show()
```



In [133]:

```
plt.figure()
#plt.hist(residuals)
sns.distplot(residuals)
plt.show()
```



In [ ]:

```
```