# RegressionReview

September 29, 2021

**SL Lab with Python 3: Regression Review**

**Statistical Learning (Sejong University)**

**Date: 2021.09.27 (By: S. M. Riazul Islam)**

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

        import statsmodels.api as sm
        import statsmodels.formula.api as smf


        %matplotlib inline
        plt.style.use('seaborn-white')
```

```
In [2]: housing = pd.read_csv('C:/DataSL/Boston.csv')
        housing.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
black      506 non-null float64
lstat      506 non-null float64
medv       506 non-null float64
```

```
dtypes: float64(11), int64(3)
memory usage: 55.4 KB
```

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

Out[3]:

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | \ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | |
| 1 | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | |
| 2 | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | |
| 3 | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | |
| 4 | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | |

| | black | lstat | medv |
|---|---|---|---|
| 0 | 396.90 | 4.98 | 24.0 |
| 1 | 396.90 | 9.14 | 21.6 |
| 2 | 392.83 | 4.03 | 34.7 |
| 3 | 394.63 | 2.94 | 33.4 |
| 4 | 396.90 | 5.33 | 36.2 |

```
In [4]: #         - CRIM per capita crime rate by town
        #         - ZN proportion of residential land zoned for lots over 25,000 sq.ft.
        #         - INDUS proportion of non-retail business acres per town
        #         - CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        #         - NOX nitric oxides concentration (parts per 10 million)
        #         - RM average number of rooms per dwelling
        #         - AGE proportion of owner-occupied units built prior to 1940
        #         - DIS weighted distances to five Boston employment centers
        #         - RAD index of accessibility to radial highways
        #         - TAX full-value property-tax rate per $10,000
        #         - PTRATIO pupil-teacher ratio by town
        #         - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        #         - LSTAT % lower status of the population
        #         - MEDV Median value of owner-occupied homes in $1000's
```

**Predicting medv (Median value of the houses in 1000's USD)**

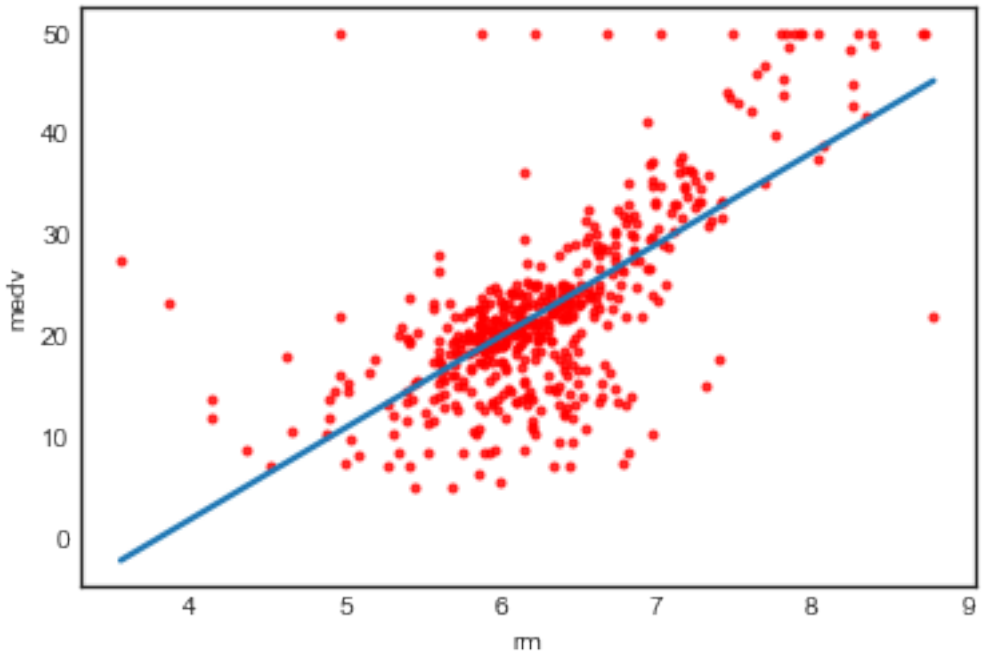
**# simple linear regression with medv and rm**

```
In [5]: # Plot for visualization
        from mpl_toolkits.mplot3d import axes3d
        import seaborn as sns

        sns.regplot(housing.rm, housing.medv, order=1, ci=None, scatter_kws={'color':'r', 's':
```

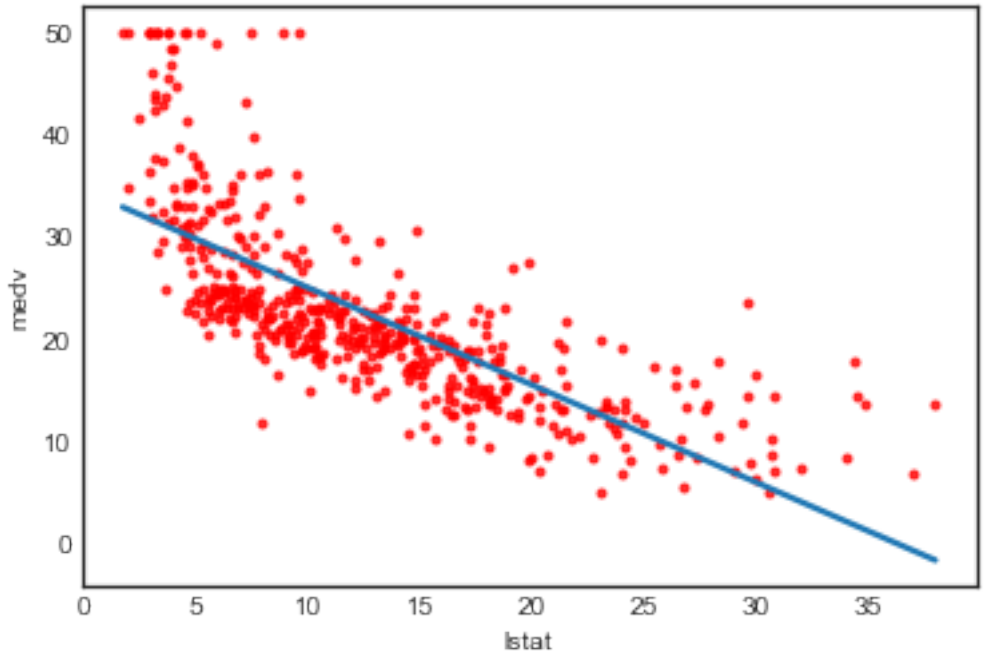
```
C:\Users\Preload\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the
  FutureWarning
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x13ee78d5518>




In [6]: sns.regplot(housing.lstat, housing.medv, order=1, ci=None, scatter_kws={'color':'r', '

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x13ee798a8d0>



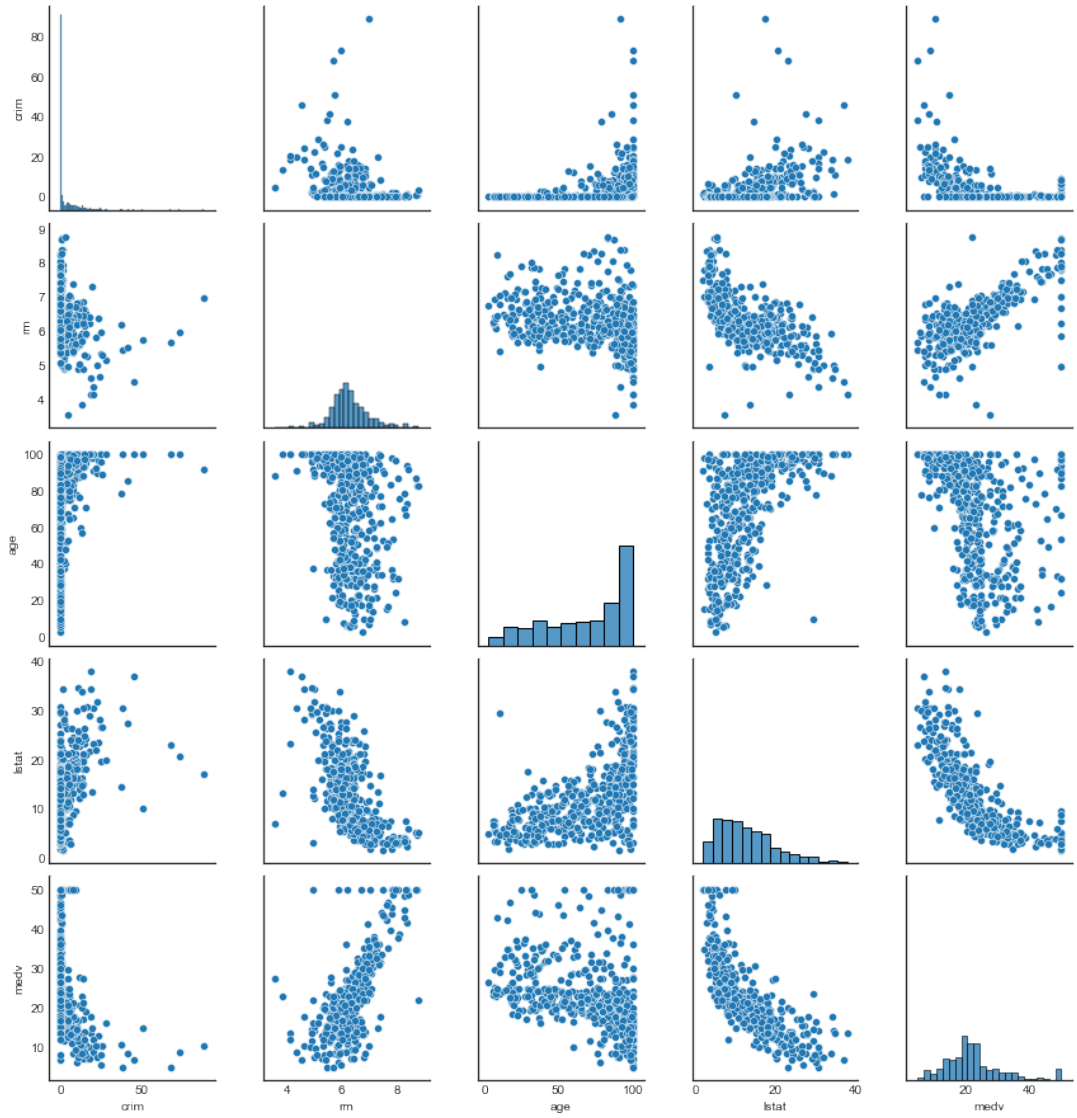
In [7]:
```
# Check correlation
housing.corr()
```

Out[7]:

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

| | dis | rad | tax | ptratio | black | lstat | medv |
|---|---|---|---|---|---|---|---|
| crim | -0.379670 | 0.625505 | 0.582764 | 0.289946 | -0.385064 | 0.455621 | -0.388305 |
| zn | 0.664408 | -0.311948 | -0.314563 | -0.391679 | 0.175520 | -0.412995 | 0.360445 |
| indus | -0.708027 | 0.595129 | 0.720760 | 0.383248 | -0.356977 | 0.603800 | -0.483725 |
| chas | -0.099176 | -0.007368 | -0.035587 | -0.121515 | 0.048788 | -0.053929 | 0.175260 |
| nox | -0.769230 | 0.611441 | 0.668023 | 0.188933 | -0.380051 | 0.590879 | -0.427321 |
| rm | 0.205246 | -0.209847 | -0.292048 | -0.355501 | 0.128069 | -0.613808 | 0.695360 |
| age | -0.747881 | 0.456022 | 0.506456 | 0.261515 | -0.273534 | 0.602339 | -0.376955 |
| dis | 1.000000 | -0.494588 | -0.534432 | -0.232471 | 0.291512 | -0.496996 | 0.249929 |
| rad | -0.494588 | 1.000000 | 0.910228 | 0.464741 | -0.444413 | 0.488676 | -0.381626 |
| tax | -0.534432 | 0.910228 | 1.000000 | 0.460853 | -0.441808 | 0.543993 | -0.468536 |
| ptratio | -0.232471 | 0.464741 | 0.460853 | 1.000000 | -0.177383 | 0.374044 | -0.507787 |
| black | 0.291512 | -0.444413 | -0.441808 | -0.177383 | 1.000000 | -0.366087 | 0.333461 |
| lstat | -0.496996 | 0.488676 | 0.543993 | 0.374044 | -0.366087 | 1.000000 | -0.737663 |
| medv | 0.249929 | -0.381626 | -0.468536 | -0.507787 | 0.333461 | -0.737663 | 1.000000 |

In [8]:
```
# Check few scatter plots
sns.pairplot(housing[['crim','rm','age','lstat','medv']]);
```

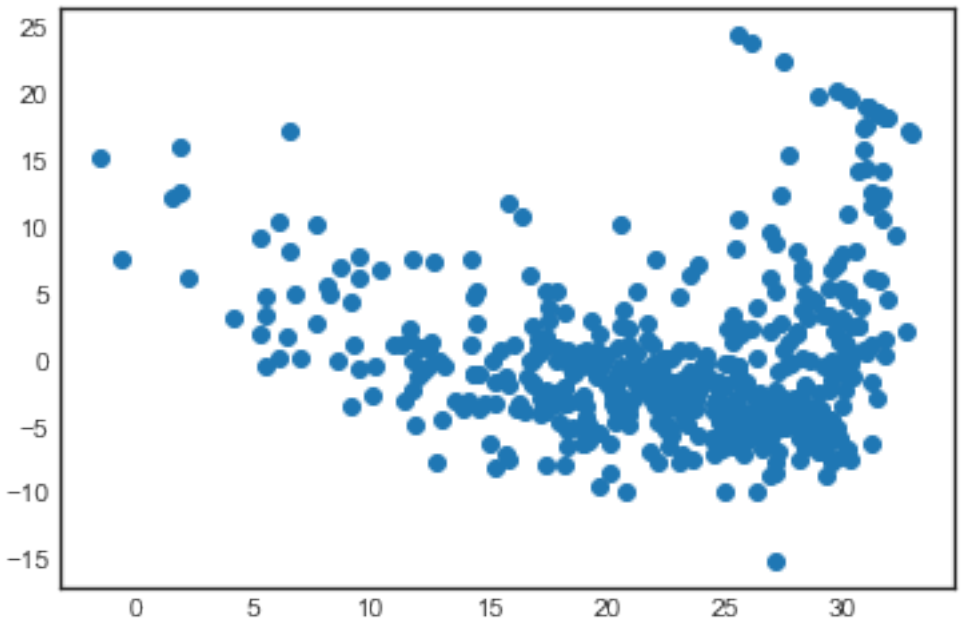



```
In [9]: # simple linear regression with medv and rm
        # Fidning B0 and B1
        est = smf.ols('medv ~ lstat', housing).fit()
        est.summary().tables[1]
```

```
Out[9]: <class 'statsmodels.iolib.table.SimpleTable'>
```

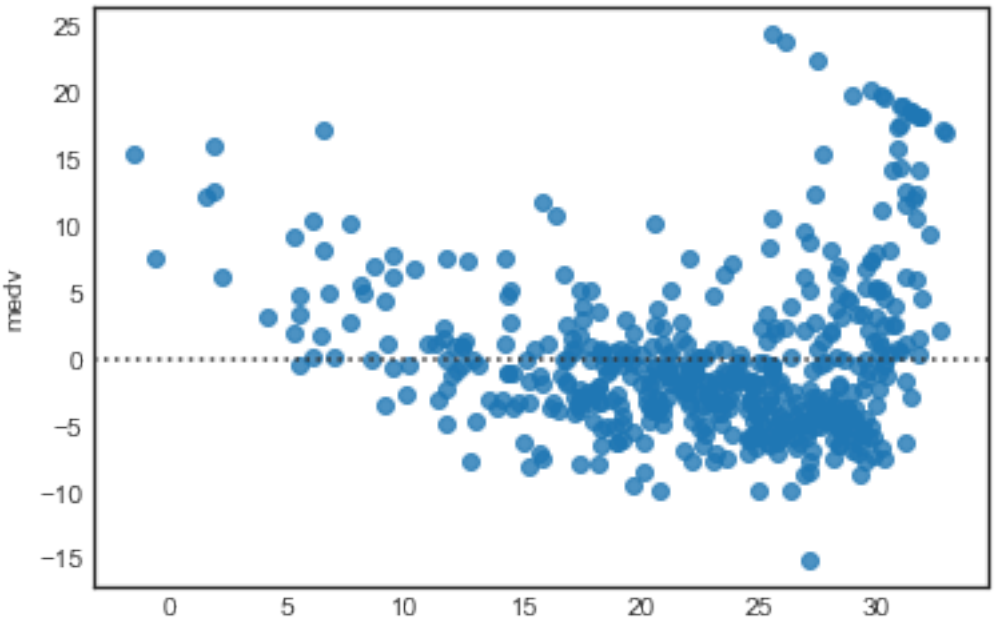
```
In [26]: # Diagnostic plot
         # fittedvalues vs. residuals
         est.fittedvalues
         est.resid
         plt.scatter(est.fittedvalues,est.resid)
```

```
Out[26]: <matplotlib.collections.PathCollection at 0x13ee7bf3be0>
```



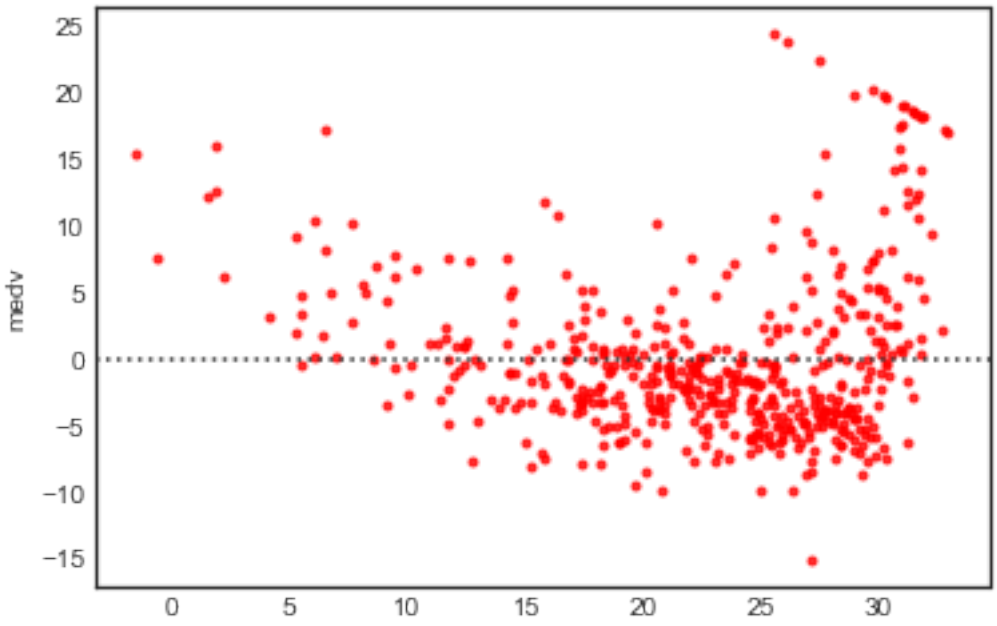

In [27]:
```
# # fittedvalues vs. residuals using Seaborn
from mpl_toolkits.mplot3d import axes3d
import seaborn as sns
sns.residplot(est.fittedvalues, housing.medv)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x13ee7bfed30>



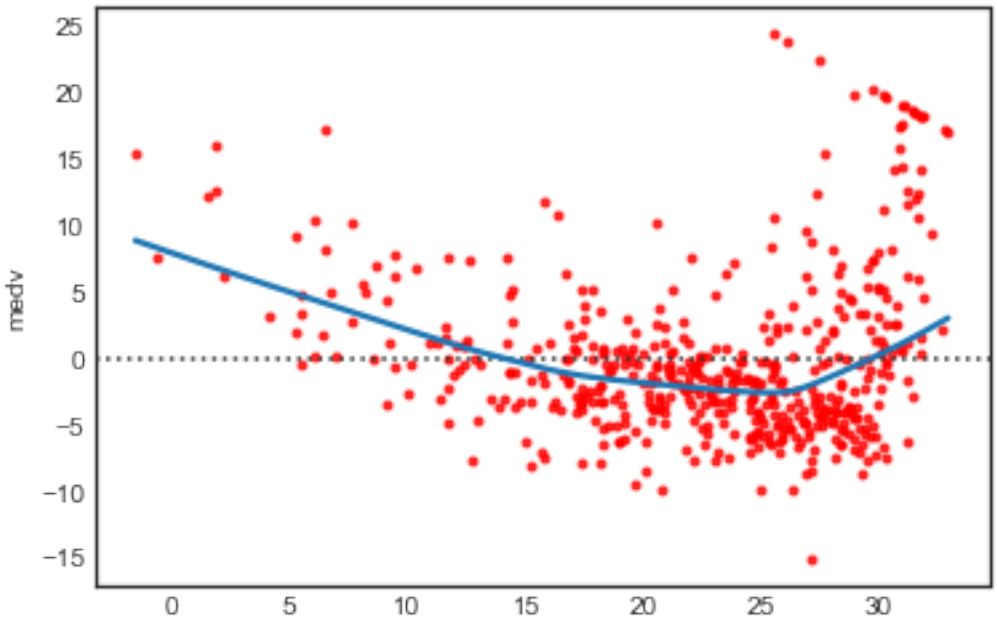
In [28]:
```
sns.residplot(est.fittedvalues, housing.medv, scatter_kws={'color':'r', 's':9})
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13ee7be0f98>
```




In [29]:
```
# Fit a lowess smoother to the residual scatterplot
sns.residplot(est.fittedvalues, housing.medv, lowess=True, scatter_kws={'color':'r',
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13ee7ed2be0>
```



**Predicting medv (Median value of the houses in 1000's USD)**

**Multiple linear regression**

**medv as response**

**predictors: age and lstat**

```
In [30]: est = smf.ols('medv ~ age+lstat', housing).fit()
         est.summary().tables[1]

Out[30]: <class 'statsmodels.iolib.table.SimpleTable'>

In [32]: # With interaction terms
         est = smf.ols('medv ~ age+lstat+age*lstat', housing).fit()
         est.summary().tables[1]

Out[32]: <class 'statsmodels.iolib.table.SimpleTable'>

In [33]: # use of : and * symbols
         # age:lstat=age*lstat
         # age*lstat=age+lstat+age*lstat

         est = smf.ols('medv ~ age*lstat', housing).fit()
         est.summary().tables[1]

Out[33]: <class 'statsmodels.iolib.table.SimpleTable'>
```