Ch03-linreg-lab

February 10, 2026

[52]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
```

[53]:
```
import statsmodels.api as sm
```

[54]:
```
from statsmodels.stats.outliers_influence \
     import variance_inflation_factor as VIF
from statsmodels.stats.anova import anova_lm
```

[55]:
```
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize,
                         poly)
```

[56]:
```
dir()
```

[56]:
```
['A',
 'Boston',
 'In',
 'MS',
 'Out',
 'VIF',
 'X',
 '_',
 '_11',
 '_12',
 '_13',
 '_14',
 '_15',
 '_16',
 '_17',
 '_18',
 '_19',
 '_27',
 '_28',
 '_29',
 '_30',
```

```
'_31',
'_33',
'_34',
'_35',
'_36',
'_37',
'_38',
'_39',
'_40',
'_41',
'_5',
'_6',
'_7',
'_8',
'_9',
'__',
'___',
'__builtin__',
'__builtins__',
'__doc__',
'__loader__',
'__name__',
'__package__',
'__spec__',
'__vsc_ipynb_file__',
'_dh',
'_i',
'_i1',
'_i10',
'_i11',
'_i12',
'_i13',
'_i14',
'_i15',
'_i16',
'_i17',
'_i18',
'_i19',
'_i2',
'_i20',
'_i21',
'_i22',
'_i23',
'_i24',
'_i25',
'_i26',
'_i27',
```

```
'_i28',
'_i29',
'_i3',
'_i30',
'_i31',
'_i32',
'_i33',
'_i34',
'_i35',
'_i36',
'_i37',
'_i38',
'_i39',
'_i4',
'_i40',
'_i41',
'_i42',
'_i43',
'_i44',
'_i45',
'_i46',
'_i47',
'_i48',
'_i49',
'_i5',
'_i50',
'_i51',
'_i52',
'_i53',
'_i54',
'_i55',
'_i56',
'_i6',
'_i7',
'_i8',
'_i9',
'_ih',
'_ii',
'_iii',
'_oh',
'abline',
'anova_lm',
'ax',
'design',
'exit',
'get_ipython',
'load_data',
```

```
 'model',
 'newX',
 'new_df',
 'new_predictions',
 'np',
 'open',
 'pd',
 'poly',
 'quit',
 'results',
 'sm',
 'subplots',
 'summarize',
 'y']
```

[57]:
```
A = np.array([3,5,11])
dir(A)
```

[57]:
```
['T',
 '__abs__',
 '__add__',
 '__and__',
 '__array__',
 '__array_finalize__',
 '__array_function__',
 '__array_interface__',
 '__array_namespace__',
 '__array_priority__',
 '__array_struct__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__buffer__',
 '__class__',
 '__class_getitem__',
 '__complex__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__divmod__',
 '__dlpack__',
 '__dlpack_device__',
 '__doc__',
 '__eq__',
```

```
'__float__',
'__floordiv__',
'__format__',
'__ge__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__ilshift__',
'__imatmul__',
'__imod__',
'__imul__',
'__index__',
'__init__',
'__init_subclass__',
'__int__',
'__invert__',
'__ior__',
'__ipow__',
'__irshift__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lshift__',
'__lt__',
'__matmul__',
'__mod__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
```

```
'__rfloordiv__',
'__rlshift__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__rpow__',
'__rrshift__',
'__rshift__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__xor__',
'all',
'any',
'argmax',
'argmin',
'argpartition',
'argsort',
'astype',
'base',
'byteswap',
'choose',
'clip',
'compress',
'conj',
'conjugate',
'copy',
'ctypes',
'cumprod',
'cumsum',
'data',
'device',
'diagonal',
'dot',
'dtype',
'dump',
'dumps',
'fill',
```

```
 'flags',
 'flat',
 'flatten',
 'getfield',
 'imag',
 'item',
 'itemsize',
 'mT',
 'max',
 'mean',
 'min',
 'nbytes',
 'ndim',
 'nonzero',
 'partition',
 'prod',
 'put',
 'ravel',
 'real',
 'repeat',
 'reshape',
 'resize',
 'round',
 'searchsorted',
 'setfield',
 'setflags',
 'shape',
 'size',
 'sort',
 'squeeze',
 'std',
 'strides',
 'sum',
 'swapaxes',
 'take',
 'to_device',
 'tobytes',
 'tofile',
 'tolist',
 'trace',
 'transpose',
 'var',
 'view']
```

[58]:
```
A.sum()
```

[58]:
```
np.int64(19)
```

[59]:
```
Boston = load_data("Boston")
Boston.columns
```

[59]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'lstat', 'medv'],
      dtype='str')
```

[60]:
```
X = pd.DataFrame({'intercept': np.ones(Boston.shape[0]),
                  'lstat': Boston['lstat']})
X[:4]
```

[60]:

	intercept	lstat
0	1.0	4.98
1	1.0	9.14
2	1.0	4.03
3	1.0	2.94

[61]:
```
y = Boston['medv']
model = sm.OLS(y, X)
results = model.fit()
```

[62]:
```
summarize(results)
```

[62]:

	coef	std err	t	P>|t|
intercept	34.5538	0.563	61.415	0.0
lstat	-0.9500	0.039	-24.528	0.0

[63]:
```
design = MS(['lstat'])
design = design.fit(Boston)
X = design.transform(Boston)
X[:4]
```

[63]:

	intercept	lstat
0	1.0	4.98
1	1.0	9.14
2	1.0	4.03
3	1.0	2.94

[64]:
```
design = MS(['lstat'])
X = design.fit_transform(Boston)
X[:4]
```

[64]:

	intercept	lstat
0	1.0	4.98
1	1.0	9.14
2	1.0	4.03
3	1.0	2.94

[65]:
```
results.summary()
```

[65]:

Dep. Variable:	medv	**R-squared:**	0.544
Model:	OLS	**Adj. R-squared:**	0.543
Method:	Least Squares	**F-statistic:**	601.6
Date:	Tue, 10 Feb 2026	**Prob (F-statistic):**	5.08e-88
Time:	11:25:45	**Log-Likelihood:**	-1641.5
No. Observations:	506	**AIC:**	3287.
Df Residuals:	504	**BIC:**	3295.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
intercept	34.5538	0.563	61.415	0.000	33.448	35.659
lstat	-0.9500	0.039	-24.528	0.000	-1.026	-0.874

Omnibus:	137.043	**Durbin-Watson:**	0.892
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	291.373
Skew:	1.453	**Prob(JB):**	5.36e-64
Kurtosis:	5.319	**Cond. No.**	29.7

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[66]:
```
results.params
```

[66]:
```
intercept    34.553841
lstat        -0.950049
dtype: float64
```

[67]:
```
new_df = pd.DataFrame({'lstat':[5, 10, 15]})
newX = design.transform(new_df)
newX
```

[67]:
```
   intercept  lstat
0        1.0      5
1        1.0     10
2        1.0     15
```

[68]:
```
new_predictions = results.get_prediction(newX);
new_predictions.predicted_mean
```

[68]:
```
array([29.80359411, 25.05334734, 20.30310057])
```

[69]:
```
new_predictions.conf_int(alpha=0.05)
```

[69]:
```
array([[29.00741194, 30.59977628],
       [24.47413202, 25.63256267],
       [19.73158815, 20.87461299]])
```

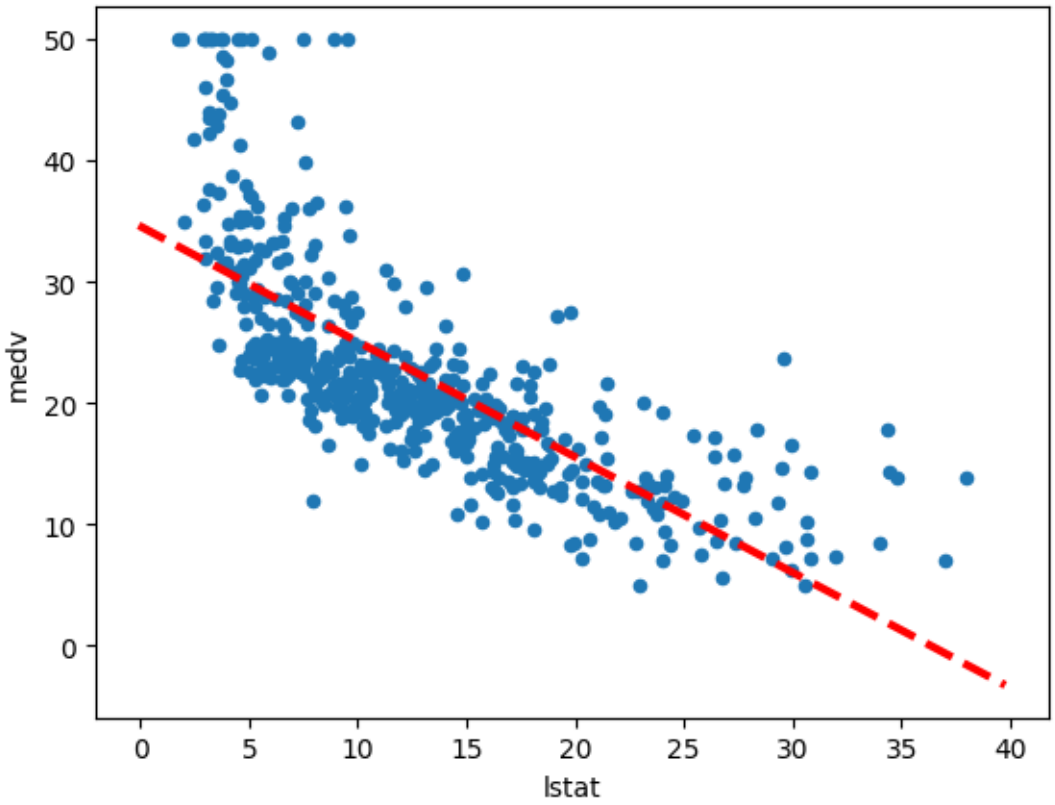
[70]:
```
new_predictions.conf_int(obs=True, alpha=0.05)
```

[70]:
```
array([[17.56567478, 42.04151344],
       [12.82762635, 37.27906833],
       [ 8.0777421 , 32.52845905]])
```

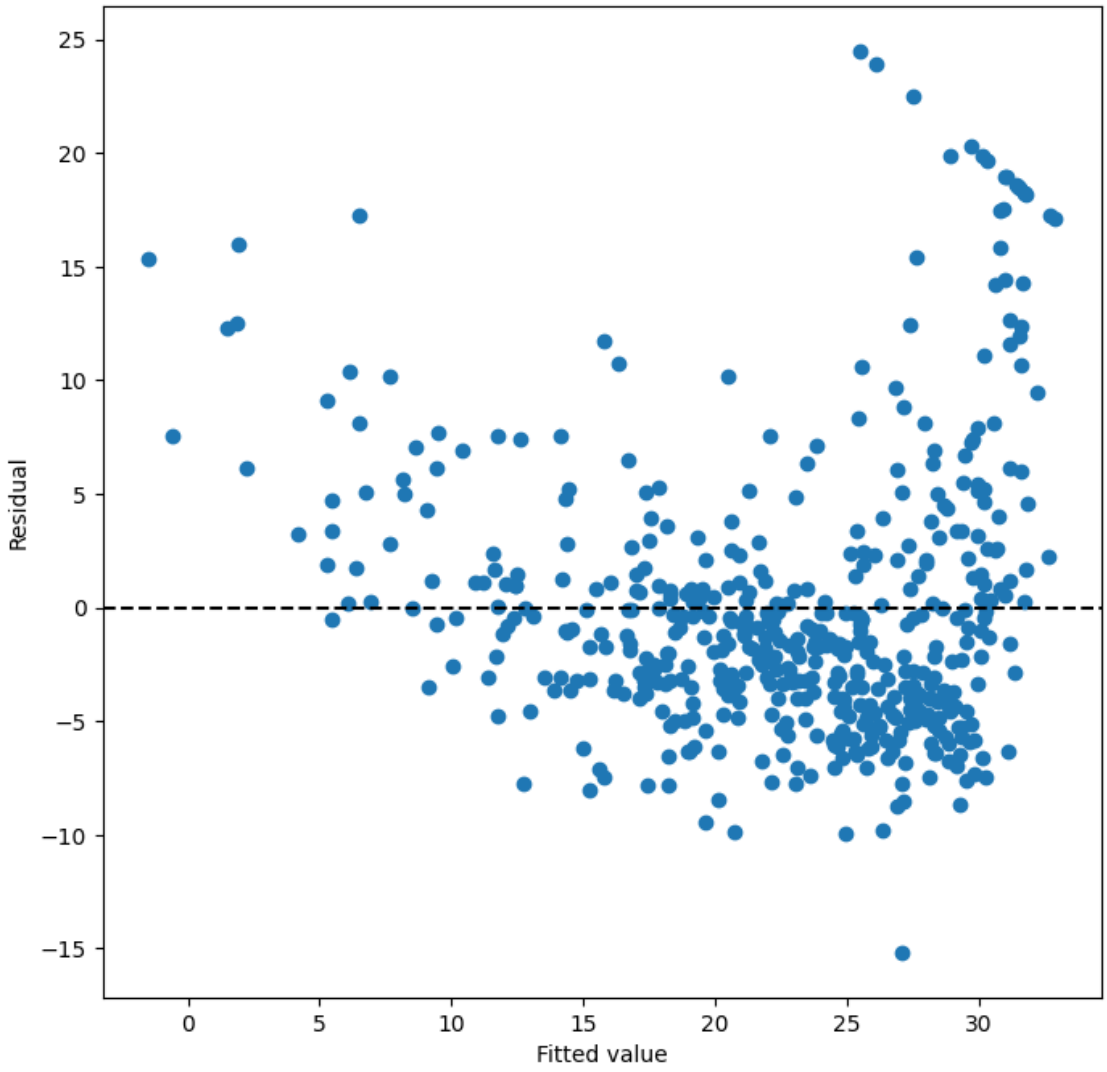
[71]:
```
def abline(ax, b, m):
    "Add a line with slope m and intercept b to ax"
    xlim = ax.get_xlim()
    ylim = [m * xlim[0] + b, m * xlim[1] + b]
    ax.plot(xlim, ylim)
```

[72]:
```
def abline(ax, b, m, *args, **kwargs):
    "Add a line with slope m and intercept b to ax"
    xlim = ax.get_xlim()
    ylim = [m * xlim[0] + b, m * xlim[1] + b]
    ax.plot(xlim, ylim, *args, **kwargs)
```

[73]:
```
ax = Boston.plot.scatter('lstat', 'medv')
abline(
       ax,
       results.params["intercept"],
       results.params["lstat"],
       'r--',
       linewidth=3
)
```

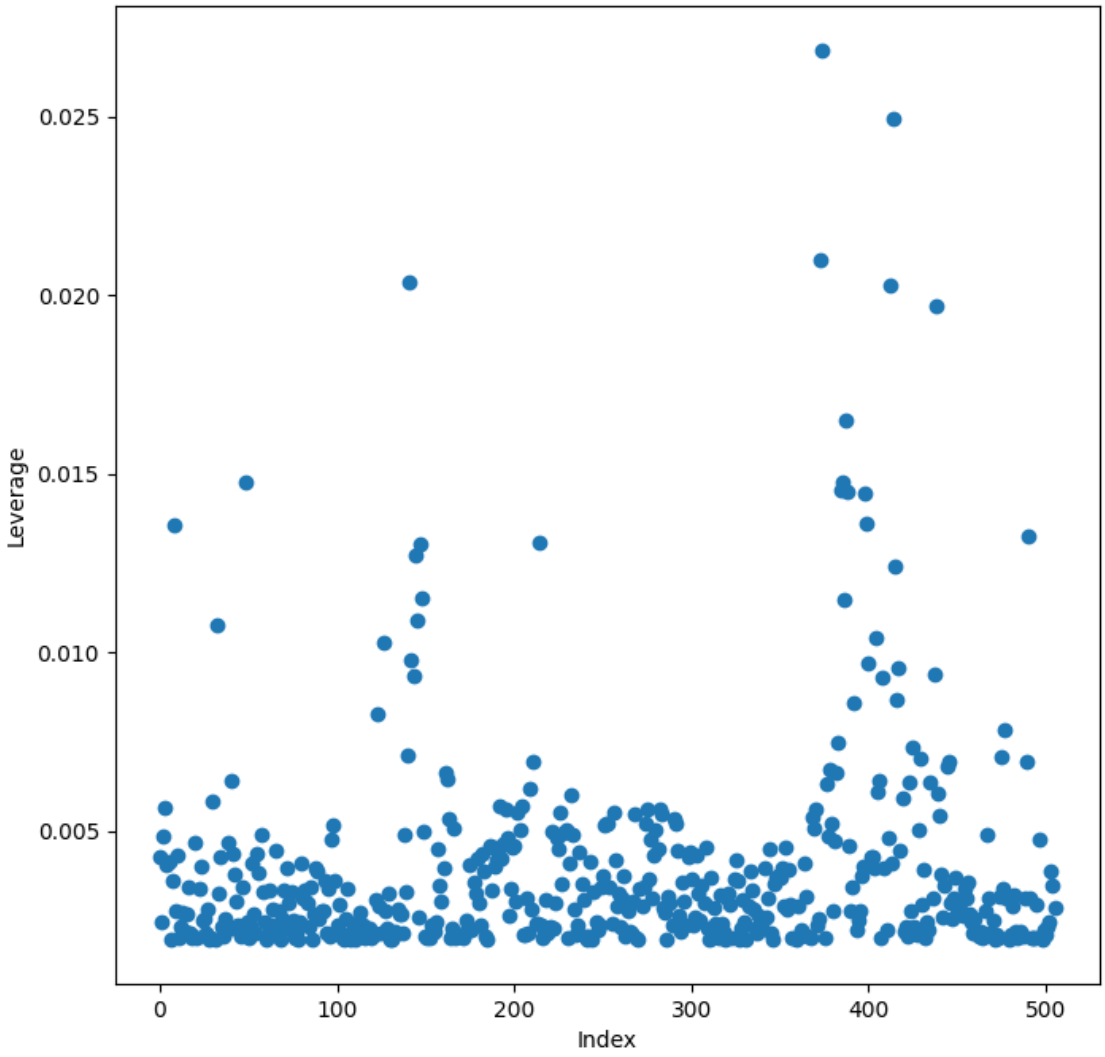


[74]:
```
ax = subplots(figsize=(8,8))[1]
ax.scatter(results.fittedvalues, results.resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



[75]:
```
infl = results.get_influence()
ax = subplots(figsize=(8,8))[1]
ax.scatter(np.arange(X.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```

[75]:
```
np.int64(374)
```



[76]:
```
X = MS(['lstat', 'age']).fit_transform(Boston)
model1 = sm.OLS(y, X)
results1 = model1.fit()
summarize(results1)
```

[76]:

	coef	std err	t	P>|t|
intercept	33.2228	0.731	45.458	0.000
lstat	-1.0321	0.048	-21.416	0.000
age	0.0345	0.012	2.826	0.005

[77]:
```
terms = Boston.columns.drop('medv')
terms
```

[77]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'lstat'],
      dtype='str')
```

[78]:
```
X = MS(terms).fit_transform(Boston)
model = sm.OLS(y, X)
results = model.fit()
summarize(results)
```

[78]:

	coef	std err	t	P>|t|
intercept	41.6173	4.936	8.431	0.000
crim	-0.1214	0.033	-3.678	0.000
zn	0.0470	0.014	3.384	0.001
indus	0.0135	0.062	0.217	0.829
chas	2.8400	0.870	3.264	0.001
nox	-18.7580	3.851	-4.870	0.000
rm	3.6581	0.420	8.705	0.000
age	0.0036	0.013	0.271	0.787
dis	-1.4908	0.202	-7.394	0.000
rad	0.2894	0.067	4.325	0.000
tax	-0.0127	0.004	-3.337	0.001
ptratio	-0.9375	0.132	-7.091	0.000
lstat	-0.5520	0.051	-10.897	0.000

[79]:
```
minus_age = Boston.columns.drop(['medv', 'age'])
Xma = MS(minus_age).fit_transform(Boston)
model1 = sm.OLS(y, Xma)
summarize(model1.fit())
```

[79]:

	coef	std err	t	P>|t|
intercept	41.5251	4.920	8.441	0.000
crim	-0.1214	0.033	-3.683	0.000
zn	0.0465	0.014	3.379	0.001
indus	0.0135	0.062	0.217	0.829
chas	2.8528	0.868	3.287	0.001
nox	-18.4851	3.714	-4.978	0.000
rm	3.6811	0.411	8.951	0.000
dis	-1.5068	0.193	-7.825	0.000
rad	0.2879	0.067	4.322	0.000
tax	-0.0127	0.004	-3.333	0.001
ptratio	-0.9346	0.132	-7.099	0.000
lstat	-0.5474	0.048	-11.483	0.000

[80]:
```
vals = [VIF(X, i)
        for i in range(1, X.shape[1])]
vif = pd.DataFrame({'vif':vals},
                   index=X.columns[1:])
```

```
vif
```

[80]:
```
              vif
crim     1.767486
zn       2.298459
indus    3.987181
chas     1.071168
nox      4.369093
rm       1.912532
age      3.088232
dis      3.954037
rad      7.445301
tax      9.002158
ptratio  1.797060
lstat    2.870777
```

[81]:
```
vals = []
for i in range(1, X.values.shape[1]):
    vals.append(VIF(X.values, i))
```

[82]:
```
X = MS(['lstat',
        'age',
        ('lstat', 'age')]).fit_transform(Boston)
model2 = sm.OLS(y, X)
summarize(model2.fit())
```

[82]:
```
              coef  std err       t  P>|t|
intercept  36.0885    1.470  24.553  0.000
lstat      -1.3921    0.167  -8.313  0.000
age        -0.0007    0.020  -0.036  0.971
lstat:age   0.0042    0.002   2.244  0.025
```

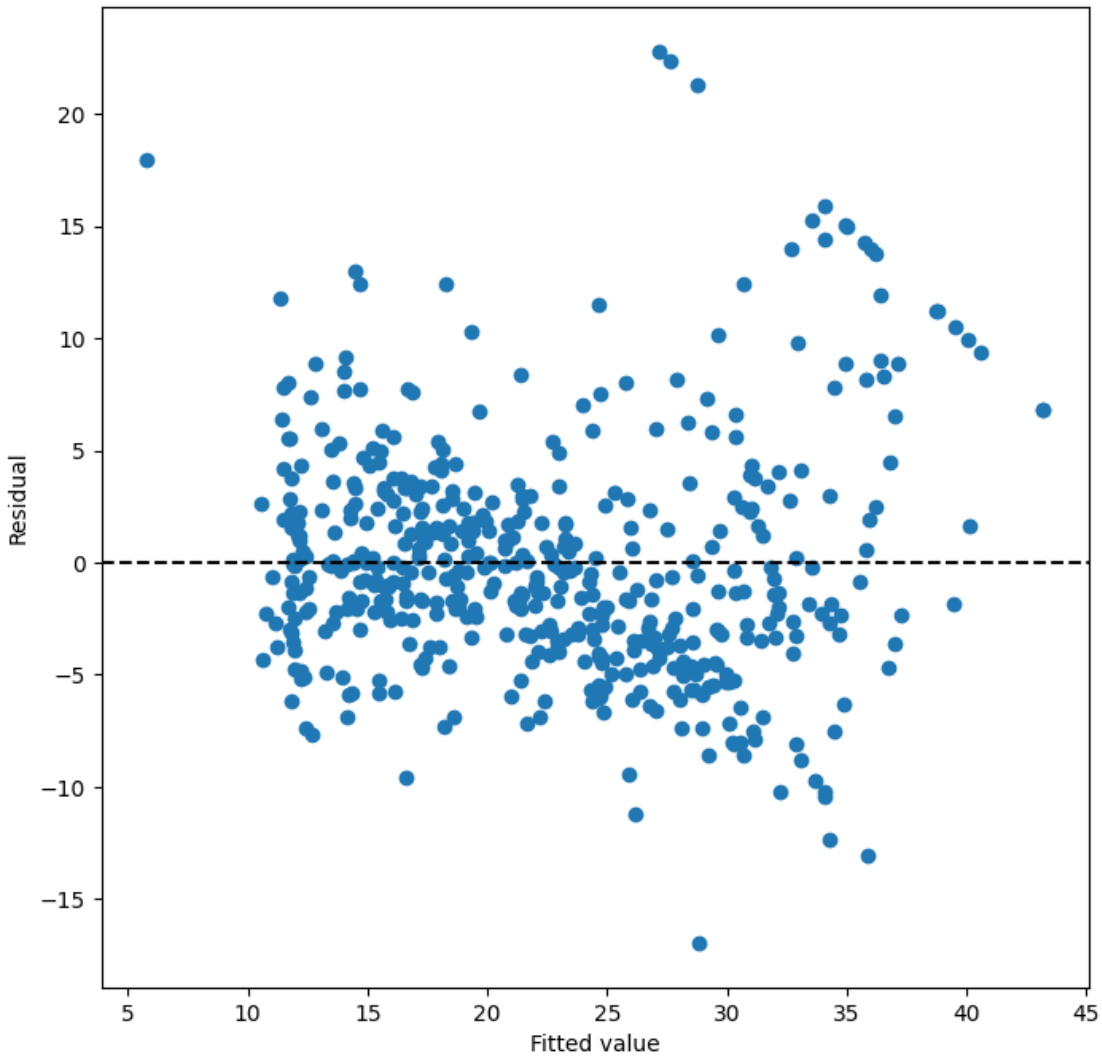
[83]:
```
X = MS([poly('lstat', degree=2), 'age']).fit_transform(Boston)
model3 = sm.OLS(y, X)
results3 = model3.fit()
summarize(results3)
```

[83]:
```
                               coef  std err       t  P>|t|
intercept                   17.7151    0.781  22.681    0.0
poly(lstat, degree=2)[0] -179.2279    6.733 -26.620    0.0
poly(lstat, degree=2)[1]   72.9908    5.482  13.315    0.0
age                         0.0703    0.011   6.471    0.0
```

[84]:
```
anova_lm(results1, results3)
```

[84]:
```
   df_resid           ssr  df_diff      ss_diff           F        Pr(>F)
0     503.0  19168.128609      0.0          NaN         NaN           NaN
1     502.0  14165.613251      1.0  5002.515357  177.278785  7.468491e-35
```

[85]:
```
ax = subplots(figsize=(8,8))[1]
ax.scatter(results3.fittedvalues, results3.resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



[86]:
```
Carseats = load_data('Carseats')
Carseats.columns
```

[86]:
```
Index(['Sales', 'CompPrice', 'Income', 'Advertising', 'Population', 'Price',
       'ShelveLoc', 'Age', 'Education', 'Urban', 'US'],
      dtype='str')
```

[87]:
```
allvars = list(Carseats.columns.drop('Sales'))
y = Carseats['Sales']
final = allvars + [('Income', 'Advertising'),
                   ('Price', 'Age')]
X = MS(final).fit_transform(Carseats)
model = sm.OLS(y, X)
summarize(model.fit())
```

[87]:

	coef	std err	t	P>|t|
intercept	6.5756	1.009	6.519	0.000
CompPrice	0.0929	0.004	22.567	0.000
Income	0.0109	0.003	4.183	0.000
Advertising	0.0702	0.023	3.107	0.002
Population	0.0002	0.000	0.433	0.665
Price	-0.1008	0.007	-13.549	0.000
ShelveLoc[Good]	4.8487	0.153	31.724	0.000
ShelveLoc[Medium]	1.9533	0.126	15.531	0.000
Age	-0.0579	0.016	-3.633	0.000
Education	-0.0209	0.020	-1.063	0.288
Urban[Yes]	0.1402	0.112	1.247	0.213
US[Yes]	-0.1576	0.149	-1.058	0.291
Income:Advertising	0.0008	0.000	2.698	0.007
Price:Age	0.0001	0.000	0.801	0.424