ExercisesAuto

February 21, 2025

1 Applied : Auto dataset - Simple Linear Regression

- Simple Linear Regression uitlizing Auto dataset

1.1 Import notebook functions

[1]:
```
from notebookfuncs import *
```

1.2 Import standard libraries

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import subplots
```

1.3 New imports

[3]:
```
import statsmodels.api as sm
```

1.4 Import statsmodel.objects

[4]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor as VIF
from statsmodels.stats.outliers_influence import summary_table
from statsmodels.stats.anova import anova_lm
```

1.5 Import ISLP objects

[5]:
```
import ISLP
from ISLP import models
from ISLP import load_data
from ISLP.models import ModelSpec as MS, summarize, poly
```

[6]:
```
Auto = load_data("Auto")
Auto.columns
```

[6]: Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'year', 'origin'],
      dtype='object')

[7]:
```
Auto.shape
```

[7]: (392, 8)

[8]:
```
Auto.describe()
```

[8]:
```
              mpg   cylinders  displacement  horsepower       weight  \
count  392.000000  392.000000    392.000000  392.000000   392.000000
mean    23.445918    5.471939    194.411990  104.469388  2977.584184
std      7.805007    1.705783    104.644004   38.491160   849.402560
min      9.000000    3.000000     68.000000   46.000000  1613.000000
25%     17.000000    4.000000    105.000000   75.000000  2225.250000
50%     22.750000    4.000000    151.000000   93.500000  2803.500000
75%     29.000000    8.000000    275.750000  126.000000  3614.750000
max     46.600000    8.000000    455.000000  230.000000  5140.000000

       acceleration        year      origin
count    392.000000  392.000000  392.000000
mean      15.541327   75.979592    1.576531
std        2.758864    3.683737    0.805518
min        8.000000   70.000000    1.000000
25%       13.775000   73.000000    1.000000
50%       15.500000   76.000000    1.000000
75%       17.025000   79.000000    2.000000
max       24.800000   82.000000    3.000000
```

1.6 Convert cylinders and origin columns to categorical types

[9]:
```
Auto["cylinders"] = Auto["cylinders"].astype("category")
Auto["origin"] = Auto["origin"].astype("category")
Auto.describe()
```

[9]:
```
              mpg  displacement  horsepower       weight  acceleration  \
count  392.000000    392.000000  392.000000   392.000000    392.000000
mean    23.445918    194.411990  104.469388  2977.584184     15.541327
std      7.805007    104.644004   38.491160   849.402560      2.758864
min      9.000000     68.000000   46.000000  1613.000000      8.000000
25%     17.000000    105.000000   75.000000  2225.250000     13.775000
50%     22.750000    151.000000   93.500000  2803.500000     15.500000
75%     29.000000    275.750000  126.000000  3614.750000     17.025000
max     46.600000    455.000000  230.000000  5140.000000     24.800000

             year
```

```
count   392.000000
mean     75.979592
std       3.683737
min      70.000000
25%      73.000000
50%      76.000000
75%      79.000000
max      82.000000
```

1.7 8) This question involves the use of Simple Linear Regression on the Auto dataset

1.7.1 (a) Use the sm.OLS() function to perform a simple linear regression with mpg as the response and horsepower as the predictor.

1.7.2 Use the summarize() function to print the results.

1.7.3 Comment on the output. For example:

i. Is there a relationship between the predictor and the response?

ii. How strong is the relationship between the predictor and the response?

iii. Is the relationship between the predictor and the response positive or negative?

iv. What is the predicted mpg associated with a horsepower of 98? What are the associated 95 % confidence and prediction intervals?

[10]:
```python
y = Auto["mpg"]
y.head()
```

[10]:
```
name
chevrolet chevelle malibu    18.0
buick skylark 320            15.0
plymouth satellite           18.0
amc rebel sst                16.0
ford torino                  17.0
Name: mpg, dtype: float64
```

[11]:
```python
design = MS(["horsepower"])
design = design.fit(Auto)
X = design.transform(Auto)
```

[11]:
```
                           intercept  horsepower
name
chevrolet chevelle malibu        1.0         130
buick skylark 320                1.0         165
plymouth satellite               1.0         150
amc rebel sst                    1.0         150
ford torino                      1.0         140
```

```
…                                 …        …
ford mustang gl                   1.0      86
vw pickup                         1.0      52
dodge rampage                     1.0      84
ford ranger                       1.0      79
chevy s-10                        1.0      82

[392 rows x 2 columns]
```

[12]:
```
model = sm.OLS(y, X)
results = model.fit()
summarize(results)
```

[12]:
```
               coef  std err       t  P>|t|
intercept   39.9359    0.717  55.660    0.0
horsepower  -0.1578    0.006 -24.489    0.0
```

- There is evidence of a linear relationship between horespower and the response mpg.

[13]:
```
results.summary()
```

[13]:

Dep. Variable:	mpg	**R-squared:**	0.606
Model:	OLS	**Adj. R-squared:**	0.605
Method:	Least Squares	**F-statistic:**	599.7
Date:	Tue, 24 Sep 2024	**Prob (F-statistic):**	7.03e-81
Time:	08:55:19	**Log-Likelihood:**	-1178.7
No. Observations:	392	**AIC:**	2361.
Df Residuals:	390	**BIC:**	2369.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
intercept	39.9359	0.717	55.660	0.000	38.525	41.347
horsepower	-0.1578	0.006	-24.489	0.000	-0.171	-0.145

Omnibus:	16.432	**Durbin-Watson:**	0.920
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	17.305
Skew:	0.492	**Prob(JB):**	0.000175
Kurtosis:	3.299	**Cond. No.**	322.

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

- The R2 value of 60.6% indicates that the regression of horsepower on mpg explains 60.6% of the variation in the model.
- The relationship between horsepower and mpg is negative, i.e., an increase in hp of 1 unit decreases the mileage by 0.1578 miles. An increase in the car's output in power is offset by a decrease in its economy.

[14]:
```python
design = MS(["horsepower"])
new_df = pd.DataFrame({"horsepower": [98]})
design = design.fit(new_df)
newX = design.transform(new_df)
```

[14]:
```
   intercept  horsepower
0        1.0          98
```

[15]:
```python
new_predictions = results.get_prediction(newX)
mileage = new_predictions.predicted_mean[0]
mileage
```

[15]:
```
24.46707715251243
```

- The predicted mileage for a horsepower of 98 is 24.47 mpg.

[16]:
```python
new_predictions.conf_int(alpha=0.05)
```

[16]:
```
array([[23.97307896, 24.96107534]])
```

- The 95% confidence interval is (23.97, 24.96)

[17]:
```python
new_predictions.conf_int(alpha=0.05, obs=True)
```

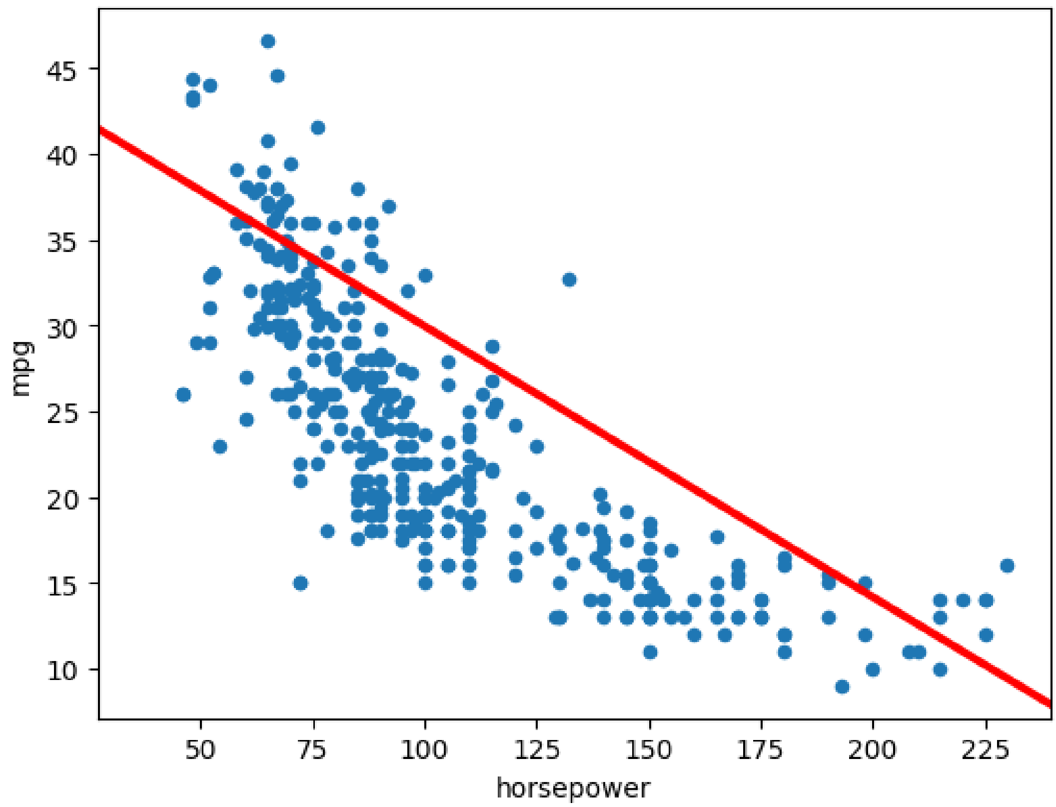
[17]:
```
array([[14.80939607, 34.12475823]])
```

- The 95% prediction interval is (14.82, 34.13)

1.7.4 (b) Plot the response and the predictor in a new set of axes ax.

1.7.5 Use the ax.axline() method or the abline() function defined in the lab to display the least squares regression line.

[18]:
```python
ax = Auto.plot.scatter("horsepower", "mpg")
ax.axline(
    (ax.get_xlim()[0], results.params.iloc[0]),
    slope=results.params.iloc[1],
    color="r",
    linewidth=3,
)
```



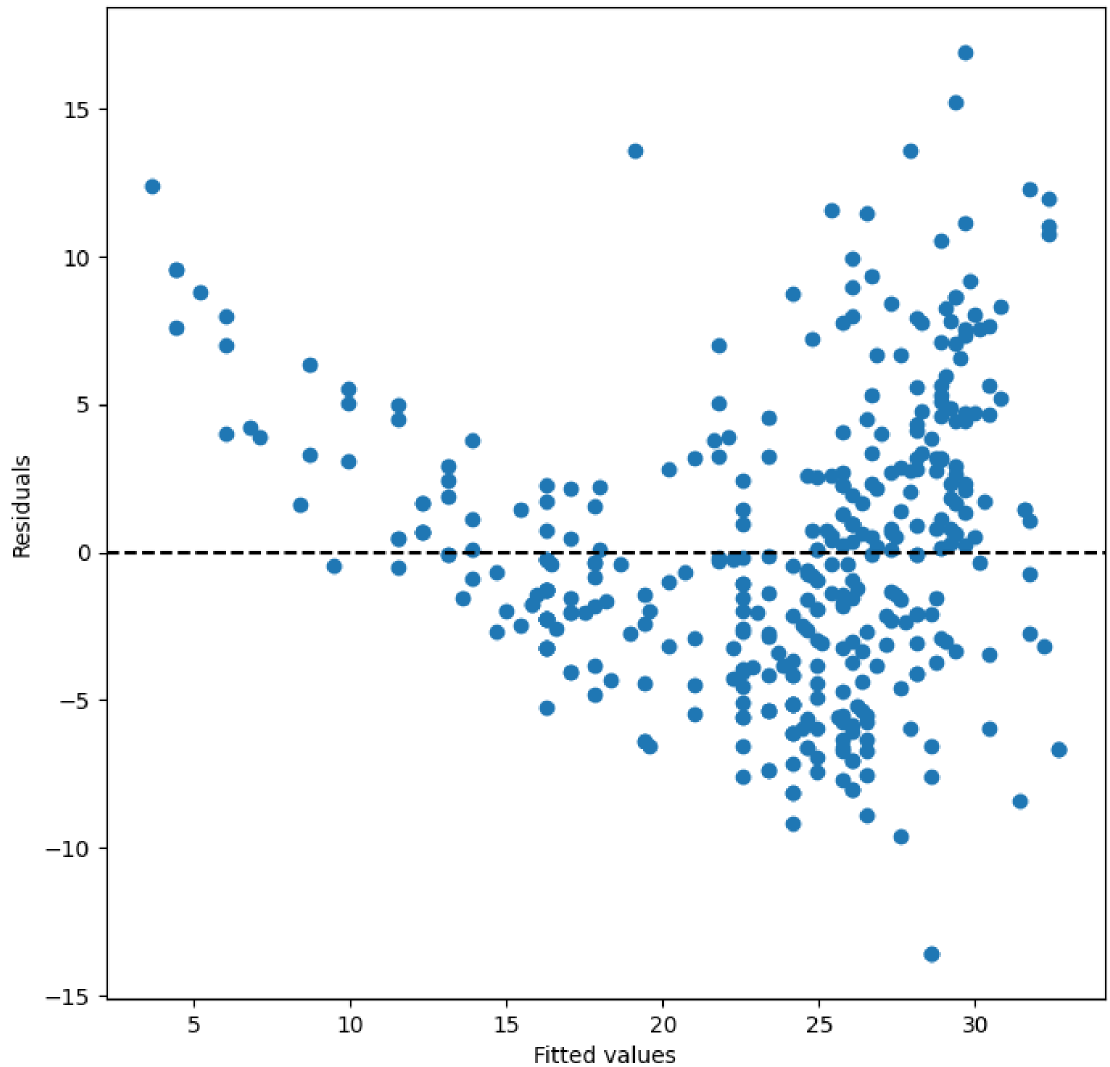
- The least squares regression line is plotted above using ax.axline(). The plot displays some evidence of non-linearity in the relationship between horsepower and mpg.

1.7.6 (c) Produce some of diagnostic plots of the least squares regression fit as described in the lab.

1.7.7 Comment on any problems you see with the fit.

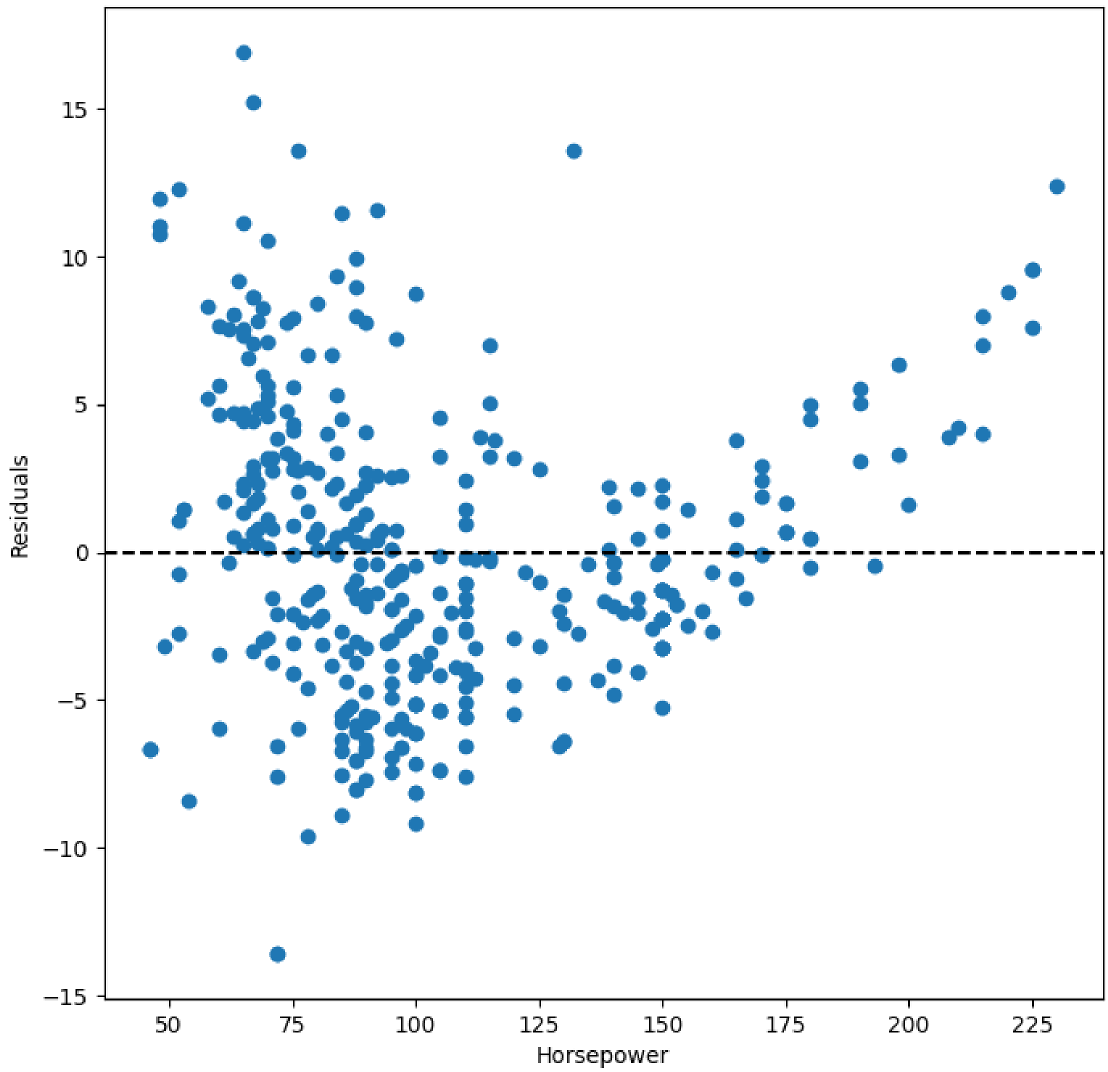
Plot of fitted values versus residuals.

[19]:
```
_, ax = subplots(figsize=(8, 8))
ax.scatter(results.fittedvalues, results.resid)
ax.set_xlabel("Fitted values")
ax.set_ylabel("Residuals")
ax.axhline(0, c="k", ls="--")
```



We can also plot the residuals vs predictor plot where horsepower is the predictor.

[20]:
```
_, ax = subplots(figsize=(8, 8))
ax.scatter(Auto["horsepower"], results.resid)
ax.set_xlabel("Horsepower")
ax.set_ylabel("Residuals")
ax.axhline(0, c="k", ls="--")
```



Conclusions:

- There is evidence of non-linearity in the relationship between residuals and fitted values.
- There is evidence of heteroskedasticity i.e., non-constant variance in the residuals across the fitted values.

[21]:
```
RSS = np.sum((y - results.fittedvalues) ** 2)
RSS
```

[21]: 9385.915871932419

[22]:
```
RSE = np.sqrt(RSS / (Auto.shape[0] - 2))
RSE
```

[22]: 4.90575691954594

1.7.8 OLSResults.scale()

- Gives us a scale factor for the covariance matrix.
- The Default value is ssr/(n-p). Note that the square root of scale is often called the standard error of the regression.
- https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLSResults.scale.html

[23]:
```
np.sqrt(results.scale)
```

[23]: 4.90575691954594

[24]:
```
mpg_mean = Auto["mpg"].mean()
```

[24]: 23.445918367346938

[25]:
```
print("Percentage error in mpg estimation using model above is: ")
np.round(RSE / mpg_mean * 100, decimals=2)
```

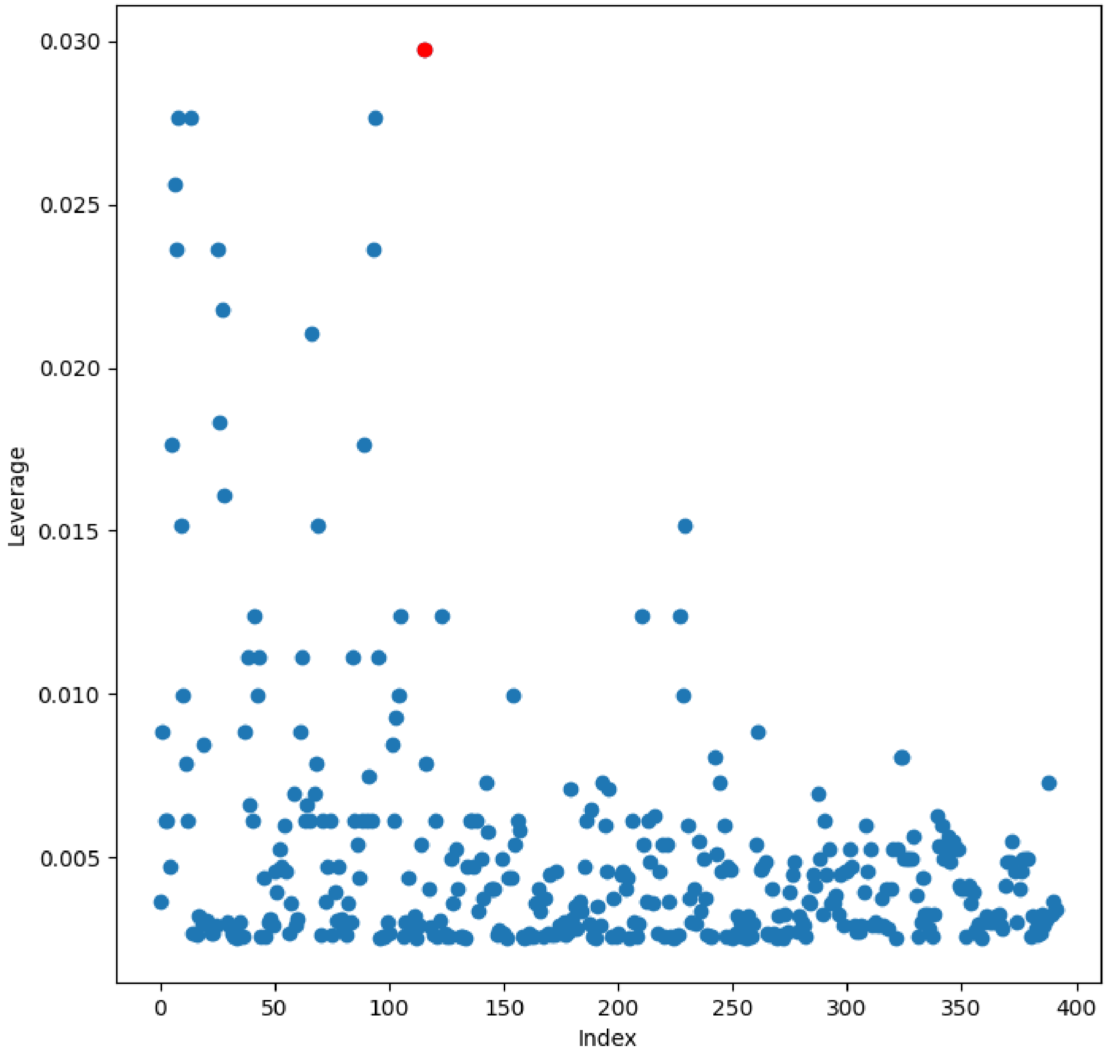
```
Percentage error in mpg estimation using model above is:
```

[25]: 20.92

1.7.9 Leverage statistics

[26]:
```
infl = results.get_influence()
_, ax = subplots(figsize=(8, 8))
ax.scatter(np.arange(X.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel("Index")
ax.set_ylabel("Leverage")
high_leverage = np.argmax(infl.hat_matrix_diag)
max_leverage = np.max(infl.hat_matrix_diag)
print("Max leverage point:")
print(high_leverage, np.round(max_leverage, decimals=2))
ax.plot(high_leverage, max_leverage, "ro")
```

```
Max leverage point:
115 0.03
```



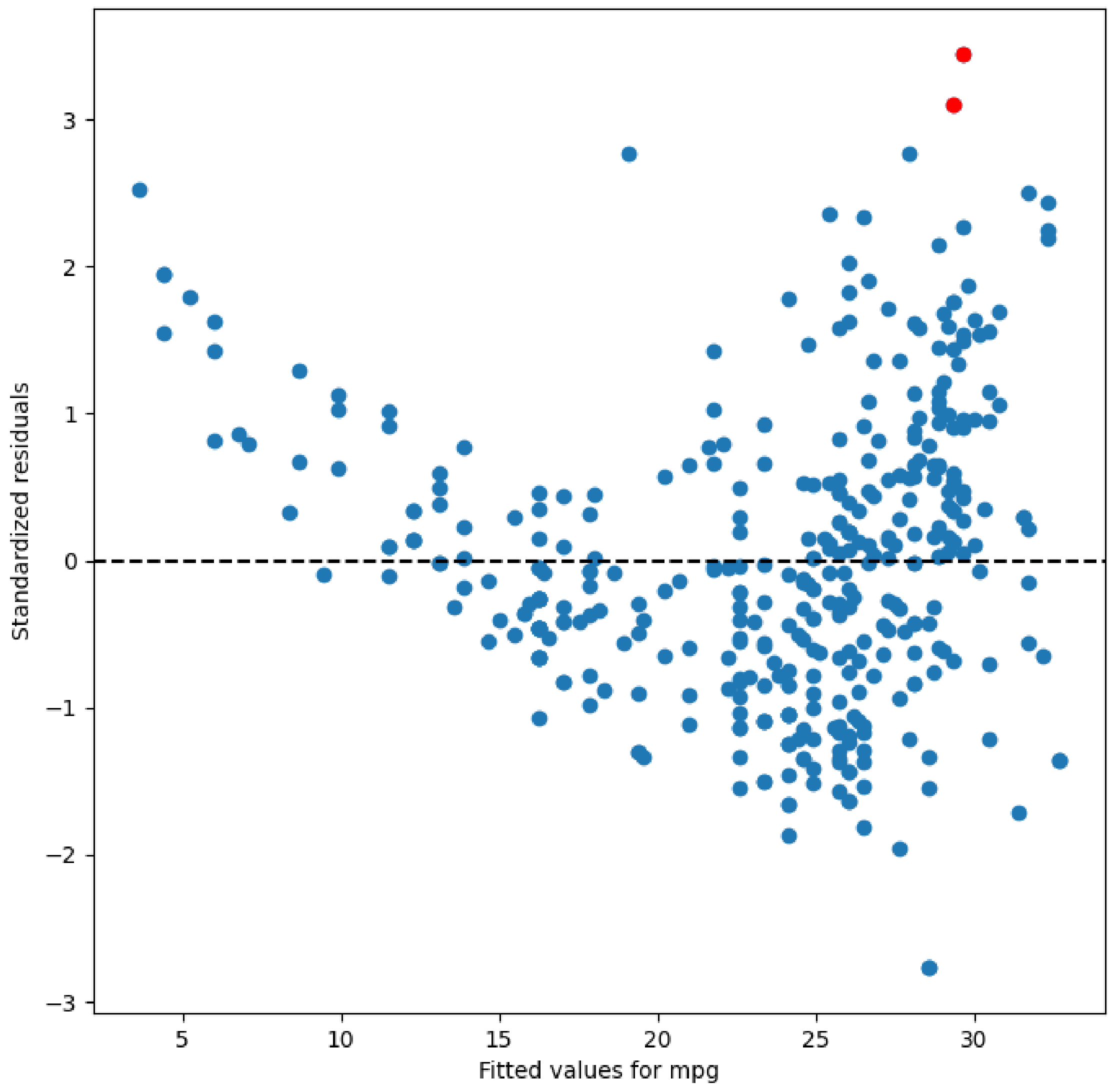
Outlier identification using Standardized Residuals versus Fitted Values plot

[27]:
```
_, ax = subplots(figsize=(8, 8))
ax.scatter(results.fittedvalues, results.resid_pearson)
ax.set_xlabel("Fitted values for mpg")
ax.set_ylabel("Standardized residuals")
ax.axhline(0, c="k", ls="--")
outliers_indexes = np.where(
    (results.resid_pearson > 3.0) | (results.resid_pearson < -3.0)
)[0]
for idx in range(len(outliers_indexes)):
    ax.plot(
        results.fittedvalues.iloc[outliers_indexes[idx]],
        results.resid_pearson[outliers_indexes[idx]],
        "ro",
```

```
    )
print("Outlier rows: ")
print(Auto.iloc[outliers_indexes])
```

```
Outlier rows:
                      mpg  cylinders  displacement  horsepower  weight  \
name
mazda glc            46.6          4          86.0          65    2110
honda civic 1500 gl  44.6          4          91.0          67    1850

                     acceleration  year  origin
name
mazda glc                    17.9    80       3
honda civic 1500 gl          13.8    80       3
```

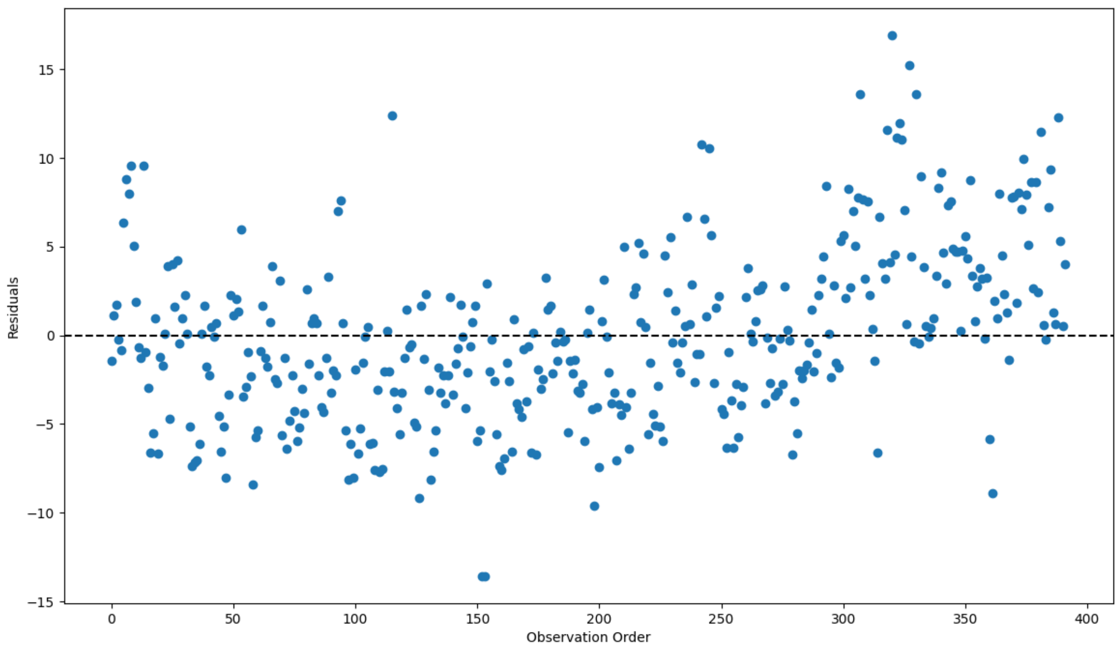


Conclusions: + From the standardized residuals versus fitted values, there are two outliers present in the data. + These points can be investigated further whether to retain them in the dataset.

Note: - We could drop the outliers from the data and regress the model without these points. That is an exercise for you!

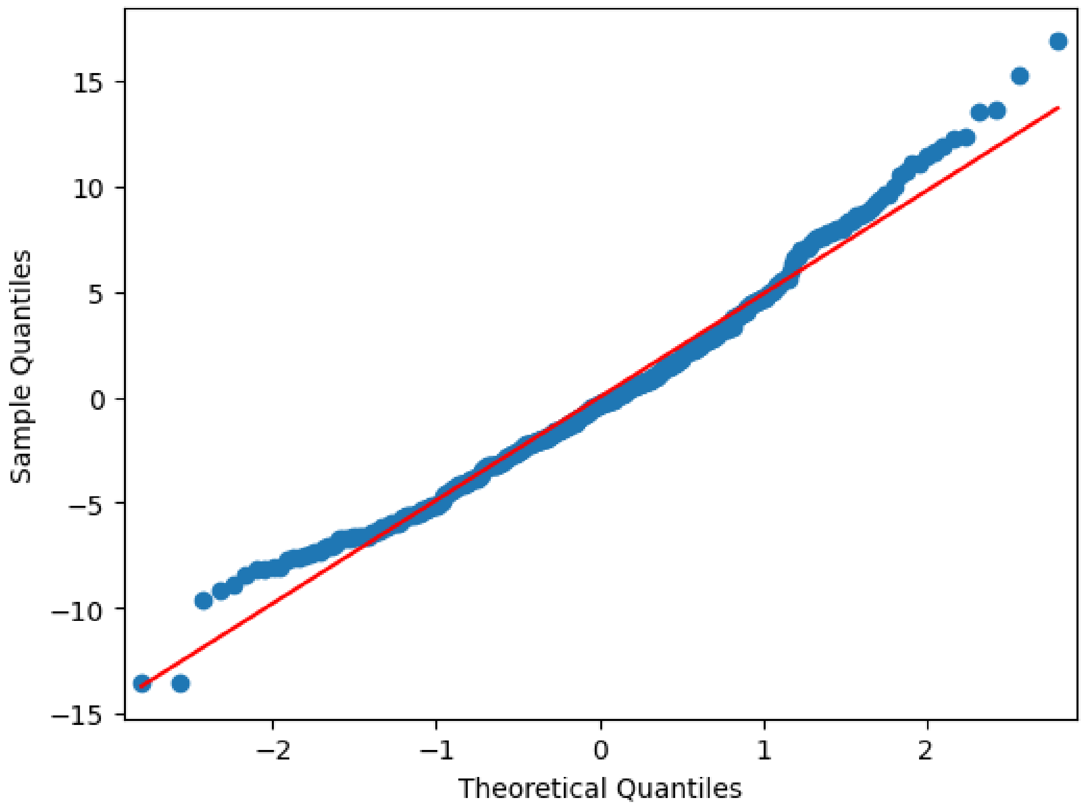
We can also plot residuals versus order.

[28]:
```
_, ax = subplots(figsize=(14, 8))
ax.scatter(np.arange(X.shape[0]), results.resid)
ax.set_xlabel("Observation Order")
ax.set_ylabel("Residuals")
ax.axhline(0, c="k", ls="--")
```

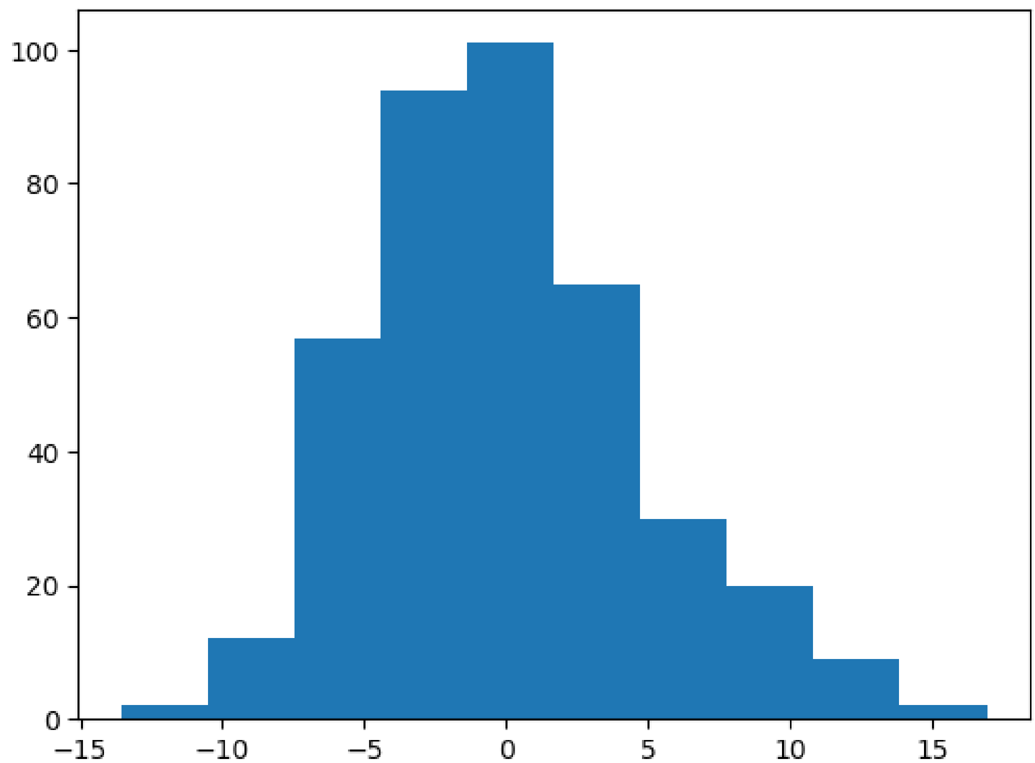


Conclusions: - While there seems to be little evidence of negative or positive correlation over time, there is evidence of underestimation from observations 300 onwards. There also seems to be a time trend in the data from observation 300 or so where the expectation of the model is that mpg will be lower, but the actual values are much higher. This indicates that fuel mileage improved much more than expected in the later models from observation 300 onwards. This indicates that column year should be added to the model.

[29]:
```
sm.qqplot(results.resid, line="s")
```



[30]:
```
# Plot histogram of residuals
plt.hist(results.resid, bins=10)
```



Conclusions: - From the above two plots for qq and histograms for residuals, we can deduce that the residuals are approximately normal.

References: https://github.com/linusjf/LearnR/tree/development/Stats462

[31]:
```
allDone()
```

<IPython.lib.display.Audio object>