In [17]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.regressionplots import influence_plot

# Load the dataset (adjust path if necessary)
data = pd.read_csv('Auto_Data.csv')

# Check the first few rows to ensure the data is loaded correctly
print(data.head())
```

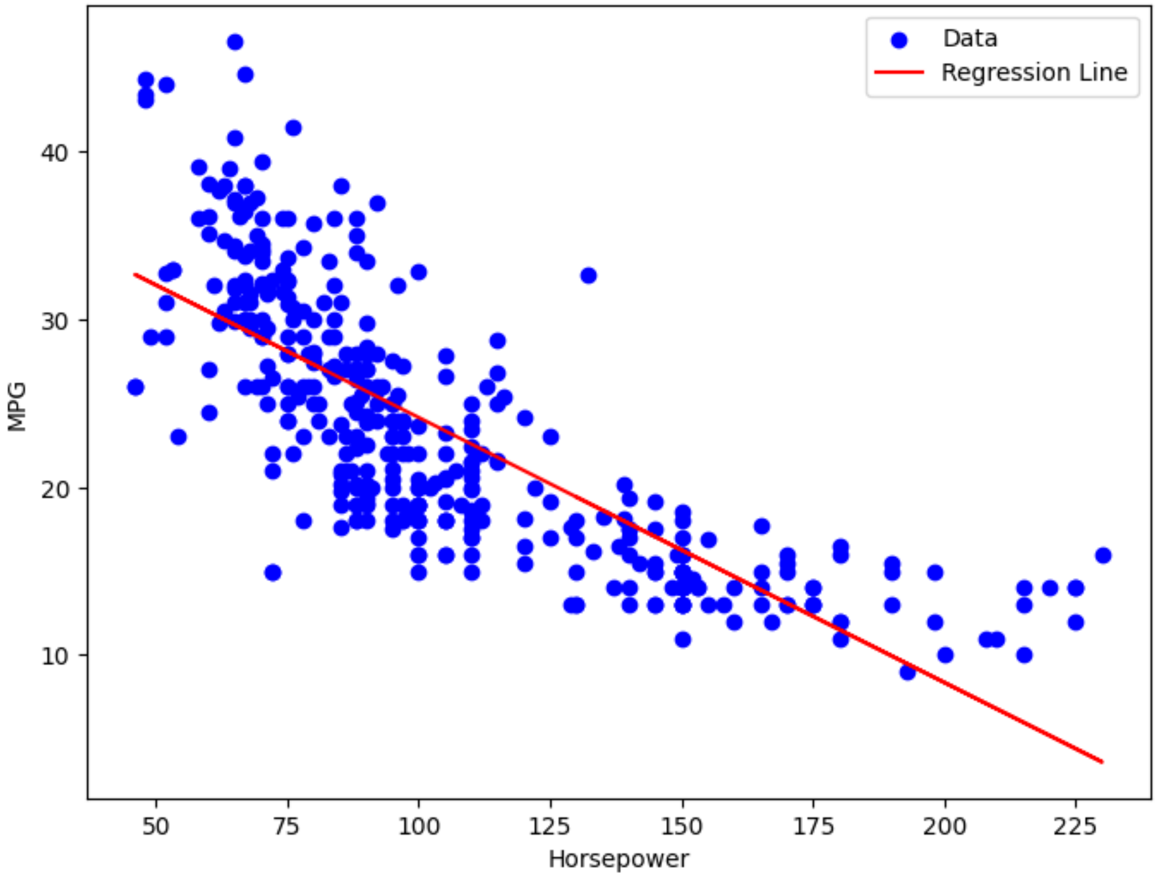
```
    mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0  18.0          8         307.0        130    3504          12.0    70
1  15.0          8         350.0        165    3693          11.5    70
2  18.0          8         318.0        150    3436          11.0    70
3  16.0          8         304.0        150    3433          12.0    70
4  17.0          8         302.0        140    3449          10.5    70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

In [18]:
```
# Assuming 'data' is your DataFrame
# Convert 'horsepower' and 'mpg' to numeric (in case there are non-numeri
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
data['mpg'] = pd.to_numeric(data['mpg'], errors='coerce')

# Drop rows with missing values
data = data.dropna(subset=['horsepower', 'mpg'])

# Define the predictor (horsepower) and response (mpg)
X = data['horsepower']
y = data['mpg']

# Add a constant (intercept) to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary of the regression
print(model.summary())
```

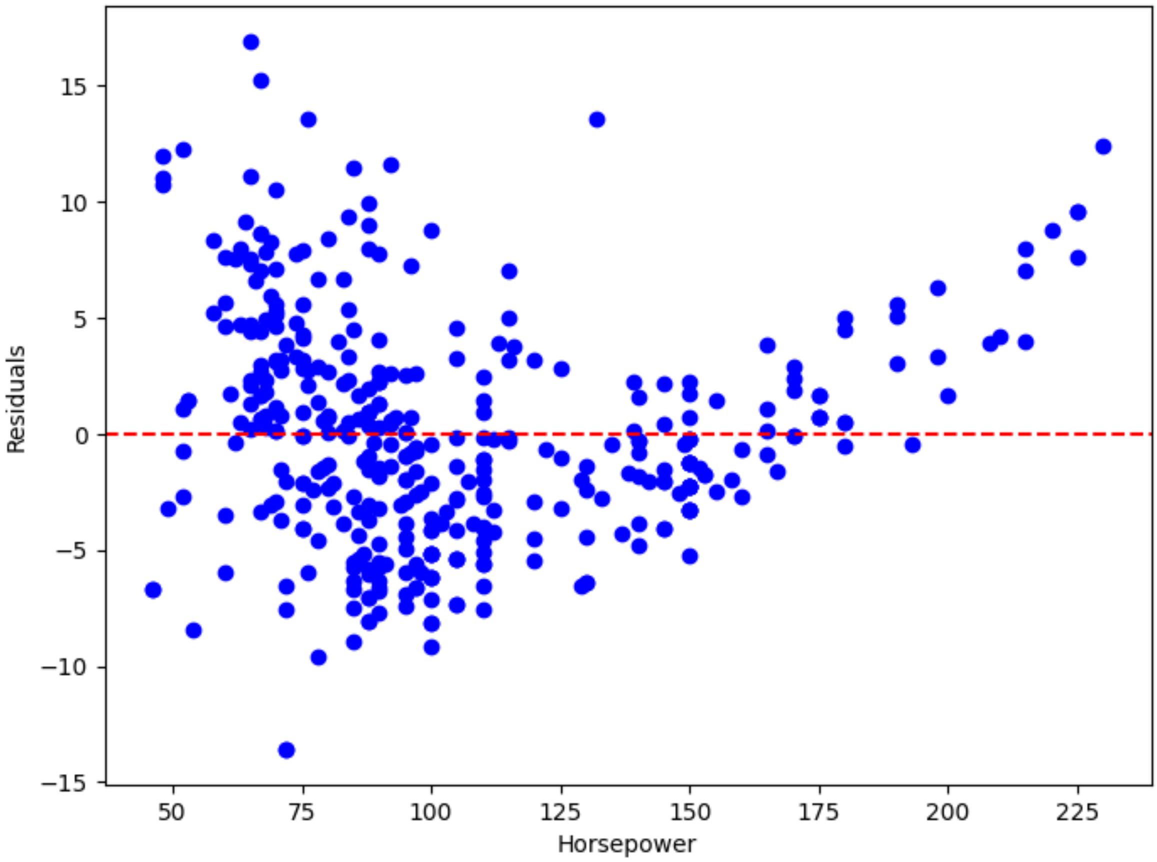
```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                    mpg   R-squared:
0.606
Model:                            OLS   Adj. R-squared:
0.605
Method:                 Least Squares   F-statistic:                     5
99.7
Date:                Wed, 26 Nov 2025   Prob (F-statistic):           7.03
e-81
Time:                        12:45:20   Log-Likelihood:                -11
78.7
No. Observations:                 392   AIC:                             2
361.
Df Residuals:                     390   BIC:                             2
369.
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
====
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
----
const         39.9359      0.717     55.660      0.000      38.525      4
1.347
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):                1
7.305
Skew:                           0.492   Prob(JB):                     0.00
0175
Kurtosis:                       3.299   Cond. No.
322.
==========================================================================
====

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

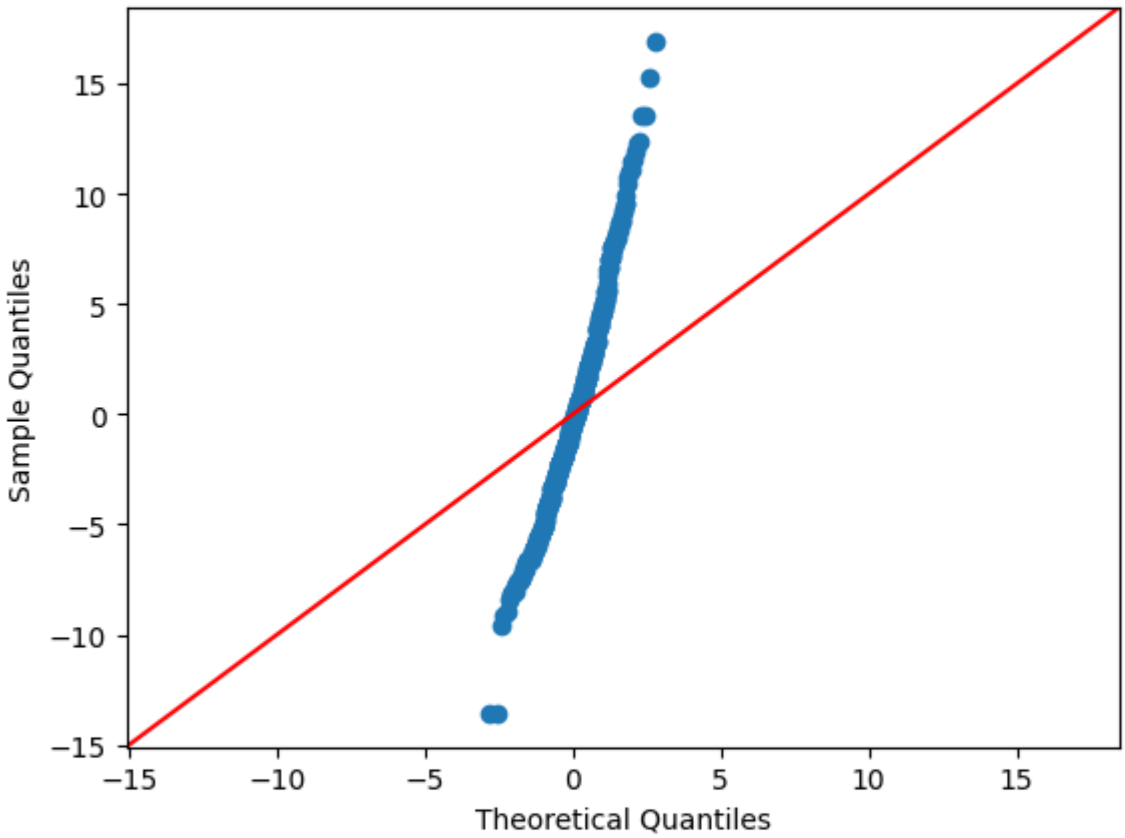
In [19]:

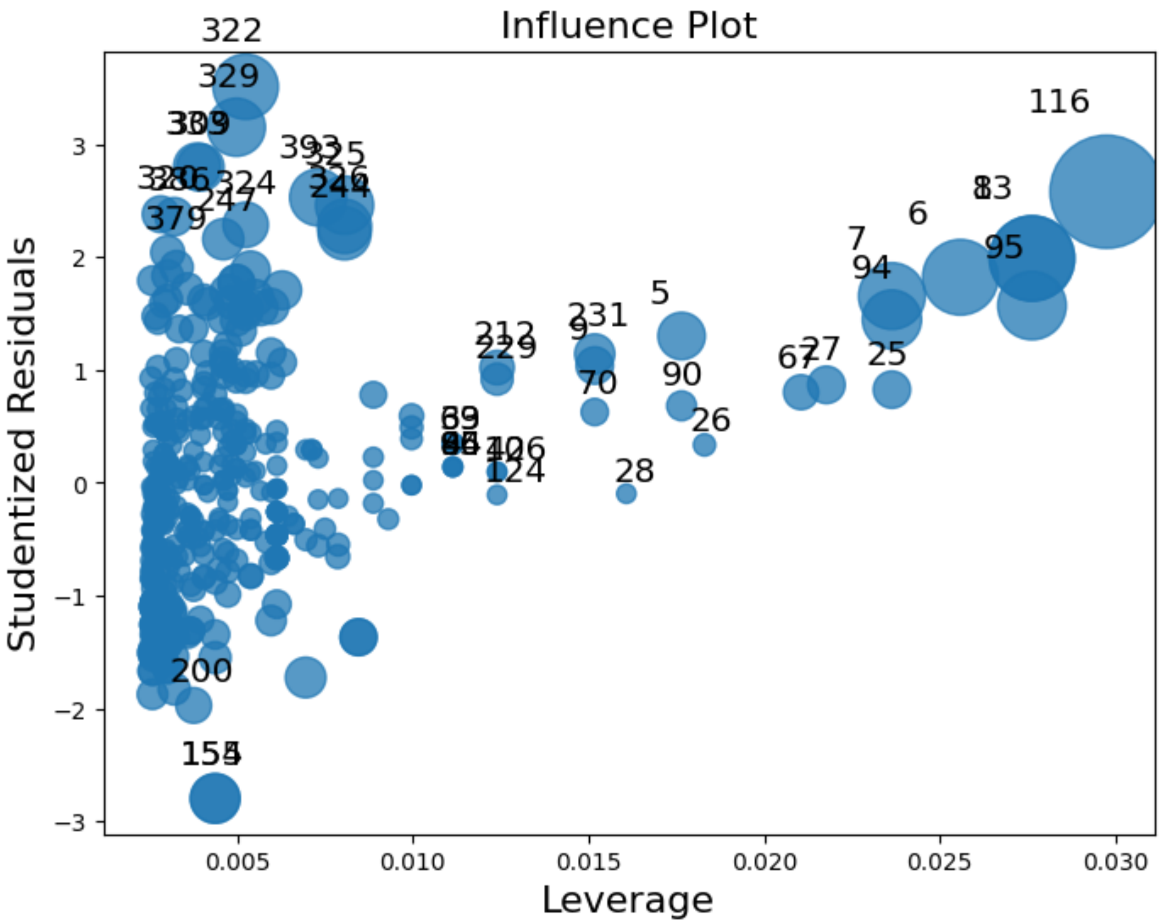
```
# Scatter plot of mpg vs horsepower
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(X['horsepower'], y, label="Data", color='blue')

# Plot the regression line (predicted values)
predicted_values = model.predict(X)
ax.plot(X['horsepower'], predicted_values, color='red', label="Regression

ax.set_xlabel('Horsepower')
ax.set_ylabel('MPG')
ax.legend()
plt.show()
```



In [20]:
```
# Residual plot
residuals = model.resid
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(X['horsepower'], residuals, color='blue')
ax.axhline(0, color='red', linestyle='--')
ax.set_xlabel('Horsepower')
ax.set_ylabel('Residuals')
plt.show()

# QQ Plot for Normality
sm.qqplot(residuals, line='45')
plt.show()

# Leverage Plot (to identify influential points)
fig, ax = plt.subplots(figsize=(8, 6))
influence_plot(model, ax=ax)
plt.show()
```


Residuals
Horsepower
15
10
5
0
−5
−10
−15
50
75
100
125
150
175
200
225


Sample Quantiles
Theoretical Quantiles
15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15

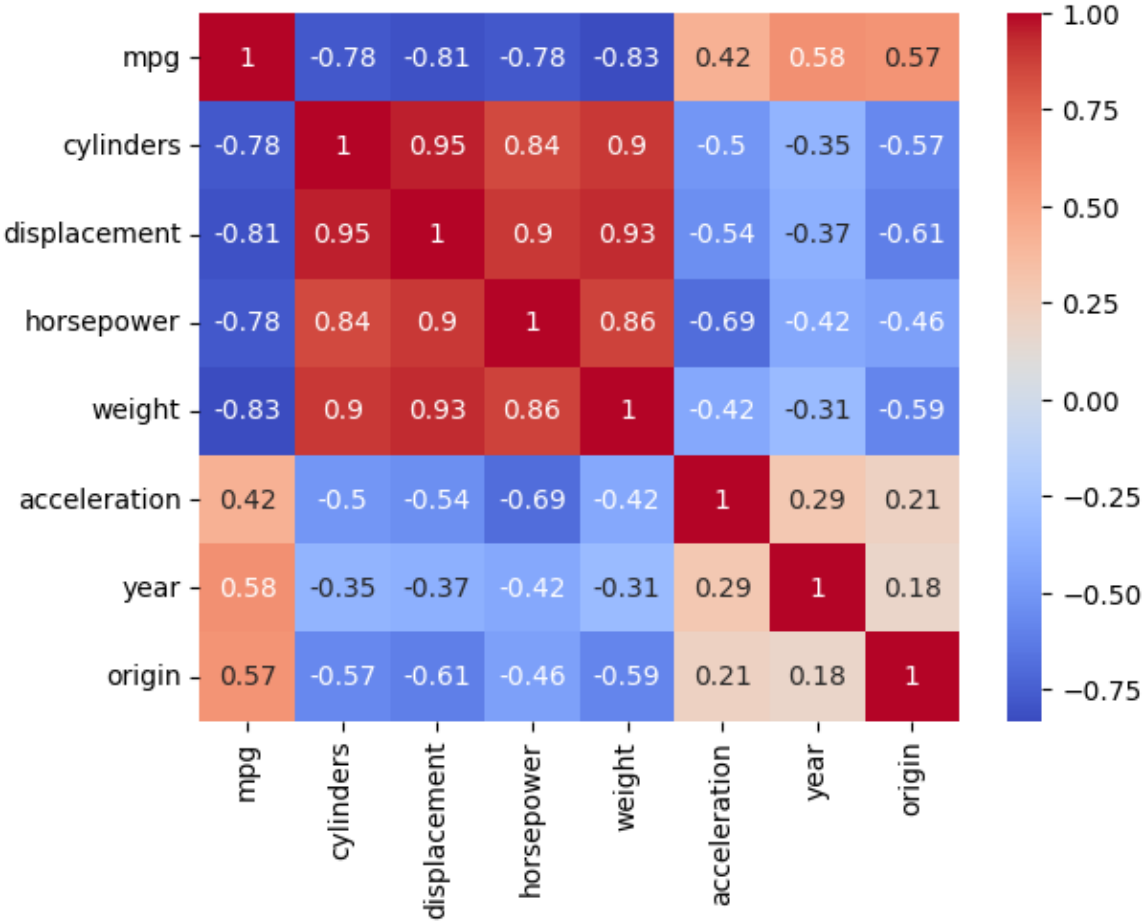


In [21]:
```
# Scatterplot matrix using seaborn
sns.pairplot(data)
plt.show()
```

In [22]:

```
# Select only numeric columns
numeric_data = data.select_dtypes(include=[np.number])

# Calculate the correlation matrix
corr_matrix = numeric_data.corr()

# Print the correlation matrix
print(corr_matrix)

# Plot the correlation matrix using heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

```
                   mpg  cylinders  displacement  horsepower    weight  \
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244
cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527
displacement -0.805127   0.950823      1.000000    0.897257  0.932994
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
weight       -0.832244   0.897527      0.932994    0.864538  1.000000
acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839
year          0.580541  -0.345647     -0.369855   -0.416361 -0.309120
origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005

              acceleration      year    origin
mpg               0.423329  0.580541  0.565209
cylinders        -0.504683 -0.345647 -0.568932
displacement     -0.543800 -0.369855 -0.614535
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.416839 -0.309120 -0.585005
acceleration      1.000000  0.290316  0.212746
year              0.290316  1.000000  0.181528
origin            0.212746  0.181528  1.000000
```



In [23]:
```python
# Drop 'mpg' (response) and 'name' (irrelevant feature) from predictors
X_multi = data.drop(columns=['mpg', 'name'])  # Exclude 'mpg' and 'name'

# Convert predictors to numeric, ensuring no non-numeric values remain
X_multi = X_multi.apply(pd.to_numeric, errors='coerce')

# Convert response variable 'mpg' to numeric
y_multi = pd.to_numeric(data['mpg'], errors='coerce')

# Handle missing values (if any)
X_multi = X_multi.dropna()  # Drop rows with NaN values in predictors
y_multi = y_multi[X_multi.index]  # Ensure 'y' aligns with the rows of 'X
```

```
# Add constant (intercept) to the model
X_multi = sm.add_constant(X_multi)

# Fit the multiple regression model
model_multi = sm.OLS(y_multi, X_multi).fit()

# Print the summary of the regression
print(model_multi.summary())
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                    mpg   R-squared:
0.821
Model:                            OLS   Adj. R-squared:
0.818
Method:                 Least Squares   F-statistic:                     2
52.4
Date:                Wed, 26 Nov 2025   Prob (F-statistic):          2.04
e-139
Time:                        12:45:28   Log-Likelihood:                -10
23.5
No. Observations:                 392   AIC:                             2
063.
Df Residuals:                     384   BIC:                             2
095.
Df Model:                           7
Covariance Type:            nonrobust
==========================================================================
======
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
------
const          -17.2184      4.644     -3.707      0.000     -26.350
-8.087
cylinders       -0.4934      0.323     -1.526      0.128      -1.129
0.142
displacement     0.0199      0.008      2.647      0.008       0.005
0.035
horsepower      -0.0170      0.014     -1.230      0.220      -0.044
0.010
weight          -0.0065      0.001     -9.929      0.000      -0.008
-0.005
acceleration     0.0806      0.099      0.815      0.415      -0.114
0.275
year             0.7508      0.051     14.729      0.000       0.651
0.851
origin           1.4261      0.278      5.127      0.000       0.879
1.973
==========================================================================
====
Omnibus:                       31.906   Durbin-Watson:
1.309
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                5
3.100
Skew:                           0.529   Prob(JB):                     2.95
e-12
Kurtosis:                       4.460   Cond. No.                     8.59
e+04
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The condition number is large, 8.59e+04. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

In [24]:
```
# Example: Add interaction term (horsepower * displacement)
data['horsepower_displacement'] = data['horsepower'] * data['displacement

# Define predictors including the interaction term
X_interaction = data[['horsepower', 'displacement', 'horsepower_displacem
X_interaction = sm.add_constant(X_interaction)

# Fit the model with interaction term
model_interaction = sm.OLS(y_multi, X_interaction).fit()

# Print the summary of the model with interaction term
print(model_interaction.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                    mpg   R-squared:
0.747
Model:                            OLS   Adj. R-squared:
0.745
Method:                 Least Squares   F-statistic:                     3
81.0
Date:                Wed, 26 Nov 2025   Prob (F-statistic):          3.00
e-115
Time:                        12:45:28   Log-Likelihood:                -10
92.1
No. Observations:                 392   AIC:                             2
192.
Df Residuals:                     388   BIC:                             2
208.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
=================
                             coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                     53.0511      1.526     34.765      0.000      5
0.051      56.051
horsepower                -0.2343      0.020    -11.960      0.000
-0.273      -0.196
displacement              -0.0980      0.007    -14.674      0.000
-0.111      -0.085
horsepower_displacement    0.0006   5.19e-05     11.222      0.000
0.000       0.001
==============================================================================
====
Omnibus:                       46.481   Durbin-Watson:
1.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               8
7.417
Skew:                           0.685   Prob(JB):                     1.04
e-19
Kurtosis:                       4.865   Cond. No.                     2.47
e+05
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The condition number is large, 2.47e+05. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

In [25]:

```python
import numpy as np

# Apply log transformation on horsepower and displacement (since 'weight'
data['log_horsepower'] = np.log(data['horsepower'])
data['log_displacement'] = np.log(data['displacement'])

# Define predictors for the transformed model
```

```
X_transformed = data[['log_horsepower', 'log_displacement']]
X_transformed = sm.add_constant(X_transformed)

# Fit the transformed model
model_transformed = sm.OLS(y_multi, X_transformed).fit()

# Print the summary of the transformed model
print(model_transformed.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                    mpg   R-squared:
0.724
Model:                            OLS   Adj. R-squared:
0.723
Method:                 Least Squares   F-statistic:                         5
10.3
Date:                Wed, 26 Nov 2025   Prob (F-statistic):               1.77
e-109
Time:                        12:45:28   Log-Likelihood:                    -11
08.9
No. Observations:                 392   AIC:                                 2
224.
Df Residuals:                     389   BIC:                                 2
236.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
==========
                       coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const              101.0700      2.915     34.667      0.000      95.338
106.802
log_horsepower      -9.0199      1.237     -7.289      0.000     -11.453
-6.587
log_displacement    -7.0675      0.798     -8.860      0.000      -8.636
-5.499
==============================================================================
====
Omnibus:                       30.256   Durbin-Watson:
0.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                   4
2.413
Skew:                           0.569   Prob(JB):                         6.17
e-10
Kurtosis:                       4.140   Cond. No.
103.
==============================================================================
====

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

In [ ]: