linear20

November 24, 2025

1 Applied Linear Regression: Auto Dataset Analysis

This notebook follows the assignment brief to analyze the `Auto` dataset. We will perform: 1. **Simple Linear Regression:** `mpg` as a function of `horsepower`. 2. **Multiple Linear Regression:** `mpg` as a function of all other relevant predictors.

We will load the data, fit the models, interpret the results, and produce diagnostic plots.

1.1 Setup: Import Libraries and Load Data

First, we import all the necessary Python libraries for data manipulation, statistical modeling, and plotting.

[1]:
```
# --- Imports ---

# For data handling
import pandas as pd
import numpy as np

# For plotting
import matplotlib.pyplot as plt
import seaborn as sns

# For statistical models
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

# For diagnostics
import scipy.stats as stats
from statsmodels.graphics.regressionplots import influence_plot

# Set plot style
sns.set_theme(style="whitegrid")
# Show plots inline in Jupyter
%matplotlib inline
```

Now, we load the `Auto.csv` dataset from the provided URL.

Important Data Cleaning Note: The `horsepower` column in this dataset contains `'?'` for missing values. We must: 1. Read the CSV and tell `pandas` to treat `'?'` as a Not-a-Number (`NaN`). 2. Drop any rows that have `NaN` values to ensure our regression models run correctly. 3. We will also drop the `name` column, as it's a unique identifier and not a useful predictor.

[4]:

```
# --- Load Data ---

# FIX: This is the correct, stable URL for the raw CSV file
data_url = "https://www.statlearning.com/s/Auto.csv"

# Read the CSV, marking '?' as missing values
df = pd.read_csv(data_url, na_values='?')

# --- Clean Data ---

# Print original shape
print(f"Original shape: {df.shape}")

# Drop rows with any missing values (especially in 'horsepower')
df_clean = df.dropna()

# Print new shape
print(f"Shape after dropping NA: {df_clean.shape}")

# Drop the 'name' column as it's not a predictor
df_clean = df_clean.drop(columns=['name'])

# Display the first 5 rows and data types to confirm
print("\n--- Data Head ---")
print(df_clean.head())
print("\n--- Data Types ---")
print(df_clean.info())
```

```
Original shape: (397, 9)
Shape after dropping NA: (392, 9)

--- Data Head ---
    mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
0  18.0          8         307.0       130.0    3504          12.0    70
1  15.0          8         350.0       165.0    3693          11.5    70
2  18.0          8         318.0       150.0    3436          11.0    70
3  16.0          8         304.0       150.0    3433          12.0    70
4  17.0          8         302.0       140.0    3449          10.5    70

   origin
0       1
1       1
2       1
3       1
```

```
4       1

--- Data Types ---
<class 'pandas.core.frame.DataFrame'>
Index: 392 entries, 0 to 396
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   year          392 non-null    int64
 7   origin        392 non-null    int64
dtypes: float64(4), int64(4)
memory usage: 27.6 KB
None
```

1.2 Task A: Simple Linear Regression (mpg ~ horsepower)

This task involves using simple linear regression with `mpg` as the response and `horsepower` as the predictor.

1.2.1 Task A(a): Fit and Summarize the Model

We use `statsmodels.formula.api.ols()` (Ordinary Least Squares) to fit the model. The formula `'mpg ~ horsepower'` automatically includes an intercept.

We will then: 1. Print the model summary. 2. Calculate the predicted `mpg` for `horsepower = 98`. 3. Get the 95% confidence and prediction intervals for that value.

[6]:
```python
# --- Task A(a): Fit and Summarize ---

# (i, ii, iii) Fit the model and print the summary
# We use df_clean which has no missing values
slr_model = smf.ols('mpg ~ horsepower', data=df_clean).fit()
print(slr_model.summary())

# (iv) Get prediction for horsepower = 98
# Create a new DataFrame for the value we want to predict
new_hp = pd.DataFrame({'horsepower': [98]})

# Get prediction
pred = slr_model.predict(new_hp)
print(f"\nPredicted mpg for horsepower=98: {pred.iloc[0]:.4f}")

# Get 95% confidence and prediction intervals
```

```python
pred_summary = slr_model.get_prediction(new_hp).summary_frame(alpha=0.05)

print("\n--- 95% Intervals for horsepower=98 ---")
print(pred_summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Mon, 24 Nov 2025   Prob (F-statistic):           7.03e-81
Time:                        13:00:38   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.

Predicted mpg for horsepower=98: 24.4671

--- 95% Intervals for horsepower=98 ---
        mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  24.467077  0.251262      23.973079      24.961075     14.809396

   obs_ci_upper
0     34.124758
```

Your Interpretation for A(a): *(This is where you'll write your comments based on the output above)*

- **i. Is there a relationship?**
  - *Hint: Look at the* `P>|t|` *(p-value) for* `horsepower`*. Is it very small (e.g., < 0.05)?*
- **ii. How strong is the relationship?**
  - *Hint: Look at the* `R-squared` *value. This represents the percentage of variance in* `mpg`

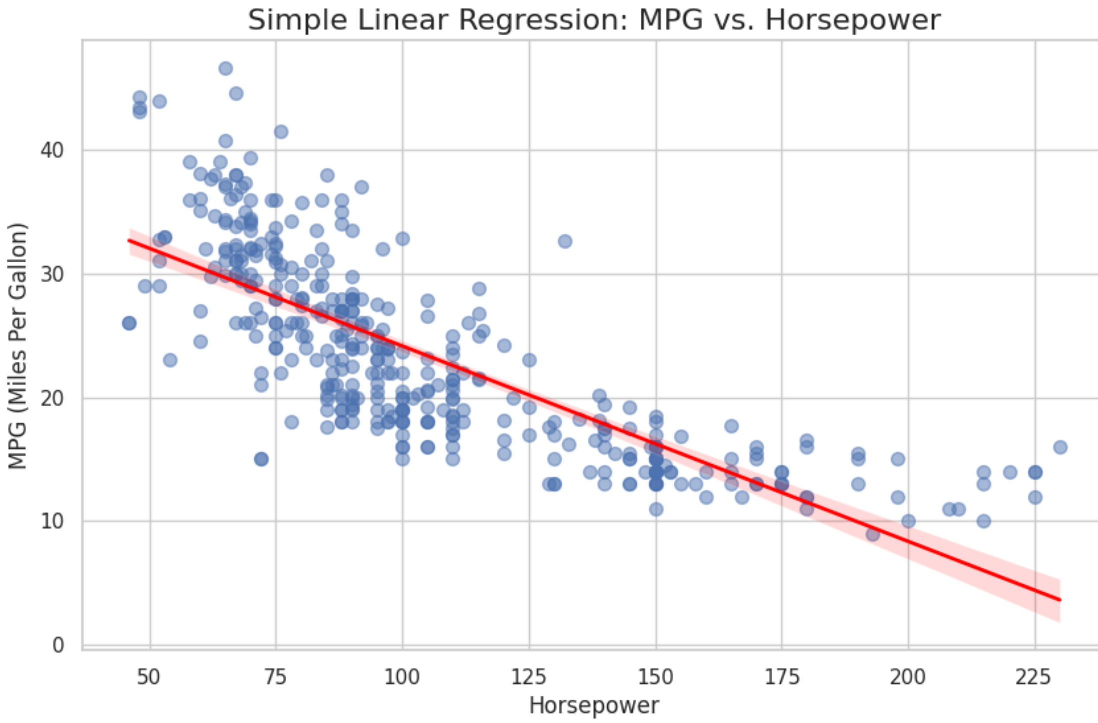
explained by ***horsepower***.

- **iii. Is the relationship positive or negative?**
  - *Hint: Look at the* ***coef*** *for* ***horsepower***. *A negative sign means as horsepower goes up, mpg goes down.*
- **iv. What is the predicted mpg?**
  - *The output gives the exact value. The* ***confidence interval*** *(e.g.,* ***mean_ci_lower***, ***mean_ci_upper****) is for the* average* mpg for a car with 98 hp. The **prediction interval** (e.g., `obs_ci_lower`, `obs_ci_upper`) is for a *single specific* car with 98 hp, so it's wider.*

1.2.2 Task A(b): Plot Response and Predictor

We'll use Seaborn's `regplot` to easily create a scatter plot and overlay the least squares regression line.

[7]:
```
# --- Task A(b): Plot ---

plt.figure(figsize=(10, 6))
sns.regplot(
    x='horsepower',
    y='mpg',
    data=df_clean,
    line_kws={'color': 'red', 'linewidth': 2},  # Style for the regression line
    scatter_kws={'alpha': 0.5, 's': 50}      # Style for the scatter points
)
plt.title('Simple Linear Regression: MPG vs. Horsepower', fontsize=16)
plt.xlabel('Horsepower', fontsize=12)
plt.ylabel('MPG (Miles Per Gallon)', fontsize=12)
plt.show()
```

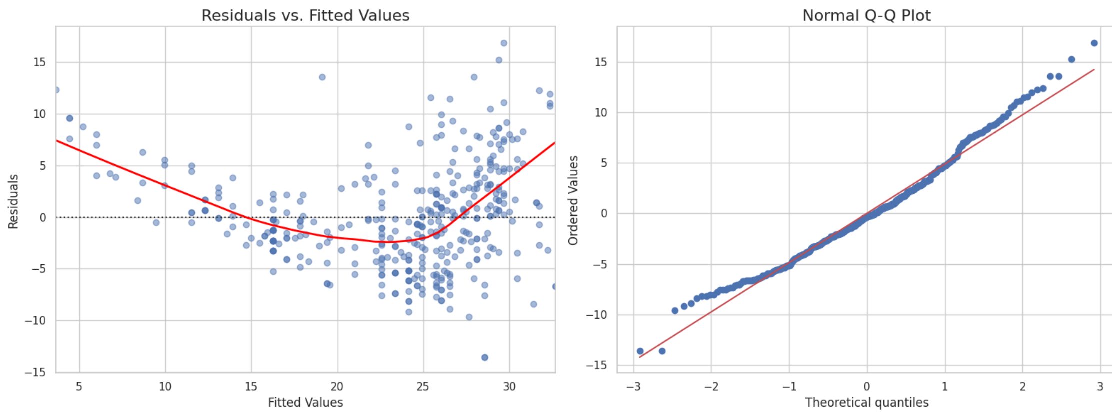


1.2.3 Task A(c): Diagnostic Plots

We need to check the assumptions of the linear model. We'll look at two key plots: 1. **Residuals vs. Fitted Values:** To check for non-linearity (i.e., patterns like a U-shape) and non-constant variance (heteroscedasticity). 2. **Normal Q-Q Plot:** To check if the residuals are normally distributed.

[8]:
```
# --- Task A(c): Diagnostics ---

# Get fitted values and residuals from the model
fitted_vals = slr_model.fittedvalues
residuals = slr_model.resid

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# 1. Residuals vs. Fitted Plot
sns.residplot(
    x=fitted_vals,
    y=residuals,
    lowess=True,  # Adds a smooth line to see trends
    ax=ax1,
    line_kws={'color': 'red', 'lw': 2},
    scatter_kws={'alpha': 0.5}
)
```

```
ax1.set_title('Residuals vs. Fitted Values', fontsize=16)
ax1.set_xlabel('Fitted Values', fontsize=12)
ax1.set_ylabel('Residuals', fontsize=12)

# 2. Normal Q-Q Plot
stats.probplot(residuals, dist="norm", plot=ax2)
ax2.set_title('Normal Q-Q Plot', fontsize=16)

plt.tight_layout()
plt.show()
```



Your Commentary for A(c): *(This is where you'll write your comments based on the plots above)*

- *Hint: Look at the* ***Residuals vs. Fitted*** *plot. Do you see a clear curve or U-shape? This would suggest the relationship is* ***not linear****, and a simple straight line isn't the best fit. Does the spread of the points get wider as the fitted values increase? That would be heteroscedasticity.*
- *Hint: Look at the* ***Q-Q Plot****. Do the blue dots fall nicely along the red line? If they curve off at the ends, it suggests the residuals are not perfectly normally distributed.*

1.3 Task B: Multiple Linear Regression

Now we'll build a model to predict `mpg` using all other variables as predictors.

1.3.1 Task B(a): Scatterplot Matrix

A scatterplot matrix (or pairplot) shows the relationship between every pair of variables.

[10]:
```
# --- Task B(a): Scatterplot Matrix ---

print("Generating pairplot... (this may take a moment)")
# We use df_clean (which doesn't have the 'name' column)
sns.pairplot(df_clean)
```

```
plt.show()
```

Generating pairplot… (this may take a moment)

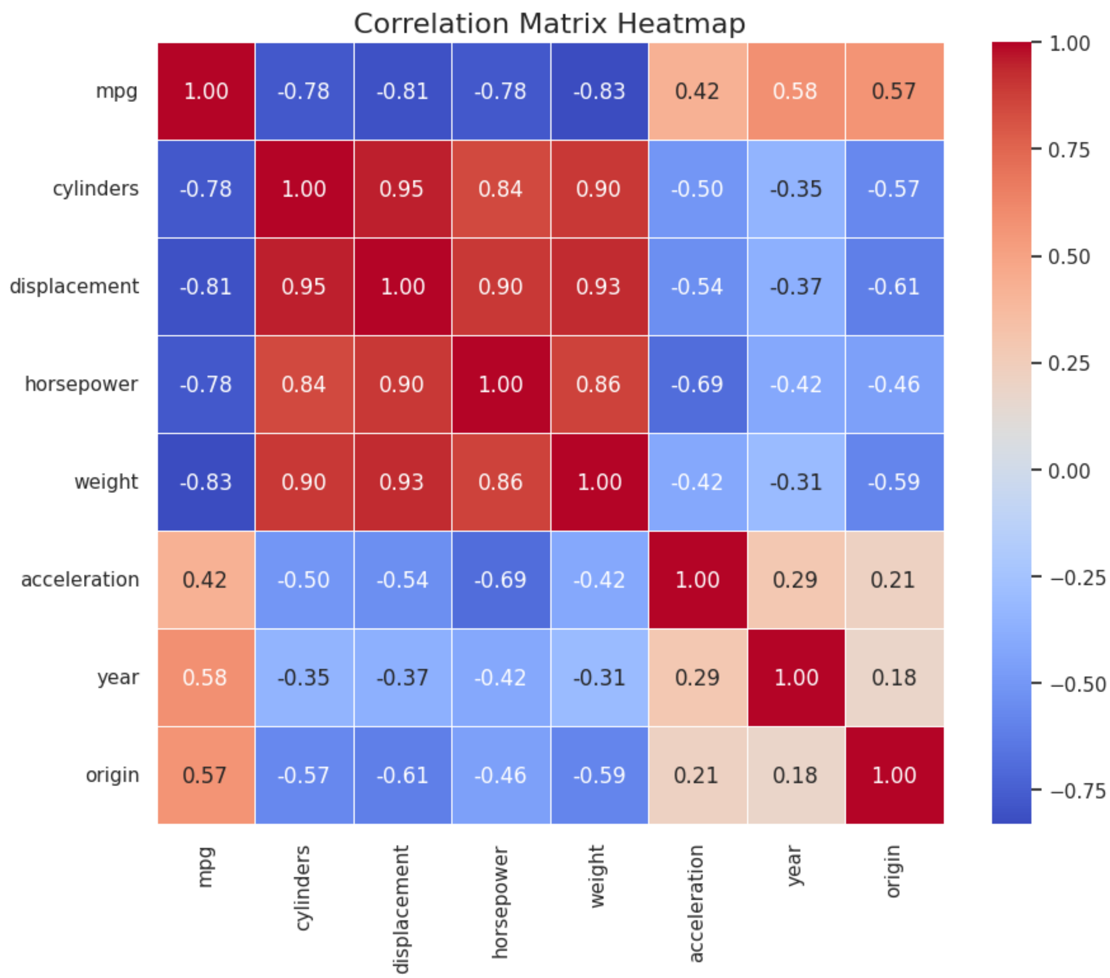
1.3.2 Task B(b): Correlation Matrix

Let's compute the exact correlation coefficients and visualize them with a heatmap.

[11]:
```
# --- Task B(b): Correlation Matrix ---

# Compute the correlation matrix
corr_matrix = df_clean.corr()
```

```python
print("--- Correlation Matrix ---")
print(corr_matrix)

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(
    corr_matrix,
    annot=True,      # Show the numbers in the cells
    cmap='coolwarm',# Color scheme
    fmt='.2f',       # Format numbers to 2 decimal places
    linewidths=0.5
)
plt.title('Correlation Matrix Heatmap', fontsize=16)
plt.show()
```

```
--- Correlation Matrix ---
                   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
```



1.3.3 Task B(c): Fit and Summarize the Model

We fit the multiple linear regression model. The formula `mpg ~ .` is a shortcut that means "use all other columns in the DataFrame as predictors." We'll also run an ANOVA test.

[12]:
```
# --- Task B(c): Fit and Summarize ---

# The formula 'mpg ~ .' uses all other columns as predictors
# Since df_clean only contains our desired columns, this is safe.
all_predictors = ' + '.join(df_clean.columns.drop('mpg'))
formula = f'mpg ~ {all_predictors}'

print(f"Using formula: {formula}\n")

# Fit the model
mlr_model = smf.ols(formula, data=df_clean).fit()
```

```python
# Print the summary
print(mlr_model.summary())

# (i) Run ANOVA test to check overall model significance
anova_table = anova_lm(mlr_model, typ=2)
print("\n--- ANOVA Table ---")
print(anova_table)
```

```
Using formula: mpg ~ cylinders + displacement + horsepower + weight +
acceleration + year + origin

                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     252.4
Date:                Mon, 24 Nov 2025   Prob (F-statistic):          2.04e-139
Time:                        13:04:58   Log-Likelihood:                -1023.5
No. Observations:                 392   AIC:                             2063.
Df Residuals:                     384   BIC:                             2095.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept      -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):               53.100
Skew:                           0.529   Prob(JB):                     2.95e-12
Kurtosis:                       4.460   Cond. No.                     8.59e+04
==============================================================================

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

--- ANOVA Table ---
                   sum_sq     df            F         PR(>F)
```

```
cylinders         25.791491    1.0    2.329125  1.277965e-01
displacement      77.612668    1.0    7.008884  8.444649e-03
horsepower        16.739754    1.0    1.511699  2.196328e-01
weight          1091.631693    1.0   98.580813  7.874953e-21
acceleration       7.358417    1.0    0.664509  4.154780e-01
year            2402.249906    1.0  216.937408  3.055983e-39
origin           291.134494    1.0   26.291171  4.665681e-07
Residual        4252.212530  384.0         NaN           NaN
```

Your Interpretation for B(c): *(This is where you'll write your comments based on the output above)*

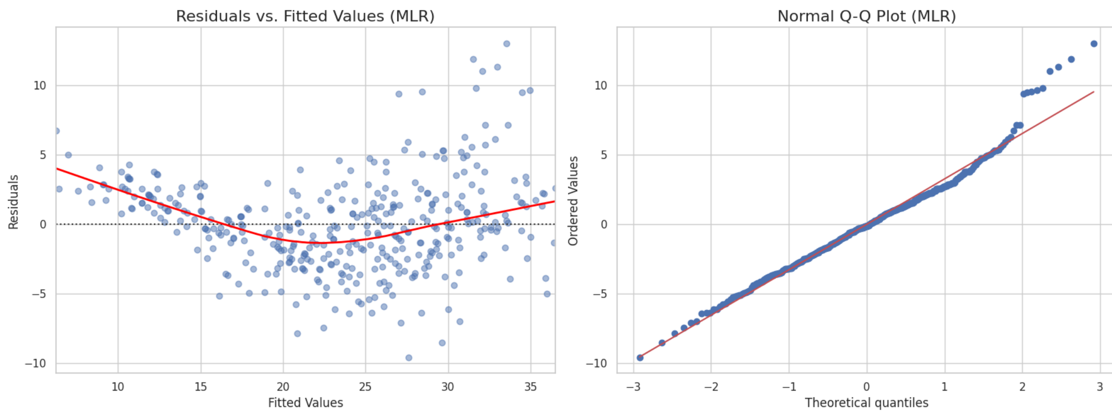
- **i. Is there a relationship?**
  - *Hint: Look at the `Prob (F-statistic)` in the main summary or the `PR(>F)` for the model in the ANOVA table. A very small value means at least one predictor is related to `mpg`.*
- **ii. Which predictors are significant?**
  - *Hint: In the main summary table, look at the `P>|t|` column for each predictor. Which ones are < 0.05? **`displacement`**, **`weight`**, **`year`**, and **`origin`** are likely candidates.*
- **iii. What does the year coefficient suggest?**
  - *Hint: The coefficient (e.g., `~0.75`) means that for each additional model year, the `mpg` is expected to **increase** by 0.75, **holding all other variables constant**.*

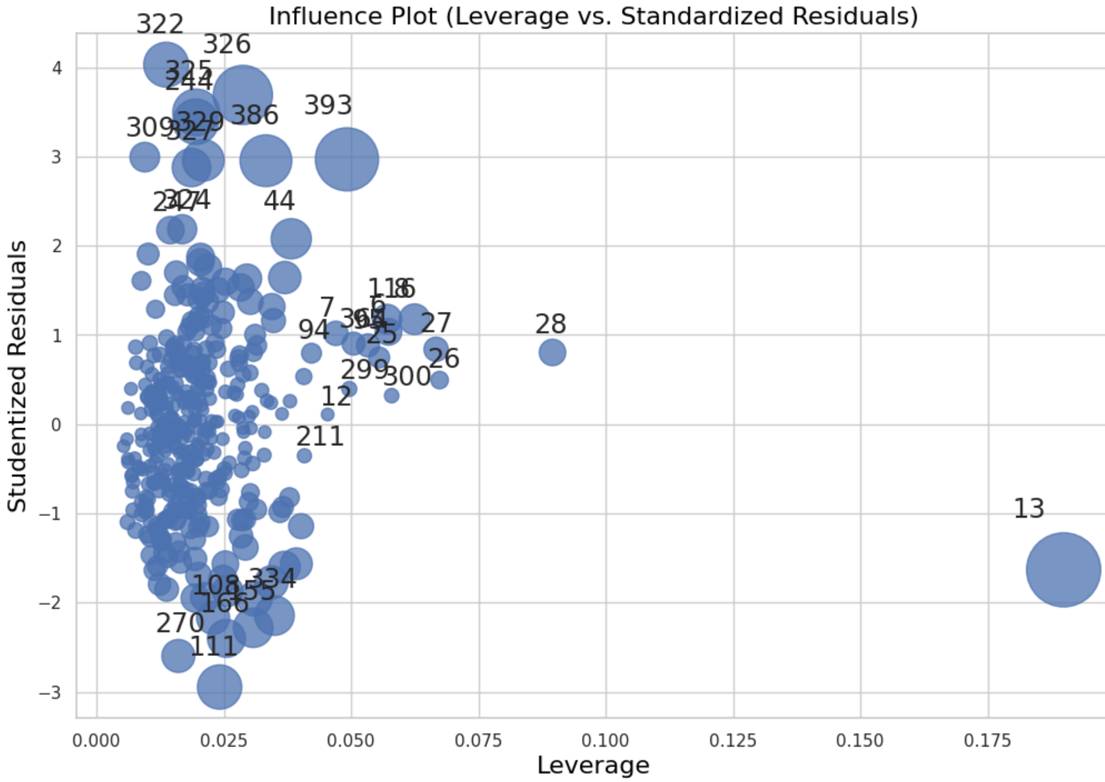
1.3.4 Task B(d): Diagnostic Plots

We repeat the diagnostic plots for the multiple regression model. We also add an **Influence Plot** to find: * **Outliers:** Points with high residuals (far from 0 on the y-axis). * **High-Leverage Points:** Points that have unusual predictor values (far from 0 on the x-axis). These points can have a strong pull on the regression line.

[14]:
```
# --- Task B(d): Diagnostics ---

# Get fitted values and residuals
mlr_fitted_vals = mlr_model.fittedvalues
mlr_residuals = mlr_model.resid

# 1. Residuals vs. Fitted
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
sns.residplot(
    x=mlr_fitted_vals,
    y=mlr_residuals,
    lowess=True,
    ax=ax1,
    line_kws={'color': 'red', 'lw': 2},
    scatter_kws={'alpha': 0.5}
)
ax1.set_title('Residuals vs. Fitted Values (MLR)', fontsize=16)
ax1.set_xlabel('Fitted Values', fontsize=12)
```

```
ax1.set_ylabel('Residuals', fontsize=12)

# 2. Normal Q-Q Plot
stats.probplot(mlr_residuals, dist="norm", plot=ax2)
ax2.set_title('Normal Q-Q Plot (MLR)', fontsize=16)
plt.tight_layout()
plt.show()

# 3. Influence Plot (Leverage vs. Residuals)
fig, ax = plt.subplots(figsize=(12, 8))
influence_plot(mlr_model, ax=ax, criterion="cooks")
plt.title('Influence Plot (Leverage vs. Standardized Residuals)', fontsize=16)
plt.show()
```





Your Commentary for B(d): *(This is where you'll write your comments based on the plots above)*

- *Hint: Does the* ***Residuals vs. Fitted*** *plot look better or worse than the simple model? Is the curve less pronounced?*
- *Hint: In the* ***Influence Plot****, look for points with high leverage (far right on x-axis) or large residuals (far up/down on y-axis). The size of the bubble indicates Cook's distance (a measure of overall influence). Are there any points that are both high-leverage and high-residual? The point labeled 13 or 14 often stands out as high leverage.*

1.3.5 Task B(e): Fit Models with Interactions

Let's see if the effect of one variable depends on another. We can test an interaction term using the `*` or `:` syntax. * `a * b` includes `a`, `b`, and the interaction `a:b`. * `a : b` includes *only* the interaction.

Let's test `horsepower * weight`.

[15]:
```
# --- Task B(e): Interactions ---

# We'll use all predictors, but add an interaction between horsepower and weight
# We use '*' which includes horsepower, weight, AND their interaction (horsepower:weight)
```

```python
interaction_formula = 'mpg ~ cylinders + displacement + horsepower * weight +␣
↪acceleration + year + origin'

interaction_model = smf.ols(interaction_formula, data=df_clean).fit()
print(interaction_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.862
Model:                            OLS   Adj. R-squared:                  0.859
Method:                 Least Squares   F-statistic:                     298.6
Date:                Mon, 24 Nov 2025   Prob (F-statistic):          1.88e-159
Time:                        13:06:34   Log-Likelihood:                -973.24
No. Observations:                 392   AIC:                             1964.
Df Residuals:                     383   BIC:                             2000.
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept             2.8757      4.511      0.638      0.524      -5.993
11.744
cylinders            -0.0296      0.288     -0.103      0.918      -0.596
0.537
displacement          0.0059      0.007      0.881      0.379      -0.007
0.019
horsepower           -0.2313      0.024     -9.791      0.000      -0.278
-0.185
weight               -0.0112      0.001    -15.393      0.000      -0.013
-0.010
horsepower:weight  5.529e-05   5.23e-06     10.577      0.000     4.5e-05
6.56e-05
acceleration         -0.0902      0.089     -1.019      0.309      -0.264
0.084
year                  0.7695      0.045     17.124      0.000       0.681
0.858
origin                0.8344      0.251      3.320      0.001       0.340
1.329
==============================================================================
Omnibus:                       40.936   Durbin-Watson:                   1.474
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.199
Skew:                           0.629   Prob(JB):                     1.27e-16
Kurtosis:                       4.703   Cond. No.                     1.23e+07
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.23e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Your Commentary for B(e): *(This is where you'll write your comments based on the output above)*

- *Hint: Look at the summary table for the new interaction term,* ***`horsepower:weight`****. Is its* `P>|t|` *value significant (e.g., < 0.05)? If yes, this suggests the effect of* **`horsepower`** *on* **`mpg`** ***depends on*** *the* **`weight`** *of the car (and vice-versa).*

1.3.6 Task B(f): Variable Transformations

The residual plots in Task A showed a non-linear pattern. This suggests we might get a better fit by transforming our predictors. Let's try `log(X)`, `sqrt(X)`, and `X^2`.

We use `I()` in the formula to perform calculations like `I(horsepower**2)`.

[17]:

```python
# --- Task B(f): Transformations ---

# We'll build a few different models to compare
# We use np.log() and I() directly in the formulas

# 1. Log transformation on horsepower
log_model = smf.ols('mpg ~ np.log(horsepower) + weight + year + origin',␣
 ↪data=df_clean).fit()
print("--- Model with log(horsepower) ---")
print(log_model.summary())

# 2. Quadratic transformation (polynomial) on horsepower
# We include both horsepower and horsepower^2
poly_model = smf.ols('mpg ~ horsepower + I(horsepower**2) + weight + year +␣
 ↪origin', data=df_clean).fit()
print("\n--- Model with horsepower^2 ---")
print(poly_model.summary())
```

```
--- Model with log(horsepower) ---
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.828
Model:                            OLS   Adj. R-squared:                  0.826
Method:                 Least Squares   F-statistic:                     464.5
Date:                Mon, 24 Nov 2025   Prob (F-statistic):          2.91e-146
Time:                        13:07:45   Log-Likelihood:                -1016.6
No. Observations:                 392   AIC:                             2043.
Df Residuals:                     387   BIC:                             2063.
Df Model:                           4
```

```
Covariance Type:            nonrobust
================================================================================
======
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept             4.3697      6.095      0.717      0.474      -7.614
16.354
np.log(horsepower)   -4.9700      1.040     -4.780      0.000      -7.014
-2.926
weight               -0.0043      0.000     -9.741      0.000      -0.005
-0.003
year                  0.6925      0.049     14.154      0.000       0.596
0.789
origin                1.2467      0.253      4.929      0.000       0.749
1.744
==============================================================================
Omnibus:                       28.598   Durbin-Watson:                   1.314
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               42.111
Skew:                           0.525   Prob(JB):                     7.17e-10
Kurtosis:                       4.215   Cond. No.                     1.16e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.16e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

--- Model with horsepower^2 ---
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.849
Method:                 Least Squares   F-statistic:                     439.5
Date:                Mon, 24 Nov 2025   Prob (F-statistic):         7.11e-157
Time:                        13:07:45   Log-Likelihood:                -988.57
No. Observations:                 392   AIC:                             1989.
Df Residuals:                     386   BIC:                             2013.
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
======
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
```

```
Intercept               -6.6457      3.915     -1.698      0.090     -14.343
1.052
horsepower              -0.2441      0.027     -9.099      0.000      -0.297
-0.191
I(horsepower ** 2)       0.0008   9.13e-05      9.170      0.000       0.001
0.001
weight                  -0.0044      0.000    -10.426      0.000      -0.005
-0.004
year                     0.7456      0.046     16.145      0.000       0.655
0.836
origin                   1.0465      0.238      4.405      0.000       0.579
1.514
==============================================================================
Omnibus:                       21.819   Durbin-Watson:                   1.500
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               32.447
Skew:                           0.414   Prob(JB):                     9.00e-08
Kurtosis:                       4.140   Cond. No.                     4.10e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.1e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Your Commentary for B(f): *(This is where you'll write your comments based on the output above)*

- *Hint: Compare the `R-squared` and `Adj. R-squared` of these new models to the original multiple regression model (`mlr_model`). Are they higher? A higher `Adj. R-squared` indicates a better model fit.*
- *Look at the p-values for the new terms. Is `np.log(horsepower)` significant? Are both `horsepower` and `I(horsepower**2)` significant in the polynomial model? If the `I(horsepower**2)` term is significant, it strongly confirms that the relationship between `horsepower` and `mpg` is non-linear (quadratic).*
- *For a complete analysis, you would also re-run the diagnostic plots (Residuals vs. Fitted) for your* best* new model to confirm that the non-linear pattern is gone.*

1.4 Reflective Summary

(This is where you write your final summary for your blog post)

- *Example: "In this analysis, we explored the `Auto` dataset to predict fuel efficiency (mpg). A simple linear regression showed a strong, negative relationship between `horsepower` and `mpg`, but diagnostic plots revealed a clear non-linear pattern. A multiple regression model identified `weight`, `year`, and `origin` as the most significant predictors. By testing transformations, we found that a model including a quadratic term (`I(horsepower**2)`) or a log term (`np.log(horsepower)`) provided a better fit, as indicated by a higher Adjusted R-squared and more significant terms. This confirms that the relationship between horsepower and fuel effi-*

ciency is not a simple straight line. The diagnostic plots for the transformed model showed... which suggests... A key takeaway is the importance of...