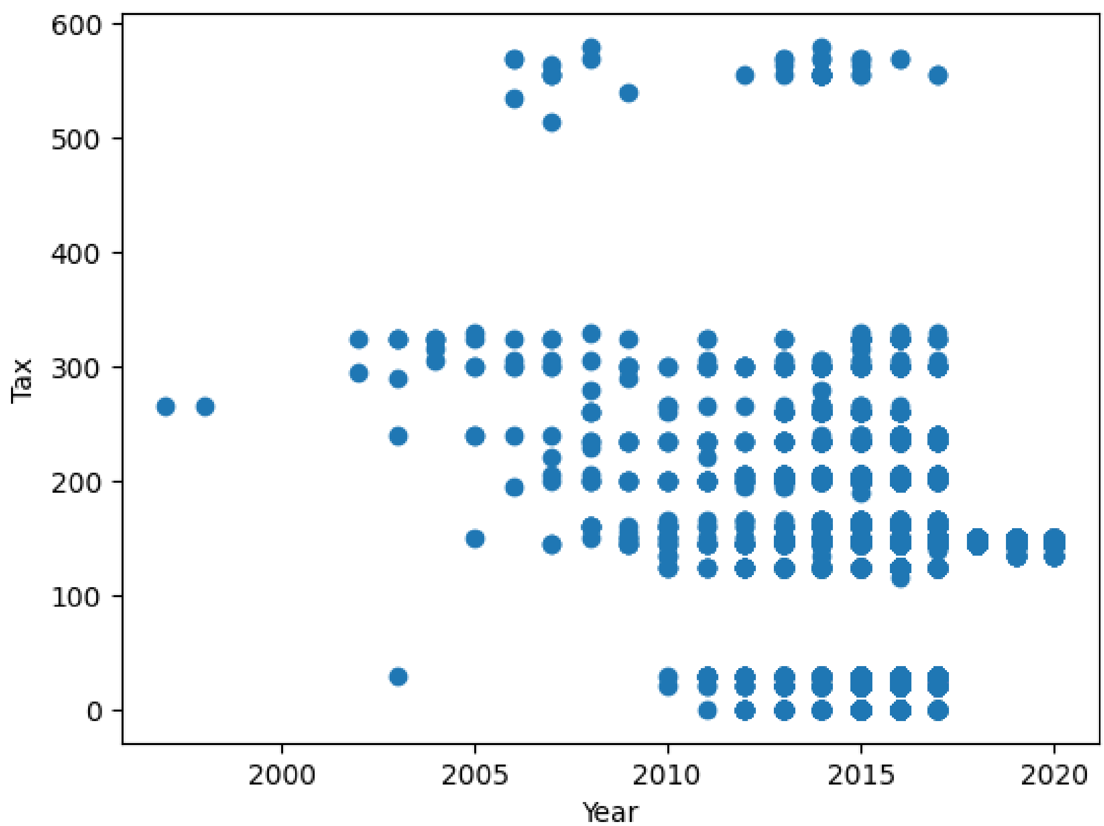
In [1]:

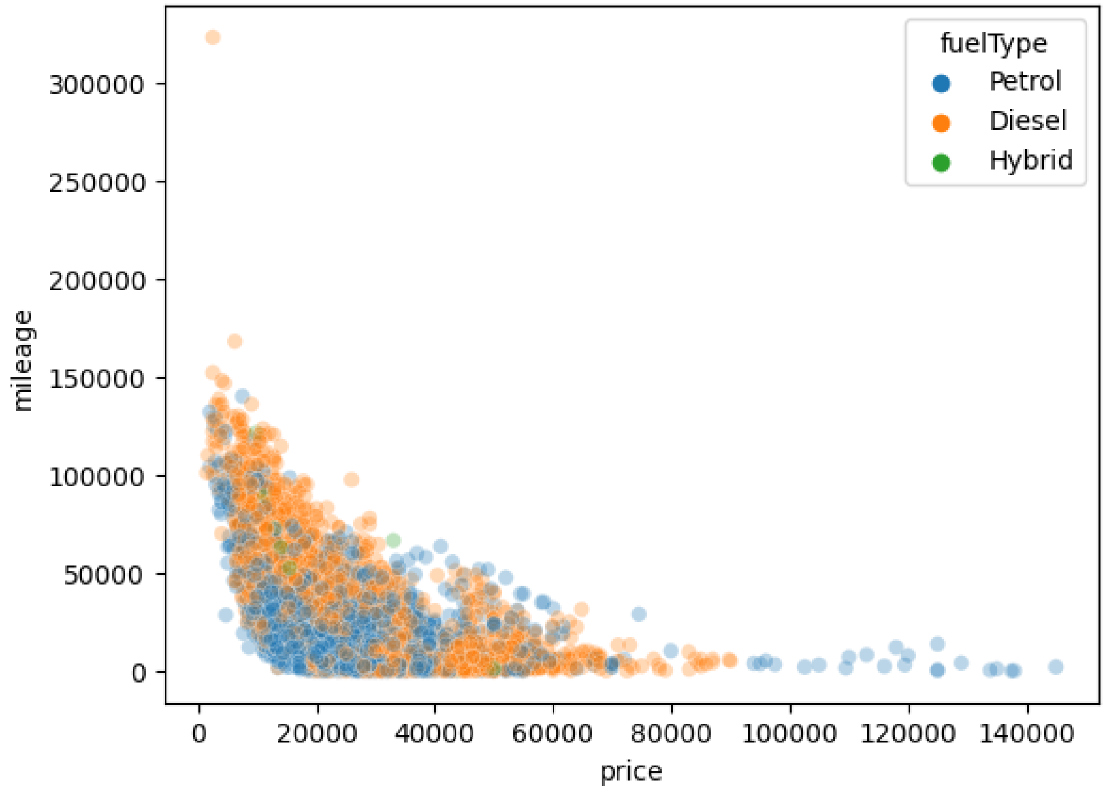
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Read the data
data = pd.read_csv("C:/Users/user/Desktop/My learning/ClinSoft/audi.csv")

# Basic data exploration
print(data.columns)
print(data.info())
print(data.describe())
```

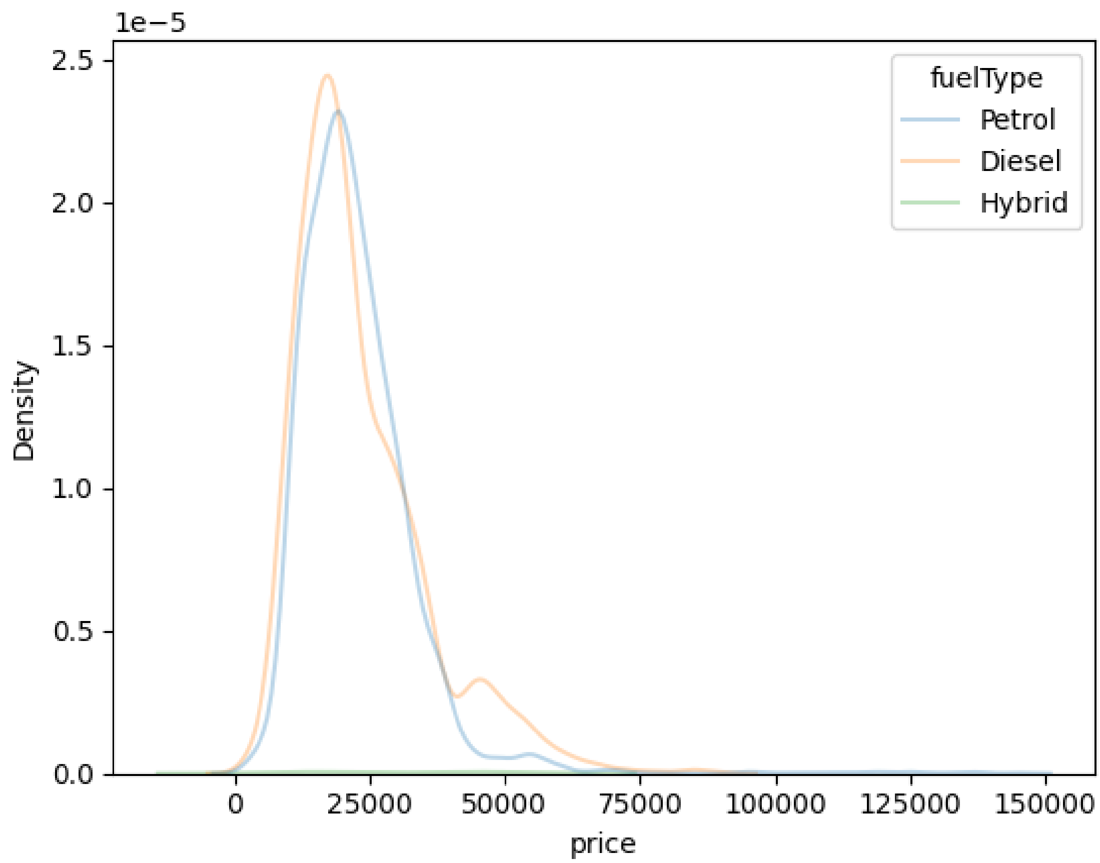
```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType',
'tax',
       'mpg', 'engineSize'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10668 entries, 0 to 10667
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         10668 non-null  object 
 1   year          10668 non-null  int64  
 2   price         10668 non-null  int64  
 3   transmission  10668 non-null  object 
 4   mileage       10668 non-null  int64  
 5   fuelType      10668 non-null  object 
 6   tax           10668 non-null  int64  
 7   mpg           10668 non-null  float64
 8   engineSize    10668 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 750.2+ KB
None
               year          price        mileage           tax      
mpg  \
count  10668.000000   10668.000000   10668.000000  10668.000000  10668.00
0000   
mean    2017.100675   22896.685039   24827.244001    126.011436     50.77
0022   
std        2.167494   11714.841888   23505.257205     67.170294     12.94
9782   
min     1997.000000    1490.000000       1.000000      0.000000     18.90
0000   
25%     2016.000000   15130.750000    5968.750000    125.000000     40.90
0000   
50%     2017.000000   20200.000000   19000.000000    145.000000     49.60
0000   
75%     2019.000000   27990.000000   36464.500000    145.000000     58.90
0000   
max     2020.000000  145000.000000  323000.000000    580.000000    188.30
0000   

         engineSize  
count  10668.000000  
mean       1.930709  
std        0.602957  
min        0.000000  
25%        1.500000  
50%        2.000000  
75%        2.000000  
max        6.300000  
```

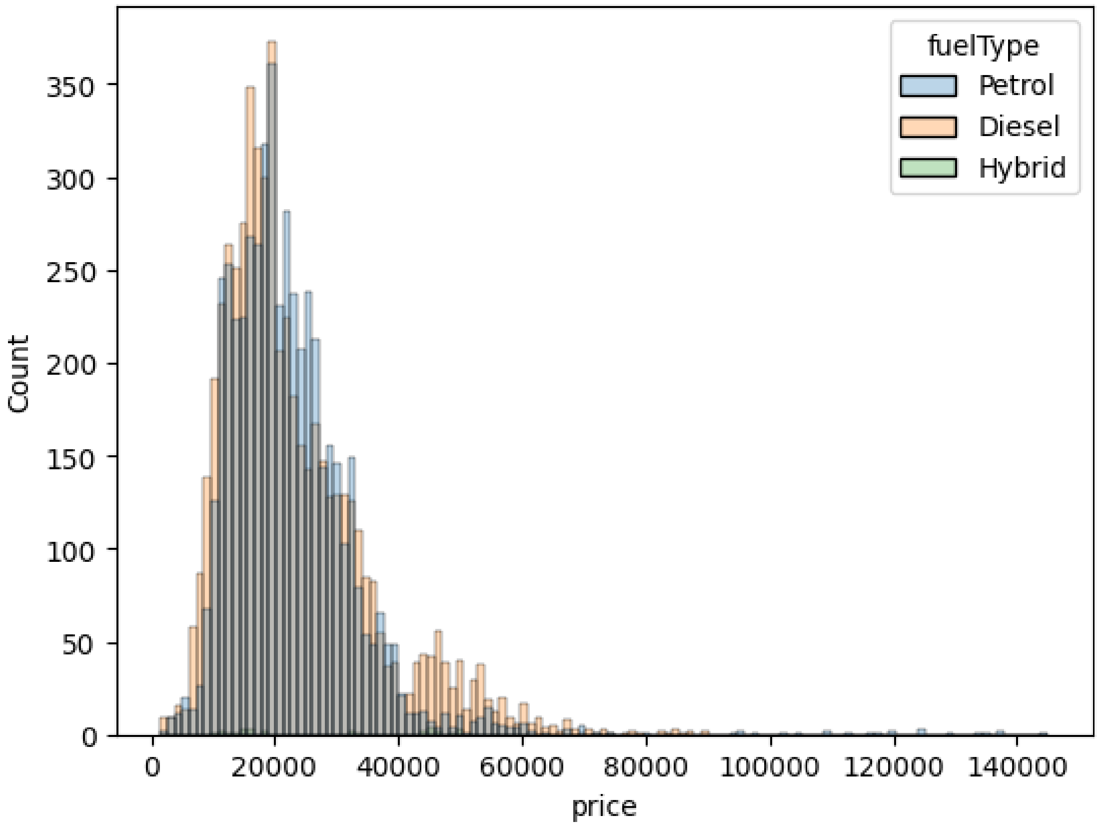
In [2]:

```
# Scatter plot
plt.scatter(data['year'], data['tax'])
plt.xlabel('Year')
plt.ylabel('Tax')
plt.show()

# Scatter plot with color
sns.scatterplot(data=data, x='price', y='mileage', hue='fuelType', alpha=0
plt.show()

# Density plot
sns.kdeplot(data=data, x='price', hue='fuelType', alpha=0.3)
plt.show()

# Histogram
sns.histplot(data=data, x='price', hue='fuelType', alpha=0.3)
plt.show()
```



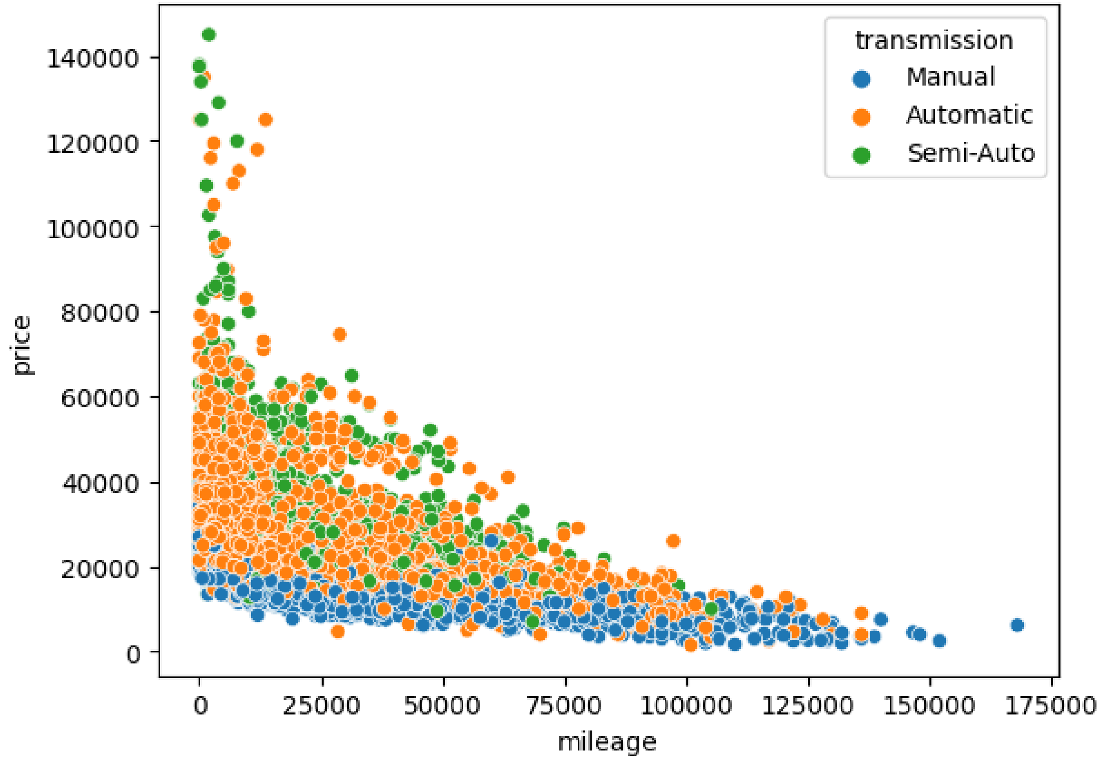

fuelType
Petrol
Diesel
Hybrid
mileage
300000
250000
200000
150000
100000
50000
0
0
20000
40000
60000
80000
100000
120000
140000
price


1e−5
fuelType
Petrol
Diesel
Hybrid
Density
2.5
2.0
1.5
1.0
0.5
0.0
0
25000
50000
75000
100000
125000
150000
price


fuelType
Petrol
Diesel
Hybrid
Count
350
300
250
200
150
100
50
0
0
20000
40000
60000
80000
100000
120000
140000
price

In [3]:

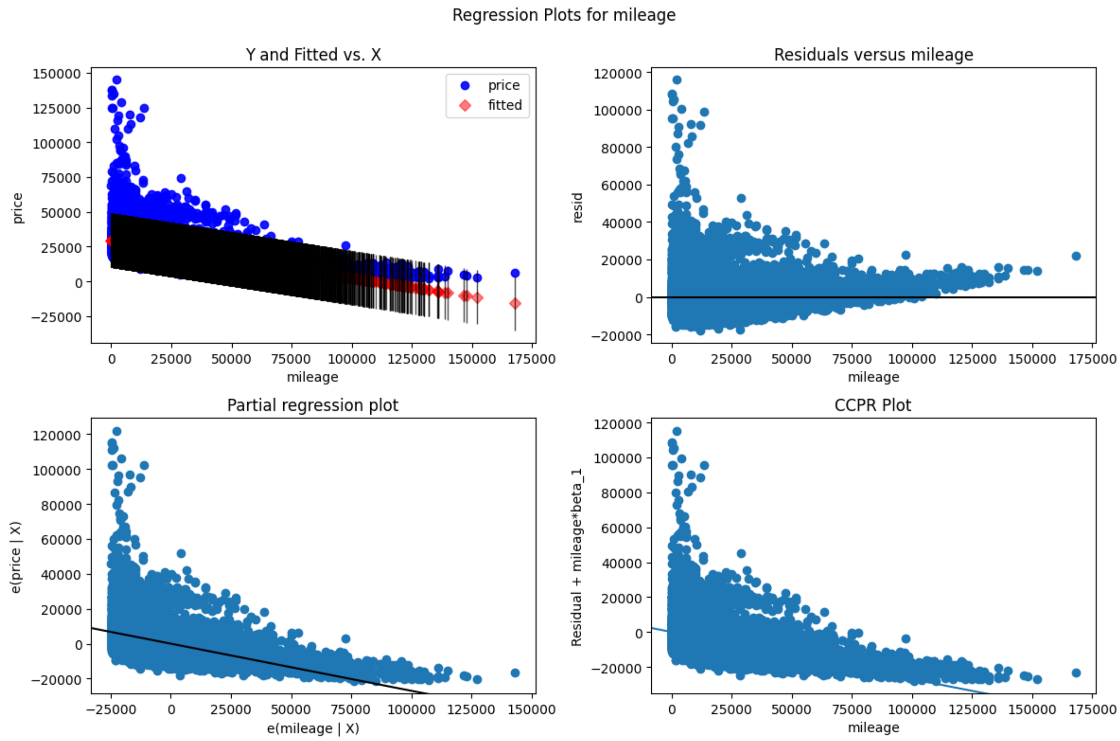
```
# Filter and scatter plot
first_data = data[data['mileage'] < 180000]
sns.scatterplot(data=first_data, x='mileage', y='price', hue='transmission
plt.show()
```



In [4]:

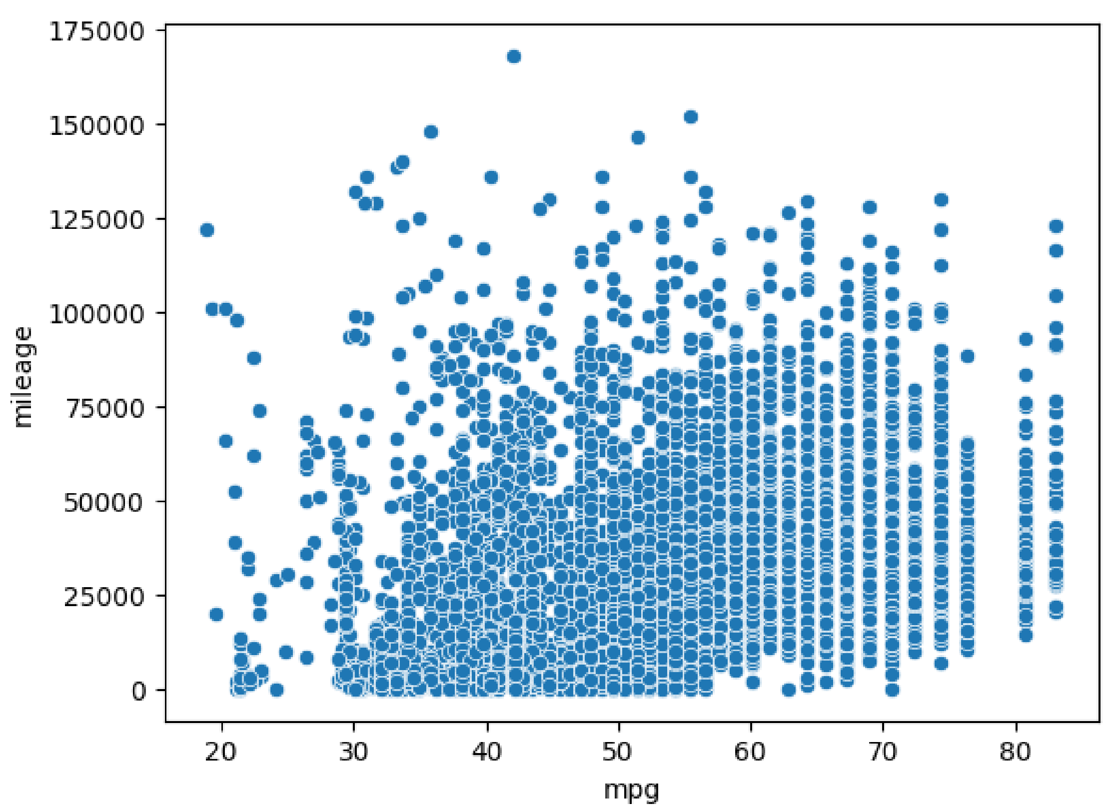
```
# Linear regression and residual analysis
model1 = sm.OLS(first_data['price'], sm.add_constant(first_data['mileage']
print(model1.summary())
sm.graphics.plot_regress_exog(model1, 'mileage', fig=plt.figure(figsize=(1
plt.show()
```

```
                            OLS Regression Results
==============================================================================
=====
Dep. Variable:                  price   R-squared:
0.289
Model:                            OLS   Adj. R-squared:
0.289
Method:                 Least Squares   F-statistic:
4331.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):
0.00
Time:                        14:57:33   Log-Likelihood:            -1.132
5e+05
No. Observations:               10667   AIC:                         2.26
5e+05
Df Residuals:                   10665   BIC:                         2.26
5e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
const       2.959e+04    139.601    211.967      0.000    2.93e+04    2.9
9e+04
mileage       -0.2699      0.004    -65.814      0.000      -0.278      -
0.262
==============================================================================
=====
Omnibus:                     7262.777   Durbin-Watson:
1.762
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           16597
7.135
Skew:                           2.958   Prob(JB):
0.00
Kurtosis:                      21.397   Cond. No.                     4.9
7e+04
==============================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
[2] The condition number is large, 4.97e+04. This might indicate that the
re are
strong multicollinearity or other numerical problems.
eval_env: 1
```



In [5]:

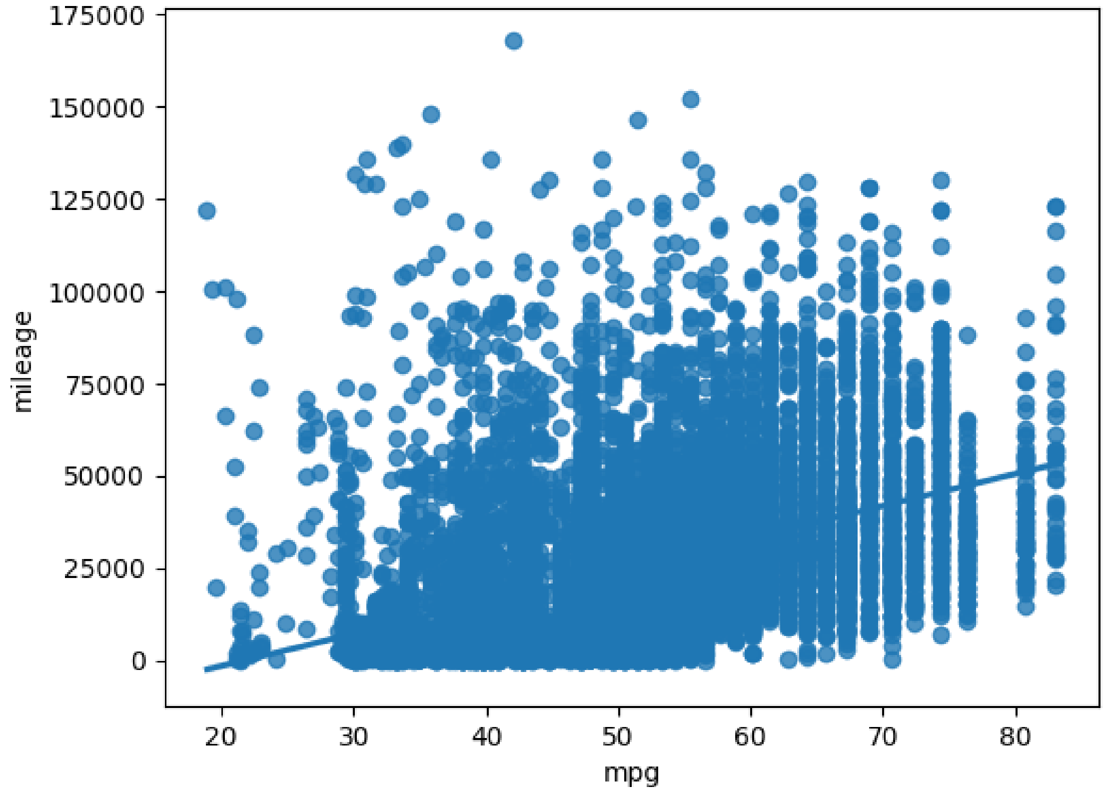
```
# Filter and scatter plot with another variable
second_data = data[(data['mileage'] < 180000) & (data['mpg'] < 100)]
sns.scatterplot(data=second_data, x='mpg', y='mileage')
plt.show()
```



In [6]:

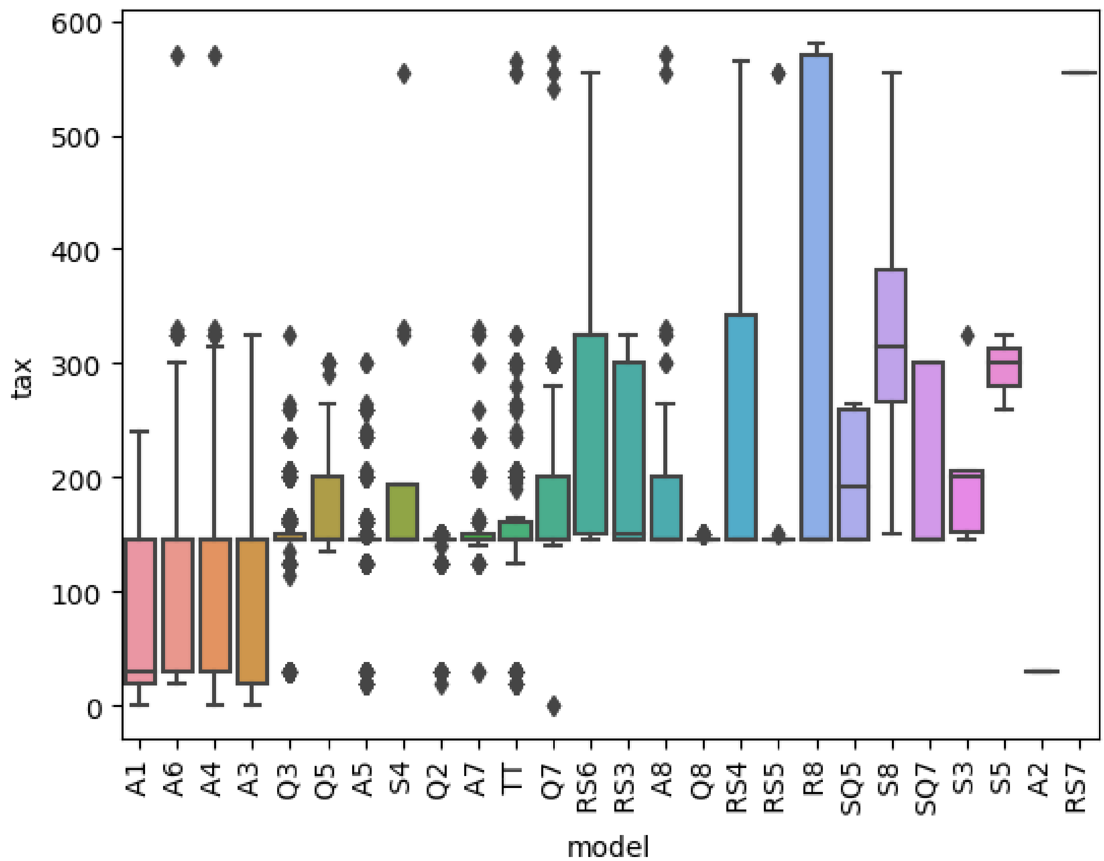
```
# Linear regression with scatter plot and regression line
model2 = sm.OLS(second_data['mileage'], sm.add_constant(second_data['mpg']
print(model2.summary())
sns.regplot(data=second_data, x='mpg', y='mileage')
plt.show()
```

```
                            OLS Regression Results
==========================================================================
=====
Dep. Variable:                mileage   R-squared:
0.189
Model:                            OLS   Adj. R-squared:
0.189
Method:                 Least Squares   F-statistic:
2474.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):
0.00
Time:                        14:58:07   Log-Likelihood:              -1.209
2e+05
No. Observations:               10634   AIC:                           2.41
8e+05
Df Residuals:                   10632   BIC:                           2.41
8e+05
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-----
const      -1.898e+04    903.199    -21.011      0.000   -2.07e+04    -1.7
2e+04
mpg          867.2376     17.435     49.740      0.000     833.061      90
1.414
==========================================================================
=====
Omnibus:                     3768.420   Durbin-Watson:
1.289
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              1401
8.667
Skew:                           1.766   Prob(JB):
0.00
Kurtosis:                       7.378   Cond. No.
230.
==========================================================================
=====

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


175000
150000
125000
100000
75000
50000
25000
0
mileage
20
30
40
50
60
70
80
mpg

In [7]:

```
# Boxplot
sns.boxplot(data=data, x='model', y='tax')
plt.xticks(rotation=90)
plt.show()
```

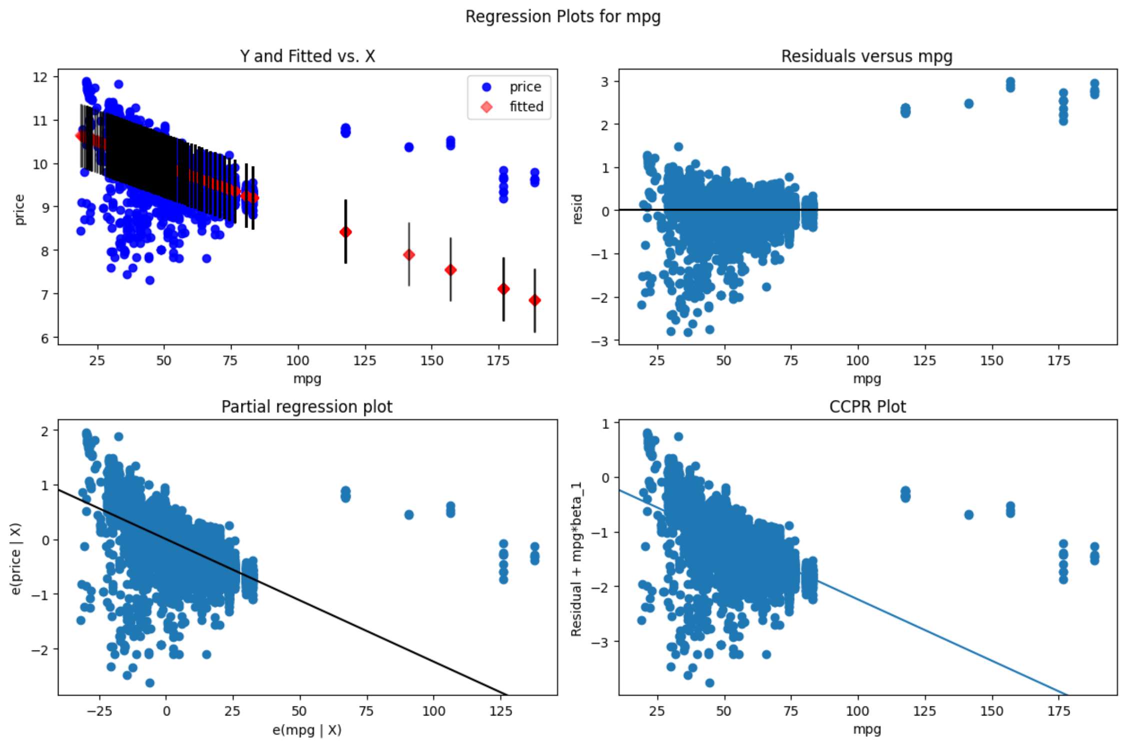


In [11]:

```
# Transformation and filtering
data['good_mpg'] = np.where((data['mpg'] >= 50) & (data['mpg'] <= 60), 'Go
print(data.head())

# Linear regression on transformed data
model3 = sm.OLS(np.log(data['price']), sm.add_constant(data['mpg'])).fit()
print(model3.summary())
sm.graphics.plot_regress_exog(model3, 'mpg', fig=plt.figure(figsize=(12, 8
plt.show()
```

```
  model  year  price transmission  mileage fuelType  tax   mpg  engineSiz
e  \
0    A1  2017  12500       Manual    15735   Petrol  150  55.4          1.
4
1    A6  2016  16500    Automatic    36203   Diesel   20  64.2          2.
0
2    A1  2016  11000       Manual    29946   Petrol   30  55.4          1.
4
3    A4  2017  16800    Automatic    25952   Diesel  145  67.3          2.
0
4    A3  2019  17300       Manual     1998   Petrol  145  49.6          1.
0

   good_mpg
0  Good MPG
1
2  Good MPG
3
4
                            OLS Regression Results
==========================================================================
=====
Dep. Variable:                  price   R-squared:
0.379
Model:                            OLS   Adj. R-squared:
0.379
Method:                 Least Squares   F-statistic:
6519.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):
0.00
Time:                        15:00:53   Log-Likelihood:                -4
558.2
No. Observations:               10668   AIC:
9120.
Df Residuals:                   10666   BIC:
9135.
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-----
const         11.0651      0.015    761.339      0.000      11.037      1
1.094
mpg           -0.0224      0.000    -80.742      0.000      -0.023      -
0.022
==========================================================================
=====
Omnibus:                     2062.976   Durbin-Watson:
1.395
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            5689
7.534
Skew:                          -0.200   Prob(JB):
0.00
Kurtosis:                      14.307   Cond. No.
```

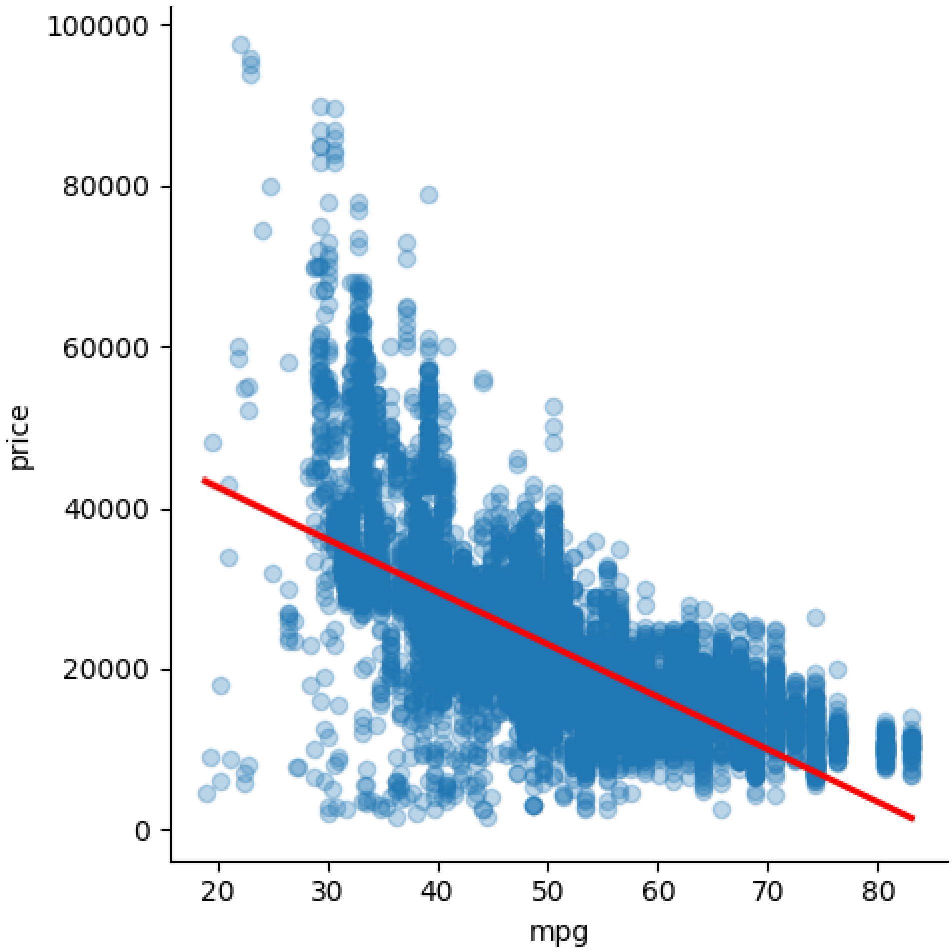
```
212.
==============================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
eval_env: 1
```



In [15]:

```
# Filtering and regression plot
filtered_data = data[(data['price'] < 100000) & (np.log(data['mpg']) < 4.5
sns.lmplot(data=filtered_data, x='mpg', y='price', line_kws={'color': 'red
plt.show()
```

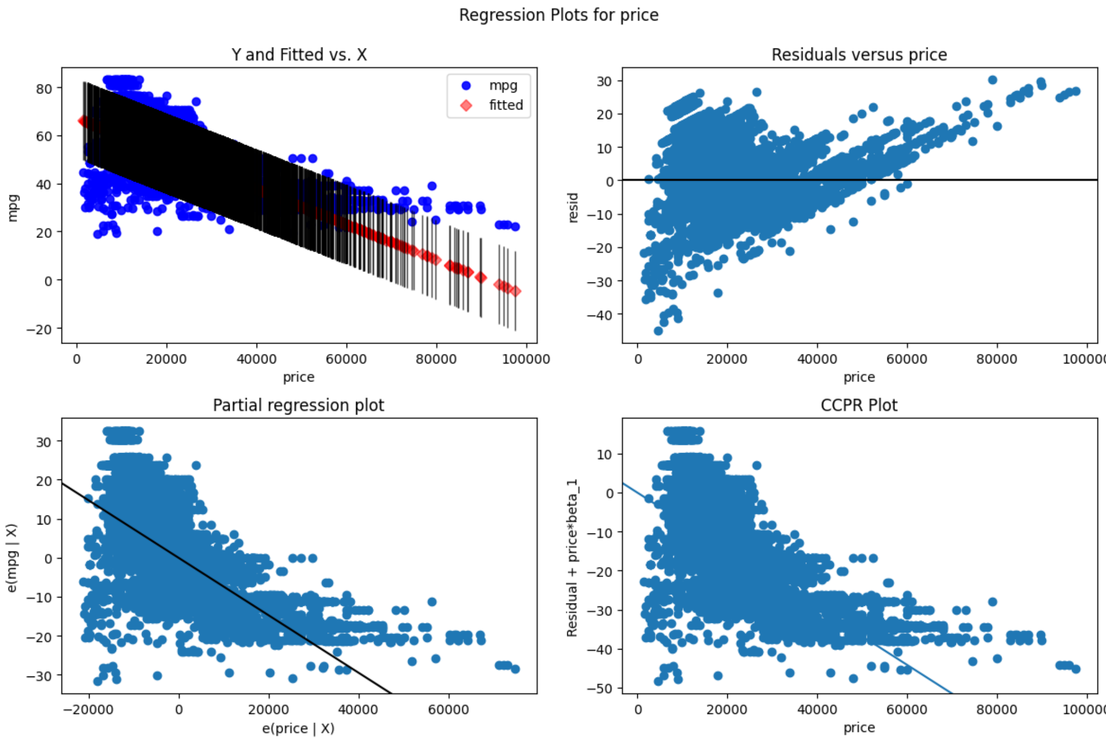
```
C:\Users\user\Anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

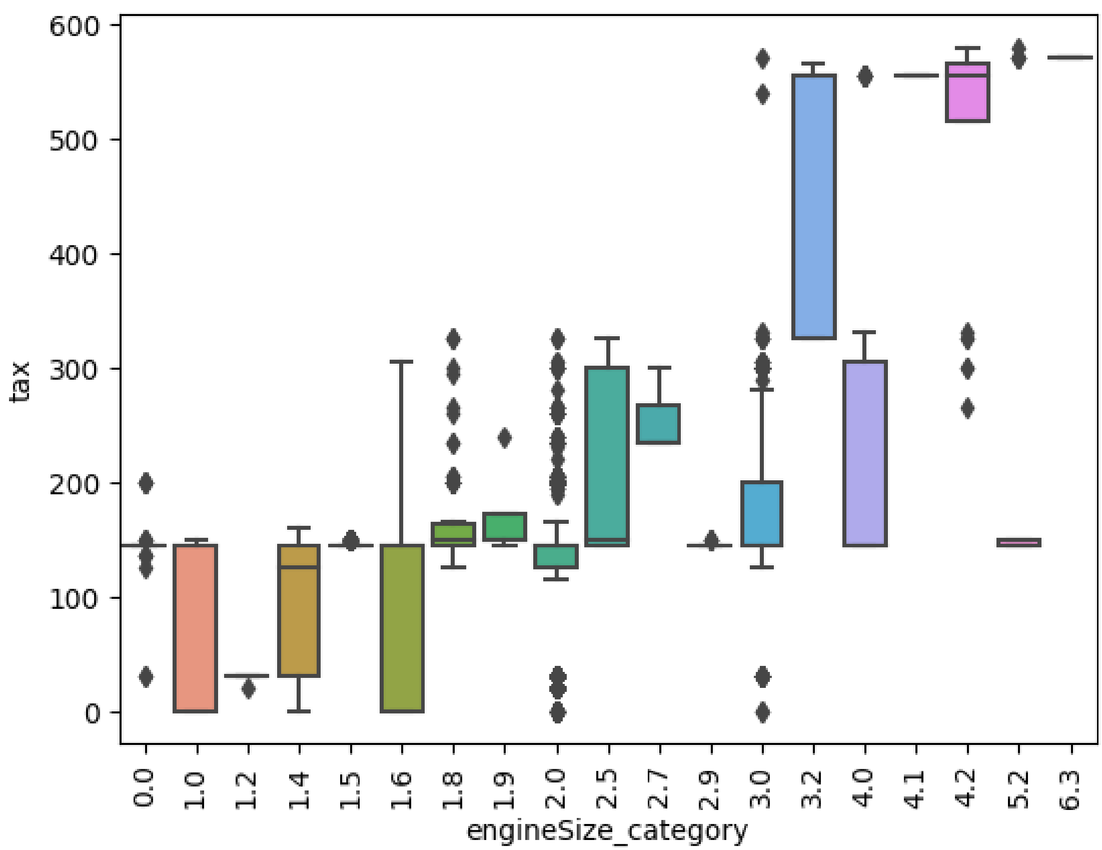


In [20]:

```
# Linear regression and residual analysis
model4 = sm.OLS(filtered_data['mpg'], sm.add_constant(filtered_data['price
print(model4.summary())
sm.graphics.plot_regress_exog(model4, 'price', fig=plt.figure(figsize=(12,
plt.show()

# Data transformation and visualization
data['age'] = abs(data['year'] - 2020)
data['engineSize_category'] = pd.Categorical(data['engineSize'])
sns.boxplot(data=data, x='engineSize_category', y='tax')
plt.xticks(rotation=90)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
=====
Dep. Variable:                    mpg   R-squared:
0.480
Model:                            OLS   Adj. R-squared:
0.480
Method:                 Least Squares   F-statistic:
9814.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):
0.00
Time:                        15:07:46   Log-Likelihood:                    -3
7635.
No. Observations:               10617   AIC:                             7.52
7e+04
Df Residuals:                   10615   BIC:                             7.52
9e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
const         67.2274      0.187    358.999      0.000      66.860      6
7.595
price         -0.0007   7.43e-06    -99.067      0.000      -0.001      -
0.001
==============================================================================
=====
Omnibus:                      148.695   Durbin-Watson:
1.778
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                  24
1.600
Skew:                           0.123   Prob(JB):                         3.4
5e-53
Kurtosis:                       3.697   Cond. No.                         5.8
0e+04
==============================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
[2] The condition number is large, 5.8e+04. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
eval_env: 1
```


Regression Plots for price
Y and Fitted vs. X
mpg
fitted
price
Residuals versus price
resid
price
Partial regression plot
e(mpg | X)
e(price | X)
CCPR Plot
Residual + price*beta_1
price


tax
engineSize_category
0.0
1.0
1.2
1.4
1.5
1.6
1.8
1.9
2.0
2.5
2.7
2.9
3.0
3.2
4.0
4.1
4.2
5.2
6.3

In [ ]: