In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
```

In [42]:
```
df_bmw = pd.read_csv("C:/Users/user/Desktop/My learning/ClinSoft/bmw.csv")
```

In [43]:
```
df_merc = pd.read_csv("C:/Users/user/Desktop/My learning/ClinSoft/merc.csv
```

In [44]:
```
df_merc['mark'] = 'mercedes'
df_bmw['mark'] = 'bmw'
```

In [ ]:

In [ ]:

In [ ]:

In [45]:
```
df = pd.concat([df_bmw, df_merc], ignore_index=True)
```

In [46]:
```
pd.set_option('display.max.rows', 10000)
```

In [47]:
```
df['price'].mean()
```

Out[47]: 23812.124435146445

In [48]:
```
df
```

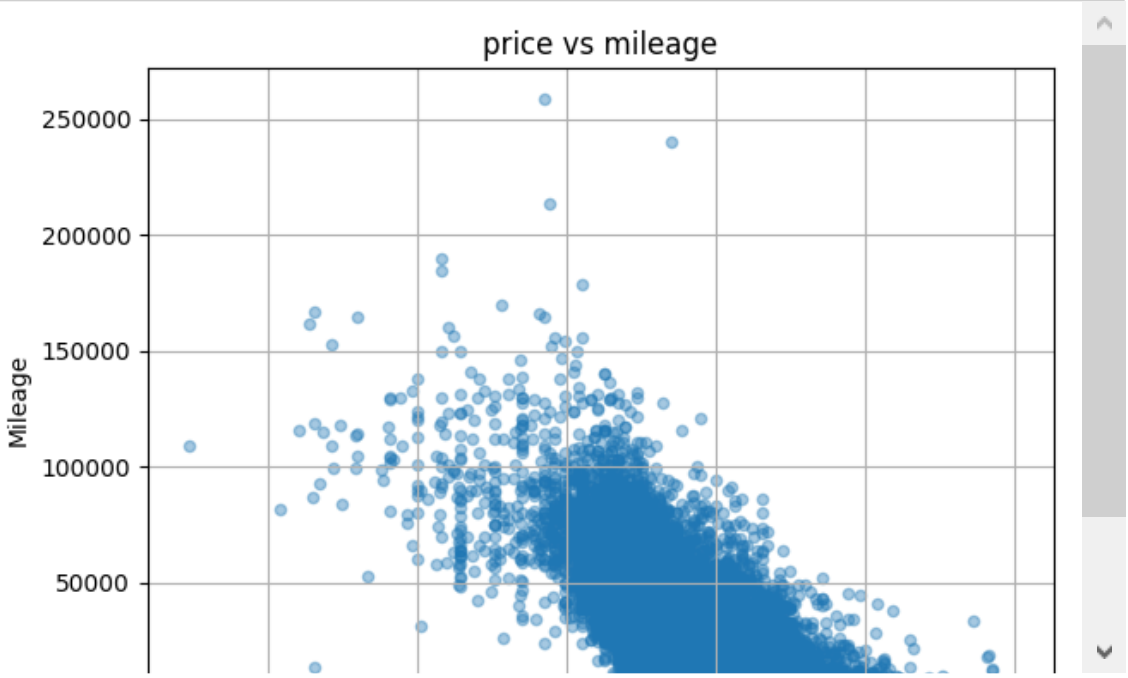
Out[48]:

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize	ma
0	5 Series	2014	11200	Automatic	67068	Diesel	125	57.6	2.0	br
1	6 Series	2018	27000	Automatic	14827	Petrol	145	42.8	2.0	br
2	5 Series	2016	16000	Automatic	62794	Diesel	160	51.4	3.0	br
3	1 Series	2017	12750	Automatic	26676	Diesel	145	72.4	1.5	br
4	7 Series	2014	14500	Automatic	39554	Diesel	160	50.4	3.0	br
...	...	...	...	...	...	...	...	...	...	
23895	C Class	2020	35999	Automatic	500	Diesel	145	55.4	2.0	merced
23896	B Class	2020	24699	Automatic	2500	Diesel	145	55.4	2.0	merced
23897	GLC Class	2019	30999	Automatic	11612	Diesel	145	41.5	2.1	merced
23898	CLS Class	2019	37990	Automatic	2426	Diesel	145	45.6	2.0	merced
23899	S Class	2019	54999	Automatic	2075	Diesel	145	52.3	2.9	merced

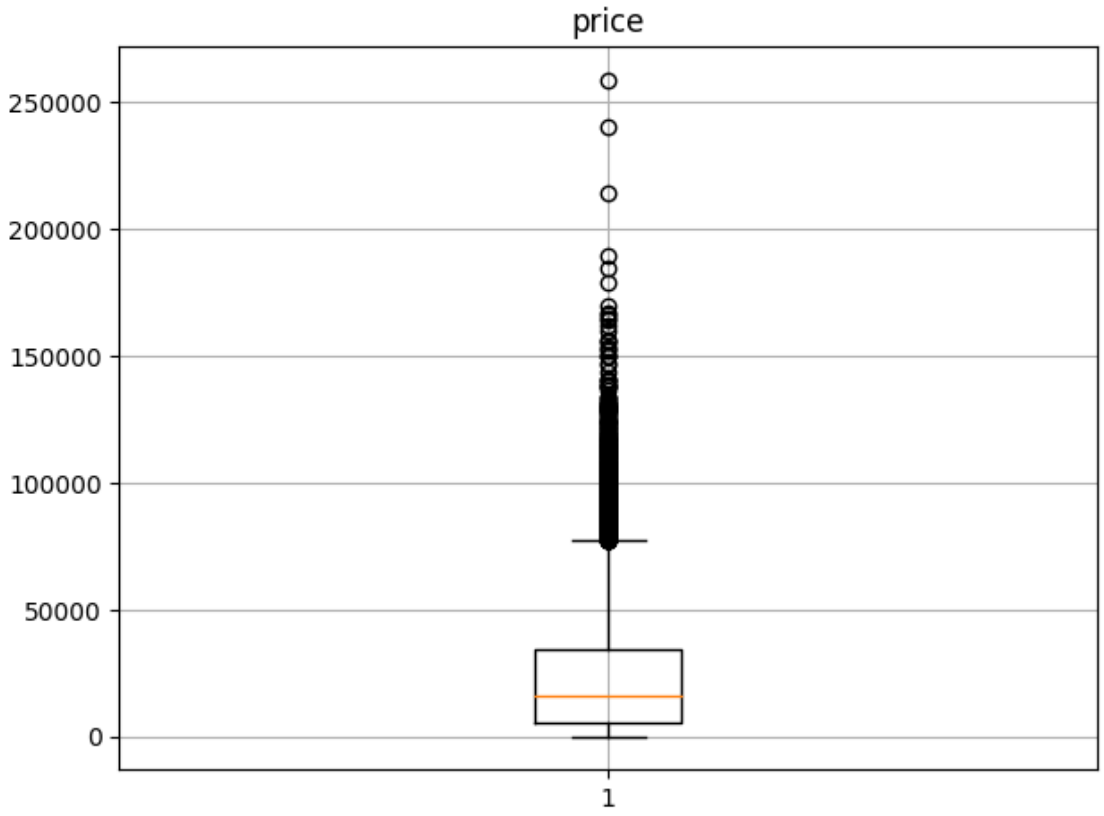
23900 rows × 10 columns

In [49]:

```
plt.scatter(x = np.log(df['price']), y = df['mileage'], s = 20, alpha = 0.

plt.title("price vs mileage")
plt.xlabel('Price')
plt.ylabel('Mileage')


plt.grid()

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



In [50]:
```
plt.boxplot(x= df['mileage'])

plt.title("price")

plt.grid()

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



In [ ]:

In [ ]:

In [ ]:

In [51]:
```
df.shape
```

Out[51]: (23900, 10)

In [52]:
```
df.corr(numeric_only = True)
```

Out[52]:

	year	price	mileage	tax	mpg	engineSize
year	1.000000	0.568400	-0.755742	0.017511	-0.077138	-0.092885
price	0.568400	1.000000	-0.569929	0.264374	-0.275865	0.481334
mileage	-0.755742	-0.569929	1.000000	-0.162643	0.143315	0.037657
tax	0.017511	0.264374	-0.162643	1.000000	-0.385382	0.377100
mpg	-0.077138	-0.275865	0.143315	-0.385382	1.000000	-0.349231
engineSize	-0.092885	0.481334	0.037657	0.377100	-0.349231	1.000000

In [53]:
```
y = df['price']
x = df[['mileage', 'tax',
       'mpg', 'engineSize']]
```

In [54]:
```
from sklearn.model_selection import train_test_split
```

In [55]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, r
```

In [56]:
```
from sklearn.linear_model import LinearRegression
```

In [57]:
```
lm = LinearRegression()
```

In [58]:
```
lm.fit(x_train, y_train)
```

Out[58]:
```
LinearRegression
LinearRegression()
```

In [59]:
```
print(lm.intercept_)
```

```
10620.398216688533
```

In [60]:
```
cdf =pd.DataFrame(lm.coef_, x.columns, columns  =['Coeff'])
cdf
```

Out[60]:

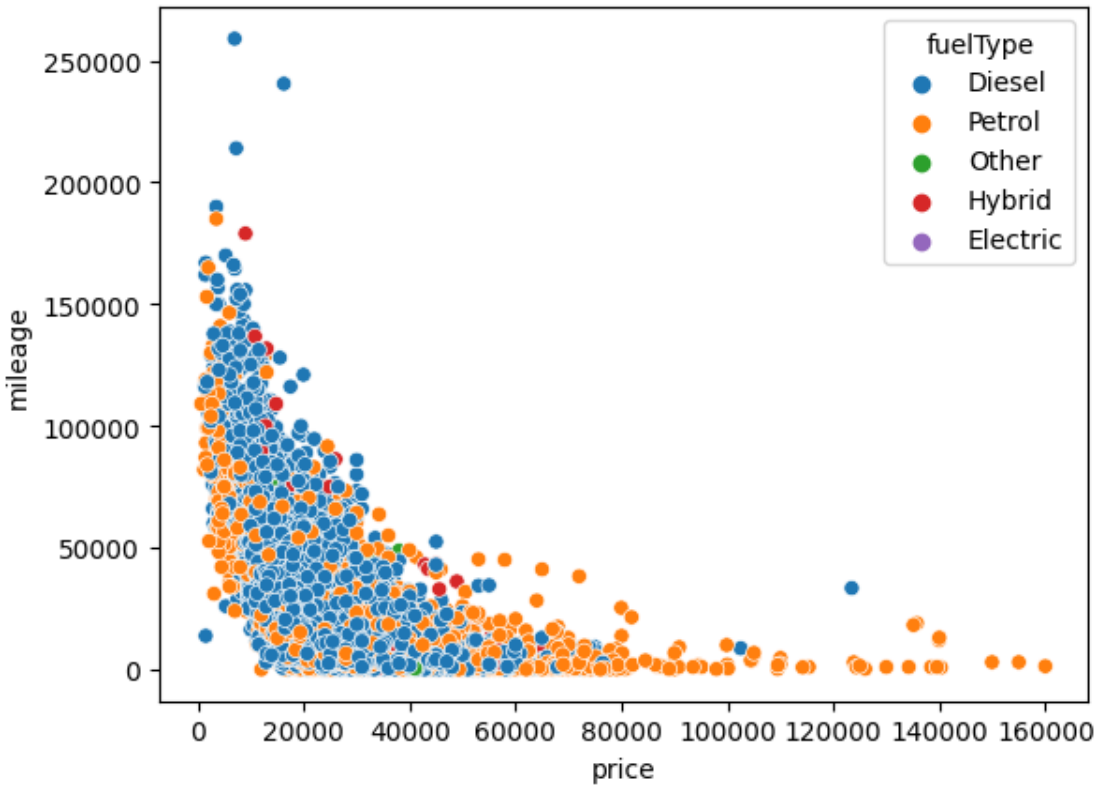
	Coeff
mileage	-0.300371
tax	-7.274084
mpg	-14.635744
engineSize	10428.660134

In [ ]:

In [ ]:

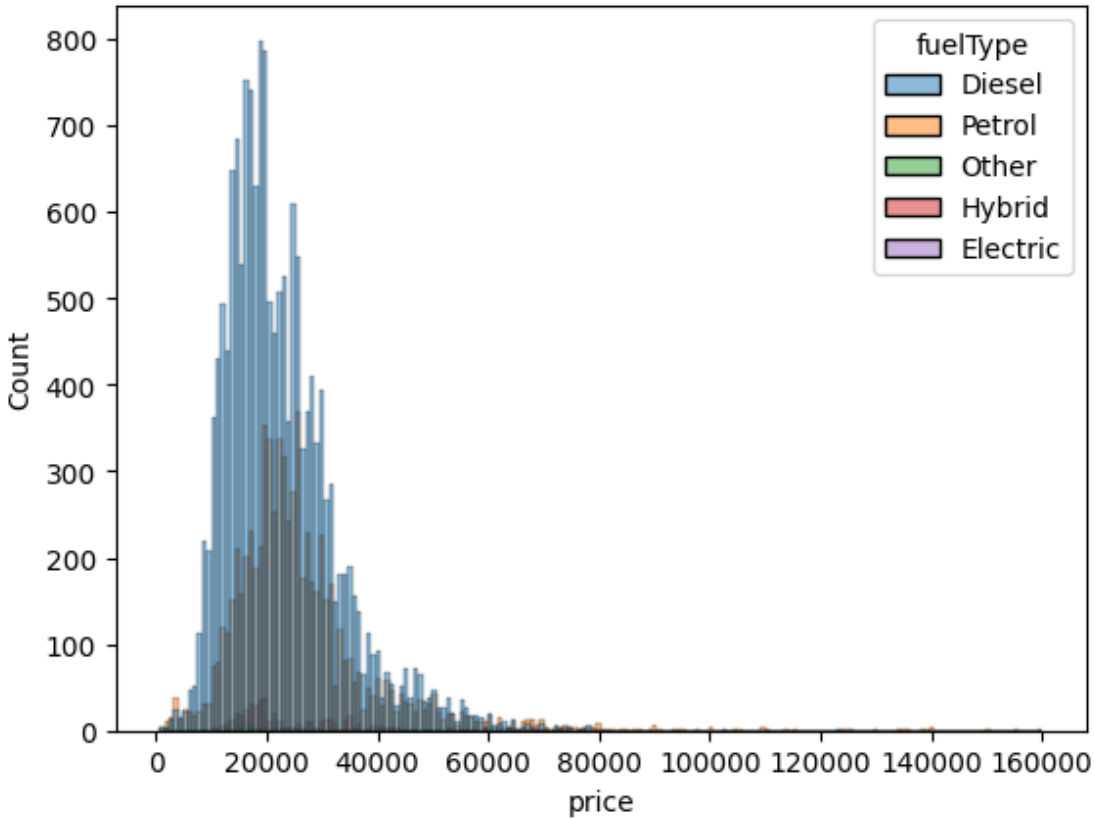
In [ ]:

In [61]:
```
sns.scatterplot(x = 'price', y = 'mileage', hue = 'fuelType', data=df)
plt.show()
```



In [62]:
```
sns.histplot(data=df, x = 'price', hue = 'fuelType', alpha=0.5)
```

Out[62]: <Axes: xlabel='price', ylabel='Count'>



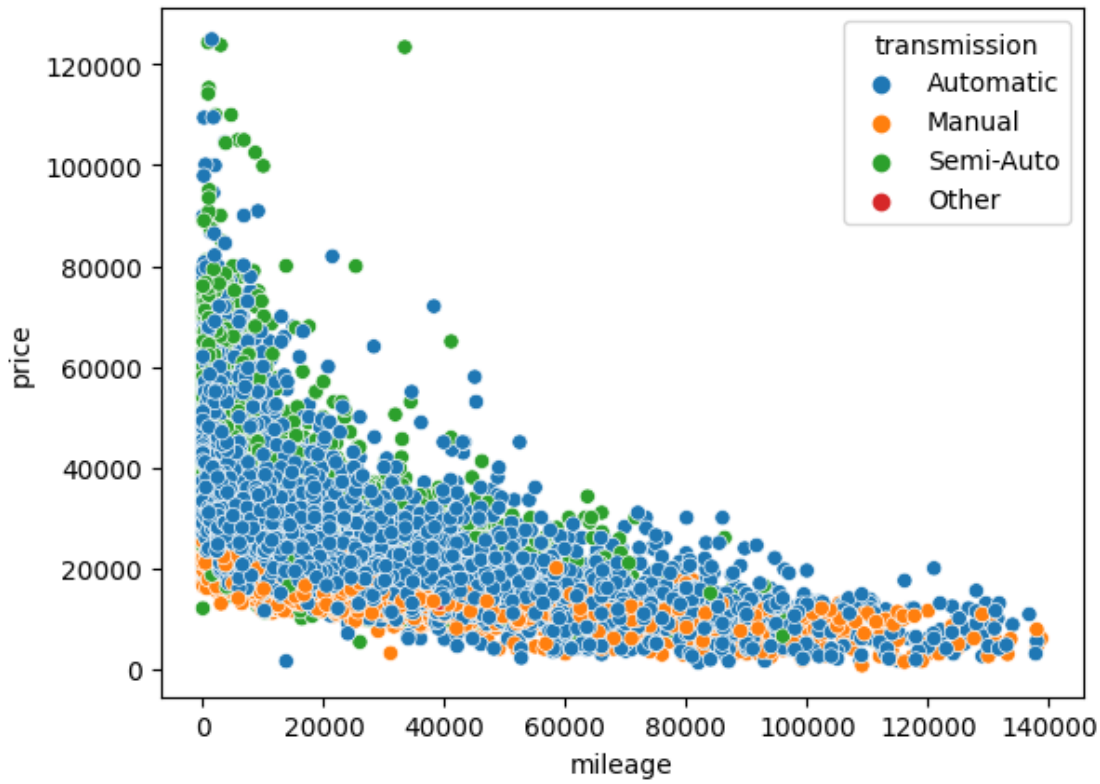
In [63]:
```
model1= sm.OLS.from_formula('price~mpg +mileage+tax', data= df)
```

In [64]:
```
result = model1.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
=====
Dep. Variable:                   price   R-squared:
0.375
Model:                             OLS   Adj. R-squared:
0.375
Method:                  Least Squares   F-statistic:
4783.
Date:                 Thu, 10 Aug 2023   Prob (F-statistic):
0.00
Time:                         16:14:38   Log-Likelihood:            -2.521
6e+05
No. Observations:               23900   AIC:                          5.04
3e+05
Df Residuals:                   23896   BIC:                          5.04
4e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
Intercept   3.146e+04    252.621    124.516      0.000     3.1e+04     3.
2e+04
mpg          -75.5270      2.723    -27.732      0.000     -80.865     -7
0.189
mileage       -0.2673      0.003   -101.574      0.000      -0.272      -
0.262
tax           21.8710      1.026     21.319      0.000      19.860      2
3.882
==============================================================================
=====
Omnibus:                    16577.113   Durbin-Watson:
1.698
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           41997
8.647
Skew:                           3.011   Prob(JB):
0.00
Kurtosis:                      22.633   Cond. No.                     1.3
9e+05
==============================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
[2] The condition number is large, 1.39e+05. This might indicate that the
re are
strong multicollinearity or other numerical problems.
```

In [65]:
```
filtered_data = df[(df['mileage'] < 140000) & (df['price'] < 125000)]
filtered_data['log_price'] = np.log(filtered_data['price'])
sns.scatterplot(data=filtered_data, x='mileage', y='price', hue='transmiss
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13928\2710468722.py:2: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  filtered_data['log_price'] = np.log(filtered_data['price'])
```



In [66]:
```
model2= sm.OLS.from_formula('log_price~mpg +mileage+tax+year+fuelType+tran
```

In [67]:

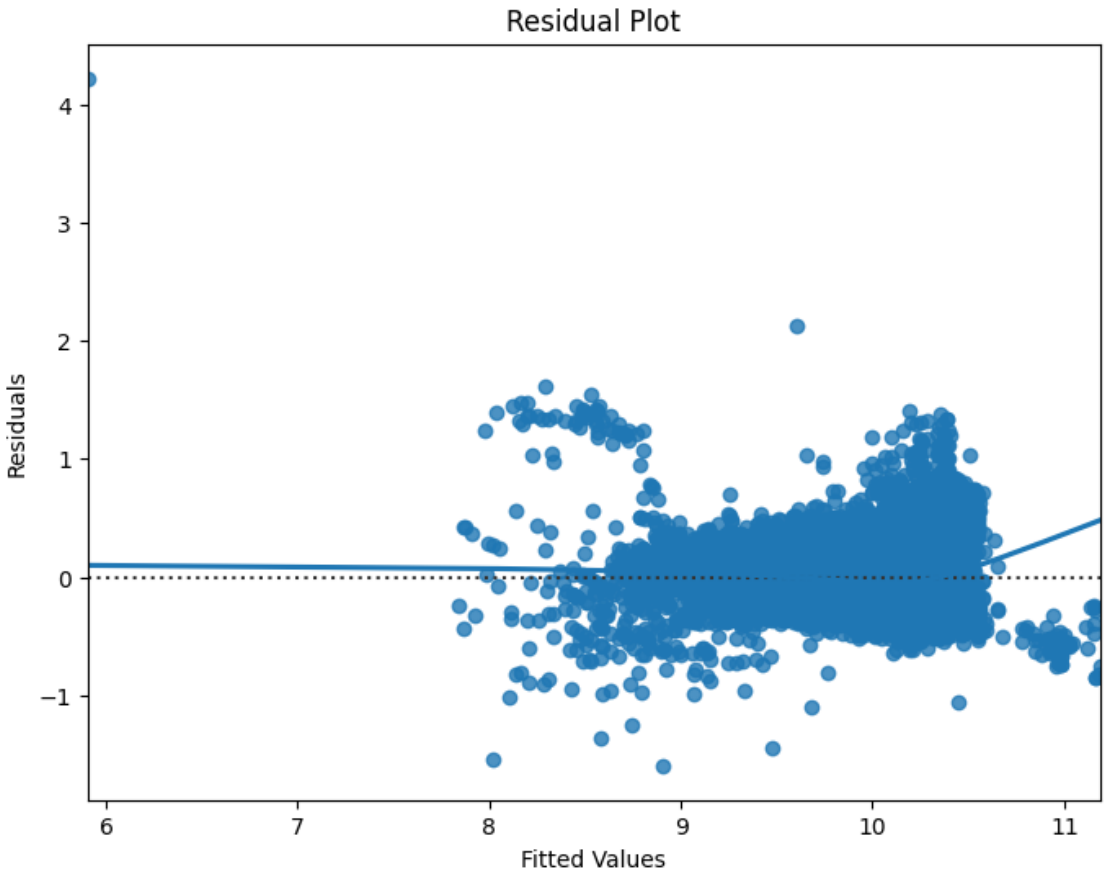
```
result2 = model2.fit()
print(result2.summary())
```

```
                            OLS Regression Results
==============================================================================
=====
Dep. Variable:              log_price   R-squared:
0.727
Model:                            OLS   Adj. R-squared:
0.727
Method:                 Least Squares   F-statistic:
5766.
Date:                Thu, 10 Aug 2023   Prob (F-statistic):
0.00
Time:                        16:14:40   Log-Likelihood:                3
77.36
No. Observations:               23857   AIC:                           -
730.7
Df Residuals:                   23845   BIC:                           -
633.8
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
====================
                                coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------------
Intercept                   -179.0559      2.154    -83.143      0.000
-183.277    -174.835
fuelType[T.Electric]          1.7985      0.143     12.608      0.000
1.519       2.078
fuelType[T.Hybrid]            0.5042      0.014     37.087      0.000
0.478       0.531
fuelType[T.Other]             0.6173      0.039     16.030      0.000
0.542       0.693
fuelType[T.Petrol]           -0.0987      0.004    -25.826      0.000
-0.106      -0.091
transmission[T.Manual]       -0.2529      0.005    -52.788      0.000
-0.262      -0.244
transmission[T.Other]        -0.4635      0.170     -2.733      0.006
-0.796      -0.131
transmission[T.Semi-Auto]     0.0018      0.004      0.496      0.620
-0.005       0.009
mpg                          -0.0044   9.26e-05    -47.653      0.000
-0.005      -0.004
mileage                   -5.903e-06    1.1e-07    -53.780      0.000   -
6.12e-06   -5.69e-06
tax                           0.0010   2.81e-05     35.253      0.000
0.001       0.001
year                          0.0939      0.001     88.027      0.000
0.092       0.096
==============================================================================
=====
Omnibus:                     7602.399   Durbin-Watson:
1.732
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            7277
4.743
Skew:                           1.255   Prob(JB):
0.00
```

```
Kurtosis:                       11.180   Cond. No.                         4.5
4e+07
==============================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
[2] The condition number is large, 4.54e+07. This might indicate that the
re are
strong multicollinearity or other numerical problems.
```

In [68]:

```
# Residual Analysis
fig, ax = plt.subplots(figsize=(8, 6))
sns.residplot(x=result2.fittedvalues, y=result2.resid, lowess=True)
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```

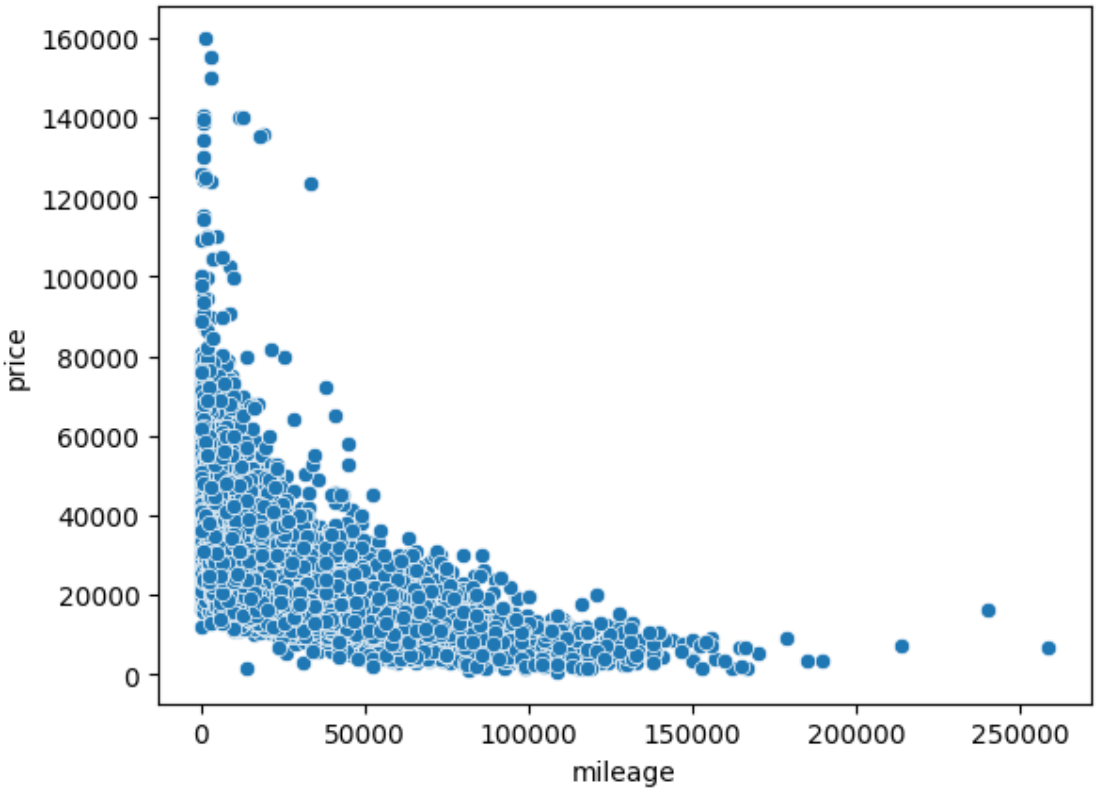


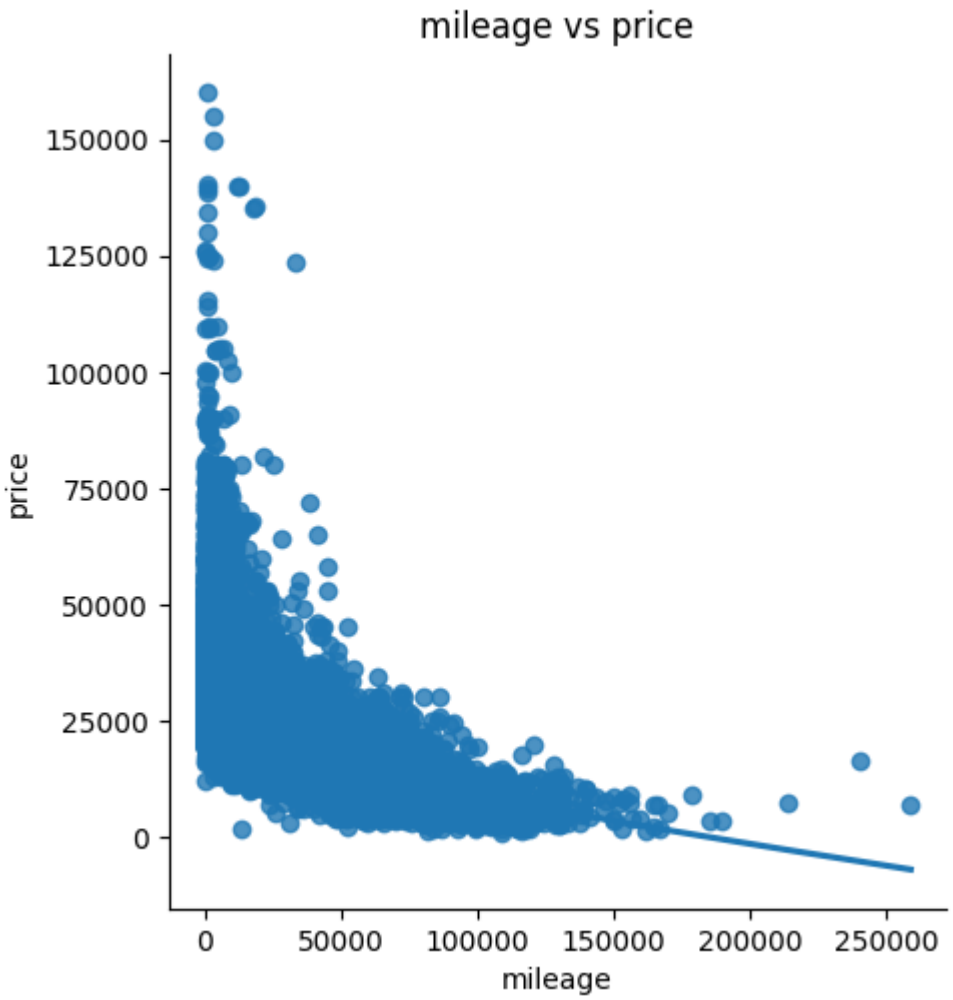
In [69]:
```
model3 = sm.OLS.from_formula('np.log(price) ~ mileage', data=df)
result3 = model3.fit()
print(result3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          np.log(price)   R-squared:                       0.513
Model:                            OLS   Adj. R-squared:                  0.513
Method:                 Least Squares   F-statistic:                 2.514e+04
Date:                Thu, 10 Aug 2023   Prob (F-statistic):               0.00
Time:                        16:16:37   Log-Likelihood:                -6790.9
No. Observations:               23900   AIC:                         1.359e+04
Df Residuals:                   23898   BIC:                         1.360e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.3098      0.003   3472.871      0.000      10.304      10.316
mileage    -1.426e-05      9e-08   -158.540      0.000   -1.44e-05   -1.41e-05
==============================================================================
Omnibus:                     1991.258   Durbin-Watson:                   1.647
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5562.808
Skew:                           0.463   Prob(JB):                         0.00
Kurtosis:                       5.174   Cond. No.                     4.71e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.71e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [70]:
```
sns.scatterplot(data=df, x='mileage', y='price')
sns.lmplot(data=df, x='mileage', y='price', lowess=True)
plt.title('mileage vs price')
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)
```





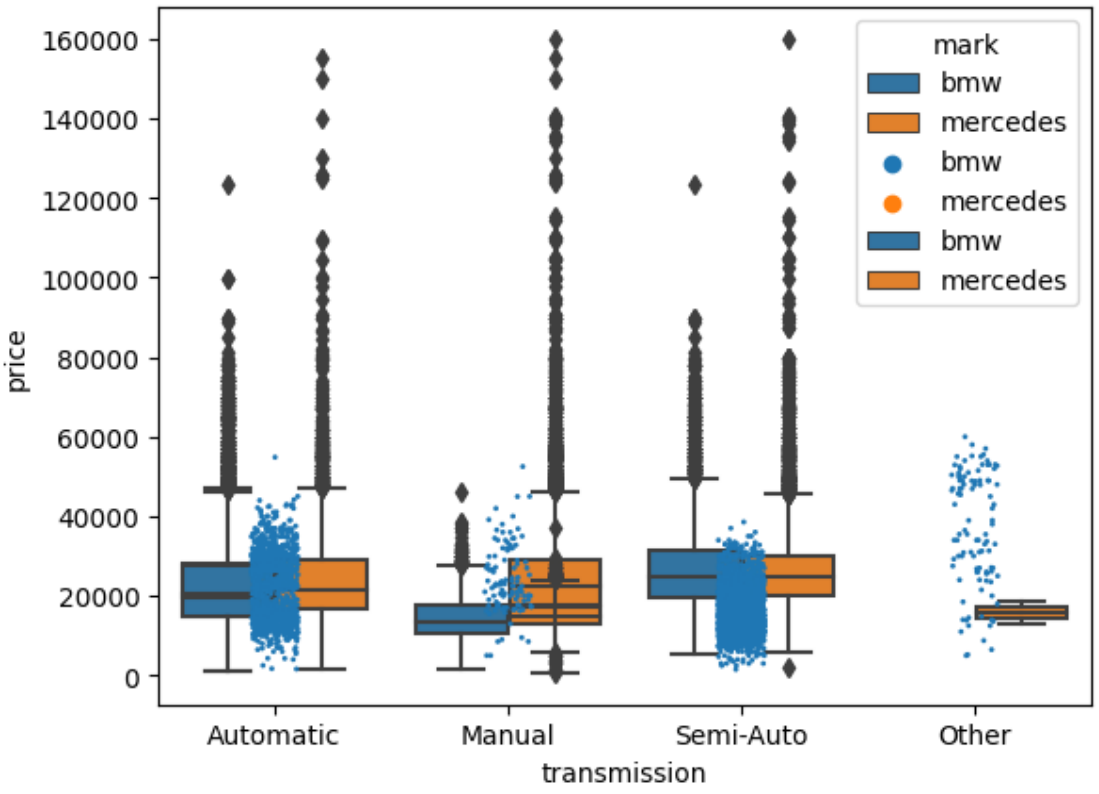
In [31]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
import scipy.stats as stats
```

In [32]:
```
lm_fit = sm.OLS(np.log(df['price']), sm.add_constant(df['mileage'])).fit()

print(lm_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                   price   R-squared:
0.513
Model:                             OLS   Adj. R-squared:
0.513
Method:                  Least Squares   F-statistic:
2.514e+04
Date:                 Thu, 10 Aug 2023   Prob (F-statistic):
0.00
Time:                         16:14:27   Log-Likelihood:
-6790.9
No. Observations:               23900   AIC:
1.359e+04
Df Residuals:                   23898   BIC:
1.360e+04
Df Model:                           1
Covariance Type:            nonrobust
```

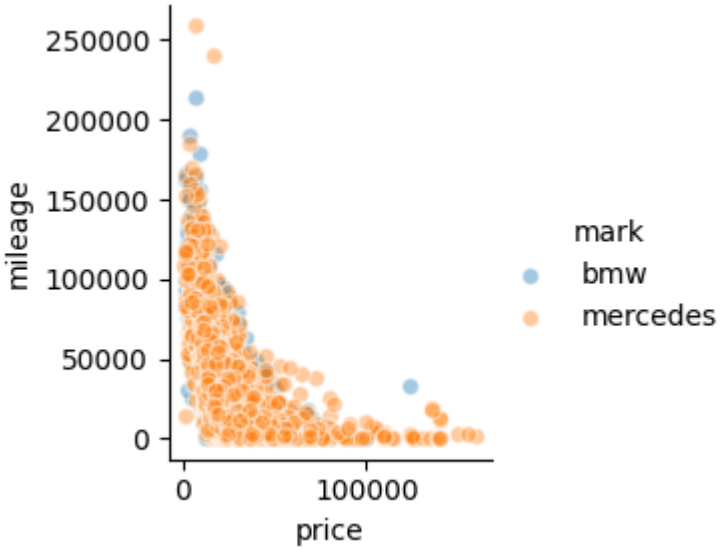
In [33]:
```
sns.boxplot(y='price', x='mark', data=df, hue = 'mark')
sns.stripplot(y='price', x='model', size = 2,hue ='mark', data=df, jitter=
sns.boxplot(y='price', x='transmission', data=df, hue = 'mark')
```

Out[33]: <Axes: xlabel='transmission', ylabel='price'>



In [34]:
```
graph = sns.FacetGrid(df,  hue ="mark")
# map the above form facetgrid with some attributes
graph.map(plt.scatter, "price", "mileage", alpha = 0.4,edgecolor ="w").add
# show the object
plt.show()
```

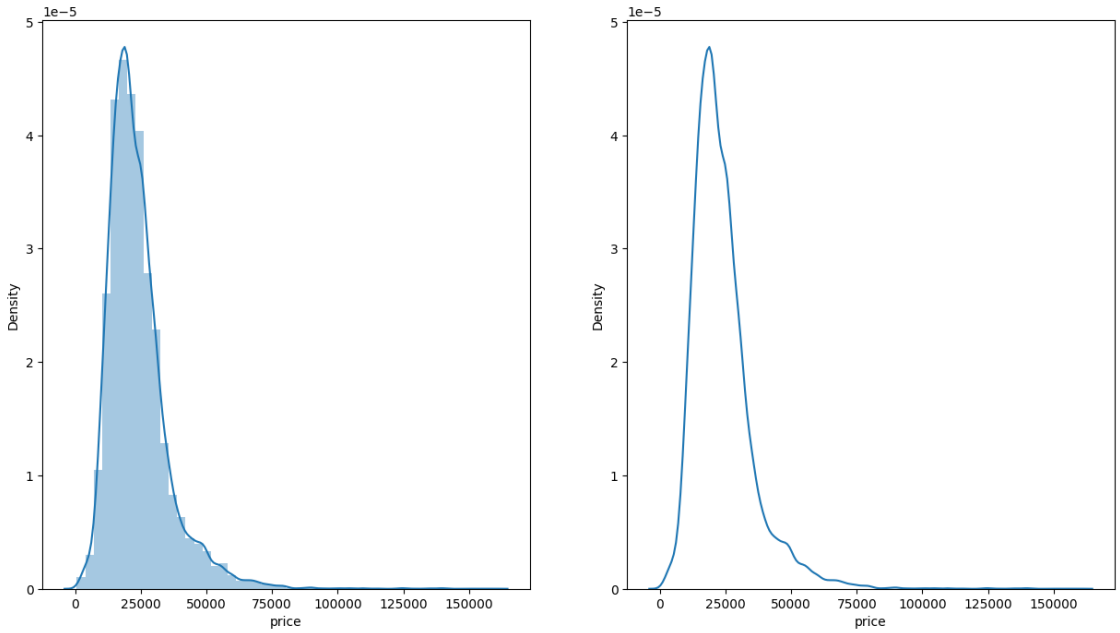
```
C:\Users\user\Anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [35]:
```
fig, axs = plt.subplots(1,2, figsize=[15,8])
sns.distplot(df['price'], ax=axs[0])
sns.kdeplot(df['price'])
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13928\942518258.py:2: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['price'], ax=axs[0])
```

Out[35]: <Axes: xlabel='price', ylabel='Density'>



In [36]:
```
df.describe()
```

Out[36]:

	year	price	mileage	tax	mpg	engineS
count	23900.000000	23900.000000	23900.000000	23900.000000	23900.000000	23900.0000
mean	2017.198243	23812.124435	23549.760544	130.752510	55.716632	2.1149
std	2.284145	11692.655891	23117.495727	63.600803	23.884771	0.5653
min	1970.000000	650.000000	1.000000	0.000000	1.100000	0.0000
25%	2016.000000	16250.000000	5841.000000	125.000000	45.600000	2.0000
50%	2017.000000	21498.000000	16382.500000	145.000000	54.300000	2.0000
75%	2019.000000	28488.000000	34400.000000	145.000000	64.200000	2.1000
max	2020.000000	159999.000000	259000.000000	580.000000	470.800000	6.6000

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [37]:
```
df1 = pd.read_csv('C:/Users/user/Desktop/My learning/ClinSoft/healthcare-d
df1
import statsmodels.formula.api as smf
```

In [38]:
```
x = df1[['age', 'hypertension', 'heart_disease', 'ever_married', 'work_typ
y  = df1[['stroke']]
```

In [39]:
```
log_reg_model = smf.logit('stroke ~ age + hypertension + heart_disease + e
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.138852
         Iterations: 35

C:\Users\user\Anaconda3\lib\site-packages\statsmodels\base\model.py:604:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [40]:
```
print(log_reg_model.summary())
```

```
                           Logit Regression Results
==============================================================================
=====
Dep. Variable:                 stroke   No. Observations:
4909
Model:                          Logit   Df Residuals:
4894
Method:                           MLE   Df Model:
14
Date:                Thu, 10 Aug 2023   Pseudo R-squ.:
0.2113
Time:                        16:14:32   Log-Likelihood:                -6
81.62
converged:                      False   LL-Null:                       -8
64.19
Covariance Type:            nonrobust   LLR p-value:                2.73
6e-69
==============================================================================
============================
                                        coef    std err          z      P
>|z|      [0.025      0.975]
------------------------------------------------------------------------------
----------------------------
Intercept                            -8.3236      0.663    -12.558
0.000      -9.623      -7.025
ever_married[T.Yes]                  -0.1159      0.247     -0.469
0.639      -0.600       0.368
work_type[T.Never_worked]            -6.8125    158.683     -0.043
0.966    -317.825     304.200
work_type[T.Private]                  0.1604      0.223      0.718
0.473      -0.277       0.598
work_type[T.Self-employed]           -0.2644      0.254     -1.039
0.299      -0.763       0.234
work_type[T.children]                 0.6804      1.114      0.611
0.541      -1.503       2.864
Residence_type[T.Urban]               0.0048      0.150      0.032
0.974      -0.289       0.299
smoking_status[T.formerly smoked]     0.2738      0.247      1.110
0.267      -0.210       0.757
smoking_status[T.never smoked]        0.2090      0.232      0.900
0.368      -0.246       0.664
smoking_status[T.smokes]              0.5884      0.266      2.210
0.027       0.066       1.110
age                                   0.0735      0.006     11.588
0.000       0.061       0.086
hypertension                          0.5249      0.175      2.999
0.003       0.182       0.868
heart_disease                         0.3467      0.206      1.683
0.092      -0.057       0.750
avg_glucose_level                     0.0046      0.001      3.597
0.000       0.002       0.007
bmi                                   0.0041      0.012      0.346
0.730      -0.019       0.027
==============================================================================
============================
```

In [ ]:

In [ ]: