In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score, confusion_matrix
```

In [2]:
```
df = pd.read_csv("C:/Users/user/Desktop/My learning/ClinSoft/merc.csv")
```

In [3]:
```
print(df.head())

# Summary statistics of the DataFrame
print(df.describe())
```

```
       model  year  price transmission  mileage fuelType  tax   mpg  \
0        SLK  2005   5200    Automatic    63000   Petrol  325  32.1
1    S Class  2017  34948    Automatic    27000   Hybrid   20  61.4
2   SL CLASS  2016  49948    Automatic     6200   Petrol  555  28.0
3    G Class  2016  61948    Automatic    16000   Petrol  325  30.4
4    G Class  2016  73948    Automatic     4000   Petrol  325  30.1

   engineSize
0         1.8
1         2.1
2         5.5
3         4.0
4         4.0
                year          price        mileage           tax
mpg  \
count  13119.000000   13119.000000   13119.000000  13119.000000  13119.00
0000
mean    2017.296288   24698.596920   21949.559037    129.972178     55.15
5843
std        2.224709   11842.675542   21176.512267     65.260286     15.22
0082
min     1970.000000     650.000000       1.000000      0.000000      1.10
0000
25%     2016.000000   17450.000000    6097.500000    125.000000     45.60
0000
50%     2018.000000   22480.000000   15189.000000    145.000000     56.50
0000
75%     2019.000000   28980.000000   31779.500000    145.000000     64.20
0000
max     2020.000000  159999.000000  259000.000000    580.000000    217.30
0000

         engineSize
count  13119.000000
mean       2.071530
std        0.572426
min        0.000000
25%        1.800000
50%        2.000000
75%        2.100000
max        6.200000
```

In [4]:

```
# Fit a linear regression model
lm_fit = sm.OLS(df['price'], sm.add_constant(df['mileage'])).fit()

# Print the summary of the linear regression
print(lm_fit.summary())

# Make predictions using the fitted model
mileage_values = np.array([5, 10, 20])
mileage_values_with_const = sm.add_constant(mileage_values)
predicted_values = lm_fit.predict(mileage_values_with_const)

print(predicted_values)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.289
Model:                            OLS   Adj. R-squared:                  0.289
Method:                 Least Squares   F-statistic:                     5321.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        12:43:00   Log-Likelihood:            -1.394e+05
No. Observations:               13119   AIC:                         2.789e+05
Df Residuals:                   13117   BIC:                         2.789e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.129e+04    125.609    249.129      0.000     3.1e+04     3.15e+04
mileage       -0.3004      0.004    -72.947      0.000      -0.309      -0.292
==============================================================================
Omnibus:                    10307.239   Durbin-Watson:                   1.658
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           359723.405
Skew:                           3.503   Prob(JB):                         0.00
Kurtosis:                      27.678   Cond. No.                     4.39e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.39e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
[31291.39352541 31289.89137707 31286.88708038]
```

In [ ]:

In [ ]:

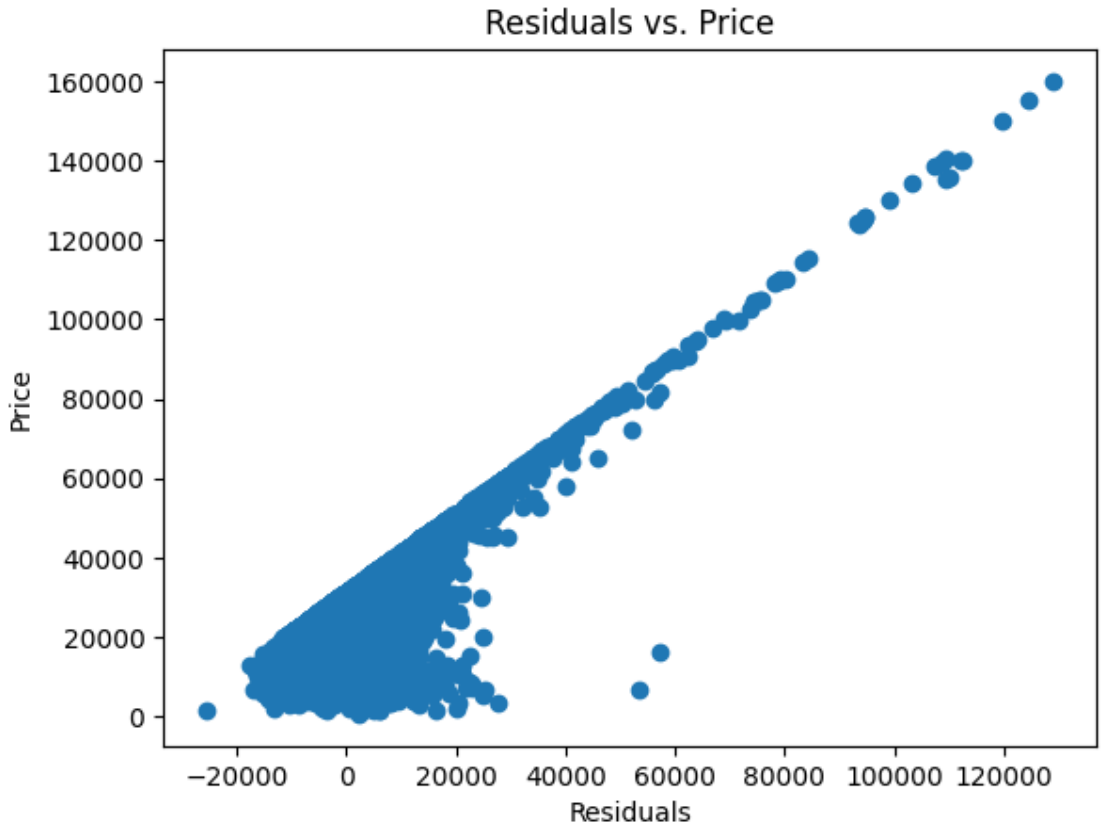
In [7]:

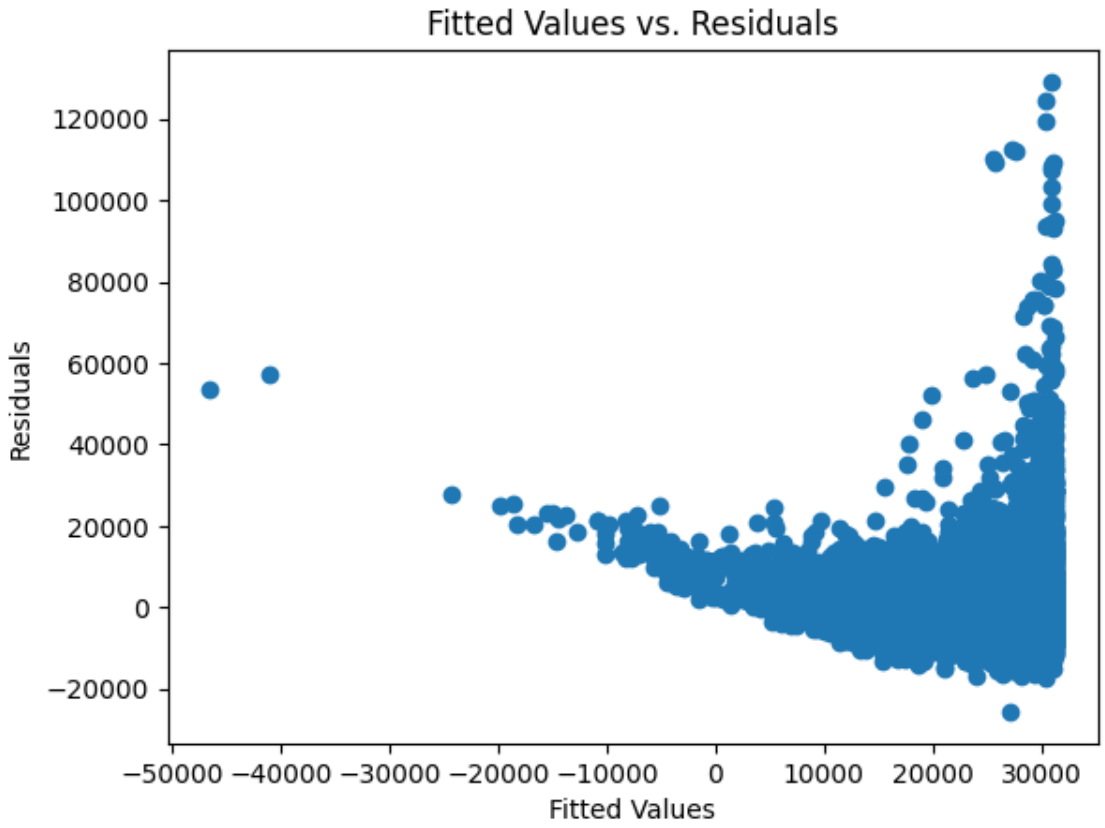
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Load the dataset
df = pd.read_csv("C:/Users/user/Desktop/My learning/ClinSoft/merc.csv")

# Linear regression
X = df['mileage']
y = df['price']

# Fit a linear regression model
X = sm.add_constant(X)
lm_fit = sm.OLS(y, X).fit()

# Print the summary of the linear regression
print(lm_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.289
Model:                            OLS   Adj. R-squared:                  0.289
Method:                 Least Squares   F-statistic:                     5321.
Date:                Sun, 06 Aug 2023   Prob (F-statistic):               0.00
Time:                        12:52:47   Log-Likelihood:            -1.394e+05
No. Observations:               13119   AIC:                         2.789e+05
Df Residuals:                   13117   BIC:                         2.789e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.129e+04    125.609    249.129      0.000     3.1e+04     3.15e+04
mileage       -0.3004      0.004    -72.947      0.000      -0.309      -0.292
==============================================================================
Omnibus:                    10307.239   Durbin-Watson:                   1.658
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           359723.405
Skew:                           3.503   Prob(JB):                         0.00
Kurtosis:                      27.678   Cond. No.                     4.39e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.39e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

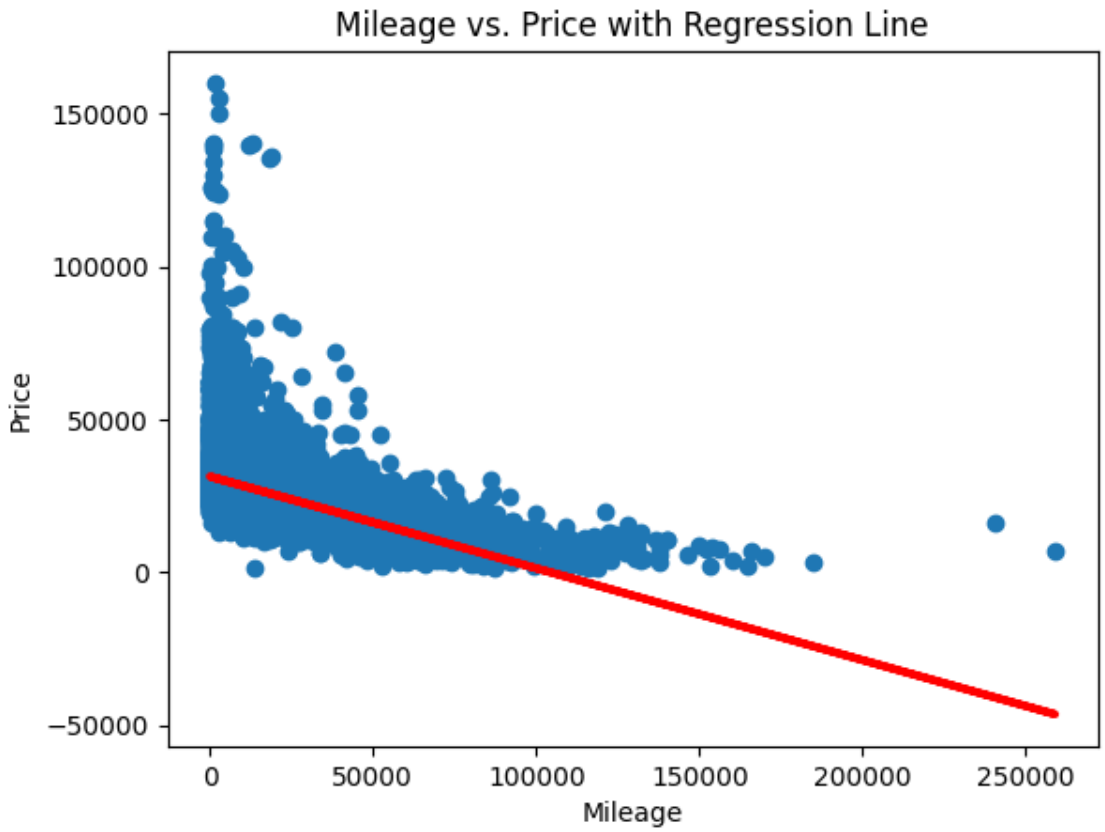
In [8]:

```
# Plot residuals vs. price
plt.scatter(lm_fit.resid, y)
plt.xlabel('Residuals')
plt.ylabel('Price')
plt.title('Residuals vs. Price')
plt.show()

# Plot fitted values vs. residuals
plt.scatter(lm_fit.fittedvalues, lm_fit.resid)
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Fitted Values vs. Residuals')
plt.show()

# Make predictions using the fitted model
mileage_values = np.array([5, 10, 20])
mileage_values_with_const = sm.add_constant(mileage_values)
predicted_values = lm_fit.predict(mileage_values_with_const)

print(predicted_values)

# Plot mileage vs. price with regression line
plt.scatter(df['mileage'], y)
plt.plot(df['mileage'], lm_fit.fittedvalues, color='red', linewidth=3)
plt.xlabel('Mileage')
plt.ylabel('Price')
plt.title('Mileage vs. Price with Regression Line')
plt.show()
```






```
[31291.39352541 31289.89137707 31286.88708038]
```



In [ ]: