In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as skl
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from scipy.stats import shapiro
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
data = pd.read_csv(r"D:\placement\projects datasets and codes\automobiles\automo
```

In [ ]:
```
data
```

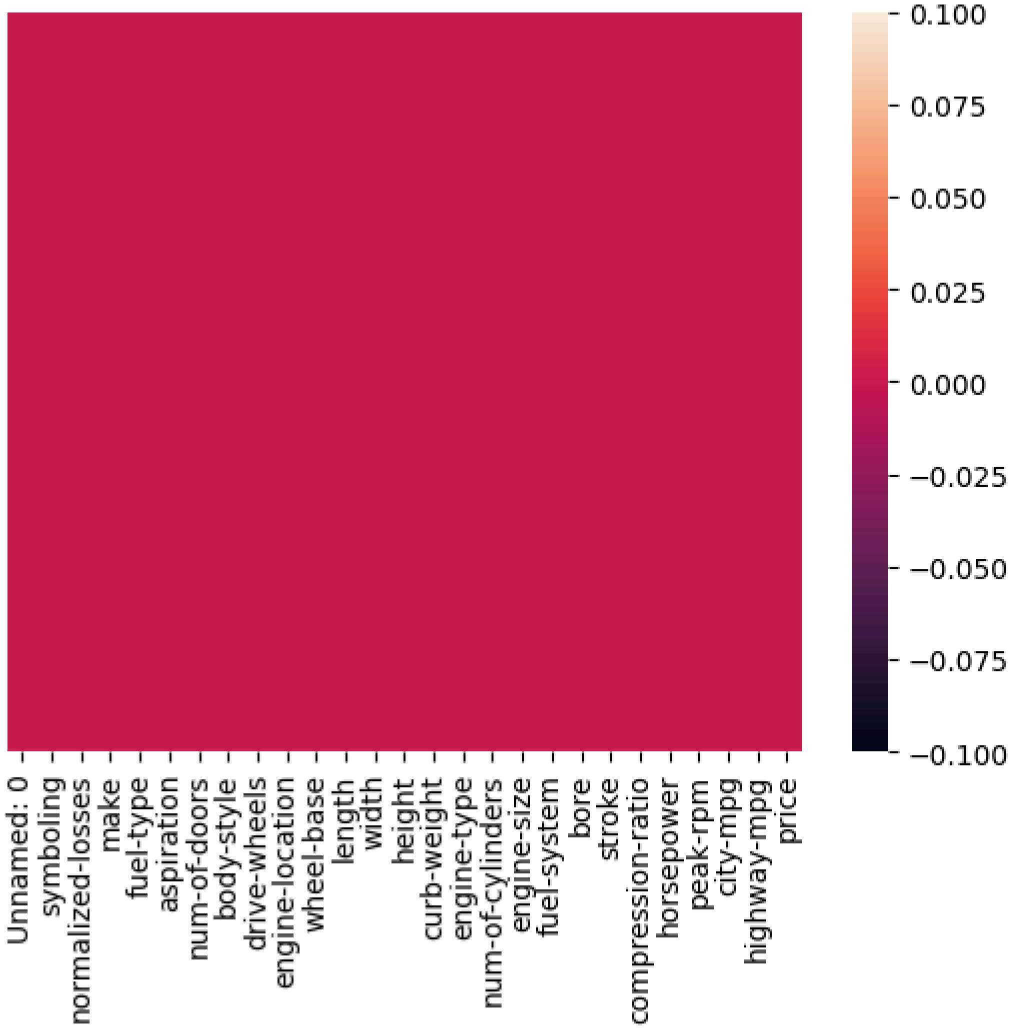
Out[ ]:

	Unnamed: 0	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style
0	0	3	?	alfa-romero	gas	std	two	convertible
1	1	3	?	alfa-romero	gas	std	two	convertible
2	2	1	?	alfa-romero	gas	std	two	hatchback
3	3	2	164	audi	gas	std	four	sedan
4	4	2	164	audi	gas	std	four	sedan
...	...	...	...	...	...	...	...	...
200	200	-1	95	volvo	gas	std	four	sedan
201	201	-1	95	volvo	gas	turbo	four	sedan
202	202	-1	95	volvo	gas	std	four	sedan
203	203	-1	95	volvo	diesel	turbo	four	sedan
204	204	-1	95	volvo	gas	turbo	four	sedan

205 rows × 27 columns

In [ ]:
```
sns.heatmap(data.isnull() , yticklabels = False)
```

Out[ ]: <Axes: >

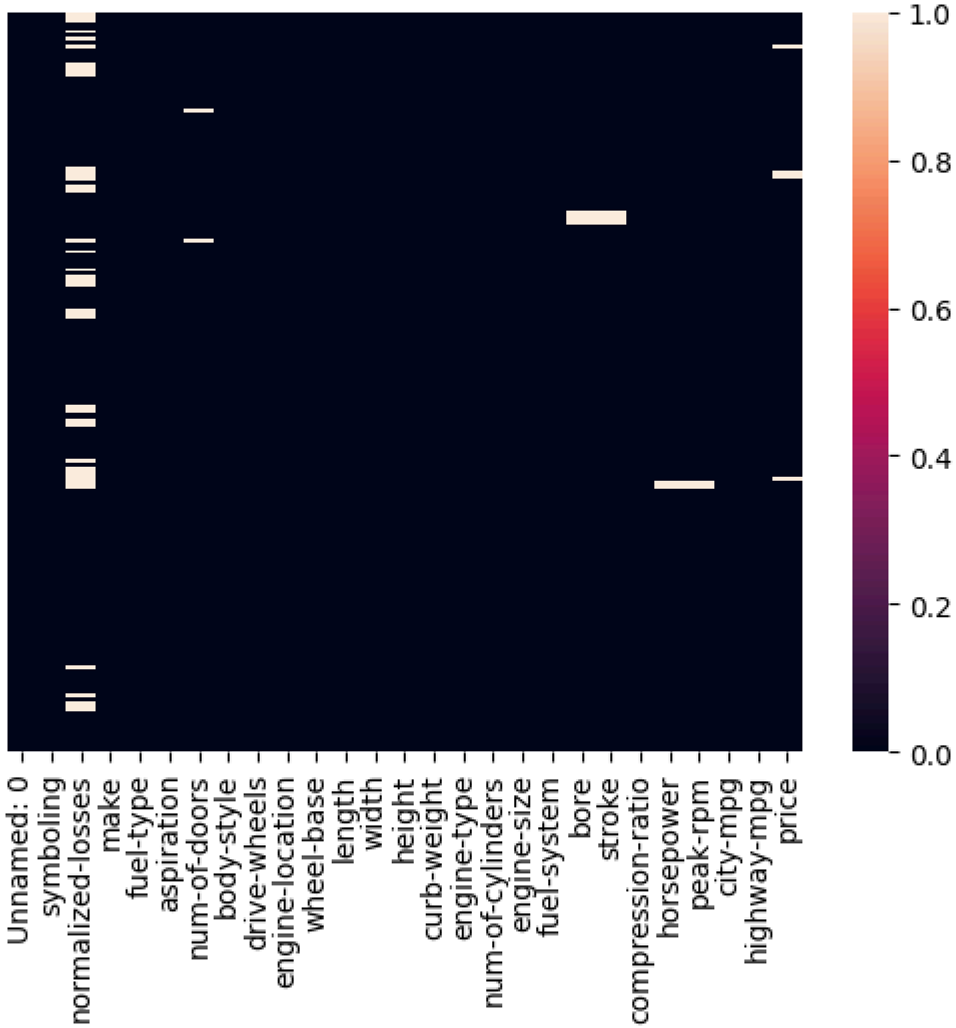


In [ ]:
```
def convert_null(val):
    if val == "?":
        return np.nan
    return val
```

In [ ]:
```
for col in data.columns:
    data[col] = data[col].apply(convert_null)
```

In [ ]:
```
sns.heatmap(data.isnull() , yticklabels = False)
```

Out[ ]: <Axes: >



In [ ]:
```
data.drop(columns=["normalized-losses"],inplace=True)
```

In [ ]:
```
data
```

Out[ ]:

	Unnamed: 0	symboling	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engi loca
0	0	3	alfa-romero	gas	std	two	convertible	rwd	f
1	1	3	alfa-romero	gas	std	two	convertible	rwd	f
2	2	1	alfa-romero	gas	std	two	hatchback	rwd	f
3	3	2	audi	gas	std	four	sedan	fwd	f
4	4	2	audi	gas	std	four	sedan	4wd	f
...	...	...	...	...	...	...	...	...	
200	200	-1	volvo	gas	std	four	sedan	rwd	f
201	201	-1	volvo	gas	turbo	four	sedan	rwd	f
202	202	-1	volvo	gas	std	four	sedan	rwd	f
203	203	-1	volvo	diesel	turbo	four	sedan	rwd	f
204	204	-1	volvo	gas	turbo	four	sedan	rwd	f

205 rows × 26 columns

In [ ]:

```
data.dropna(inplace=True)
```

In [ ]:

```
data.drop("Unnamed: 0",axis = 1 , inplace = True)
```

In [ ]:

```
def eda_summary(df):
    """
    Perform EDA for each variable in the DataFrame.
    """
    for col in df.columns:
        print(f"\n--- EDA for '{col}' ---")
        print(f"Data Type: {df[col].dtype}")
        print(f"Number of unique values: {df[col].nunique()}")
        print(f"Number of missing values: {df[col].isna().sum()}")
        print(f"Description:\n{df[col].describe()}")

        # Visualization
        plt.figure(figsize=(10, 4))

        if df[col].dtype == 'object':  # Categorical
            sns.countplot(data=df, x=col)
            plt.xticks(rotation=45)
            plt.title(f"Distribution of {col}")
        else:  # Numerical
            sns.histplot(df[col], kde=True)
            plt.title(f"Distribution of {col}")

        plt.show()
```

In [ ]:
```
data.describe()
```

Out[ ]:

	symboling	wheel-base	length	width	height	curb-weight	eng
count	193.000000	193.000000	193.000000	193.000000	193.000000	193.000000	193.00(
mean	0.797927	98.923834	174.326425	65.893782	53.869948	2561.507772	128.124
std	1.235582	6.152409	12.478593	2.137795	2.394770	526.700026	41.59(
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.00(
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	98.00(
50%	1.000000	97.000000	173.200000	65.400000	54.100000	2414.000000	120.00(
75%	2.000000	102.400000	184.600000	66.900000	55.700000	2952.000000	146.00(
max	3.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.00(

In [ ]:
```
data['price'] = data['price'].apply(int)
data['peak-rpm'] = data['peak-rpm'].apply(int)
data['horsepower'] = data['horsepower'].apply(int)
data['stroke'] = data['stroke'].apply(float)
data['bore'] = data['bore'].apply(float)
```

outliers by zscore

In [ ]:
```
mu  = np.mean(data['price'])
sig = np.var(data['price'])

def sec(val):
    return (val - mu)/np.sqrt(sig)
zscore = data['price'].apply(sec)
```

In [ ]:
```
zscore = list(zscore)
```

In [ ]:
```
for i in range(len(zscore)):
    if abs(zscore[i]) > 3:
        print([i , zscore[i]])
```

```
[15, 3.4741730819282663]
[64, 3.4301726329809763]
[65, 3.9804881071386315]
```

In [ ]:
```
data.iloc[[15 , 64 , 65]]
```

Out[ ]:

	symboling	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
16	0	bmw	gas	std	two	sedan	rwd	front	103.5
73	0	mercedes-benz	gas	std	four	sedan	rwd	front	120.9
74	1	mercedes-benz	gas	std	two	hardtop	rwd	front	112.0

3 rows × 25 columns

In [ ]:
```
prices_to_remove = [40960, 41315, 45400]
data = data[~data['price'].isin(prices_to_remove)]
```

In [ ]:
```
data.describe()
```

Out[ ]:

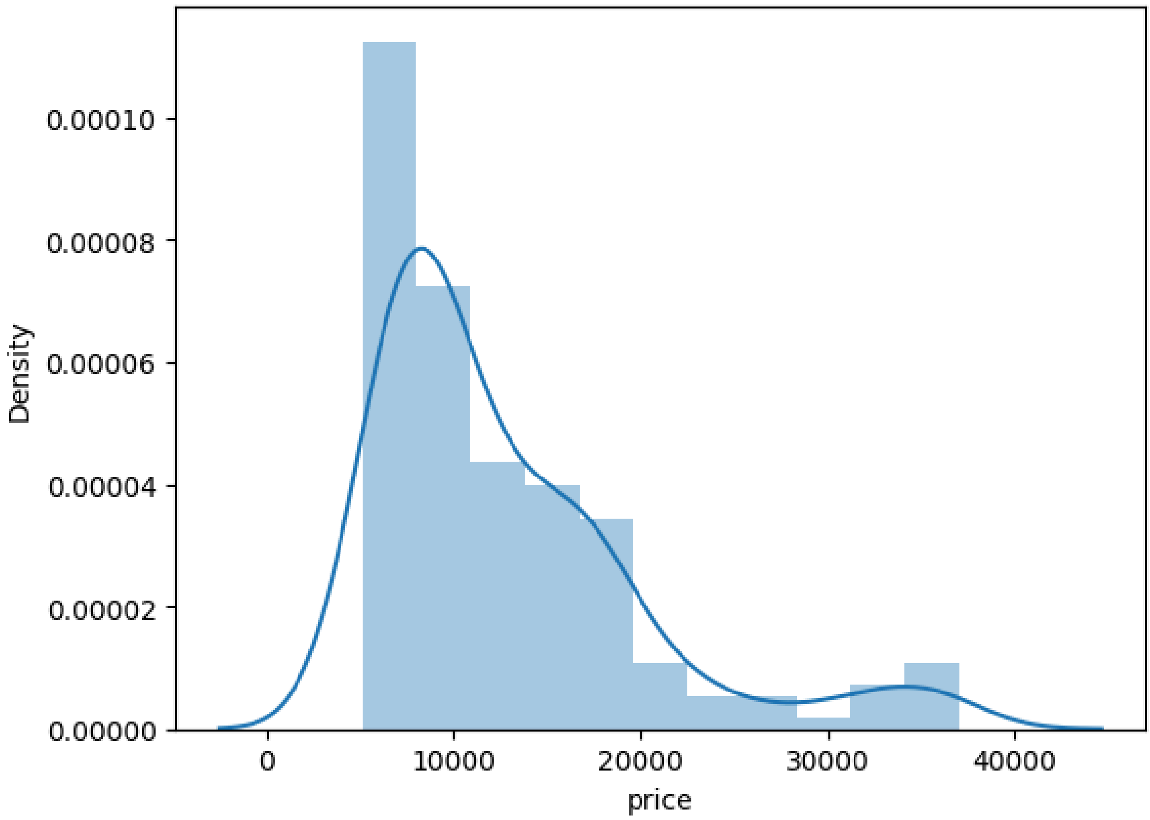
	symboling	wheel-base	length	width	height	curb-weight	eng
count	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	190.00(
mean	0.805263	98.715263	173.915263	65.820526	53.847895	2544.084211	125.82(
std	1.242533	5.902410	12.112053	2.059216	2.402197	511.319305	37.18
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.00(
25%	0.000000	94.500000	166.300000	64.025000	52.000000	2141.250000	98.00(
50%	1.000000	97.000000	173.200000	65.400000	54.100000	2412.000000	119.50(
75%	2.000000	102.400000	183.400000	66.500000	55.675000	2924.750000	141.00(
max	3.000000	115.600000	202.600000	71.700000	59.800000	4066.000000	326.00(

In [ ]:
```
data.shape
```

Out[ ]: (190, 25)

In [ ]:
```
sns.distplot(data['price'])
```

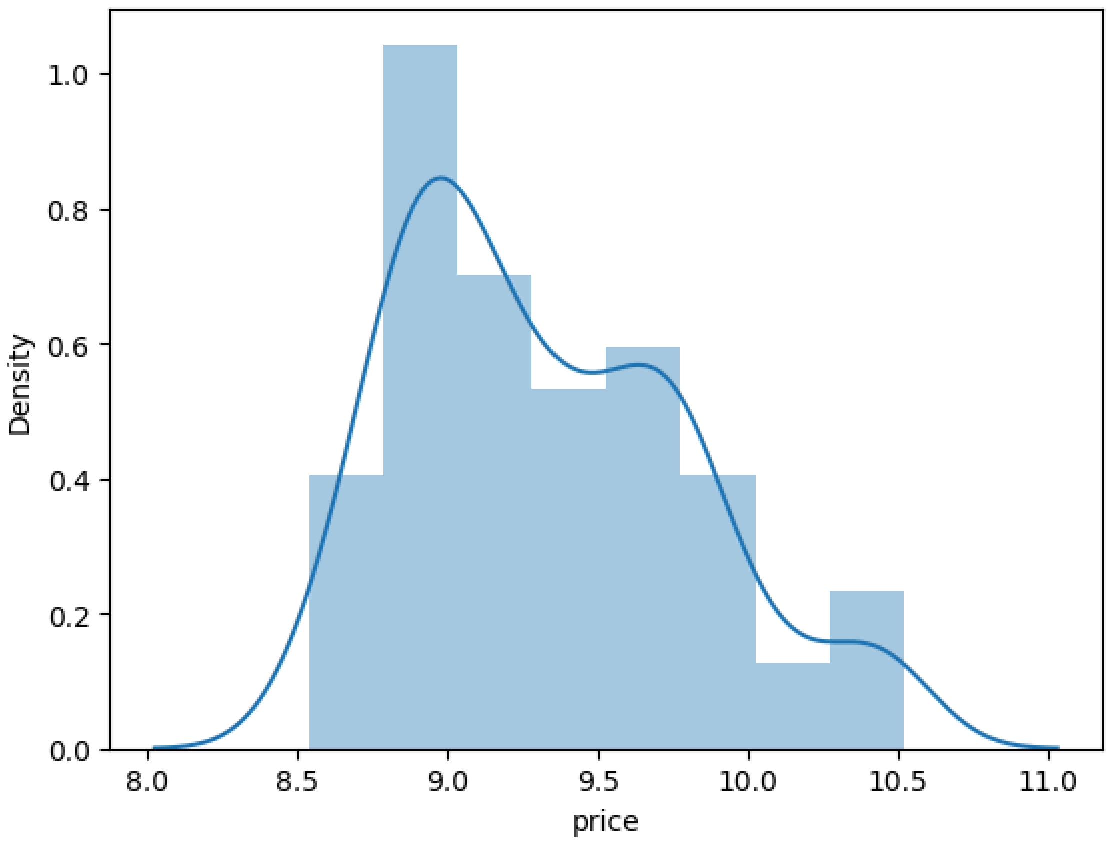
Out[ ]: <Axes: xlabel='price', ylabel='Density'>



In [ ]:
```
log_y = data['price'].apply(np.log)
```

In [ ]:
```
sns.distplot(log_y)
```

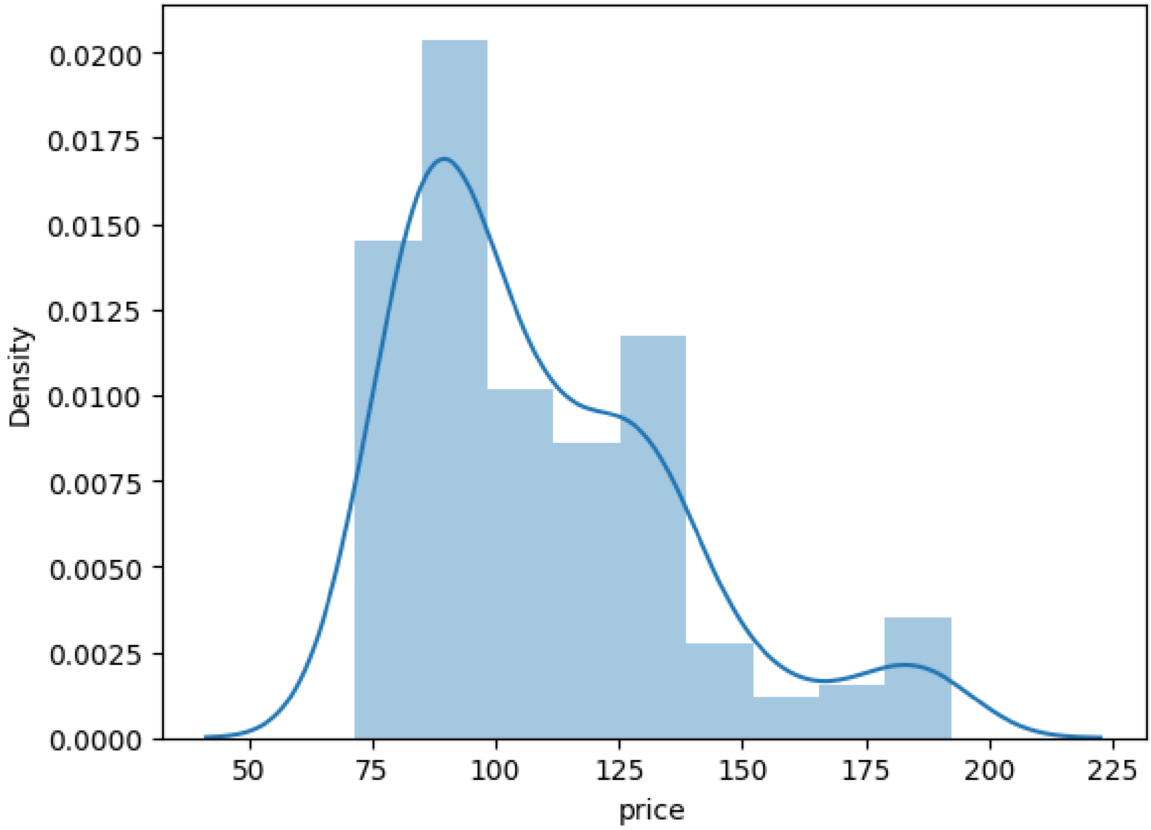
Out[ ]: <Axes: xlabel='price', ylabel='Density'>



In [ ]:
```
root_y = data['price'].apply(np.sqrt)
```

In [ ]:
```
sns.distplot(root_y)
```

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



In [ ]:
```
print(f"logy{shapiro(log_y)}")
print(f"Root Y{shapiro(root_y)}")
shapiro(data['price'])
```

```
logyShapiroResult(statistic=np.float64(0.9499400511989339), pvalue=np.float64(3.2
130411930727824e-06))
Root YShapiroResult(statistic=np.float64(0.8982722488925792), pvalue=np.float64
(4.1492832892444547e-10))
```

Out[ ]:
```
ShapiroResult(statistic=np.float64(0.8211612291256641), pvalue=np.float64(5.274
4354728519726e-14))
```

In [ ]:
```
data.columns
```

Out[ ]:
```
Index(['symboling', 'make', 'fuel-type', 'aspiration', 'num-of-doors',
       'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length',
       'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders',
       'engine-size', 'fuel-system', 'bore', 'stroke', 'compression-ratio',
       'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price'],
      dtype='object')
```

In [ ]:
```
data.describe()
```

Out[ ]:

	symboling	wheel-base	length	width	height	curb-weight	eng
count	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	190.00(
mean	0.805263	98.715263	173.915263	65.820526	53.847895	2544.084211	125.826
std	1.242533	5.902410	12.112053	2.059216	2.402197	511.319305	37.18'
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.00(
25%	0.000000	94.500000	166.300000	64.025000	52.000000	2141.250000	98.00(
50%	1.000000	97.000000	173.200000	65.400000	54.100000	2412.000000	119.50(
75%	2.000000	102.400000	183.400000	66.500000	55.675000	2924.750000	141.00(
max	3.000000	115.600000	202.600000	71.700000	59.800000	4066.000000	326.00(

In [ ]:
```
data.columns
```

Out[ ]:
```
Index(['symboling', 'make', 'fuel-type', 'aspiration', 'num-of-doors',
       'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length',
       'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders',
       'engine-size', 'fuel-system', 'bore', 'stroke', 'compression-ratio',
       'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price'],
      dtype='object')
```

In [ ]:
```
data['aspiration'].value_counts()
```

Out[ ]:
```
aspiration
std      155
turbo     35
Name: count, dtype: int64
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 190 entries, 0 to 204
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          190 non-null    int64  
 1   make               190 non-null    object 
 2   fuel-type          190 non-null    object 
 3   aspiration         190 non-null    object 
 4   num-of-doors       190 non-null    object 
 5   body-style         190 non-null    object 
 6   drive-wheels       190 non-null    object 
 7   engine-location    190 non-null    object 
 8   wheel-base         190 non-null    float64
 9   length             190 non-null    float64
 10  width              190 non-null    float64
 11  height             190 non-null    float64
 12  curb-weight        190 non-null    int64  
 13  engine-type        190 non-null    object 
 14  num-of-cylinders   190 non-null    object 
 15  engine-size        190 non-null    int64  
 16  fuel-system        190 non-null    object 
 17  bore               190 non-null    float64
 18  stroke             190 non-null    float64
 19  compression-ratio  190 non-null    float64
 20  horsepower         190 non-null    int64  
 21  peak-rpm           190 non-null    int64  
 22  city-mpg           190 non-null    int64  
 23  highway-mpg        190 non-null    int64  
 24  price              190 non-null    int64  
dtypes: float64(7), int64(8), object(10)
memory usage: 38.6+ KB
```

In [ ]:
```
df_encoded = pd.get_dummies(data, drop_first=True).astype(int)
```

In [ ]:
```
df_encoded.describe()
```

Out[ ]:

	symboling	wheel-base	length	width	height	curb-weight	eng
count	190.000000	190.000000	190.000000	190.000000	190.000000	190.000000	190.00
mean	0.805263	98.252632	173.415789	65.336842	53.378947	2544.084211	125.82
std	1.242533	5.956577	12.015896	2.070762	2.432926	511.319305	37.18
min	-2.000000	86.000000	141.000000	60.000000	47.000000	1488.000000	61.00
25%	0.000000	94.000000	166.000000	64.000000	52.000000	2141.250000	98.00
50%	1.000000	97.000000	173.000000	65.000000	54.000000	2412.000000	119.50
75%	2.000000	102.000000	183.000000	66.000000	55.000000	2924.750000	141.00
max	3.000000	115.000000	202.000000	71.000000	59.000000	4066.000000	326.00

8 rows × 60 columns

In [ ]:
```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 190 entries, 0 to 204
Data columns (total 60 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   symboling               190 non-null    int64
 1   wheel-base              190 non-null    int64
 2   length                  190 non-null    int64
 3   width                   190 non-null    int64
 4   height                  190 non-null    int64
 5   curb-weight             190 non-null    int64
 6   engine-size             190 non-null    int64
 7   bore                    190 non-null    int64
 8   stroke                  190 non-null    int64
 9   compression-ratio       190 non-null    int64
 10  horsepower              190 non-null    int64
 11  peak-rpm                190 non-null    int64
 12  city-mpg                190 non-null    int64
 13  highway-mpg             190 non-null    int64
 14  price                   190 non-null    int64
 15  make_audi               190 non-null    int64
 16  make_bmw                190 non-null    int64
 17  make_chevrolet          190 non-null    int64
 18  make_dodge              190 non-null    int64
 19  make_honda              190 non-null    int64
 20  make_isuzu              190 non-null    int64
 21  make_jaguar             190 non-null    int64
 22  make_mazda              190 non-null    int64
 23  make_mercedes-benz      190 non-null    int64
 24  make_mercury            190 non-null    int64
 25  make_mitsubishi         190 non-null    int64
 26  make_nissan             190 non-null    int64
 27  make_peugot             190 non-null    int64
 28  make_plymouth           190 non-null    int64
 29  make_porsche            190 non-null    int64
 30  make_saab               190 non-null    int64
 31  make_subaru             190 non-null    int64
 32  make_toyota             190 non-null    int64
 33  make_volkswagen         190 non-null    int64
 34  make_volvo              190 non-null    int64
 35  fuel-type_gas           190 non-null    int64
 36  aspiration_turbo        190 non-null    int64
 37  num-of-doors_two        190 non-null    int64
 38  body-style_hardtop      190 non-null    int64
 39  body-style_hatchback    190 non-null    int64
 40  body-style_sedan        190 non-null    int64
 41  body-style_wagon        190 non-null    int64
 42  drive-wheels_fwd        190 non-null    int64
 43  drive-wheels_rwd        190 non-null    int64
 44  engine-location_rear    190 non-null    int64
 45  engine-type_l           190 non-null    int64
 46  engine-type_ohc         190 non-null    int64
 47  engine-type_ohcf        190 non-null    int64
 48  engine-type_ohcv        190 non-null    int64
 49  num-of-cylinders_five   190 non-null    int64
 50  num-of-cylinders_four   190 non-null    int64
 51  num-of-cylinders_six    190 non-null    int64
 52  num-of-cylinders_three  190 non-null    int64
 53  num-of-cylinders_twelve 190 non-null    int64
 54  fuel-system_2bbl        190 non-null    int64
```

```
 55  fuel-system_idi        190 non-null    int64
 56  fuel-system_mfi        190 non-null    int64
 57  fuel-system_mpfi       190 non-null    int64
 58  fuel-system_spdi       190 non-null    int64
 59  fuel-system_spfi       190 non-null    int64
dtypes: int64(60)
memory usage: 90.5 KB
```

In [ ]:
```
df_encoded.head()
```

Out[ ]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compress r
0	3	88	168	64	48	2548	130	3	2	
1	3	88	168	64	48	2548	130	3	2	
2	1	94	171	65	52	2823	152	2	3	
3	2	99	176	66	54	2337	109	3	3	
4	2	99	176	66	54	2824	136	3	3	

5 rows × 60 columns

In [ ]:
```
df_encoded.columns
```

Out[ ]:
```
Index(['symboling', 'wheel-base', 'length', 'width', 'height', 'curb-weight',
       'engine-size', 'bore', 'stroke', 'compression-ratio', 'horsepower',
       'peak-rpm', 'city-mpg', 'highway-mpg', 'price', 'make_audi', 'make_bmw',
       'make_chevrolet', 'make_dodge', 'make_honda', 'make_isuzu',
       'make_jaguar', 'make_mazda', 'make_mercedes-benz', 'make_mercury',
       'make_mitsubishi', 'make_nissan', 'make_peugot', 'make_plymouth',
       'make_porsche', 'make_saab', 'make_subaru', 'make_toyota',
       'make_volkswagen', 'make_volvo', 'fuel-type_gas', 'aspiration_turbo',
       'num-of-doors_two', 'body-style_hardtop', 'body-style_hatchback',
       'body-style_sedan', 'body-style_wagon', 'drive-wheels_fwd',
       'drive-wheels_rwd', 'engine-location_rear', 'engine-type_l',
       'engine-type_ohc', 'engine-type_ohcf', 'engine-type_ohcv',
       'num-of-cylinders_five', 'num-of-cylinders_four',
       'num-of-cylinders_six', 'num-of-cylinders_three',
       'num-of-cylinders_twelve', 'fuel-system_2bbl', 'fuel-system_idi',
       'fuel-system_mfi', 'fuel-system_mpfi', 'fuel-system_spdi',
       'fuel-system_spfi'],
      dtype='object')
```

In [ ]:
```
for col in df_encoded.columns:
    print(col)
```

```
symboling
wheel-base
length
width
height
curb-weight
engine-size
bore
stroke
compression-ratio
horsepower
peak-rpm
city-mpg
highway-mpg
price
make_audi
make_bmw
make_chevrolet
make_dodge
make_honda
make_isuzu
make_jaguar
make_mazda
make_mercedes-benz
make_mercury
make_mitsubishi
make_nissan
make_peugot
make_plymouth
make_porsche
make_saab
make_subaru
make_toyota
make_volkswagen
make_volvo
fuel-type_gas
aspiration_turbo
num-of-doors_two
body-style_hardtop
body-style_hatchback
body-style_sedan
body-style_wagon
drive-wheels_fwd
drive-wheels_rwd
engine-location_rear
engine-type_l
engine-type_ohc
engine-type_ohcf
engine-type_ohcv
num-of-cylinders_five
num-of-cylinders_four
num-of-cylinders_six
num-of-cylinders_three
num-of-cylinders_twelve
fuel-system_2bbl
fuel-system_idi
fuel-system_mfi
fuel-system_mpfi
fuel-system_spdi
fuel-system_spfi
```

In [ ]:
```
len(df_encoded.columns)
```
Out[ ]:
```
60
```

In [ ]:
```
df_encoded.shape
```
Out[ ]:
```
(190, 60)
```

In [ ]:
```
Y = df_encoded['price']
X = df_encoded.drop('price'  ,axis = 1)
```

In [ ]:
```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_
```

In [ ]:
```
from statsmodels.stats.diagnostic import het_breuschpagan
import statsmodels.api as sm
```

linear regression Model without removing multicollinearity

In [ ]:
```
# Perform Breusch-Pagan test
# We need to add a constant column to the independent variables
X_train_sm = sm.add_constant(X_train)
X_test_sm = np.concatenate([np.ones((X_test.shape[0], 1)), X_test], axis=1)
model = sm.OLS(y_train, X_train_sm).fit()
residuals = model.resid
print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.976
Model:                            OLS   Adj. R-squared:                  0.962
Method:                 Least Squares   F-statistic:                     71.20
Date:                Sun, 01 Sep 2024   Prob (F-statistic):           9.06e-59
Time:                        14:00:12   Log-Likelihood:                -1284.2
No. Observations:                 152   AIC:                             2680.
Df Residuals:                      96   BIC:                             2850.
Df Model:                          55                                         
Covariance Type:            nonrobust                                         
================================================================================
==========
                              coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                   -3562.6881   1.27e+04     -0.280      0.780   -2.88e+04
2.17e+04
symboling                 110.6032    307.672      0.359      0.720    -500.121
721.327
wheel-base                218.0377     92.838      2.349      0.021      33.756
402.319
length                    -49.3524     52.973     -0.932      0.354    -154.502
55.797
width                     182.8369    254.927      0.717      0.475    -323.189
688.862
height                   -259.2092    144.072     -1.799      0.075    -545.189
26.771
curb-weight                 6.2861      1.599      3.932      0.000       3.113
9.459
engine-size                -7.1448     23.144     -0.309      0.758     -53.085
38.795
bore                     -648.6617    797.084     -0.814      0.418   -2230.861
933.538
stroke                   -483.1015    885.317     -0.546      0.587   -2240.442
1274.239
compression-ratio        -141.6028    382.044     -0.371      0.712    -899.955
616.749
horsepower                  8.5163     23.590      0.361      0.719     -38.309
55.342
peak-rpm                    0.8719      0.634      1.375      0.172      -0.387
2.131
city-mpg                  -90.3840    138.181     -0.654      0.515    -364.671
183.903
highway-mpg               110.6330    115.179      0.961      0.339    -117.996
339.262
make_audi                1966.7872   2356.890      0.834      0.406   -2711.602
6645.177
make_bmw                 5692.6570   1834.112      3.104      0.003    2051.974
9333.340
make_chevrolet          -2211.5013   2400.916     -0.921      0.359   -6977.283
2554.280
make_dodge              -3788.8507   1701.180     -2.227      0.028   -7165.667
-412.035
make_honda                -58.2272   2023.491     -0.029      0.977   -4074.826
3958.371
make_isuzu              -3898.0965   2241.767     -1.739      0.085   -8347.968
551.775
make_jaguar              8921.6976   3118.002      2.861      0.005    2732.513
```

```
1.51e+04
make_mazda              -886.0349   1553.435     -0.570      0.570   -3969.579
2197.509
make_mercedes-benz      5658.7947   2331.412      2.427      0.017    1030.979
1.03e+04
make_mercury           -2335.9685   2386.366     -0.979      0.330   -7072.867
2400.930
make_mitsubishi        -4061.8468   1807.702     -2.247      0.027   -7650.107
-473.587
make_nissan            -1357.3444   1533.711     -0.885      0.378   -4401.737
1687.048
make_peugot            -1339.8610   1006.279     -1.331      0.186   -3337.310
657.588
make_plymouth          -4438.1786   1793.849     -2.474      0.015   -7998.940
-877.417
make_porsche            6008.7639   2454.339      2.448      0.016    1136.940
1.09e+04
make_saab               1093.0622   1903.990      0.574      0.567   -2686.327
4872.452
make_subaru            -4848.1038   1316.302     -3.683      0.000   -7460.943
-2235.265
make_toyota            -2071.6908   1412.949     -1.466      0.146   -4876.372
732.991
make_volkswagen        -1060.7665   1736.442     -0.611      0.543   -4507.577
2386.044
make_volvo              -225.8645   1829.319     -0.123      0.902   -3857.035
3405.306
fuel-type_gas          -3135.9619   6059.894     -0.517      0.606   -1.52e+04
8892.832
aspiration_turbo        2282.3728    824.567      2.768      0.007     645.621
3919.125
num-of-doors_two        -654.4867    485.704     -1.347      0.181   -1618.603
309.629
body-style_hardtop     -3621.1487   1059.902     -3.416      0.001   -5725.038
-1517.260
body-style_hatchback   -3200.1801    984.402     -3.251      0.002   -5154.203
-1246.157
body-style_sedan       -3275.1019   1093.881     -2.994      0.004   -5446.438
-1103.766
body-style_wagon       -3864.6691   1187.707     -3.254      0.002   -6222.248
-1507.090
drive-wheels_fwd        -114.7347    961.083     -0.119      0.905   -2022.470
1793.001
drive-wheels_rwd         794.5513   1256.567      0.632      0.529   -1699.715
3288.818
engine-location_rear    7743.4025   1754.721      4.413      0.000    4260.309
1.12e+04
engine-type_l          -1339.8610   1006.279     -1.331      0.186   -3337.310
657.588
engine-type_ohc          753.2795   1060.560      0.710      0.479   -1351.916
2858.475
engine-type_ohcf        2895.2987    983.835      2.943      0.004     942.402
4848.196
engine-type_ohcv       -1015.7030   1204.908     -0.843      0.401   -3407.426
1376.020
num-of-cylinders_five  -7004.7439   2018.425     -3.470      0.001    -1.1e+04
-2998.202
num-of-cylinders_four  -7853.8428   2457.494     -3.196      0.002   -1.27e+04
-2975.755
num-of-cylinders_six   -5100.5293   2267.519     -2.249      0.027   -9601.520
```

```
-599.539
num-of-cylinders_three  -1.615e-12    1.02e-12     -1.584      0.117    -3.64e-12
4.09e-13
num-of-cylinders_twelve -2691.8760    5271.467     -0.511      0.611    -1.32e+04
7771.903
fuel-system_2bbl         2184.2329    1320.292      1.654      0.101     -436.526
4804.991
fuel-system_idi          -426.7262    7640.074     -0.056      0.956    -1.56e+04
1.47e+04
fuel-system_mfi          1923.4358    2466.417      0.780      0.437    -2972.363
6819.235
fuel-system_mpfi         1862.6080    1397.489      1.333      0.186     -911.386
4636.602
fuel-system_spdi         2457.4086    1757.518      1.398      0.165    -1031.238
5946.055
fuel-system_spfi         2645.0590    2828.953      0.935      0.352    -2970.367
8260.485
==============================================================================
Omnibus:                        6.760   Durbin-Watson:                   1.796
Prob(Omnibus):                  0.034   Jarque-Bera (JB):                8.952
Skew:                           0.246   Prob(JB):                       0.0114
Kurtosis:                       4.082   Cond. No.                     1.00e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 4.95e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Ridge lasso without removing multicoll.

In [ ]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Ridge and Lasso
alpha_values = np.logspace(-10, 10, 50)
ridge = Ridge()
lasso = Lasso()

ridge_cv = GridSearchCV(ridge, {'alpha': alpha_values}, cv=5, scoring='r2')
lasso_cv = GridSearchCV(lasso, {'alpha': alpha_values}, cv=5, scoring='r2')

# Fit models
ridge_cv.fit(X_train_sm, y_train)
lasso_cv.fit(X_train_sm, y_train)

print(f'Best alpha for Ridge: {ridge_cv.best_params_["alpha"]}')
print(f'Best alpha for Lasso: {lasso_cv.best_params_["alpha"]}')

# Predict and evaluate Ridge
ridge_best = ridge_cv.best_estimator_
y_pred_ridge = ridge_best.predict(X_test_sm)
r2_ridge = r2_score(y_test, y_pred_ridge)


print(f'Ridge Regression R2: {r2_ridge}')


# Predict and evaluate Lasso
lasso_best = lasso_cv.best_estimator_
```

```
y_pred_lasso = lasso_best.predict(X_test_sm)
r2_lasso = r2_score(y_test, y_pred_lasso)

print(f'Lasso Regression R2: {r2_lasso}')
```

```
Best alpha for Ridge: 0.09540954763499963
Best alpha for Lasso: 4.094915062380419
Ridge Regression R2: 0.9210840564884467
Lasso Regression R2: 0.9133891857628154
```

null hypo= homoscadasticity is accepted

In [ ]:
```
bp_test = het_breuschpagan(residuals, X_train_sm)
labels = ['LM Statistic', 'LM-Test p-value', 'F-Statistic', 'F-Test p-value']
print(dict(zip(labels, bp_test)))
```

```
{'LM Statistic': np.float64(69.26235044339563), 'LM-Test p-value': np.float64(0.1
696640111912708), 'F-Statistic': np.float64(1.4611762004138344), 'F-Test p-valu
e': np.float64(0.05208971087556576)}
```

remove multicollinearity

In [ ]:
```
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Function to calculate VIF and remove features with VIF > threshold
def remove_high_vif_features(X, threshold=10):
    while True:
        vif_data = pd.DataFrame()
        vif_data["feature"] = X.columns
        vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range

        max_vif = vif_data["VIF"].max()

        if max_vif > threshold:
            feature_to_remove = vif_data.loc[vif_data["VIF"].idxmax(), "feature"
            X = X.drop(columns=[feature_to_remove])
        else:
            break

    return X, vif_data

# Example usage with X_train
X_train_reduced, final_vif_df = remove_high_vif_features(X_train)
# Apply the same feature selection to X_test
X_test_reduced = X_test[X_train_reduced.columns]
# Print the final VIF DataFrame
print("Final VIF values after removing features with VIF greater than 10:")
print(final_vif_df)
```

```
Final VIF values after removing features with VIF greater than 10:
                     feature       VIF
0                  symboling  6.481206
1                  make_audi  2.825594
2                   make_bmw  2.148518
3             make_chevrolet  1.142317
4                 make_dodge  1.720043
5                 make_honda  1.253226
6                 make_isuzu  2.725020
7                make_jaguar  2.641662
8                 make_mazda  1.937558
9         make_mercedes-benz  4.253093
10              make_mercury  1.263865
11           make_mitsubishi  2.637868
12               make_nissan  2.630996
13             make_plymouth  1.546900
14              make_porsche  4.661725
15                 make_saab  1.511655
16               make_toyota  2.895494
17           make_volkswagen  1.535618
18                make_volvo  3.586337
19          aspiration_turbo  3.069805
20          num-of-doors_two  5.950474
21        body-style_hardtop  1.674088
22      body-style_hatchback  3.604320
23          body-style_wagon  1.386276
24          drive-wheels_rwd  8.775172
25      engine-location_rear  5.876483
26             engine-type_l  2.814629
27          engine-type_ohcf  2.004726
28          engine-type_ohcv  4.419439
29     num-of-cylinders_five  3.920994
30      num-of-cylinders_six  5.516013
31    num-of-cylinders_three       NaN
32   num-of-cylinders_twelve  3.265510
33          fuel-system_2bbl  5.217543
34           fuel-system_idi  2.510412
35           fuel-system_mfi  1.400017
36          fuel-system_spdi  2.570263
37          fuel-system_spfi  2.230798
```

linear regression after removing the multi.

In [ ]:

```

```

In [ ]:

```
import statsmodels.api as sm

# Fit the OLS model
model = sm.OLS(y_train, X_train_reduced).fit()

residuals=model.resid
# View model summary (optional)
print(model.summary())
y_pred_b = model.predict(X_test_reduced)
r = r2_score(y_pred_b , y_test)
print(f"The R2 score on test is: {r}")
residuals1=y_test-y_pred_b
```

```
                                 OLS Regression Results
=================================================================================
======
Dep. Variable:                  price   R-squared (uncentered):
0.978
Model:                            OLS   Adj. R-squared (uncentered):
0.971
Method:                 Least Squares   F-statistic:
138.8
Date:                Sun, 01 Sep 2024   Prob (F-statistic):                   1.
37e-79
Time:                        14:02:43   Log-Likelihood:                          -
1387.5
No. Observations:                 152   AIC:
2849.
Df Residuals:                     115   BIC:
2961.
Df Model:                          37
Covariance Type:            nonrobust
=================================================================================
==========
                              coef    std err          t      P>|t|      [0.025
0.975]
---------------------------------------------------------------------------------
----------
symboling                 346.4925    367.606      0.943      0.348    -381.665
1074.650
make_audi                1.619e+04   1926.549      8.405      0.000    1.24e+04
2e+04
make_bmw                 1.499e+04   1679.944      8.925      0.000    1.17e+04
1.83e+04
make_chevrolet           7604.0290   2739.072      2.776      0.006    2178.454
1.3e+04
make_dodge               7337.9756   1372.159      5.348      0.000    4619.994
1.01e+04
make_honda               8894.6311    828.198     10.740      0.000    7254.130
1.05e+04
make_isuzu               1474.1657   2991.439      0.493      0.623   -4451.299
7399.630
make_jaguar              2.703e+04   2945.329      9.177      0.000    2.12e+04
3.29e+04
make_mazda               9571.7358   1128.074      8.485      0.000    7337.238
1.18e+04
make_mercedes-benz       2.252e+04   2157.680     10.438      0.000    1.82e+04
2.68e+04
make_mercury             8917.6738   2881.115      3.095      0.002    3210.740
1.46e+04
make_mitsubishi          8220.6469   1471.607      5.586      0.000    5305.677
1.11e+04
make_nissan              7454.9492   1039.227      7.174      0.000    5396.441
9513.457
make_plymouth            6298.3709   1593.718      3.952      0.000    3141.523
9455.219
make_porsche              1.68e+04   2766.646      6.072      0.000    1.13e+04
2.23e+04
make_saab                1.388e+04   1409.132      9.848      0.000    1.11e+04
1.67e+04
make_toyota              7929.2374    824.124      9.621      0.000    6296.806
9561.668
make_volkswagen          9972.6894   1058.597      9.421      0.000    7875.813
```

```
1.21e+04
make_volvo              1.107e+04   1534.743      7.212      0.000    8027.815
1.41e+04
aspiration_turbo        3058.1797    819.795      3.730      0.000    1434.324
4682.035
num-of-doors_two        -643.4723    807.068     -0.797      0.427   -2242.120
955.175
body-style_hardtop     -2656.5370   1353.704     -1.962      0.052   -5337.965
24.891
body-style_hatchback    -749.3786    695.062     -1.078      0.283   -2126.164
627.406
body-style_wagon         -25.7726    711.212     -0.036      0.971   -1434.546
1383.001
drive-wheels_rwd        5573.5049   1023.662      5.445      0.000    3545.827
7601.183
engine-location_rear    2043.8814   3586.810      0.570      0.570   -5060.899
9148.661
engine-type_l           9201.6045   1625.069      5.662      0.000    5982.656
1.24e+04
engine-type_ohcf        8540.6471   1094.061      7.806      0.000    6373.524
1.07e+04
engine-type_ohcv        4054.8322   1624.417      2.496      0.014     837.175
7272.489
num-of-cylinders_five   1370.3391   1691.559      0.810      0.420   -1980.313
4720.991
num-of-cylinders_six    2947.1174   1380.849      2.134      0.035     211.922
5682.313
num-of-cylinders_three -1.621e-12   1.08e-12     -1.498      0.137   -3.76e-12
5.22e-13
num-of-cylinders_twelve  -14.2425   4631.117     -0.003      0.998   -9187.594
9159.109
fuel-system_2bbl        -262.6706    827.863     -0.317      0.752   -1902.507
1377.166
fuel-system_idi        -1817.5298   1048.424     -1.734      0.086   -3894.256
259.197
fuel-system_mfi         2921.2180   3032.333      0.963      0.337   -3085.250
8927.686
fuel-system_spdi        2052.4588   1837.443      1.117      0.266   -1587.163
5692.081
fuel-system_spfi        4700.1953   3827.722      1.228      0.222   -2881.785
1.23e+04
==============================================================================
Omnibus:                       35.722   Durbin-Watson:                   1.988
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               84.872
Skew:                           0.972   Prob(JB):                     3.72e-19
Kurtosis:                       6.102   Cond. No.                     1.11e+16
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contai
n a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The smallest eigenvalue is 3.14e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
The R2 score on test is: 0.6309658777185684
```

In [ ]:

```
residuals1
```

Out[ ]:
```
190      360.683910
195    -2854.086109
123     3257.564766
77     -1069.110403
113     1945.663197
17     13366.180558
10     -4186.214764
18    -10692.097017
156     -728.566763
139     -525.110671
121      309.807126
107    -2875.109383
20      -766.358415
179      -98.486336
97       486.001465
135      939.456832
116     1934.240673
78      -589.110403
66      5016.289096
47     -3300.000000
33     -1319.272704
26       187.202423
170      303.281947
109    -2309.336803
80     -3018.911976
142     -502.976556
138    -2460.110671
21      -456.946632
50     -3067.706761
87     -4398.777861
183    -2047.202209
79     -4595.926917
5      -2363.009627
147     1683.125434
184     -853.144746
178      461.513664
90      1758.560349
124    -3865.140928
dtype: float64
```

as the linear regression model is overfitting to the training data so we use the l1 l2 regul.

In [ ]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Ridge and Lasso
alpha_values = np.logspace(-10, 10, 50)
ridge = Ridge()
lasso = Lasso()

ridge_cv = GridSearchCV(ridge, {'alpha': alpha_values}, cv=5, scoring='r2')
lasso_cv = GridSearchCV(lasso, {'alpha': alpha_values}, cv=5, scoring='r2')

# Fit models
ridge_cv.fit(X_train_reduced, y_train)
lasso_cv.fit(X_train_reduced, y_train)

print(f'Best alpha for Ridge: {ridge_cv.best_params_["alpha"]}')
print(f'Best alpha for Lasso: {lasso_cv.best_params_["alpha"]}')
```

```
# Predict and evaluate Ridge
ridge_best = ridge_cv.best_estimator_
y_pred_ridge = ridge_best.predict(X_test_reduced)
r2_ridge = r2_score(y_test, y_pred_ridge)


print(f'Ridge Regression R2: {r2_ridge}')


# Predict and evaluate Lasso
lasso_best = lasso_cv.best_estimator_
y_pred_lasso = lasso_best.predict(X_test_reduced)
r2_lasso = r2_score(y_test, y_pred_lasso)

print(f'Lasso Regression R2: {r2_lasso}')
```

```
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 2.611e+07, tolerance: 7.218e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.491e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.710e+06, tolerance: 7.218e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.492e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 1.305e+06, tolerance: 7.218e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
```

```
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.493e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
```

```
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.494e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
C:\Users\gunav\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_c
oordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You mig
ht want to increase the number of iterations, check the scale of the features or
consider increasing regularisation. Duality gap: 6.495e+07, tolerance: 5.896e+05
  model = cd_fast.enet_coordinate_descent(
Best alpha for Ridge: 0.09540954763499963
Best alpha for Lasso: 10.481131341546874
Ridge Regression R2: 0.7890328838282821
Lasso Regression R2: 0.7839008571848591
```

l1 l2 gives better r2 score

In [ ]:
```
from statsmodels.stats.stattools import durbin_watson

residuals = model.resid

# Perform the Durbin-Watson test
dw_stat = durbin_watson(residuals)

print(f'Durbin-Watson statistic: {dw_stat}')
```

```
Durbin-Watson statistic: 1.9879918634485427
```

In [ ]:
```
X_train_reduced_c= sm.add_constant(X_train_reduced)
```

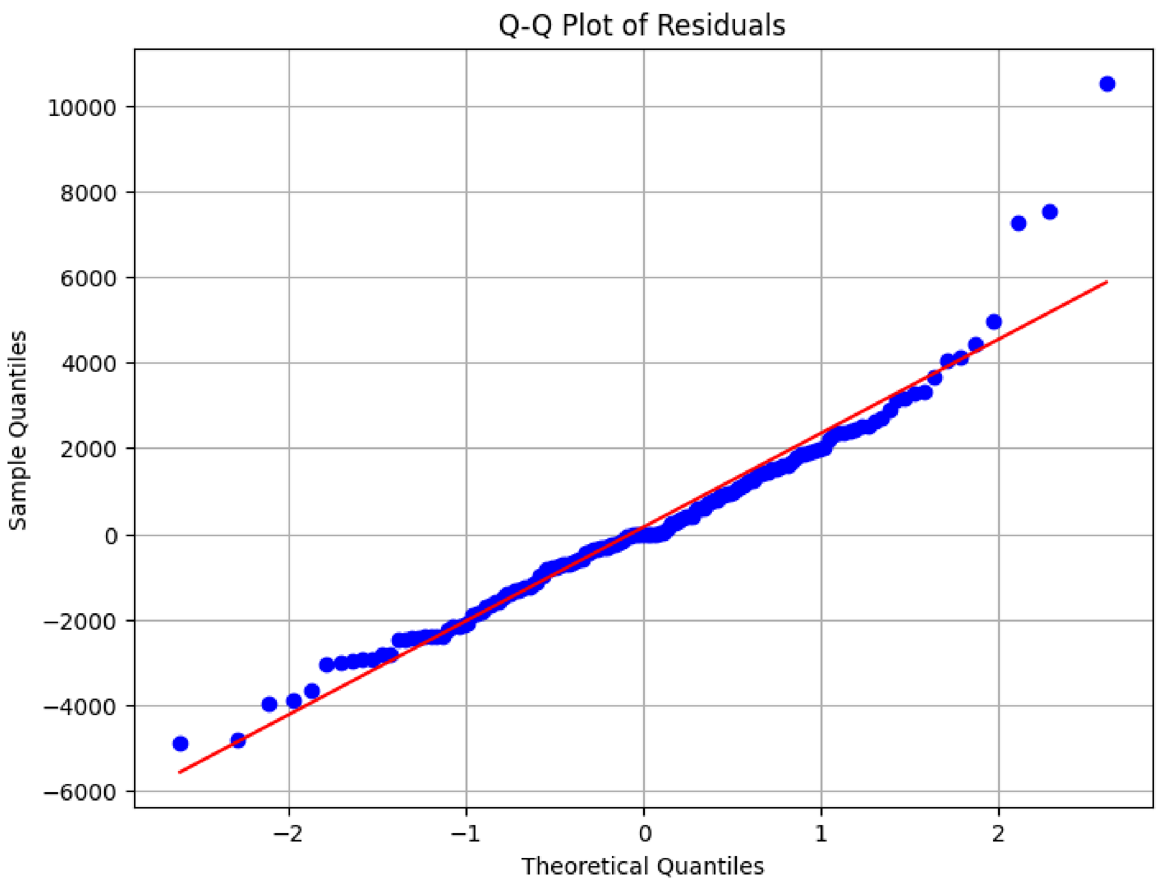
In [ ]:
```
bp_test = het_breuschpagan(residuals, X_train_reduced_c)
labels = ['LM Statistic', 'LM-Test p-value', 'F-Statistic', 'F-Test p-value']
print(dict(zip(labels, bp_test)))
```

```
{'LM Statistic': np.float64(98.10367855729513), 'LM-Test p-value': np.float64(3.2
638839142648266e-07), 'F-Statistic': np.float64(5.608274922967953), 'F-Test p-val
ue': np.float64(5.707494571574805e-13)}
```

dw test signifies that residuals are uncorrelated

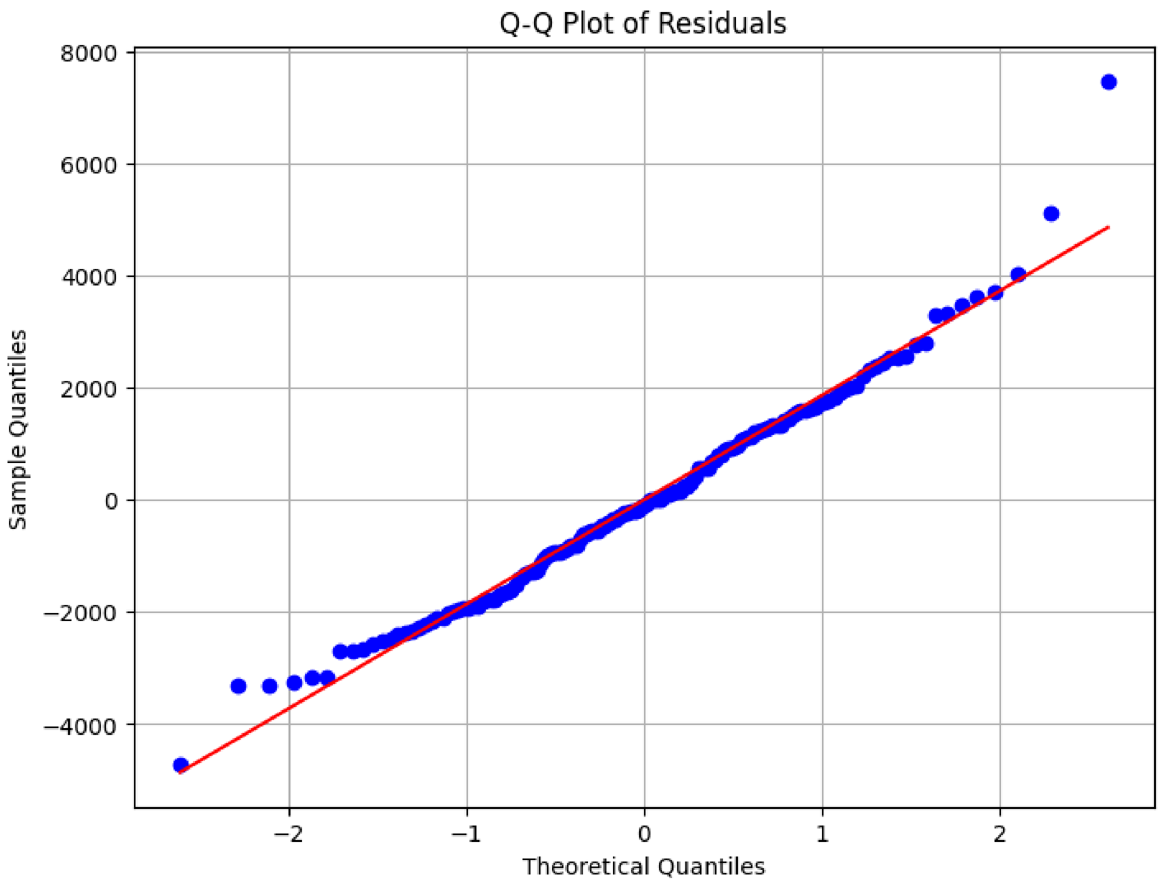
In [ ]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import scipy.stats as stats

model = sm.OLS(y_train, X_train_reduced).fit()

# Get the residuals
residuals = model.resid

# Plot Q-Q plot
plt.figure(figsize=(8, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
```

```
plt.grid(True)
plt.show()
```

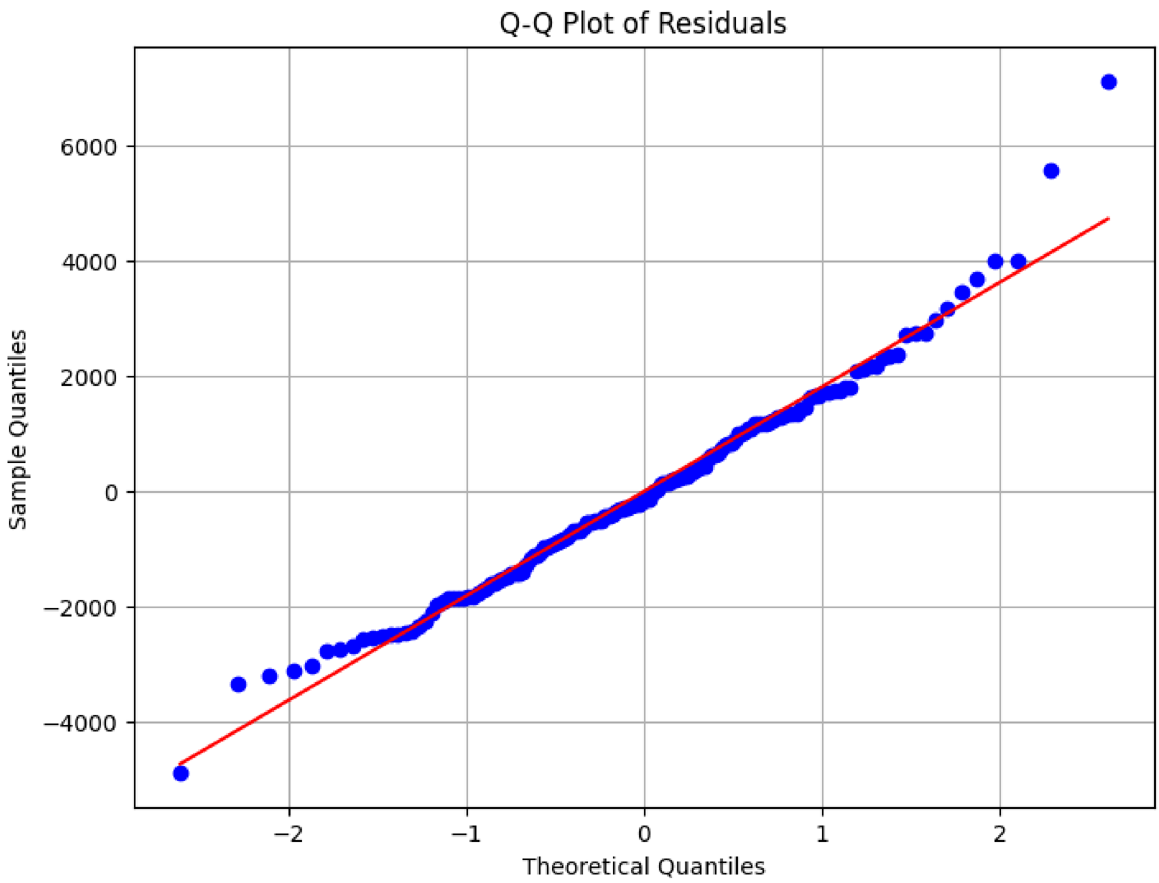


In [ ]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import scipy.stats as stats


# Get the residuals
residuals = y_train - lasso_best.predict(X_train_reduced)

# Plot Q-Q plot
plt.figure(figsize=(8, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid(True)
plt.show()
```



In [ ]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import scipy.stats as stats


# Get the residuals
residuals = y_train - ridge_best.predict(X_train_reduced)

# Plot Q-Q plot
plt.figure(figsize=(8, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid(True)
plt.show()
```



all the above plots shows normality

In [ ]:
```
plt.scatter(y_pred_b,residuals1)
```

Out[ ]: <matplotlib.collections.PathCollection at 0x1c534da90d0>