In [1]:

```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import bartlett
```

In [2]:

```python
db = pd.read_excel('db.xls')
```

In [3]:

```python
db.head()
```

Out[3]:

	Wieght	Engine Size	Bore	Price
0	2548	130	3.47	27
1	2548	130	3.47	27
2	2823	152	2.68	26
3	2337	109	3.19	30
4	2824	136	3.19	22

In [4]:

```python
db.columns = ['Weight', 'EngineSize', 'Bore', 'Price']
```

In [5]:

```python
db.head()
```

Out[5]:

	Weight	EngineSize	Bore	Price
0	2548	130	3.47	27
1	2548	130	3.47	27
2	2823	152	2.68	26
3	2337	109	3.19	30
4	2824	136	3.19	22

In [9]:

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
from patsy import dmatrices
```

In [10]:

```python
expr = 'Price ~ EngineSize + Bore + Weight'
```

In [11]:

```python
olsr_results = smf.ols(expr, db).fit()
```

In [12]:

```python
olsr_results.summary()
```

Out[12]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.674
Model:	OLS	**Adj. R-squared:**	0.669
Method:	Least Squares	**F-statistic:**	135.6
Date:	Tue, 02 Nov 2021	**Prob (F-statistic):**	1.14e-47
Time:	14:17:31	**Log-Likelihood:**	-559.40
No. Observations:	201	**AIC:**	1127.
Df Residuals:	197	**BIC:**	1140.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	64.0948	3.703	17.309	0.000	56.792	71.397
EngineSize	-0.0158	0.013	-1.192	0.235	-0.042	0.010
Bore	-2.6998	1.349	-2.001	0.047	-5.360	-0.040
Weight	-0.0087	0.001	-7.862	0.000	-0.011	-0.006

Omnibus:	48.875	**Durbin-Watson:**	1.635
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	88.717
Skew:	1.221	**Prob(JB):**	5.44e-20
Kurtosis:	5.152	**Cond. No.**	3.66e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.66e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [13]:

```python
from statsmodels.stats.diagnostic import het_white
from statsmodels.compat import lzip
```

In [14]:

```python
y, X = dmatrices(expr, db, return_type='dataframe')
keys = ['Lagrange Multiplier statistic:', 'LM test\'s p-value:', 'F-statistic:', 'F-test\'s p-value:']
results = het_white(olsr_results.resid, X)
lzip(keys, results)
```

Out[14]:

```
[('Lagrange Multiplier statistic:', 13.915730557586318),
 ("LM test's p-value:", 0.12535458976604108),
 ('F-statistic:', 1.5785545581028562),
 ("F-test's p-value:", 0.12404783910497975)]
```

The LM test's p-value is 0.00059 which is less than 0.025. So we reject reject the null hypothesis that there is no heteroscedastisticity for the database. And the F-test's p-value is 0.00035 which confirmed the rejection of the null hypothesis. Overall, we conclude that the given dataset has heteroscedastisticity (heteroscedastisticity is violated).

Question 1 (b) Fixed Heteroscedasticity

In [15]:

```python
db_logged = np.log2(db)
```

In [16]:

```python
olsr_results = smf.ols(expr, db_logged).fit()
y, X = dmatrices(expr, db_logged, return_type='dataframe')
keys = ['Lagrange Multiplier statistic:', 'LM test\'s p-value:', 'F-statistic:', 'F-test\'s p-value:']
results = het_white(olsr_results.resid, X)
lzip(keys, results)
```

Out[16]:

```
[('Lagrange Multiplier statistic:', 10.517115547072354),
 ("LM test's p-value:", 0.3102641131952513),
 ('F-statistic:', 1.171740778274042),
 ("F-test's p-value:", 0.3152069117321394)]
```

In [17]:

```python
db_logged.head()
```

Out[17]:

	Weight	EngineSize	Bore	Price
0	11.315150	7.022368	1.794936	4.754888
1	11.315150	7.022368	1.794936	4.754888
2	11.463013	7.247928	1.422233	4.700440
3	11.190442	6.768184	1.673556	4.906891
4	11.463524	7.087463	1.673556	4.459432

As we could see here, the LM test's p-value and F-test's p-value are 0.310 and 0.315, which is larger than before, which means the Heteroscedasticity is fixed.

Question 2 (a) Check multicollinarity.

In [18]:

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [35]:

```python
db = db[~db.isin([np.nan, np.inf, -np.inf]).any(1)]
expr = 'Price ~ EngineSize + Bore + Weight'

y, X = dmatrices(expr, data=db, return_type='dataframe')
vif_data = pd.DataFrame()
```

In [36]:

```python
vif_data["feature"] = X.columns
```

In [37]:

```python
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
```

In [38]:

```python
vif_data
```

Out[38]:

	feature	VIF
0	Intercept	380.975988
1	Price	2.332696
2	EngineSize	2.330632
3	Bore	1.691136

The VIF values of Engine Size, Bore, and Weight are 3.84, 1.74, and 4.29, which indicate that there are moderate correlation between those given features and y variable 'Price' in our model 'Price ~ EngineSize + Bore + Weight' but this is often not severe enough to require attention.

In [53]:

```python
def getVIFDataFrame(expression):
    y, X = dmatrices(expression, data=db, return_type='dataframe')
    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
    vif_data[expression.split(' ~')[0]] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
    return vif_data
```

Ignore the string after '~', I just use the expression for convenience, in other words, 'Price ~ EngineSize + Bore + Weight' just means 'Price'

In [54]:

```python
getVIFDataFrame('Price ~ EngineSize + Bore + Weight')
```

Out[54]:

	feature	Price
0	Intercept	176.475115
1	EngineSize	3.843120
2	Bore	1.743360
3	Weight	4.296775

In [55]:

```python
getVIFDataFrame('EngineSize ~ Price + Bore + Weight')
```

Out[55]:

	feature	EngineSize
0	Intercept	441.405378
1	Price	3.042694
2	Bore	1.767149
3	Weight	3.398855

In [56]:

```python
getVIFDataFrame('Bore ~ Price + EngineSize + Weight')
```

Out[56]:

	feature	Bore
0	Intercept	239.751825
1	Price	3.003579
2	EngineSize	3.845483
3	Weight	5.366796

In [57]:

```python
getVIFDataFrame('Weight ~ Price + EngineSize + Bore')
```

Out[57]:

	feature	Weight
0	Intercept	380.975988
1	Price	2.332696
2	EngineSize	2.330632
3	Bore	1.691136

From the table above, we could found that the VIF value between Weight and Bore is 5.367 which is greater than 5 which indicates high correlation between them. And for others, the VIF vaue are all between 1 and 5 which indicates moderate correlation between given values, but this is often not severe enough to require attention.

Question 2 (b) Fix High Correlation issue

Since Weight and Bore have high correlation, so in order to fix the high correlation issue, we could simply remove one of them. By observing the table above, we could find that the VIF for Bore is always greater, so **we remove the Bore variable** in our model.