

# Multilinear Regression: Auto dataset

## Import notebook functions

```
from notebookfuncs import *
```

## Import standard libraries

```
import numpy as np
import pandas as pd

pd.set_option("display.max_rows", 1000)
pd.set_option("display.max_columns", 1000)
pd.set_option("display.width", 1000)
pd.set_option("display.max_colwidth", None)
import matplotlib.pyplot as plt
from matplotlib.pyplot import subplots
import seaborn as sns
import itertools
```

## New imports

```
import statsmodels.api as sm
```

## Import statsmodels.objects

```

from statsmodels.stats.outliers_influence import variance_inflation_factor as VIF
from statsmodels.stats.outliers_influence import summary_table
from statsmodels.stats.anova import anova_lm
import statsmodels.formula.api as smf

```

## Import ISLP objects

```

import ISLP
from ISLP import models
from ISLP import load_data
from ISLP.models import ModelSpec as MS, summarize, poly

```

## Import user functions

```

from userfuncs import *

```

## Set level of significance (alpha)

```

LOS_Alpha = 0.01

```

0.01

```

Auto = load_data("Auto")
Auto = Auto.sort_values(by=["year"], ascending=True)
Auto.head()
Auto.columns

```

```

Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year', '']

```

```

Auto.shape

```

(392, 8)

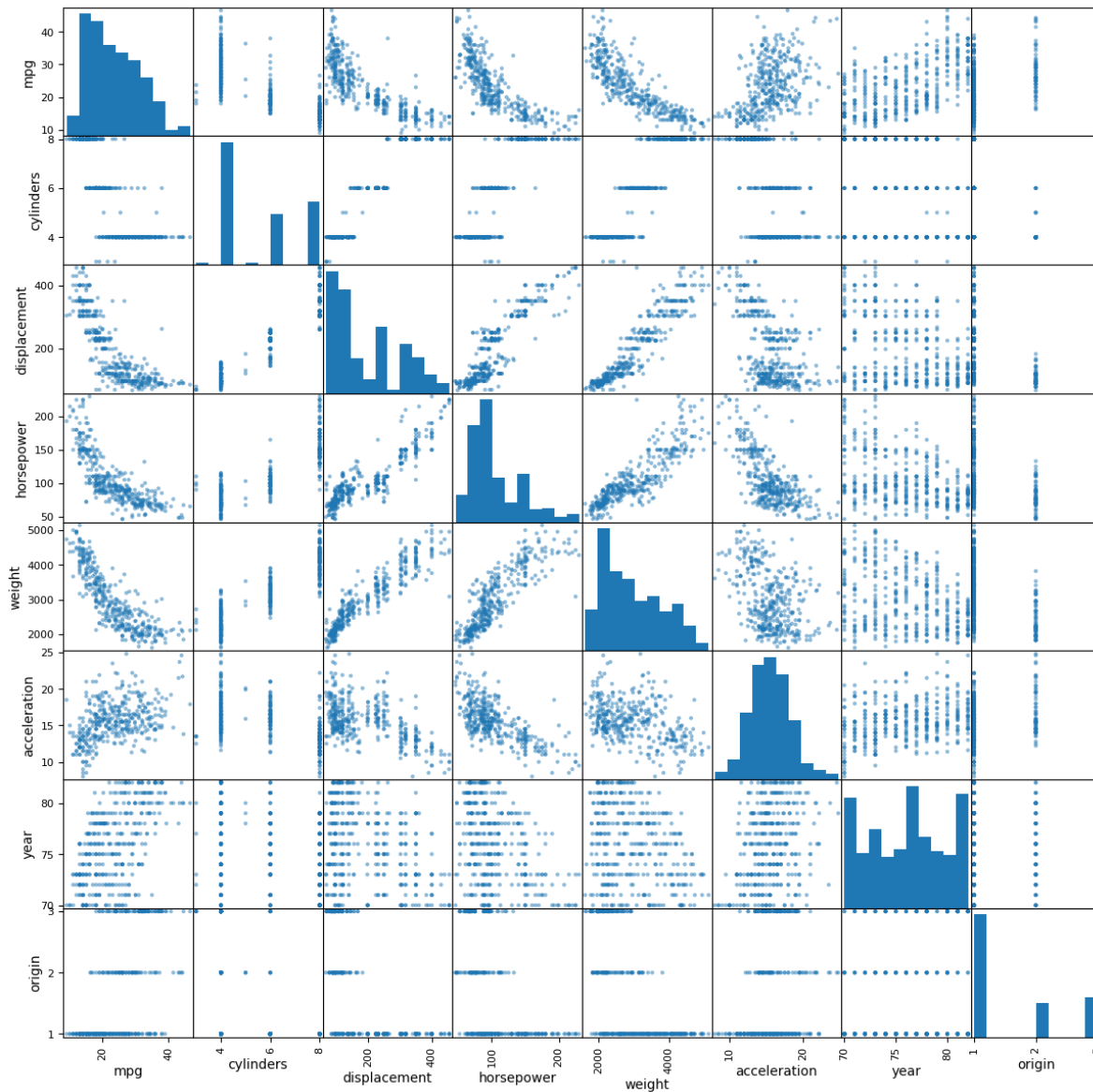
```
Auto.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.979592	1.570000
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.683737	0.800000
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.000000	1.000000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

**9. This question involves the use of multiple linear regression on the Auto data set.**

**(a) Produce a scatterplot matrix which includes all of the variables in the data set.**

```
pd.plotting.scatter_matrix(Auto, figsize=(14, 14))
```



**(b) Compute the matrix of correlations between the variables using the `DataFrame.corr()` method.**

```
Auto.corr()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	0.561
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.56
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.61
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.45
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.58
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.21
year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	0.18
origin	0.565209	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528	1.00

**(c) Use the `sm.OLS()` function to perform a multiple linear regression with `mpg` as the response and all other variables except `name` as the predictors. Use the `summarize()` function to print the results. Comment on the output. For instance:**

**Convert year and origin columns to categorical types**

```
Auto["origin"] = Auto["origin"].astype("category")
Auto["origin"] = Auto["origin"].cat.rename_categories(
    {1: "America", 2: "Europe", 3: "Japan"}
)
Auto["year"] = Auto["year"].astype("category")
Auto.describe()
```

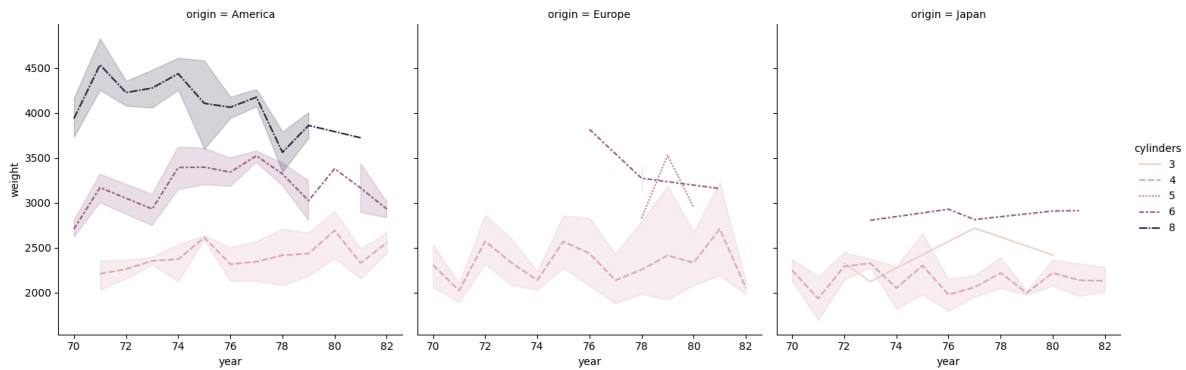
	mpg	cylinders	displacement	horsepower	weight	acceleration
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000

```
sns.relplot(
    Auto,
    x="year",
    y="weight",
    col="origin",
```

```

hue="cylinders",
style="cylinders",
estimator="mean",
kind="line",
)

```

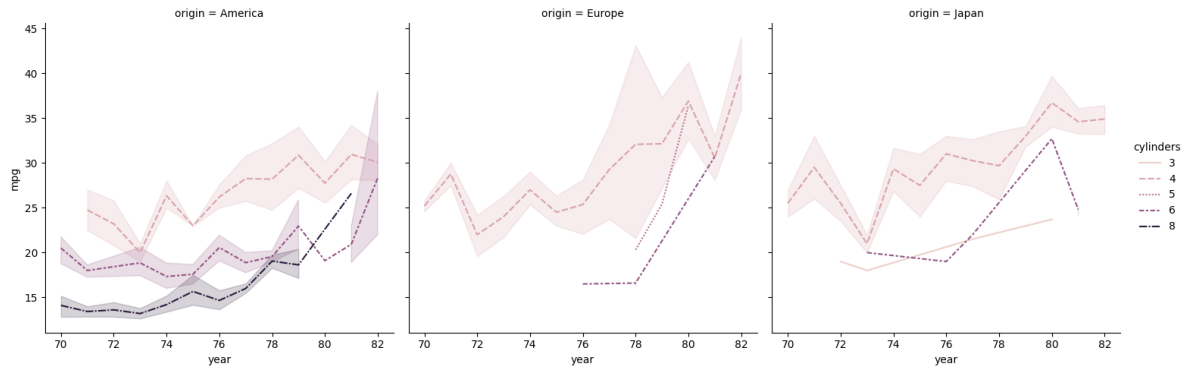


The weight of the 8-cylinder American made models show a decline from the highs of 1972. It can also be seen that American made cars are heavier than their European and Japanese counterparts especially in the most common models with 4 cylinders.

```

sns.relplot(
    Auto,
    x="year",
    y="mpg",
    col="origin",
    hue="cylinders",
    style="cylinders",
    estimator="mean",
    kind="line",
)

```



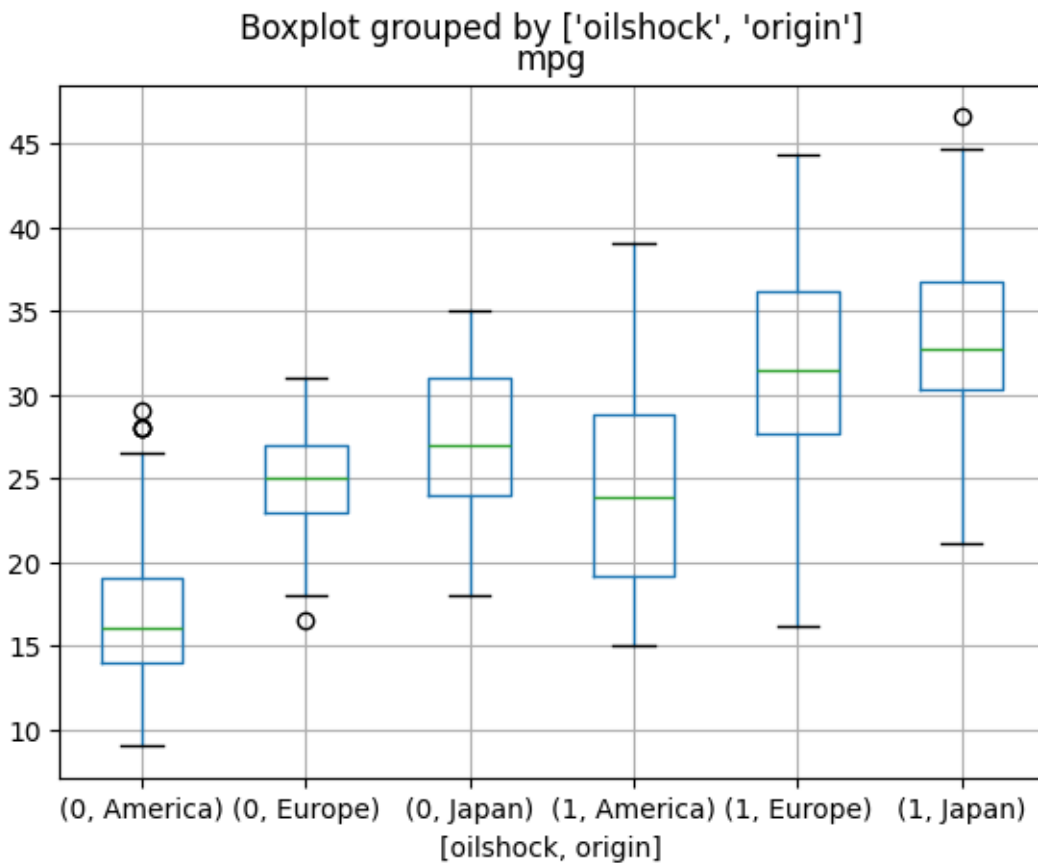
It can be seen that after the [oil shock of 1973](#) and the regulations and actions taken by the US government, the mileage for American made cars rose across all models. This was, however, matched by the European and Japanese models which were already lighter and more fuel efficient.

Encode categorical variables as dummy variables dropping the first to remove multicollinearity.

```
def categorize_for_oil_shock(row):
    # we add 3 years because it takes approximately that long for car manufacturers to introduce new models
    if row["year"] in (70, 71, 72, 73, 74, 75, 76):
        return 0
    return 1
```

```
Auto["oilshock"] = Auto.apply(categorize_for_oil_shock, axis=1)
```

```
Auto.boxplot(column="mpg", by=["oilshock", "origin"])
```



```
Auto_os = Auto.drop(["year"], axis=1)
Auto_os.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'origin',
```

```
# standardizing dataframes
Auto_os["oilshock"] = Auto_os["oilshock"].astype("category")
Auto_os = Auto_os.apply(standardize)
Auto_os.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration
count	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02
mean	1.812609e-16	-1.087565e-16	-7.250436e-17	-1.812609e-16	-3.625218e-17	-8.519262e-16
std	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00
min	-1.853218e+00	-1.451004e+00	-1.209563e+00	-1.520975e+00	-1.608575e+00	-2.736983e+00



	mpg	cylinders	displacement	horsepower	weight	acceleration
25%	-8.269250e-01	-8.640136e-01	-8.555316e-01	-7.665929e-01	-8.868535e-01	-6.410551e-01
50%	-8.927701e-02	-8.640136e-01	-4.153842e-01	-2.853488e-01	-2.052109e-01	-1.499869e-02
75%	7.125143e-01	1.483947e+00	7.782764e-01	5.600800e-01	7.510927e-01	5.384714e-01
max	2.970359e+00	1.483947e+00	2.493416e+00	3.265452e+00	2.549061e+00	3.360262e+00

```
Auto_os = pd.get_dummies(
    Auto_os, columns=list(["origin"]), drop_first=True, dtype=np.uint8
)
Auto_os.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'oilshock
```

```
y = Auto_os["mpg"]
```

```
cols = list(Auto_os.columns)
cols.remove("mpg")
formula = " + ".join(cols)
model = smf.ols(f"mpg ~ {formula}", data=Auto_os)
results = model.fit()
results.summary()
```

<b>Dep. Variable:</b>	mpg	<b>R-squared:</b>	0.808
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.804
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	201.7
<b>Date:</b>	Sat, 28 Sep 2024	<b>Prob (F-statistic):</b>	3.05e-132
<b>Time:</b>	04:09:22	<b>Log-Likelihood:</b>	-232.60
<b>No. Observations:</b>	392	<b>AIC:</b>	483.2
<b>Df Residuals:</b>	383	<b>BIC:</b>	518.9
<b>Df Model:</b>	8		
<b>Covariance Type:</b>	nonrobust		

	coef	std err	t	P>  t	[0.025	0.975]
<b>Intercept</b>	-0.4186	0.041	-10.263	0.000	-0.499	-0.338
<b>oilshock[T.1]</b>	0.6363	0.048	13.204	0.000	0.542	0.731
<b>cylinders</b>	-0.1382	0.073	-1.885	0.060	-0.282	0.006
<b>displacement</b>	0.2845	0.107	2.659	0.008	0.074	0.495
<b>horsepower</b>	-0.2213	0.069	-3.192	0.002	-0.358	-0.085
<b>weight</b>	-0.5923	0.073	-8.085	0.000	-0.736	-0.448
<b>acceleration</b>	0.0053	0.036	0.146	0.884	-0.066	0.076
<b>origin_Europe</b>	0.3038	0.076	4.015	0.000	0.155	0.453
<b>origin_Japan</b>	0.3819	0.074	5.156	0.000	0.236	0.528
<b>Omnibus:</b>	20.039		<b>Durbin-Watson:</b>	1.331		
<b>Prob(Omnibus):</b>	0.000		<b>Jarque-Bera (JB):</b>	27.583		
<b>Skew:</b>	0.413		<b>Prob(JB):</b>	1.02e-06		
<b>Kurtosis:</b>	4.004		<b>Cond. No.</b>	11.9		

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

i. Is there a relationship between the predictors and the response? Use the `anova_lm()` function from `statsmodels` to answer this question.

ii. Which predictors appear to have a statistically significant relationship to the response?

iii. What does the coefficient for the year variable suggest?

```
anova_lm(results)
```

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	542.347509	2.293307e-75
cylinders	1.0	170.845795	170.845795	870.163964	1.267122e-100
displacement	1.0	11.934469	11.934469	60.785485	6.078862e-14
horsepower	1.0	3.951021	3.951021	20.123619	9.610639e-06
weight	1.0	17.796189	17.796189	90.640818	1.988543e-19
acceleration	1.0	0.009116	0.009116	0.046430	8.295108e-01
origin_Europe	1.0	0.564108	0.564108	2.873155	9.088094e-02
origin_Japan	1.0	5.218909	5.218909	26.581317	4.058867e-07
Residual	383.0	75.197253	0.196337	NaN	NaN

There seems to be a statistical relationship between all of the predictors and the response variable, mpg, except for acceleration.

Even though some of the categorical variables are insignificant, even if one of the levels is significant, it is advisable to retain them all in the model.

<https://stats.stackexchange.com/questions/24298/can-i-ignore-coefficients-for-non-significant-levels-of-factors-in-a-linear-model>

Note: Year has been converted to a categorical variable oilshock to better capture the effects of the oil shock of 1973 on the mileage.

**(d) Produce some of diagnostic plots of the linear regression fit as described in the lab. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusually high leverage?**

**Before producing the diagnostic plots, let's first test for collinearity using correlation matrix and variance inflation factors.**

```
Auto_os.corr(numeric_only=True)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	origin_Europe
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.244313
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.352324
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.371633
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.284948
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.293841
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.208298
origin_Europe	0.244313	-0.352324	-0.371633	-0.284948	-0.293841	0.208298	1.000000
origin_Japan	0.451454	-0.404209	-0.440825	-0.321936	-0.447929	0.115020	-0.230157

```
vifdf = calculate_VIFs("mpg ~ " + " + ".join(Auto_os.columns) + " - mpg", Auto_os)
vifdf
```

	VIF
Feature	
oilshock[T.1]	1.149269
cylinders	10.737464

Feature	VIF
displacement	22.861475
horsepower	9.594564
weight	10.715246
acceleration	2.614133
origin_Europe	1.639338
origin_Japan	1.762590

```
identify_highest_VIF_feature(vifdf)
```

We find the highest VIF in this model is displacement with a VIF of 22.861474853464927. Hence, we drop displacement from the model to be fitted.

```
('displacement', 22.861474853464927)
```

```
vifdf = calculate_VIFs(
    "mpg ~ " + " + ".join(Auto_os.columns) + " - mpg - displacement", Auto_os
)
vifdf
```

Feature	VIF
oilshock[T.1]	1.139339
cylinders	6.190903
horsepower	8.641303
weight	9.024884
acceleration	2.591157
origin_Europe	1.450726
origin_Japan	1.591434

```
identify_highest_VIF_feature(vifdf)
```

No variables are significantly collinear.

# Linear Regression for mpg ~ cylinders + horsepower + weight + acceleration + oilshock + origin\_Europe + origin\_Japan

```
cols = list(Auto_os.columns)
cols.remove("mpg")
cols.remove("displacement")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_os)
```

## OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.805			
Model:	OLS	Adj. R-squared:	0.801			
Method:	Least Squares	F-statistic:	225.9			
Date:	Sat, 28 Sep 2024	Prob (F-statistic):	6.41e-132			
Time:	04:09:23	Log-Likelihood:	-236.18			
No. Observations:	392	AIC:	488.4			
Df Residuals:	384	BIC:	520.1			
Df Model:	7					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
Intercept	-0.3890	0.040	-9.837	0.000	-0.467	-0.311
oilshock[T.1]	0.6243	0.048	12.911	0.000	0.529	0.719
cylinders	-0.0113	0.056	-0.202	0.840	-0.122	0.099
horsepower	-0.1632	0.066	-2.461	0.014	-0.294	-0.033
weight	-0.5149	0.068	-7.599	0.000	-0.648	-0.382
acceleration	-0.0038	0.036	-0.103	0.918	-0.075	0.068
origin_Europe	0.2356	0.072	3.283	0.001	0.095	0.377
origin_Japan	0.3205	0.071	4.518	0.000	0.181	0.460
=====						
Omnibus:	25.646	Durbin-Watson:	1.305			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	40.287			
Skew:	0.456	Prob(JB):	1.79e-09			
Kurtosis:	4.278	Cond. No.	7.67			
=====						

## Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

df	sum_sq	mean_sq	F	PR(>F)
----	--------	---------	---	--------

oilshock	1.0	106.483141	106.483141	533.906720	1.149811e-74
cylinders	1.0	170.845795	170.845795	856.621225	7.985118e-100
horsepower	1.0	12.927972	12.927972	64.820882	1.039468e-14
weight	1.0	20.649905	20.649905	103.538670	1.085729e-21
acceleration	1.0	0.003626	0.003626	0.018183	8.928058e-01
origin_Europe	1.0	0.432514	0.432514	2.168627	1.416711e-01
origin_Japan	1.0	4.071523	4.071523	20.414626	8.312108e-06
Residual	384.0	76.585524	0.199441	NaN	NaN

```
identify_least_significant_feature(results, alpha=LOS_Alpha)
```

We find the least significant variable in this model is acceleration with a p-value of 0.917. Using the backward methodology, we suggest dropping acceleration from the new model.

**Linear Regression after dropping acceleration. The model now is mpg ~ cylinders + horsepower + weight + oilshock + origin\_Europe + origin\_Japan**

```
cols.remove("acceleration")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_os)
simple_model = results
models = []
models.append(
    {
        "name": "simple_model",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)
```

#### OLS Regression Results

```
=====
Dep. Variable:          mpg      R-squared:          0.805
Model:                  OLS      Adj. R-squared:      0.802
Method:                 Least Squares      F-statistic:      264.3
Date:                  Sat, 28 Sep 2024      Prob (F-statistic):      3.80e-133
Time:                  04:09:23      Log-Likelihood:      -236.19
No. Observations:      392      AIC:          486.4
Df Residuals:          385      BIC:          514.2
Df Model:              6
Covariance Type:      nonrobust
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.3889	0.039	-9.849	0.000	-0.467	-0.311
oilshock[T.1]	0.6245	0.048	12.935	0.000	0.530	0.719
cylinders	-0.0105	0.055	-0.189	0.850	-0.120	0.099
horsepower	-0.1585	0.048	-3.285	0.001	-0.253	-0.064
weight	-0.5182	0.060	-8.704	0.000	-0.635	-0.401
origin_Europe	0.2352	0.072	3.287	0.001	0.095	0.376
origin_Japan	0.3202	0.071	4.524	0.000	0.181	0.459
Omnibus:		25.330	Durbin-Watson:			1.305
Prob(Omnibus):		0.000	Jarque-Bera (JB):			39.508
Skew:		0.454	Prob(JB):			2.64e-09
Kurtosis:		4.263	Cond. No.			6.84

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	535.282217	7.450502e-75
cylinders	1.0	170.845795	170.845795	858.828126	4.453195e-100
horsepower	1.0	12.927972	12.927972	64.987879	9.610211e-15
weight	1.0	20.649905	20.649905	103.805416	9.634452e-22
origin_Europe	1.0	0.434982	0.434982	2.186618	1.400323e-01
origin_Japan	1.0	4.070552	4.070552	20.462339	8.111817e-06
Residual	385.0	76.587654	0.198929	NaN	NaN

**Linear Regression after dropping cylinders. The model now is mpg ~ horsepower + weight + oilshock + origin\_Europe + origin\_Japan**

```
cols.remove("cylinders")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_os)
simple_model = results
models = []
models.append(
    {
        "name": "simple_model",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
```

```
}
)
```

OLS Regression Results						
=====						
Dep. Variable:	mpg	R-squared:	0.805			
Model:	OLS	Adj. R-squared:	0.802			
Method:	Least Squares	F-statistic:	317.9			
Date:	Sat, 28 Sep 2024	Prob (F-statistic):	2.06e-134			
Time:	04:09:23	Log-Likelihood:	-236.21			
No. Observations:	392	AIC:	484.4			
Df Residuals:	386	BIC:	508.2			
Df Model:	5					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
Intercept	-0.3901	0.039	-10.030	0.000	-0.467	-0.314
oilshock[T.1]	0.6250	0.048	12.983	0.000	0.530	0.720
horsepower	-0.1613	0.046	-3.510	0.001	-0.252	-0.071
weight	-0.5245	0.050	-10.576	0.000	-0.622	-0.427
origin_Europe	0.2386	0.069	3.448	0.001	0.103	0.375
origin_Japan	0.3222	0.070	4.611	0.000	0.185	0.460
=====						
Omnibus:	24.971	Durbin-Watson:	1.304			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	38.456			
Skew:	0.453	Prob(JB):	4.46e-09			
Kurtosis:	4.239	Cond. No.	5.60			
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	536.622900	4.863116e-75
horsepower	1.0	165.048555	165.048555	831.763917	2.445119e-98
weight	1.0	39.079210	39.079210	196.940090	1.939884e-36
origin_Europe	1.0	0.574647	0.574647	2.895939	8.960825e-02
origin_Japan	1.0	4.219706	4.219706	21.265252	5.446537e-06
Residual	386.0	76.594741	0.198432	NaN	NaN

We can now try and plot the diagnostics for the model.



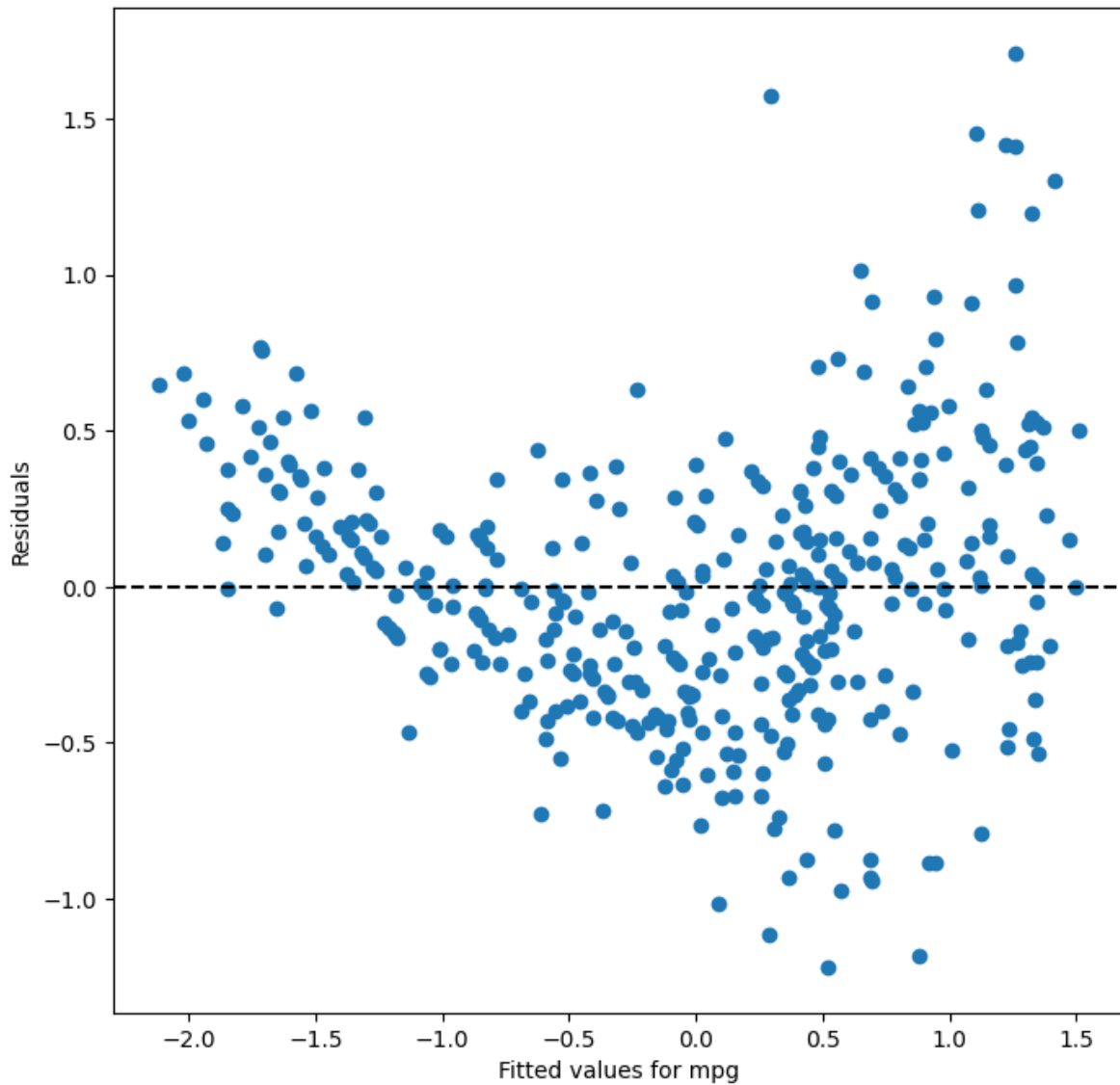
```
TSS = np.sum((y - np.mean(y)) ** 2)
TSS
RSS = np.sum((y - results.fittedvalues) ** 2)
RSS
RSE = np.sqrt(RSS / results.df_model)
display("RSE " + str(RSE))
display("R-squared adjusted : " + str(results.rsquared_adj))
display("F-statistic : " + str(results.fvalue))
```

```
'RSE 3.9139428061794668'
```

```
'R-squared adjusted : 0.8020742313429469'
```

```
'F-statistic : 317.8976193276657'
```

```
display_residuals_plot(results)
```



There is some evidence of non-linearity and heteroskedasticity from the residuals plot above.

**(e) Fit some models with interactions as described in the lab. Do any interactions appear to be statistically significant?**

```
formula = " + ".join(cols)
formula += " + " + "horsepower: weight"
results = perform_analysis("mpg", formula, Auto_os)
```

```

numeric_interactions = results
models.append(
    {
        "name": "numeric_interactions",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)

```

#### OLS Regression Results

```

=====
Dep. Variable:          mpg      R-squared:          0.847
Model:                  OLS      Adj. R-squared:       0.845
Method:                 Least Squares      F-statistic:       355.8
Date:                   Sat, 28 Sep 2024    Prob (F-statistic):    1.15e-153
Time:                   04:09:24    Log-Likelihood:       -187.98
No. Observations:       392      AIC:                390.0
Df Residuals:           385      BIC:                417.8
Df Model:                6
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.5631	0.038	-14.715	0.000	-0.638	-0.488
oilshock[T.1]	0.6508	0.043	15.243	0.000	0.567	0.735
horsepower	-0.3723	0.045	-8.185	0.000	-0.462	-0.283
weight	-0.4926	0.044	-11.192	0.000	-0.579	-0.406
origin_Europe	0.1565	0.062	2.535	0.012	0.035	0.278
origin_Japan	0.2061	0.063	3.278	0.001	0.082	0.330
horsepower:weight	0.2300	0.022	10.364	0.000	0.186	0.274

```

=====
Omnibus:                27.116    Durbin-Watson:          1.364
Prob(Omnibus):           0.000    Jarque-Bera (JB):        45.242
Skew:                    0.457    Prob(JB):                1.50e-10
Kurtosis:                4.390    Cond. No.                 7.09
=====

```

#### Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	684.553356	1.927144e-87
horsepower	1.0	165.048555	165.048555	1061.055689	1.099814e-112

weight	1.0	39.079210	39.079210	251.230425	6.522402e-44
origin_Europe	1.0	0.574647	0.574647	3.694260	5.533771e-02
origin_Japan	1.0	4.219706	4.219706	27.127429	3.109409e-07
horsepower:weight	1.0	16.707504	16.707504	107.408348	2.313061e-22
Residual	385.0	59.887237	0.155551	NaN	NaN

```

formula = " + ".join(cols)
formula += " + " + "horsepower: weight"
formula += " + " + "oilshock: weight"
formula += " + " + "oilshock: horsepower"
results = perform_analysis("mpg", formula, Auto_os)
oilshock_interactions = results
models.append(
    {
        "name": "oilshock_interactions",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)

```

#### OLS Regression Results

```

=====
Dep. Variable:          mpg      R-squared:          0.861
Model:                  OLS      Adj. R-squared:       0.858
Method:                 Least Squares      F-statistic:       297.5
Date:                   Sat, 28 Sep 2024    Prob (F-statistic):   3.50e-159
Time:                   04:09:24           Log-Likelihood:     -168.92
No. Observations:      392             AIC:              355.8
Df Residuals:          383             BIC:              391.6
Df Model:               8
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.5350	0.037	-14.531	0.000	-0.607	-0.463
oilshock[T.1]	0.5913	0.042	14.071	0.000	0.509	0.674
horsepower	-0.2801	0.051	-5.456	0.000	-0.381	-0.179
oilshock[T.1]:horsepower	-0.2276	0.088	-2.598	0.010	-0.400	-0.055
weight	-0.4591	0.050	-9.226	0.000	-0.557	-0.361
oilshock[T.1]:weight	-0.0963	0.082	-1.181	0.238	-0.257	0.064
origin_Europe	0.1804	0.059	3.039	0.003	0.064	0.297
origin_Japan	0.1929	0.060	3.209	0.001	0.075	0.311

horsepower:weight	0.1715	0.023	7.403	0.000	0.126	0.217
-------------------	--------	-------	-------	-------	-------	-------

```
=====
Omnibus:                23.934    Durbin-Watson:                1.456
Prob(Omnibus):          0.000    Jarque-Bera (JB):        43.610
Skew:                   0.377    Prob(JB):                3.39e-10
Kurtosis:               4.450    Cond. No.                11.3
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	750.563317	2.852400e-92
horsepower	1.0	165.048555	165.048555	1163.370934	3.985476e-118
oilshock:horsepower	1.0	19.155472	19.155472	135.020381	6.081794e-27
weight	1.0	35.071635	35.071635	247.207986	2.471356e-43
oilshock:weight	1.0	0.468549	0.468549	3.302640	6.994992e-02
origin_Europe	1.0	0.971997	0.971997	6.851278	9.208817e-03
origin_Japan	1.0	2.687942	2.687942	18.946386	1.727234e-05
horsepower:weight	1.0	7.776131	7.776131	54.811293	8.507915e-13
Residual	383.0	54.336579	0.141871	NaN	NaN

```
formula = " + ".join(cols)
formula += " + " + "oilshock: horsepower"
formula += " + " + "origin_Europe: horsepower"
formula += " + " + "origin_Japan: horsepower"
formula += " + " + "origin_Europe: weight"
formula += " + " + "origin_Japan: weight"
formula += " + " + "oilshock: weight"
formula += " + " + "oilshock: horsepower"
results = perform_analysis("mpg", formula, Auto_os)
origin_interactions = results
```

#### OLS Regression Results

```
=====
Dep. Variable:                mpg    R-squared:                0.855
Model:                        OLS    Adj. R-squared:          0.851
Method:                        Least Squares    F-statistic:            204.1
Date:                          Sat, 28 Sep 2024    Prob (F-statistic):      6.22e-152
Time:                          04:09:24    Log-Likelihood:         -177.44
No. Observations:              392    AIC:                    378.9
Df Residuals:                  380    BIC:                    426.5
Df Model:                      11
```

Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.4245	0.035	-12.121	0.000	-0.493	-0.356
oilshock[T.1]	0.5690	0.044	13.054	0.000	0.483	0.655
horsepower	-0.0708	0.048	-1.470	0.142	-0.166	0.024
oilshock[T.1]:horsepower	-0.1615	0.096	-1.687	0.092	-0.350	0.027
weight	-0.4713	0.054	-8.712	0.000	-0.578	-0.365
oilshock[T.1]:weight	-0.2333	0.087	-2.694	0.007	-0.404	-0.063
origin_Europe	0.0297	0.082	0.363	0.717	-0.131	0.191
origin_Japan	-0.0010	0.131	-0.007	0.994	-0.259	0.257
origin_Europe:horsepower	-0.5852	0.130	-4.515	0.000	-0.840	-0.330
origin_Japan:horsepower	-0.2801	0.204	-1.370	0.172	-0.682	0.122
origin_Europe:weight	0.1640	0.120	1.370	0.171	-0.071	0.399
origin_Japan:weight	-0.1326	0.245	-0.541	0.589	-0.614	0.349
=====						
Omnibus:	20.717	Durbin-Watson:			1.585	
Prob(Omnibus):	0.000	Jarque-Bera (JB):			32.838	
Skew:	0.373	Prob(JB):			7.40e-08	
Kurtosis:	4.205	Cond. No.			25.3	
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	712.972849	3.365342e-89
horsepower	1.0	165.048555	165.048555	1105.105820	1.586816e-114
oilshock:horsepower	1.0	19.155472	19.155472	128.258155	8.134152e-26
weight	1.0	35.071635	35.071635	234.827067	1.302452e-41
oilshock:weight	1.0	0.468549	0.468549	3.137234	7.732495e-02
origin_Europe	1.0	0.971997	0.971997	6.508146	1.112920e-02
origin_Japan	1.0	2.687942	2.687942	17.997494	2.781984e-05
origin_Europe:horsepower	1.0	3.024522	3.024522	20.251113	9.040705e-06
origin_Japan:horsepower	1.0	1.977640	1.977640	13.241566	3.116551e-04
origin_Europe:weight	1.0	0.313437	0.313437	2.098659	1.482531e-01
origin_Japan:weight	1.0	0.043767	0.043767	0.293050	5.885897e-01
Residual	380.0	56.753344	0.149351	NaN	NaN

- From the above analysis, we can see that there is no significant interaction between origin and weight.
- So we can omit them from the model.

```

formula = " + ".join(cols)
formula += " + " + "oilshock: horsepower"
formula += " + " + "origin_Europe: horsepower"
formula += " + " + "origin_Japan: horsepower"
formula += " + " + "oilshock: weight"
formula += " + " + "oilshock: horsepower"
results = perform_analysis("mpg", formula, Auto_os)
origin_interactions = results

```

#### OLS Regression Results

```

=====
Dep. Variable:          mpg      R-squared:          0.854
Model:                  OLS      Adj. R-squared:      0.851
Method:                 Least Squares      F-statistic:      248.9
Date:                  Sat, 28 Sep 2024      Prob (F-statistic):      8.18e-154
Time:                  04:09:24      Log-Likelihood:      -178.67
No. Observations:      392      AIC:          377.3
Df Residuals:          382      BIC:          417.1
Df Model:              9
Covariance Type:      nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.4296	0.034	-12.650	0.000	-0.496	-0.363
oilshock[T.1]	0.5693	0.043	13.280	0.000	0.485	0.654
horsepower	-0.0797	0.047	-1.689	0.092	-0.172	0.013
oilshock[T.1]:horsepower	-0.1958	0.092	-2.133	0.034	-0.376	-0.015
weight	-0.4571	0.051	-8.947	0.000	-0.558	-0.357
oilshock[T.1]:weight	-0.2019	0.084	-2.408	0.016	-0.367	-0.037
origin_Europe	0.0044	0.080	0.055	0.956	-0.153	0.162
origin_Japan	0.0750	0.084	0.892	0.373	-0.090	0.240
origin_Europe:horsepower	-0.4667	0.096	-4.884	0.000	-0.655	-0.279
origin_Japan:horsepower	-0.3682	0.101	-3.637	0.000	-0.567	-0.169

```

=====
Omnibus:                20.114      Durbin-Watson:          1.577
Prob(Omnibus):          0.000      Jarque-Bera (JB):      31.613
Skew:                   0.366      Prob(JB):              1.37e-07
Kurtosis:               4.183      Cond. No.              10.2
=====

```

#### Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	712.242504	2.550768e-89
horsepower	1.0	165.048555	165.048555	1103.973788	9.879451e-115
oilshock:horsepower	1.0	19.155472	19.155472	128.126771	8.220841e-26
weight	1.0	35.071635	35.071635	234.586518	1.270254e-41
oilshock:weight	1.0	0.468549	0.468549	3.134021	7.747196e-02
origin_Europe	1.0	0.971997	0.971997	6.501479	1.116822e-02
origin_Japan	1.0	2.687942	2.687942	17.979058	2.804561e-05
origin_Europe:horsepower	1.0	3.024522	3.024522	20.230368	9.121050e-06
origin_Japan:horsepower	1.0	1.977640	1.977640	13.228002	3.136339e-04
Residual	382.0	57.110548	0.149504	NaN	NaN

- From the above analysis, it is evident that with the interaction between origin and horsepower, the interaction between oilshock and weight and horsepower is insignificant. We can drop these from the model as well.

```
formula = " + ".join(cols)
formula += " + " + "oilshock: horsepower"
formula += " + " + "origin_Europe: horsepower"
formula += " + " + "origin_Japan: horsepower"
results = perform_analysis("mpg", formula, Auto_os)
origin_interactions = results
models.append(
    {
        "name": "origin_interactions",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)
```

#### OLS Regression Results

```
=====
Dep. Variable:          mpg      R-squared:          0.852
Model:                  OLS      Adj. R-squared:       0.849
Method:                 Least Squares      F-statistic:       275.8
Date:                   Sat, 28 Sep 2024    Prob (F-statistic):   8.41e-154
Time:                   04:09:24           Log-Likelihood:    -181.63
No. Observations:      392             AIC:              381.3
Df Residuals:          383             BIC:              417.0
Df Model:               8
Covariance Type:       nonrobust
=====
```



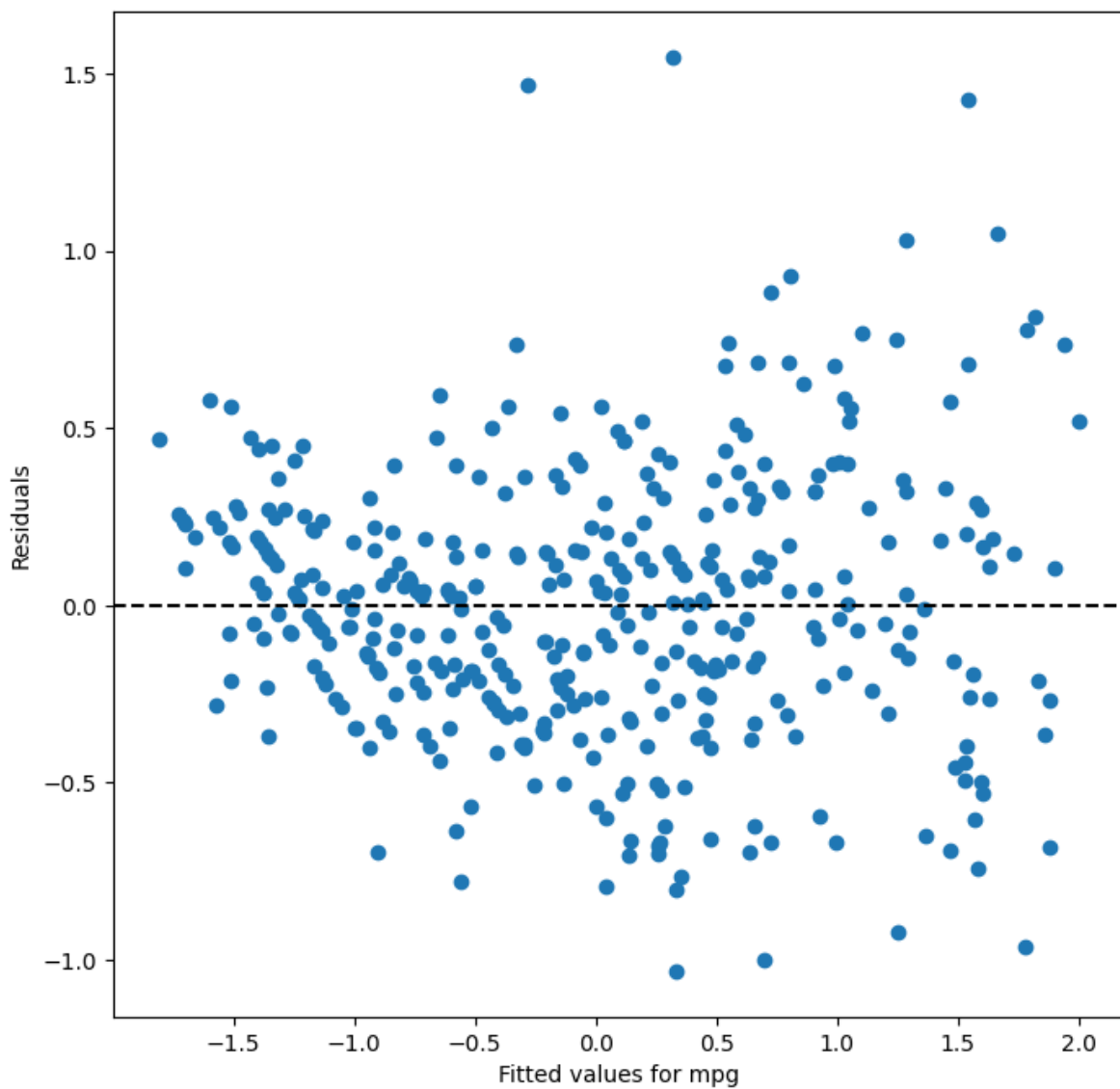
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.4267	0.034	-12.495	0.000	-0.494	-0.360
oilshock[T.1]	0.5628	0.043	13.073	0.000	0.478	0.647
horsepower	-0.0265	0.042	-0.633	0.527	-0.109	0.056
oilshock[T.1]:horsepower	-0.3804	0.051	-7.497	0.000	-0.480	-0.281
weight	-0.5231	0.043	-12.051	0.000	-0.608	-0.438
origin_Europe	0.0041	0.081	0.051	0.959	-0.155	0.163
origin_Japan	0.0877	0.084	1.039	0.299	-0.078	0.254
origin_Europe:horsepower	-0.4423	0.096	-4.626	0.000	-0.630	-0.254
origin_Japan:horsepower	-0.3589	0.102	-3.526	0.000	-0.559	-0.159
=====						
Omnibus:	19.159	Durbin-Watson:		1.576		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		29.046		
Skew:	0.362	Prob(JB):		4.93e-07		
Kurtosis:	4.119	Cond. No.		9.30		
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	703.425918	9.826044e-89
horsepower	1.0	165.048555	165.048555	1090.308104	4.259700e-114
oilshock:horsepower	1.0	19.155472	19.155472	126.540737	1.468876e-25
weight	1.0	35.071635	35.071635	231.682658	2.992243e-41
origin_Europe	1.0	0.792887	0.792887	5.237801	2.264506e-02
origin_Japan	1.0	2.840217	2.840217	18.762427	1.893486e-05
origin_Europe:horsepower	1.0	2.748574	2.748574	18.157033	2.563625e-05
origin_Japan:horsepower	1.0	1.881783	1.881783	12.431031	4.733982e-04
Residual	383.0	57.977737	0.151378	NaN	NaN

```
display_residuals_plot(results)
```



```
anova_lm(simple_model, numeric_interactions, oilshock_interactions, origin_interactions)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	386.0	76.594741	0.0	NaN	NaN	NaN
1	385.0	59.887237	1.0	16.707504	110.369506	7.218441e-23
2	383.0	54.336579	2.0	5.550658	18.333780	2.489897e-08
3	383.0	57.977737	-0.0	-3.641158	inf	NaN

```
pd.DataFrame(models)
```

	name	model
0	simple_model	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan
1	numeric_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + horsepower
2	oilshock_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + horsepower
3	origin_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + oilshock

**(f) Try a few different transformations of the variables, such as  $\log(X)$ ,  $\sqrt{X}$ ,  $X^2$  . Comment on your findings.**

```
formula = simple_model.model.formula
formula = formula[formula.rindex("~") + 1 : ]
# Add higher order transformations for horsepower and weight
formula += " + " + "I(horsepower**2)"
formula += " + " + "I(weight**2)"
results = perform_analysis("mpg", formula, Auto_os)
squared_transformations = results
models.append(
    {
        "name": "squared_transformation",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)
```

#### OLS Regression Results

```
=====
Dep. Variable:          mpg      R-squared:                0.848
Model:                  OLS      Adj. R-squared:           0.846
Method:                 Least Squares      F-statistic:        306.8
Date:                  Sat, 28 Sep 2024      Prob (F-statistic):    5.75e-153
Time:                  04:09:25      Log-Likelihood:       -186.58
No. Observations:      392      AIC:                  389.2
Df Residuals:          384      BIC:                  420.9
Df Model:              7
Covariance Type:       nonrobust
=====
```

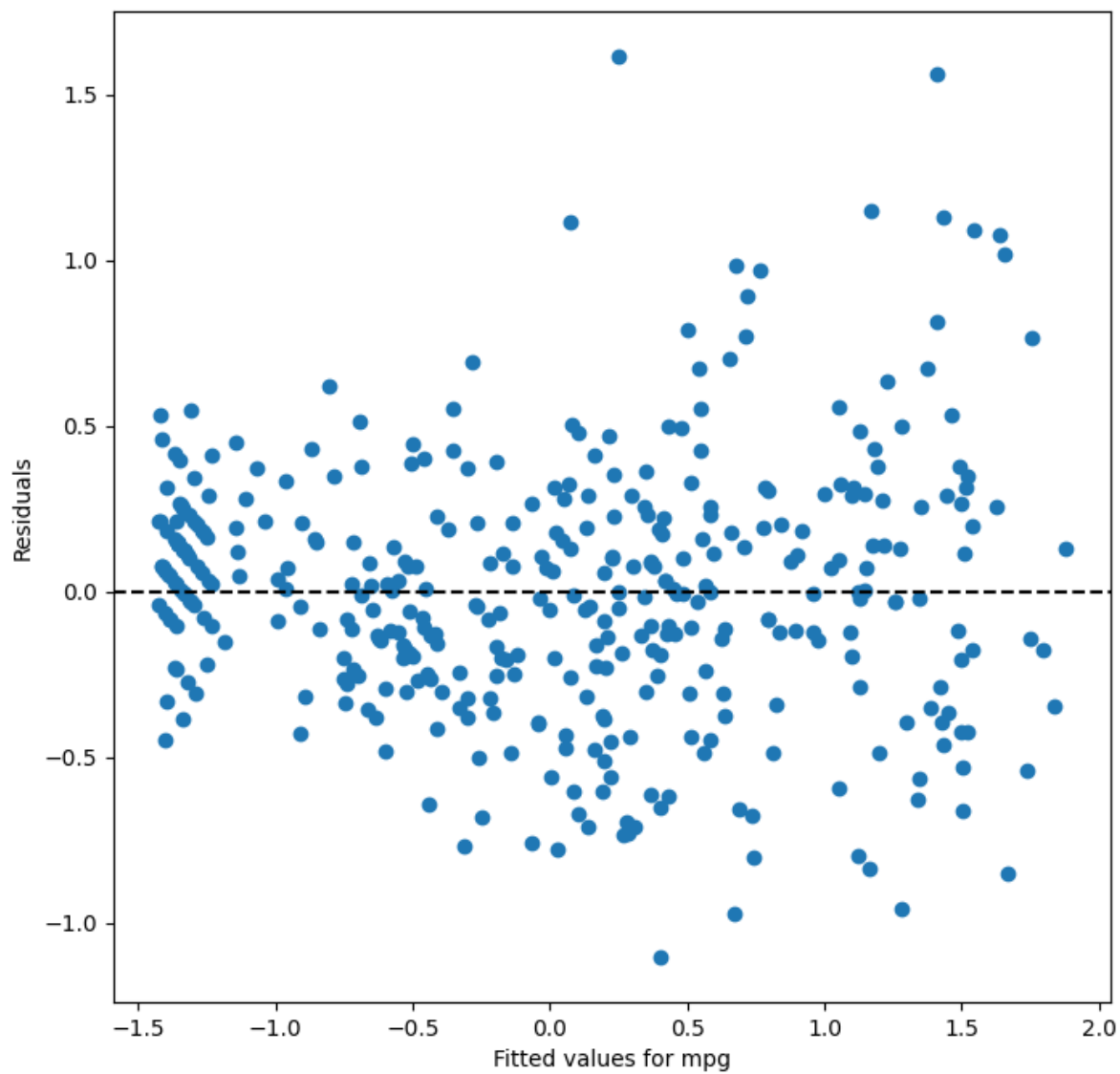
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-0.6020	0.040	-15.072	0.000	-0.681	-0.524
oilshock[T.1]	0.6580	0.043	15.334	0.000	0.574	0.742
horsepower	-0.3798	0.054	-7.058	0.000	-0.486	-0.274
weight	-0.5069	0.050	-10.044	0.000	-0.606	-0.408
origin_Europe	0.1436	0.062	2.324	0.021	0.022	0.265
origin_Japan	0.1959	0.064	3.047	0.002	0.069	0.322
I(horsepower ** 2)	0.0976	0.021	4.667	0.000	0.056	0.139
I(weight ** 2)	0.1413	0.026	5.382	0.000	0.090	0.193
=====						
Omnibus:	26.672	Durbin-Watson:		1.394		
Prob(Omnibus):	0.000	Jarque-Bera (JB):		46.435		
Skew:	0.436	Prob(JB):		8.26e-11		
Kurtosis:	4.443	Cond. No.		10.5		
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	106.483141	106.483141	687.677177	1.332036e-87
horsepower	1.0	165.048555	165.048555	1065.897602	7.775452e-113
weight	1.0	39.079210	39.079210	252.376864	4.864770e-44
origin_Europe	1.0	0.574647	0.574647	3.711118	5.478894e-02
origin_Japan	1.0	4.219706	4.219706	27.251220	2.932534e-07
I(horsepower ** 2)	1.0	12.648520	12.648520	81.685220	7.946432e-18
I(weight ** 2)	1.0	4.485871	4.485871	28.970134	1.281966e-07
Residual	384.0	59.460351	0.154845	NaN	NaN

```
display_residuals_plot(results)
```



```
anova_lm(simple_model, squared_transformations)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	386.0	76.594741	0.0	NaN	NaN	NaN
1	384.0	59.460351	2.0	17.134391	55.327677	7.681995e-22

```
pd.DataFrame(models)
```

	name	model
0	simple_model	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan
1	numeric_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + horsepower:weight
2	oilshock_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + oilshock:weight
3	origin_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + oilshock:weight
4	squared_transformation	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + I(horsepower^2 + weight^2 + oilshock^2)

- Since we've standardized the variables, we cannot run log or square root transformations on the negative valued columns.
- We can reload the data and run the log and sqrt transformations on the original un-standardized data.

```
Auto = load_data("Auto")
Auto = Auto.sort_values(by=["year"], ascending=True)
Auto.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year', 'origin'])
```

```
print("Minimums:")
print(Auto.min())
print("Maximums:")
print(Auto.max())
```

```
Minimums:
mpg          9.0
cylinders     3.0
displacement  68.0
horsepower    46.0
weight      1613.0
acceleration   8.0
year         70.0
origin        1.0
dtype: float64
Maximums:
mpg          46.6
cylinders     8.0
displacement 455.0
```

```
horsepower      230.0
weight          5140.0
acceleration    24.8
year            82.0
origin          3.0
dtype: float64
```

- From the above, we can see that the values for displacement, horsepower and weight are quite large.
- Hence, we log or square root transform only these variables.

### Now let's categorize the variables

```
Auto["origin"] = Auto["origin"].astype("category")
Auto["origin"] = Auto["origin"].cat.rename_categories(
    {1: "America", 2: "Europe", 3: "Japan"}
)
Auto["year"] = Auto["year"].astype("category")
Auto["oilshock"] = Auto.apply(categorize_for_oil_shock, axis=1)
```

### Log Transformed Model

```
Auto_log = Auto.copy(deep=True)
```

```
Auto_log["log_displacement"] = np.log(Auto_log["displacement"])
Auto_log["log_horsepower"] = np.log(Auto_log["horsepower"])
Auto_log["log_weight"] = np.log(Auto_log["weight"])
Auto_log = Auto_log.drop(
    columns=[
        "displacement",
        "weight",
        "horsepower",
        "year",
    ]
)
Auto_log.columns
```

```
Index(['mpg', 'cylinders', 'acceleration', 'origin', 'oilshock', 'log_displacement', 'log_ho
```

```
Auto_log.corr(numeric_only=True)
```

	mpg	cylinders	acceleration	oilshock	log_displacement	log_horsepower	log_weight
mpg	1.000000	-0.777618	0.423329	0.521192	-0.828453	-0.817517	-0.844194
cylinders	-0.777618	1.000000	-0.504683	-0.273703	0.942814	0.843204	0.884303
acceleration	0.423329	-0.504683	1.000000	0.195892	-0.497107	-0.698328	-0.401563
oilshock	0.521192	-0.273703	0.195892	1.000000	-0.268161	-0.299037	-0.250520
log_displacement	-0.828453	0.942814	-0.497107	-0.268161	1.000000	0.872149	0.942850
log_horsepower	-0.817517	0.843204	-0.698328	-0.299037	0.872149	1.000000	0.873956
log_weight	-0.844194	0.884303	-0.401563	-0.250520	0.942850	0.873956	1.000000

```
Auto_log = pd.get_dummies(
    Auto_log, columns=list(["origin"]), drop_first=True, dtype=np.uint8
)
Auto_log.columns
```

```
Index(['mpg', 'cylinders', 'acceleration', 'oilshock', 'log_displacement', 'log_horsepower',
```

```
cols = list(Auto_log.columns)
cols.remove("mpg")
```

```
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_log)
vifdf
```

Feature	VIF
cylinders	9.828626
acceleration	3.304749
oilshock	1.147770
log_displacement	25.969595
log_horsepower	11.414709
log_weight	16.146573
origin_Europe	1.876698
origin_Japan	2.097688

```
identify_highest_VIF_feature(vifdf)
```



We find the highest VIF in this model is log\_displacement with a VIF of 25.96959512578754  
Hence, we drop log\_displacement from the model to be fitted.

```
('log_displacement', 25.96959512578754)
```

```
cols.remove("log_displacement")
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_log)
vifdf
```

	VIF
Feature	
cylinders	5.535070
acceleration	3.179336
oilshock	1.142791
log_horsepower	11.411764
log_weight	10.608718
origin_Europe	1.451961
origin_Japan	1.652749

```
identify_highest_VIF_feature(vifdf)
```

We find the highest VIF in this model is log\_horsepower with a VIF of 11.411764499222897  
Hence, we drop log\_horsepower from the model to be fitted.

```
('log_horsepower', 11.411764499222897)
```

```
cols.remove("log_horsepower")
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_log)
vifdf
```

	VIF
Feature	
cylinders	5.517868
acceleration	1.377517
oilshock	1.118666
log_weight	5.014899
origin_Europe	1.451265

VIF	
Feature	
origin_Japan	1.608682

```
identify_highest_VIF_feature(vifdf)
```

No variables are significantly collinear.

```
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_log)
```

#### OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.823
Model:	OLS	Adj. R-squared:	0.821
Method:	Least Squares	F-statistic:	299.3
Date:	Sat, 28 Sep 2024	Prob (F-statistic):	1.36e-141
Time:	04:09:28	Log-Likelihood:	-1021.3
No. Observations:	392	AIC:	2057.
Df Residuals:	385	BIC:	2084.
Df Model:	6		
Covariance Type:	nonrobust		
=====			
	coef	std err	t
			P> t
			[0.025
			0.975]
Intercept	168.2416	9.696	17.351
cylinders	0.0129	0.230	0.056
acceleration	0.1805	0.071	2.538
oilshock	5.1312	0.355	14.470
log_weight	-18.9156	1.331	-14.211
origin_Europe	1.3692	0.531	2.578
origin_Japan	1.5602	0.528	2.956
=====			
Omnibus:	30.158	Durbin-Watson:	1.253
Prob(Omnibus):	0.000	Jarque-Bera (JB):	51.811
Skew:	0.493	Prob(JB):	5.62e-12
Kurtosis:	4.484	Cond. No.	1.08e+03
=====			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
 [2] The condition number is large, 1.08e+03. This might indicate that there are strong multicollinearity or other numerical problems.

	df	sum_sq	mean_sq	F	PR(>F)
cylinders	1.0	14403.083079	14403.083079	1318.559127	2.133814e-126
acceleration	1.0	30.471304	30.471304	2.789557	9.569322e-02
oilshock	1.0	2422.051542	2422.051542	221.731566	6.308582e-40
log_weight	1.0	2639.878573	2639.878573	241.672978	1.215817e-42
origin_Europe	1.0	22.569818	22.569818	2.066199	1.514086e-01
origin_Japan	1.0	95.449335	95.449335	8.738101	3.308129e-03
Residual	385.0	4205.489819	10.923350	NaN	NaN

```
identify_least_significant_feature(results, alpha=LOS_Alpha)
```

We find the least significant variable in this model is cylinders with a p-value of 0.955447.  
 Using the backward methodology, we suggest dropping cylinders from the new model

```
cols.remove("cylinders")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_log)
```

#### OLS Regression Results

```
=====
Dep. Variable:          mpg      R-squared:                0.823
Model:                  OLS      Adj. R-squared:           0.821
Method:                 Least Squares      F-statistic:         360.0
Date:                  Sat, 28 Sep 2024      Prob (F-statistic):      6.83e-143
Time:                  04:09:28      Log-Likelihood:         -1021.3
No. Observations:      392      AIC:                    2055.
Df Residuals:          386      BIC:                    2078.
Df Model:               5
Covariance Type:       nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	167.8689	7.032	23.870	0.000	154.042	181.696
acceleration	0.1792	0.067	2.673	0.008	0.047	0.311
oilshock	5.1289	0.352	14.574	0.000	4.437	5.821
log_weight	-18.8571	0.820	-22.992	0.000	-20.470	-17.245
origin_Europe	1.3628	0.518	2.631	0.009	0.344	2.381
origin_Japan	1.5576	0.525	2.967	0.003	0.525	2.590

```
=====
Omnibus:                 30.308    Durbin-Watson:                 1.253
Prob(Omnibus):           0.000    Jarque-Bera (JB):         52.282
Skew:                    0.493    Prob(JB):                 4.44e-12
Kurtosis:                4.492    Cond. No.                 751.
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
acceleration	1.0	4268.531557	4268.531557	391.783092	1.076057e-60
oilshock	1.0	4757.627552	4757.627552	436.674301	2.076230e-65
log_weight	1.0	10466.602734	10466.602734	960.667136	8.818191e-107
origin_Europe	1.0	24.823504	24.823504	2.278402	1.320051e-01
origin_Japan	1.0	95.884166	95.884166	8.800637	3.198611e-03
Residual	386.0	4205.523957	10.895140	NaN	NaN

```
identify_least_significant_feature(results, alpha=LOS_Alpha)
```

No variables are statistically insignificant.

The model mpg ~ acceleration + oilshock + log\_weight + origin\_Europe + origin\_Japan cannot be

```
models.append(
    {
        "name": "log_transformation",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)
```

```
pd.DataFrame(models)
```

	name	model
0	simple_model	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan
1	numeric_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + horsepower:weight
2	oilshock_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + oilshock:weight
3	origin_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + origin_Europe:origin_Japan
4	squared_transformation	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + I(horsepower**2)
5	log_transformation	mpg ~ acceleration + oilshock + log_weight + origin_Europe + origin_Japan

## Square Root Transformed Model

```
Auto_sqrt = Auto.copy(deep=True)
```

```
Auto_sqrt["sqrt_displacement"] = np.sqrt(Auto_sqrt["displacement"])
Auto_sqrt["sqrt_horsepower"] = np.sqrt(Auto_sqrt["horsepower"])
Auto_sqrt["sqrt_weight"] = np.sqrt(Auto_sqrt["weight"])
Auto_sqrt = Auto_sqrt.drop(
    columns=[
        "displacement",
        "weight",
        "horsepower",
        "year",
    ]
)
Auto_sqrt.columns
```

```
Index(['mpg', 'cylinders', 'acceleration', 'origin', 'oilshock', 'sqrt_displacement', 'sqrt_horsepower', 'sqrt_weight'], dtype='object', length=8)
```

```
Auto_sqrt.corr(numeric_only=True)
```

	mpg	cylinders	acceleration	oilshock	sqrt_displacement	sqrt_horsepower
mpg	1.000000	-0.777618	0.423329	0.521192	-0.821331	-0.802311
cylinders	-0.777618	1.000000	-0.504683	-0.273703	0.953208	0.849266
acceleration	0.423329	-0.504683	1.000000	0.195892	-0.521812	-0.696702
oilshock	0.521192	-0.273703	0.195892	1.000000	-0.284587	-0.306247
sqrt_displacement	-0.821331	0.953208	-0.521812	-0.284587	1.000000	0.886470
sqrt_horsepower	-0.802311	0.849266	-0.696702	-0.306247	0.886470	1.000000
sqrt_weight	-0.840095	0.893465	-0.409829	-0.260664	0.939868	0.872045

```
Auto_sqrt = pd.get_dummies(
    Auto_sqrt, columns=list(["origin"]), drop_first=True, dtype=np.uint8
)
Auto_sqrt.columns
```

```
Index(['mpg', 'cylinders', 'acceleration', 'oilshock', 'sqrt_displacement', 'sqrt_horsepower', 'sqrt_weight', 'origin_1', 'origin_2', 'origin_3'], dtype='object', length=11)
```

```
cols = list(Auto_sqrt.columns)
cols.remove("mpg")
```

```
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_sqrt)
vifdf
```

Feature	VIF
cylinders	11.465746
acceleration	3.010771
oilshock	1.151324
sqrt_displacement	27.042946
sqrt_horsepower	10.615281
sqrt_weight	13.450552
origin_Europe	1.774827
origin_Japan	1.944729

```
identify_highest_VIF_feature(vifdf)
```

We find the highest VIF in this model is sqrt\_displacement with a VIF of 27.042946454149405. Hence, we drop sqrt\_displacement from the model to be fitted.

```
('sqrt_displacement', 27.042946454149405)
```

```
cols.remove("sqrt_displacement")
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_sqrt)
vifdf
```

Feature	VIF
cylinders	5.974510
acceleration	2.934605
oilshock	1.141428
sqrt_horsepower	10.446261
sqrt_weight	9.963350
origin_Europe	1.450840
origin_Japan	1.623907

```
identify_highest_VIF_feature(vifdf)
```

We find the highest VIF in this model is `sqrt_horsepower` with a VIF of 10.446261176837464. Hence, we drop `sqrt_horsepower` from the model to be fitted.

```
('sqrt_horsepower', 10.446261176837464)
```

```
cols.remove("sqrt_horsepower")
vifdf = calculate_VIFs("mpg ~ " + " + ".join(cols), Auto_sqrt)
vifdf
```

	VIF
Feature	
cylinders	5.907717
acceleration	1.377206
oilshock	1.119726
sqrt_weight	5.331435
origin_Europe	1.446456
origin_Japan	1.581460

```
identify_highest_VIF_feature(vifdf)
```

No variables are significantly collinear.

```
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_sqrt)
```

```

                                OLS Regression Results
=====
Dep. Variable:                  mpg      R-squared:                  0.814
Model:                          OLS      Adj. R-squared:              0.811
Method:                        Least Squares      F-statistic:                281.0
Date:                          Sat, 28 Sep 2024      Prob (F-statistic):          2.76e-137
Time:                          04:09:30      Log-Likelihood:              -1031.4
No. Observations:                392      AIC:                        2077.
Df Residuals:                    385      BIC:                        2105.
Df Model:                        6
Covariance Type:                nonrobust

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	54.3622	2.396	22.687	0.000	49.651	59.073
cylinders	0.0148	0.244	0.061	0.952	-0.466	0.495
acceleration	0.1748	0.073	2.395	0.017	0.031	0.318
oilshock	5.0506	0.364	13.873	0.000	4.335	5.766
sqrt_weight	-0.6785	0.052	-13.130	0.000	-0.780	-0.577
origin_Europe	1.5511	0.544	2.851	0.005	0.481	2.621
origin_Japan	1.9033	0.537	3.544	0.000	0.847	2.959
Omnibus:		25.773	Durbin-Watson:			1.269
Prob(Omnibus):		0.000	Jarque-Bera (JB):			41.514
Skew:		0.449	Prob(JB):			9.67e-10
Kurtosis:		4.317	Cond. No.			803.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
cylinders	1.0	14403.083079	14403.083079	1252.209748	4.496819e-123
acceleration	1.0	30.471304	30.471304	2.649187	1.044209e-01
oilshock	1.0	2422.051542	2422.051542	210.574120	2.284075e-38
sqrt_weight	1.0	2365.801178	2365.801178	205.683691	1.125346e-37
origin_Europe	1.0	24.780703	24.780703	2.154444	1.429743e-01
origin_Japan	1.0	144.484455	144.484455	12.561536	4.421781e-04
Residual	385.0	4428.321210	11.502133	NaN	NaN

```
identify_least_significant_feature(results, alpha=LOS_Alpha)
```

We find the least significant variable in this model is cylinders with a p-value of 0.951625. Using the backward methodology, we suggest dropping cylinders from the new model.

```
cols.remove("cylinders")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_sqrt)
```

#### OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.814
Model:	OLS	Adj. R-squared:	0.812



```

Method:          Least Squares    F-statistic:          338.0
Date:            Sat, 28 Sep 2024  Prob (F-statistic):       1.43e-138
Time:            04:09:30          Log-Likelihood:         -1031.4
No. Observations: 392            AIC:                     2075.
Df Residuals:    386            BIC:                     2099.
Df Model:        5
Covariance Type: nonrobust

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	54.3324	2.342	23.198	0.000	49.728	58.937
acceleration	0.1733	0.069	2.512	0.012	0.038	0.309
oilshock	5.0486	0.362	13.946	0.000	4.337	5.760
sqrt_weight	-0.6760	0.031	-21.968	0.000	-0.737	-0.616
origin_Europe	1.5438	0.530	2.913	0.004	0.502	2.586
origin_Japan	1.8999	0.534	3.561	0.000	0.851	2.949

```

Omnibus:          25.925    Durbin-Watson:          1.269
Prob(Omnibus):    0.000    Jarque-Bera (JB):         41.952
Skew:             0.450    Prob(JB):                 7.77e-10
Kurtosis:         4.326    Cond. No.                 783.

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
acceleration	1.0	4268.531557	4268.531557	372.068179	1.546709e-58
oilshock	1.0	4757.627552	4757.627552	414.700418	3.906846e-63
sqrt_weight	1.0	10191.062423	10191.062423	888.307839	3.798699e-102
origin_Europe	1.0	27.910362	27.910362	2.432817	1.196386e-01
origin_Japan	1.0	145.497978	145.497978	12.682387	4.152294e-04
Residual	386.0	4428.363596	11.472445	NaN	NaN

```
identify_least_significant_feature(results, alpha=LOS_Alpha)
```

We find the least significant variable in this model is acceleration with a p-value of 0.012. Using the backward methodology, we suggest dropping acceleration from the new model

```

cols.remove("acceleration")
formula = " + ".join(cols)
results = perform_analysis("mpg", formula, Auto_sqrt)

```

# OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.811			
Model:	OLS	Adj. R-squared:	0.809			
Method:	Least Squares	F-statistic:	415.3			
Date:	Sat, 28 Sep 2024	Prob (F-statistic):	1.50e-138			
Time:	04:09:31	Log-Likelihood:	-1034.6			
No. Observations:	392	AIC:	2079.			
Df Residuals:	387	BIC:	2099.			
Df Model:	4					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
Intercept	58.2668	1.753	33.230	0.000	54.819	61.714
oilshock	5.1556	0.362	14.244	0.000	4.444	5.867
sqrt_weight	-0.6999	0.029	-23.759	0.000	-0.758	-0.642
origin_Europe	1.6530	0.532	3.108	0.002	0.607	2.699
origin_Japan	1.8263	0.536	3.405	0.001	0.772	2.881
=====						
Omnibus:	31.883	Durbin-Watson:	1.241			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	60.472			
Skew:	0.483	Prob(JB):	7.39e-14			
Kurtosis:	4.664	Cond. No.	574.			
=====						

## Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

	df	sum_sq	mean_sq	F	PR(>F)
oilshock	1.0	6470.207217	6470.207217	556.341155	7.004177e-77
sqrt_weight	1.0	12674.256790	12674.256790	1089.796728	1.354903e-114
origin_Europe	1.0	38.907051	38.907051	3.345425	6.816142e-02
origin_Japan	1.0	134.840521	134.840521	11.594270	7.307851e-04
Residual	387.0	4500.781890	11.629927	NaN	NaN

```
models.append(
    {
        "name": "sqrt_transformation",
        "model": results.model.formula,
        "R-squared adjusted": results.rsquared_adj,
    }
)
```

```
pd.DataFrame(models)
```

	name	model
0	simple_model	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan
1	numeric_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + hor
2	oilshock_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + hor
3	origin_interactions	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + oils
4	squared_transformation	mpg ~ horsepower + weight + oilshock + origin_Europe + origin_Japan + I(h
5	log_transformation	mpg ~ acceleration + oilshock + log_weight + origin_Europe + origin_Japan
6	sqrt_transformation	mpg ~ oilshock + sqrt_weight + origin_Europe + origin_Japan

```
allDone()
```

```
<IPython.lib.display.Audio object>
```