Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import math
```

Importing Dataset

```
dataset=pd.read_csv('car performance.csv')
dataset
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl

Finding missing data

```
dataset.isnull().any()
```

```
mpg             False
cylinders       False
displacement    False
horsepower      False
```

```
weight          False
acceleration    False
model year      False
origin          False
car name        False
dtype: bool
```

There are no null characters in the columns but there is a special character '?' in the 'horsepower' column. So we we replaced '?' with nan and replaced nan values with mean of the column.

```
dataset['horsepower']=dataset['horsepower'].replace('?',np.nan)
```

```
dataset['horsepower'].isnull().sum()
```

```
0
```

```
dataset['horsepower']=dataset['horsepower'].astype('float64')
```

```
dataset['horsepower'].fillna((dataset['horsepower'].mean()),inplace=True)
```

```
dataset.isnull().any()
```

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model year      False
origin          False
car name        False
dtype: bool
```

```
#Pandas dataframe.info() function is used to get a quick overview of the dataset.
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

```
#Pandas describe() is used to view some basic statistical details of a data frame or a series of nu
dataset.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.165829	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.298676	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	92.000000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.175000	79.000000

There is no use with car name attribute so drop it

```
#dropping the unwanted column.
dataset=dataset.drop('car name',axis=1)
```

```
#Pandas dataframe.corr() is used to find the pairwise correlation of all columns in the dataframe.
corr_table=dataset.corr()
corr_table
```

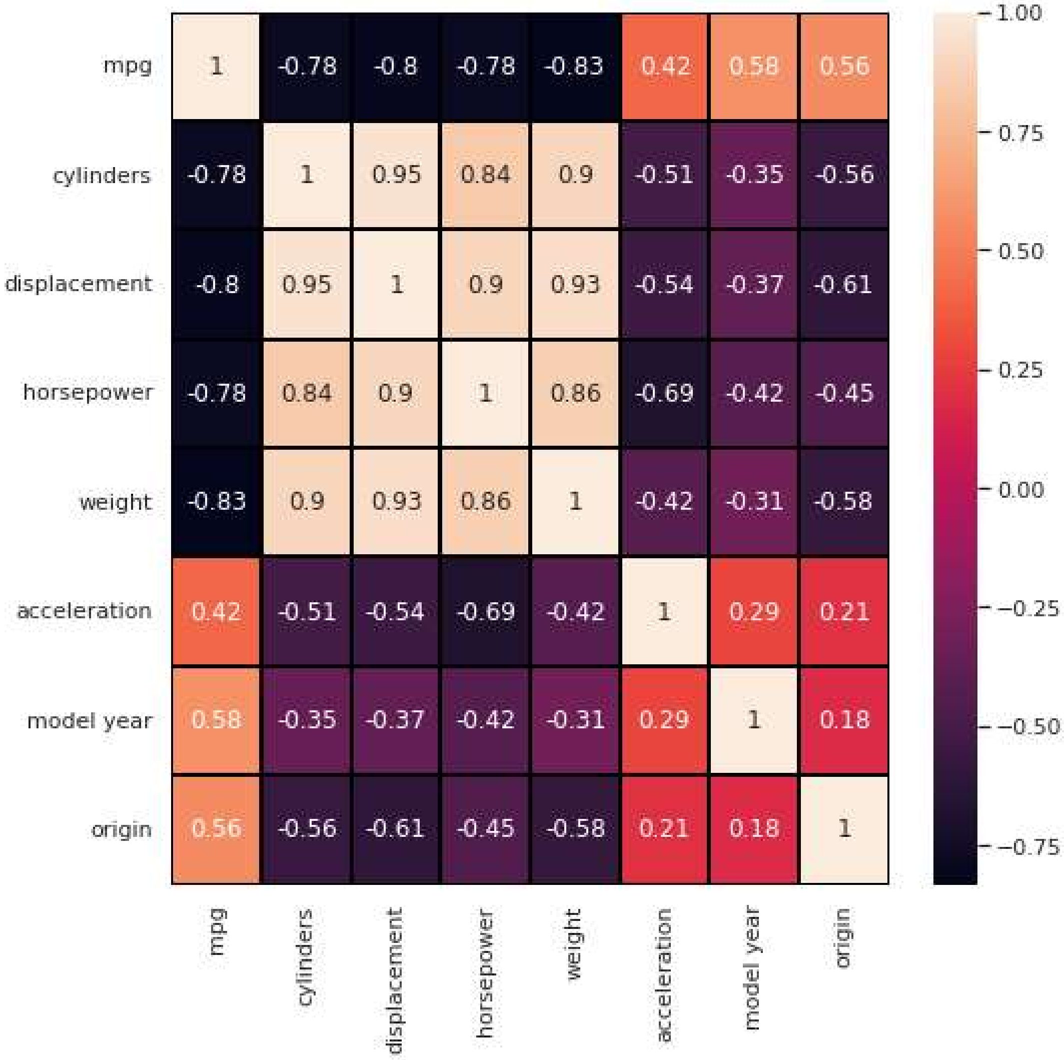
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
mpg	1.000000	-0.775396	-0.804203	-0.777501	-0.831741	0.420289	0.579267
cylinders	-0.775396	1.000000	0.950721	0.842437	0.896017	-0.505419	-0.348746
displacement	-0.804203	0.950721	1.000000	0.897082	0.932824	-0.543684	-0.370164
horsepower	-0.777501	0.842437	0.897082	1.000000	0.863990	-0.686436	-0.417081
weight	-0.831741	0.896017	0.932824	0.863990	1.000000	-0.417457	-0.306564
acceleration	0.420289	-0.505419	-0.543684	-0.686436	-0.417457	1.000000	0.288137
model year	0.579267	-0.348746	-0.370164	-0.417081	-0.306564	0.288137	1.000000

Data Visualizations

Heatmap : which represents correlation between attributes

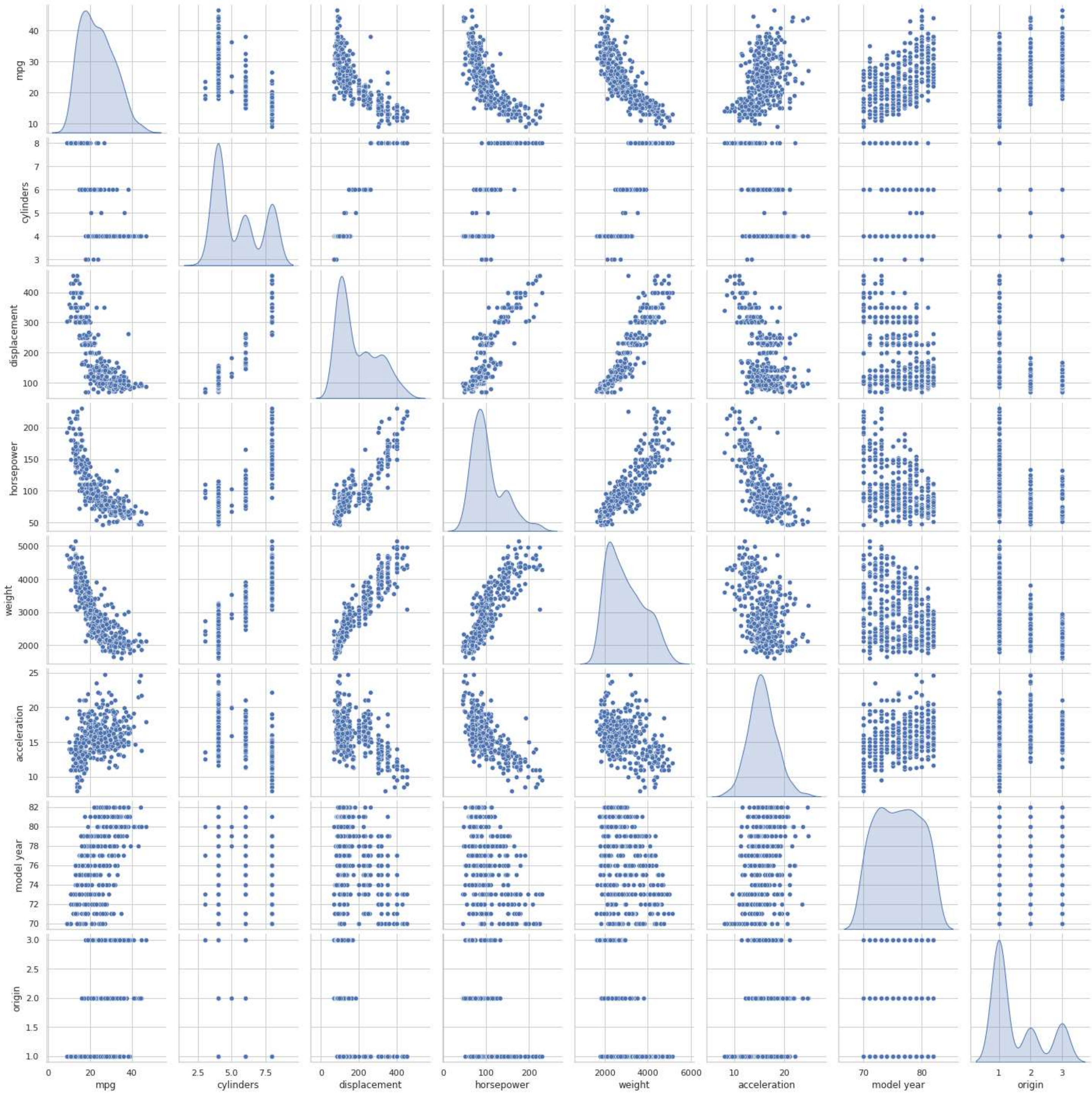
Double-click (or enter) to edit

```
#Heatmap is a way to show some sort of matrix plot,annot is used for correlation.
sns.heatmap(dataset.corr(),annot=True,linecolor ='black', linewidths = 1)
fig=plt.gcf()
fig.set_size_inches(8,8)
```



Visualizations of each attributes w.r.t rest of all attributes

```
#pairplot represents pairwise relation across the entire dataframe.
sns.pairplot(dataset,diag_kind='kde')
plt.show()
```



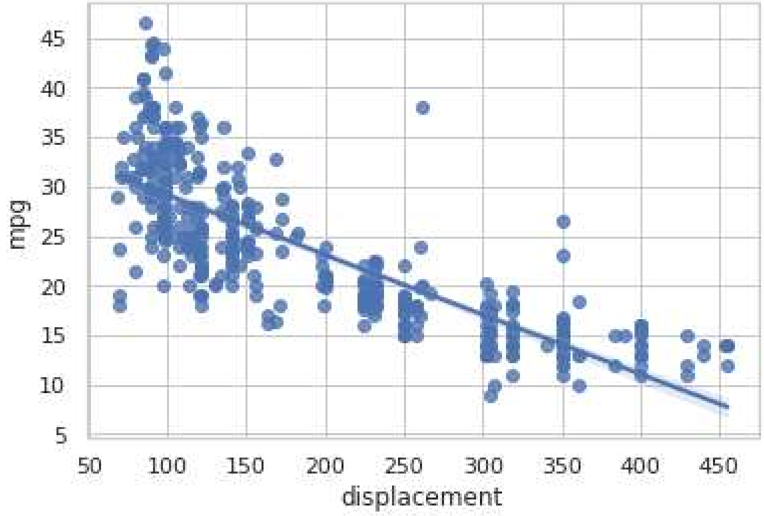
Regression plots(regplot()) creates a regression line between 2 parameters and helps to visualize their linear relationships.

```
sns.regplot(x="cylinders", y="mpg", data=dataset)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc037c3d0>

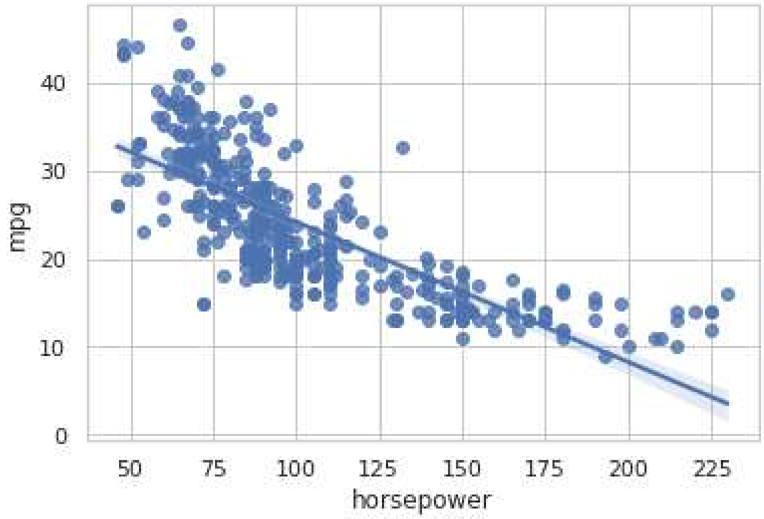
```
sns.regplot(x="displacement", y="mpg", data=dataset)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc03b2490>




```
sns.regplot(x="horsepower", y="mpg", data=dataset)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5bbfedd590>

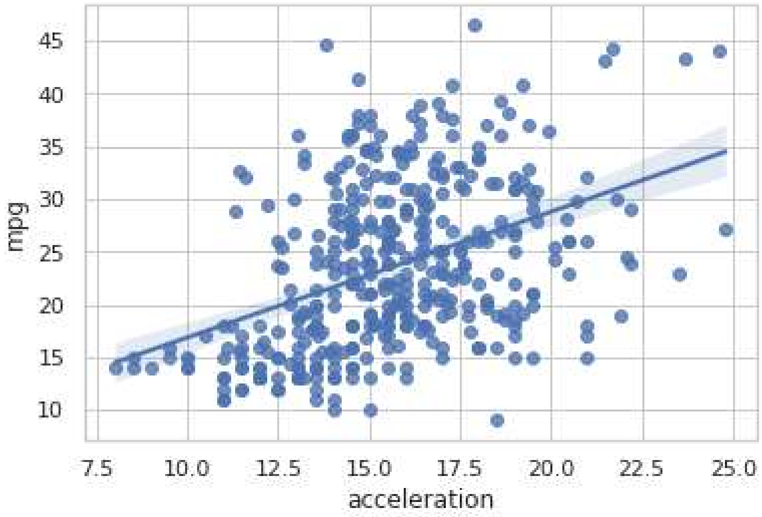



```
sns.regplot(x="weight", y="mpg", data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc0460190>
```

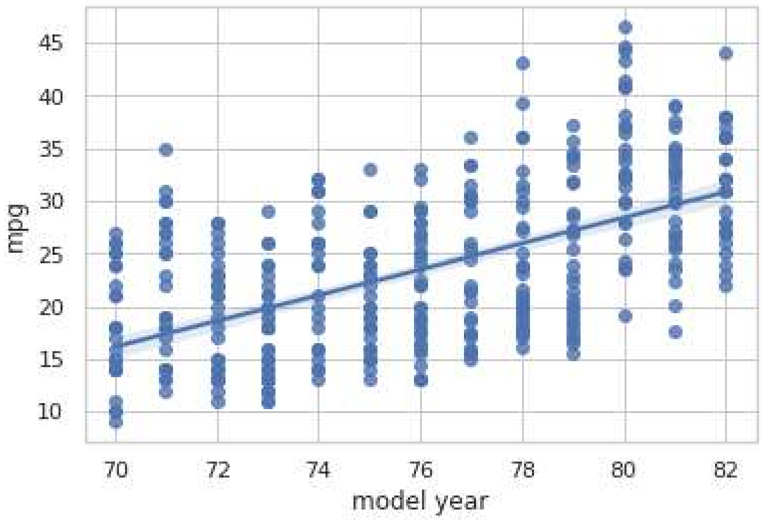
```
sns.regplot(x="acceleration", y="mpg", data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc0b50dd0>
```



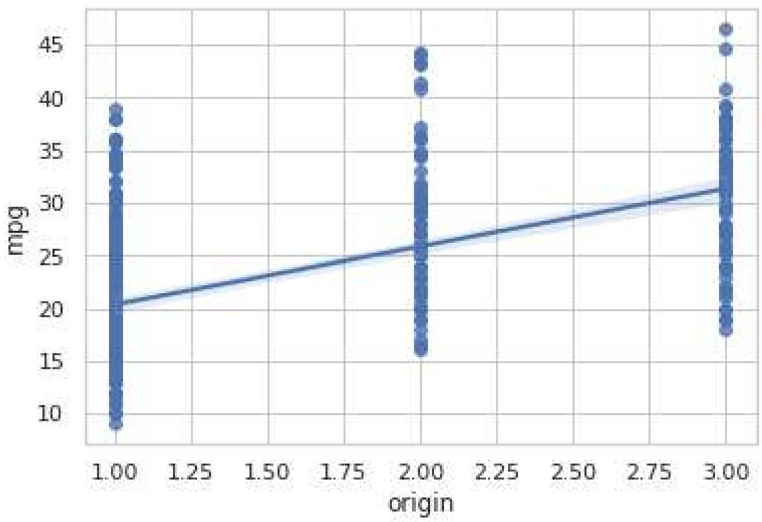

```
sns.regplot(x="model year", y="mpg", data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc04d38d0>
```




```
sns.regplot(x="origin", y="mpg", data=dataset)
```

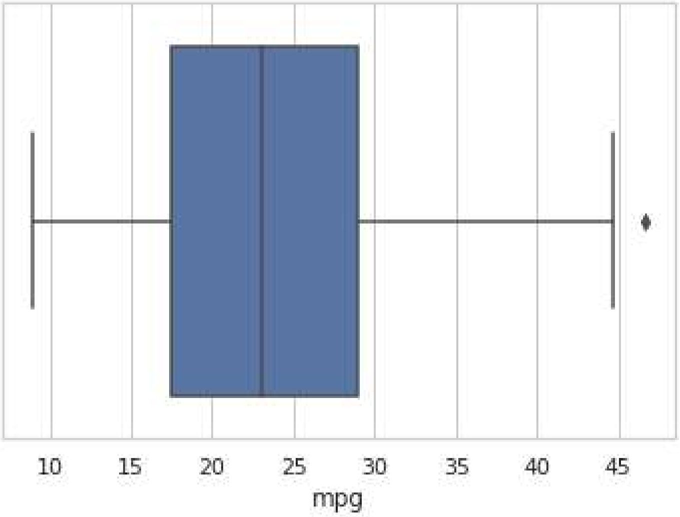
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc06a8190>
```




```
sns.set(style="whitegrid")
```

```
sns.boxplot(x=dataset["mpg"])
```

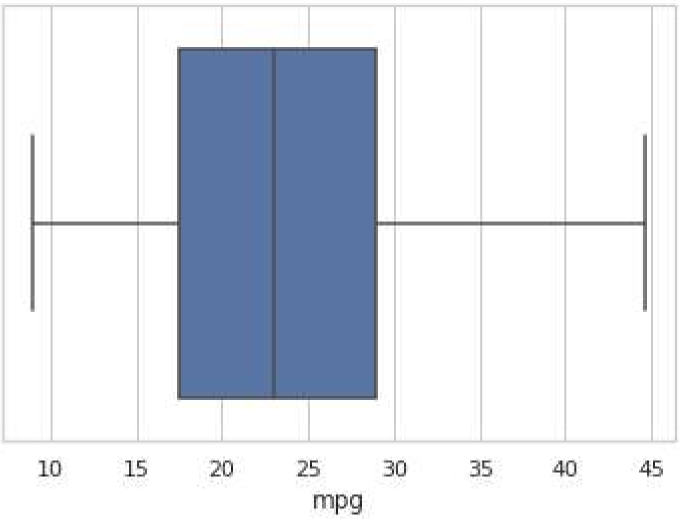
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bc0ad5e10>
```




```
#finding the interquartilerange of mpg column and replacing the outliers with mean
q1=dataset['mpg'].quantile(0.25)
q3=dataset['mpg'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
dataset['mpg']=np.where(dataset['mpg']>upper_bound,dataset['mpg'].mean(),np.where(dataset['mpg']<lo
```

```
#boxplot for mpg column
sns.boxplot(dataset['mpg'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the foll
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f5bbd23b050>
```



Finding quartiles for mgp

The P-value is the probability value that the correlation between these two variables is statistically significant.

Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

By convention, when the

- p-value is $<$ 0.001: we say there is strong evidence that the correlation is significant.
- the p-value is $<$ 0.05: there is moderate evidence that the correlation is significant.
- the p-value is $<$ 0.1: there is weak evidence that the correlation is significant.
- the p-value is $>$ 0.1: there is no evidence that the correlation is significant

```
from scipy import stats
```

Cylinders vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'Cylinders' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['cylinders'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.77764675219798  with a P-value of P = 7.858432477700
```

Conclusion:

Since the p-value is $<$ 0.001, the correlation between cylinders and mpg is statistically significant, and the coefficient of ~ -0.775 shows that the relationship is negative and moderately strong.

Displacement vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'Displacement' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['displacement'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.805459196737545  with a P-value of P = 5.30475981248
```

Conclusion:

Since the p-value is $<$ 0.1, the correlation between displacement and mpg is statistically significant, and the linear negative relationship is quite strong (~-0.809, close to -1)

Horsepower vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'horsepower' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['horsepower'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.7785160459711739  with a P-value of P = 3.9815800890
```

Conclusion:

Since the p-value is $<$ 0.001, the correlation between horsepower and mpg is statistically significant, and the coefficient of ~ -0.771 shows that the relationship is negative and moderately strong.

Weght vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'weight' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['weight'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.8334127823939774  with a P-value of P = 4.9210938207
```

Conclusion:

Since the p-value is $<$ 0.001, the correlation between weight and mpg is statistically significant, and the linear negative relationship is quite strong (~-0.831, close to -1)

Acceleration vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'Acceleration' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['acceleration'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.4186346290331828  with a P-value of P = 2.55443395965
```

Conclusion:

Since the p-value is $>$ 0.1, the correlation between acceleration and mpg is statistically significant, but the linear relationship is weak (~0.420).

Model year vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'Model year' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['model year'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.5776371269001912  with a P-value of P = 8.51521448588
```

Conclusion:

Since the p-value is $<$ 0.001, the correlation between model year and mpg is statistically significant, but the linear relationship is only moderate (~0.579).

Origin vs mpg

Let's calculate the Pearson Correlation Coefficient and P-value of 'Origin' and 'mpg'.

```
pearson_coef, p_value = stats.pearsonr(dataset['origin'], dataset['mpg'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.556374784654882  with a P-value of P = 1.006632653279
```

Conclusion:

Since the p-value is $<$ 0.001, the correlation between origin and mpg is statistically significant, but the linear relationship is only moderate (~0.563).

Ordinary Least Squares Statistics

```
test=smf.ols('mpg~cylinders+displacement+horsepower+weight+acceleration+origin',dataset).fit()
test.summary()
```

OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.721
Model:	OLS	**Adj. R-squared:**	0.716
Method:	Least Squares	**F-statistic:**	168.0

Inference as in the above summary the p value of the accelaration is maximum(i.e 0.972) so we can remove the acc variable from the dataset

Df Residuals:	391	**BIC:**	2291.

Seperating into Dependent and Independent variables

Intercept	43.8965	2.657	16.519	0.000	38.672	49.121

```
x=dataset[['cylinders','displacement','horsepower','weight','model year','origin']].values
x
```

```
array([[8.000e+00, 3.070e+02, 1.300e+02, 3.504e+03, 7.000e+01, 1.000e+00],
       [8.000e+00, 3.500e+02, 1.650e+02, 3.693e+03, 7.000e+01, 1.000e+00],
       [8.000e+00, 3.180e+02, 1.500e+02, 3.436e+03, 7.000e+01, 1.000e+00],
       ...,
       [4.000e+00, 1.350e+02, 8.400e+01, 2.295e+03, 8.200e+01, 1.000e+00],
       [4.000e+00, 1.200e+02, 7.900e+01, 2.625e+03, 8.200e+01, 1.000e+00],
       [4.000e+00, 1.190e+02, 8.200e+01, 2.720e+03, 8.200e+01, 1.000e+00]])
```

Kurtosis:	3.867	**Cond. No.**	4.00e+04

```
y=dataset.iloc[:,0:1].values
y
```

```
       [23.5      ],
       [30.       ],
       [39.1      ],
       [39.       ],
       [35.1      ],
       [32.3      ],
       [37.       ],
       [37.7      ],
       [34.1      ],
       [34.7      ],
       [34.4      ],
       [29.9      ],
       [33.       ],

       [34.5      ],
       [33.7      ],
       [32.4      ],
       [32.9      ],
       [31.6      ],
       [28.1      ],
       [30.7      ],
       [25.4      ],
       [24.2      ],
       [22.4      ],
       [26.6      ],
       [20.2      ],
       [17.6      ],
       [28.       ],
       [27.       ],
       [34.       ],
       [31.       ],
       [29.       ],
       [27.       ],
       [24.       ],
```

```
       [23.        ],
       [36.        ],
       [37.        ],
       [31.        ],
       [38.        ],
       [36.        ],
       [36.        ],
       [36.        ],
       [34.        ],
       [38.        ],
       [32.        ],
       [38.        ],
       [25.        ],
       [38.        ],
       [26.        ],
       [22.        ],
       [32.        ],
       [36.        ],
       [27.        ],
       [27.        ],
       [44.        ],
       [32.        ],
       [28.        ],
       [31.        ]])
```

Splitting into train and test data.

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
```

we are splitting as 90% train data and 10% test data