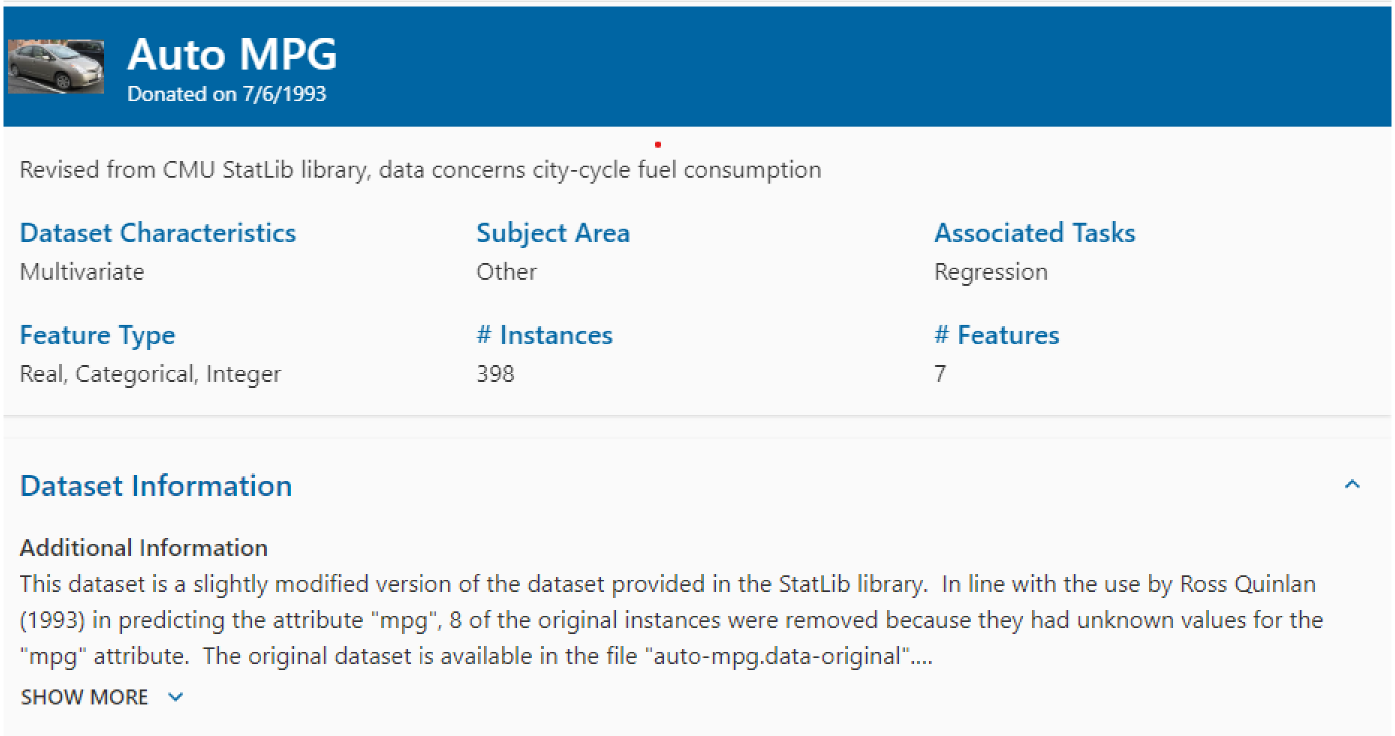
In [1]:
```python
from IPython.display import Image
Image(filename = "C:/Users/HP/Downloads/auto.png")
```

Out[1]:



Import Basic Packages

In [2]:
```python
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set()

import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```python
file_path = "C:/Users/HP/Downloads/mpg.txt"
with open(file_path,"r") as file:
    datadiscription = file.read()
print(datadiscription)
```

```
1. Title: Auto-Mpg Data

2. Sources:
   (a) Origin:  This dataset was taken from the StatLib library which is
                maintained at Carnegie Mellon University. The dataset was
                used in the 1983 American Statistical Association Exposition.
   (c) Date: July 7, 1993

3. Past Usage:
    -  See 2b (above)
    -  Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning.
       In Proceedings on the Tenth International Conference of Machine
       Learning, 236-243, University of Massachusetts, Amherst. Morgan
```

```
        Kaufmann.

4. Relevant Information:

   This dataset is a slightly modified version of the dataset provided in
   the StatLib library.  In line with the use by Ross Quinlan (1993) in
   predicting the attribute "mpg", 8 of the original instances were removed
   because they had unknown values for the "mpg" attribute.  The original
   dataset is available in the file "auto-mpg.data-original".

   "The data concerns city-cycle fuel consumption in miles per gallon,
    to be predicted in terms of 3 multivalued discrete and 5 continuous
    attributes." (Quinlan, 1993)

5. Number of Instances: 398

6. Number of Attributes: 9 including the class attribute

7. Attribute Information:

    1. mpg:           continuous
    2. cylinders:     multi-valued discrete
    3. displacement:  continuous
    4. horsepower:    continuous
    5. weight:        continuous
    6. acceleration:  continuous
    7. model year:    multi-valued discrete
    8. origin:        multi-valued discrete
    9. car name:      string (unique for each instance)

8. Missing Attribute Values:  horsepower has 6 missing values
```

In [4]:

```
data = pd.read_csv('mpg.csv')
data.head()
```

Out[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	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

In [5]:

```
data.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    object 
 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   name          398 non-null    object 
```

```
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

No missing values

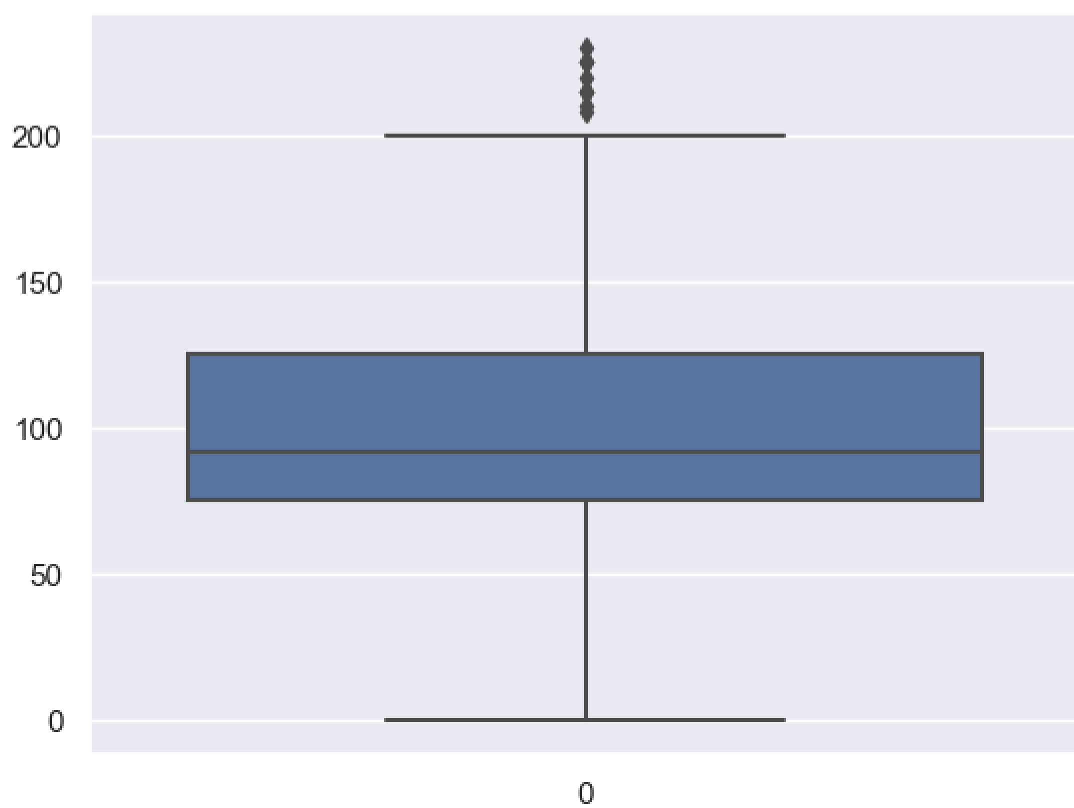
Handling char val

As horsepower is numerical value but here in dataset it is given object .. so we will convert this column into int

In [6]:
```python
def horsepwr(h):
    h = h.replace('?','0')
    h = int(h)
    return h
data['horsepower_new'] = data['horsepower'].map(horsepwr)
data.drop('horsepower', axis=1,inplace = True)
```

In [7]:
```python
sns.boxplot(data['horsepower_new'])
```

Out[7]:
```
<Axes: >
```



In [8]:
```python
data['horsepower_new'] = data['horsepower_new'].replace(0, data['horsepower_new'].median
```

In [9]:
```python
data.isnull().sum()
```

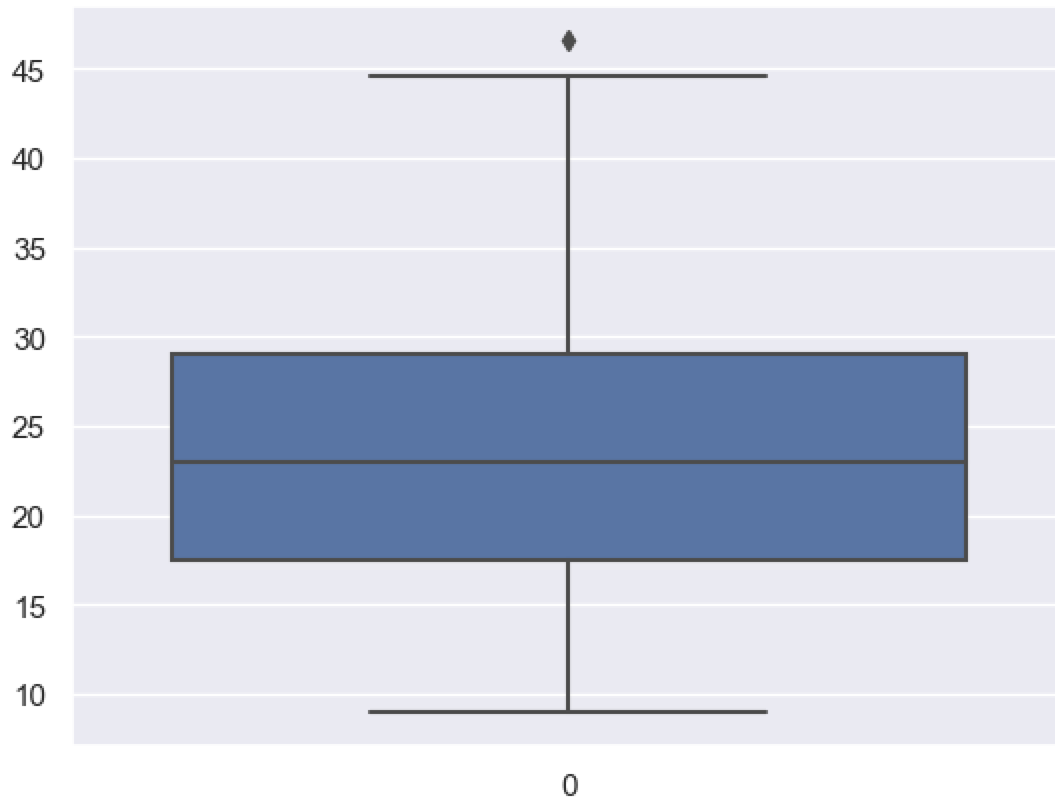
Out[9]:
```
mpg             0
cylinders       0
displacement    0
weight          0
```

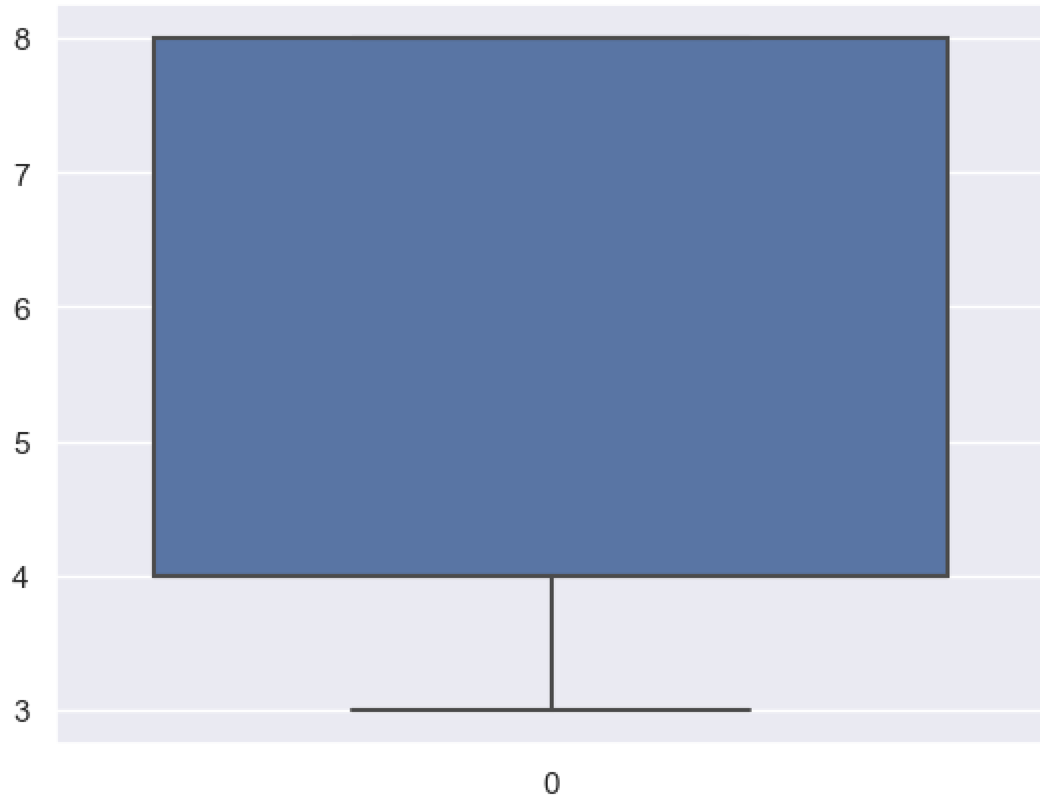
```
acceleration      0
model_year        0
origin            0
name              0
horsepower_new    0
dtype: int64
```

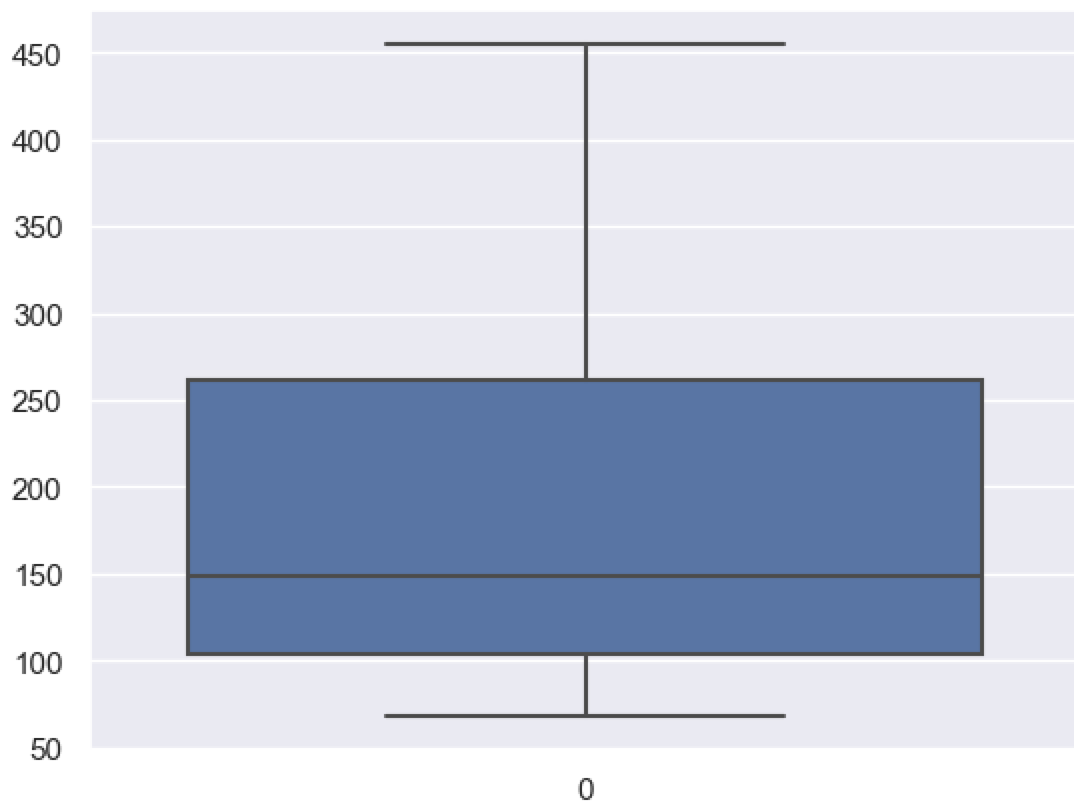
Treating outlier

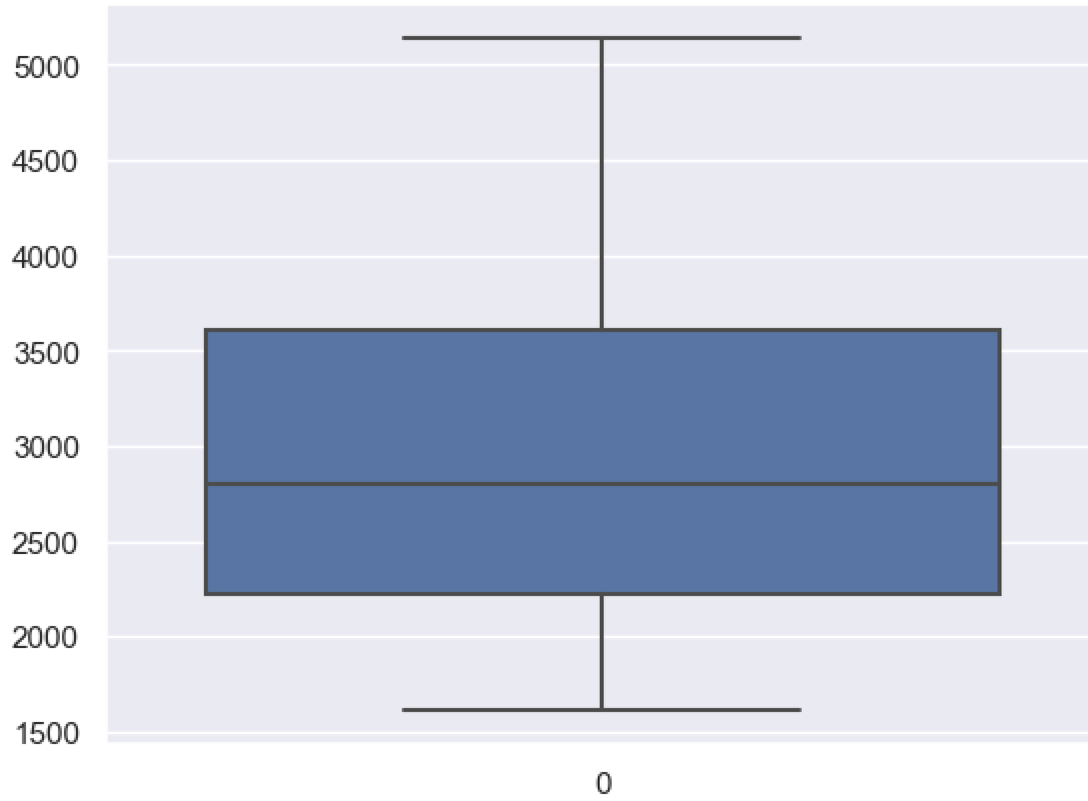
In [10]:

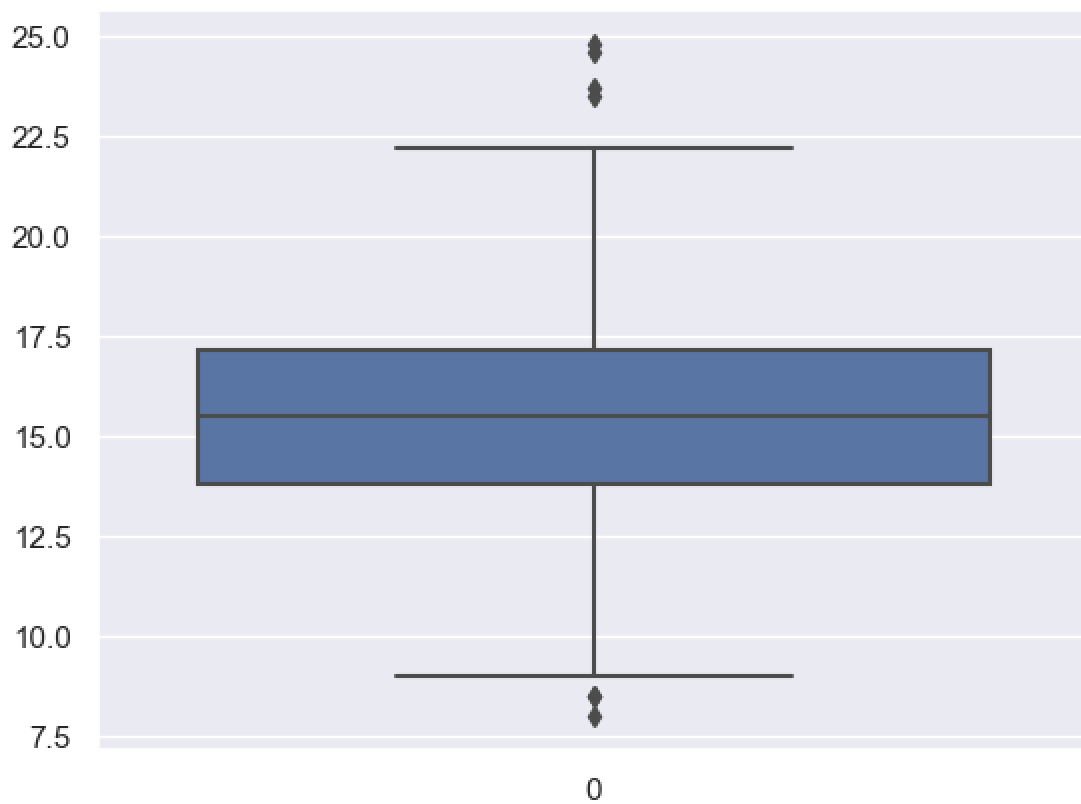
```
def boxplot(col):
    sns.boxplot(data[col])
    plt.show()
for i in list(data.select_dtypes(exclude = ['object']).columns)[0:]:
    boxplot(i)
```

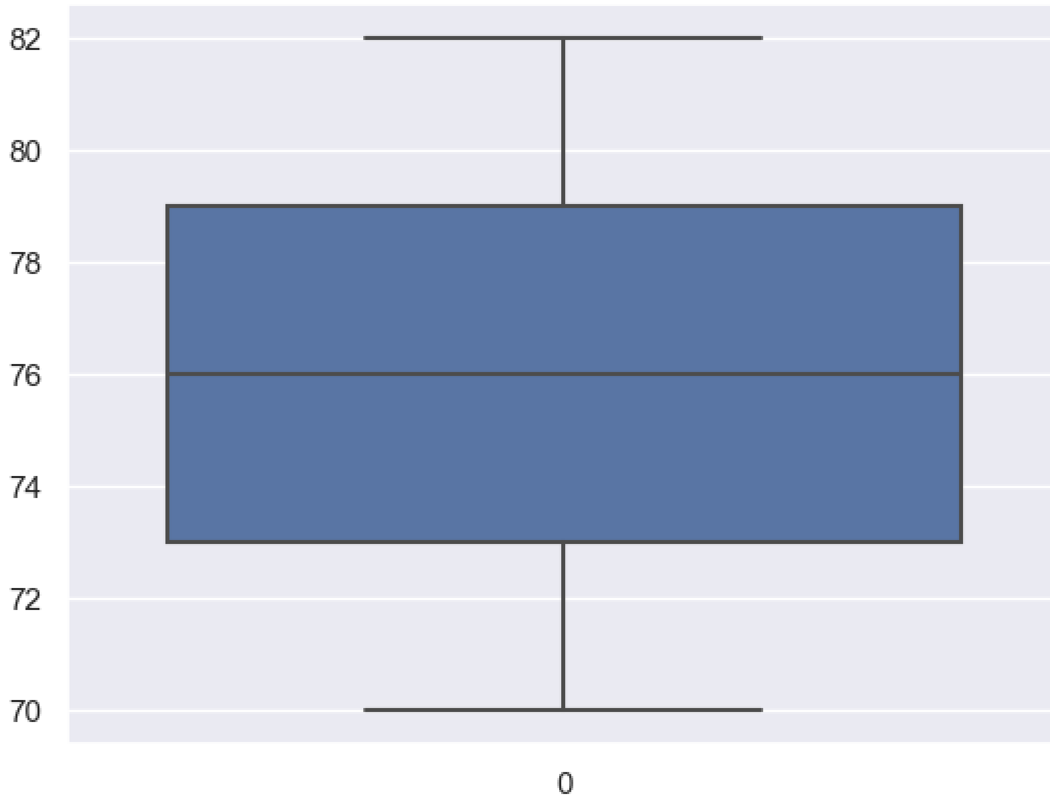


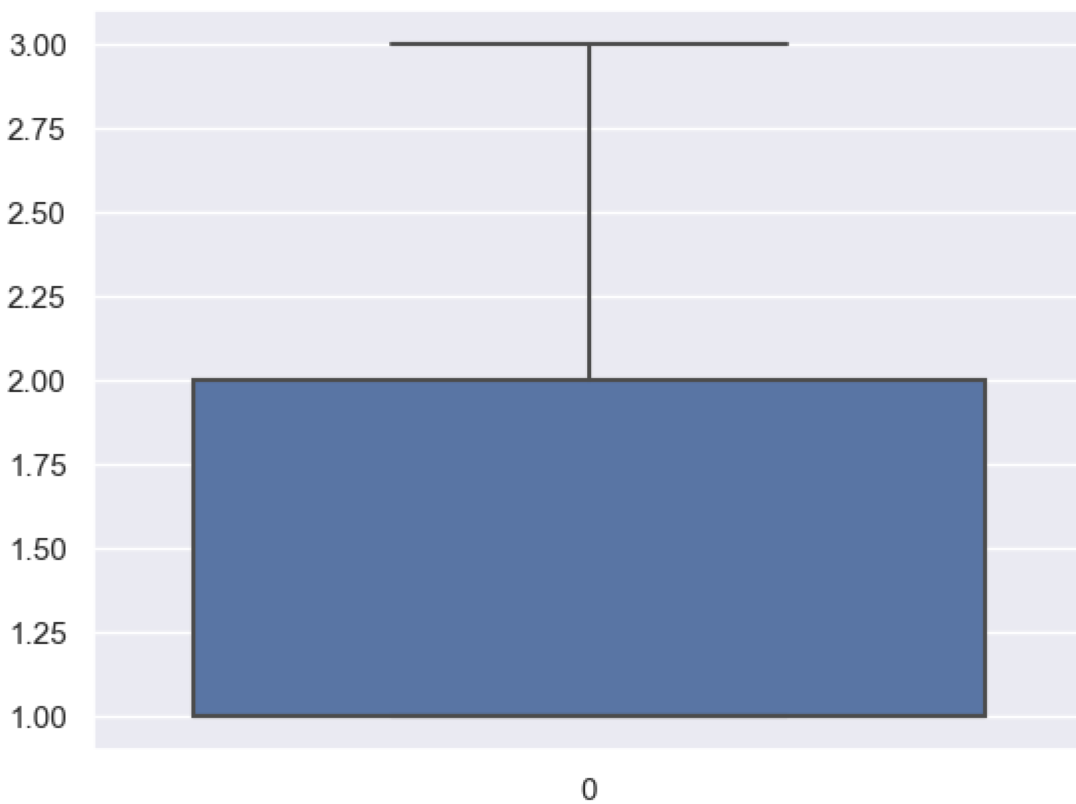

8
7
6
5
4
3
0

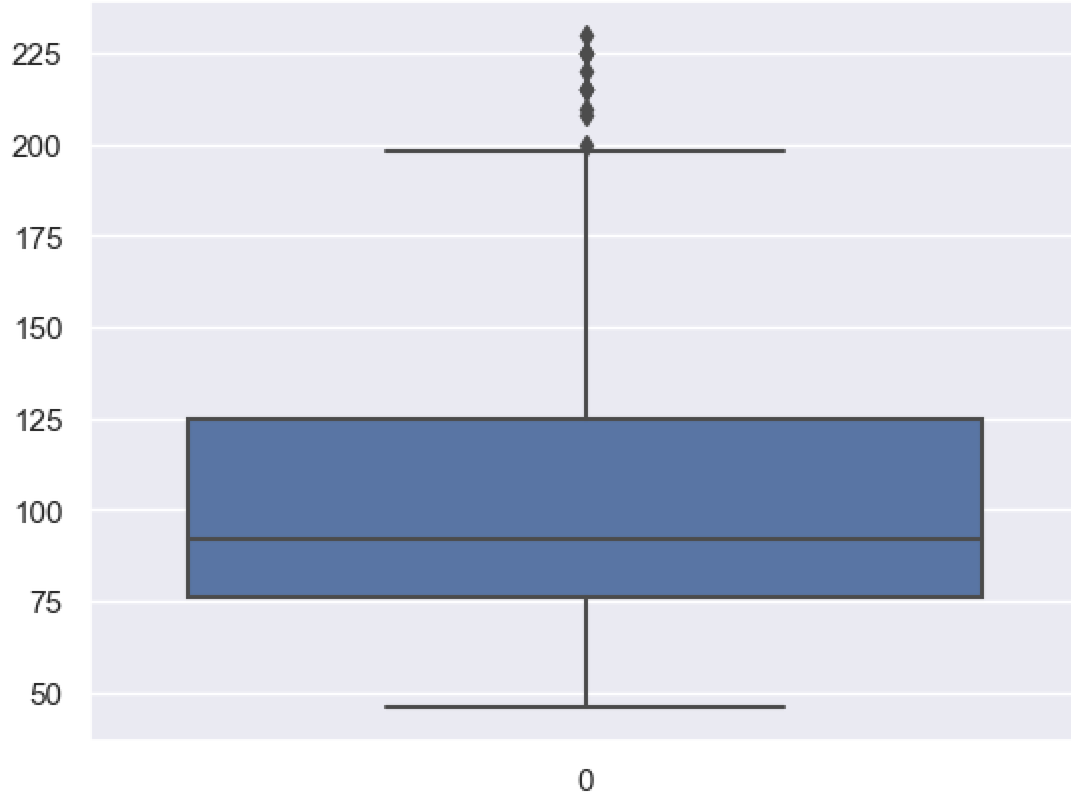

450
400
350
300
250
200
150
100
50
0


5000
4500
4000
3500
3000
2500
2000
1500
0


25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
0


82
80
78
76
74
72
70
0


3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0

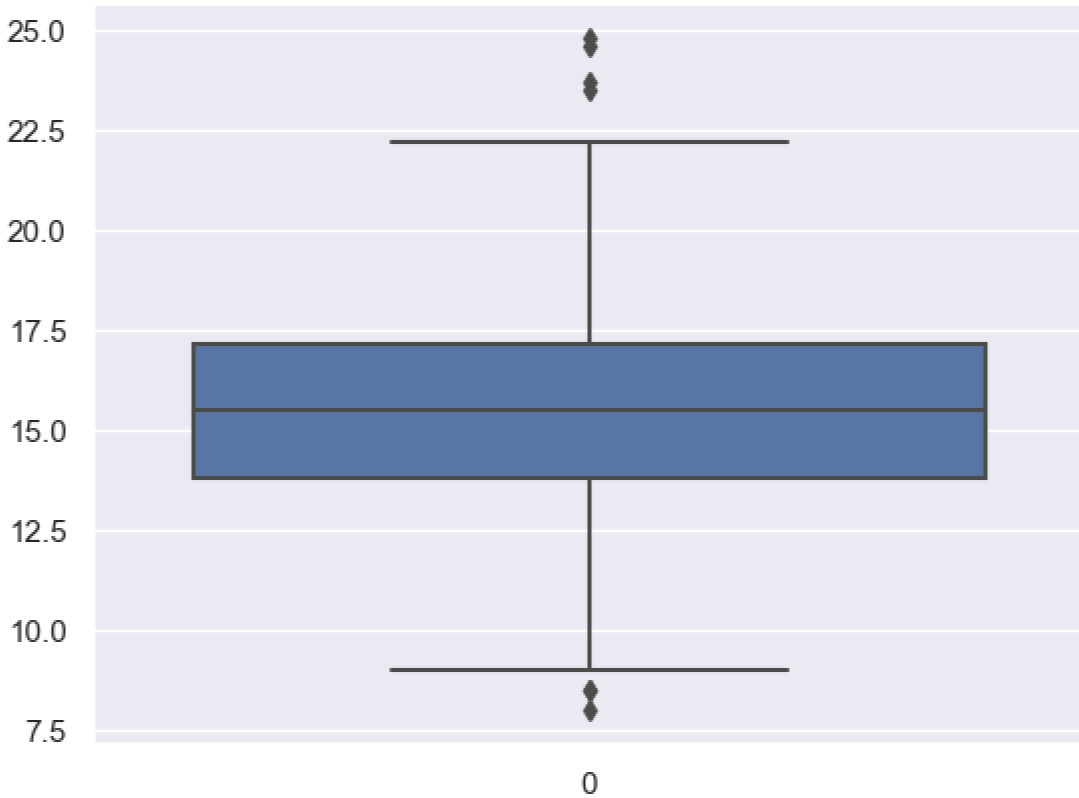


In [11]:
```
data.info()
```

```
<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   weight          398 non-null    int64  
 4   acceleration    398 non-null    float64
 5   model_year      398 non-null    int64  
 6   origin          398 non-null    int64  
 7   name            398 non-null    object 
 8   horsepower_new  398 non-null    int64  
dtypes: float64(3), int64(5), object(1)
memory usage: 28.1+ KB
```

In [12]:
```
sns.boxplot(data['acceleration'])
```

Out[12]:
```
<Axes: >
```



Outlier Treatment

In [13]:
```
Q1 = data['acceleration'].quantile(0.25)
Q3 = data['acceleration'].quantile(0.75)
IQR = Q3 - Q1
upper_limit = Q3 +1.5*IQR
lower_limit = Q1  - 1.5*IQR
print('Q1: ',Q1)
print("Q3: ",Q3)
print('IQR: ',IQR)
print('upper limit: ',upper_limit)
print('lower limit: ',lower_limit)
```

```
Q1:  13.825000000000001
Q3:  17.175
IQR:  3.3499999999999996
upper limit:  22.2
lower limit:  8.8
```

In [14]:
```
data['acceleration'] = np.where(data['acceleration']>upper_limit, upper_limit,
                                np.where(data['acceleration']<lower_limit,lower_limit,
                                        data['acceleration']))
```

In [15]:
```
#label encoding for name
data['origin'].value_counts()
```

Out[15]:
```
1    249
3     79
2     70
Name: origin, dtype: int64
```

In [16]:
```
# categorical_features = ['origin','cylinders','model year' ]
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
origin_enco = LabelEncoder()
data['origin'] = origin_enco.fit_transform(data['origin'])
```

In [18]:
```
cylinder_enco = LabelEncoder()
data['cylinders'] = cylinder_enco.fit_transform(data['cylinders'])
```

In [19]:
```
data['cylinders'].value_counts()
```

Out[19]:
```
1    204
4    103
3     84
0      4
2      3
Name: cylinders, dtype: int64
```

In [20]:
```
model_enco = LabelEncoder()
data['model_year'] = model_enco.fit_transform(data['model_year'])
```

Feature Scaling

In [21]:
```
data.drop("name",axis = 1, inplace = True)
```

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 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   weight          398 non-null    int64  
 4   acceleration    398 non-null    float64
 5   model_year      398 non-null    int64  
 6   origin          398 non-null    int64  
 7   horsepower_new  398 non-null    int64  
dtypes: float64(3), int64(5)
memory usage: 25.0 KB
```

In [23]:
```
x = data.iloc[:,2:]
y = data.iloc[:,0:1]
```

In [24]:
```
x.head()
```

Out[24]:

	displacement	weight	acceleration	model_year	origin	horsepower_new
0	307.0	3504	12.0	0	0	130
1	350.0	3693	11.5	0	0	165
2	318.0	3436	11.0	0	0	150
3	304.0	3433	12.0	0	0	150
4	302.0	3449	10.5	0	0	140

In [25]:
```
y.head()
```

Out[25]:

	mpg
0	18.0
1	15.0

2	18.0
3	16.0
4	17.0

In [26]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
sc_x = scaler.fit_transform(x)
pd.DataFrame(sc_x)
```

Out[26]:

	0	1	2	3	4	5
0	1.090604	0.630870	-1.320595	-1.627426	-0.715145	0.673589
1	1.503514	0.854333	-1.506489	-1.627426	-0.715145	1.590266
2	1.196232	0.550470	-1.692383	-1.627426	-0.715145	1.197404
3	1.061796	0.546923	-1.320595	-1.627426	-0.715145	1.197404
4	1.042591	0.565841	-1.878278	-1.627426	-0.715145	0.935497
...	...	...	...	...	...	...
393	-0.513026	-0.213324	0.017842	1.621983	-0.715145	-0.478804
394	-0.925936	-0.993671	2.471644	1.621983	0.533222	-1.369289
395	-0.561039	-0.798585	-1.469311	1.621983	-0.715145	-0.531185
396	-0.705077	-0.408411	1.133207	1.621983	-0.715145	-0.662139
397	-0.714680	-0.296088	1.430637	1.621983	-0.715145	-0.583567

398 rows × 6 columns

In [27]:
```
pd.DataFrame(sc_x).describe()
```

Out[27]:

	0	1	2	3	4	5
count	3.980000e+02	3.980000e+02	3.980000e+02	3.980000e+02	3.980000e+02	3.980000e+02
mean	-1.785283e-17	-1.606755e-16	-1.071170e-16	2.142340e-16	-5.355850e-17	1.428227e-16
std	1.001259e+00	1.001259e+00	1.001259e+00	1.001259e+00	1.001259e+00	1.001259e+00
min	-1.204411e+00	-1.604943e+00	-2.510317e+00	-1.627426e+00	-7.151448e-01	-1.526434e+00
25%	-8.563178e-01	-8.828266e-01	-6.420819e-01	-8.150739e-01	-7.151448e-01	-7.407114e-01
50%	-4.314040e-01	-1.973624e-01	-1.933672e-02	-2.721449e-03	-7.151448e-01	-3.216593e-01
75%	6.584879e-01	7.538337e-01	6.034085e-01	8.096310e-01	5.332220e-01	5.426356e-01
max	2.511784e+00	2.565185e+00	2.471644e+00	1.621983e+00	1.781589e+00	3.292665e+00

In [28]:
```
var = sc_x
var.shape
```

Out[28]:
```
(398, 6)
```

Checking multicollinearity

In [29]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
var = sc_x
vif =pd.DataFrame()
vif['variance_inflation_factor'] = [variance_inflation_factor(var,i) for i in range(var.
vif['features'] = x.columns
```

In [30]:
```
vif
```

Out[30]:

	variance_inflation_factor	features
0	12.178992	displacement
1	10.498106	weight
2	2.596991	acceleration
3	1.244622	model_year
4	1.729181	origin
5	9.432523	horsepower_new

Vif of displacement column is higher so we will remove it

In [31]:
```
data.drop('displacement',axis = 1, inplace = True)
data.head()
```

Out[31]:

	mpg	cylinders	weight	acceleration	model_year	origin	horsepower_new
0	18.0	4	3504	12.0	0	0	130
1	15.0	4	3693	11.5	0	0	165
2	18.0	4	3436	11.0	0	0	150
3	16.0	4	3433	12.0	0	0	150
4	17.0	4	3449	10.5	0	0	140

In [32]:
```
x1 = data.iloc[:,2:]
y1 = data.iloc[:,0:1]
```

In [33]:
```
y1.head()
```

Out[33]:

	mpg
0	18.0
1	15.0
2	18.0
3	16.0
4	17.0

In [34]:
```
scaler1 = StandardScaler()
sc_x1 = scaler1.fit_transform(x1)
pd.DataFrame(sc_x1)
```

Out[34]:

	0	1	2	3	4

0	0.630870	-1.320595	-1.627426	-0.715145	0.673589
1	0.854333	-1.506489	-1.627426	-0.715145	1.590266
2	0.550470	-1.692383	-1.627426	-0.715145	1.197404
3	0.546923	-1.320595	-1.627426	-0.715145	1.197404
4	0.565841	-1.878278	-1.627426	-0.715145	0.935497
...	...	...	...	...	...
393	-0.213324	0.017842	1.621983	-0.715145	-0.478804
394	-0.993671	2.471644	1.621983	0.533222	-1.369289
395	-0.798585	-1.469311	1.621983	-0.715145	-0.531185
396	-0.408411	1.133207	1.621983	-0.715145	-0.662139
397	-0.296088	1.430637	1.621983	-0.715145	-0.583567

398 rows × 5 columns

In [35]:
```
var1 = sc_x1
var1.shape
```

Out[35]:
```
(398, 5)
```

In [36]:
```
vif1 = pd.DataFrame()
vif1['variance_inflation_factor1'] = [variance_inflation_factor(var1,i) for i in range(v
vif1['features'] = x1.columns
```

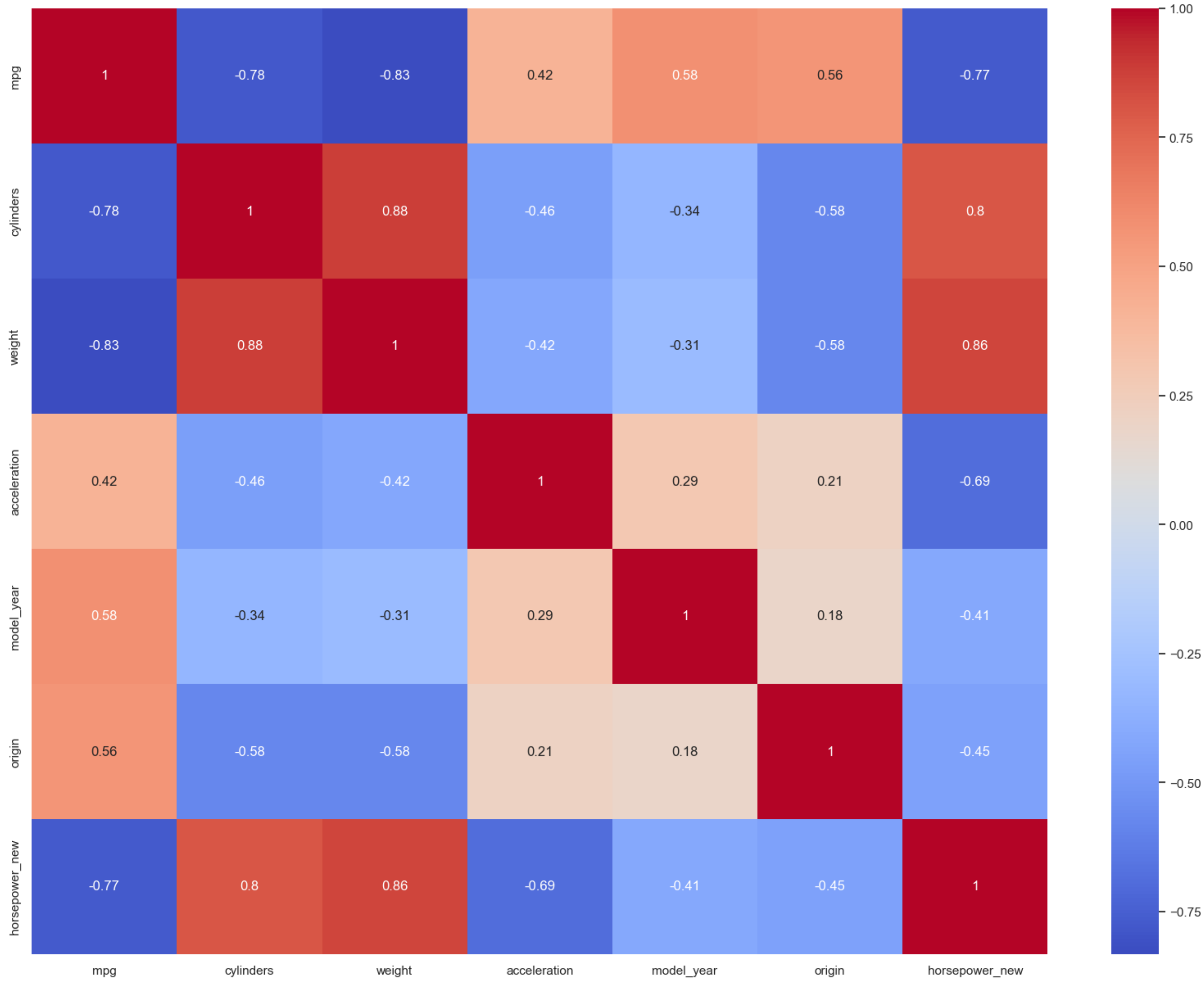
In [37]:
```
vif1
```

Out[37]:

	variance_inflation_factor1	features
0	6.120806	weight
1	2.499726	acceleration
2	1.228353	model_year
3	1.538155	origin
4	8.699791	horsepower_new

Finding correlation

In [38]:
```
plt.figure(figsize=(20,15))
sns.heatmap(data.corr(), annot = True,cmap='coolwarm')
plt.show()
```



In [39]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =0.2,random_state=101)
```

Rergrssion model 1

In [40]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train,y_train)
```

Out[40]:
```
▾ LinearRegression
LinearRegression()
```

In [41]:
```
print(lm.coef_)
print('***********'*5)
print(lm.intercept_)
```

```
[[ 0.01520499 -0.00688301  0.0592133   0.78616418  1.16995869 -0.01721717]]
*******************************************************
[36.60074904]
```

In [42]:
```
x_test.head()
```

Out[42]:

	displacement	weight	acceleration	model_year	origin	horsepower_new

130	122.0	2451	16.5	4	0	80
202	258.0	3193	17.8	6	0	95
322	86.0	2110	17.9	10	2	65
104	400.0	4906	12.5	3	0	167
91	400.0	4464	12.0	3	0	150

Predict test dataset wih linear model

In [43]:

```
y_pred = lm.predict(x_test)
y_pred
```

Out[43]:

```
array([[24.32978998],
       [22.68151893],
       [33.52757722],
       [ 9.13806293],
       [12.44344079],
       [26.08186269],
       [34.43798179],
       [25.11370197],
       [27.03714153],
       [24.0735922 ],
       [25.75123137],
       [26.39348539],
       [34.75037624],
       [28.60873554],
       [17.20164285],
       [18.55360875],
       [20.71415173],
       [19.86982306],
       [25.66218644],
       [25.39752065],
       [ 8.58575882],
       [24.33951096],
       [29.39886739],
       [20.72518866],
       [15.50028393],
       [32.80409456],
       [25.35716315],
       [29.64592711],
       [17.42654862],
       [ 9.77994572],
       [20.60806978],
       [34.06511164],
       [24.67313184],
       [26.07496705],
       [25.8439045 ],
       [11.60777446],
       [28.3439356 ],
       [30.20284646],
       [15.95486995],
       [24.47684353],
       [32.71758896],
       [16.40550344],
       [26.66595053],
       [14.16428926],
       [21.65874182],
       [19.54208705],
       [29.05947031],
       [22.66456658],
```

```
       [21.20134455],
       [33.00662901],
       [21.428808  ],
       [22.16992788],
       [23.22771654],
       [25.69373339],
       [15.66447169],
       [27.83069754],
       [34.78846079],
       [25.32294323],
       [16.80688139],
       [32.01631988],
       [30.40880856],
       [25.43215002],
       [35.08598643],
       [19.09575745],
       [21.40004846],
       [23.79206428],
       [31.05429952],
       [32.55619221],
       [12.68297108],
       [13.22560415],
       [18.92585729],
       [25.0982099 ],
       [19.39392958],
       [27.9750004 ],
       [18.9963028 ],
       [12.11979723],
       [25.45758   ],
       [13.62522468],
       [21.31933959],
       [20.75414116]])
```

Evaluation for model 1

In [44]:
```
from sklearn.metrics import r2_score
print("Accuracy: ", r2_score(y_test,y_pred))
```

```
Accuracy:  0.8050952184103181
```

linear model 2

In [45]:
```
(x1_train,x1_test,y1_train,y1_test) = train_test_split(x1,y1,test_size = 0.2,random_stat
```

In [46]:
```
lm1 = LinearRegression()
lm1.fit(x1_train,y1_train)
```

Out[46]:
```
▾ LinearRegression
LinearRegression()
```

In [47]:
```
print(lm1.coef_)
```

```
[[-0.00565688 -0.01653479  0.77483059  0.93703182 -0.01014362]]
```

In [48]:
```
print(lm1.intercept_)
```

```
[36.52142764]
```

In [49]:
```
y1_pred = lm1.predict(x1_test)
```

```
y1_pred
```

Out[49]:
```
array([[24.67141909],
       [21.85002448],
       [33.25246854],
       [ 9.19258794],
       [11.87363863],
       [26.84693733],
       [33.81272775],
       [25.41645485],
       [26.2725117 ],
       [24.34405765],
       [25.41768553],
       [26.84256453],
       [34.26322503],
       [28.16717412],
       [17.01226883],
       [18.17578991],
       [19.90359117],
       [19.45393895],
       [25.7879024 ],
       [25.80089887],
       [ 8.52586436],
       [24.30678357],
       [29.42283771],
       [20.20832814],
       [15.73158117],
       [32.06505479],
       [25.70529563],
       [29.84726814],
       [17.31077758],
       [ 9.0420631 ],
       [20.81716152],
       [33.79204695],
       [25.19650056],
       [26.51599687],
       [27.093569  ],
       [11.61432794],
       [28.5560865 ],
       [30.48753813],
       [15.75164895],
       [24.2109772 ],
       [32.8024156 ],
       [16.36215631],
       [26.55833956],
       [14.32627697],
       [21.58586971],
       [18.99067271],
       [28.9632257 ],
       [23.43148299],
       [20.89292508],
       [32.51698249],
       [20.59101696],
       [21.96319569],
       [22.63250839],
       [26.18607954],
       [15.41712019],
       [28.40804476],
       [34.40706377],
       [25.18803771],
       [16.9366266 ],
       [31.66258724],
       [30.35631524],
       [25.4684354 ],
       [34.73539456],
       [19.16741528],
```

```
       [21.34044718],
       [23.36415765],
       [30.79613519],
       [32.59075319],
       [13.18962885],
       [12.94689774],
       [18.40691334],
       [25.21394   ],
       [19.27554839],
       [28.3054246 ],
       [16.61662767],
       [12.6632652 ],
       [25.78774242],
       [13.5767135 ],
       [20.76031012],
       [20.64604912]])
```

In [50]:
```
from sklearn.metrics import r2_score
```

In [51]:
```
print("Accuracy: ",r2_score(y1_test,y1_pred))
```

```
Accuracy:  0.8078071902824695
```

OLS method

In [52]:
```
# model1

from statsmodels.regression.linear_model import OLS
import statsmodels.regression.linear_model as smf
```

In [53]:
```
reg1 = smf.OLS(endog = y_train,exog=x_train).fit()
```

In [54]:
```
reg_test1 = smf.OLS(endog = y_test,exog = x_test).fit()
```

In [55]:
```
reg1.summary()
```

Out[55]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared (uncentered):**	0.967
Model:	OLS	**Adj. R-squared (uncentered):**	0.967
Method:	Least Squares	**F-statistic:**	1543.
Date:	Fri, 19 Jan 2024	**Prob (F-statistic):**	1.37e-228
Time:	22:25:17	**Log-Likelihood:**	-929.90
No. Observations:	318	**AIC:**	1872.
Df Residuals:	312	**BIC:**	1894.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
displacement	0.0137	0.009	1.573	0.117	-0.003	0.031
weight	-0.0088	0.001	-8.521	0.000	-0.011	-0.007
acceleration	1.6214	0.084	19.239	0.000	1.456	1.787
model_year	1.0404	0.074	14.139	0.000	0.896	1.185

	coef	std err	t	P>|t|	[0.025	0.975]
origin	2.1056	0.404	5.210	0.000	1.310	2.901
horsepower_new	0.1359	0.016	8.257	0.000	0.103	0.168

Omnibus:	1.485	**Durbin-Watson:**	2.004
Prob(Omnibus):	0.476	**Jarque-Bera (JB):**	1.533
Skew:	0.163	**Prob(JB):**	0.465
Kurtosis:	2.900	**Cond. No.**	4.91e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 4.91e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [56]:
```
reg_test1.summary()
```

Out[56]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared (uncentered):**	0.977
Model:	OLS	**Adj. R-squared (uncentered):**	0.975
Method:	Least Squares	**F-statistic:**	514.5
Date:	Fri, 19 Jan 2024	**Prob (F-statistic):**	3.30e-58
Time:	22:25:17	**Log-Likelihood:**	-217.81
No. Observations:	80	**AIC:**	447.6
Df Residuals:	74	**BIC:**	461.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
displacement	-0.0257	0.015	-1.760	0.083	-0.055	0.003
weight	-0.0041	0.001	-2.962	0.004	-0.007	-0.001
acceleration	1.4029	0.117	11.946	0.000	1.169	1.637
model_year	0.6845	0.135	5.073	0.000	0.416	0.953
origin	3.0225	0.748	4.043	0.000	1.533	4.512
horsepower_new	0.1234	0.028	4.431	0.000	0.068	0.179

Omnibus:	3.085	**Durbin-Watson:**	2.136
Prob(Omnibus):	0.214	**Jarque-Bera (JB):**	2.425
Skew:	0.406	**Prob(JB):**	0.297
Kurtosis:	3.258	**Cond. No.**	5.44e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 5.44e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [57]:
```
reg2 = smf.OLS(endog=y1_train,exog = x1_train).fit()
```

In [58]:
```
reg2_test = smf.OLS(endog = y1_test,exog=x1_test).fit()
```

In [59]:
```
reg2.summary()
```

Out[59]:

OLS Regression Results

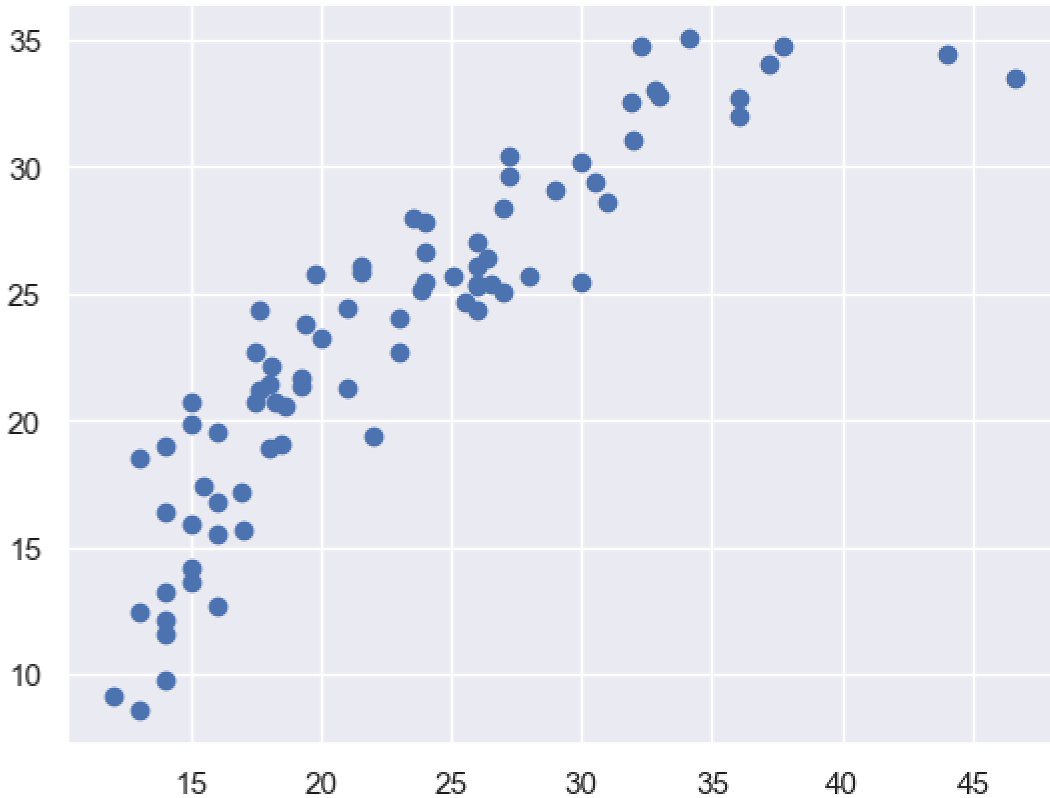
Dep. Variable:	mpg	**R-squared (uncentered):**	0.967
Model:	OLS	**Adj. R-squared (uncentered):**	0.967
Method:	Least Squares	**F-statistic:**	1842.
Date:	Fri, 19 Jan 2024	**Prob (F-statistic):**	1.04e-229
Time:	22:25:17	**Log-Likelihood:**	-931.15
No. Observations:	318	**AIC:**	1872.
Df Residuals:	313	**BIC:**	1891.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
weight	-0.0077	0.001	-10.132	0.000	-0.009	-0.006
acceleration	1.5502	0.071	21.747	0.000	1.410	1.691
model_year	1.0298	0.073	14.021	0.000	0.885	1.174
origin	1.8944	0.382	4.958	0.000	1.143	2.646
horsepower_new	0.1419	0.016	8.851	0.000	0.110	0.173

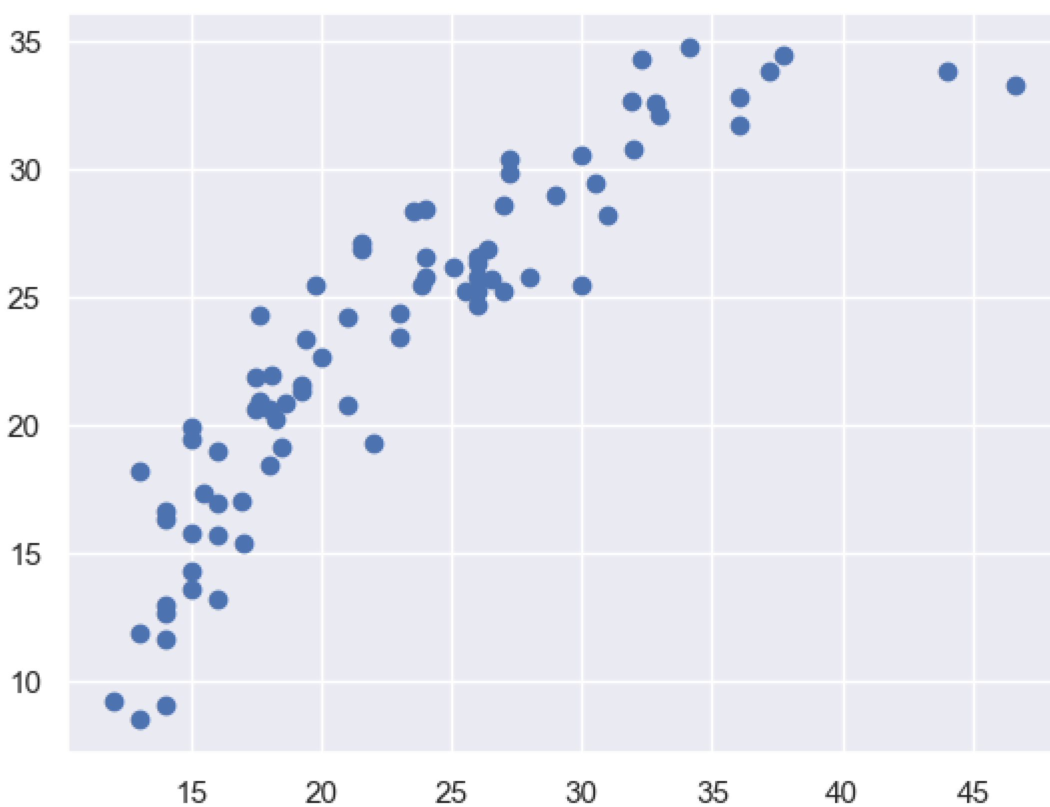
Omnibus:	0.757	**Durbin-Watson:**	1.984
Prob(Omnibus):	0.685	**Jarque-Bera (JB):**	0.750
Skew:	0.117	**Prob(JB):**	0.687
Kurtosis:	2.959	**Cond. No.**	4.64e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 4.64e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

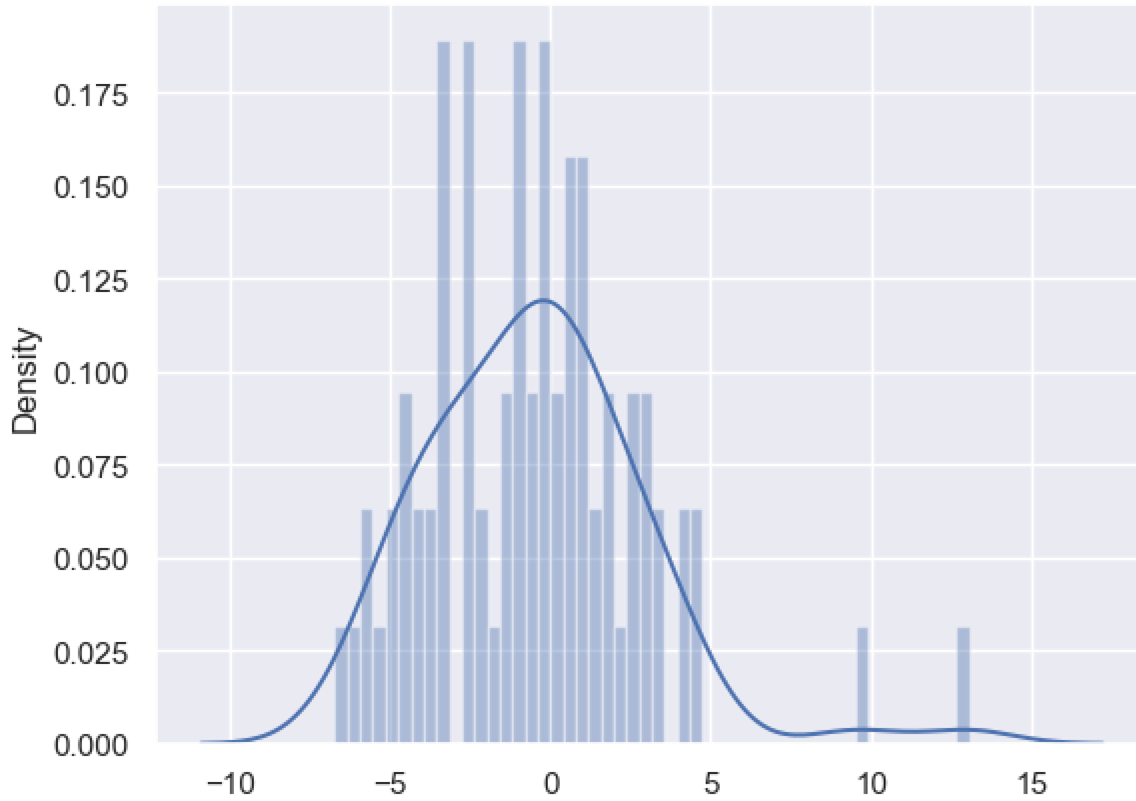
In [60]:
```
plt.scatter(y_test,y_pred)
plt.show()
```



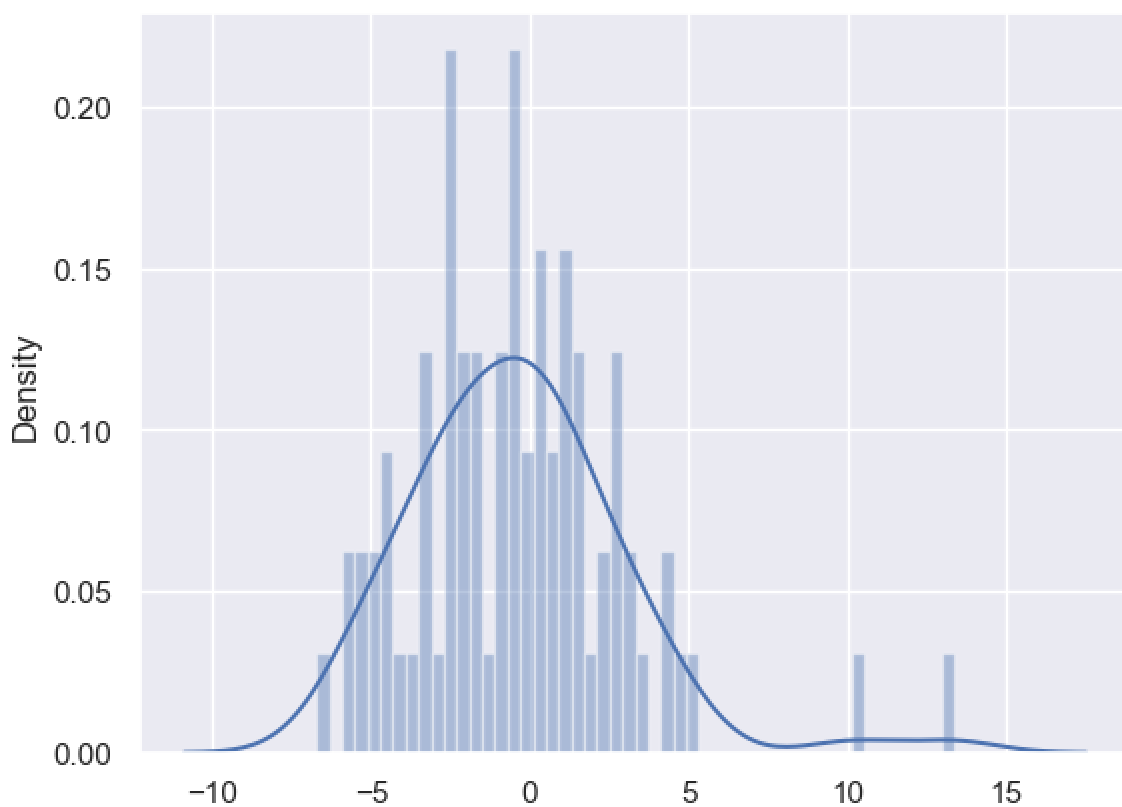
In [61]:
```
plt.scatter(y1_test,y1_pred)
plt.show()
```



In [62]:
```
sns.distplot((y_test-y_pred),bins = 50)
plt.show()
```



In [63]:
```
sns.distplot((y1_test-y1_pred),bins = 50)
plt.show()
```



In [64]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [65]:
```
data1 = pd.read_csv('mpg.csv')
data1.head()
```

Out[65]:

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	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

In [66]:

```
model = ols('mpg~displacement', data=data1).fit()
annova_result = sm.stats.anova_lm(model,typ=2)
print(annova_result)
```

```
                    sum_sq     df           F         PR(>F)
displacement  15685.163618    1.0  724.994303  1.655889e-91
Residual       8567.411859  396.0         NaN           NaN
```

In [67]:

```
data.head()
```

Out[67]:

	mpg	cylinders	weight	acceleration	model_year	origin	horsepower_new
0	18.0	4	3504	12.0	0	0	130
1	15.0	4	3693	11.5	0	0	165
2	18.0	4	3436	11.0	0	0	150
3	16.0	4	3433	12.0	0	0	150
4	17.0	4	3449	10.5	0	0	140