

Train the model on IBM

Team ID	PNT2022TMID02503
Project Name	Machine Learning based Vehicle Performance Analyzer

Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

Importing Dataset

In [2]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your c
redentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Uede0uog_DlYeHmTn0uQW3GYQhYgYHbY1kvWgQybaYm1',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'vehicleperformanceprediction-donotdelete-pr-kj6qz2159y6996'
object_key = 'car performance.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(__iter__, body )

dataset = pd.read_csv(body)
dataset.head()
```

Out[2]:

	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

Finding missing data

In [3]:

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

Out[3]:

```
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 replaced '?' with nan and replaced nan values with mean of the column.

In [4]:

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

In [5]:

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

Out[5]:

```
6
```

In [6]:

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

In [7]:

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

In [8]:

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

Out[8]:

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

In [9]:

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

```
<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
```

In [10]:

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

Out[10]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	38.199187	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	76.000000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	95.000000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

There is no use with car name attribute so drop it

In [11]:

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

In [12]:

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

Out[12]:

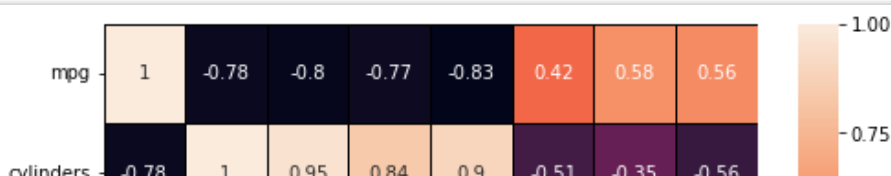
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.771437	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.838939	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.893646	0.932824	-0.543684	-0.370164	-0.609409
horsepower	-0.771437	0.838939	0.893646	1.000000	0.860574	-0.684259	-0.411651	-0.453669
weight	-0.831741	0.896017	0.932824	0.860574	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.684259	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.411651	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.453669	-0.581024	0.205873	0.180662	1.000000

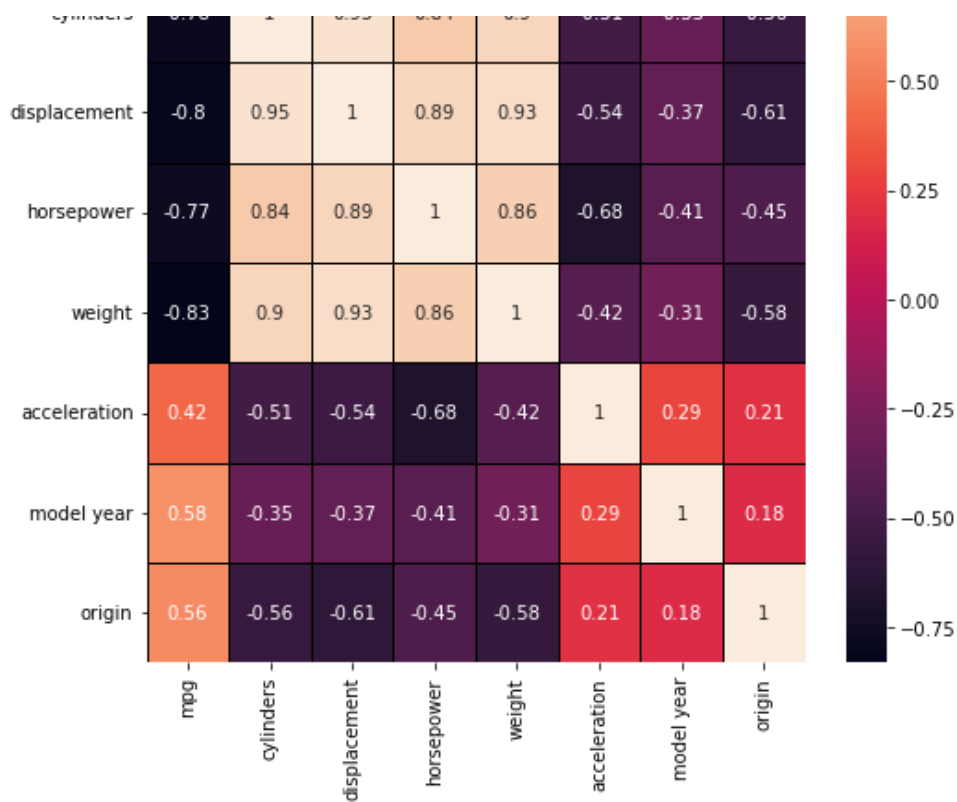
Data Visualizations

Heatmap : which represents correlation between attributes

In [13]:

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

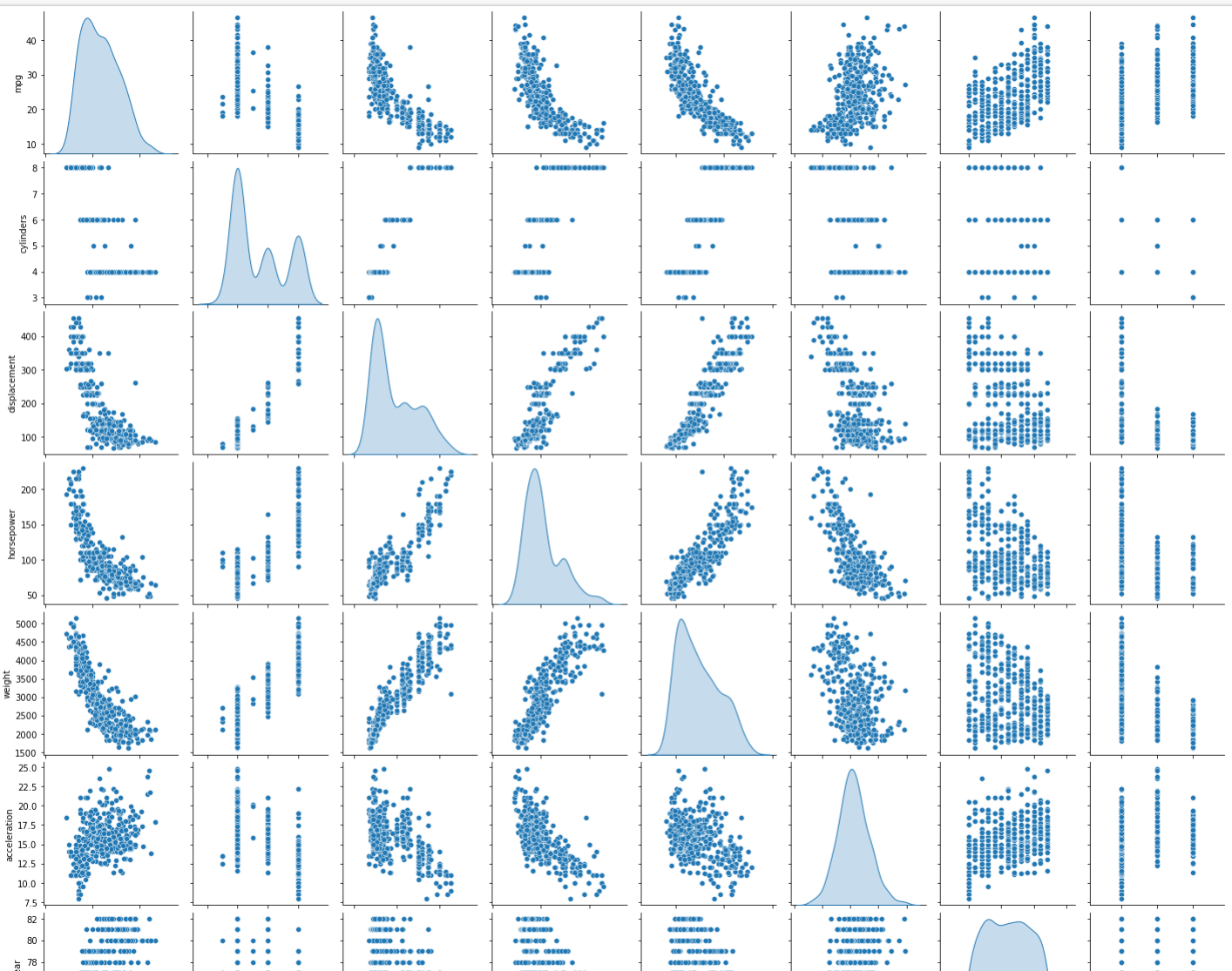


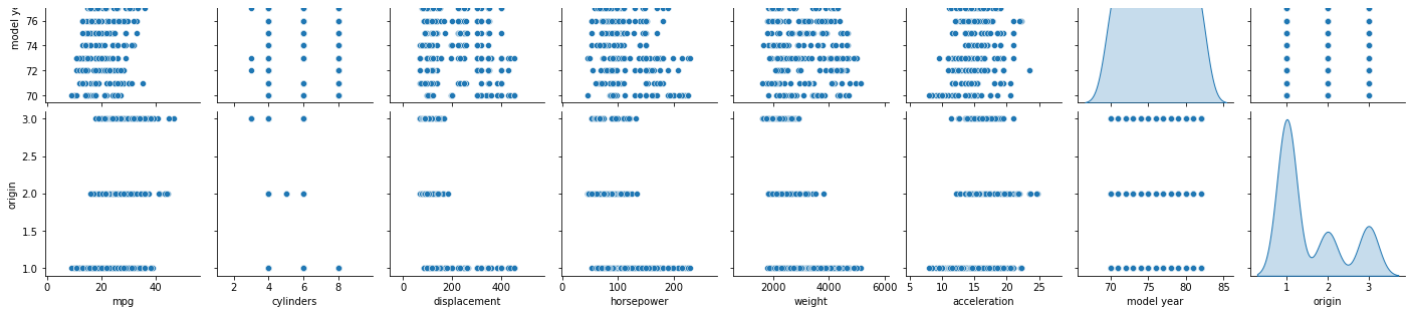


Visualizations of each attributes w.r.t rest of all attributes

In [14]:

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





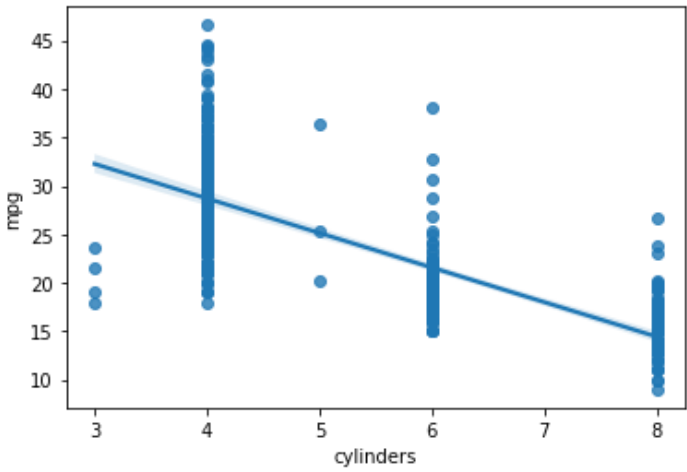
Regression plots(`regplot()`) creates a regression line between 2 parameters and helps to visualize their linear relationships.

In [15]:

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

Out[15]:

<AxesSubplot:xlabel='cylinders', ylabel='mpg'>

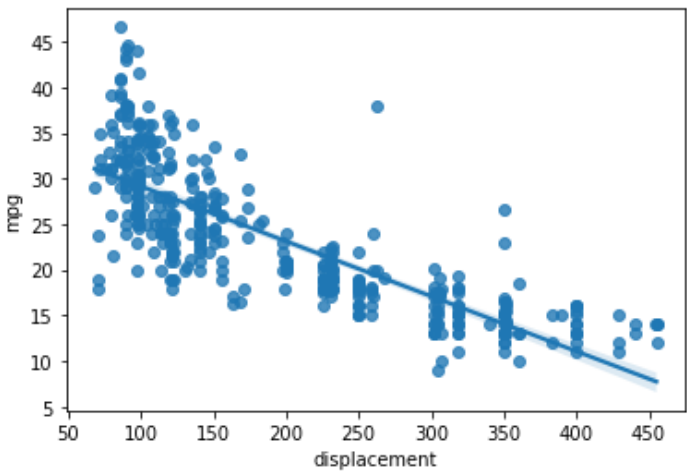


In [16]:

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

Out[16]:

<AxesSubplot:xlabel='displacement', ylabel='mpg'>



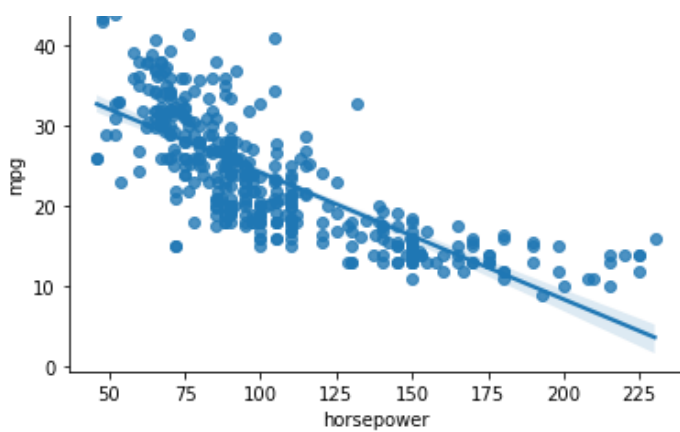
In [17]:

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

Out[17]:

<AxesSubplot:xlabel='horsepower', ylabel='mpg'>



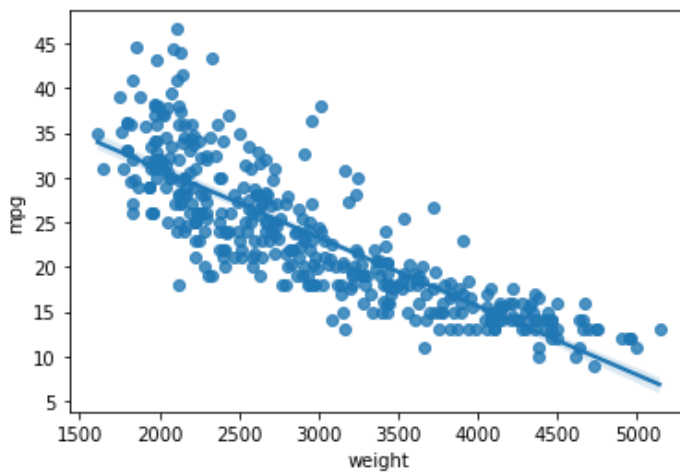


In [18]:

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

Out[18]:

```
<AxesSubplot:xlabel='weight', ylabel='mpg'>
```

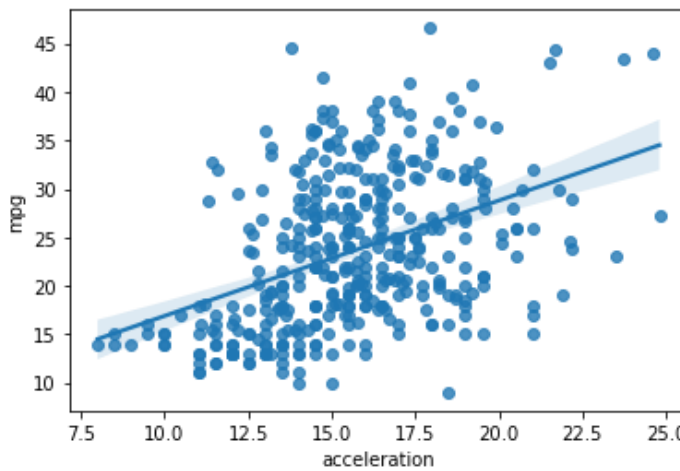


In [19]:

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

Out[19]:

```
<AxesSubplot:xlabel='acceleration', ylabel='mpg'>
```



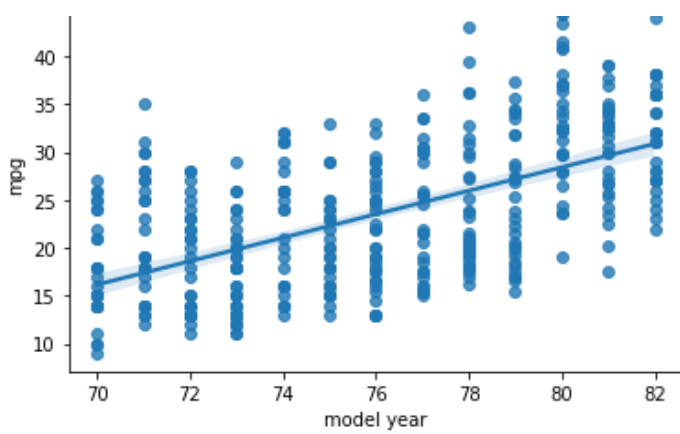
In [20]:

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

Out[20]:

```
<AxesSubplot:xlabel='model year', ylabel='mpg'>
```



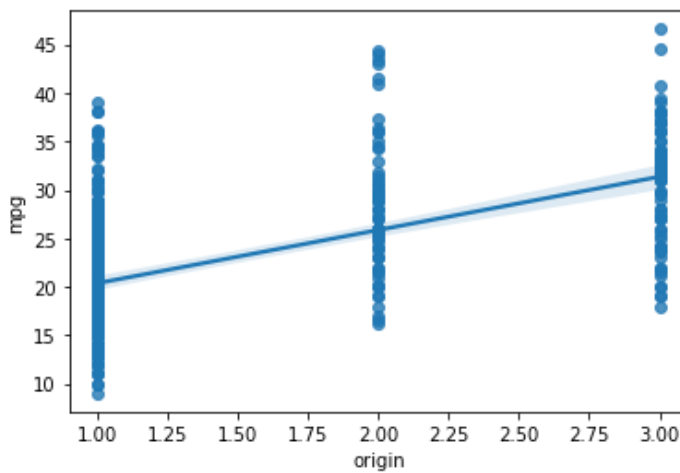


In [21]:

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

Out[21]:

<AxesSubplot:xlabel='origin', ylabel='mpg'>

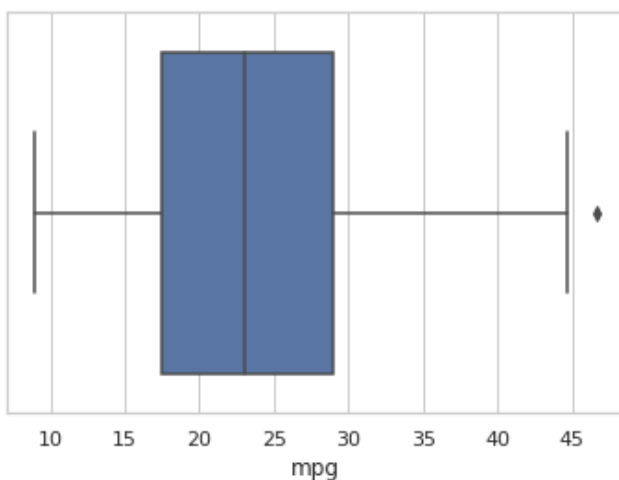


In [22]:

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

Out[22]:

<AxesSubplot:xlabel='mpg'>



Finding quartiles for mpg

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.

In [23]:

```
from scipy import stats
```

Cylinders vs mpg

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

In [24]:

```
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 -0.7753962854205542 with a P-value of P = 4.503992246177055e-81

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'.

In [25]:

```
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 -0.8042028248058978 with a P-value of P = 1.6558889101930157e-91

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'.

In [26]:

```
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 -0.7714371350025526 with a P-value of P = 9.255477533166725e-80

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'.

In [27]:

```
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 -0.831740933244335 with a P-value of P = 2.972799
5640500577e-103
```

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'.

In [28]:

```
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 0.4202889121016507 with a P-value of P = 1.823091
535078553e-18
```

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'.

In [29]:

```
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 0.5792671330833096 with a P-value of P = 4.844935
813365483e-37
```

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'.

In [30]:

```
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 0.5634503597738431 with a P-value of P = 1.0114822102336483e-34

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

In [31]:

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

Out[31]:

OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.717
Model:	OLS	Adj. R-squared:	0.713
Method:	Least Squares	F-statistic:	165.5
Date:	Sun, 13 Nov 2022	Prob (F-statistic):	4.84e-104
Time:	15:17:17	Log-Likelihood:	-1131.1
No. Observations:	398	AIC:	2276.
Df Residuals:	391	BIC:	2304.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	42.7111	2.693	15.861	0.000	37.417	48.005
cylinders	-0.5256	0.404	-1.302	0.194	-1.320	0.268
displacement	0.0106	0.009	1.133	0.258	-0.008	0.029
horsepower	-0.0529	0.016	-3.277	0.001	-0.085	-0.021
weight	-0.0051	0.001	-6.441	0.000	-0.007	-0.004
acceleration	0.0043	0.120	0.036	0.972	-0.232	0.241
origin	1.4269	0.345	4.136	0.000	0.749	2.105

Omnibus:	32.659	Durbin-Watson:	0.886
Prob(Omnibus):	0.000	Jarque-Bera (JB):	43.338
Skew:	0.624	Prob(JB):	3.88e-10
Kurtosis:	4.028	Cond. No.	3.99e+04

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 3.99e+04. This might indicate that there are

strong multicollinearity or other numerical problems.

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

Seperating into Dependent and Independent variables

Independent variables

In [32]:

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

Out[32]:

```
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]])
```

Dependent variables

In [33]:

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

Out[33]:

```
array([[18. ],
       [15. ],
       [18. ],
       [16. ],
       [17. ],
       [15. ],
       [14. ],
       [14. ],
       [14. ],
       [15. ],
       [15. ],
       [14. ],
       [15. ],
       [14. ],
       [24. ],
       [22. ],
       [18. ],
       [21. ],
       [27. ],
       [26. ],
       [25. ],
       [24. ],
       [25. ],
       [26. ],
       [21. ],
       [10. ],
       [10. ],
       [11. ],
       [ 9. ],
       [27. ],
       [28. ],
       [25. ],
       [25. ],
       [19. ]], dtype=float64)
```

[16.],
[17.],
[19.],
[18.],
[14.],
[14.],
[14.],
[14.],
[12.],
[13.],
[13.],
[18.],
[22.],
[19.],
[18.],
[23.],
[28.],
[30.],
[30.],
[31.],
[35.],
[27.],
[26.],
[24.],
[25.],
[23.],
[20.],
[21.],
[13.],
[14.],
[15.],
[14.],
[17.],
[11.],
[13.],
[12.],
[13.],
[19.],
[15.],
[13.],
[13.],
[14.],
[18.],
[22.],
[21.],
[26.],
[22.],
[28.],
[23.],
[28.],
[27.],
[13.],
[14.],
[13.],
[14.],
[15.],
[12.],
[13.],
[13.],
[14.],
[13.],
[12.],
[13.],
[18.],
[16.],
[18.],
[18.],
[23.],
[26.],
[11.],
[12.],
[13.],

[12.],
[18.],
[20.],
[21.],
[22.],
[18.],
[19.],
[21.],
[26.],
[15.],
[16.],
[29.],
[24.],
[20.],
[19.],
[15.],
[24.],
[20.],
[11.],
[20.],
[21.],
[19.],
[15.],
[31.],
[26.],
[32.],
[25.],
[16.],
[16.],
[18.],
[16.],
[13.],
[14.],
[14.],
[14.],
[29.],
[26.],
[26.],
[31.],
[32.],
[28.],
[24.],
[26.],
[24.],
[26.],
[31.],
[19.],
[18.],
[15.],
[15.],
[16.],
[15.],
[16.],
[14.],
[17.],
[16.],
[15.],
[18.],
[21.],
[20.],
[13.],
[29.],
[23.],
[20.],
[23.],
[24.],
[25.],
[24.],
[18.],
[29.],
[19.],
[23.],

[23.],
[22.],
[25.],
[33.],
[28.],
[25.],
[25.],
[26.],
[27.],
[17.5],
[16.],
[15.5],
[14.5],
[22.],
[22.],
[24.],
[22.5],
[29.],
[24.5],
[29.],
[33.],
[20.],
[18.],
[18.5],
[17.5],
[29.5],
[32.],
[28.],
[26.5],
[20.],
[13.],
[19.],
[19.],
[16.5],
[16.5],
[13.],
[13.],
[13.],
[31.5],
[30.],
[36.],
[25.5],
[33.5],
[17.5],
[17.],
[15.5],
[15.],
[17.5],
[20.5],
[19.],
[18.5],
[16.],
[15.5],
[15.5],
[16.],
[29.],
[24.5],
[26.],
[25.5],
[30.5],
[33.5],
[30.],
[30.5],
[22.],
[21.5],
[21.5],
[43.1],
[36.1],
[32.8],
[39.4],
[36.1],
[19.9],

[19.4],
[20.2],
[19.2],
[20.5],
[20.2],
[25.1],
[20.5],
[19.4],
[20.6],
[20.8],
[18.6],
[18.1],
[19.2],
[17.7],
[18.1],
[17.5],
[30.],
[27.5],
[27.2],
[30.9],
[21.1],
[23.2],
[23.8],
[23.9],
[20.3],
[17.],
[21.6],
[16.2],
[31.5],
[29.5],
[21.5],
[19.8],
[22.3],
[20.2],
[20.6],
[17.],
[17.6],
[16.5],
[18.2],
[16.9],
[15.5],
[19.2],
[18.5],
[31.9],
[34.1],
[35.7],
[27.4],
[25.4],
[23.],
[27.2],
[23.9],
[34.2],
[34.5],
[31.8],
[37.3],
[28.4],
[28.8],
[26.8],
[33.5],
[41.5],
[38.1],
[32.1],
[37.2],
[28.],
[26.4],
[24.3],
[19.1],
[34.3],
[29.8],
[31.3],
[37.],
[32.2],

[46.6],
[27.9],
[40.8],
[44.3],
[43.4],
[36.4],
[30.],
[44.6],
[40.9],
[33.8],
[29.8],
[32.7],
[23.7],
[35.],
[23.6],
[32.4],
[27.2],
[26.6],
[25.8],
[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. ]])
```

Normalizing

In [34]:

```
from sklearn.preprocessing import StandardScaler  
sd = StandardScaler()  
x_train = sd.fit_transform(x_train)  
x_test = sd.fit_transform(x_test)  
y_train = sd.fit_transform(y_train)  
y_test = sd.fit_transform(y_test)  
  
x_train
```

```
-----  
NameError                                Traceback (most recent call last)  
/tmp/wsuser/ipykernel_165/1600422500.py in <module>  
      1 from sklearn.preprocessing import StandardScaler  
      2 sd = StandardScaler()  
----> 3 x_train = sd.fit_transform(x_train)  
      4 x_test = sd.fit_transform(x_test)  
      5 y_train = sd.fit_transform(y_train)
```

NameError: name 'x_train' is not defined

Splitting into train and test data.

In [35]:

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

In [36]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

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

random forest regressor

In [37]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [38]:

```
rf= RandomForestRegressor(n_estimators=10)  
rf.fit(x_train,np.ravel(y_train))
```

Out[38]:

```
RandomForestRegressor(n_estimators=10)
```

In [39]:

```
x_train.shape
```

Out[39]:

```
(318, 6)
```

In [40]:

```
!!pip install ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3.4)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (0.10.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (1.15.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (2.0.4)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm-watson-machine-learning) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm-watson-machine-learning) (3.0.4)
```

In [43]:

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "k0ToNjB4fREMsVxEr0C3pjHT0bNJzgZvVt1S0SikVpMJ",
}
client = APIClient(wml_credentials)
```

In [44]:

```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    print(space)
    return(next(item for item in space['resources'] if item['entity']['name'] == space_name) ['metadata']['id'])
```

In [45]:

```
space_uid = guid_from_space_name(client, 'models')
print("Space UID-" + space_uid)

{'resources': [{'entity': {'compute': [{'crn': 'crn:v1:bluemix:public:pm-20:us-south:a/d1
```

2096019c744baabc99e2caa9c44ac5:dbc6da1f-aca8-4c0f-937f-da20bd5e08c6::', 'guid': 'dbc6da1f-aca8-4c0f-937f-da20bd5e08c6', 'name': 'WatsonMachine Learning-72', 'type': 'machine_learning'}, {'description': '', 'name': 'models', 'scope': {'bss_account_id': 'd12096019c744baabc99e2caa9c44ac5'}, 'stage': {'production': False}, 'status': {'state': 'active'}, 'storage': {'properties': {'bucket_name': '2e6d79af-2319-41ef-8bb1-13843958cd79', 'bucket_region': 'us-south', 'credentials': {'admin': {'access_key_id': 'e2dcdab6300f4292acfd3a9e99fe8419', 'api_key': 'Ccb36wfmVayFlvEWymud07rYIVHUT18VXxjXXSf8wNVx', 'secret_access_key': '09d27d5ba8570a63f8ecd4477903c61c66702acd0a8a0cb2', 'service_id': 'ServiceId-113a7385-0acf-4aca-b0a4-f20ee5949b94'}, 'editor': {'access_key_id': '0b31cfa63a054a40bbe5cbe6a1ddc189', 'api_key': 'Nrtr6aPyPH387BLYG2rJrcg3kiA1-ZVXbGR3J7CZQK9e', 'resource_key_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/d12096019c744baabc99e2caa9c44ac5:71972f87-6ebf-4b35-907f-5b09ff80a63a::', 'secret_access_key': '8db16c6ef99ec8799a31e00fa98d5952b212f3198510de25', 'service_id': 'ServiceId-48a7874c-09b6-4ba0-8c14-d4b63928000f'}, 'viewer': {'access_key_id': '6a971a0391e947e7a10aa6d0a205eee2', 'api_key': 'p_RHNGsyVx7ybtSaRSn8Osq2ZkOsOhj8PPAYQEdgGoLK', 'resource_key_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/d12096019c744baabc99e2caa9c44ac5:71972f87-6ebf-4b35-907f-5b09ff80a63a::', 'secret_access_key': 'c530568f37883f9c765343301136c7eb763cc3ccf4197718', 'service_id': 'ServiceId-1a4e051b-6867-4ac8-acab-b942b7b43a75'}}}, {'endpoint_url': 'https://s3.us-south.cloud-object-storage.appdomain.cloud', 'guid': '71972f87-6ebf-4b35-907f-5b09ff80a63a', 'resource_crn': 'crn:v1:bluemix:public:cloud-object-storage:global:a/d12096019c744baabc99e2caa9c44ac5:71972f87-6ebf-4b35-907f-5b09ff80a63a::', 'type': 'bmcos_object_storage'}}}, {'metadata': {'created_at': '2022-11-12T15:24:43.319Z', 'creator_id': 'IBMid-6640044ZX7', 'id': 'a7c26d83-e37a-4e0b-b024-7d3f1f77740b', 'updated_at': '2022-11-12T15:25:03.183Z', 'url': '/v2/spaces/a7c26d83-e37a-4e0b-b024-7d3f1f77740b'}}}]

Space UID-a7c26d83-e37a-4e0b-b024-7d3f1f77740b

In [46]:

```
client.set.default_space(space_uid)
```

Out[46]:

'SUCCESS'

In [47]:

```
client.software_specifications.list()
```

NAME	ASSET_ID	TYPE
default_py3.6	0062b8c9-8b7d-44a0-a9b9-46c416adcbd9	base
kernel-spark3.2-scala2.12	020d69ce-7ac1-5e68-ac1a-31189867356a	base
pytorch-onnx_1.3-py3.7-edt	069ea134-3346-5748-b513-49120e15d288	base
scikit-learn_0.20-py3.6	09c5a1d0-9c1e-4473-a344-eb7b665ff687	base
spark-mllib_3.0-scala_2.12	09f4cff0-90a7-5899-b9ed-1ef348aebdee	base
pytorch-onnx_rt22.1-py3.9	0b848dd4-e681-5599-be41-b5f6fccc6471	base
ai-function_0.1-py3.6	0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda	base
shiny-r3.6	0e6e79df-875e-4f24-8ae9-62dcc2148306	base
tensorflow_2.4-py3.7-horovod	1092590a-307d-563d-9b62-4eb7d64b3f22	base
pytorch_1.1-py3.6	10ac12d6-6b30-4ccd-8392-3e922c096a92	base
tensorflow_1.15-py3.6-ddl	111e41b3-de2d-5422-a4d6-bf776828c4b7	base
autoai-kb_rt22.2-py3.10	125b6d9a-5b1f-5e8d-972a-b251688ccf40	base
runtime-22.1-py3.9	12b83a17-24d8-5082-900f-0ab31fbfd3cb	base
scikit-learn_0.22-py3.6	154010fa-5b3b-4ac1-82af-4d5ee5abbc85	base
default_r3.6	1b70aec3-ab34-4b87-8aa0-a4a3c8296a36	base
pytorch-onnx_1.3-py3.6	1bc6029a-cc97-56da-b8e0-39c3880dbbe7	base
kernel-spark3.3-r3.6	1c9e5454-f216-59dd-a20e-474a5cdf5988	base
pytorch-onnx_rt22.1-py3.9-edt	1d362186-7ad5-5b59-8b6c-9d0880bde37f	base
tensorflow_2.1-py3.6	1eb25b84-d6ed-5dde-b6a5-3fbddf1665666	base
spark-mllib_3.2	20047f72-0a98-58c7-9ff5-a77b012eb8f5	base
tensorflow_2.4-py3.8-horovod	217c16f6-178f-56bf-824a-b19f20564c49	base
runtime-22.1-py3.9-cuda	26215f05-08c3-5a41-a1b0-da66306ce658	base
do_py3.8	295addb5-9ef9-547e-9bf4-92ae3563e720	base
autoai-ts_3.8-py3.8	2aa0c932-798f-5ae9-abd6-15e0c2402fb5	base
tensorflow_1.15-py3.6	2b73a275-7cbf-420b-a912-eae7f436e0bc	base
kernel-spark3.3-py3.9	2b7961e2-e3b1-5a8c-a491-482c8368839a	base
pytorch_1.2-py3.6	2c8ef57d-2687-4b7d-acce-01f94976dac1	base
spark-mllib_2.3	2e51f700-bca0-4b0d-88dc-5c6791338875	base
pytorch-onnx_1.1-py3.6-edt	32983cea-3f32-4400-8965-dde874a8d67e	base
spark-mllib_3.0-py37	36507ebe-8770-55ba-ab2a-eafe787600e9	base
spark-mllib_2.4	390d21f8-e58b-4fac-9c55-d7ceda621326	base
autoai-ts_rt22.2-py3.10	396b2e83-0953-5b86-9a55-7ce1628a406f	base

```
--2022-11-13 15:30:12-- https://raw.githubusercontent.com/IBM/monitor-wml-model-with-watson-openscale/master/data/additional_feedback_data.json
```

```
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.1
99.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 16506 (16K) [text/plain]
Saving to: 'additional_feedback_data.json'

additional_feedback 100%[=====>] 16.12K --.-KB/s in 0.001s

2022-11-13 15:30:12 (19.6 MB/s) - 'additional_feedback_data.json' saved [16506/16506]
```

In [72]:

```
software_spec_uid = client.software_specifications.get_id_by_name("runtime-22.1-py3.9")
metadata = {
    client.repository.ModelMetaNames.NAME: 'Gradient',
    client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}

published_model = client.repository.store_model(
    model=rf,
    meta_props=metadata)
```

In [73]:

```
published_model
```

Out[73]:

```
{'entity': {'hybrid_pipeline_software_specs': [],
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
    'name': 'runtime-22.1-py3.9',
    'type': 'scikit-learn_1.0'},
  'metadata': {'created_at': '2022-11-13T15:47:35.584Z',
    'id': '34aa257f-40e9-4469-8d59-36d032d38a10',
    'modified_at': '2022-11-13T15:47:39.066Z',
    'name': 'Gradient',
    'owner': 'IBMid-6640044ZX7',
    'resource_key': '31d25942-962e-46a0-8ff9-4407d77716d0',
    'space_id': 'a7c26d83-e37a-4e0b-b024-7d3f1f77740b'},
  'system': {'warnings': []}}
```

In [74]:

```
x_train[0]
```

Out[74]:

```
array([4.000e+00, 1.210e+02, 7.600e+01, 2.511e+03, 7.200e+01, 2.000e+00])
```

In [75]:

```
rf.predict([[4.000e+00, 1.210e+02, 7.600e+01, 2.511e+03, 7.200e+01, 2.000e+00]])
```

Out[75]:

```
array([23.2])
```

In []:

```
import requests

# NOTE: you must manually set API KEY below using information retrieved from your IBM Cloud account.
API_KEY = "k0ToNjB4fREMsVxEr0C3pjHT0bNJzgZvVt1S0SikVpMJ"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
    API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [['cylinders', 'displacement', 'horsepower', 'weight', 'model year', 'origin']],
    "values": [[8,307,130,3504,70,1]]}]

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/3950d430-efb8-43ea-b408-28233df071d7/predictions?version=2022-11-13', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
```