

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
```

Importing Dataset

```
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 credentials.
# You might want to remove those credentials before you share the
# notebook.
cos_client = ibm_boto3.client(service_name='s3',
                              ibm_api_key_id='eASimw06bTq5Rjts5iIC_g7P6axTb0h5x736EBNISKSM',
                              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 = 'mlbasedvehicleperformanceanalyser-donotdelete-pr-
0isa20qlnpyfgm'
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(7)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration
model year \						
0	18.0	8	307.0	130	3504	12.0
70						
1	15.0	8	350.0	165	3693	11.5

```

70
2 18.0      8      318.0      150      3436      11.0
70
3 16.0      8      304.0      150      3433      12.0
70
4 17.0      8      302.0      140      3449      10.5
70
5 15.0      8      429.0      198      4341      10.0
70
6 14.0      8      454.0      220      4354      9.0
70

```

```

      origin      car name
0         1  chevrolet chevelle malibu
1         1      buick skylark 320
2         1    plymouth satellite
3         1      amc rebel sst
4         1      ford torino
5         1    ford galaxie 500
6         1    chevrolet impala

```

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()
```

```
6
```

```
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
```

`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
```

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

	mpg	cylinders	displacement	horsepower	weight
count	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623
std	7.815984	1.701004	104.269838	38.199187	846.841774
min	9.000000	3.000000	68.000000	46.000000	1613.000000
25%	17.500000	4.000000	104.250000	76.000000	2223.750000
50%	23.000000	4.000000	148.500000	95.000000	2803.500000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000

max	46.600000	8.000000	455.000000	230.000000	5140.000000
-----	-----------	----------	------------	------------	-------------

	acceleration	model year	origin
count	398.000000	398.000000	398.000000
mean	15.568090	76.010050	1.572864
std	2.757689	3.697627	0.802055
min	8.000000	70.000000	1.000000
25%	13.825000	73.000000	1.000000
50%	15.500000	76.000000	1.000000
75%	17.175000	79.000000	2.000000
max	24.800000	82.000000	3.000000

There is no use with car name attribute so drop it

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

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

```
corr_table
```

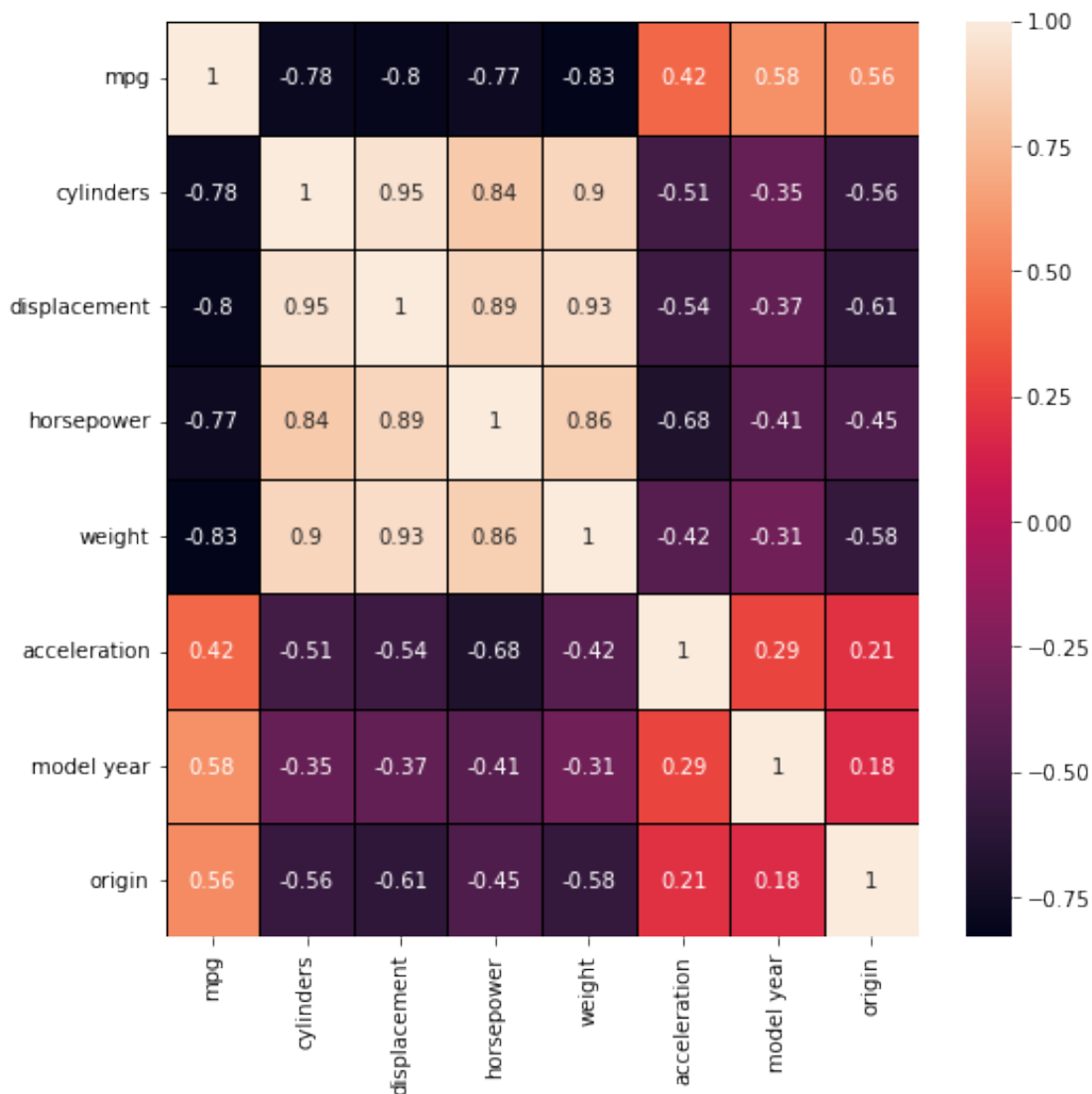
	mpg	cylinders	displacement	horsepower	weight
\					
mpg	1.000000	-0.775396	-0.804203	-0.771437	-0.831741
cylinders	-0.775396	1.000000	0.950721	0.838939	0.896017
displacement	-0.804203	0.950721	1.000000	0.893646	0.932824
horsepower	-0.771437	0.838939	0.893646	1.000000	0.860574
weight	-0.831741	0.896017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

	acceleration	model year	origin
mpg	0.420289	0.579267	0.563450
cylinders	-0.505419	-0.348746	-0.562543
displacement	-0.543684	-0.370164	-0.609409
horsepower	-0.684259	-0.411651	-0.453669
weight	-0.417457	-0.306564	-0.581024
acceleration	1.000000	0.288137	0.205873
model year	0.288137	1.000000	0.180662
origin	0.205873	0.180662	1.000000

Data Visualizations

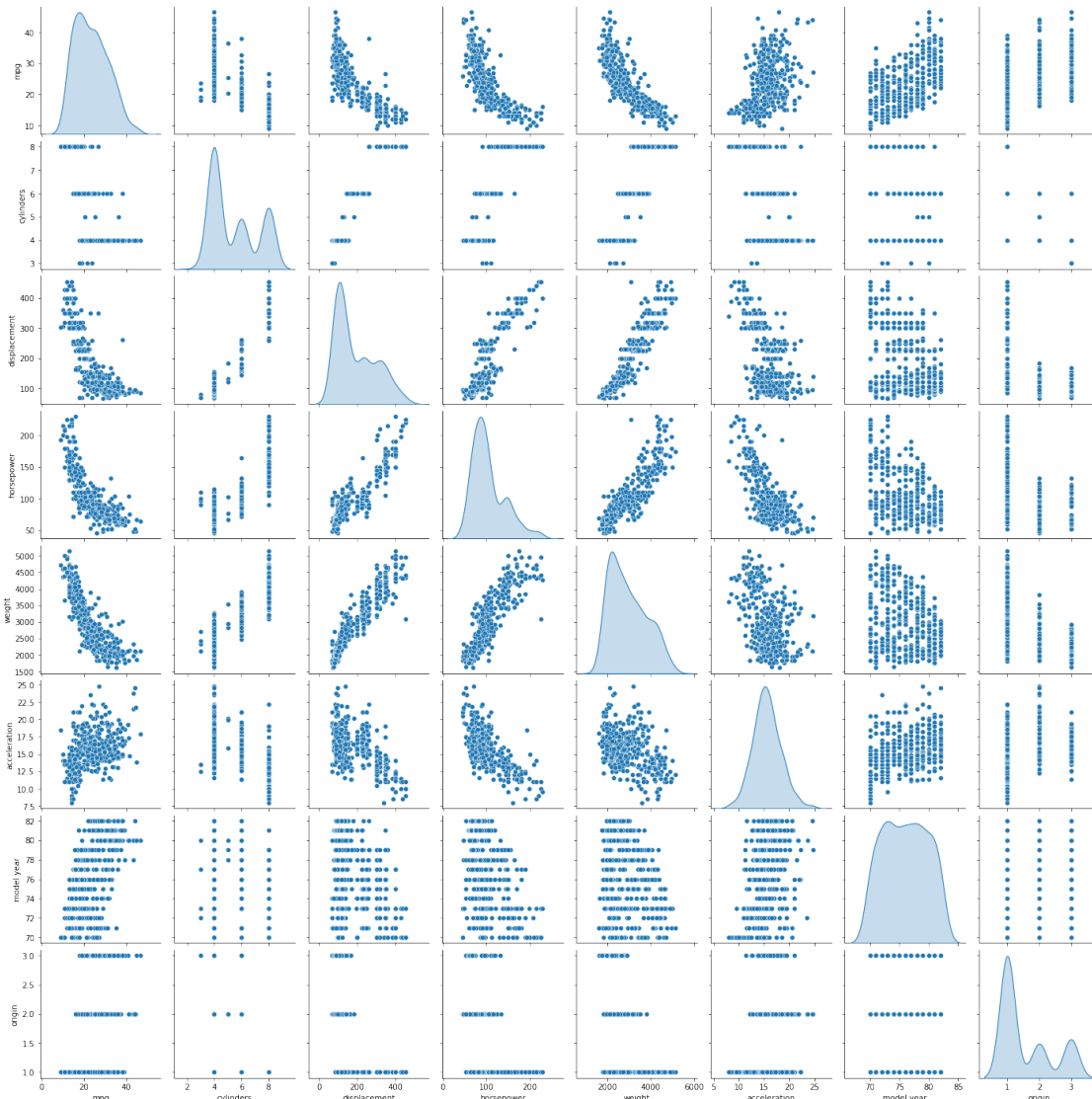
Heatmap : which represents correlation between attributes

```
sns.heatmap(dataset.corr(),annot=True,linewidth='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

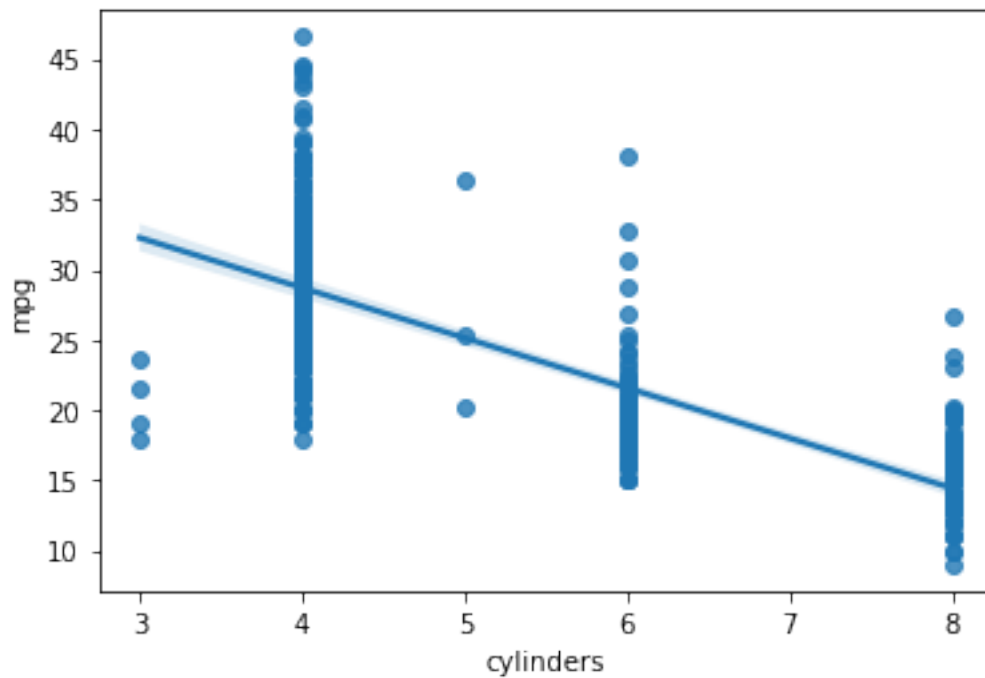
```
sns.pairplot(dataset,diag_kind='kde') #pairplot represents pairwise relation across the entire dataframe.  
plt.show()
```



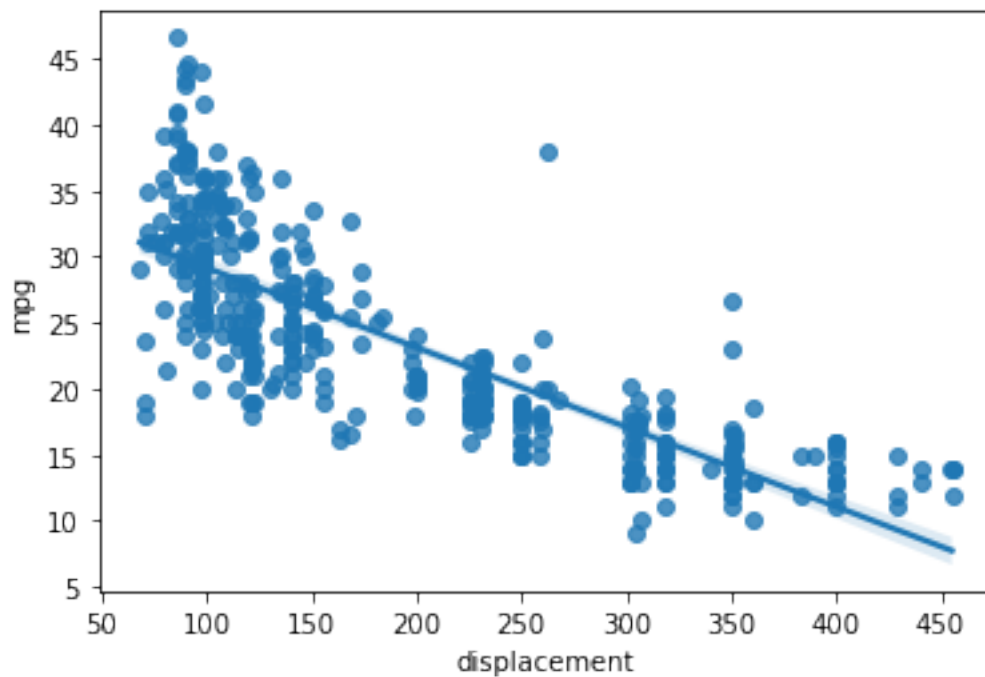
Regression plots(`regplot()`) creates a regression line between 2 parameters and helps to visualize their linear relationships.

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

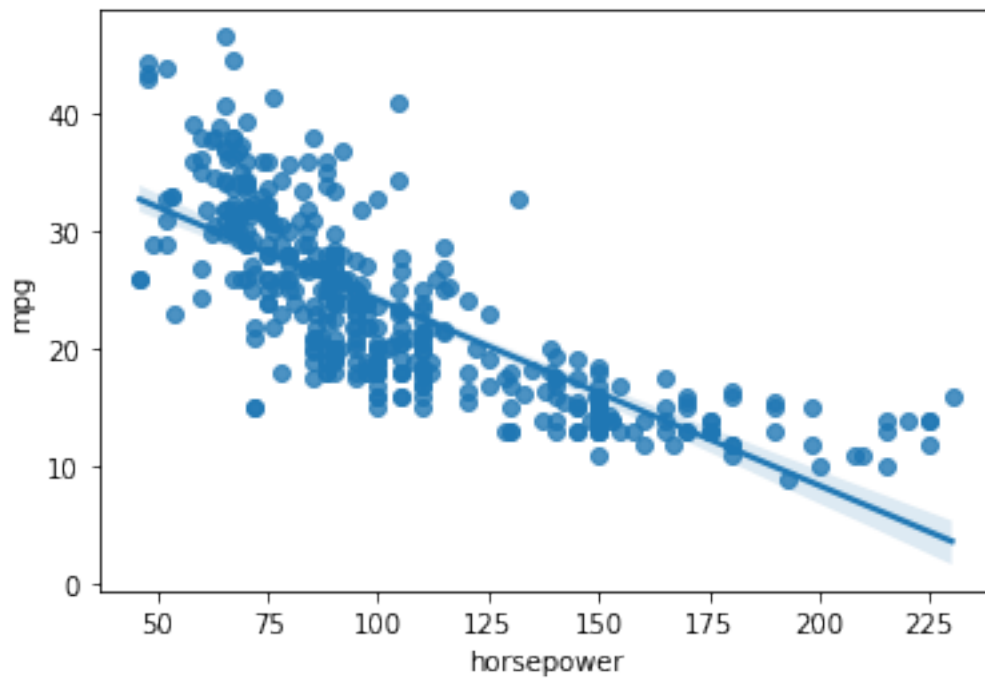
```
<AxesSubplot:xlabel='cylinders', ylabel='mpg'>
```



```
sns.regplot(x="displacement", y="mpg", data=dataset)
<AxesSubplot:xlabel='displacement', ylabel='mpg'>
```

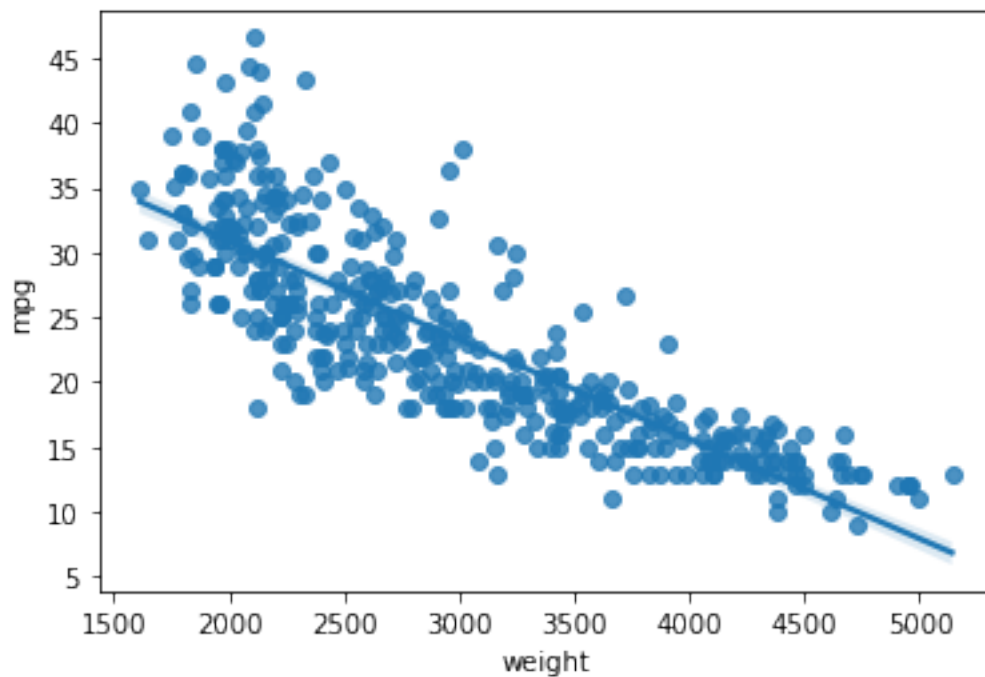


```
sns.regplot(x="horsepower", y="mpg", data=dataset)
<AxesSubplot:xlabel='horsepower', ylabel='mpg'>
```



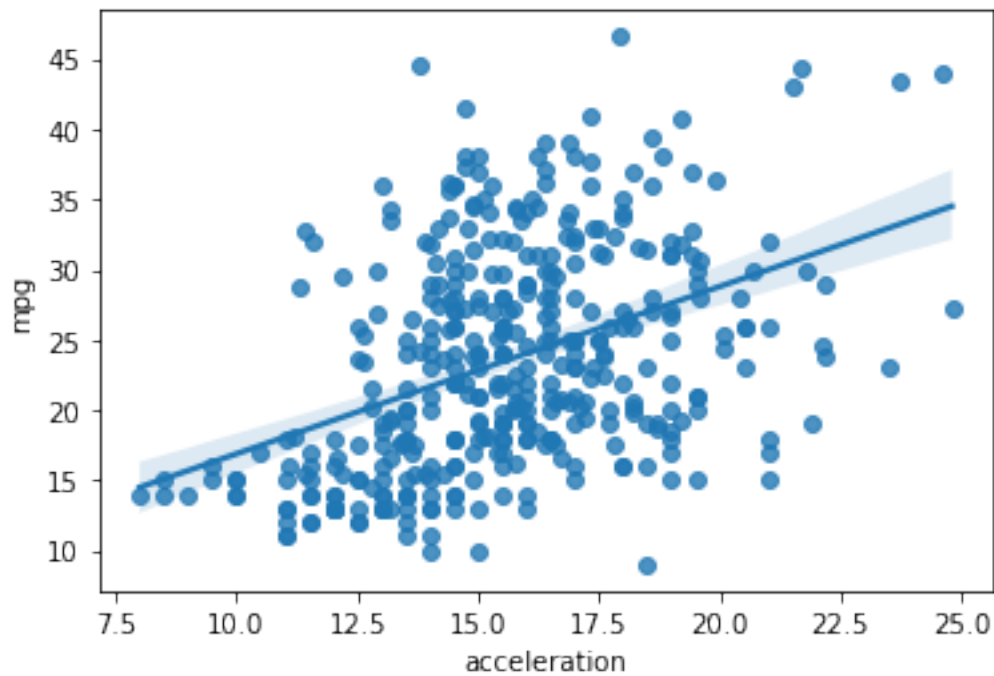
```
sns.regplot(x="weight", y="mpg", data=dataset)
```

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

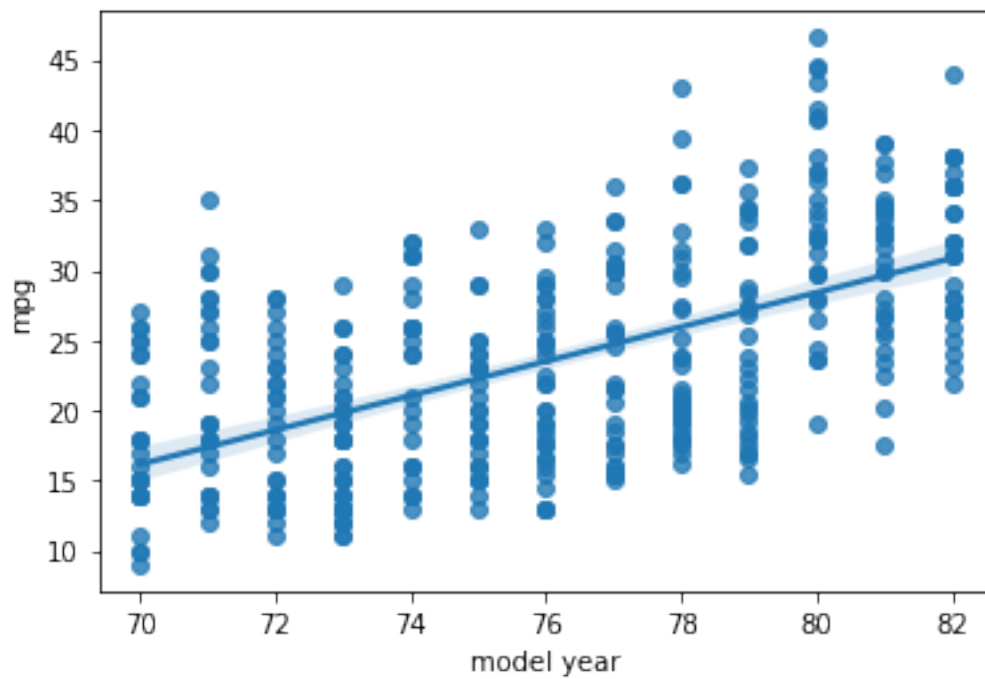


```
sns.regplot(x="acceleration", y="mpg", data=dataset)
```

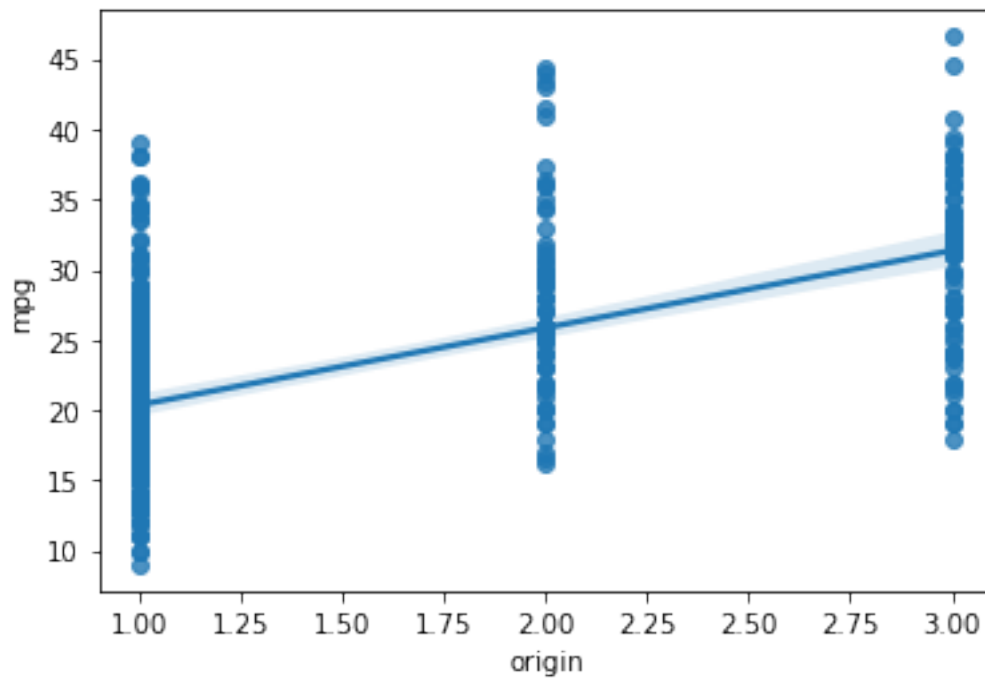
```
<AxesSubplot:xlabel='acceleration', ylabel='mpg'>
```

```
sns.regplot(x="model year", y="mpg", data=dataset)
<AxesSubplot:xlabel='model year', ylabel='mpg'>
```

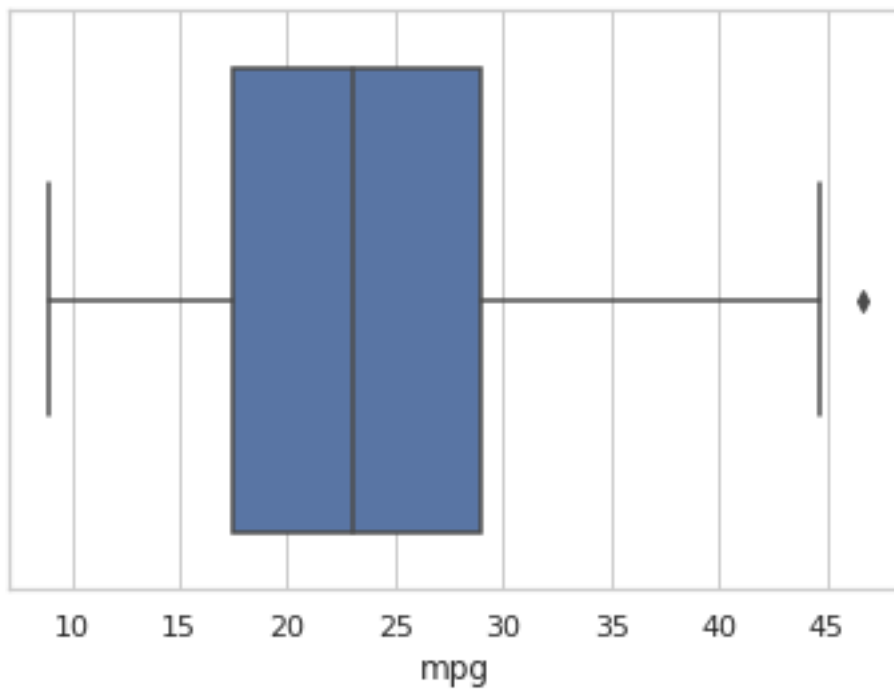


```
sns.regplot(x="origin", y="mpg", data=dataset)
<AxesSubplot:xlabel='origin', ylabel='mpg'>
```



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

<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.

```
from scipy import stats
```

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

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.

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

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

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.

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 -0.831740933244335 with a P-value of P = 2.9727995640500577e-103

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 0.4202889121016507 with a P-value of $P = 1.823091535078553e-18$

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 0.5792671330833096 with a P-value of P = 4.844935813365483e-37

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 0.5634503597738431 with a P-value of $P = 1.0114822102336483 \times 10^{-34}$

Ordinary Least Squares Statistics

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

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

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: Wed, 16 Nov 2022 Prob (F-statistic):
4.84e-104
Time: 03:33:54 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

Separating into Dependent and Independent variables

Independent variables

```
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]])
```

Dependent variables

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

```
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.],
[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. ]])
```

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

Normalisation

```
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
array([[ 1.46858608,  2.48230464,  2.97856512,  1.62455076, -
 1.61295698,
        -0.71873488],
       [ 1.46858608,  1.48729292,  1.55429873,  0.84358808, -
 1.61295698,
        -0.71873488],
       [-0.86550411, -0.70364636, -0.64684023, -0.36507278,
 0.82235108,
        -0.71873488],
       ...,
       [-0.86550411, -1.21071964, -1.44960856, -1.31380657, -
 0.80118763,
        0.53032865],
       [ 0.30154098,  0.53055088, -0.12892518,  0.35799706, -1.3423672
,
        -0.71873488],
       [-0.86550411, -1.00023639, -0.87990201, -0.89319732, -
 0.26000806,
        0.53032865]])

```

decision tree regressor

```

from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(random_state=0,criterion="mae")
dt.fit(x_train,y_train)

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/tree/
_classes.py:366: FutureWarning: Criterion 'mae' was deprecated in v1.0
and will be removed in version 1.2. Use `criterion='absolute_error'`
which is equivalent.
  warnings.warn(

```

```
DecisionTreeRegressor(criterion='mae', random_state=0)
```

```

import pickle
pickle.dump(dt,open('decision_model.pkl','wb'))

```

```

y_pred=dt.predict(x_test)
y_pred

```

```

array([-1.21248604,  0.40341575, -1.21248604, -0.56612532, -
 0.82466961,
        0.98514039,  2.07102639, -0.17830889, -1.08321389,
 0.20950753,

```

```
1.43759289, 2.74324153, -0.43685318, 0.33877968, -
0.95394175,
1.41173846, 0.41634296, 0.33877968, -0.82466961,
1.39881125,
-0.95394175, -0.04903675, -0.24294497, -0.56612532,
1.33417517,
0.20950753, 0.79123218, 0.98514039, 0.98514039, -
0.43685318,
-0.88930568, 1.50222896, -1.08321389, 1.04977646, -
0.54027089,
-0.39807154, -1.08321389, -0.95394175, 0.79123218, -
1.60030246])
```

y_test

```
array([[ -1.29002284],
       [ 0.03307751],
       [ -1.41030469],
       [ -0.44804989],
       [ -0.80889544],
       [ 1.23589601],
       [ 1.12764234],
       [ -0.56833174],
       [ -1.16974099],
       [ -0.14734527],
       [ 1.94555892],
       [ 1.50051608],
       [ -0.80889544],
       [ -0.20748619],
       [ -1.10960007],
       [ 1.3200933 ],
       [ 0.75476861],
       [ 0.27364121],
       [ -0.80889544],
       [ 1.51254426],
       [ -1.10960007],
       [ -0.20748619],
       [ -0.08720434],
       [ -0.80889544],
       [ 1.17575508],
       [ 0.08119025],
       [ 1.36820604],
       [ 1.11561416],
       [ 0.63448676],
       [ -1.04945914],
       [ -0.73672633],
       [ 1.47645971],
       [ -1.16974099],
       [ 1.05547323],
       [ -0.2796553 ],
       [ -0.08720434],
```



```

        [-0.68861359],
        [-0.94120548],
        [ 0.86302227],
        [-1.53058654]])

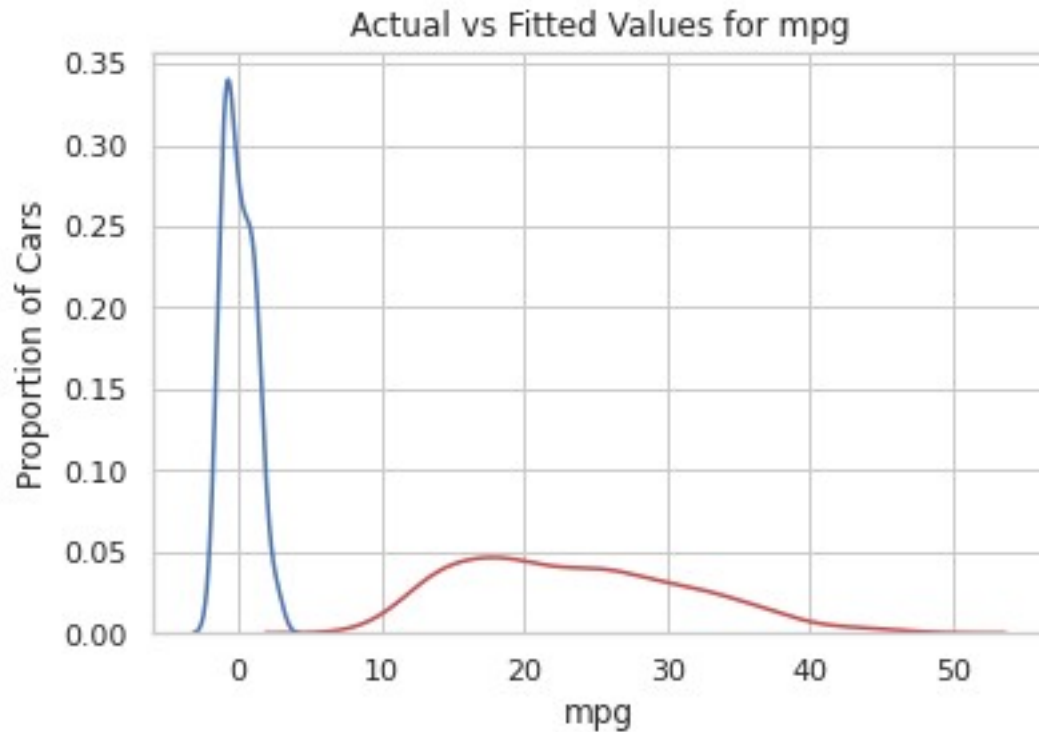
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r",
label="Actual Value")
sns.distplot(y_pred, hist=False, color="b", label="Fitted Values" ,
ax=ax1)

plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density
plots).
  warnings.warn(msg, FutureWarning)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distrib
utions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use
either `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)

```



We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

$$R\text{-squared} = \frac{\text{Explained variation}}{\text{Total variation}}$$

Mean Squared Error (MSE)

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
r2_score(y_test, y_pred)
```

```
0.8693935947768933
```

```
mean_squared_error(y_test, y_pred)
```

```
0.1306064052231067
```

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
0.36139508190221226
```

random forest regressor

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

```

rf=
RandomForestRegressor(n_estimators=10,random_state=0,criterion='mae')
rf.fit(x_train,y_train)

/tmp/wsuser/ipykernel_165/4164760693.py:2: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples,), for example using ravel().
    rf.fit(x_train,y_train)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/ensembl
e/_forest.py:403: FutureWarning: Criterion 'mae' was deprecated in
v1.0 and will be removed in version 1.2. Use
`criterion='absolute_error'` which is equivalent.
    warn(

RandomForestRegressor(criterion='mae', n_estimators=10,
random_state=0)

y_pred2=rf.predict(x_test)
y_pred2

array([-1.22541325,  0.04145375, -1.27712211, -0.21321237, -
0.73417911,
        0.94635875,  1.42983656,  0.05438096, -1.08321389, -
0.09057716,
        1.36778593,  1.89133811, -0.52734368,  0.26380183, -
0.95394175,
        1.04460558,  0.5818113 ,  0.23406924, -0.86991486,
1.10794893,
        -0.97333257,  0.19140943,  0.03757558, -0.29982471,
0.86879546,
        -0.02318232,  1.1066562 ,  1.07563089,  1.09760715, -
0.87767119,
        -0.39677881,  1.0278002 , -0.82596233,  0.97221318, -
0.18735794,
        -0.03610954, -0.59844336, -0.92808732,  1.28375904, -
1.47103032])

ax1 = sns.distplot(dataset['mpg'], hist=False, color="r",
label="Actual Value")
sns.distplot(y_pred2, hist=False, color="b", label="Fitted Values" ,
ax=ax1)

plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()

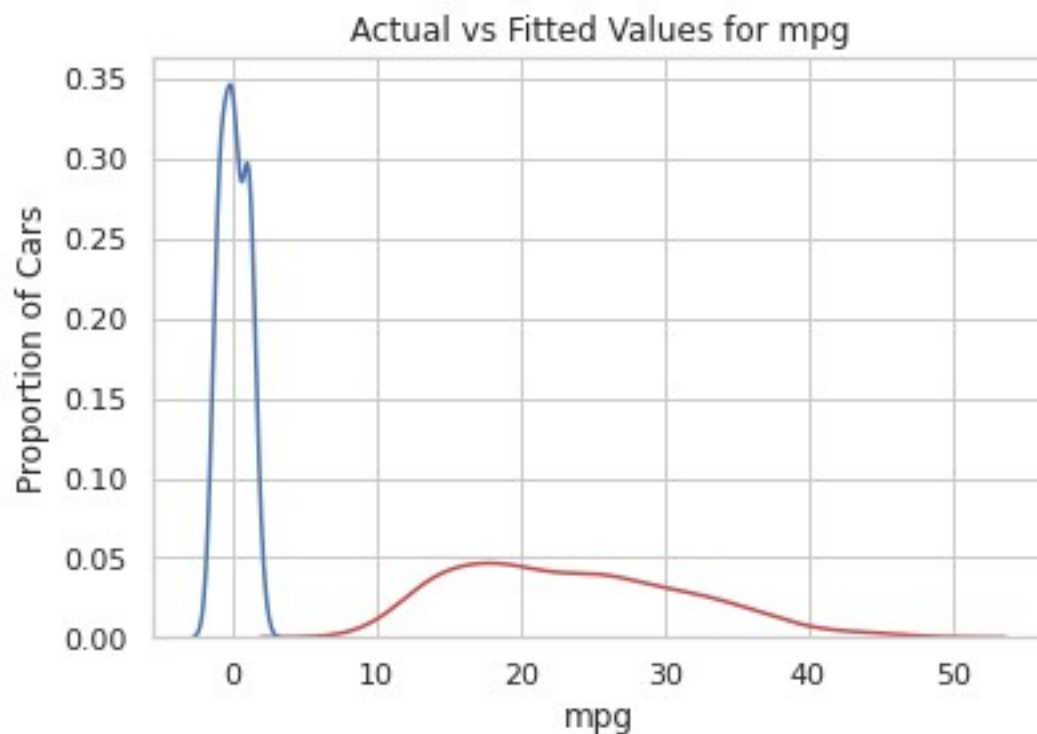
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
```

```
warnings.warn(msg, FutureWarning)
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
```

```
warnings.warn(msg, FutureWarning)
```



We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
r2_score(y_test, y_pred2)
```

```
0.9172449209441422
```

```
mean_squared_error(y_test, y_pred2)
```

```
0.08275507905585772
```

```
np.sqrt(mean_squared_error(y_test, y_pred2))
```

0.28767182527292745

linear regression

```
from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
```

```
LinearRegression()
```

```
y_pred3=mr.predict(x_test)
y_pred3
```

```
array([[ -1.48354417],
       [  0.02018821],
       [ -1.68903988],
       [ -0.42532853],
       [ -0.85893857],
       [  0.68603096],
       [  1.21891251],
       [ -0.13204094],
       [ -1.23799478],
       [  0.3208166 ],
       [  1.05032664],
       [  1.27944186],
       [ -0.35801006],
       [  0.22377607],
       [ -1.07318175],
       [  0.84489227],
       [  0.55090001],
       [  0.59162161],
       [ -0.86938749],
       [  0.90060746],
       [ -1.19252105],
       [  0.06263748],
       [  0.34334107],
       [ -0.49842421],
       [  0.7257113 ],
       [  0.5306685 ],
       [  0.90224828],
       [  0.75809371],
       [  0.75230078],
       [ -0.81562435],
       [ -0.51790152],
       [  0.92628973],
       [ -0.41719037],
       [  1.02755713],
       [ -0.05714634],
       [  0.26677556],
       [ -0.37496781],
```

```

        [-1.01152038],
        [ 1.04592662],
        [-2.01630218]])

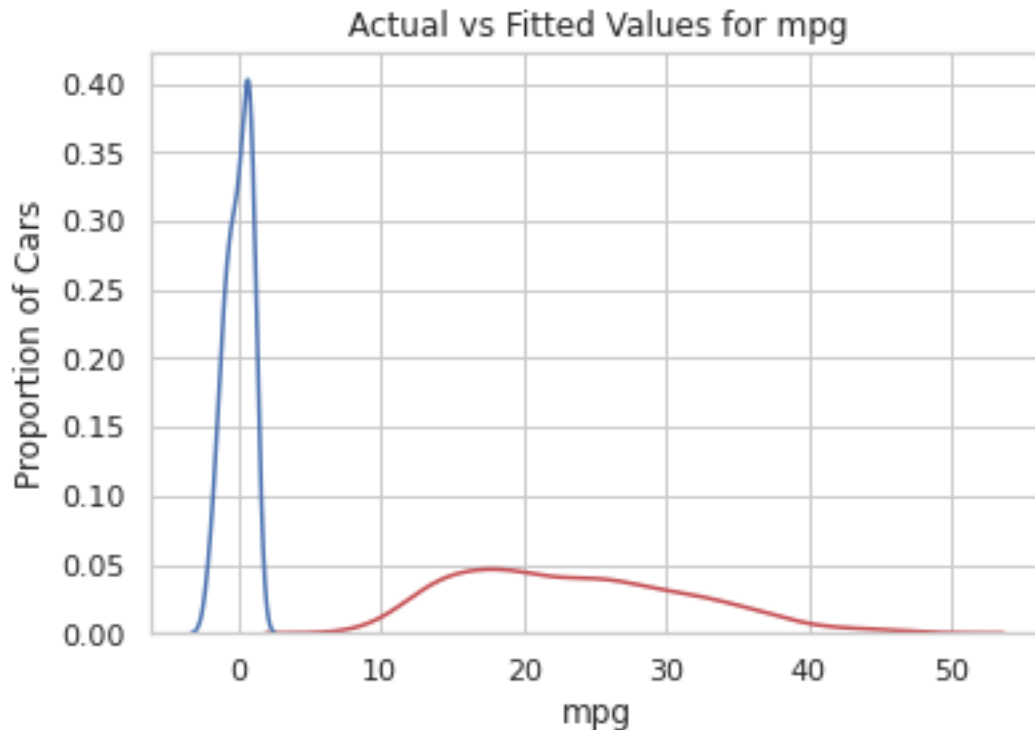
ax1 = sns.distplot(dataset['mpg'], hist=False, color="r",
label="Actual Value")
sns.distplot(y_pred3, hist=False, color="b", label="Fitted Values" ,
ax=ax1)


plt.title('Actual vs Fitted Values for mpg')
plt.xlabel('mpg')
plt.ylabel('Proportion of Cars')

plt.show()
plt.close()

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `kdeplot` (an axes-level function for kernel density
plots).
  warnings.warn(msg, FutureWarning)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/distrib
utions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use
either `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)

```



We can see that the fitted values are not as close to the actual values, since the two distributions overlap a bit. However, there is definitely some room for improvement.

```
from sklearn.metrics import r2_score, mean_squared_error
r2_score(y_test, y_pred3)
```

```
0.8631101197005312
```

```
mean_squared_error(y_test, y_pred3)
```

```
0.13688988029946877
```

```
np.sqrt(mean_squared_error(y_test, y_pred3))
```

```
0.3699863244762822
```

Conclusion: When comparing models, the model with the higher R-squared value is a better fit for the data. When comparing models, the model with the smallest MSE value is a better fit for the data.

Comparing these three models, we conclude that the DecisionTree model is the best model to be able to predict mpg from our dataset. But We use Random Forest Regressor

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
```

```
"apikey": "c0AaDte5MvL14V1NwKh1SgGHH34bvKX_VN8iFiLyAPQ5"
```

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

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

```
Space UID = ec50822e-7eed-4693-abad-e892683a6177
```

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

```
'SUCCESS'
```

```
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-b6a5-3fbdf1665666
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	
xgboost_0.82-py3.6	39e31acd-5f30-41dc-ae44-60233c80306e
base	
pytorch-onnx_1.2-py3.6-edt	40589d0e-7019-4e28-8daa-fb03b6f4fe12
base	
pytorch-onnx_rt22.2-py3.10	40e73f55-783a-5535-b3fa-0c8b94291431
base	
default_r36py38	41c247d3-45f8-5a71-b065-8580229facf0
base	
autoai-ts_rt22.1-py3.9	4269d26e-07ba-5d40-8f66-2d495b0c71f7
base	
autoai-obm_3.0	42b92e18-d9ab-567f-988a-4240ba1ed5f7
base	

pmml-3.0_4.3	493bcb95-16f1-5bc5-bee8-81b8af80e9c7
base	
spark-mllib_2.4-r_3.6	49403dff-92e9-4c87-a3d7-a42d0021c095
base	
xgboost_0.90-py3.6	4ff8d6c2-1343-4c18-85e1-689c965304d3
base	
pytorch-onnx_1.1-py3.6	50f95b2a-bc16-43bb-bc94-b0bed208c60b
base	
autoai-ts_3.9-py3.8	52c57136-80fa-572e-8728-a5e7cbb42cde
base	
spark-mllib_2.4-scala_2.11	55a70f99-7320-4be5-9fb9-9edb5a443af5
base	
spark-mllib_3.0	5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9
base	
autoai-obm_2.0	5c2e37fa-80b8-5e77-840f-d912469614ee
base	
spss-modeler_18.1	5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b
base	
cuda-py3.8	5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e
base	
autoai-kb_3.1-py3.7	632d4b22-10aa-5180-88f0-f52dfb6444d7
base	
pytorch-onnx_1.7-py3.8	634d3cdc-b562-5bf9-a2d4-ea90a478456b
base	

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

```
software_spec_uid =
client.software_specifications.get_uid_by_name("default_py3.6")
software_spec_uid
```

```
'0062b8c9-8b7d-44a0-a9b9-46c416adcbd9'
```

```
!pip install -U pyspark==2.1.2
```

```
# model_details = client.repository.store_model(model=rf,meta_props={
#     client.repository.ModelMetaNames.NAME:"car-performance",
#     client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
#
# client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid }
# )
```

```
# model_id = client.repository.get_model_uid(model_details)
```

```
Requirement already satisfied: pyspark==2.1.2 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (2.1.2)
Requirement already satisfied: py4j==0.10.4 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from pyspark==2.1.2) (0.10.4)
```

```
# metadata = {
#     client.repository.ModelMetaNames.NAME: 'ML-Model',
#     client.repository.ModelMetaNames.TYPE: 'scikit-learn_0.22',
#     client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
# }
```

```
# published_model = client.repository.store_model(
#     model=rf,
#     meta_props=metadata,
#     training_data=x_train,
#     training_target=y_train)
```

Traceback (most recent call last):

```
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/mlrepositoryclient/model_collection.py", line
254, in _save_scikit_pipeline_model
    model_artifact = self._create_pipeline_model(artifact,
query_param)
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/mlrepositoryclient/model_collection.py", line
514, in _create_pipeline_model
    model_artifact =
self._create_pipeline_model_v4_cloud(model_artifact, query_param)
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/mlrepositoryclient/model_collection.py", line
1005, in _create_pipeline_model_v4_cloud
    model_output =
self.repository_api.ml_assets_model_creation_v4_cloud(model_input,
query_param, headers)
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/apis/repository_api.py", line
301, in ml_assets_model_creation_v4_cloud
    (data) =
self.ml_assets_model_creation_v4_cloud_with_http_info(models_input, que
ry_param, headers, **kwargs)
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/apis/repository_api.py", line
374, in ml_assets_model_creation_v4_cloud_with_http_info
    return self.api_client.call_api(resource_path, 'POST',
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/api_client.py", line 386, in
call_api
```

```

        return self.__call_api(resource_path, method,
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/api_client.py", line 206, in
__call_api
    response_data = self.request(method, url,
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/api_client.py", line 421, in
request
    return self.rest_client.POST(url,
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/rest.py", line 230, in POST
    return self.request("POST", url,
File
"/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mac
hine_learning/libs/repo/swagger_client/rest.py", line 202, in request
    raise ApiException(http_resp=r)
ibm_watson_machine_learning.libs.repo.swagger_client.rest.ApiException
: (400)
Reason: Bad Request
HTTP response headers: HTTPHeaderDict({'Date': 'Wed, 16 Nov 2022
03:34:03 GMT', 'Content-Type': 'application/json', 'Content-Length':
'292', 'Connection': 'keep-alive', 'server-timing':
'intid;desc=e7597afd3f05a666, intid;desc=e7597afd3f05a666', 'Strict-
Transport-Security': 'max-age=31536000; includeSubDomains', 'CF-Cache-
Status': 'DYNAMIC', 'Server': 'cloudflare', 'CF-RAY':
'76ad2b2d4c29e470-DFW'})
HTTP response body: {
  "trace": "f9988c9fccf6fb2c9d7662c5c902f67a",
  "errors": [{
    "code": "invalid_request_entity",
    "message": "Invalid request entity: Unsupported software
specification 'default_py3.6'.",
    "more_info": "https://cloud.ibm.com/apidocs/machine-learning"
  }],
  "status_code": "400"
}

```

Publishing model failed.

Reason: (400)

Reason: Bad Request

```

HTTP response headers: HTTPHeaderDict({'Date': 'Wed, 16 Nov 2022
03:34:03 GMT', 'Content-Type': 'application/json', 'Content-Length':
'292', 'Connection': 'keep-alive', 'server-timing':
'intid;desc=e7597afd3f05a666, intid;desc=e7597afd3f05a666', 'Strict-
Transport-Security': 'max-age=31536000; includeSubDomains', 'CF-Cache-

```

```
Status': 'DYNAMIC', 'Server': 'cloudflare', 'CF-RAY':
'76ad2b2d4c29e470-DFW'})
HTTP response body: {
  "trace": "f9988c9fccf6fb2c9d7662c5c902f67a",
  "errors": [{
    "code": "invalid_request_entity",
    "message": "Invalid request entity: Unsupported software
specification 'default_py3.6'.",
    "more_info": "https://cloud.ibm.com/apidocs/machine-learning"
  }],
  "status_code": "400"
}
```

```
-----
-----
ApiException                                Traceback (most recent call
last)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/models.py in _publish_from_object_cloud(self, model,
meta_props, training_data, training_target, pipeline, feature_names,
label_column_names, project_id)
    2585         query_param_for_repo_client =
self._client._params()
-> 2586             saved_model =
self._client.repository._ml_repository_client.models.save(model_artifa
ct,
    2587
query_param=query_param_for_repo_client)

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/mlrepositoryclient/model_collection.py in
save(self, artifact, query_param)
    212         if
artifact.meta.prop(MetaNames.TYPE).startswith("scikit-learn"):
--> 213             return self._save_scikit_pipeline_model(artifact,
query_param)
    214         elif
artifact.meta.prop(MetaNames.TYPE).startswith("xgboost"):

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/mlrepositoryclient/model_collection.py in
_save_scikit_pipeline_model(self, artifact, query_param)
    265         print(traceback.format_exc())
-> 266         raise e
    267

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/mlrepositoryclient/model_collection.py in
_save_scikit_pipeline_model(self, artifact, query_param)
```

```

253             if artifact.uid is None:
--> 254                 model_artifact =
self._create_pipeline_model(artifact, query_param)
255             else:

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/mlrepositoryclient/model_collection.py in
_create_pipeline_model(self, model_artifact, query_param)
513         def _create_pipeline_model(self, model_artifact,
query_param=None):
--> 514             model_artifact =
self._create_pipeline_model_v4_cloud(model_artifact, query_param)
515             return model_artifact

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/mlrepositoryclient/model_collection.py in
_create_pipeline_model_v4_cloud(self, model_artifact, query_param)
1004
-> 1005         model_output =
self.repository_api.ml_assets_model_creation_v4_cloud(model_input,
query_param, headers)
1006         if model_output is not None:

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/apis/repository_api.py in
ml_assets_model_creation_v4_cloud(self, models_input, query_param,
headers, **kwargs)
300         else:
--> 301             (data) =
self.ml_assets_model_creation_v4_cloud_with_http_info(models_input, que
ry_param, headers, **kwargs)
302             return data

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/apis/repository_api.py in
ml_assets_model_creation_v4_cloud_with_http_info(self, models_input,
query_param, headers, **kwargs)
373
--> 374         return self.api_client.call_api(resource_path, 'POST',
375                                             path_params,

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/api_client.py in call_api(self,
resource_path, method, path_params, query_params, header_params, body,
post_params, files, response_type, auth_settings, callback,
_return_http_data_only)
385         if callback is None:
--> 386             return self.__call_api(resource_path, method,
387                                     path_params, query_params,
header_params,

```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/api_client.py in
__call_api(self, resource_path, method, path_params, query_params,
header_params, body, post_params, files, response_type, auth_settings,
callback, _return_http_data_only)
```

```
205
--> 206         response_data = self.request(method, url,
207
query_params=query_params,
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/api_client.py in request(self,
method, url, query_params, headers, post_params, body)
```

```
420         elif method == "POST":
--> 421             return self.rest_client.POST(url,
422
query_params=query_params,
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/rest.py in POST(self, url,
headers, query_params, post_params, body)
```

```
229     def POST(self, url, headers=None, query_params=None,
post_params=None, body=None):
--> 230         return self.request("POST", url,
231                             headers=headers,
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/libs/repo/swagger_client/rest.py in request(self, method,
url, query_params, headers, body, post_params)
```

```
201         if r.status not in range(200, 206):
--> 202             raise ApiException(http_resp=r)
203
```

ApiException: (400)

Reason: Bad Request

HTTP response headers: HTTPHeaderDict({'Date': 'Wed, 16 Nov 2022 03:34:03 GMT', 'Content-Type': 'application/json', 'Content-Length': '292', 'Connection': 'keep-alive', 'server-timing': 'intid;desc=e7597afd3f05a666, intid;desc=e7597afd3f05a666', 'Strict-Transport-Security': 'max-age=31536000; includeSubDomains', 'CF-Cache-Status': 'DYNAMIC', 'Server': 'cloudflare', 'CF-RAY': '76ad2b2d4c29e470-DFW'})

HTTP response body: {

```
    "trace": "f9988c9fccf6fb2c9d7662c5c902f67a",
    "errors": [{
        "code": "invalid_request_entity",
        "message": "Invalid request entity: Unsupported software
specification 'default_py3.6'.",
        "more_info": "https://cloud.ibm.com/apidocs/machine-learning"
```

```

    }},
    "status_code": "400"
}

```

During handling of the above exception, another exception occurred:

```

WMLClientError                                Traceback (most recent call
last)
/tmp/wsuser/ipykernel_165/2277397662.py in <module>
      5 }
      6
----> 7 published_model = client.repository.store_model(
      8     model=rf,
      9     meta_props=metadata,

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/repository.py in store_model(self, model, meta_props,
training_data, training_target, pipeline, feature_names,
label_column_names, subtrainingId, round_number, experiment_metadata)
    410     """
    411
--> 412     return self._client._models.store(model,
meta_props=meta_props, training_data=training_data,
training_target=training_target, pipeline=pipeline,
feature_names=feature_names,
label_column_names=label_column_names, subtrainingId=subtrainingId, roun
d_number=round_number, experiment_metadata=experiment_metadata)
    413
    414     @docstring_parameter({'str_type': STR_TYPE_NAME})

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/models.py in store(self, model, meta_props,
training_data, training_target, pipeline, version, artifactId,
feature_names, label_column_names, subtrainingId, round_number,
experiment_metadata)
    1609         meta_props=new_meta_props)
    1610         else:
-> 1611             saved_model =
self._publish_from_object_cloud(model=model,
    1612
meta_props=meta_props,
    1613
training_data=training_data,

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_mach
ine_learning/models.py in _publish_from_object_cloud(self, model,
meta_props, training_data, training_target, pipeline, feature_names,
label_column_names, project_id)
    2589

```



```

2590         except Exception as e:
-> 2591             raise WMLClientError(u'Publishing model failed.',
e)
2592
2593     def _upload_autoai_model_content(self, file, url,
qparams):

WMLClientError: Publishing model failed.
Reason: (400)
Reason: Bad Request
HTTP response headers: HTTPHeaderDict({'Date': 'Wed, 16 Nov 2022
03:34:03 GMT', 'Content-Type': 'application/json', 'Content-Length':
'292', 'Connection': 'keep-alive', 'server-timing':
'intid;desc=e7597afd3f05a666, intid;desc=e7597afd3f05a666', 'Strict-
Transport-Security': 'max-age=31536000; includeSubDomains', 'CF-Cache-
Status': 'DYNAMIC', 'Server': 'cloudflare', 'CF-RAY':
'76ad2b2d4c29e470-DFW'})
HTTP response body: {
  "trace": "f9988c9fccf6fb2c9d7662c5c902f67a",
  "errors": [{
    "code": "invalid_request_entity",
    "message": "Invalid request entity: Unsupported software
specification 'default_py3.6'.",
    "more_info": "https://cloud.ibm.com/apidocs/machine-learning"
  }],
  "status_code": "400"
}

```

```

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)

```

```

!wget https://https://raw.githubusercontent.com/IBM/monitor-wml-model-
with with-watson-openscale/master/data/additional_feedback_data.json

```

```
--2022-11-16 03:36:43--
```

```
https://https://raw.githubusercontent.com/IBM/monitor-wml-model-with
Resolving https (https)... failed: Name or service not known.
```

```
wget: unable to resolve host address 'https'
```

```
--2022-11-16 03:36:43--
```

```
http://with-watson-openscale/master/data/additional_feedback_data.json
Resolving with-watson-openscale (with-watson-openscale)... failed:
```

Name or service not known.
wget: unable to resolve host address 'with-watson-openscale'

```
# best_model = 'rf'  
# MODEL_NAME="Car-Performance"  
# DEPLOYMENT_NAME = "Car-Performance-Deployment"  
# BEST_MODEL = best_model
```

published_model

```
{'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': '2022-11-16T03:36:38.795Z',  
    'id': '155b0257-fd27-455f-ab11-830c50d82d81',  
    'modified_at': '2022-11-16T03:36:42.209Z',  
    'name': 'Gradient',  
    'owner': 'IBMId-6620041XAB',  
    'resource_key': 'fa8d1e62-bcc4-426e-88b9-e9c92ae3cb56',  
    'space_id': 'ec50822e-7eed-4693-abad-e892683a6177'},  
  'system': {'warnings': []}}
```

```
# model_props={  
#     client.repository.ModelMetaNames.NAME:"models".format(rf)  
# }
```

```
#  
published_model_details=client.repository.store_model(model=BEST_MODEL  
,meta_props=model_props,training_data=x_train,training_target=y_train)  
# stored_model_details =  
client.repository.create_version(trained_model_guid,  
model_uid="MODELID" ,meta_props=metadata)
```

model_id

x_train[0]

```
array([ 1.46858608,  2.48230464,  2.97856512,  1.62455076, -  
1.61295698,  
       -0.71873488])
```

```
rf.predict([[1.46858608,  2.48230464,  2.97856512,  1.62455076, -  
1.61295698,  
           -0.71873488]])
```

```
array([-1.18663161])
```

