Machine learning - Assignment 1

Name: Yudhveer Singh Runnoo

Reg. no.: 22MCA0421

Q1. Design a simple linear regression model using all possible features and find the minimum MSE and R2 Score

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
from sklearn import linear_model
from sklearn.metrics import r2_score
```

In [2]:
```
df = pd.read_csv("FuelConsumption.csv")
# take a look at the dataset
df.head()
```

Out[2]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYI
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	

In [3]:
```
# summarize the data
df.describe()
```

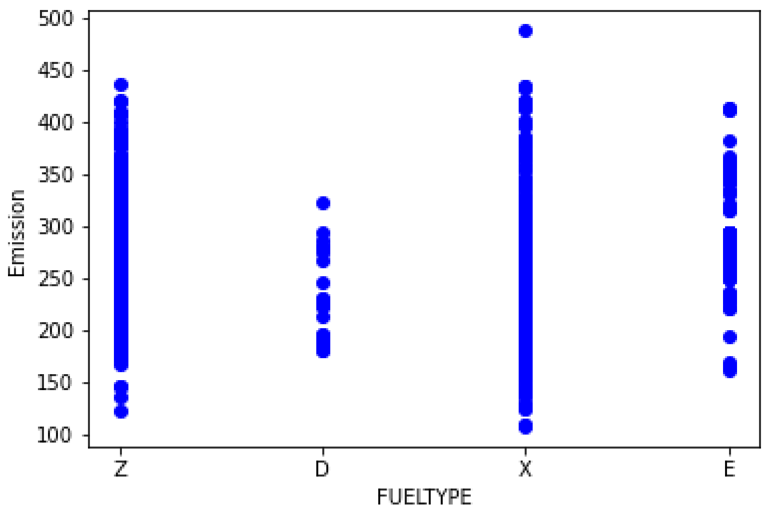
Out[3]:

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY
count	1067.0	1067.000000	1067.000000	1067.000000	1067.000000
mean	2014.0	3.346298	5.794752	13.296532	9.474602
std	0.0	1.415895	1.797447	4.101253	2.794510
min	2014.0	1.000000	3.000000	4.600000	4.900000
25%	2014.0	2.000000	4.000000	10.250000	7.500000
50%	2014.0	3.400000	6.000000	12.600000	8.800000
75%	2014.0	4.300000	8.000000	15.550000	10.850000
max	2014.0	8.400000	12.000000	30.200000	20.500000

In [4]:
```
pd.unique(df['FUELTYPE'])
```

Out[4]:
```
array(['Z', 'D', 'X', 'E'], dtype=object)
```

In [5]:
```
plt.scatter(df.FUELTYPE, df.CO2EMISSIONS, color='blue')
plt.xlabel("FUELTYPE")
plt.ylabel("Emission")
plt.show()
```

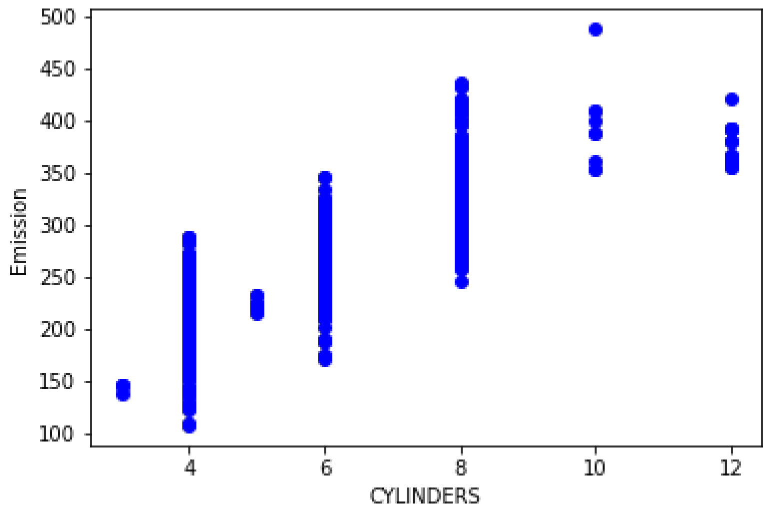


In [6]:
```
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)
```

Out[6]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

In [7]:
```
plt.scatter(cdf.CYLINDERS, cdf.CO2EMISSIONS, color='blue')
plt.xlabel("CYLINDERS")
plt.ylabel("Emission")
plt.show()
```



In [8]:
```
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
print(train)
print(test)
```

```
      ENGINESIZE  CYLINDERS  FUELCONSUMPTION_COMB  CO2EMISSIONS
0            2.0          4                   8.5           196
1            2.4          4                   9.6           221
2            1.5          4                   5.9           136
3            3.5          6                  11.1           255
4            3.5          6                  10.6           244
...          ...        ...                   ...           ...
1061         3.2          6                  11.2           258
1062         3.0          6                  11.8           271
1064         3.0          6                  11.8           271
1065         3.2          6                  11.3           260
1066         3.2          6                  12.8           294

[833 rows x 4 columns]
      ENGINESIZE  CYLINDERS  FUELCONSUMPTION_COMB  CO2EMISSIONS
5            3.5          6                  10.0           230
8            3.7          6                  11.6           267
20           2.0          4                  10.0           230
21           2.0          4                   9.3           214
22           2.0          4                  10.0           230
...          ...        ...                   ...           ...
1041         2.0          4                   6.9           186
1048         2.0          4                   7.1           192
1054         3.6          6                  12.2           281
1060         3.0          6                  11.5           264
1063         3.2          6                  11.5           264

[234 rows x 4 columns]
```

In [9]:
```
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[38.93471163]]
Intercept:  [125.52787705]
```

In [10]:
```
test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
#print(test_y)
```

```python
test_y_ = regr.predict(test_x)
#print(test_y_)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 24.39
Residual sum of squares (MSE): 1050.69
R2-score: 0.64
```

In [11]:

```python
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['CYLINDERS']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[29.98316342]]
Intercept:  [82.45502337]
```

In [12]:

```python
test_x = np.asanyarray(test[['CYLINDERS']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
#print(test_y)
test_y_ = regr.predict(test_x)
#print(test_y_)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 26.04
Residual sum of squares (MSE): 1112.77
R2-score: 0.63
```

Q 2. Develop a multiple linear regression (MLR) using more than one feature and obtain the minimum possible error.

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
from sklearn import linear_model
from sklearn.metrics import r2_score
```

In [2]:
```
df = pd.read_csv("FuelConsumption.csv")
# take a look at the dataset
df.head()
```

Out[2]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYI
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	

In [3]:
```
# summarize the data
df.describe()
```

Out[3]:

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY
count	1067.0	1067.000000	1067.000000	1067.000000	1067.000000
mean	2014.0	3.346298	5.794752	13.296532	9.474602
std	0.0	1.415895	1.797447	4.101253	2.794510
min	2014.0	1.000000	3.000000	4.600000	4.900000
25%	2014.0	2.000000	4.000000	10.250000	7.500000
50%	2014.0	3.400000	6.000000	12.600000	8.800000
75%	2014.0	4.300000	8.000000	15.550000	10.850000
max	2014.0	8.400000	12.000000	30.200000	20.500000

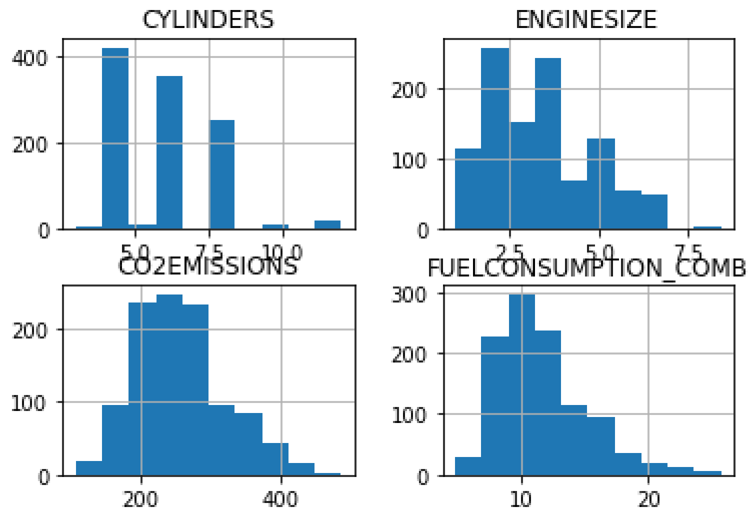
In [4]:
```
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)
```

Out[4]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

In [5]:
```
viz = cdf[['CYLINDERS','ENGINESIZE','CO2EMISSIONS','FUELCONSUMPTION_COMB']]
viz.hist()
plt.show()
```



In [6]:
```
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
print(train)
print(test)
```

```
      ENGINESIZE  CYLINDERS  FUELCONSUMPTION_COMB  CO2EMISSIONS
0            2.0          4                   8.5           196
1            2.4          4                   9.6           221
2            1.5          4                   5.9           136
3            3.5          6                  11.1           255
4            3.5          6                  10.6           244
...          ...        ...                   ...           ...
1060         3.0          6                  11.5           264
1061         3.2          6                  11.2           258
1062         3.0          6                  11.8           271
1064         3.0          6                  11.8           271
1065         3.2          6                  11.3           260

[850 rows x 4 columns]
      ENGINESIZE  CYLINDERS  FUELCONSUMPTION_COMB  CO2EMISSIONS
5            3.5          6                  10.0           230
8            3.7          6                  11.6           267
12           5.9         12                  15.6           359
13           5.9         12                  15.6           359
40           2.0          4                   9.2           212
...          ...        ...                   ...           ...
1032         2.0          4                   7.2           194
1042         1.4          4                   5.4           124
1048         2.0          4                   7.1           192
1063         3.2          6                  11.5           264
```

```
1066         3.2          6                  12.8          294

[217 rows x 4 columns]
```

In [7]:
```
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE', 'FUELCONSUMPTION_COMB']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[19.09277508 10.05170185]]
Intercept:  [76.17520539]
```

In [8]:
```
test_x = np.asanyarray(test[['ENGINESIZE', 'FUELCONSUMPTION_COMB']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 17.85
Residual sum of squares (MSE): 623.88
R2-score: 0.83
```

In [9]:
```
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE', 'CYLINDERS']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[27.2783142  10.24073354]]
Intercept:  [105.60906455]
```

In [10]:
```
test_x = np.asanyarray(test[['ENGINESIZE', 'CYLINDERS']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 23.38
Residual sum of squares (MSE): 930.63
R2-score: 0.73
```

In [11]:
```
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['CYLINDERS','FUELCONSUMPTION_COMB']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[14.01904391 10.81328844]]
Intercept:  [50.06131774]
```

In [12]:
```
test_x = np.asanyarray(test[['CYLINDERS','FUELCONSUMPTION_COMB']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 18.74
Residual sum of squares (MSE): 659.46
R2-score: 0.83
```

In [13]:

```
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[9.77393562 8.20703436 9.87735185]]
Intercept:  [61.84041662]
```

In [14]:

```
test_x = np.asanyarray(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 17.97
Residual sum of squares (MSE): 616.56
R2-score: 0.84
```

In [ ]:

```

```

Q 3. Use MLR to estimate the Mileage per gallon (MPG) using Auto-MPG dataset.

In [17]:
```
import pandas as pd
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn import linear_model
```

In [18]:
```
df = pd.read_csv("auto-mpg.csv")
```

In [19]:
```
df.head()
```

Out[19]:

	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

In [20]:
```
column_name = 'car name'
df = df.drop(column_name, axis=1)
```

In [21]:
```
df.head()
```

Out[21]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
0	18.0	8	307.0	130	3504	12.0	70	1
1	15.0	8	350.0	165	3693	11.5	70	1
2	18.0	8	318.0	150	3436	11.0	70	1
3	16.0	8	304.0	150	3433	12.0	70	1
4	17.0	8	302.0	140	3449	10.5	70	1

In [22]:
```
cdf = df[['cylinders','displacement','horsepower','weight','acceleration',
          'model year','origin','mpg']]
```

In [23]:
```
cdf.head()
```

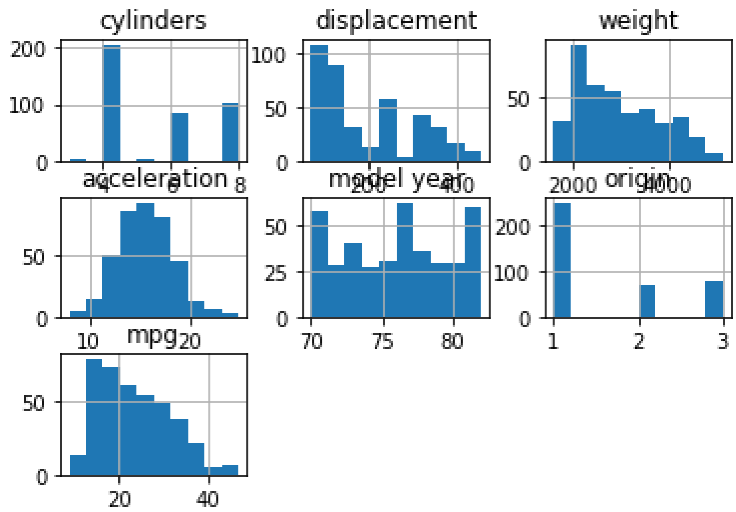
Out[23]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin	mpg
0	8	307.0	130	3504	12.0	70	1	18.0
1	8	350.0	165	3693	11.5	70	1	15.0
2	8	318.0	150	3436	11.0	70	1	18.0
3	8	304.0	150	3433	12.0	70	1	16.0

	cylinders	displacement	horsepower	weight	acceleration	model year	origin	mpg
4	8	302.0	140	3449	10.5	70	1	17.0

In [24]:

```
viz = cdf[['cylinders','displacement','horsepower','weight','acceleration',
            'model year','origin','mpg']]
viz.hist()
plt.show()
```



In [25]:

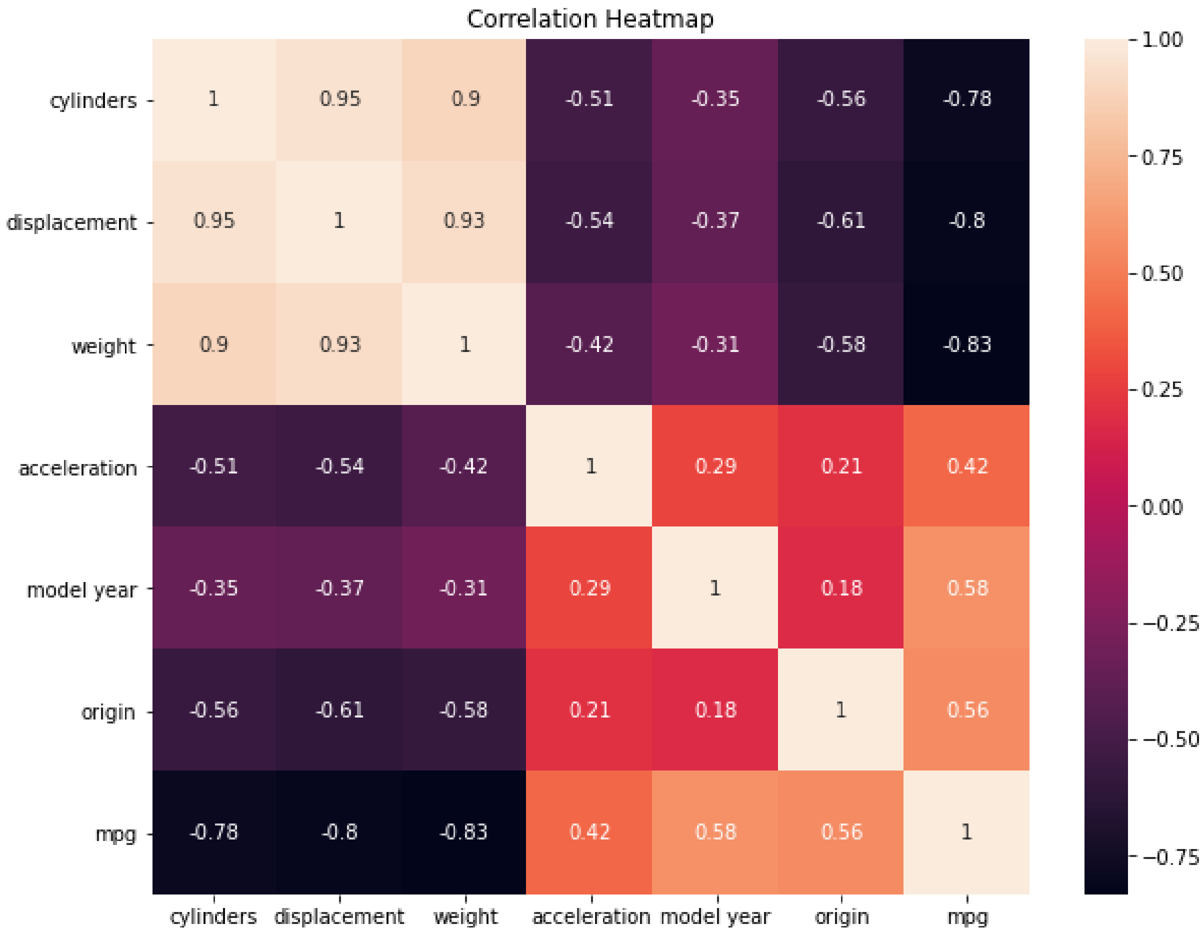
```
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
print(train)
print(test)
```

```
     cylinders  displacement  horsepower  weight  acceleration  model year  \
0            8         307.0         130    3504          12.0          70
1            8         350.0         165    3693          11.5          70
2            8         318.0         150    3436          11.0          70
3            8         304.0         150    3433          12.0          70
4            8         302.0         140    3449          10.5          70
..         ...           ...         ...     ...           ...         ...
393          4         140.0          86    2790          15.6          82
394          4          97.0          52    2130          24.6          82
395          4         135.0          84    2295          11.6          82
396          4         120.0          79    2625          18.6          82
397          4         119.0          82    2720          19.4          82

     origin   mpg
0         1  18.0
1         1  15.0
2         1  18.0
3         1  16.0
4         1  17.0
..      ...   ...
393       1  27.0
394       2  44.0
395       1  32.0
396       1  28.0
397       1  31.0

[326 rows x 8 columns]
     cylinders  displacement  horsepower  weight  acceleration  model year  \
10           8         383.0         170    3563          10.0          70
12           8         400.0         150    3761           9.5          70
20           4         110.0          87    2672          17.5          70
22           4         104.0          95    2375          17.5          70
29           4          97.0          88    2130          14.5          71
..         ...           ...         ...     ...           ...         ...
```

```
381          4     107.0     75   2205     14.5    82
382          4     108.0     70   2245     16.9    82
384          4      91.0     67   1965     15.7    82
386          6     181.0    110   2945     16.4    82
390          4     144.0     96   2665     13.9    82

     origin   mpg
10        1  15.0
12        1  15.0
20        2  25.0
22        2  25.0
29        3  27.0
..      ...   ...
381       3  36.0
382       3  34.0
384       3  32.0
386       1  25.0
390       3  32.0

[72 rows x 8 columns]
```

In [26]:

```python
# Calculate the correlation matrix
import seaborn as sns
correlation_matrix = cdf.corr()

# Create a heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



Simple linear regression

In [27]:

```python
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['cylinders']])
train_y = np.asanyarray(train[['mpg']])
```

```
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[-3.56892871]]
Intercept:  [43.12207679]
```

In [28]:
```
test_x = np.asanyarray(test[['cylinders']])
test_y = np.asanyarray(test[['mpg']])
#print(test_y)
test_y_ = regr.predict(test_x)
#print(test_y_)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 3.15
Residual sum of squares (MSE): 14.93
R2-score: 0.60
```

In [29]:
```
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['displacement']])
train_y = np.asanyarray(train[['mpg']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[-0.06099215]]
Intercept:  [35.41953102]
```

In [30]:
```
test_x = np.asanyarray(test[['displacement']])
test_y = np.asanyarray(test[['mpg']])
#print(test_y)
test_y_ = regr.predict(test_x)
#print(test_y_)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 3.27
Residual sum of squares (MSE): 14.90
R2-score: 0.65
```

In [31]:
```
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['weight']])
train_y = np.asanyarray(train[['mpg']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[-0.00770563]]
Intercept:  [46.51894909]
```

In [32]:
```
test_x = np.asanyarray(test[['weight']])
test_y = np.asanyarray(test[['mpg']])
#print(test_y)
test_y_ = regr.predict(test_x)
#print(test_y_)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 2.79
Residual sum of squares (MSE): 12.12
```

```
R2-score: 0.71
```

Multiple liniear regression

```
In [33]:
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['cylinders', 'displacement','weight']])
train_y = np.asanyarray(train[['mpg']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[-0.17889553 -0.01419798 -0.00577035]]
Intercept:  [44.48715502]
```

```
In [34]:
test_x = np.asanyarray(test[['cylinders', 'displacement','weight']])
test_y = np.asanyarray(test[['mpg']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 2.81
Residual sum of squares (MSE): 11.80
R2-score: 0.72
```

```
In [35]:
from sklearn import linear_model
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['cylinders', 'displacement']])
train_y = np.asanyarray(train[['mpg']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[-0.33114017 -0.05586234]]
Intercept:  [36.23523441]
```

```
In [36]:
test_x = np.asanyarray(test[['cylinders', 'displacement']])
test_y = np.asanyarray(test[['mpg']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 3.25
Residual sum of squares (MSE): 14.69
R2-score: 0.65
```

```
In [37]:
from sklearn import linear_model
regr_m = linear_model.LinearRegression()
train_x = np.asanyarray(train[['displacement', 'weight']])
train_y = np.asanyarray(train[['mpg']])
regr_m.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr_m.coef_)
print ('Intercept: ',regr_m.intercept_)
```

```
Coefficients:  [[-0.0168571  -0.00578506]]
Intercept:  [44.06846878]
```

```
In [38]:
test_x = np.asanyarray(test[['displacement', 'weight']])
test_y = np.asanyarray(test[['mpg']])
test_y_ = regr_m.predict(test_x)
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y_ , test_y) )
```

```
Mean absolute error: 2.81
Residual sum of squares (MSE): 11.84
R2-score: 0.72
```

In [ ]:

Comparison tables for FuelConsumption.csv:

Linear regression to predict CO2EMISSIONS:

Attributes	MAE	MSE	R2-Score
ENGINESIZE	24.39	1050.69	0.64
CYLINDERS	26.04	1112.77	0.63

Multiple linear regression (MLR) to predict CO2EMISSIONS:

Attributes	MAE	MSE	R2-Score
ENGINESIZE, FUELCONSUMPTION_COMB	17.85	623.88	0.83
ENGINESIZE, CYLINDERS	23.38	930.63	0.73
CYLINDERS, FUELCONSUMPTION_COMB	18.74	659.46	0.83
ENGINESIZE, CYLINDERS, FUELCONSUMPTION_COMB	17.97	616.56	0.84

Comparison tables for auto-mpg.csv:

Linear regression to predict mpg:

Attributes	MAE	MSE	R2-Score
cylinders	3.15	14.93	0.60
displacement	3.27	14.90	0.65
weight	2.79	12.12	0.71

Multiple linear regression (MLR) to predict mpg:

Attributes	MAE	MSE	R2-Score
cylinders, displacement, weight	2.81	11.80	0.72
cylinders, displacement	3.25	14.69	0.65
displacement, weight	2.81	11.84	0.72