Mula Anvesh Reddy Simple and Multi Linear Regressions Assingment-4

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
#Reading the dataset
dataset=pd.read_csv('student_scores.csv')
```

In [4]:
```
dataset
```

Out[4]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [5]:
```
dataset.head()
```

Out[5]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [6]:
```
dataset.isnull().any()
```

Out[6]:
```
Hours     False
Scores    False
dtype: bool
```

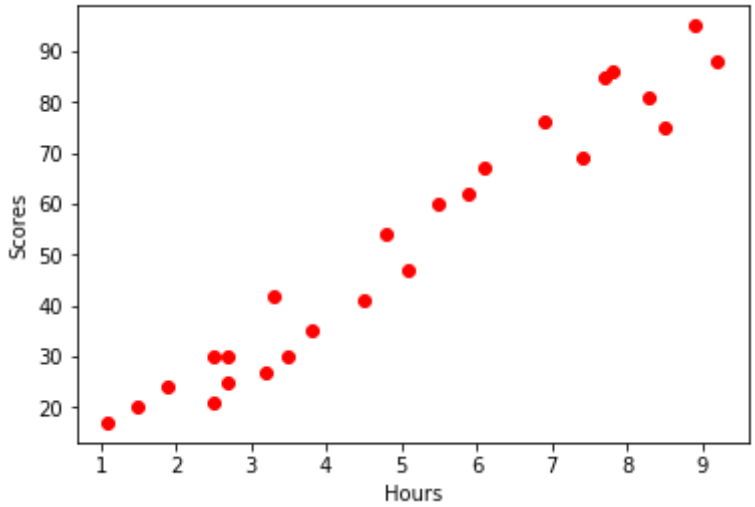
In [7]:
```
dataset.describe()
```

Out[7]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [9]:
```
plt.scatter(dataset['Hours'],dataset['Scores'],color='red')
plt.xlabel('Hours')
plt.ylabel('Scores')
```

Out[9]: Text(0, 0.5, 'Scores')



In [10]:
```
#independent and dependent variable
x=dataset.iloc[:,0:1]
y=dataset.iloc[:,1:]
```

In [11]:
```
x.head()
```

Out[11]:

	Hours
0	2.5
1	5.1
2	3.2
3	8.5
4	3.5

In [12]:
```
x=dataset.iloc[:,0:1].values
y=dataset.iloc[:,1:].values
```

In [14]:
```
x.ndim
```

Out[14]: 2

In [15]:
```
y.ndim
```

Out[15]: 2

In [16]:
```
x.shape
```

Out[16]: (25, 1)

In [17]:
```
y.shape
```

Out[17]: (25, 1)

In [18]:
```
#splitting the data into test and training set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [19]:
```
x_train.shape
```

Out[19]: (20, 1)

In [20]:
```
x_test.shape
```

Out[20]: (5, 1)

In [21]:
```
y_train.shape
```

Out[21]: (20, 1)

In [22]:
```
y_test.shape
```

Out[22]:
```
(5, 1)
```

In [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

In [24]:
```
lr.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [25]:
```
y_pred=lr.predict(x_test)
```

In [26]:
```
y_pred
```

Out[26]:
```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

In [27]:
```
y_test
```

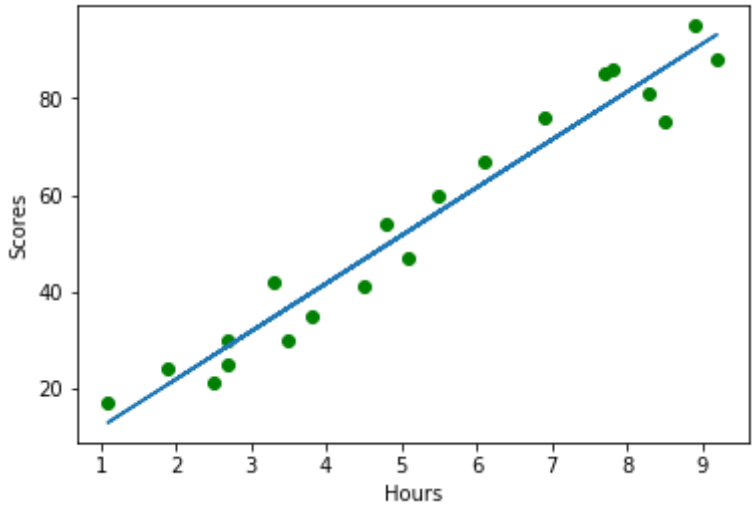
Out[27]:
```
array([[20],
       [27],
       [69],
       [30],
       [62]], dtype=int64)
```

In [28]:
```
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy
```

Out[28]:
```
0.9454906892105356
```

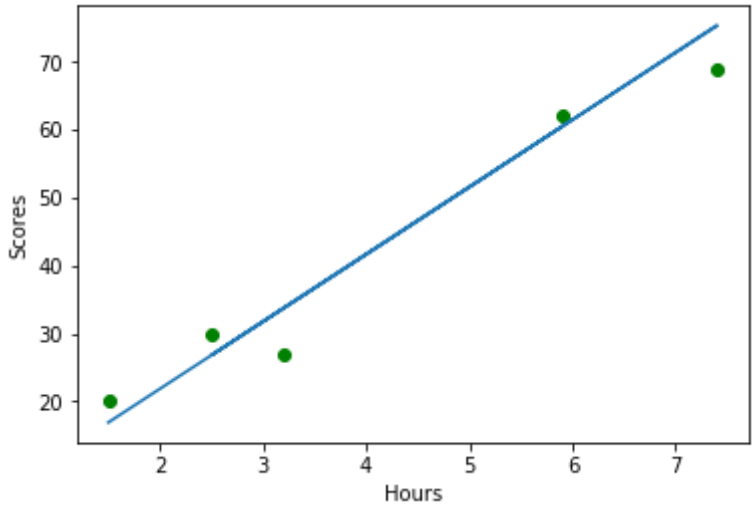
In [29]:
```
plt.scatter(x_train,y_train,color='green')
plt.plot(x_train,lr.predict(x_train))
plt.xlabel('Hours')
plt.ylabel('Scores')
```

Out[29]: Text(0, 0.5, 'Scores')



In [30]:
```
plt.scatter(x_test,y_test,color='green')
plt.plot(x_test,y_pred)
plt.xlabel('Hours')
plt.ylabel('Scores')
```

Out[30]: Text(0, 0.5, 'Scores')



In [31]:
```
lr.intercept_
```

Out[31]: array([2.01816004])

In [32]:
```
lr.coef_
```

Out[32]: array([[9.91065648]])

Multi Linear Regression with petrol_consumption.csv

In [33]:
```
dataset = pd.read_csv('petrol_consumption.csv')
dataset
```

Out[33]:

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.00	3571	1976	0.525	541
1	9.00	4092	1250	0.572	524
2	9.00	3865	1586	0.580	561
3	7.50	4870	2351	0.529	414
4	8.00	4399	431	0.544	410
5	10.00	5342	1333	0.571	457
6	8.00	5319	11868	0.451	344
7	8.00	5126	2138	0.553	467
8	8.00	4447	8577	0.529	464
9	7.00	4512	8507	0.552	498
10	8.00	4391	5939	0.530	580
11	7.50	5126	14186	0.525	471
12	7.00	4817	6930	0.574	525
13	7.00	4207	6580	0.545	508
14	7.00	4332	8159	0.608	566
15	7.00	4318	10340	0.586	635
16	7.00	4206	8508	0.572	603
17	7.00	3718	4725	0.540	714
18	7.00	4716	5915	0.724	865
19	8.50	4341	6010	0.677	640
20	7.00	4593	7834	0.663	649
21	8.00	4983	602	0.602	540
22	9.00	4897	2449	0.511	464
23	9.00	4258	4686	0.517	547
24	8.50	4574	2619	0.551	460

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
25	9.00	3721	4746	0.544	566
26	8.00	3448	5399	0.548	577
27	7.50	3846	9061	0.579	631
28	8.00	4188	5975	0.563	574
29	9.00	3601	4650	0.493	534
30	7.00	3640	6905	0.518	571
31	7.00	3333	6594	0.513	554
32	8.00	3063	6524	0.578	577
33	7.50	3357	4121	0.547	628
34	8.00	3528	3495	0.487	487
35	6.58	3802	7834	0.629	644
36	5.00	4045	17782	0.566	640
37	7.00	3897	6385	0.586	704
38	8.50	3635	3274	0.663	648
39	7.00	4345	3905	0.672	968
40	7.00	4449	4639	0.626	587
41	7.00	3656	3985	0.563	699
42	7.00	4300	3635	0.603	632
43	7.00	3745	2611	0.508	591
44	6.00	5215	2302	0.672	782
45	9.00	4476	3942	0.571	510
46	7.00	4296	4083	0.623	610
47	7.00	5002	9794	0.593	524

In [34]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
Petrol_tax                      48 non-null float64
Average_income                  48 non-null int64
Paved_Highways                  48 non-null int64
Population_Driver_licence(%)    48 non-null float64
Petrol_Consumption              48 non-null int64
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
```

In [35]:
```
dataset.isnull().any()
```

Out[35]:
```
Petrol_tax                      False
Average_income                  False
Paved_Highways                  False
Population_Driver_licence(%)    False
Petrol_Consumption              False
dtype: bool
```

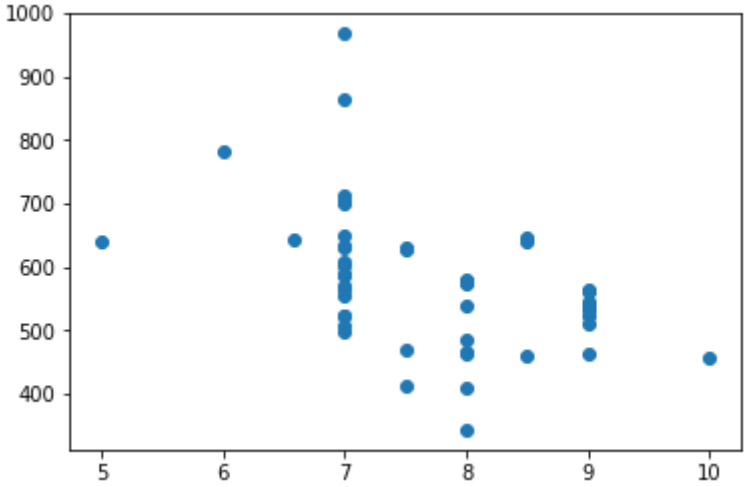
In [36]:
```
dataset.describe()
```

Out[36]:

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
count	48.000000	48.000000	48.000000	48.000000	48.000000
mean	7.668333	4241.833333	5565.416667	0.570333	576.770833
std	0.950770	573.623768	3491.507166	0.055470	111.885816
min	5.000000	3063.000000	431.000000	0.451000	344.000000
25%	7.000000	3739.000000	3110.250000	0.529750	509.500000
50%	7.500000	4298.000000	4735.500000	0.564500	568.500000
75%	8.125000	4578.750000	7156.000000	0.595250	632.750000
max	10.000000	5342.000000	17782.000000	0.724000	968.000000

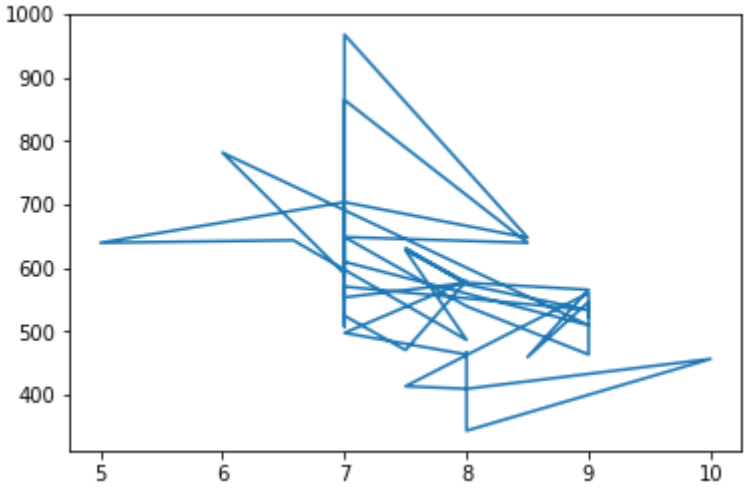
In [37]:
```
plt.scatter(dataset['Petrol_tax'],dataset['Petrol_Consumption'])
```

Out[37]:
```
<matplotlib.collections.PathCollection at 0x1fc1f77af60>
```



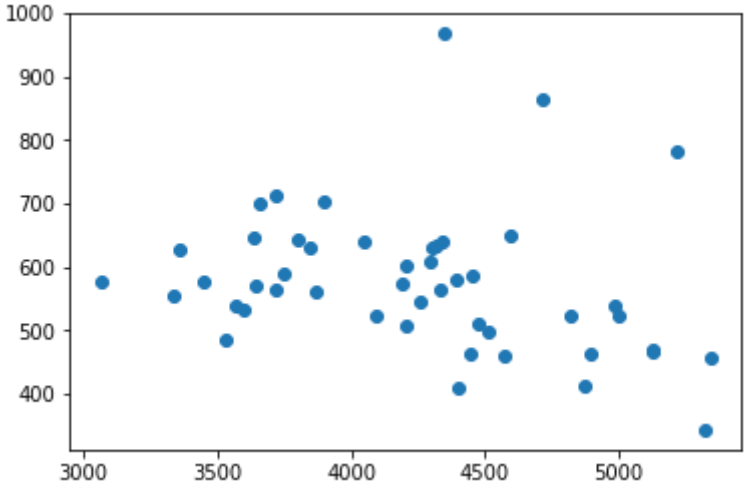
In [38]:
```
plt.plot(dataset['Petrol_tax'],dataset['Petrol_Consumption'])
```

Out[38]:
```
[<matplotlib.lines.Line2D at 0x1fc1fb5ac18>]
```



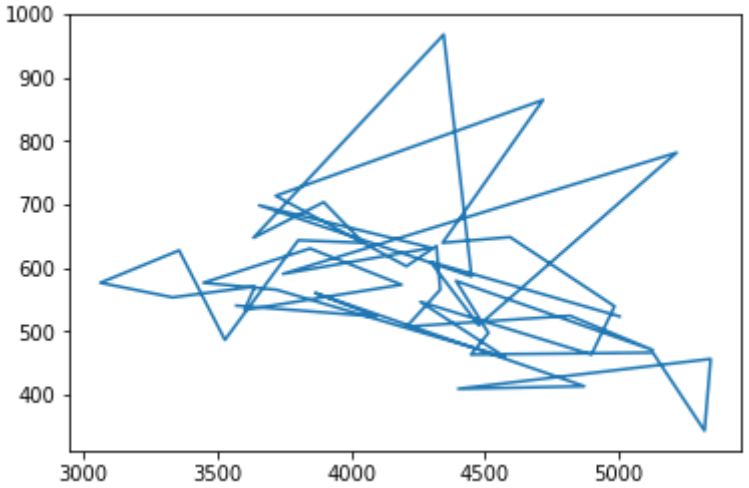
In [39]:
```
plt.scatter(dataset['Average_income'],dataset['Petrol_Consumption'])
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x1fc1fbb8ba8>
```



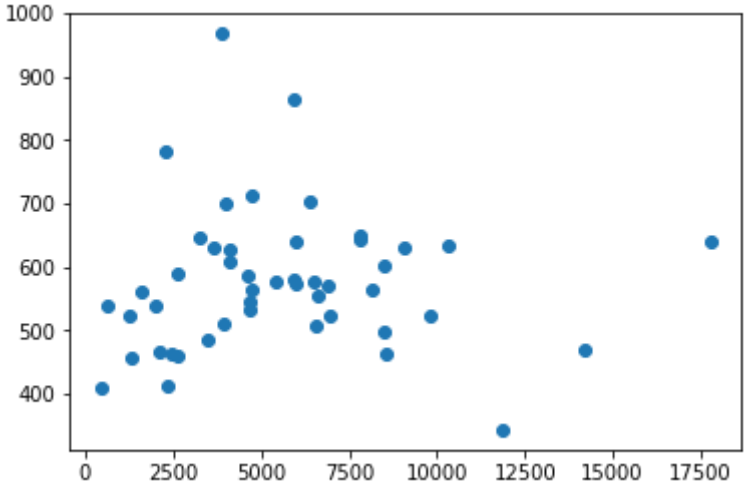
In [40]:
```
plt.plot(dataset['Average_income'],dataset['Petrol_Consumption'])
```

Out[40]:
```
[<matplotlib.lines.Line2D at 0x1fc1fc173c8>]
```



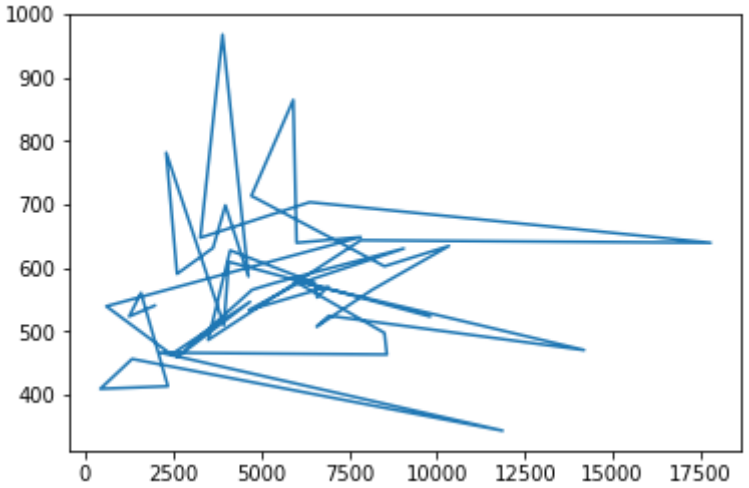
In [41]:
```
plt.scatter(dataset['Paved_Highways'],dataset['Petrol_Consumption'])
```

Out[41]:
```
<matplotlib.collections.PathCollection at 0x1fc1fc6afd0>
```



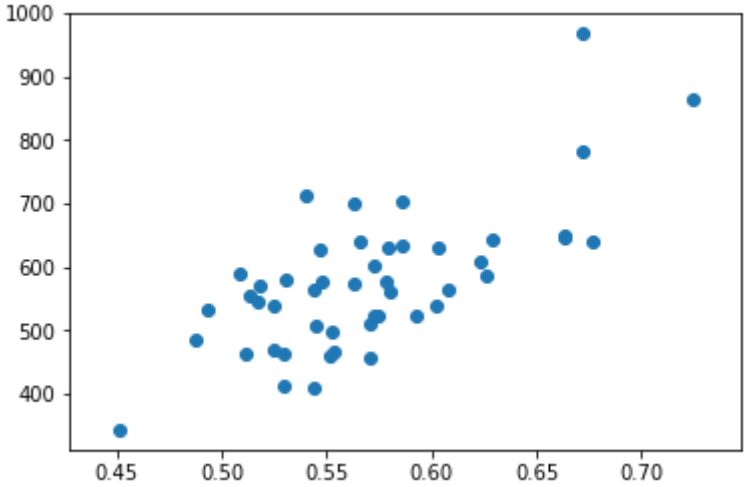
In [42]:
```
plt.plot(dataset['Paved_Highways'],dataset['Petrol_Consumption'])
```

Out[42]:
```
[<matplotlib.lines.Line2D at 0x1fc1fd615c0>]
```



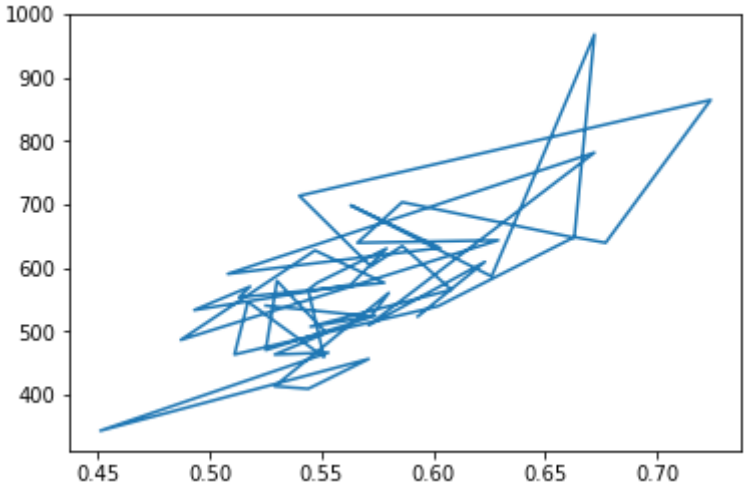
In [43]:
```
plt.scatter(dataset['Population_Driver_licence(%)'],dataset['Petrol_Consumption'])
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x1fc1fdc0f28>
```



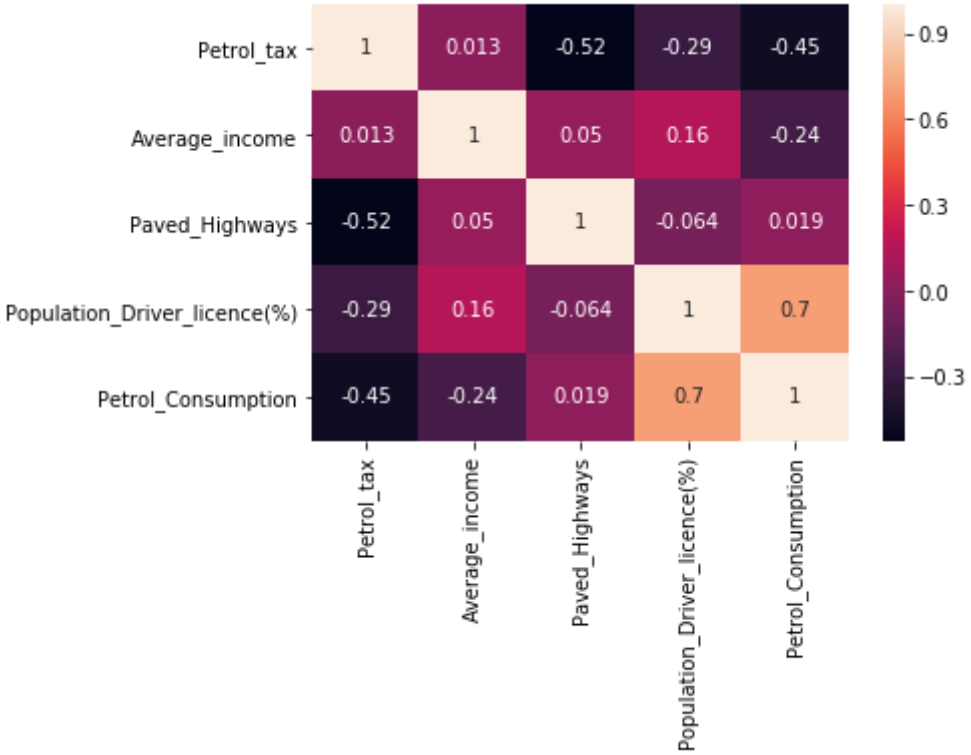
In [44]:
```
plt.plot(dataset['Population_Driver_licence(%)'],dataset['Petrol_Consumption'])
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x1fc1fe1ec88>]
```



In [45]:
```
import seaborn as sns
sns.heatmap(dataset.corr(),annot=True)
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc2019d7b8>



In [46]:
```
x = dataset.iloc[:,0:4].values
y = dataset.iloc[:,4:].values
```

In [47]:
```
x
```

Out[47]:
```
array([[9.0000e+00, 3.5710e+03, 1.9760e+03, 5.2500e-01],
       [9.0000e+00, 4.0920e+03, 1.2500e+03, 5.7200e-01],
       [9.0000e+00, 3.8650e+03, 1.5860e+03, 5.8000e-01],
       [7.5000e+00, 4.8700e+03, 2.3510e+03, 5.2900e-01],
       [8.0000e+00, 4.3990e+03, 4.3100e+02, 5.4400e-01],
       [1.0000e+01, 5.3420e+03, 1.3330e+03, 5.7100e-01],
       [8.0000e+00, 5.3190e+03, 1.1868e+04, 4.5100e-01],
       [8.0000e+00, 5.1260e+03, 2.1380e+03, 5.5300e-01],
       [8.0000e+00, 4.4470e+03, 8.5770e+03, 5.2900e-01],
       [7.0000e+00, 4.5120e+03, 8.5070e+03, 5.5200e-01],
       [8.0000e+00, 4.3910e+03, 5.9390e+03, 5.3000e-01],
       [7.5000e+00, 5.1260e+03, 1.4186e+04, 5.2500e-01],
       [7.0000e+00, 4.8170e+03, 6.9300e+03, 5.7400e-01],
       [7.0000e+00, 4.2070e+03, 6.5800e+03, 5.4500e-01],
       [7.0000e+00, 4.3320e+03, 8.1590e+03, 6.0800e-01],
       [7.0000e+00, 4.3180e+03, 1.0340e+04, 5.8600e-01],
       [7.0000e+00, 4.2060e+03, 8.5080e+03, 5.7200e-01],
       [7.0000e+00, 3.7180e+03, 4.7250e+03, 5.4000e-01],
       [7.0000e+00, 4.7160e+03, 5.9150e+03, 7.2400e-01],
       [8.5000e+00, 4.3410e+03, 6.0100e+03, 6.7700e-01],
       [7.0000e+00, 4.5930e+03, 7.8340e+03, 6.6300e-01],
       [8.0000e+00, 4.9830e+03, 6.0200e+02, 6.0200e-01],
       [9.0000e+00, 4.8970e+03, 2.4490e+03, 5.1100e-01],
       [9.0000e+00, 4.2580e+03, 4.6860e+03, 5.1700e-01],
       [8.5000e+00, 4.5740e+03, 2.6190e+03, 5.5100e-01],
       [9.0000e+00, 3.7210e+03, 4.7460e+03, 5.4400e-01],
       [8.0000e+00, 3.4480e+03, 5.3990e+03, 5.4800e-01],
       [7.5000e+00, 3.8460e+03, 9.0610e+03, 5.7900e-01],
       [8.0000e+00, 4.1880e+03, 5.9750e+03, 5.6300e-01],
       [9.0000e+00, 3.6010e+03, 4.6500e+03, 4.9300e-01],
       [7.0000e+00, 3.6400e+03, 6.9050e+03, 5.1800e-01],
       [7.0000e+00, 3.3330e+03, 6.5940e+03, 5.1300e-01],
       [8.0000e+00, 3.0630e+03, 6.5240e+03, 5.7800e-01],
       [7.5000e+00, 3.3570e+03, 4.1210e+03, 5.4700e-01],
       [8.0000e+00, 3.5280e+03, 3.4950e+03, 4.8700e-01],
       [6.5800e+00, 3.8020e+03, 7.8340e+03, 6.2900e-01],
       [5.0000e+00, 4.0450e+03, 1.7782e+04, 5.6600e-01],
       [7.0000e+00, 3.8970e+03, 6.3850e+03, 5.8600e-01],
       [8.5000e+00, 3.6350e+03, 3.2740e+03, 6.6300e-01],
       [7.0000e+00, 4.3450e+03, 3.9050e+03, 6.7200e-01],
       [7.0000e+00, 4.4490e+03, 4.6390e+03, 6.2600e-01],
```

```
[7.0000e+00, 3.6560e+03, 3.9850e+03, 5.6300e-01],
[7.0000e+00, 4.3000e+03, 3.6350e+03, 6.0300e-01],
[7.0000e+00, 3.7450e+03, 2.6110e+03, 5.0800e-01],
[6.0000e+00, 5.2150e+03, 2.3020e+03, 6.7200e-01],
[9.0000e+00, 4.4760e+03, 3.9420e+03, 5.7100e-01],
[7.0000e+00, 4.2960e+03, 4.0830e+03, 6.2300e-01],
[7.0000e+00, 5.0020e+03, 9.7940e+03, 5.9300e-01]])
```

In [48]:
```
y
```

Out[48]:
```
array([[541],
       [524],
       [561],
       [414],
       [410],
       [457],
       [344],
       [467],
       [464],
       [498],
       [580],
       [471],
       [525],
       [508],
       [566],
       [635],
       [603],
       [714],
       [865],
       [640],
       [649],
       [540],
       [464],
       [547],
       [460],
       [566],
       [577],
       [631],
       [574],
       [534],
       [571],
       [554],
       [577],
       [628],
       [487],
       [644],
       [640],
       [704],
       [648],
       [968],
       [587],
```

```
       [699],
       [632],
       [591],
       [782],
       [510],
       [610],
       [524]], dtype=int64)
```

In [70]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 ,random_state=0)
```

In [50]:
```
x_train.shape
```

Out[50]:
```
(38, 4)
```

In [51]:
```
y_train.shape
```

Out[51]:
```
(38, 1)
```

In [52]:
```
mlr = LinearRegression()
mlr.fit(x_train , y_train)
y_pred = mlr.predict(x_test)
y_pred
```

Out[52]:
```
array([[469.39198872],
       [545.64546431],
       [589.66839402],
       [569.7304133 ],
       [649.77480909],
       [646.63116356],
       [511.60814841],
       [672.47517717],
       [502.07478157],
       [501.2707342 ]])
```

In [53]:
```
y_test
```

Out[53]:
```
array([[534],
       [410],
       [577],
       [571],
       [577],
       [704],
       [487],
       [587],
       [467],
       [580]], dtype=int64)
```

In [71]:
```
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy
```

Out[71]:
```
0.20361932410121653
```

In [75]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 ,random_state=10)
mlr = LinearRegression()
mlr.fit(x_train , y_train)
y_pred = mlr.predict(x_test)
accuracy = r2_score(y_test , y_pred)
accuracy
```

Out[75]:
```
0.5270817614402223
```

In [111]:
```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size = 0.2 ,random_state=27)
mlr.fit(x_train , y_train)
y_pred = mlr.predict(x_test)
accuracy = r2_score(y_test , y_pred)
accuracy
```

Out[111]:
```
0.8679114383537712
```

In [ ]:
```

```