In [1]:
```
import numpy as np
```

In [2]:
```
from sklearn.linear_model import LinearRegression
```

In [6]:
```
CIE = np.array([25,50,60,44,60,]).reshape(-1,1)
SEE = np.array([22,39,70,90,88])
```

In [7]:
```
model = LinearRegression()
model.fit(CIE,SEE)
print("intercept:",model.intercept_)
print("coeffecient:",model.coef_)
```

```
intercept: -5.021749521988525
coeffecient: [1.39794455]
```

In [1]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm

data = {'cie':[20,30,40,60,70],
       'see':[49,53,89,60,70]}
```

In [2]:
```
df = pd.DataFrame(data)
```

In [14]:
```
df
```

Out[14]:

	cie	see
0	20	49
1	30	53
2	40	89
3	60	60

In [15]:
```
x = sm.add_constant(df['cie'])
```

In [23]:
```
model = sm.OLS(df['see'],x).fit()
```

In [24]:
```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    see   R-squared:                       0.117
Model:                            OLS   Adj. R-squared:                 -0.324
Method:                 Least Squares   F-statistic:                    0.2662
Date:                Sun, 05 Nov 2023   Prob (F-statistic):              0.657
Time:                        22:39:48   Log-Likelihood:                -16.430
No. Observations:                   4   AIC:                             36.86
Df Residuals:                       2   BIC:                             35.63
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         49.1429     28.350      1.733      0.225     -72.837     171.123
cie            0.3629      0.703      0.516      0.657      -2.663       3.389
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   2.730
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.912
Skew:                         1.117   Prob(JB):                        0.634
Kurtosis:                     2.307   Cond. No.                         110.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
```

```
C:\Users\HARISH BJ\anaconda3\Lib\site-packages\statsmodels\stats\stattools.py:74: Val
ueWarning: omni_normtest is not valid with less than 8 observations; 4 samples were g
iven.
  warn("omni_normtest is not valid with less than 8 observations; %i "
```

In [1]:
```
import pandas as pd
data = pd.read_csv("C:/Users/HARISH BJ/Downloads/boston.csv")
```

In [2]:
```
data
```

Out[2]:

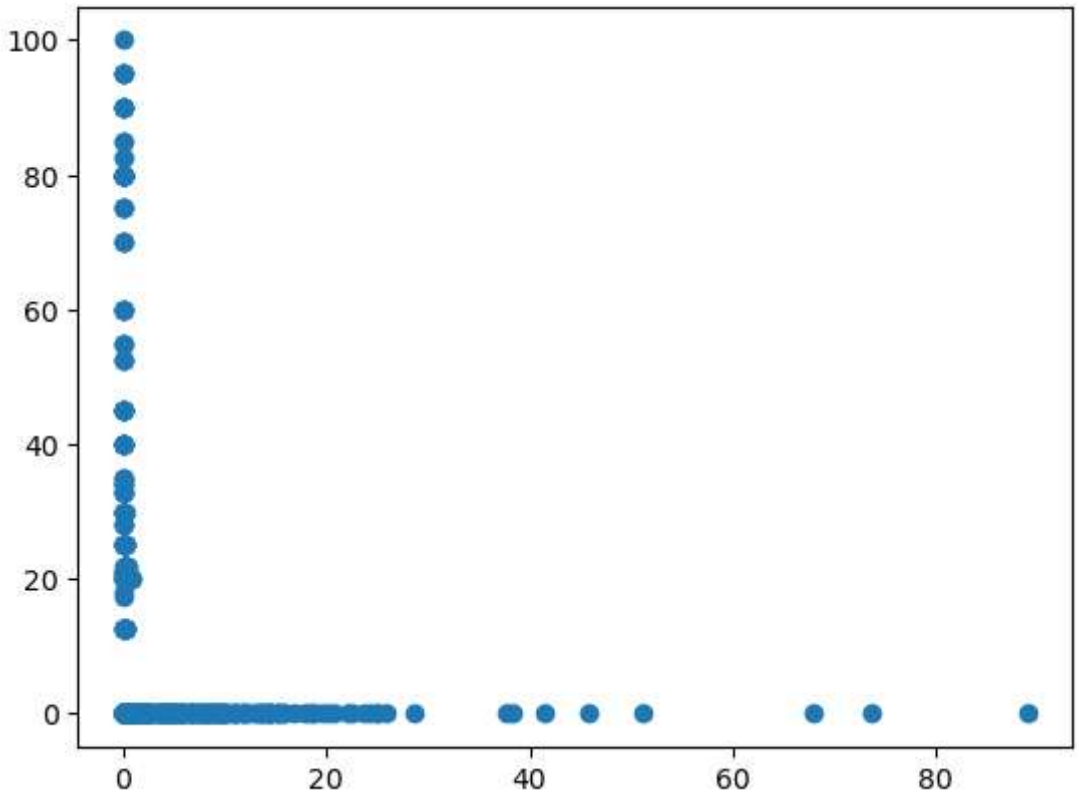
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	M
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	

506 rows × 14 columns

In [4]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

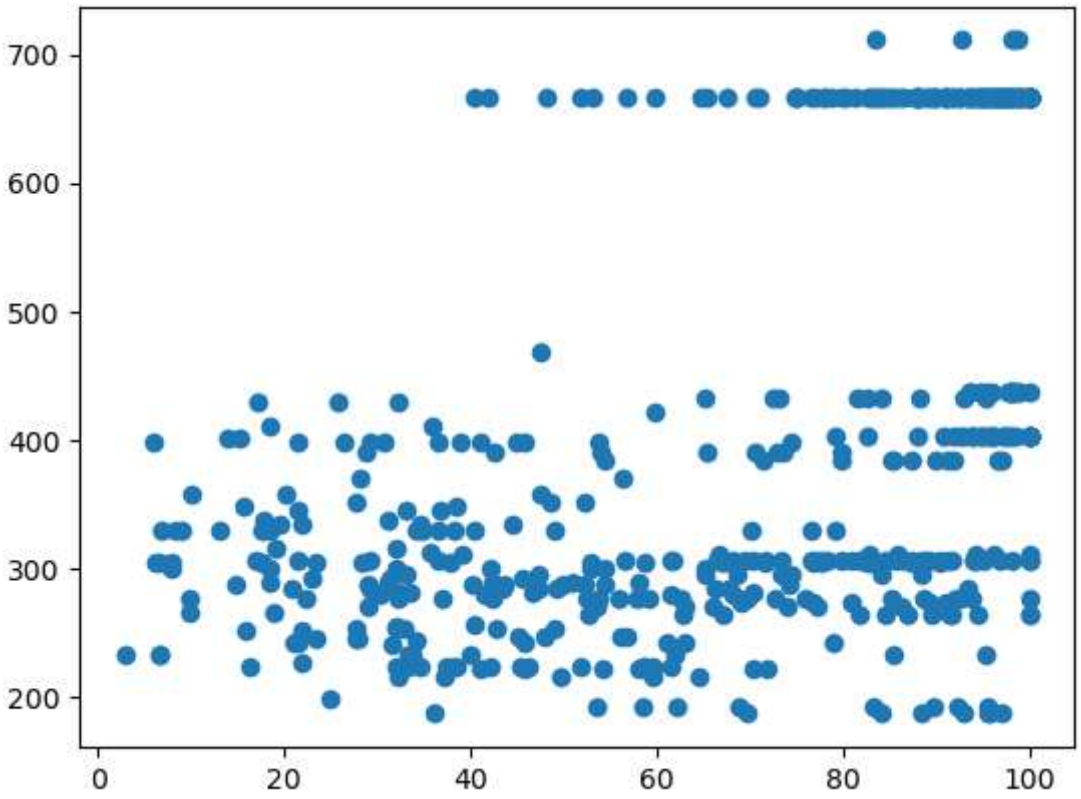
In [5]:
```
plt.scatter(data['CRIM'],data['ZN'])
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x25ec958f790>
```



In [6]:
```
plt.scatter(data['AGE'],data['TAX'])
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x25ecb538490>
```



In [34]:
```
x = data ['CRIM']
y = data ['TAX']
```

In [10]:
```
x
```

Out[10]:
```
0      0.00632
1      0.02731
2      0.02729
3      0.03237
4      0.06905
        ...
501    0.06263
502    0.04527
503    0.06076
504    0.10959
505    0.04741
Name: CRIM, Length: 506, dtype: float64
```

In [11]:
```
y
```

Out[11]:
```
0      296
1      242
2      242
3      222
4      222
      ...
501    273
502    273
503    273
504    273
505    273
Name: TAX, Length: 506, dtype: int64
```

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

In [16]:
```
x_train
```

Out[16]:
```
300     0.04417
491     0.10574
427    37.66190
359     4.26131
105     0.13262
         ...
37      0.08014
136     0.32264
325     0.19186
235     0.33045
202     0.02177
Name: CRIM, Length: 404, dtype: float64
```

In [17]:
```
x_test
```

Out[17]:
```
355    0.10659
294    0.08199
4      0.06905
330    0.04544
147    2.36862
        ...
218    0.11069
128    0.32543
310    2.63548
117    0.15098
257    0.61154
Name: CRIM, Length: 102, dtype: float64
```

In [18]:
```
y_train
```

Out[18]:
```
300    358
491    711
427    666
359    666
105    384
      ...
37     279
136    437
325    287
235    307
202    348
Name: TAX, Length: 404, dtype: int64
```

In [19]:
```
y_test
```

Out[19]:
```
355    334
294    289
4      222
330    430
147    403
      ...
218    276
128    437
310    304
117    432
257    264
Name: TAX, Length: 102, dtype: int64
```

In [20]:
```
from sklearn.linear_model import LinearRegression
```

In [21]:
```
clf = LinearRegression()
```

In [30]:
```
clf.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[30], line 1
----> 1 clf.fit(x_train,y_train)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:1151, in _fit_context.<locals>.dec
orator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\linear_model\_base.py:678, in LinearRegres
sion.fit(self, X, y, sample_weight)
    674 n_jobs_ = self.n_jobs
    676 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
--> 678 X, y = self._validate_data(
    679     X, y, accept_sparse=accept_sparse, y_numeric=True, multi_output=True
    680 )
    682 has_sw = sample_weight is not None
    683 if has_sw:

File ~\anaconda3\Lib\site-packages\sklearn\base.py:621, in BaseEstimator._validate_da
ta(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_params)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1147, in check_X_y(X,
y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2
d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estima
tor)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but the target y is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=est
imator)
   1165 check_consistent_length(X, y)
```

```
File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:940, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    938     # If input is 1D raise error
    939     if array.ndim == 1:
--> 940         raise ValueError(
    941             "Expected 2D array, got 1D array instead:\narray={}.\n"
    942             "Reshape your data either using array.reshape(-1, 1) if "
    943             "your data has a single feature or array.reshape(1, -1) "
    944             "if it contains a single sample.".format(array)
    945         )
    947 if dtype_numeric and hasattr(array.dtype, "kind") and array.dtype.kind in "USV":
    948     raise ValueError(
    949         "dtype='numeric' is not compatible with arrays of bytes/strings."
    950         "Convert your data to numeric values explicitly instead."
    951     )

ValueError: Expected 2D array, got 1D array instead:
array=[4.41700e-02 1.05740e-01 3.76619e+01 4.26131e+00 1.32620e-01 7.87500e-02
 2.37857e+00 1.26500e-01 7.01300e-02 1.75050e-01 2.00849e+01 3.50200e-02
 7.75223e+00 1.19294e+00 4.35710e-01 3.75780e-01 5.73116e+00 1.50380e-01
 1.61282e+00 1.23290e-01 4.07710e-01 3.51140e-01 3.35900e-02 1.06718e+01
 1.22690e-01 7.25800e-01 9.84900e-02 4.59000e-02 8.38700e-02 2.07162e+01
 2.72900e-02 2.53560e-01 1.77830e-01 5.66998e+00 8.66400e-02 6.64200e-02
 1.68118e+01 1.02330e+01 2.22120e-01 1.31170e-01 5.08300e-02 2.89550e-01
 5.66637e+00 9.51363e+00 8.82900e-02 1.05393e+00 6.80117e+00 1.73310e-01
 6.66400e-02 5.82115e+00 4.01100e-02 7.02200e-02 1.83377e+00 1.20482e+01
 2.81838e+00 4.92980e-01 1.11320e-01 3.57800e-02 5.29305e+00 2.99160e-01
 3.53700e-02 6.79208e+01 7.90410e-01 8.24400e-02 1.15040e-01 1.78667e+01
 3.67822e+00 4.29400e-02 1.28020e-01 1.29320e-01 1.69020e-01 5.20140e-01
 1.14320e-01 4.12380e-01 5.42500e-02 2.33099e+00 4.74100e-02 1.71710e-01
 7.35341e+01 1.81590e-01 3.87350e-01 9.32909e+00 2.83920e-01 2.63630e-01
 1.27570e-01 4.57461e+01 8.02710e-01 2.29270e-01 2.20511e+01 5.36000e-02
 4.02020e-01 9.74400e-02 4.75470e-01 1.41385e+00 5.78000e-02 9.23230e+00
 6.04700e-02 1.52880e+01 1.88110e+01 4.87141e+00 5.82401e+00 5.58107e+00
 4.93200e-02 2.17190e-01 5.70818e+00 1.67600e-01 1.22040e-01 4.03841e+00
 1.96091e+01 3.39830e-01 1.77800e-02 1.34284e+00 4.34879e+00 3.76800e-02
 1.22472e+01 6.23560e-01 2.73397e+00 4.09740e+00 7.40389e+00 1.35540e-01
 2.92400e+00 1.83370e-01 1.71420e-01 2.31390e+00 1.00000e-01 6.12900e-02
 7.85700e-01 1.20742e+00 6.32000e-03 3.41090e-01 1.55757e+01 1.35870e-01
 5.60200e-02 4.47910e-01 5.69175e+00 9.17800e-02 3.77498e+00 5.66000e-02
 8.52040e-01 6.58800e-02 2.21880e-01 2.98190e-01 5.56100e-02 5.37200e-02
 7.05042e+00 6.72400e-02 1.96570e-01 3.49400e-01 2.37934e+00 1.70040e-01
 1.07930e-01 5.20580e-01 1.40520e-01 3.42700e-02 1.39134e+01 1.35222e+01
 4.54192e+00 3.04900e-02 7.24400e-02 5.73500e-02 8.40540e-01 6.86000e-02
 6.15100e-02 2.53870e-01 8.79212e+00 9.39063e+00 1.40300e-01 1.96500e-02
 1.44550e-01 3.04100e-02 6.53876e+00 4.66883e+00 6.89900e-02 8.98296e+00
 6.26300e-02 5.05900e-02 1.22358e+00 1.41500e-01 5.49700e-02 1.04690e-01
 1.53800e-02 5.09017e+00 3.47428e+00 2.14918e+00 6.27390e-01 1.00080e-01
 2.59406e+01 1.78990e-01 1.28023e+01 6.29760e-01 3.29820e-01 9.59571e+00
 4.64689e+00 5.30200e-02 7.89600e-02 2.73100e-02 5.44114e+00 1.39140e-01
 3.30600e-02 5.75290e-01 6.21100e-02 3.68940e-01 1.30100e-02 2.89900e-02
 3.82140e-01 9.92485e+00 1.25790e-01 3.16360e+00 1.49320e-01 2.54300e-02
 6.63510e-01 2.05500e-02 1.35472e+00 4.83567e+00 9.51200e-02 2.49800e-01
 2.01019e+00 3.87100e-02 8.81250e-01 6.16200e-02 5.02300e-02 5.90050e-01
 1.43900e-02 9.76170e-01 8.20058e+00 7.61620e-01 3.58400e-02 5.26930e-01
 4.89822e+00 9.29900e-02 4.11300e-02 1.65660e+00 2.51990e-01 2.07460e-01
 1.80846e+01 2.41030e-01 7.72990e-01 1.64390e-01 4.55587e+00 3.54800e-02
 1.06120e-01 2.11610e-01 3.18270e-01 1.33598e+01 1.27346e+00 1.09590e-01
```

```
 8.87300e-02 9.82349e+00 2.24380e-01 5.64600e-02 5.11830e-01 6.96215e+00
 3.40060e-01 5.87205e+00 1.18123e+01 1.23247e+00 2.98500e-02 1.25179e+00
 2.45220e-01 9.26600e-02 1.15172e+00 1.40507e+01 1.13290e-01 1.71340e-01
 6.07600e-02 2.86558e+01 1.91330e-01 2.11240e-01 1.10270e-01 7.97800e-02
 2.68380e-01 5.57780e-01 1.39600e-01 1.00623e+01 6.61700e-02 3.44500e-02
 1.84982e+01 6.39312e+00 1.36781e+01 1.44760e-01 1.38100e-02 4.33700e-02
 4.30100e-02 1.98020e-01 1.46336e+00 4.75237e+00 1.50860e-01 3.55100e-02
 1.51902e+00 7.16500e-02 3.15000e-02 3.93200e-02 1.08342e+01 1.30580e-01
 1.74460e-01 1.42362e+01 3.70500e-02 5.37000e-01 3.83684e+00 3.51000e-02
 8.26725e+00 4.52700e-02 2.36482e+01 1.31100e-02 6.12700e-02 1.80028e+00
 9.91655e+00 1.43337e+01 1.44383e+01 8.44700e-02 2.15505e+00 2.43938e+01
 1.87000e-02 1.43200e-02 1.50100e-02 1.54450e-01 3.67367e+00 7.67202e+00
 4.41780e-01 2.50461e+01 8.89762e+01 7.02259e+00 3.46600e-02 1.70900e-02
 3.03470e-01 8.37000e-02 3.15330e-01 2.89600e-01 3.23700e-02 9.25200e-02
 3.65900e-02 1.19511e+01 1.28160e-01 1.11604e+01 9.33889e+00 4.68400e-02
 2.28760e-01 3.32105e+00 3.69311e+00 7.83932e+00 2.18700e-02 1.36000e-02
 1.42310e-01 5.64400e-02 6.71772e+00 8.26500e-02 1.02900e-01 7.15100e-02
 3.11300e-02 4.66600e-02 2.24236e+00 2.14090e-01 3.61500e-02 5.78900e-02
 8.70700e-02 5.11358e+01 6.46600e-02 4.81213e+00 3.73800e-02 1.62864e+00
 1.12658e+00 3.53501e+00 1.50234e+01 7.99248e+00 1.58744e+01 1.95100e-02
 8.30800e-02 2.48017e+01 6.88800e-02 8.24809e+00 9.16400e-02 5.51500e-02
 2.00900e-02 1.49632e+00 9.55770e-01 4.20300e-02 9.88430e-01 1.59360e-01
 8.22100e-02 5.20177e+00 1.00245e+00 8.05579e+00 5.47900e-02 1.88360e-01
 4.46200e-02 7.36711e+00 1.58603e+01 1.71200e-01 3.96100e-02 7.84200e-01
 2.10380e-01 1.13081e+00 7.52601e+00 1.44208e+01 1.00840e-01 2.25971e+01
 1.48660e-01 8.18700e-02 2.24890e-01 8.01400e-02 3.22640e-01 1.91860e-01
 3.30450e-01 2.17700e-02].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature
or array.reshape(1, -1) if it contains a single sample.
```

In [35]:
```
len(x_train)
```

Out[35]:
```
404
```

In [36]:
```
len(x_test)
```

Out[36]:
```
102
```

In [37]:
```
len(y_train)
```

Out[37]:
```
404
```

In [38]:
```
len(y_test)
```

Out[38]:
```
102
```

In [ ]:
```

```