Practical No: 1

Design the Machine Learning Model

AIM: Design a simple machine learning model to train the training instances and test the same.

Description:

1. Training Data

Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.

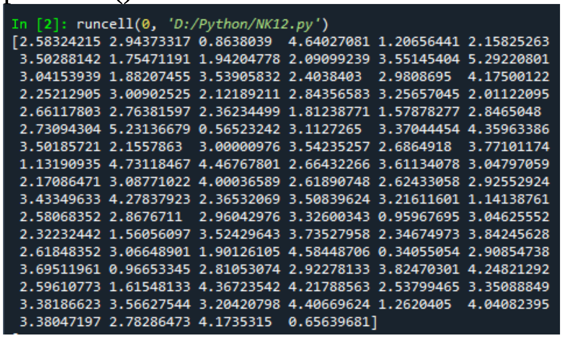
Training data is always more or equal in size than test data

2. Test Data

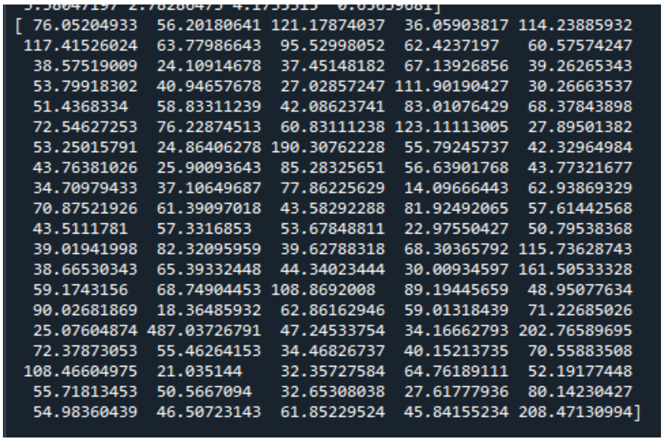
Testing data is used to evaluate our model performance.

Code with output

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)
x = numpy.random.normal(3,1,100)
print(x)
y = numpy.random.normal(150,40,100) /x
print(y)
plt.scatter(x,y)
plt.show()
```



```
In [2]: runcell(0, 'D:/Python/NK12.py')
[2.58324215 2.94373317 0.8638039  4.64027081 1.20656441 2.15825263
 3.50288142 1.75471191 1.94204778 2.09099239 3.55145404 5.29220801
 3.04153939 1.88207455 3.53905832 2.4038403  2.9808695  4.17500122
 2.25212905 3.00902525 2.12189211 2.84356583 3.25657045 2.01122095
 2.66117803 2.76381597 2.36234499 1.81238771 1.57878277 2.8465048
 2.73094304 5.23136679 0.56523242 3.1127265  3.37044454 4.35963386
 3.50185721 2.1557863  3.00000976 3.54235257 2.6864918  3.77101174
 1.13190935 4.73118467 4.46767801 2.66432266 3.61134078 3.04797059
 2.17086471 3.08771022 4.00036589 2.61890748 2.62433058 2.92552924
 3.43349633 4.27837923 2.36532069 3.50839624 3.21611601 1.14138761
 2.58068352 2.8676711  2.96042976 3.32600343 0.95967695 3.04625552
 2.32232442 1.56056097 3.52429643 3.73527958 2.34674973 3.84245628
 2.61848352 3.06648901 1.90126105 4.58448706 0.34055054 2.90854738
 3.69511961 0.96653345 2.81053074 2.92278133 3.82470301 4.24821292
 2.59610773 1.61548133 4.36723542 4.21788563 2.53799465 3.35088849
 3.38186623 3.56627544 3.20420798 4.40669624 1.2620405  4.04082395
 3.38047197 2.78286473 4.1735315  0.65639681]
```

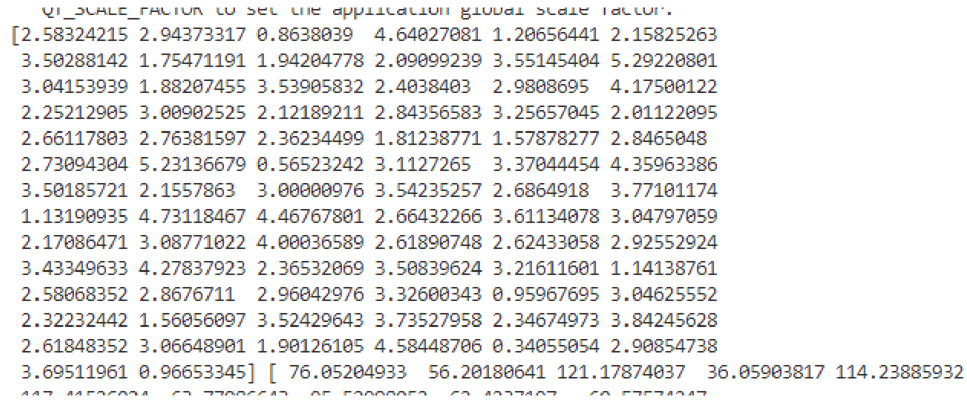


```
 3.38047197 2.78280473 4.1733313  0.63639081]
[ 76.05204933  56.20180641 121.17874037  36.05903817 114.23885932
 117.41526024  63.77986643  95.52998052  62.4237197   60.57574247
  38.57519009  24.10914678  37.45148182  67.13926856  39.26265343
  53.79918302  40.94657678  27.02857247 111.90190427  30.26663537
  51.4368334   58.83311239  42.08623741  83.01076429  68.37843898
  72.54627253  76.22874513  60.83111238 123.11113005  27.89501382
  53.25015791  24.86406278 190.30762228  55.79245737  42.32964984
  43.76381026  25.90093643  85.28325651  56.63901768  43.77321677
  34.70979433  37.10649687  77.86225629  14.09666443  62.93869329
  70.87521926  61.39097018  43.58292288  81.92492065  57.61442568
  43.5111781   57.3316853   53.67848811  22.97550427  50.79538368
  39.01941998  82.32095959  39.62788318  68.30365792 115.73628743
  38.66530343  65.39332448  44.34023444  30.00934597 161.50533328
  59.1743156   68.74904453 108.8692008   89.19445659  48.95077634
  90.02681869  18.36485932  62.86162946  59.01318439  71.22685026
  25.07604874 487.03726791  47.24533754  34.16662793 202.76589695
  72.37873053  55.46264153  34.46826737  40.15213735  70.55883508
 108.46604975  21.035144    32.35727584  64.76189111  52.19177448
  55.71813453  50.5667094   32.65308038  27.61777936  80.14230427
  54.98360439  46.50723143  61.85229524  45.84155234 208.47130994]
```

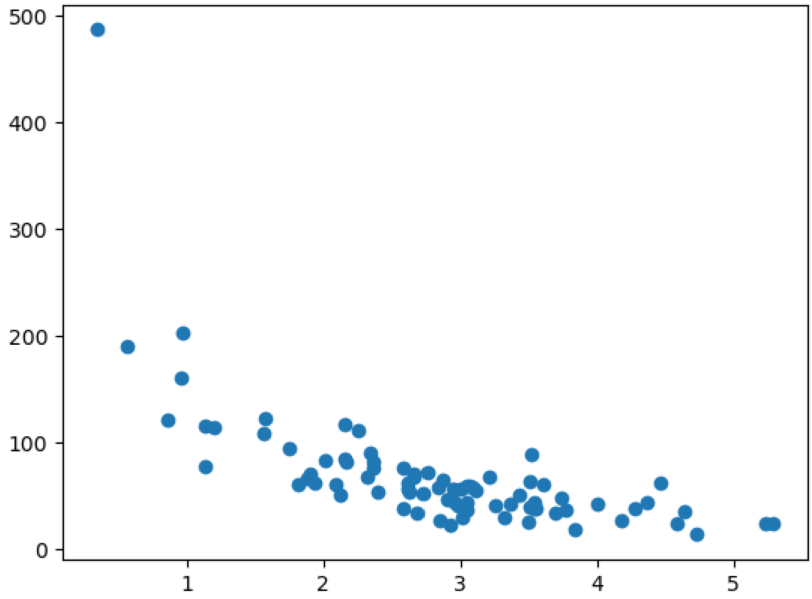
```
train_x = x[:80]
train_y = y[:80]

test_x = x[:20]
test_y = y[:20]

print(train_x,train_y,test_x,test_y)
```



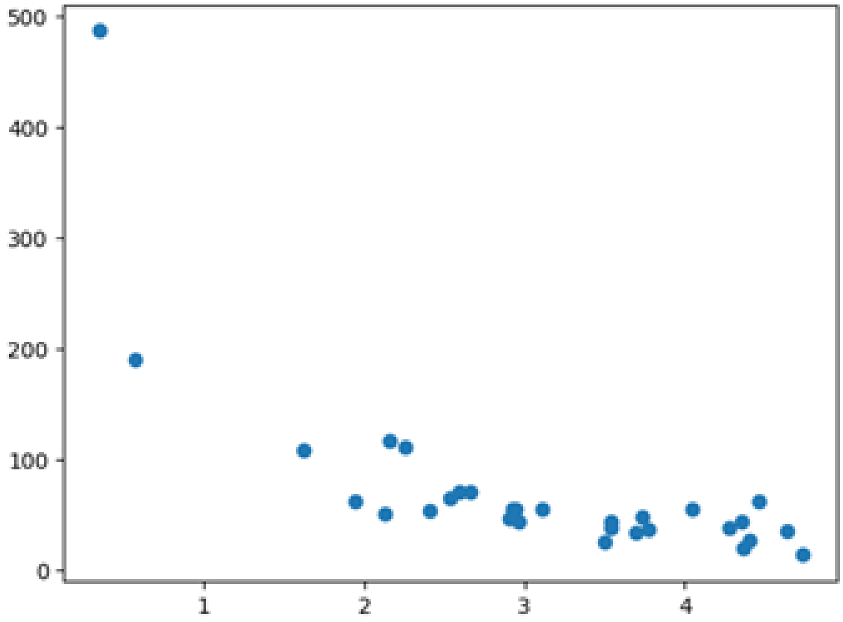
```
  QT_SCALE_FACTOR to set the application global scale factor.
[2.58324215 2.94373317 0.8638039  4.64027081 1.20656441 2.15825263
 3.50288142 1.75471191 1.94204778 2.09099239 3.55145404 5.29220801
 3.04153939 1.88207455 3.53905832 2.4038403  2.9808695  4.17500122
 2.25212905 3.00902525 2.12189211 2.84356583 3.25657045 2.01122095
 2.66117803 2.76381597 2.36234499 1.81238771 1.57878277 2.8465048
 2.73094304 5.23136679 0.56523242 3.1127265  3.37044454 4.35963386
 3.50185721 2.1557863  3.00000976 3.54235257 2.6864918  3.77101174
 1.13190935 4.73118467 4.46767801 2.66432266 3.61134078 3.04797059
 2.17086471 3.08771022 4.00036589 2.61890748 2.62433058 2.92552924
 3.43349633 4.27837923 2.36532069 3.50839624 3.21611601 1.14138761
 2.58068352 2.8676711  2.96042976 3.32600343 0.95967695 3.04625552
 2.32232442 1.56056097 3.52429643 3.73527958 2.34674973 3.84245628
 2.61848352 3.06648901 1.90126105 4.58448706 0.34055054 2.90854738
 3.69511961 0.96653345] [ 76.05204933  56.20180641 121.17874037  36.05903817 114.23885932
 117.41526024  63.77986643  95.52998052  62.4237197   60.57574247
```



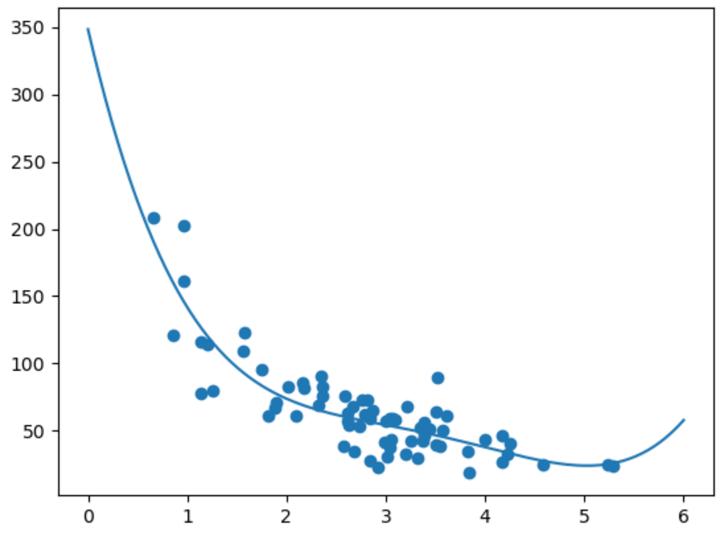
```
plt.scatter(train_x,train_y)
plt.show()

train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.3)

plt.scatter(test_x,test_y)
plt.show()
```

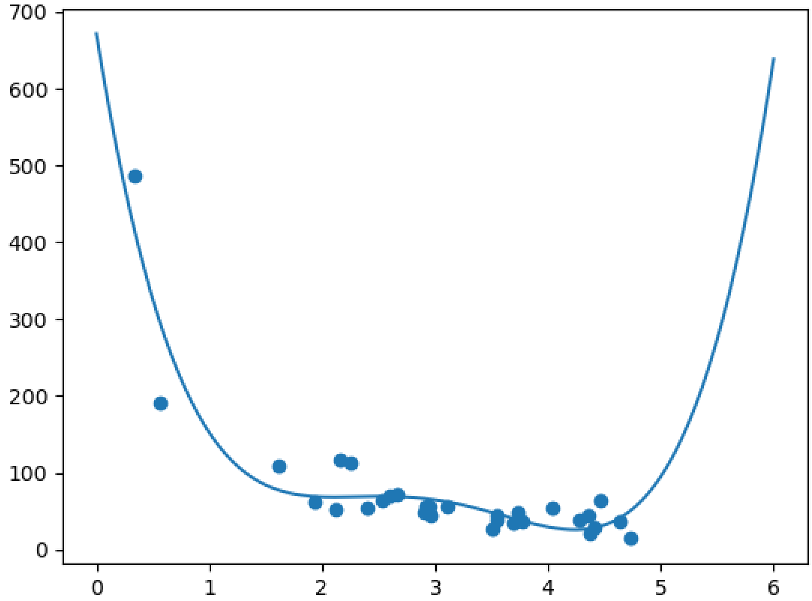


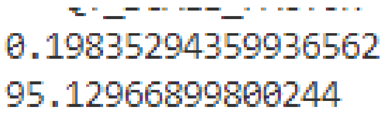
```
mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))
myline = numpy.linspace(0,6,200)
plt.scatter(train_x, train_y)
plt.plot(myline, mymodel(myline))
plt.show()
```



```
mymodel = numpy.poly1d(numpy.polyfit(test_x, test_y, 4))
myline = numpy.linspace(0,6,200)
plt.scatter(test_x, test_y)
plt.plot(myline, mymodel(myline))
plt.show()

r2 = r2_score(train_y, mymodel(train_x))
print(r2)
print(mymodel(5))
```





```
0.19835294359936562
95.12966899800244
```

# As we get high rscore the model is working good

Learnings

First we have created random data for x and y.

Then we have divided it into train test part with 80:20 ratio. visualizes the data and the fitted models.

Then after fitting model, we have evaluated model performance using r square. Then make prediction using trained model