Polynomial Regression

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

In [2]:
```
dataset=pd.read_csv(r"D:\ML_Course\Works_on_python\Linear_Regression\data.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	sno	Temperature	Pressure
0	1	0	0.0002
1	2	20	0.0012
2	3	40	0.0060
3	4	60	0.0300
4	5	80	0.0900

In [4]:
```
dataset.isnull().any()
```

Out[4]:
```
sno            False
Temperature    False
Pressure       False
dtype: bool
```

In [5]:
```
x=dataset.iloc[:,1:2].values
y=dataset.iloc[:,2:3].values
```

In [6]:
```
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 [7]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x)
```

In [8]:
```
x_poly
```

Out[8]:
```
array([[1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00],
       [1.000e+00, 2.000e+01, 4.000e+02, 8.000e+03, 1.600e+05],
       [1.000e+00, 4.000e+01, 1.600e+03, 6.400e+04, 2.560e+06],
       [1.000e+00, 6.000e+01, 3.600e+03, 2.160e+05, 1.296e+07],
       [1.000e+00, 8.000e+01, 6.400e+03, 5.120e+05, 4.096e+07],
       [1.000e+00, 1.000e+02, 1.000e+04, 1.000e+06, 1.000e+08]])
```

In [11]:
```
poly_reg.fit(x_poly,y)
```

Out[11]:
```
PolynomialFeatures(degree=4, include_bias=True, interaction_only=False)
```

In [12]:
```
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(x_poly,y)
```

Out[12]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [13]:
```
poly=lin_reg.predict(x_poly)
```

In [14]:
```
y
```

Out[14]:
```
array([[2.0e-04],
       [1.2e-03],
       [6.0e-03],
       [3.0e-02],
       [9.0e-02],
       [2.7e-01]])
```

In [15]:
```
poly
```

Out[15]:
```
array([[ 4.61111118e-04],
       [-1.05555553e-04],
       [ 8.61111111e-03],
       [ 2.73888889e-02],
       [ 9.13055556e-02],
       [ 2.69738889e-01]])
```

In [16]:
```
from sklearn.metrics import r2_score
accuracy=r2_score(y,poly)
```

In [17]:
```
accuracy
```

Out[17]:
```
0.9996910780718014
```

In [18]:
```
import matplotlib.pyplot as plt
plt.scatter(x,y,color="red")
plt.plot(x,poly)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x1d72da46eb8>]
```

Decision Tree

In [19]:
```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor(random_state=0)
dtr.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

In [20]:
```
y_dtr=dtr.predict(x_test)
```

In [21]:
```
accuracy_dtr=r2_score(y_test,y_dtr)
```

In [22]:
```
accuracy_dtr
```

Out[22]:
```
0.06958677685950398
```

Random Forest

In [31]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators = 4,random_state = 0)#n_estimator is more then more accu or may be not
rfr.fit(x_train,y_train)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: 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().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[31]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=4, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [32]:
```
y_rfr=rfr.predict(x_test)
```

In [33]:
```
accuracy_rfr=r2_score(y_test,y_rfr)
```

In [34]:
```
accuracy_rfr
```

Out[34]:
```
0.0700826446280991
```

In [ ]: