In [ ]:
```
#19IT016
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

In [10]:
```
df = pd.read_csv("WeatherDataP.csv")
df
```

Out[10]:

	Pressure	Humidity
0	1014.40	0.62
1	1014.20	0.66
2	1014.47	0.79
3	1014.45	0.82
4	1014.49	0.83
5	1014.52	0.85
6	1014.16	0.83
7	1014.24	0.78
8	1014.25	0.72
9	1013.96	0.61
10	1013.85	0.52
11	1013.04	0.46
12	1012.22	0.40
13	1011.44	0.40
14	1010.52	0.37
15	1009.83	0.40
16	1009.26	0.36
17	1008.76	0.43
18	1008.36	0.50
19	1008.11	0.53
20	1008.15	0.55
21	1007.85	0.58
22	1007.89	0.59
23	1007.36	0.60
24	1007.26	0.63

Why you want to apply regression on selected dataset?

Because our Y which is target variable is of continuous type so we can apply regression

How many total observations in data?

There are total 25 observations.

How many independent variables?

In This dataset,we could say pressure is the independent variables.

Which is dependent variable?

Humidity is the dependent variable.

Which are most useful variable in estimation? Prove using correlation.

In this dataset there is only one independent variable. So there is no selection of most useful variable. But following is correlation between independent and dependent variable.

In [6]:
```
df.corr()
```

Out[6]:

	Pressure	Humidity
Pressure	1.00000	0.55263
Humidity	0.55263	1.00000

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

Out[7]:

	Pressure	Humidity
count	25.000000	25.0000
mean	1011.481600	0.5932
std	2.873799	0.1590
min	1007.260000	0.3600
25%	1008.360000	0.4600
50%	1012.220000	0.5900
75%	1014.240000	0.7200
max	1014.520000	0.8500

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pressure  25 non-null     float64
 1   Humidity  25 non-null     float64
dtypes: float64(2)
memory usage: 528.0 bytes
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Pressure    0
Humidity    0
dtype: int64
```

In [75]:
```
# why we are not taking series and using .values we are taking array ?
# bcz reshape will not work in pandas series , why actually reshape() is numpy method so x must be a array
x = df['Pressure'].values
y = df['Humidity'].values
x

#x=df.Pressure
#x
```

Out[75]:
```
array([1014.4 , 1014.2 , 1014.47, 1014.45, 1014.49, 1014.52, 1014.16,
       1014.24, 1014.25, 1013.96, 1013.85, 1013.04, 1012.22, 1011.44,
       1010.52, 1009.83, 1009.26, 1008.76, 1008.36, 1008.11, 1008.15,
       1007.85, 1007.89, 1007.36, 1007.26])
```

In [76]:
```
x= x.reshape(-1,1) # reshape(rows, columns)
x

#Unknown Dimension
#You are allowed to have one "unknown" dimension.

#Meaning that you do not have to specify an exact number for one of the dimensions in the reshape method.

#Pass -1 as the value, and NumPy will calculate this number for you.
# Note: We can not pass -1 to more than one dimension.
```

Out[76]:
```
array([[1014.4 ],
       [1014.2 ],
       [1014.47],
       [1014.45],
       [1014.49],
       [1014.52],
       [1014.16],
       [1014.24],
       [1014.25],
       [1013.96],
       [1013.85],
       [1013.04],
       [1012.22],
       [1011.44],
       [1010.52],
       [1009.83],
       [1009.26],
       [1008.76],
       [1008.36],
       [1008.11],
       [1008.15],
       [1007.85],
       [1007.89],
       [1007.36],
       [1007.26]])
```

In [77]:
```
poly = PolynomialFeatures(degree=1)
poly
```

Out[77]:
```
PolynomialFeatures(degree=1)
```

In [78]:
```
x_poly = poly.fit_transform(x) #this function return valueerror if we have store x values in series instead of array
x_poly

#Often, the input features for a predictive modeling task interact in unexpected and often nonlinear ways.

#These interactions can be identified and modeled by a learning algorithm. Another approach is to engineer new features that expose these interacti

#These features are called interaction and polynomial features and allow the use of simpler modeling algorithms as some of the complexity of interp
```

Out[78]:
```
array([[1.00000e+00, 1.01440e+03],
       [1.00000e+00, 1.01420e+03],
       [1.00000e+00, 1.01447e+03],
       [1.00000e+00, 1.01445e+03],
       [1.00000e+00, 1.01449e+03],
       [1.00000e+00, 1.01452e+03],
       [1.00000e+00, 1.01416e+03],
       [1.00000e+00, 1.01424e+03],
       [1.00000e+00, 1.01425e+03],
       [1.00000e+00, 1.01396e+03],
       [1.00000e+00, 1.01385e+03],
       [1.00000e+00, 1.01304e+03],
       [1.00000e+00, 1.01222e+03],
       [1.00000e+00, 1.01144e+03],
       [1.00000e+00, 1.01052e+03],
       [1.00000e+00, 1.00983e+03],
       [1.00000e+00, 1.00926e+03],
       [1.00000e+00, 1.00876e+03],
       [1.00000e+00, 1.00836e+03],
       [1.00000e+00, 1.00811e+03],
       [1.00000e+00, 1.00815e+03],
       [1.00000e+00, 1.00785e+03],
       [1.00000e+00, 1.00789e+03],
       [1.00000e+00, 1.00736e+03],
       [1.00000e+00, 1.00726e+03]])
```

In [85]:
```
poly.fit(x_poly,y) # TO FIND OUT BEST FIT CO-EFFICENT / BEST LINE FIT CURVE FOR THE LINE OF INPUT(X_POLY),DEPENDS ON Y
```

Out[85]:
```
PolynomialFeatures(degree=5)
```

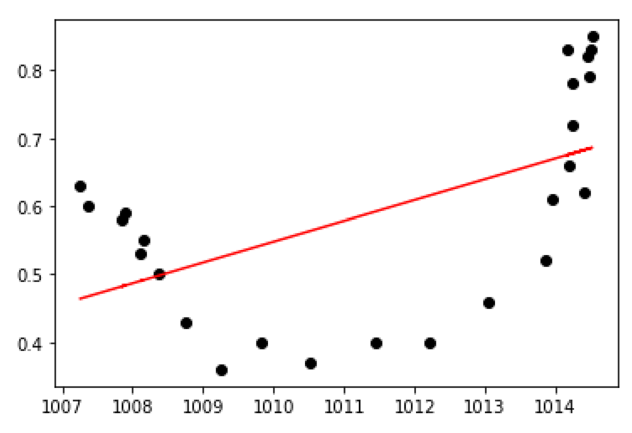
In [80]:
```
reg = LinearRegression()
reg.fit(x_poly,y)
```

Out[80]:
```
LinearRegression()
```

In [81]:
```
y_preg = reg.predict(x_poly)
```

In [82]:
```
plt.scatter(x,y,color='black')
plt.plot(x, y_preg, color='red')
```

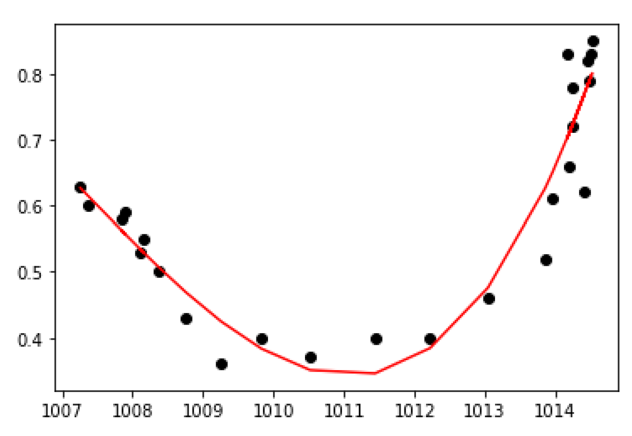
Out[82]:
```
[<matplotlib.lines.Line2D at 0x1f13c520190>]
```



In [83]:
```
#polynomial of degree 3

poly = PolynomialFeatures(degree=3)
x_poly = poly.fit_transform(x)
poly.fit(x_poly,y)
reg.fit(x_poly,y)
y_preg = reg.predict(x_poly)
plt.scatter(x,y,color='black')
plt.plot(x, y_preg, color='red')
```

Out[83]:
```
[<matplotlib.lines.Line2D at 0x1f13c58c2e0>]
```



In [84]:
```
#polynomial of degree 5

poly = PolynomialFeatures(degree=5)
x_poly = poly.fit_transform(x)
poly.fit(x_poly,y)
reg.fit(x_poly,y)
y_preg = reg.predict(x_poly)
plt.scatter(x,y,color='black')
plt.plot(x, y_preg, color='red')
```

Out[84]:
```
[<matplotlib.lines.Line2D at 0x1f13c5fa280>]
```