Polynomial Regression

https://en.wikipedia.org/wiki/Polynomial_regression (https://en.wikipedia.org/wiki/Polynomial_regression)

We use this regression model when our dataset is not linear in nature. Linear models fail in prediction if the dataset is not linearly dependent w.r.t to dependant and independent variable. Polynomial regression models are more effective to approach real time datasets like employee salary based on experience, population growth estimate and more.

The typical polynomial regression equation will be as below:
$y = b0 + b1 * x^2 + b2 * x^3 ... + bn * x^n$
It is still considered to be part of linear regression because coefficients still have the unit power.
They are called by Curvilinear Regression.

We shall be using sklearn for the above.
Considering a dataset of china's per capita expenditure from 1978-2008.

In [1]:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('china_carbon.csv')
X = dataset.iloc[:, 0:1].values
y = dataset.iloc[:, 6].values
print("Data for first ten years from 1978-1987\n")
print("Year  Percapita expenditure")
for i in range(10):
    print("{}   {}".format(X[i][0], y[i]))
```

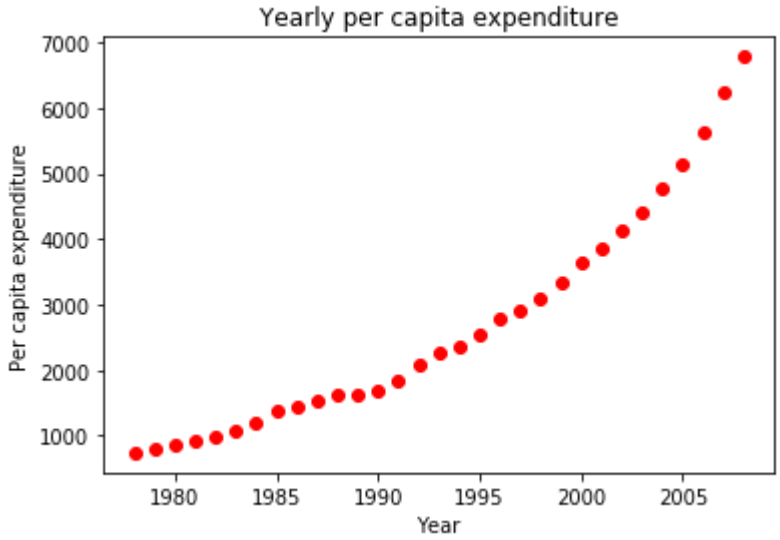
```
Data for first ten years from 1978-1987

Year  Percapita expenditure
1978   740
1979   791
1980   862
1981   934
1982   997
1983   1079
1984   1207
1985   1370
1986   1435
1987   1520
```

Visualising the dataset can show us the idea about it's distribution and non-linearlity.

In [2]:

```
plt.plot(X,y,'ro')
plt.title("Yearly per capita expenditure")
plt.xlabel("Year")
plt.ylabel("Per capita expenditure")
plt.show()
```

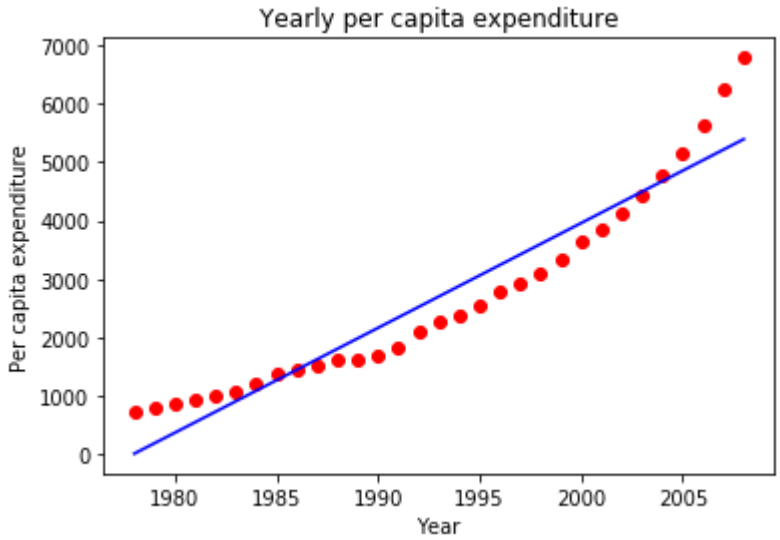


Let's create linear model for the above datapoints and observe the regressor line w.r.t datapoints.

In [3]:

```
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)

# Visualising the result - linear regression
plt.plot(X,y,'ro')
plt.plot(X,lin_reg.predict(X),'b-')
plt.title("Yearly per capita expenditure")
plt.xlabel("Year")
plt.ylabel("Per capita expenditure")
plt.show()
```

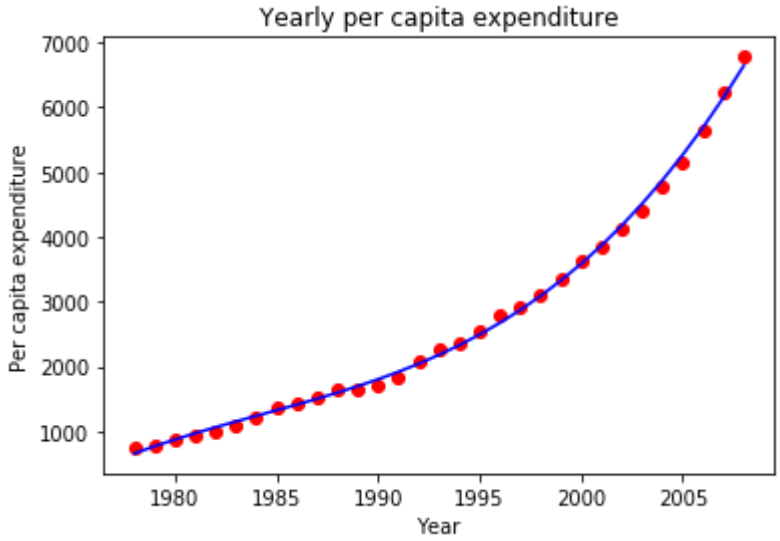


We can see from above plotting that linear regression doesn't fit effectively with datapoints.
Let's try with polynomial regression (Non-linear regression).

In [4]:

```
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4) #Considering degree of polynomial to b
e 4
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)

# Visualising the result - Polynomial Regression
plt.plot(X,y,'ro')
plt.plot(X,lin_reg_2.predict(poly_reg.fit_transform(X)),'b-')
plt.title("Yearly per capita expenditure")
plt.xlabel("Year")
plt.ylabel("Per capita expenditure")
plt.show()
```



We can observe that polynomial regressor line fits well with datapoints, which effective in predictions in scenarios like the above (where growth or data distribution is not linear).
There may be datasets where data distribution is not quadratic or cubic in nature. In such cases, polynomial regression is not effective way to model because the ploynomial doesn't allow sudden change in the curve.

The polynomial regression increases the number of features by adding new terms to achieve or fit model with dataset, which in turn can become overhead as computational overhead needlessly.

Further Reading:

1.https://onlinecourses.science.psu.edu/stat501/node/324 (https://onlinecourses.science.psu.edu/stat501/node/324)
2.https://www.coursera.org/learn/machine-learning/lecture/Rqgfz/features-and-polynomial-regression (https://www.coursera.org/learn/machine-learning/lecture/Rqgfz/features-and-polynomial-regression)
3.https://en.wikipedia.org/wiki/Polynomial_regression (https://en.wikipedia.org/wiki/Polynomial_regression)