Polynomial Regression

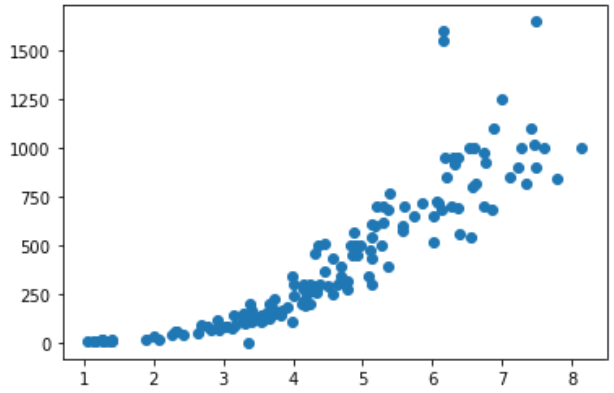
Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

In a curvilinear relationship, the value of the target variable changes in a non-uniform manner with respect to the predictor (s).

Lets understand the need for polynomial regression

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [2]:
```
data = pd.read_csv('polynomial_fish.csv')
x=data.iloc[:,1:2].values
y= data.iloc[:,-1].values
plt.scatter(x,y)
plt.show()
```



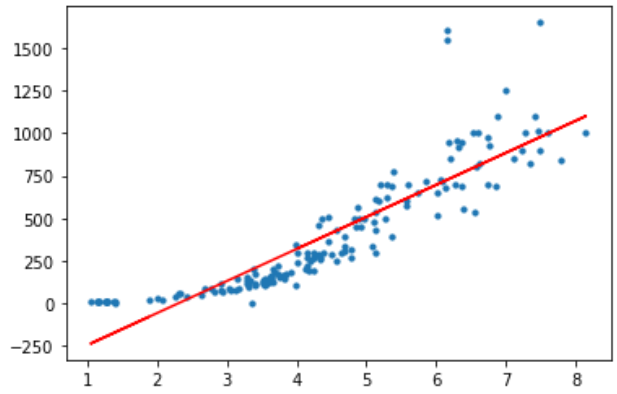
Let’s apply a linear regression model to this dataset

In [38]:
```
regressor = LinearRegression()
regressor.fit(x,y)
```

Out[38]:
```
LinearRegression()
```

In [39]:
```
y_pred = regressor.predict(x)
```

In [40]:
```
plt.scatter(x, y, s=10)
plt.plot(x, y_pred, color='r')
plt.show()
```



In [42]:
```
# accuracy metrics
from sklearn.metrics import r2_score
r2score = r2_score(x, regressor.predict(x))
r2score
```

Out[42]:
```
-90004.68257519293
```

We can see that the straight line is unable to capture the patterns in the data.

So for such cases, where data points are arranged in a non-linear fashion, we need the Polynomial Regression model.

Equation of the Polynomial Regression Model

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + ... + b_n x^n$$

Building the Polynomial regression model

It will be a little different from the Simple Linear model.

Because here we will use PolynomialFeatures class of preprocessing library.

We are using this class to add some extra features to our dataset.

In [6]:
```
# Splitting the dataset into the Training set and Test set
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 [32]:
```
from sklearn.preprocessing import PolynomialFeatures
polynomial_features= PolynomialFeatures(degree=4)
 # converting our feature matrix into polynomial feature matrix and then fitting it to the Polynomic
x_poly = polynomial_features.fit_transform(x_train)
```

In [33]:
```
x_poly
```

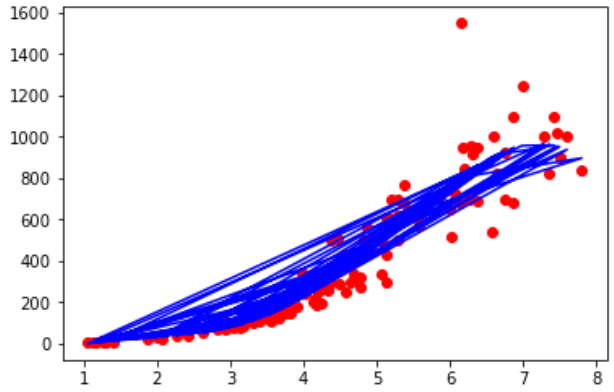
Out[33]:
```
array([[1.00000000e+00, 6.14400000e+00, 3.77487360e+01, 2.31928234e+02,
        1.42496707e+03],
       [1.00000000e+00, 4.38440000e+00, 1.92229634e+01, 8.42811606e+01,
        3.69522320e+02],
       [1.00000000e+00, 5.28540000e+00, 2.79354532e+01, 1.47650044e+02,
        7.80389543e+02],
       [1.00000000e+00, 3.72300000e+00, 1.38607290e+01, 5.16034941e+01,
        1.92119808e+02],
       [1.00000000e+00, 4.63540000e+00, 2.14869332e+01, 9.96005300e+01,
        4.61688297e+02],
       [1.00000000e+00, 5.28010000e+00, 2.78794560e+01, 1.47206316e+02,
        7.77264067e+02],
       [1.00000000e+00, 6.74080000e+00, 4.54383846e+01, 3.06291063e+02,
        2.06464680e+03],
       [1.00000000e+00, 3.52500000e+00, 1.24256250e+01, 4.38003281e+01,
        1.54396157e+02],
       [1.00000000e+00, 6.01800000e+00, 3.62163240e+01, 2.17949838e+02,
        1.31162212e+03],
       [1.00000000e+00, 2.43200000e+00, 5.91462400e+00, 1.43843656e+01,
```

In [34]:
```
regressor1 = LinearRegression()
regressor1.fit(x_poly,y_train)
```

Out[34]:
```
LinearRegression()
```

In [35]:
```
# Visualitaion

plt.scatter(x_train,y_train,color='red')
plt.plot(x_train,regressor1.predict(polynomial_features.fit_transform(x_train)),color='blue')
plt.show()
```



In [36]:
```
y_pred = regressor1.predict(polynomial_features.fit_transform(x_test))
```

In [37]:
```
# accuracy metrics
from sklearn.metrics import r2_score
r2score = r2_score(y_test, y_pred)
r2score
```

Out[37]: 0.7676201122277835

In [25]:
```
# accuracy metrics
from sklearn.metrics import r2_score
r2score = r2_score(y_test, regressor.predict(x_test))
r2score
```

Out[25]: 0.739595402766277

In [ ]: