Polynomial Regression

Importing the Libraries

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

Importing the dataset

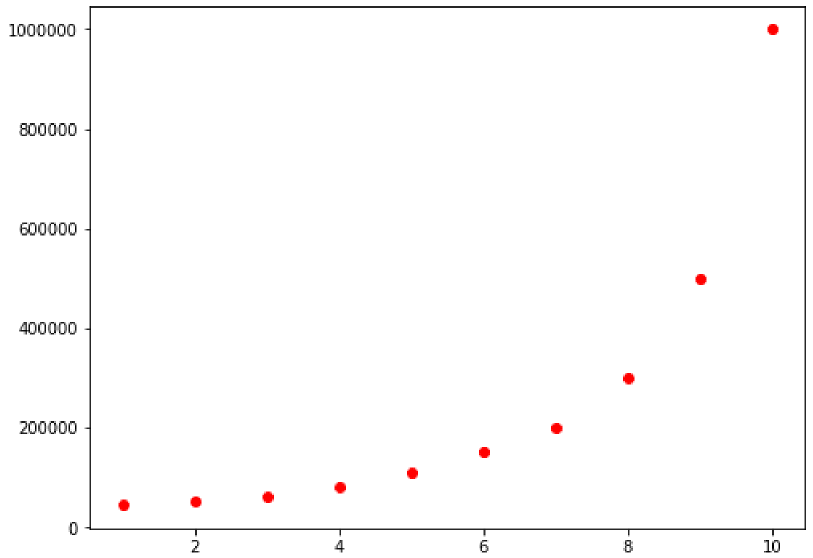
In [2]:

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

Plotting the data distribution

In [3]:

```
plt.figure(figsize=(8,6))
plt.scatter(X, y, color = 'red')
plt.show()
```



Splitting the dataset into the Training set and Test set

Since we need the entire dataset for training (for this particular problem), we will not split the data into training and test sets

In [4]:

```
# 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)
```

Fitting Linear Regression to the dataset

In [5]:

```
from sklearn.linear_model import LinearRegression
linear_regressor = LinearRegression()
linear_regressor.fit(X, y)
```

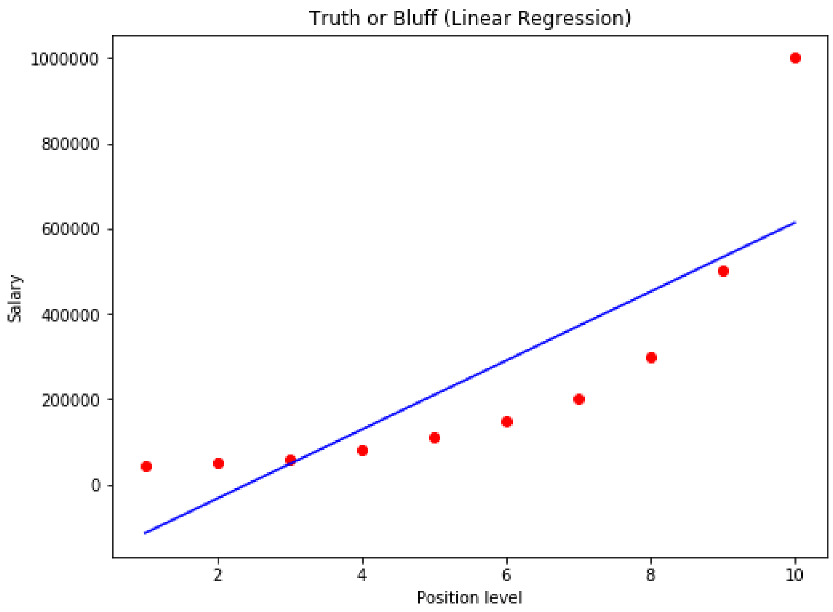
Out[5]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Visualising the Linear Regression results

In [6]:

```
plt.figure(figsize=(8,6))
plt.scatter(X, y, color = 'red')
plt.plot(X, linear_regressor.predict(X), color = 'blue')

plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Fitting Polynomial Regression to the dataset

In [7]:

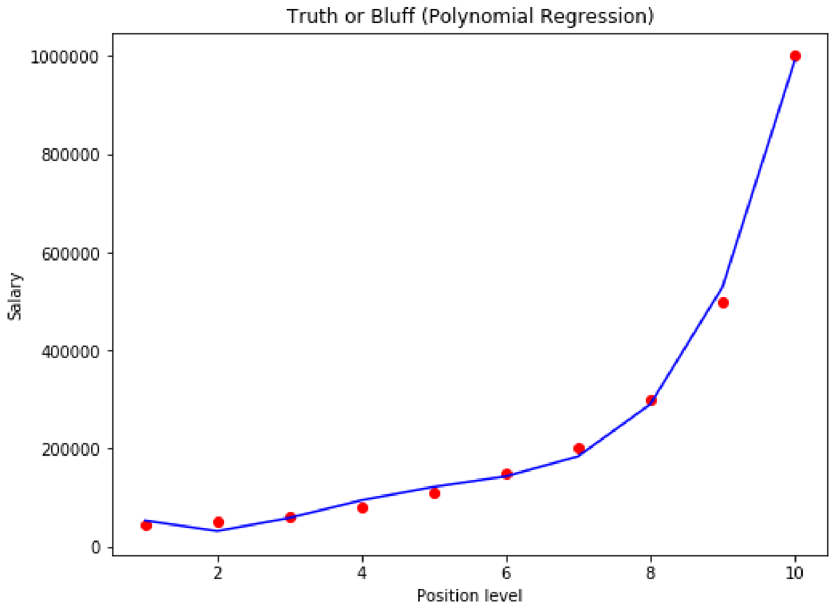
```
from sklearn.preprocessing import PolynomialFeatures
poly_features_transformer = PolynomialFeatures(degree = 4)
X_poly = poly_features_transformer.fit_transform(X)

# Fitting the higher order terms to the model
polynomial_regressor = LinearRegression()
polynomial_regressor.fit(X_poly, y)
```

Out[7]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

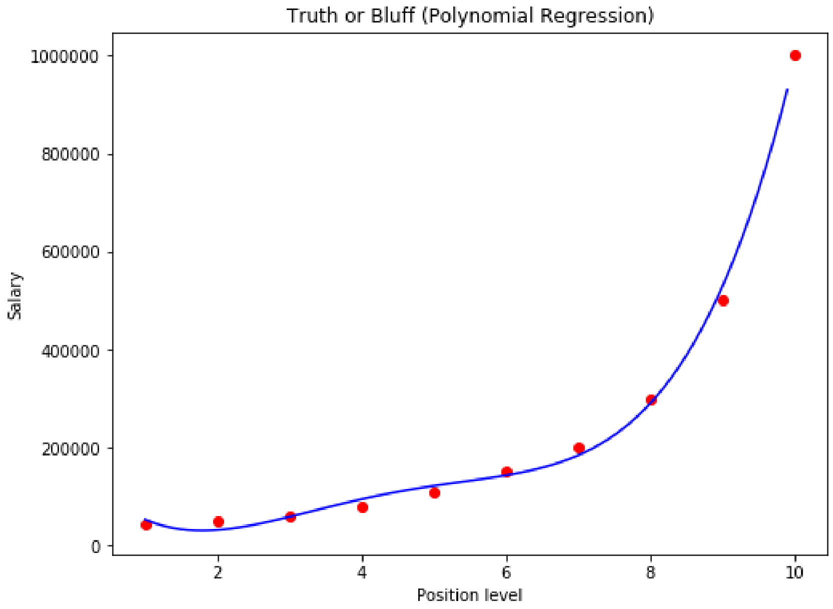
Visualising the Polynomial Regression results

```
In [8]: plt.figure(figsize=(8,6))
        plt.scatter(X, y, color = 'red')
        plt.plot(X, polynomial_regressor.predict(poly_features_transformer.fit_transform(X)), color = 'blue')
        plt.title('Truth or Bluff (Polynomial Regression)')
        plt.xlabel('Position level')
        plt.ylabel('Salary')
        plt.show()
```



Visualising the Polynomial Regression results (for higher resolution and smoother curve)

```
In [9]: plt.figure(figsize=(8,6))
        X_grid = np.arange(min(X), max(X), 0.1)
        X_grid = X_grid.reshape((len(X_grid), 1))
        plt.scatter(X, y, color = 'red')
        plt.plot(X_grid, polynomial_regressor.predict(poly_features_transformer.fit_transform(X_grid)), color
        plt.title('Truth or Bluff (Polynomial Regression)')
        plt.xlabel('Position level')
        plt.ylabel('Salary')
        plt.show()
```



As we can see above, the higher degrees of the expression leads to better fit. This however, can also culminate into a over-fitted model.

Please ensure that the regressor is not overfitted to suit the training data

Predicting a new result with Linear Regression

In [10]:

```
linear_regressor.predict(np.array([[10]]))
```

Out[10]:

```
array([613454.54545455])
```

Predicting a new result with Polynomial Regression

In [11]:

```
polynomial_regressor.predict(poly_features_transformer.fit_transform(np.array([[10]])))
```

Out[11]:

```
array([988916.08391567])
```

Residuals

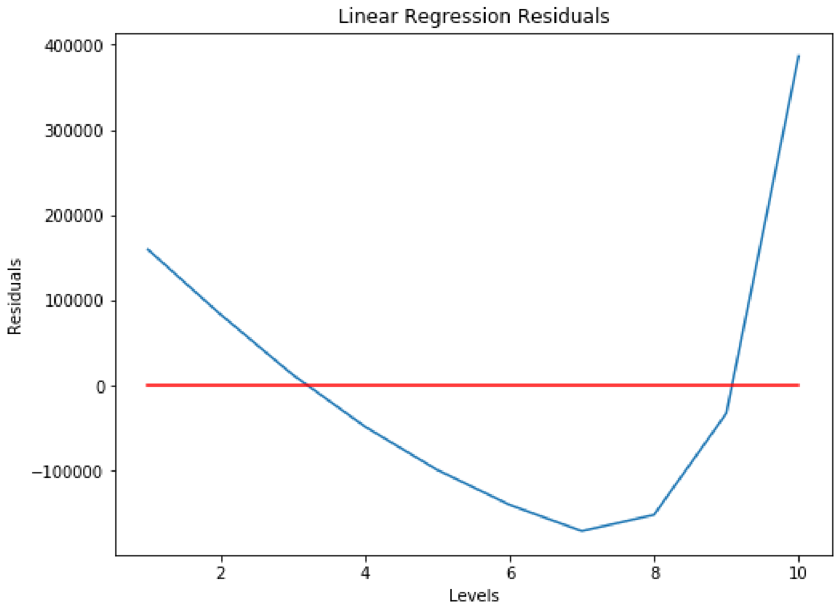
Linear Regression Residuals Plot

In [12]:

```
linear_regression_residuals = y - linear_regressor.predict(X)
```

In [13]:

```
plt.figure(figsize=(8,6))
plt.plot(X, linear_regression_residuals)
plt.plot(X, [0]*len(X), color='r')
plt.title("Linear Regression Residuals")
plt.xlabel("Levels")
plt.ylabel("Residuals")
plt.show()
```



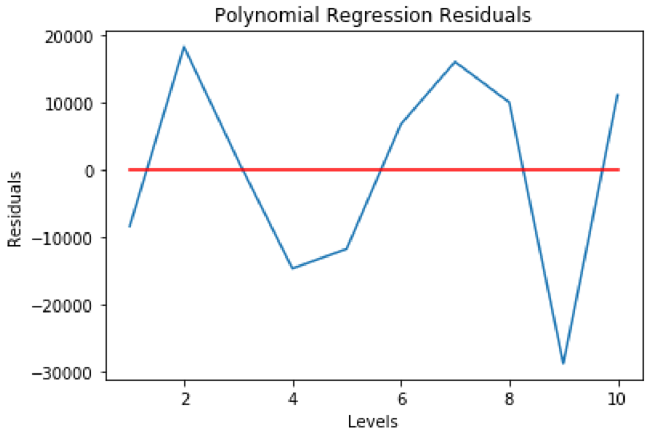
Polynomial Regression Residuals Plot

In [14]:

```
polynomial_regression_residuals = y - polynomial_regressor.predict(poly_features_transformer.fit_trar
```

In [15]:

```
plt.figure(figsize=(6,4))
plt.plot(X, polynomial_regression_residuals)
plt.plot(X, [0]*len(X), color='r')
plt.title("Polynomial Regression Residuals")
plt.xlabel("Levels")
plt.ylabel("Residuals")
plt.show()
```



In [16]:

```
residuals = pd.DataFrame(columns=['Linear', 'Polynomial'], index=dataset.Level)
```

In [17]:

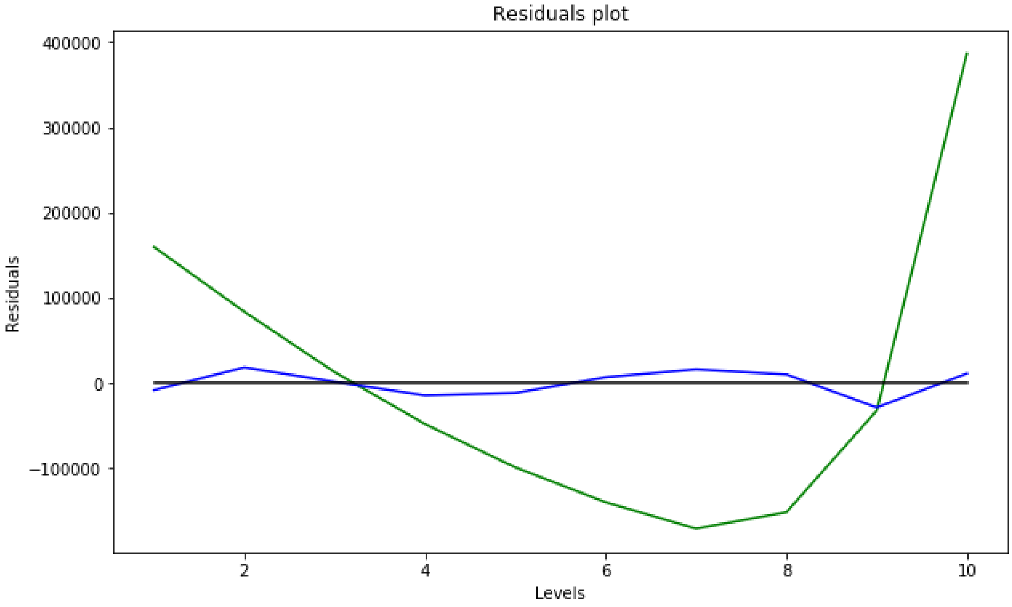
```
residuals['Linear'] = linear_regression_residuals
residuals['Polynomial'] = polynomial_regression_residuals
residuals
```

Out[17]:

	Linear	Polynomial
Level		
1	159454.545455	-8356.643357
2	83575.757576	18240.093240
3	12696.969697	1357.808858
4	-48181.818182	-14632.867133
5	-99060.606061	-11724.941725
6	-139939.393939	6724.941725
7	-170818.181818	15996.503496
8	-151696.969697	10005.827506
9	-32575.757576	-28694.638694
10	386545.454545	11083.916084

Plot of Residuals

```
In [18]: plt.figure(figsize=(10,6))
         plt.plot(X, linear_regression_residuals, color='green')
         plt.plot(X, polynomial_regression_residuals, color='blue')
         plt.plot(X, [0] * len(X), color='black')
         plt.title("Residuals plot")
         plt.xlabel("Levels")
         plt.ylabel("Residuals")
         plt.show()
```


Residuals plot
Residuals
Levels