Polynomial Regression

Different Equations:

- Simple Linear Regression:

$$y = b_0 + b_1 X_1$$

- Multiple Linear Regression:

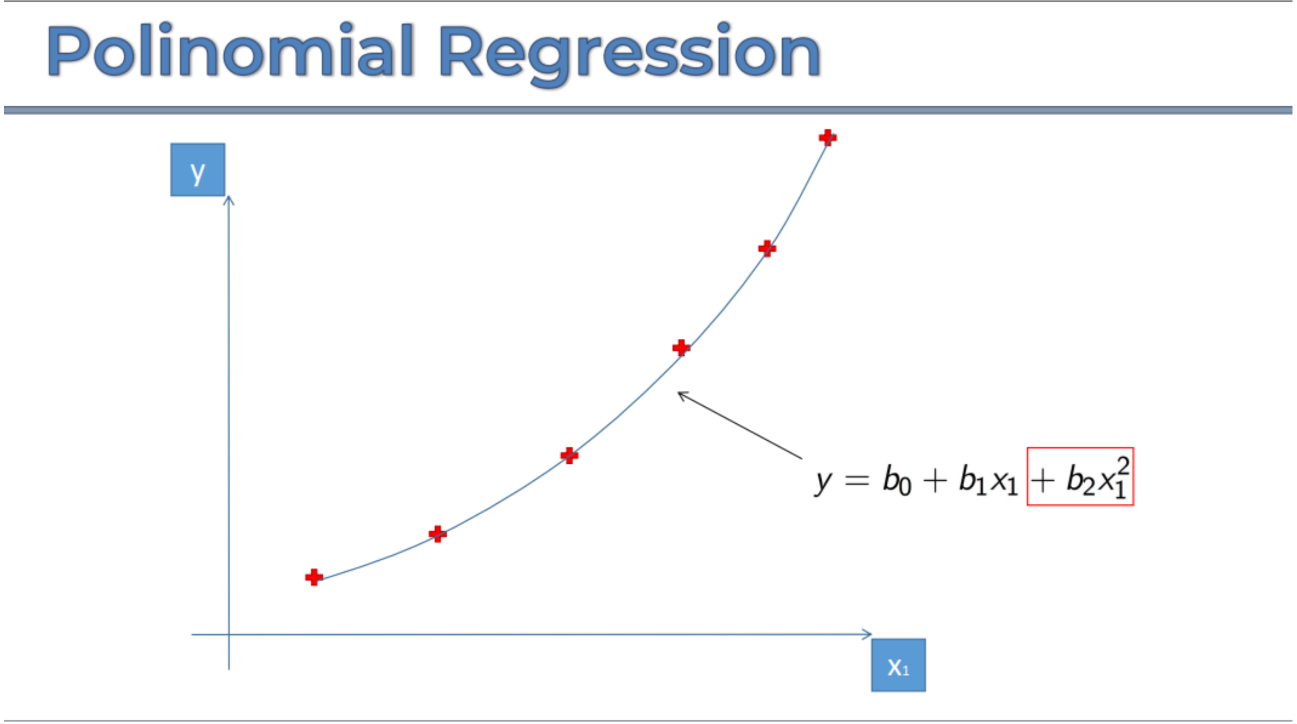
$$y = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_n X_n$$

- Polynomial Linear Regression:

$$y = b_0 + b_1 X_1 + b_2 X_1^2 + \ldots + b_n X_1^n$$

When to use Polynomial Regressions:

- Use this regression model when relationship between independent variable and dependent variable can be approximated using a polynomial function.
- Usually a case when there is a nonlinear relationship with curved pattern, meaning if we see a curved pattern in the scatter plot; it is best to use a polynomial regression.
- Important to note, the reason we call this Polynomial Regression as Polynomial Linear Regression is because of the Linear Relation between the power and base of the coeffcient. For example, b_2 and X^2



Example:

Given the dataset below, we have different positions (based on levels) and their respective salaries. The level and position are do not share a linear relation with the salary. Salary is often seen as a exponential relation to experience and position/level.

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Applying Polynomial Linear Regressions

- **Import Libraries**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

- **Import Dataset**

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

Because the dataset contains a column `level` which pretty much encodes the categorical data, in this case column one; which is the position name. Meaning we can skip the first column when importing the dataset. Set matrix of features for independent value `X` to the data in column `Level` and the dependent value remains the same, in this case we do not need to encode the data as it is not categorical.

Observe Difference between Linear Regression and Polynomial Regression

- **Training the Linear Regression model on the whole dataset**

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

There is a reason we did not split the data, our dataset is very small. We simply use maximum data as our training set.

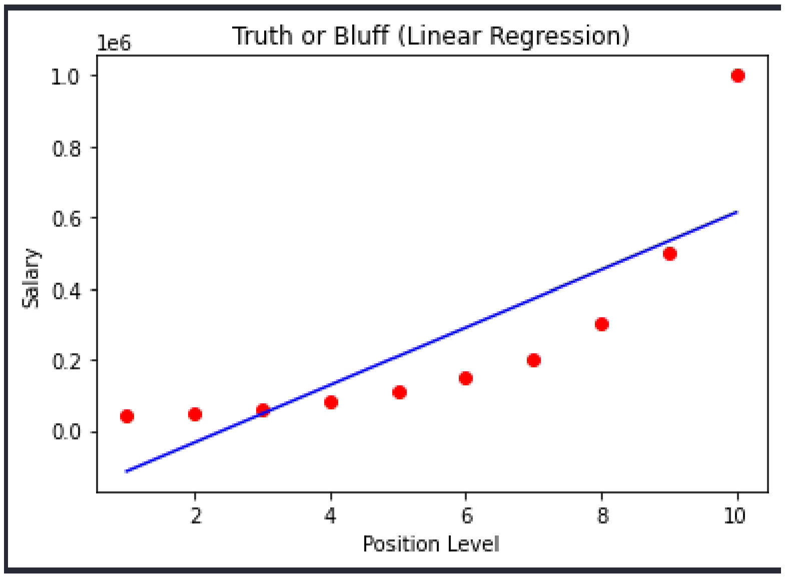
- **Training the Polynomial Regression Model on the Whole Dataset**

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

First create a matrix of features of the powers, where X_1 as the first feature, X_1^2 as the second feature and X_1^n as the nth feature, this is called `x_poly`. Next we will create a linear regressor to integrate these powered features from the matrix of features containing the powers, called `lin_reg_2`. The Degree value is something we would adjust, it represents the value of `n`.

- **Visualizing Linear Regression Model on whole Dataset**

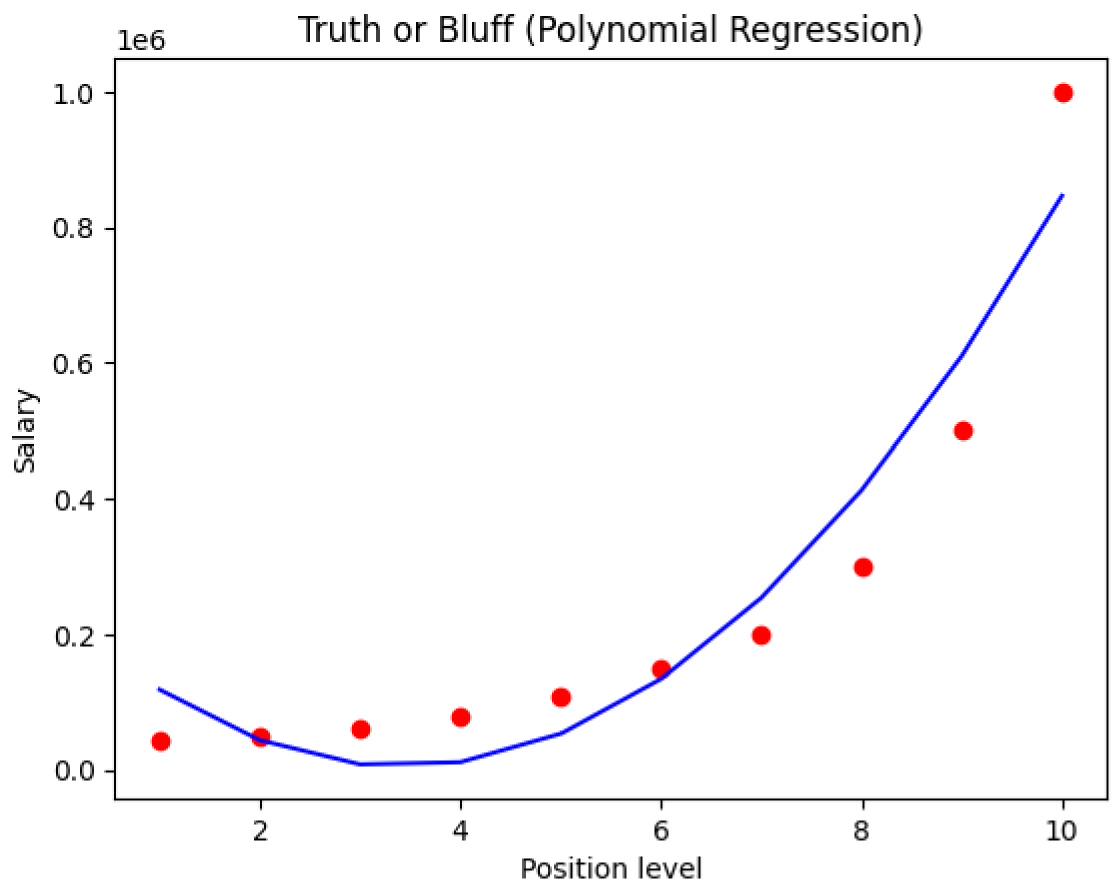
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```



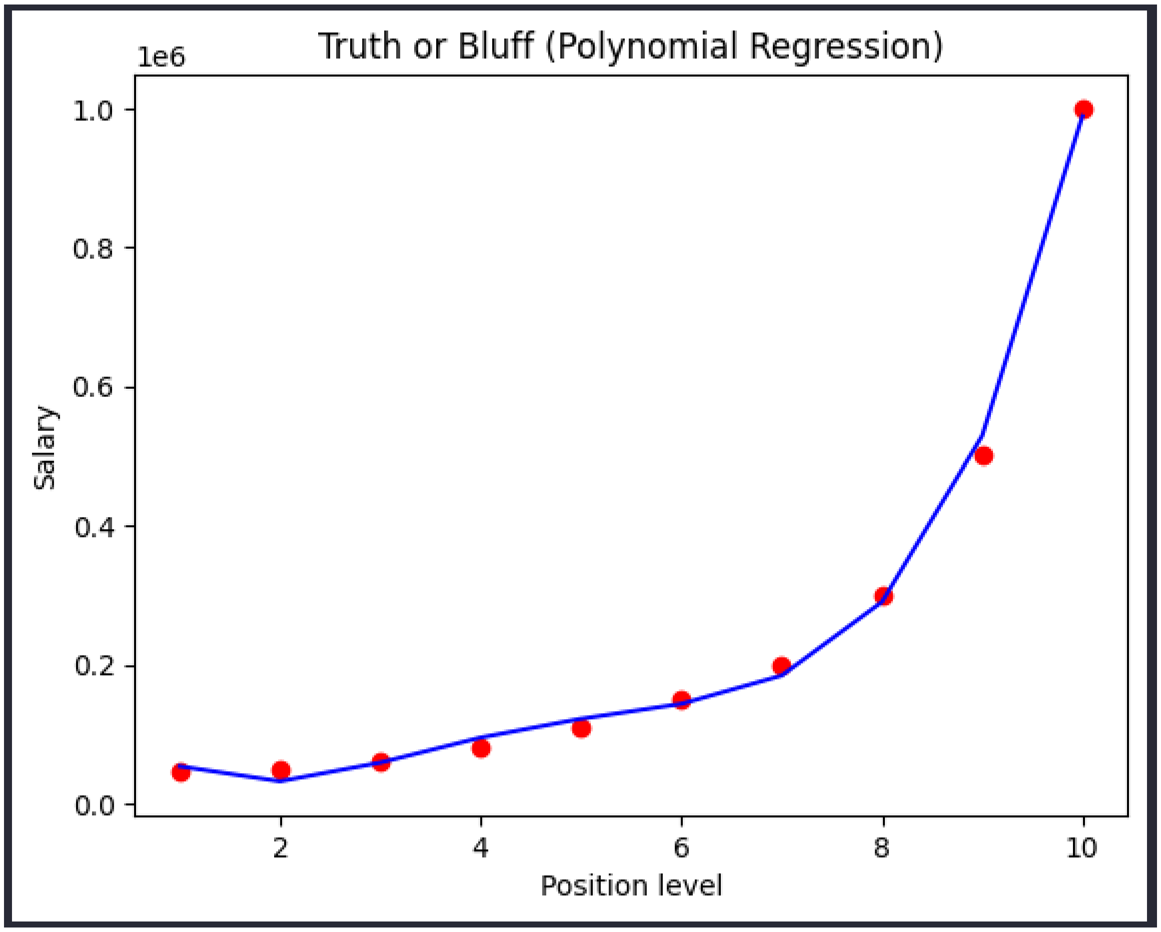
- **Visualizing Polynomial Regression Model on whole Dataset**

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Remember to change the regressor to Polynomial, and when predicting; include the transformed matrix of features based on the degree.



This plot shows the regression model when n or degree is set to two. But this regression can be made better by increasing the value or degree.



This plot shows how the regression model which perfectly fits the plot. But to prepare this plot for smother curves and higher resolution, use the following code:

```
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, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

The idea is to not only take the integers for `Level`, but rather including the decimal points like 1.1, 1.2, etc. This allows the plot to give accurate results with smother curves.

Predicting Salaries based on Level with both Linear and Polynomial Regressions

- **Predicting a new result with Linear Regression**

```
lin_reg.predict([[6.5]])
#Output: array([330378.78787879])
```

- **Predicting a new result with Polynomial Regression**

```
lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
#Output: array([158862.45265155])
```