Polynomial regression

Polynomial regression is a type of regression analysis used in statistics and machine learning to model relationships between variables when the relationship isn't linear.

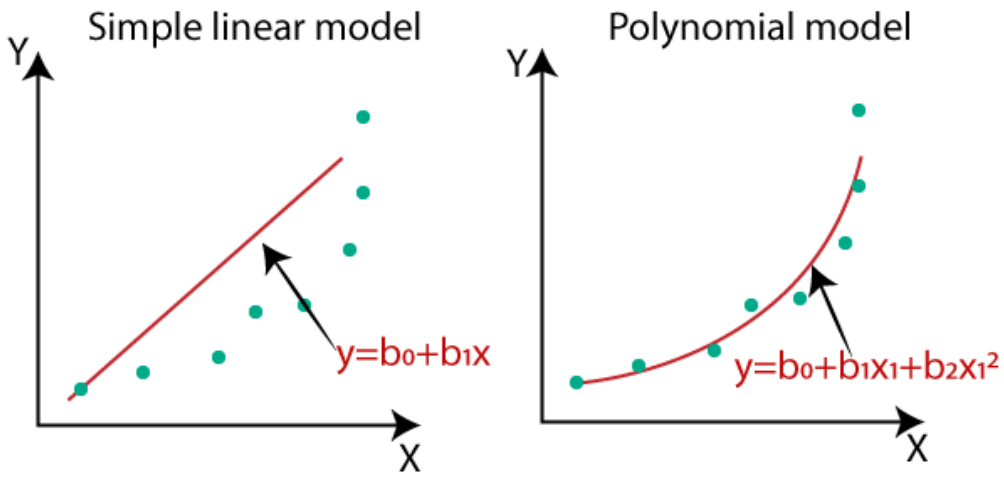
linear regression works only when the regression in the data is linear.

Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.

In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model.

In traditional linear regression, the relationship between the independent variable (usually denoted as "x") and the dependent variable ("y") is assumed to be a straight line. However, in real-world scenarios, the data might follow a more complex pattern that cannot be accurately captured by a straight line.

- If we apply a linear model on a linear dataset, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a non-linear dataset, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.
- So for such cases, where data points are arranged in a non-linear fashion, we need the Polynomial Regression model. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



Types of Polynomial Regression

- Linear, when the degree is 1.
- Quadratic, the degree of this equation is 2.
- Cubic with a degree as three continues, based on the degree used.

Equation of the Polynomial Regression Model:

Simple Linear Regression equation:

$y = b_0+b_1x$ .....(a)

Multiple Linear Regression equation:

$y= b_0+b_1x+ b_2x_2+ b_3x_3+....+ b_nx_n$ .....(b)

Polynomial Regression equation:

$y= b_0+b_1x + b_2x^2+ b_3x^3+....+ b_nx^n$ .....(c)

Predict the salary

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

```
df = pd.read_csv('Position_Salaries.csv')
df
```

```
X = df.iloc[:, 1:2].values
y = df.iloc[:, 2].values
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)
```

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly_reg = PolynomialFeatures(degree = 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 Linear Regression results
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()
```

```
# Visualising the Polynomial Regression results
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()
```

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

```
# Predicting a new result with Linear Regression
lin_reg.predict(np.array(6.5).reshape(1,-1))
```

```
# Predicting a new result with Polynomial Regression
lin_reg_2.predict(poly_reg.fit_transform(np.array(6.5).reshape(1,-1)))
```

Polynomial linear regression Banglore house price

```
import pandas as pd
df=pd.read_csv('bangalore house price prediction OHE-data.csv')
df.head()
```

split the data

```
X = df.drop('price', axis=1)
y = df['price']

print('Shape of X = ', X.shape)
print('Shape of y = ', y.shape)
```

```
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=51)

print('Shape of X_train = ', X_train.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of X_test = ', X_test.shape)
print('Shape of y_test = ', y_test.shape)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

```
Polynomial Linear Regression - ML Model Training
```

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

```
poly_reg = PolynomialFeatures(degree=2)
poly_reg.fit(X_train)
X_train_poly = poly_reg.transform(X_train)
X_test_poly = poly_reg.transform(X_test)
```

```
X_train_poly.shape, X_test_poly.shape
```

```
lr = LinearRegression()

lr.fit(X_train_poly, y_train)
```

```
lr.score(X_test_poly, y_test,)
```

```
lr.predict([X_test_poly[0,:]])
```

```
y_pred = lr.predict(X_test_poly)
y_pred
```

```
y_test
```

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

mse, rmse
```