Polynomial-Regression

January 14, 2025

[ ]:
```
'''
    Polynomial Regression -->

    Polynomial regression is a form of regression analysis that models the␣
 ↪relationship between the independent variable
    x and the dependent variable y. y as an n-th degree polynomial. It is␣
 ↪useful when data shows a curvilinear relationship
    rather than a straight line, as linear regression does.
'''
```

[ ]:
```
'''
    Polynomial Regression -->

    y = 0 + 1x + 2x^2 + 3x^3 +  + nx^n +

    Where -->

    y is the dependent variable (target)
    x is the independent variable (input)
     0, 1, …, n are the coefficients to be learned
    n is the degree of the polynomial
      is the error term
'''
```

[21]:
```
#   Importing Libraries -->

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

[22]:
```
#   Importing Dataset -->

data = pd.read_csv('Data/Position_Salaries.csv')
data
```

[22]:

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

[23]:
```
x_data = data.iloc[:, 1: -1].values
y_data = data.iloc[:, -1].values
```

[24]:
```
#   Building Linear Model -->

linear_model = LinearRegression()
linear_model.fit(x_data, y_data)
```

[24]:
```
LinearRegression()
```

[34]:
```
#   Applying Polynomial Features To Data -->

polynomial_model = PolynomialFeatures(degree = 4)
x_poly = polynomial_model.fit_transform(x_data)
x_poly
```

[34]:
```
array([[1.000e+00, 1.000e+00, 1.000e+00, 1.000e+00, 1.000e+00],
       [1.000e+00, 2.000e+00, 4.000e+00, 8.000e+00, 1.600e+01],
       [1.000e+00, 3.000e+00, 9.000e+00, 2.700e+01, 8.100e+01],
       [1.000e+00, 4.000e+00, 1.600e+01, 6.400e+01, 2.560e+02],
       [1.000e+00, 5.000e+00, 2.500e+01, 1.250e+02, 6.250e+02],
       [1.000e+00, 6.000e+00, 3.600e+01, 2.160e+02, 1.296e+03],
       [1.000e+00, 7.000e+00, 4.900e+01, 3.430e+02, 2.401e+03],
       [1.000e+00, 8.000e+00, 6.400e+01, 5.120e+02, 4.096e+03],
       [1.000e+00, 9.000e+00, 8.100e+01, 7.290e+02, 6.561e+03],
       [1.000e+00, 1.000e+01, 1.000e+02, 1.000e+03, 1.000e+04]])
```

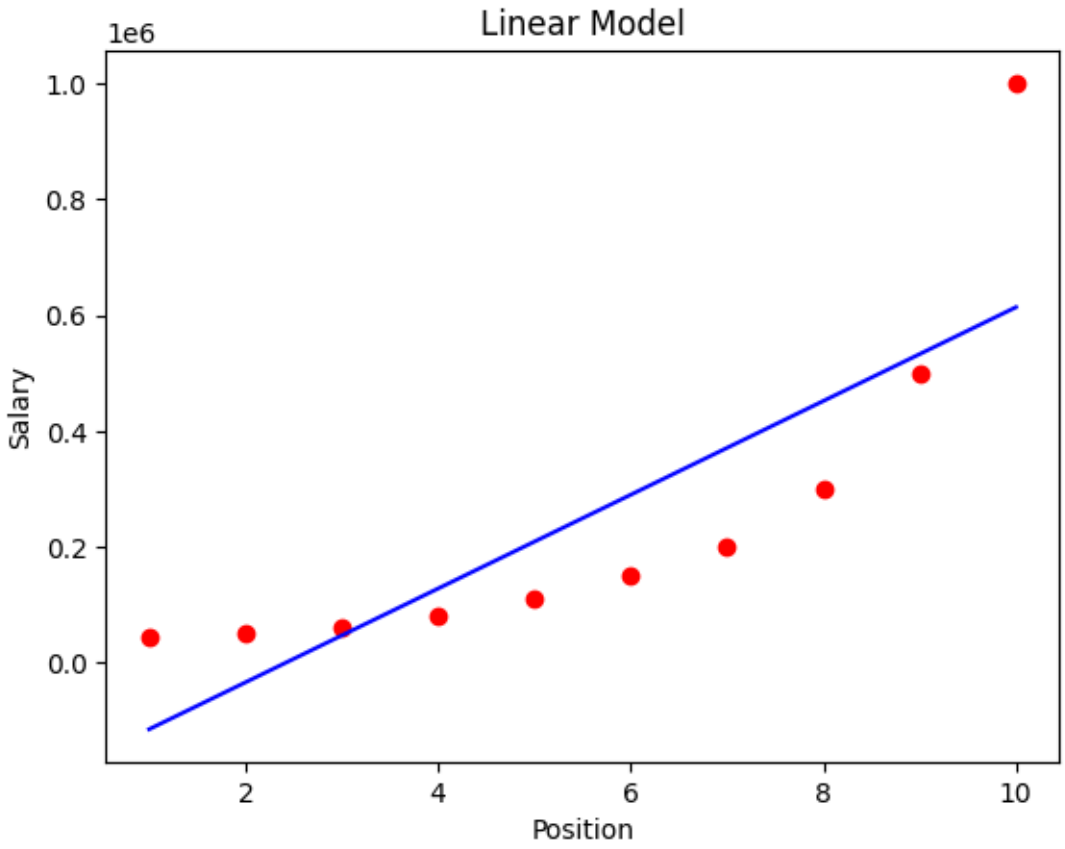
[35]:
```
#   Building Polynomial Model -->

model = LinearRegression()
model.fit(x_poly, y_data)
```

[35]:
```
LinearRegression()
```

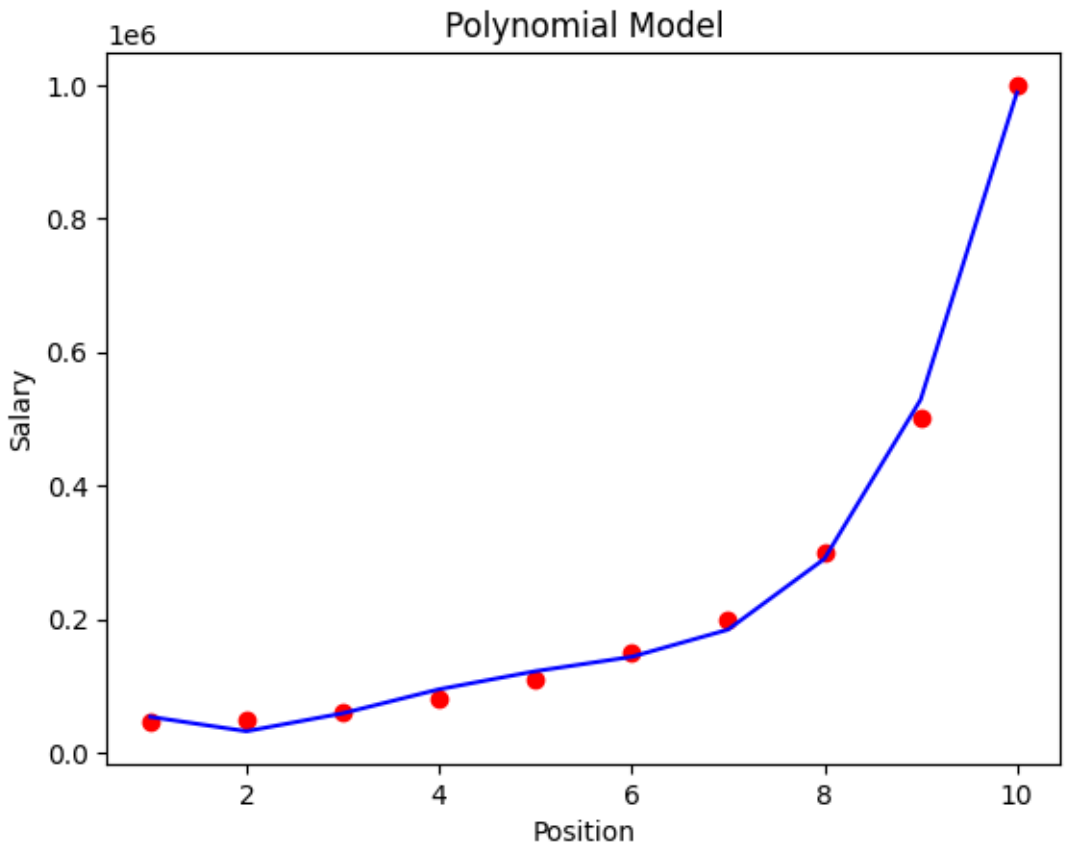
[36]:
```
#   Visualizing Linear Model -->

plt.scatter(x_data, y_data, color = 'red')
plt.plot(x_data, linear_model.predict(x_data), color = 'blue')
plt.title("Linear Model")
plt.xlabel("Position")
plt.ylabel("Salary")
plt.show()
```



[37]:
```
#   Visualizing Polynomial Model -->

plt.scatter(x_data, y_data, color = 'red')
plt.plot(x_data, model.predict(x_poly), color = 'blue')
plt.title("Polynomial Model")
plt.xlabel("Position")
plt.ylabel("Salary")
plt.show()
```



[ ]:
```
#   Predicting Salary with Linear Model -->

linear_model.predict([[6.5]])
```

[ ]:
```
array([330378.78787879])
```

[41]:
```
#   Linear Model has very high Error Lets check with Polynomial Model -->

model.predict(polynomial_model.fit_transform([[6.5]]))
```

[41]:
```
array([158862.45265155])
```

[ ]:
```
'''
   Summary -->

   Polynomial regression is linear because it is linear in the coefficients,␣
 ↪even though it involves polynomial features
   Degree 4 is chosen when the data shows more complex relationships than can␣
 ↪be captured by a lower degree
   (like degree 2 or 3), but care should be taken to avoid overfitting
```

```
  Data transformation (e.g., creating polynomial features like x^2, x^3)␣
↪allowing a linear regression model
  to handle more complex patterns in the data
'''
```