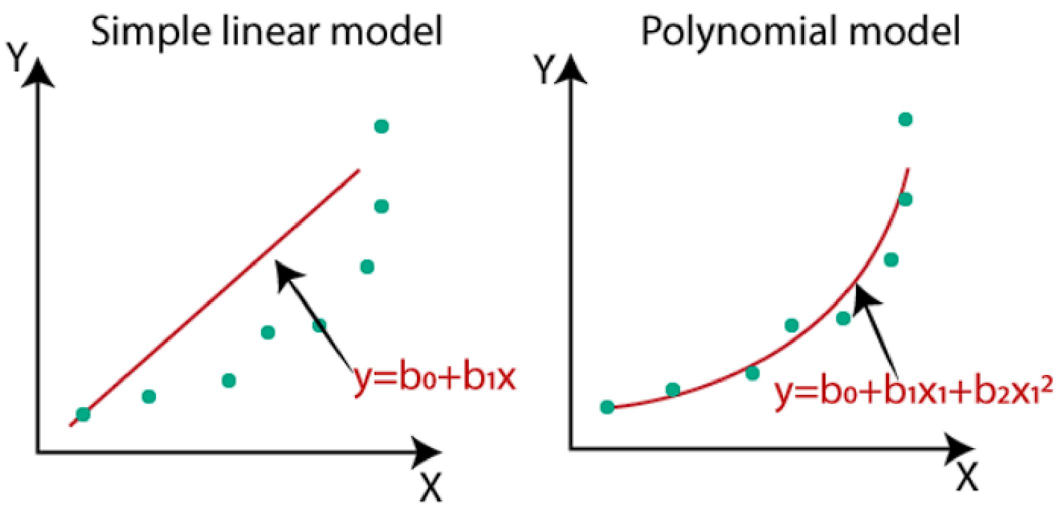
Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.

In [ ]:
```
# y= b0+b1x1+ b2x12+ b2x13+...... bnx1n

"""It is a linear model with some modification in order to increase the accu

The dataset used in Polynomial regression for training is of non-linear natu

It makes use of a linear regression model to fit the complicated and non-lin

```

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

In [ ]:
```
"""Need for Polynomial Regression:

If we apply a linear model on a linear dataset, then it provides us a good r
but if we apply the same model without any modification on a non-linear data
Due to which loss function will increase, the error rate will be high, and a

So for such cases, where data points are arranged in a non-linear fashion, w
We can understand it in a better way using the below comparison diagram of t
```



In [ ]:
```
"""Equation of the Polynomial Regression Model:


Simple Linear Regression equation:         y = b0+b1x         .........(a)

Multiple Linear Regression equation:         y= b0+b1x+ b2x2+ b3x3+....+ bnx

Polynomial Regression equation:         y= b0+b1x + b2x2+ b3x3+....+ bnxn
```

Implementation of Polynomial Regression using Python:

In [ ]:

```
#Problem Description:
"""There is a Human Resource company, which is going to hire a new candidate
The candidate has told his previous salary 160K per annum, and the HR have t
check whether he is telling the truth or bluff.
So to identify this, they only have a dataset of his previous company in whi
the salaries of the top 10 positions are mentioned with their levels.
By checking the dataset available, we have found that there is a non-linear
relationship between the Position levels and the salaries.
Our goal is to build a Bluffing detector regression model,
so HR can hire an honest candidate. Below are the steps to build such a mode
```

In [ ]:

```
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

In [ ]:

```
df=pd.read_csv("/content/sample_data/Position_Salaries_Polynomial_DataSet.cs
```

In [ ]:

```
df.head(5)
```

Out[8]:

	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

In [ ]:

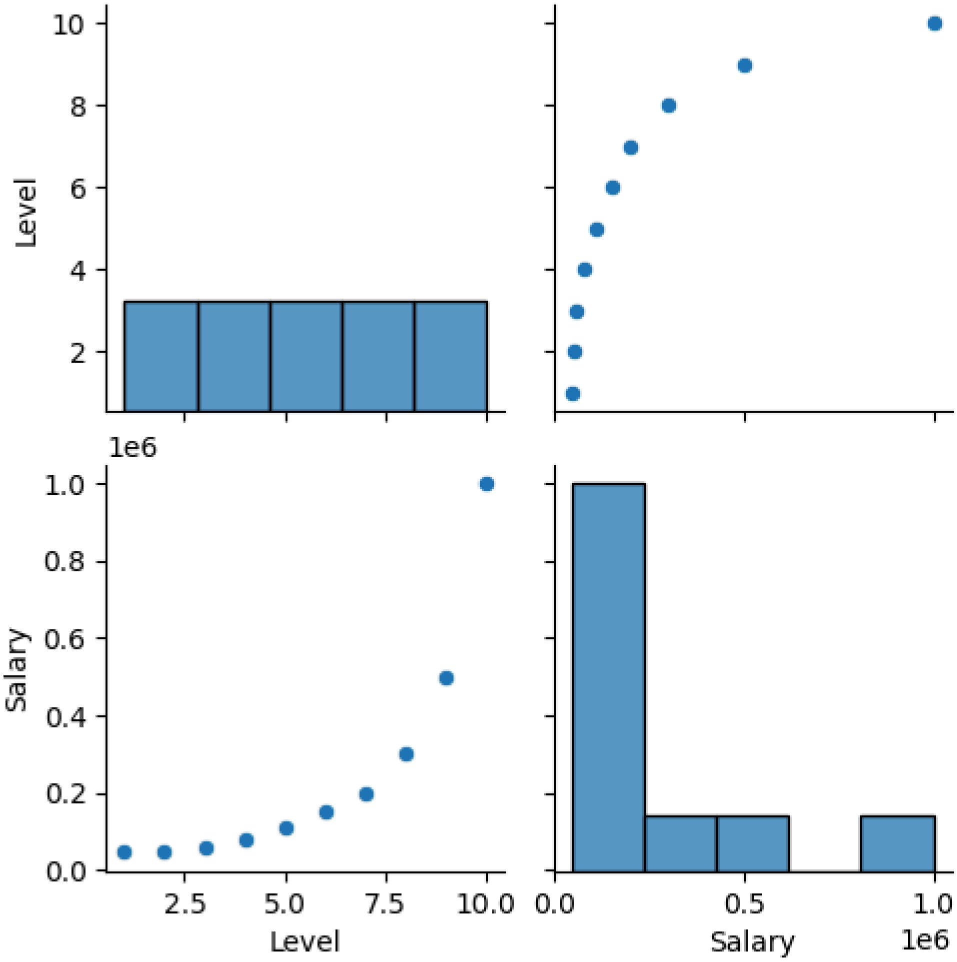
```
df
```

Out[9]:

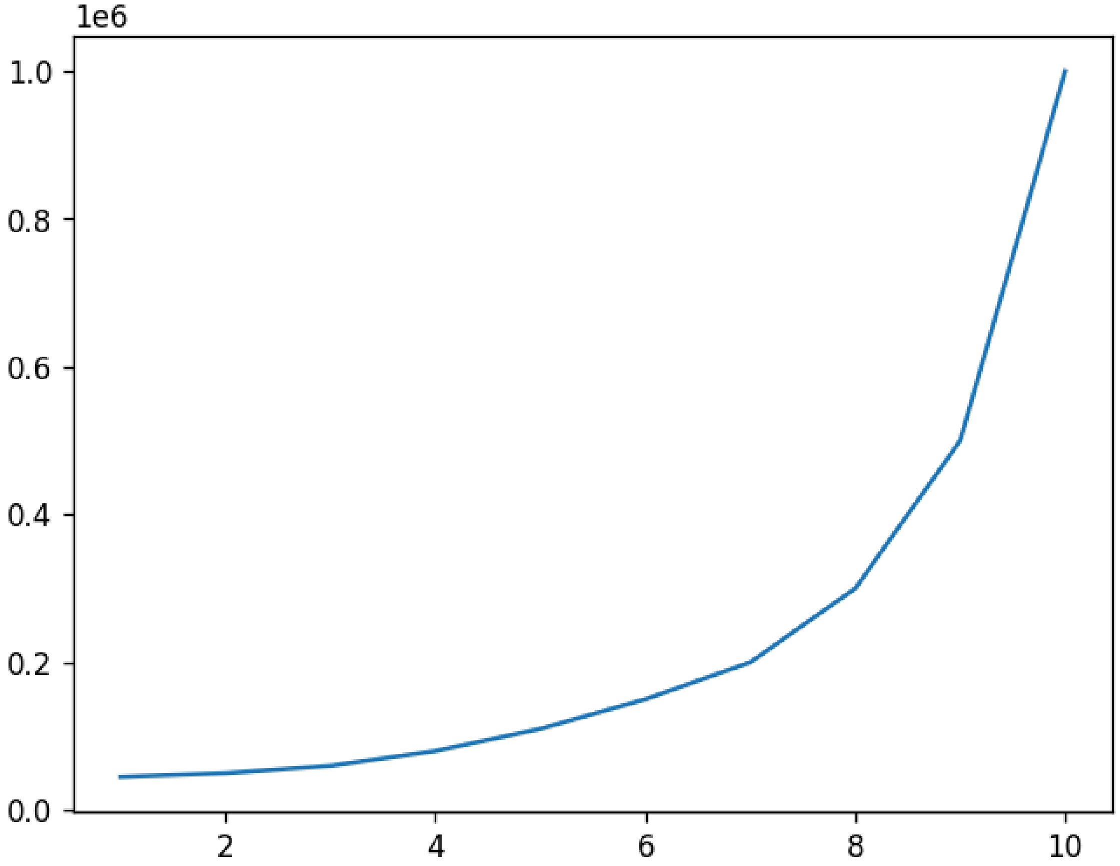
	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

In [ ]:
```
sns.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x7bfeb8ea0b50>
```



```
In [ ]: plt.figure(dpi=120)
        plt.plot(df["Level"],df["Salary"])

Out[11]: [<matplotlib.lines.Line2D at 0x7bfeb6afcfa0>]
```



```
In [ ]: x=df[["Level"]].values
```

```
In [ ]: y=df[["Salary"]].values
```

```
In [ ]: from sklearn.linear_model import LinearRegression
```

```
In [ ]: reg=LinearRegression()
```

```
In [ ]: reg.fit(x,y)

Out[16]: LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [ ]: reg.score(x,y)

Out[17]: 0.6690412331929895
```

In [ ]:
```
pred=reg.predict(x)
```

In [ ]:
```
pred
```

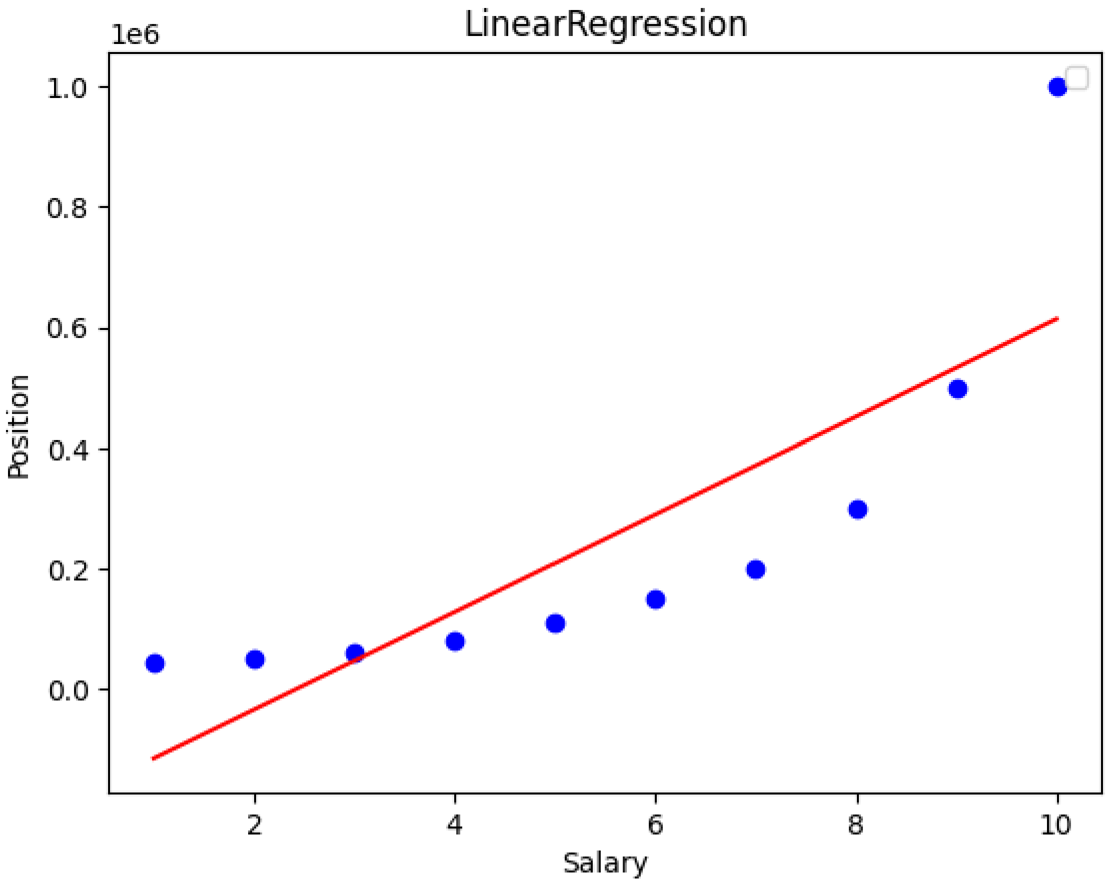
Out[19]:
```
array([[-114454.54545455],
       [ -33575.75757576],
       [  47303.03030303],
       [ 128181.81818182],
       [ 209060.60606061],
       [ 289939.39393939],
       [ 370818.18181818],
       [ 451696.96969697],
       [ 532575.75757576],
       [ 613454.54545455]])
```

In [ ]:
```
#Now find the best fit line
```

In [ ]:
```
plt.figure(dpi=100)
plt.title("LinearRegression")
plt.xlabel("Salary")
plt.ylabel("Position")
plt.plot(x,y,"ob")
plt.plot(x,pred,"red") #best fit Line
plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.
Note that artists whose label start with an underscore are ignored when le
gend() is called with no argument.
```

Out[21]:
```
<matplotlib.legend.Legend at 0x7bfeb23eccd0>
```



In [ ]:
```
#introducing the variable with degree =2 (PLR - Polynomial Linear Regressio
#we will build the Polynomial Regression model, but it will be a little dif
```

In [ ]:
```
from sklearn.preprocessing import PolynomialFeatures
poly=PolynomialFeatures(degree=2)
x_poly=poly.fit_transform(x)
poly
```

Out[22]:
```
PolynomialFeatures()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
x_poly
```

Out[23]:
```
array([[  1.,   1.,   1.],
       [  1.,   2.,   4.],
       [  1.,   3.,   9.],
       [  1.,   4.,  16.],
       [  1.,   5.,  25.],
       [  1.,   6.,  36.],
       [  1.,   7.,  49.],
       [  1.,   8.,  64.],
       [  1.,   9.,  81.],
       [  1.,  10., 100.]])
```

In [ ]:
```
poly_reg2=LinearRegression()
poly_reg2.fit(x_poly,y)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
poly_reg2.score(x_poly,y)
```

Out[25]:
```
0.9162082221443942
```

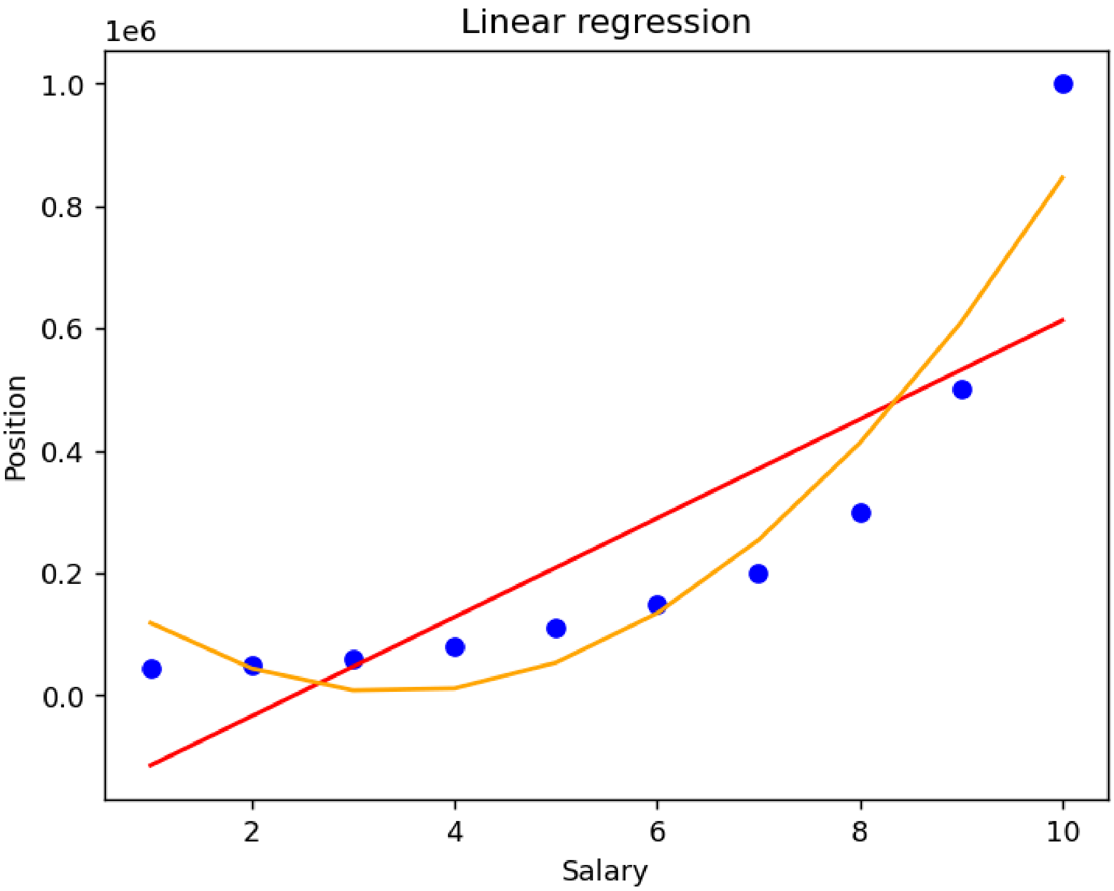
In [ ]:
```
pred2=poly_reg2.predict(x_poly)
```

In [ ]:
```
pred2
```

Out[27]:
```
array([[118727.27272727],
       [ 44151.51515151],
       [  8439.39393939],
       [ 11590.90909091],
       [ 53606.06060606],
       [134484.84848485],
       [254227.27272727],
       [412833.33333333],
       [610303.03030303],
       [846636.36363636]])
```

```
In [ ]: plt.figure(dpi=130)
        plt.title('Linear regression')
        plt.xlabel('Salary')
        plt.ylabel('Position')
        plt.plot(x,y,'ob');
        plt.plot(x,pred,'red');
        plt.plot(x,pred2,'orange')
```

```
Out[28]: [<matplotlib.lines.Line2D at 0x7bfeb1e18880>]
```



```
In [ ]: #Use PolynomialFeatures class of preprocessing.
```

```
In [ ]: from sklearn.preprocessing import PolynomialFeatures
```

```
In [ ]: poly=PolynomialFeatures(degree=3)
        x_poly=poly.fit_transform(x)
        poly
```

```
Out[31]: PolynomialFeatures(degree=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
x_poly
```

Out[32]:
```
array([[   1.,    1.,    1.,    1.],
       [   1.,    2.,    4.,    8.],
       [   1.,    3.,    9.,   27.],
       [   1.,    4.,   16.,   64.],
       [   1.,    5.,   25.,  125.],
       [   1.,    6.,   36.,  216.],
       [   1.,    7.,   49.,  343.],
       [   1.,    8.,   64.,  512.],
       [   1.,    9.,   81.,  729.],
       [   1.,   10.,  100., 1000.]])
```

In [ ]:
```
poly_reg3=LinearRegression()
poly_reg3.fit(x_poly,y)
```

Out[33]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
poly_reg3
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
poly_reg3.score(x_poly,y)
```

Out[35]:
```
0.9812097727913366
```

In [ ]:
```
pred3=poly_reg3.predict(x_poly)
```

In [ ]:
```
pred3
```

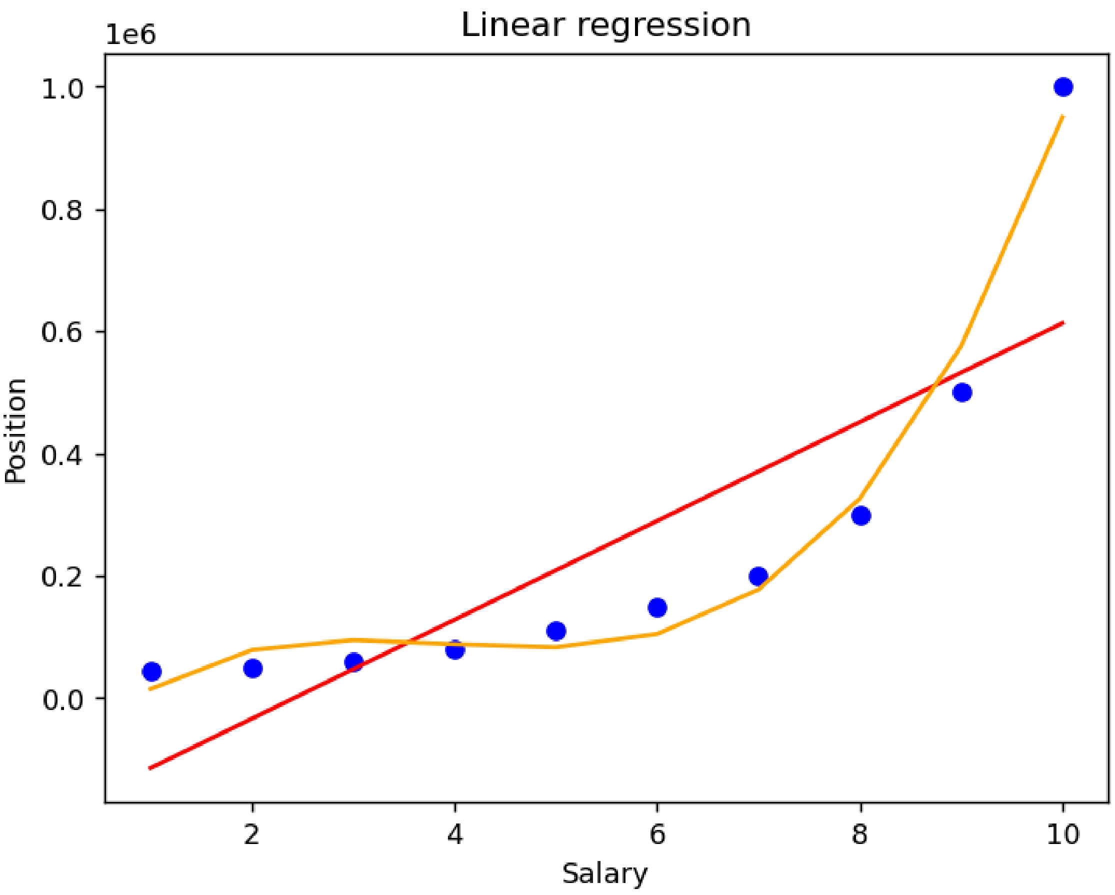
Out[37]:
```
array([[ 14902.09790211],
       [ 78759.90675991],
       [ 94960.37296037],
       [ 88223.77622378],
       [ 83270.3962704 ],
       [104820.51282052],
       [177594.40559441],
       [326312.35431236],
       [575694.63869463],
       [950461.53846152]])
```

In [ ]:

```
plt.figure(dpi=130)
plt.title('Linear regression')
plt.xlabel('Salary')
plt.ylabel('Position')
plt.plot(x,y,'ob');
plt.plot(x,pred,'red');
plt.plot(x,pred3,'orange')
```

Out[38]:

```
[<matplotlib.lines.Line2D at 0x7bfeb1ebdc90>]
```



In [ ]:

```
from sklearn.preprocessing import PolynomialFeatures
poly=PolynomialFeatures(degree=9)
x_poly=poly.fit_transform(x)
poly
```

Out[39]:

```
PolynomialFeatures(degree=9)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:

```
poly_reg9=LinearRegression()
poly_reg9.fit(x_poly,y)
```

Out[40]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [ ]: poly_reg9.score(x_poly,y)
```

```
Out[41]: 0.9999999999999828
```

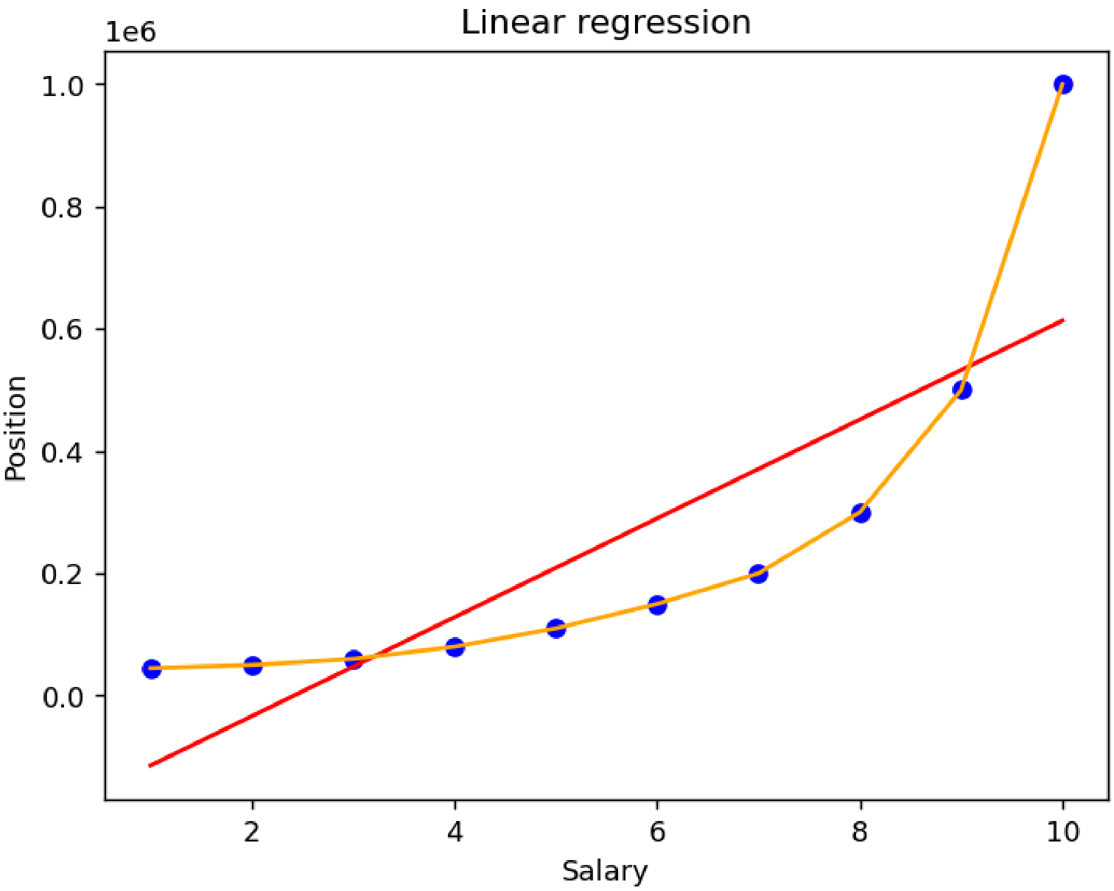
```
In [ ]: pred9=poly_reg9.predict(x_poly)
```

```
In [ ]: pred9
```

```
Out[43]: array([[  44999.95524136],
                [  50000.00570684],
                [  60000.01207874],
                [  79999.99142668],
                [ 110000.02227248],
                [ 149999.99920151],
                [ 200000.01396641],
                [ 300000.00429198],
                [ 499999.92400357],
                [1000000.07181498]])
```

```
In [ ]: plt.figure(dpi=130)
        plt.title('Linear regression')
        plt.xlabel('Salary')
        plt.ylabel('Position')
        plt.plot(x,y,'ob');
        plt.plot(x,pred,'red');
        plt.plot(x,pred9,'orange')
```

```
Out[44]: [<matplotlib.lines.Line2D at 0x7bfeafd02fb0>]
```



```
In [ ]:
```