```
In [51]: import pandas as pd
         import numpy as np
         from skmisc.loess import loess
         from sklearn.linear_model import LinearRegression as l
         import math
         import statsmodels.api as sm
```

```
In [52]: Z=[1,0,1,4,3,2,5,6,9,13,15,16]
         z=np.array(Z)
         z

Out[52]: array([ 1,  0,  1,  4,  3,  2,  5,  6,  9, 13, 15, 16])
```

```
In [53]: X=[[1,1],
         [2,1],
         [2,2],
         [3,2],
         [5,4],
         [5,6],
         [6,5],
         [7,4],
         [10,8],
         [11,7],
         [11,9],
         [12,10]]
         x=np.array(X)
         x

Out[53]: array([[ 1,  1],
                [ 2,  1],
                [ 2,  2],
                [ 3,  2],
                [ 5,  4],
                [ 5,  6],
                [ 6,  5],
                [ 7,  4],
                [10,  8],
                [11,  7],
                [11,  9],
                [12, 10]])
```

LOESS

```
In [54]: lm=loess(X,Z,degree=2,span=0.9)
         lm.fit()
         SZ=lm.predict(X)
         sz=SZ.values
         sz

Out[54]: array([ 0.15816794,  1.26175196,  4.76667098,  3.53781019,  3.2108452 ,
                 2.32615925,  6.7806428 ,  4.76667098, 11.30364892, 12.66442148,
                12.86530323, 16.48432032])
```

In [55]:

```
residual=z-sz
residual
```

Out[55]:

```
array([ 0.84183206, -1.26175196, -3.76667098,  0.46218981, -0.2108452 ,
       -0.32615925, -1.7806428 ,  1.23332902, -2.30364892,  0.33557852,
        2.13469677, -0.48432032])
```

In [56]:

```
w = 1 / math.e**(residual)
w
```

Out[56]:

```
array([ 0.43092033,  3.5316033 , 43.23589194,  0.62990276,  1.2347212 ,
        1.38563601,  5.93366933,  0.29132115, 10.01064388,  0.71492437,
        0.11828045,  1.62307146])
```

Weighted Linear Regression

In [57]:

```
lr=l()
lr.fit(X,Z,w)
rw=lr.score(X,Z,w)
print(f'R2 of WLS: {rw}')
y_w=lr.predict(X)
y_w=pd.DataFrame(y_w)
prw=y_w.describe()
```

```
R2 of WLS: 0.9450943174557025
```

Linear Regression

In [58]:

```
#R2 Value
lr=l()
lr.fit(X,Z)
ro=lr.score(X,Z)
print(f'R2 of OLS: {ro}')
y_o=lr.predict(X)
y_o=pd.DataFrame(y_o)
pro=y_o.describe()
```

```
R2 of OLS: 0.903737802561535
```

In [59]:

```
#Standard Error
stdw=prw.loc['std']
ew=stdw/len(y_w)**(1/2)
print(f'Standard Error of WLS: {ew}')
stdo=pro.loc['std']
eo=stdo/len(y_w)**(1/2)
print(f'Standard Error of OLS: {eo}')
```

```
Standard Error of WLS: 0    1.270393
Name: std, dtype: float64
Standard Error of OLS: 0    1.557361
Name: std, dtype: float64
```

In [60]:

```
#adjusted R-square
n=len(z)
print(f'adjusted R-square of WLS: {1-(1-rw)*(n-1)/(n-2-1)}')
print(f'adjusted R-square of OLS: {1-(1-ro)*(n-1)/(n-2-1)}')
```

```
adjusted R-square of WLS: 0.9328930546680808
adjusted R-square of OLS: 0.8823462031307651
```

In [61]:

```
model = sm.OLS(z, x)
results = model.fit()
print(f'p-values of OLS:{results.pvalues}')
```

```
p-values of OLS:[0.02866399 0.59300176]
```

In [62]:

```
model = sm.WLS(z, x,w)
results = model.fit()
print(f'p-values of WLS:{results.pvalues}')
```

```
p-values of WLS:[0.01228962 0.15391743]
```

In [63]:

```
d=np.transpose(X)
d
```

Out[63]:

```
array([[ 1,  2,  2,  3,  5,  5,  6,  7, 10, 11, 11, 12],
       [ 1,  1,  2,  2,  4,  6,  5,  4,  8,  7,  9, 10]])
```

Covariance Matrix

In [65]:

```
cov_matrix = np.cov(d)
cov_matrix
```

Out[65]:

```
array([[15.47727273, 11.65909091],
       [11.65909091,  9.71969697]])
```