IMPORTING THE LIBRARIES

In [25]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

READ THE DATASET

In [6]:
```
data=pd.read_csv('Salary_Data.csv')
```

DATA ANALYSIS

In [7]:
```
data.head()
```

Out[7]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [10]:
```
data.tail()
```

Out[10]:

	YearsExperience	Salary
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [11]:
```
data.columns
```

Out[11]:
```
Index(['YearsExperience', 'Salary'], dtype='object')
```

In [12]:
```
data.describe()
```

Out[12]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```

In [14]:
```
data.shape
```

Out[14]:
```
(30, 2)
```

IDENTIFYING MISSING DATA

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
YearsExperience    0
Salary             0
dtype: int64
```

SPLITTING THE DATA

In [16]:
```
x=data.iloc[:, :-1].values
```

In [17]:
```
x
```

Out[17]:
```
array([[ 1.1],
       [ 1.3],
       [ 1.5],
       [ 2. ],
       [ 2.2],
       [ 2.9],
       [ 3. ],
       [ 3.2],
       [ 3.2],
       [ 3.7],
       [ 3.9],
       [ 4. ],
       [ 4. ],
       [ 4.1],
       [ 4.5],
       [ 4.9],
       [ 5.1],
       [ 5.3],
       [ 5.9],
       [ 6. ],
       [ 6.8],
       [ 7.1],
       [ 7.9],
       [ 8.2],
       [ 8.7],
       [ 9. ],
       [ 9.5],
       [ 9.6],
       [10.3],
       [10.5]])
```

In [18]:
```
y=data['Salary'].values
```

In [19]:
```
y
```

Out[19]:
```
array([ 39343.,  46205.,  37731.,  43525.,  39891.,  56642.,  60150.,
        54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57081.,
        61111.,  67938.,  66029.,  83088.,  81363.,  93940.,  91738.,
        98273., 101302., 113812., 109431., 105582., 116969., 112635.,
       122391., 121872.])
```

In [23]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

LINEAR REGRESSION

In [27]:
```
reg1=LinearRegression()
```

In [37]:
```
reg1.fit(x_train,y_train)
```

Out[37]:
```
LinearRegression()
```

In [29]:
```
y_pred=reg1.predict(x_test)
```

In [30]:
```
y_pred
```

Out[30]:
```
array([ 40817.78327049, 123188.08258899,  65154.46261459,  63282.41035735,
       115699.87356004, 108211.66453108, 116635.89968866,  64218.43648597,
        76386.77615802])
```

In [31]:
```
y_test
```

Out[31]:
```
array([ 37731., 122391.,  57081.,  63218., 116969., 109431., 112635.,
        55794.,  83088.])
```

In [32]:
```
from sklearn.metrics import *
```

In [33]:
```
r_sq=r2_score(y_test,y_pred)
```

In [34]:
```
r_sq
```

Out[34]:
```
0.9740993407213511
```

In [35]:
```
mse=mean_squared_error(y_test,y_pred)
```

In [36]:
```
mse
```

Out[36]:
```
23370078.800832972
```

DECISION TREE REGRESSION

In [38]:
```
from sklearn.tree import *
reg1=DecisionTreeRegressor(random_state=1)
```

In [39]:
```
reg1.fit(x_train,y_train)
```

Out[39]:
```
DecisionTreeRegressor(random_state=1)
```

In [40]:
```
y_pred=reg1.predict(x_test)
```

In [41]:
```
y_pred
```

Out[41]:
```
array([ 46205., 121872.,  56957.,  56957., 105582., 105582., 105582.,
        56957.,  66029.])
```

In [42]:
```
y_test
```

Out[42]:
```
array([ 37731., 122391.,  57081.,  63218., 116969., 109431., 112635.,
        55794.,  83088.])
```

In [43]:
```
r_sq=r2_score(y_test,y_pred)
r_sq
```

Out[43]:
```
0.9263756616003317
```

In [44]:
```
mse=mean_squared_error(y_test,y_pred)
mse
```

Out[44]:
```
66430995.88888889
```

R-SQUARED RESULT

LINEAR REGRESSION - 97.41 %

DECISION TREE REGRESSION - 92.64 %