MACHINE LEARNING MAJOR PROJECT

Problem Statement :-

Take any Dataset of your choice, perform EDA (Exploratory Data Analysis) and apply a suitable Classifier, Regressor or Clusterer and calculate the accuracy of the model.

I have chosen the 'Salary data' dataset and I will be performing Linear regression on it.
Link of the dataset - https://github.com/ameenmanna8824/DATASETS/blob/main/Salary_Data.csv
Raw - https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Salary_Data.csv

---

Creating dataframe

In [53]:
```
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Salary_Data.csv')
df
```

Out[53]:

	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
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0

	YearsExperience	Salary
24	8.7	109431.0
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

Exploratory Data Analysis (EDA)

In [54]:
```
df.shape
```

Out[54]:
```
(30, 2)
```

In [55]:
```
df.YearsExperience.nunique()
```

Out[55]:
```
28
```

In [56]:
```
df.Salary.nunique()
```

Out[56]:
```
30
```

In [57]:
```
df.Salary.unique()
```

Out[57]:
```
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 [58]:
```
df.YearsExperience.unique()
```

Out[58]:
```
array([ 1.1,  1.3,  1.5,  2. ,  2.2,  2.9,  3. ,  3.2,  3.7,  3.9,  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])
```

Salary analysis

In [59]:
```
# Salary above 40k

df['Salary above 40k'] = df['Salary'] > 40000
df.groupby('Salary above 40k').size()
```

Out[59]:
```
Salary above 40k
False     3
True     27
dtype: int64
```

In [60]:
```
# Salary above 70k

df['Salary above 70k'] = df['Salary'] > 70000
df.groupby('Salary above 70k').size()
```

Out[60]:
```
Salary above 70k
False    17
True     13
dtype: int64
```

In [62]:
```
# Salary above 100k

df['Salary above 100k'] = df['Salary'] > 100000
df.groupby('Salary above 100k').size()
```

Out[62]:
```
Salary above 100k
False    22
True      8
dtype: int64
```

From the above cell blocks, we can deduce that

- Employees with salary above 40,000 : 27 / 30

- Employees with salary above 70,000 : 13 / 30
- Employees with salary above 100,000 : 8 / 30

Years of exp. analysis

In [63]:
```
# Years of Experience above 2

df['Year of exp. > 2'] = df['YearsExperience'] > 2
df.groupby('Year of exp. > 2').size()
```

Out[63]:
```
Year of exp. > 2
False     4
True     26
dtype: int64
```

In [64]:
```
# Years of Experience above 5

df['Year of exp. > 5'] = df['YearsExperience'] > 5
df.groupby('Year of exp. > 5').size()
```

Out[64]:
```
Year of exp. > 5
False    16
True     14
dtype: int64
```

In [65]:
```
# Years of Experience above 8

df['Year of exp. > 8'] = df['YearsExperience'] > 8
df.groupby('Year of exp. > 8').size()
```

Out[65]:
```
Year of exp. > 8
False    23
True      7
dtype: int64
```

In [66]:
```
# Years of Experience above 10

df['Year of exp. > 10'] = df['YearsExperience'] > 10
df.groupby('Year of exp. > 10').size()
```

Out[66]:
```
Year of exp. > 10
False    28
True      2
dtype: int64
```

From the above cell blocks, we can deduce that

- Employees with years of exp. above 2 : 26 / 30
- Employees with years of exp. above 5 : 14 / 30
- Employees with years of exp. above 8 : 7 / 30
- Employees with years of exp. above 10 : 2 / 30

Min. and Max. years of experience and salary

In [67]:
```
df.YearsExperience.min()
```

Out[67]:
```
1.1
```

In [68]:
```
df.YearsExperience.max()
```

Out[68]:
```
10.5
```

In [69]:
```
df.Salary.min()
```

Out[69]:
```
37731.0
```

In [70]:
```
df.Salary.max()
```

Out[70]:
```
122391.0
```

From the above cells, we can deduce that

- Min. Years of exp. : 1.1 yrs
- Max. Years of exp. : 10.5 yrs

- Min. Salary : 37731.0
- Max. Salary : 122391.0

Data Visualisation

In [71]:
```
import matplotlib.pyplot as plt
plt.scatter(df['YearsExperience'],df['Salary'])
plt.title("YEARS OF EXPERIENCE VS SALARY")
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```

Out[71]:
```
Text(0, 0.5, 'Salary')
```

Dividing data into input and output

In [72]:
```
x = df.iloc[:,0]
x
```

Out[72]:
```
0      1.1
1      1.3
2      1.5
3      2.0
4      2.2
5      2.9
6      3.0
7      3.2
8      3.2
9      3.7
10     3.9
11     4.0
12     4.0
13     4.1
14     4.5
15     4.9
16     5.1
17     5.3
18     5.9
19     6.0
20     6.8
21     7.1
22     7.9
23     8.2
24     8.7
25     9.0
26     9.5
27     9.6
28    10.3
29    10.5
Name: YearsExperience, dtype: float64
```

In [73]:
```
x = df.iloc[:29,:1].values
x
```

Out[73]:
```
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]])
```

In [74]:
```
y = df.iloc[:,1]
y
```

Out[74]:
```
0      39343.0
1      46205.0
2      37731.0
3      43525.0
4      39891.0
5      56642.0
6      60150.0
7      54445.0
8      64445.0
9      57189.0
10     63218.0
11     55794.0
12     56957.0
13     57081.0
14     61111.0
15     67938.0
16     66029.0
17     83088.0
18     81363.0
19     93940.0
20     91738.0
21     98273.0
22    101302.0
23    113812.0
24    109431.0
25    105582.0
26    116969.0
27    112635.0
28    122391.0
29    121872.0
Name: Salary, dtype: float64
```

In [75]:
```
y = df.iloc[1:30,1].values
y
```

Out[75]:
```
array([ 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.])
```

Running Regressor

In [76]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

Fitting the model

In [77]:
```
model.fit(x,y)
```

Out[77]:
```
LinearRegression()
```

Predicting the output

In [78]:
```
# Predicted Output Values

y_pred = model.predict(x)
y_pred
```

Out[78]:
```
array([ 38160.81764789,  40099.42155936,  42038.02547083,  46884.53524951,
        48823.13916098,  55608.25285114,  56577.55480687,  58516.15871835,
        58516.15871835,  63362.66849703,  65301.2724085 ,  66270.57436423,
        66270.57436423,  67239.87631997,  71117.08414292,  74994.29196586,
        76932.89587733,  78871.4997888 ,  84687.31152322,  85656.61347896,
        93411.02912485,  96318.93499206, 104073.35063794, 106981.25650515,
       111827.76628383, 114735.67215104, 119582.18192972, 120551.48388546,
       127336.59757561])
```

In [79]:
```
# Actual Output Values

y
```

Out[79]:
```
array([ 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.])
```

Conculsion

When we compare y and pred_y, we come to know that there is a huge difference. This huge difference does not mean that my model has predicted wrong. It only means that my model is NOT LINEAR / LESS LINEAR.

Individual Prediction

In [80]:
```
model.predict([[5]])
```

Out[80]:
```
array([75963.5939216])
```

Cross-verifiaction

y = mx + c

In [81]:
```
m = model.coef_
m
```

Out[81]:
```
array([9693.01955736])
```

In [82]:
```
c = model.intercept_
c
```

Out[82]:
```
27498.496134789544
```

In [83]:
```
m * 5 + c
```

Out[83]:
```
array([75963.5939216])
```

Accuracy of the model (R-squared score)

Since the answer to the above cell block is the same as the 'Individual Prediction' block, our model has predicted correct.

Furthermore, i will find the R-squared score of the model.
(The closer the R-squared score is to 1, the more accurate the model.)

In [84]:
```
from sklearn.metrics import r2_score

r2_score = r2_score(y, y_pred)
print("R-squared score:", r2_score)
```

```
R-squared score: 0.9472760886559446
```

Since the R-squared score is very close to 1, we can say that our model is fairly accurate.

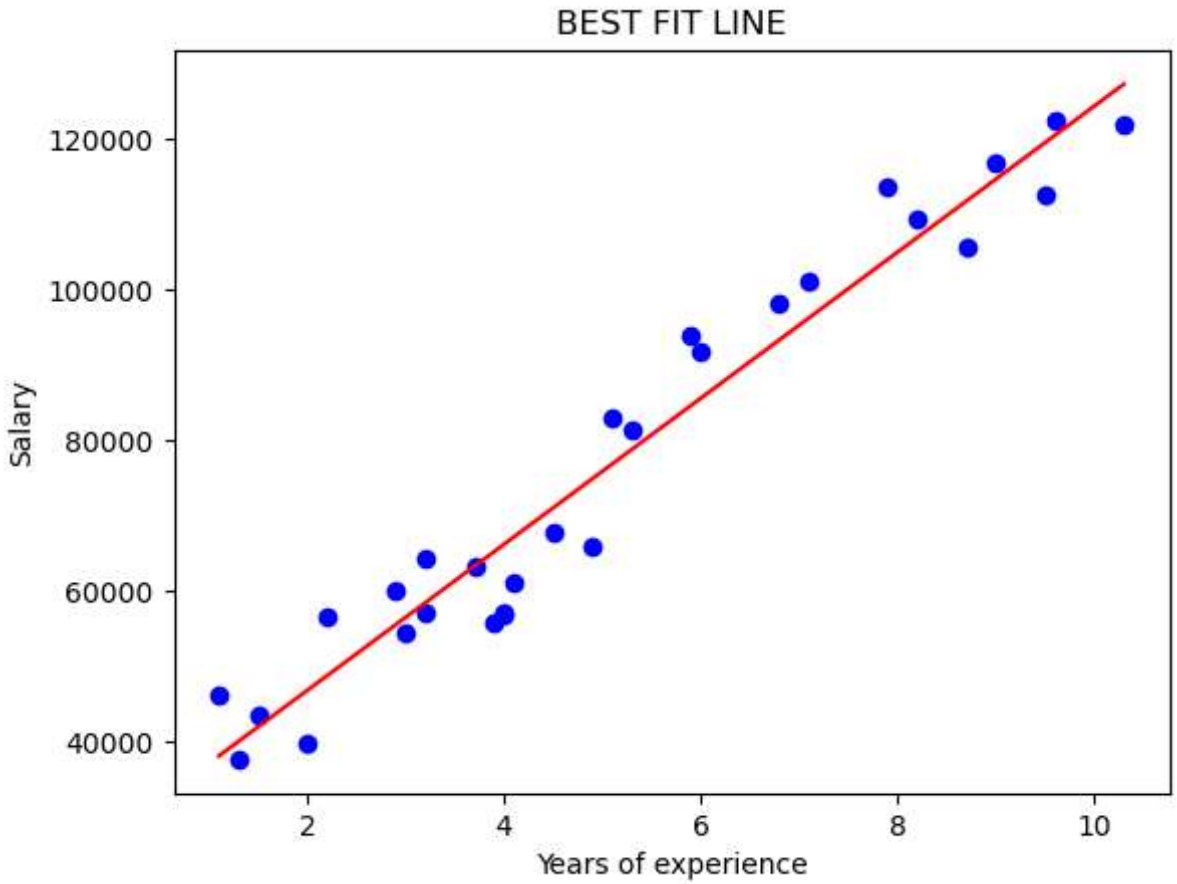
Final Visualisation

In [85]:
```
plt.scatter(x,y,c='blue')  # Actual output values
plt.plot(x,y_pred,color='red') # Predicted output values

# the scattered points will be the actual values and the line will represent the predicted values

plt.title("BEST FIT LINE")
plt.xlabel("Years of experience")
plt.ylabel("Salary")
```

Out[85]:
```
Text(0, 0.5, 'Salary')
```


BEST FIT LINE
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Years of experience