Predicting Employee Salary Based on Experience

Background:

In the corporate world, employee compensation is a crucial factor for both the employers and the employees. Determining a fair and competitive salary based on an employee's experience is important for maintaining job satisfaction, motivation, and retention. This dataset contains data on employees' years of experience and their corresponding salaries.

Objective:

The objective of this analysis is to build a predictive model that can accurately forecast an employee's salary based on their years of experience. This model will help in understanding the salary trends related to experience and assist companies in establishing fair compensation practices.

Dataset Description:

The dataset consists of the following columns:

1. Experience_Years: Number of years of experience the employee has.
2. Salary: Salary of the employee (in dollars).

Importing libraries

In [3]:
```
import pandas as pd import numpy as
np  import  seaborn  as  sns  import
matplotlib.pyplot as plt
```

What does the code aim to achieve?

Importing data

In [6]:
```
h=pd.read_csv("salary_exp.csv")
```

In [7]:
```
h
```

Out[7]:

	Experience Years	Salar y
0	1.1	39343
1	1.2	42774
2	1.3	46205
3	1.5	37731
4	2.0	43525
5	2.2	39891
6	2.5	48266
7	2.9	56642
8	3.0	60150
9	3.2	54445
10	3.2	64445
11	3.5	60000
12	3.7	57189
13	3.8	60200
14	3.9	63218
15	4.0	55794
16	4.0	56957
17	4.1	57081
18	4.3	59095
19	4.5	61111
20	4.7	64500
21	4.9	67938
22	5.1	66029
23	5.3	83088
24	5.5	82200
25	5.9	81363
26	6.0	93940
27	6.2	91000
28	6.5	90000
29	6.8	91738
30	7.1	98273
31	7.9	101302
32	8.2	113812

	Experience Years	Salar y
33	8.5	111620
34	8.7	109431
35	9.0	105582
36	9.5	116969
37	9.6	112635
38	10.3	122391
39	10.5	121872

Data columns

In [9]:
```
h.columns
```

Out[9]:
```
Index(['Experience Years', 'Salary'], dtype='object')
```

Checking shape of dataset

In [11]:
```
h.shape
```

Out[11]:
```
(40, 2)
```

We can see that our dataset consists of 40 observations and 2 rows only. These are very small dataset as it is created randomly to show how we can build a regression model.

Checking dataset Information

In [14]:
```
h.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Experience Years  40 non-null     float64
 1   Salary            40 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 772.0 bytes
```

Checking count

In [16]:
```
h.count()
```

Out[16]:
```
Experience Years    40
Salary              40
dtype: int64
```

We observed that there are no missing values are present in our dataset.

Why is linear regression chosen for this scenario?

Model Building

Splitting of dataset into Independent and Dependent variables.

In [28]:
```
x=h.iloc[:,:-1].values
y=h.iloc[:,1].values
```

Independent variable

A variable (often denoted by x) whose variation does not depend on that of another.

In [31]:
```
x
```

Out[31]:
```
array([[ 1.1],
       [  1.2],
       [  1.3],
       [  1.5],
       [  2. ],
       [  2.2],
       [  2.5],
       [  2.9],
       [  3. ],
       [  3.2],
       [  3.2],
       [  3.5],
       [  3.7],
       [  3.8],
       [  3.9],
       [  4. ],
       [  4. ],
       [  4.1],
       [  4.3],
       [  4.5],
       [  4.7],
       [  4.9],
       [  5.1],
       [  5.3],
       [  5.5],
       [  5.9],
       [  6. ],
       [  6.2],
       [  6.5],
       [  6.8],
       [  7.1],
       [  7.9],
       [  8.2],
       [  8.5],
       [  8.7],
       [  9. ],
       [  9.5],
       [  9.6],
       [10.3],
       [10.5]])
```

Dependent variable

A variable (often denoted by y) whose value depends on that of another.

In [34]:
```
y
```

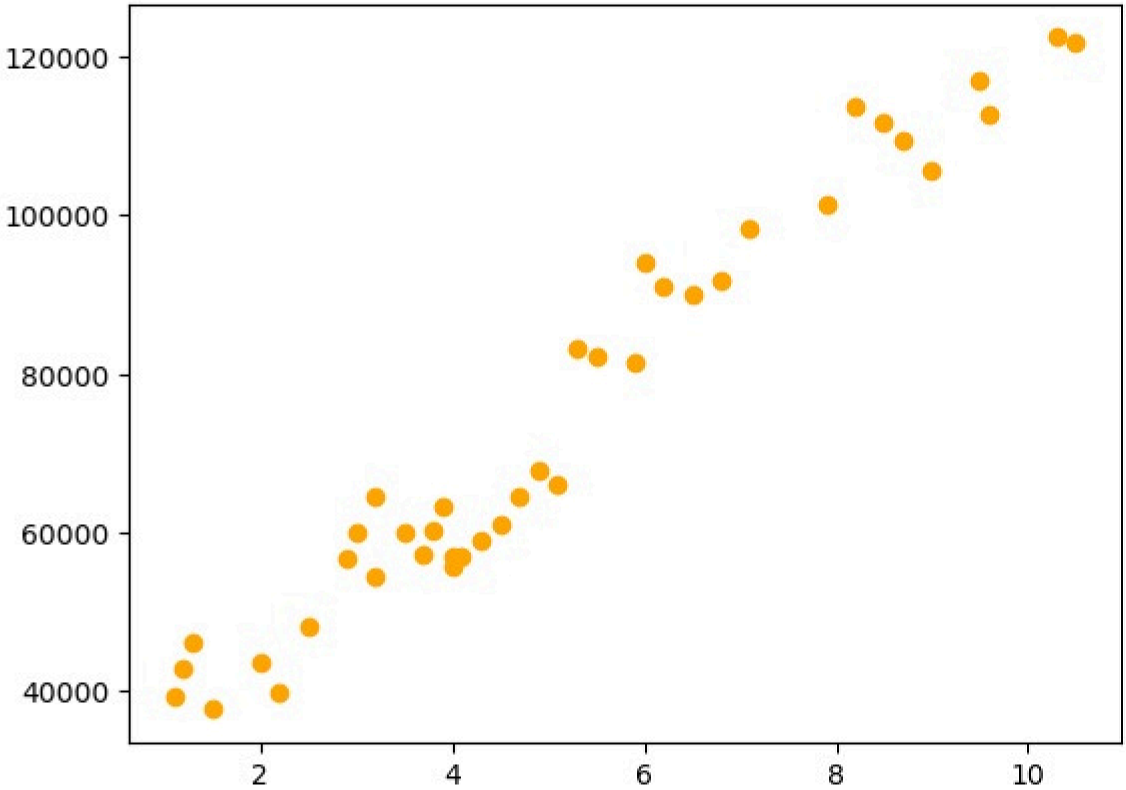
Out[34]:
```
array([ 39343, 42774, 46205, 37731, 43525, 39891, 48266, 56642,
        60150, 54445, 64445, 60000, 57189, 60200, 63218, 55794,
        56957, 57081, 59095, 61111, 64500, 67938, 66029, 83088,
        82200, 81363, 93940, 91000, 90000, 91738, 98273, 101302,
       113812, 111620, 109431, 105582, 116969, 112635, 122391, 121872],
      dtype=int64)
```

Plotting Scatter plot to check relationship between Independent variable and Dependent variable

In [37]:
```
import matplotlib.pyplot as plt

plt.scatter(x,y,color="orange")
```

Out[37]: <matplotlib.collections.PathCollection at 0x15bfb865350>



We observed that the Independent and Dependent variables are linearly related to each other.

Splitting dataset for training set and testing set

In [41]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state
```

Displaying x_train

In [43]:
```
x_train
```

Out[43]:
```
array([[10.5],
       [ 7.1],
       [ 8.7],
       [ 4. ],
       [ 9.5],
       [ 3. ],
       [ 3.8],
       [ 2.2],
       [ 4.1],
       [ 3.9],
       [ 8.5],
       [ 2.9],
       [ 8.2],
       [ 1.2],
       [ 6. ],
       [ 3.7],
       [ 7.9],
       [ 5.5],
       [ 2.5],
       [ 5.3],
       [ 4.9],
       [ 4.5],
       [ 3.2],
       [10.3],
       [ 1.5],
       [ 1.1]])
```

Displaying y_train

In [45]:
```
y_train
```

Out[45]:
```
array([121872, 98273, 109431, 56957, 116969, 60150, 60200, 39891,
        57081, 63218, 111620, 56642, 113812, 42774, 93940, 57189,
        101302, 82200, 48266, 83088, 67938, 61111, 54445, 122391,
        37731, 39343], dtype=int64)
```

Displaying x_test

In [50]:
```
x_test
```

Out[50]:
```
array([[5.1],
[4.7],  [5.9],  [2.  ],
[3.2],  [4.  ],  [6.5],
[3.5],  [4.3],  [6.8],
[6.2],  [9.  ],  [9.6],
[1.3]])
```

Displaying y_test

In [53]:
```
y_test
```

Out[53]:
```
array([ 66029, 64500, 81363, 43525, 64445, 55794, 90000, 60000,
        59095, 91738, 91000, 105582, 112635, 46205], dtype=int64)
```

Fitting Simple Linear regression to the training set

In [56]:
```
from sklearn.linear_model import LinearRegression regressor =
LinearRegression() regressor.fit(x_train,y_train)
```

Out[56]:
```
▾ LinearRegression
LinearRegression()
```

Linear regression analysis is used to predict the value of a variable based on the value of another variable.

Predicting the Test set results

In [59]:
```
y_pred = regressor.predict(x_test)
```

Displaying y_pred (Predicted salary)

In [62]:
```
y_pred
```

Out[62]:
```
array([ 74773.24876271, 70850.52528566, 82618.69571679, 44372.14181561,
        56140.31224674,  63985.75920083,  88502.78093236,  59082.35485453,
        66927.80180862, 91444.82354014, 85560.73832458, 113019.80266389,
       118903.88787946, 37507.37573078])
```

Displaying y_test (Real salary)

In [65]:
```
y_test
```

Out[65]:
```
array([ 66029, 64500, 81363, 43525, 64445, 55794, 90000, 60000,
        59095, 91738, 91000, 105582, 112635, 46205], dtype=int64)
```

Calculating Error

In [68]:
```
error= y_pred - y_test
```

```
error
```

Out[68]:
```
array([ 8744.24876271,  6350.52528566,  1255.69571679,   847.14181561,
       -8304.68775326,  8191.75920083, -1497.21906764,  -917.64514547,
        7832.80180862,  -293.17645986, -5439.26167542,  7437.80266389,
        6268.88787946, -8697.62426922])
```

What insights can be gained from the fitting line in the scatter plot?

Visualizing the training set results

In [72]:
```
plt.scatter(x_train,y_train,color="red")
plt.plot(x_train,regressor.predict(x_train),color="green")
plt.title("Salary VS Experience(Training Set)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```

Visualizing the testing set results

In [75]:
```
plt.scatter(x_test,y_test,color="red")
plt.plot(x_train,regressor.predict(x_train),color="green")
plt.title("Salary VS Experience(Training Set)")
plt.xlabel("Years of Experience")
```

```
plt.ylabel("Salary")
plt.show()
```

Calculating Intercept and Coefficient

Intercept:

It represents the mean value of the response variable when all the predictor variables in the model are equal to zero.

Coefficient:

Coefficients are the values that multiply the predictor values.

In [78]:
```
print(regressor.coef_)
print(regressor.intercept_)
```

```
[9806.80869261]
24758.52443038839
```

In [80]:
```
y_test.shape
```

Out[80]: (14,)

In [82]:
```
y_pred.shape
```

Out[82]: (14,)

Regressor:

Regressor is a statistical term.It refers to any variable in a regression model that is used to predict a response variable.

In [84]:
```
regressor.predict([[5.7]])
```

Out[84]: array([80657.33397827])

What do Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) represent in this context?

Mean Squared Error:

Mean squared error measures the average squared difference between predicted and actual values.

Root Mean Squared Error:

It measures the average difference between values predicted by a model and the actual values.

Model Evaluation

In [59]:
```
from sklearn.metrics import r2_score from
sklearn.metrics import mean_squared_error rmse =
np.sqrt(mean_squared_error(y_test,y_pred)) r2 =
r2_score(y_test,y_pred) print("RMSE =", rmse)
print("R2 Score=",r2)
```

```
RMSE = 6091.7673348888875
R2 Score= 0.9148126520965504
```

Conclusion

Here, We build a regression model and check the model RMSE which is equal to 6091.7673348888875. We also checked for R2 score of our model which is equal to 0.9148126520965504 or 91%. Which is a very good R2 score.