Superwise learning and unsuperwise learning (LAB-1)

Name:- Ayub Alam ** Roll no:- TNU2021053100031L

dept:-CSE (AI & ML)***Assigment date:-

In [35]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [135…

```
#insert your data using pandas packes
data=pd.read_csv(r'linear_regression.csv')
data
```

Out[135…

	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

	YearsExperience	Salary
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
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

In [138…

```
#split the indipendent and dependent so x is dependent
x=data.iloc[:,:-1].values
x
```

Out[138…

```
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 [141…

```
# y is dependent variable
y=data.iloc[:,-1].values
y
```

Out[141…
```
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 [144…
```python
# use the sklearn model for spliting the  testing or training  data
from sklearn.model_selection import train_test_split
#here we spliting the hole data in 1/3 (there is X_train,y_trian (75) for training and
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
x_train
```

Out[144…
```
array([[ 2.9],
       [ 5.1],
       [ 3.2],
       [ 4.5],
       [ 8.2],
       [ 6.8],
       [ 1.3],
       [10.5],
       [ 3. ],
       [ 2.2],
       [ 5.9],
       [ 6. ],
       [ 3.7],
       [ 3.2],
       [ 9. ],
       [ 2. ],
       [ 1.1],
       [ 7.1],
       [ 4.9],
       [ 4. ]])
```

In [145…
```python
y_train
```

Out[145…
```
array([ 56642.,  66029.,  64445.,  61111., 113812.,  91738.,  46205.,
       121872.,  60150.,  39891.,  81363.,  93940.,  57189.,  54445.,
       105582.,  43525.,  39343.,  98273.,  67938.,  56957.])
```

In [146…
```python
x_test
```

Out[146…
```
array([[ 1.5],
       [10.3],
       [ 4.1],
       [ 3.9],
       [ 9.5],
       [ 8.7],
       [ 9.6],
       [ 4. ],
       [ 5.3],
       [ 7.9]])
```

In [147…
```python
y_test
```

Out[147…
```
array([ 37731., 122391.,  57081.,  63218., 116969., 109431., 112635.,
        55794.,  83088., 101302.])
```

In [148…
```python
# use sklearn.linear_model for  linearregression that will help you
```

```
from sklearn.linear_model import LinearRegression
linear=LinearRegression()
linear
```

Out[148… LinearRegression()

In [150…

```
model=linear.fit(x,y)
model
```

Out[150… LinearRegression()

In [152…

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

Out[152…

```
array([ 36187.15875227,  38077.15121656,  39967.14368085,  44692.12484158,
        46582.11730587,  53197.09093089,  54142.08716303,  56032.07962732,
        56032.07962732,  60757.06078805,  62647.05325234,  63592.04948449,
        63592.04948449,  64537.04571663,  68317.03064522,  72097.0155738 ,
        73987.00803809,  75877.00050238,  81546.97789525,  82491.9741274 ,
        90051.94398456,  92886.932681  , 100446.90253816, 103281.8912346 ,
       108006.87239533, 110841.86109176, 115566.84225249, 116511.83848464,
       123126.81210966, 125016.80457395])
```

In [109…

```
#find the  coefficient
model.coef_
```

Out[109… array([9449.96232146])

In [110…

```
#find the intercept value
model.intercept_
```

Out[110… 25792.20019866871

In [153…

```
#find the mean_absolute_error and mean_squared_error using the sklearn.metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error
print(mean_squared_error(y,y_prd))
```

31270951.722280968

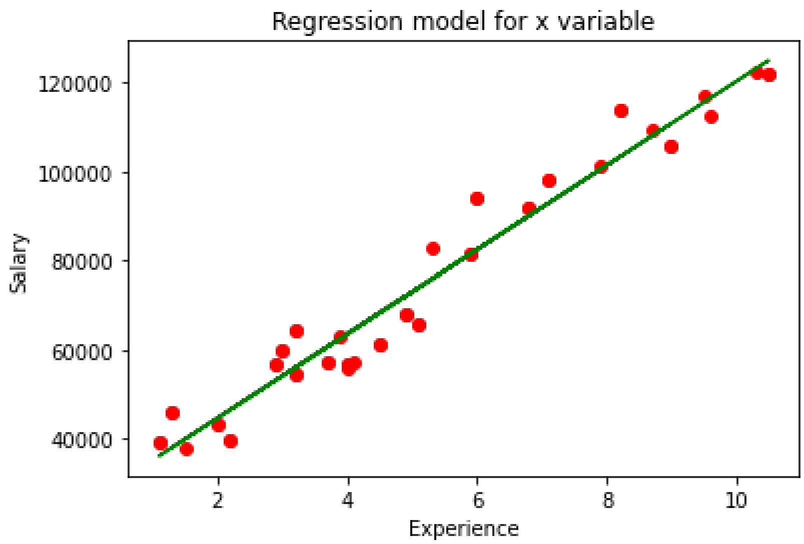
In [154…

```
model.predict([[7.3]])
```

Out[154… array([94776.92514529])

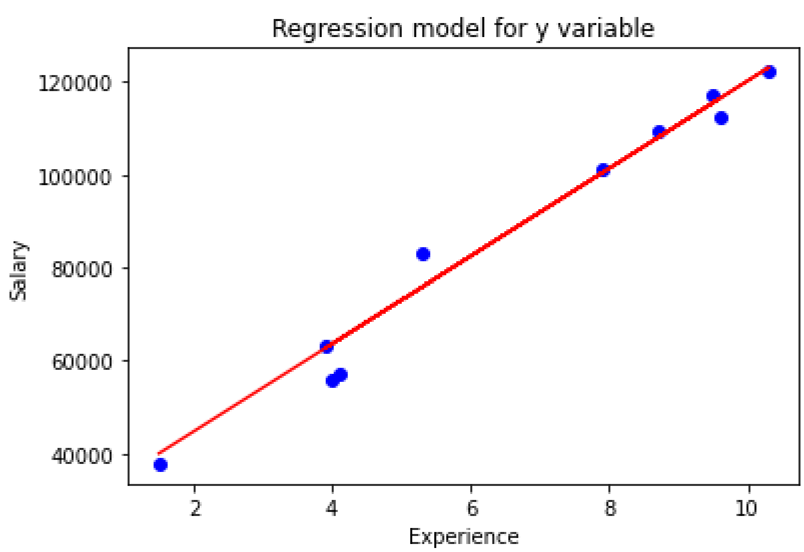
In [155…

```
plt.scatter(x,y,color="red")
plt.scatter(x_train,y_train,color="red")
plt.plot(x_train,model.predict(x_train),color="green")
plt.title('Regression model for x variable')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()
```



In [156...

```
plt.scatter(x_test,y_test,color="blue")
plt.plot(x_test,model.predict(x_test),color="red")
plt.title('Regression model for y variable')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()
```



for manual predicting value of the data set

In [117...

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [118...

```
#
dataset=pd.read_csv(r'Linear_Regression.csv')
```

In [119...

```
dataset
```

Out[119...

	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
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

In [120...

```
x=dataset.iloc[:,:-1].values
x
```

Out[120...

```
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 [121…

```
y=dataset.iloc[:,-1].values
Y=np.reshape(y,(30,1))
# Y=np.reshape(y,(5,1))
Y
```

Out[121…

```
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 [122…

```
# calculate the mean of x
x_mean=np.mean(x)
x_mean
```

Out[122…

```
5.3133333333333335
```

In [123…

```
y_mean=np.mean(Y)
y_mean
```

Out[123…

```
76003.0
```

In [124…

```
# for finding the value of (x-x_mean) for each value
each_sub_x=x-x_mean
each_sub_x
```

Out[124…

```
array([[-4.21333333],
       [-4.01333333],
       [-3.81333333],
       [-3.31333333],
       [-3.11333333],
       [-2.41333333],
       [-2.31333333],
       [-2.11333333],
       [-2.11333333],
       [-1.61333333],
       [-1.41333333],
       [-1.31333333],
       [-1.31333333],
       [-1.21333333],
       [-0.81333333],
       [-0.41333333],
       [-0.21333333],
       [-0.01333333],
       [ 0.58666667],
       [ 0.68666667],
       [ 1.48666667],
       [ 1.78666667],
       [ 2.58666667],
       [ 2.88666667],
       [ 3.38666667],
       [ 3.68666667],
       [ 4.18666667],
       [ 4.28666667],
       [ 4.98666667],
       [ 5.18666667]])
```

In [125…

```
#for finding the value of (y-y_mean) for each value
each_sub_y=Y-y_mean
each_sub_y
```

Out[125…

```
array([[-36660.],
       [-29798.],
       [-38272.],
       [-32478.],
       [-36112.],
```

```
       [-19361.],
       [-15853.],
       [-21558.],
       [-11558.],
       [-18814.],
       [-12785.],
       [-20209.],
       [-19046.],
       [-18922.],
       [-14892.],
       [ -8065.],
       [ -9974.],
       [  7085.],
       [  5360.],
       [ 17937.],
       [ 15735.],
       [ 22270.],
       [ 25299.],
       [ 37809.],
       [ 33428.],
       [ 29579.],
       [ 40966.],
       [ 36632.],
       [ 46388.],
       [ 45869.]])
```

In [126…

```
#for multiplying the x-x_mean* y-y_mean
mul_x_y=each_sub_x*each_sub_y
mul_x_y
```

Out[126…

```
array([[ 1.54460800e+05],
       [ 1.19589307e+05],
       [ 1.45943893e+05],
       [ 1.07610440e+05],
       [ 1.12428693e+05],
       [ 4.67245467e+04],
       [ 3.66732733e+04],
       [ 4.55592400e+04],
       [ 2.44259067e+04],
       [ 3.03532533e+04],
       [ 1.80694667e+04],
       [ 2.65411533e+04],
       [ 2.50137467e+04],
       [ 2.29586933e+04],
       [ 1.21121600e+04],
       [ 3.33353333e+03],
       [ 2.12778667e+03],
       [-9.44666667e+01],
       [ 3.14453333e+03],
       [ 1.23167400e+04],
       [ 2.33927000e+04],
       [ 3.97890667e+04],
       [ 6.54400800e+04],
       [ 1.09141980e+05],
       [ 1.13209493e+05],
       [ 1.09047913e+05],
       [ 1.71510987e+05],
       [ 1.57029173e+05],
       [ 2.31321493e+05],
       [ 2.37907213e+05]])
```

In [127…

```
# for total sum  of (each_sub_x)*(each_sub_y)
total_mul=np.sum(mul_x_y)
```

```
total_mul
```

Out[127...

```
2207082.8000000003
```

In [157...

```
#the value of (x-x_mean)**2
square_mean=each_sub_x**2
square_mean
```

Out[157...

```
array([[1.77521778e+01],
       [1.61068444e+01],
       [1.45415111e+01],
       [1.09781778e+01],
       [9.69284444e+00],
       [5.82417778e+00],
       [5.35151111e+00],
       [4.46617778e+00],
       [4.46617778e+00],
       [2.60284444e+00],
       [1.99751111e+00],
       [1.72484444e+00],
       [1.72484444e+00],
       [1.47217778e+00],
       [6.61511111e-01],
       [1.70844444e-01],
       [4.55111111e-02],
       [1.77777778e-04],
       [3.44177778e-01],
       [4.71511111e-01],
       [2.21017778e+00],
       [3.19217778e+00],
       [6.69084444e+00],
       [8.33284444e+00],
       [1.14695111e+01],
       [1.35915111e+01],
       [1.75281778e+01],
       [1.83755111e+01],
       [2.48668444e+01],
       [2.69015111e+01]])
```

In [158...

```
# total sum of mean_square
total_sum=np.sum(square_mean)
total_sum
```

Out[158...

```
233.55466666666666
```

In [159...

```
# for finding the slope value is m=(x-x_mean)(y-y_mean)/**2
slope_m=total_mul/total_sum
slope_m
```

Out[159...

```
9449.962321455077
```

In [160...

```
# now find the value of coeficient from straight line formula : y=mx+c so we can do c=y_
c=y_mean-slope_m*x_mean
c
```

Out[160...

```
25792.20019866869
```

In [163…

```
# for finding the error for each element error =(y-(mx+c))**2
error=(Y-(slope_m*x+c))**2
error
```

Out[163…

```
array([[9.95933398e+06],
       [6.60619258e+07],
       [5.00033856e+06],
       [1.36218040e+06],
       [4.47710508e+07],
       [1.18673985e+07],
       [3.60950167e+07],
       [2.51882174e+06],
       [7.07772292e+07],
       [1.27310578e+07],
       [3.25980189e+05],
       [6.08095758e+07],
       [4.40238817e+07],
       [5.55926177e+07],
       [5.19268777e+07],
       [1.72974105e+07],
       [6.33298919e+07],
       [5.19985138e+07],
       [3.38478659e+04],
       [1.31057296e+08],
       [2.84278489e+06],
       [2.90097212e+07],
       [7.31191669e+05],
       [1.10883191e+08],
       [2.02813943e+06],
       [2.76661387e+07],
       [1.96604635e+06],
       [1.50298766e+07],
       [5.41419461e+05],
       [9.88979581e+06]])
```

In [164…

```
total_error=np.sum(error)
total_error
# mean_squre_error=(total_error/5)
mean_squre_error=(total_error/30)
mean_squre_error
```

Out[164…

```
31270951.72228097
```