CS-CULT (Data Science Division)

Topic : Prediction Of Salary On The Basis Of Years Of Experience

Language Used : Python

Platform Used : Jupyter Notebook

In [1]:

```
# Importing all the necessary libraries

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [2]:

```
# pandas provide us function to load our data set to our model
#(using read_csv() function)

df = pd.read_csv("Salary_Data.csv")
```

In [3]:

```
df # Printing the data set (YearsExperience and Salary)
```

Out[3]:

	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 [4]:

```
# Lets have a look over the shape of dataset using shape function!

df.shape # Tells us the number of rows and number of colomns in our data set
```

Out[4]:

```
(30, 2)
```

In [5]:

```
# head and tail functions let us know about the first five observations and last
#observations in our data set respectively!

df.head()

# prints the first five observation of our data set!
```

Out[5]:

	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 [6]:

```
df.tail()

# Prints the last five observations of our data set!
```

Out[6]:

	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 [7]:

```
# This process is to check the wether our dataset is filled with any empty
# values and tells us through a boolean value to make
# inference

# As we can see that entire datset comes out to be with a false value on isna()
# function which means that no null values are
# present


df.isna()
```

Out[7]:

	YearsExperience	Salary
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False
25	False	False
26	False	False
27	False	False
28	False	False

	YearsExperience	Salary
29	False	False

In [8]:

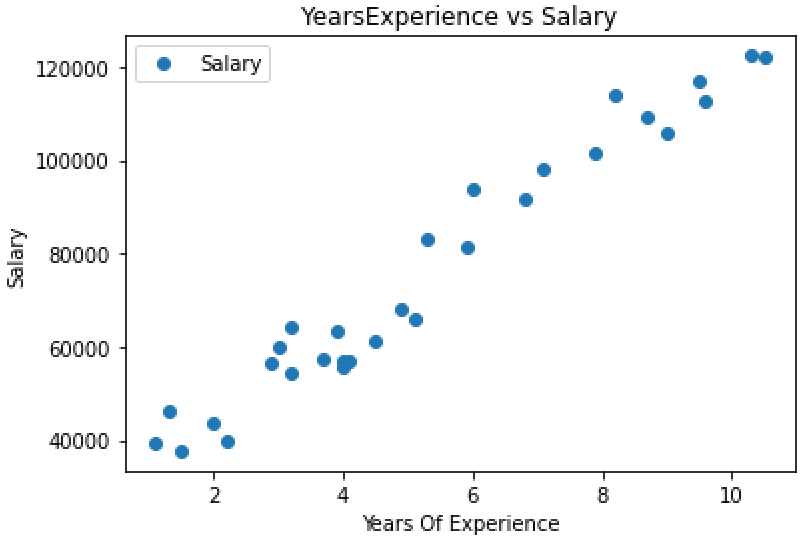
```
# This step is to confirm that no null values are present in our data set
# so we can move ahead with further operations!


df.isna().sum()
```

Out[8]:

```
YearsExperience    0
Salary             0
dtype: int64
```

Next step is to move on over the concept of graphs and plots to visually represent our data

In [9]:
```
# matplotlib is the library used for plotting the graphs which we have already
#imported above

# This means that we need to plot YearsExperience on x-axis and Salary on y-axis
df.plot(x = 'YearsExperience' , y = 'Salary' , style = 'o')

# This title function gives the us the options to provide a specefic title to our
# plot!
plt.title('YearsExperience vs Salary')

# Now its time to give the titles to x-axis and y-axis using xlabel() and ylabel(
# functions respectively!

# xlabel() gives the title to the x-axis
plt.xlabel('Years Of Experience')

# ylabel() gives the title to y-axis
plt.ylabel('Salary')

# Now we have given all the specifications to our graph , so lets have a look on
# our plot using show() function!
plt.show()
```



From the above plot we can infer that their exists a linear relationship between Years Of

Experience and Salary

That means as the Years Of Experience increases , Salary also Increases!

In [10]:

```
# Now its time to make our model which will predict the resuts on the given data

# So dividing our data into features(inputs) and labels(output) using iloc()
# function in python

# This gives us the entire x attributes into an specefic array!
x = df.iloc[:,:-1].values
```

In [11]:

```
x
```

Out[11]:

```
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 [12]:

```
# This gives us the entire y attributes into a specefic array!
y = df.iloc[:,1].values
```

In [13]:

```
y
```

Out[13]:

```
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 [14]:

```
# Lets train our model and test it accordingly!

from sklearn.model_selection import train_test_split
```

In [15]:

```
# We are training our data set with a test_size of 0.2 which means that we are
# only using 20% of our data to train our model!

X_train , X_test , y_train , y_test = train_test_split(x , y , test_size = 0.2
                                                , random_state = 0)
```

Implementing Linear Regression Algorithm

In [16]:

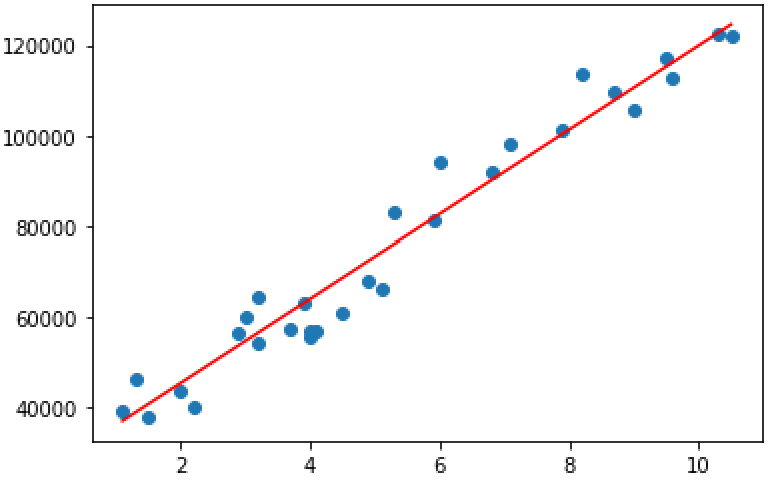
```
# Importing linear regression algorithm which comes under sklearn

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

# We fitted our training variables in the LinearRegression Model using fit
# Function
regressor.fit(X_train , y_train)
# Linear Regression model called Successfully!
# Training complete
```

Out[16]:

```
LinearRegression()
```

In [17]:

```
# Plotting the regression line
line = regressor.coef_*x+regressor.intercept_
# compare it with equation of straight line [y = mx + c])


# Plotting for the test data

plt.scatter(x, y)
plt.plot(x, line , color = 'red');
plt.show()
```



In [18]:

```
# Testing our model

print(X_test) # these are the attributes of Years Of Experience we have tested!
```

```
[[ 1.5]
 [10.3]
 [ 4.1]
 [ 3.9]
 [ 9.5]
 [ 8.7]]
```

In [19]:

```
# Making predictions on the values we have tested!

y_pred = regressor.predict(X_test) # Predicted
```

In [20]:

```
# Creating a data frame with Actual and Predicted Values!

data_frame = pd.DataFrame({'Actual' : y_test , 'Predicted' : y_pred})
data_frame
```

Out[20]:

	Actual	Predicted
0	37731.0	40748.961841
1	122391.0	122699.622956
2	57081.0	64961.657170
3	63218.0	63099.142145
4	116969.0	115249.562855
5	109431.0	107799.502753

In [24]:

```
# Checking our Model finally at 2.5 years of Experience

years_exp = [[2.5]]
own_pred = regressor.predict(years_exp)
print("Years of exp is {}" .format(years_exp))
print("Predicted salary is {}" .format(own_pred[0]))
```

```
Years of exp is [[2.5]]
Predicted salary is 50061.53696745115
```

Hence our model predicted almost with the same value as was in the data set

In [ ]:

```

```