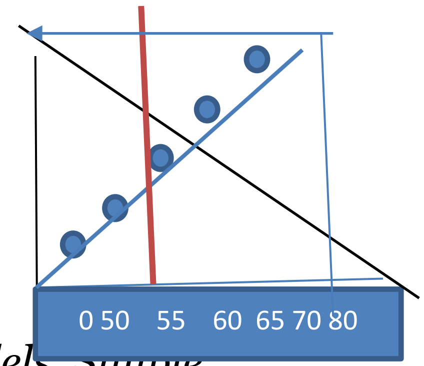
Simple Linear Regression in a Comprehensive way

Regression is a form of predictive modelling technique, which investigate’s the relationship between independent and dependent vectors. Regression is of many types and Linear Regression is one among them. Linear Regression predicts the dependent vector by assuming the relationship between the independent and dependent vector is a straight line



What is Simple Linear Regression ?

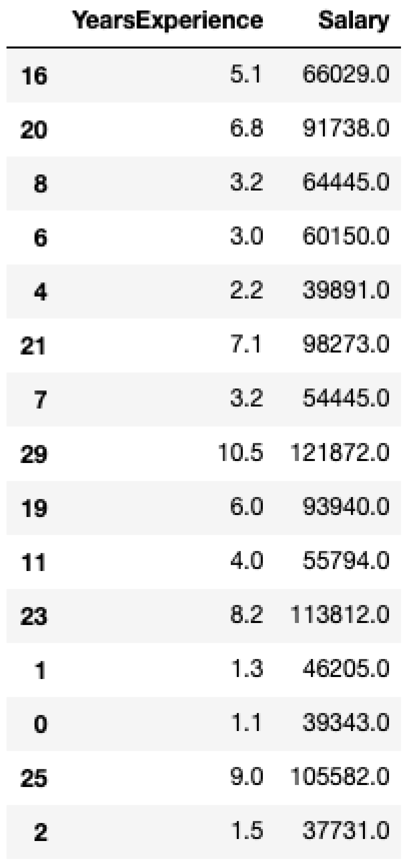
It’s the easiest approach among the Regression models. Simple Linear Regression is applied only when our data has one independent variable and it predicts the dependent vector, by estimating the relation among the dependent and independent vectors as a straight line. It expresses the relation among the dependent and independent vector’s as a straight and is in the form as below.

```
y = mx + c where
y is the dependent vector
x is the independent vector
c is the constant and also called as bias and also known as
intercept, which is added to the line
m is slope and it is an offset that equals both vectors
```

Mathematically, c is the y-intercept that determines the value of y when x is 0 and m is the slope which determines the angle of line.

How it works?

The Simple Linear Regression model assumes and plots a Regression line, in such a way that the Regression line should be as close as to all data points of the dataset. To get more clear idea about how it works, let's go through an example. We have an salary dataset comprised of Years of Experience and Salary. The dataset is as follows.

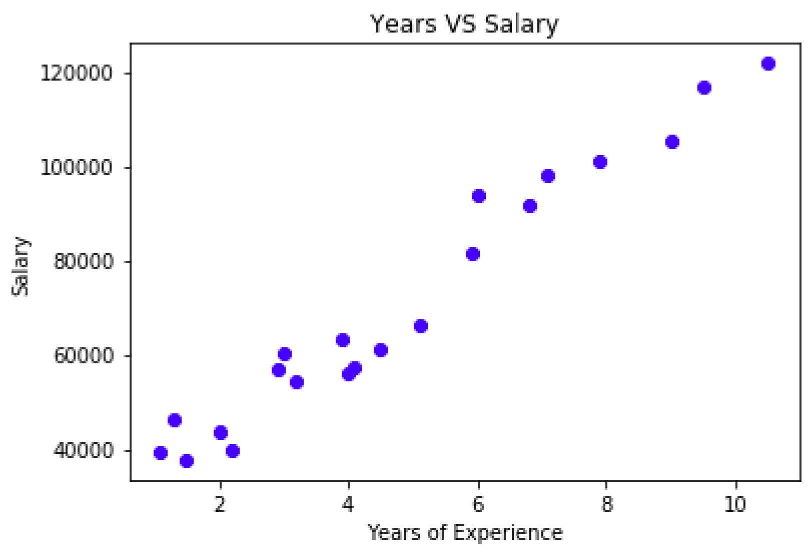


	YearsExperience	Salary
16	5.1	66029.0
20	6.8	91738.0
8	3.2	64445.0
6	3.0	60150.0
4	2.2	39891.0
21	7.1	98273.0
7	3.2	54445.0
29	10.5	121872.0
19	6.0	93940.0
11	4.0	55794.0
23	8.2	113812.0
1	1.3	46205.0
0	1.1	39343.0
25	9.0	105582.0
2	1.5	37731.0

Overview of dataset

Here dependent variable is salary and independent variable is YearsExperience. If there is an increase in dependent variable when independent variable increases, then there is a positive correlation among them. If decreases, then there is a negative correlation among them.

Now let’s plot a scatter plot among YearsExperience and Salary.



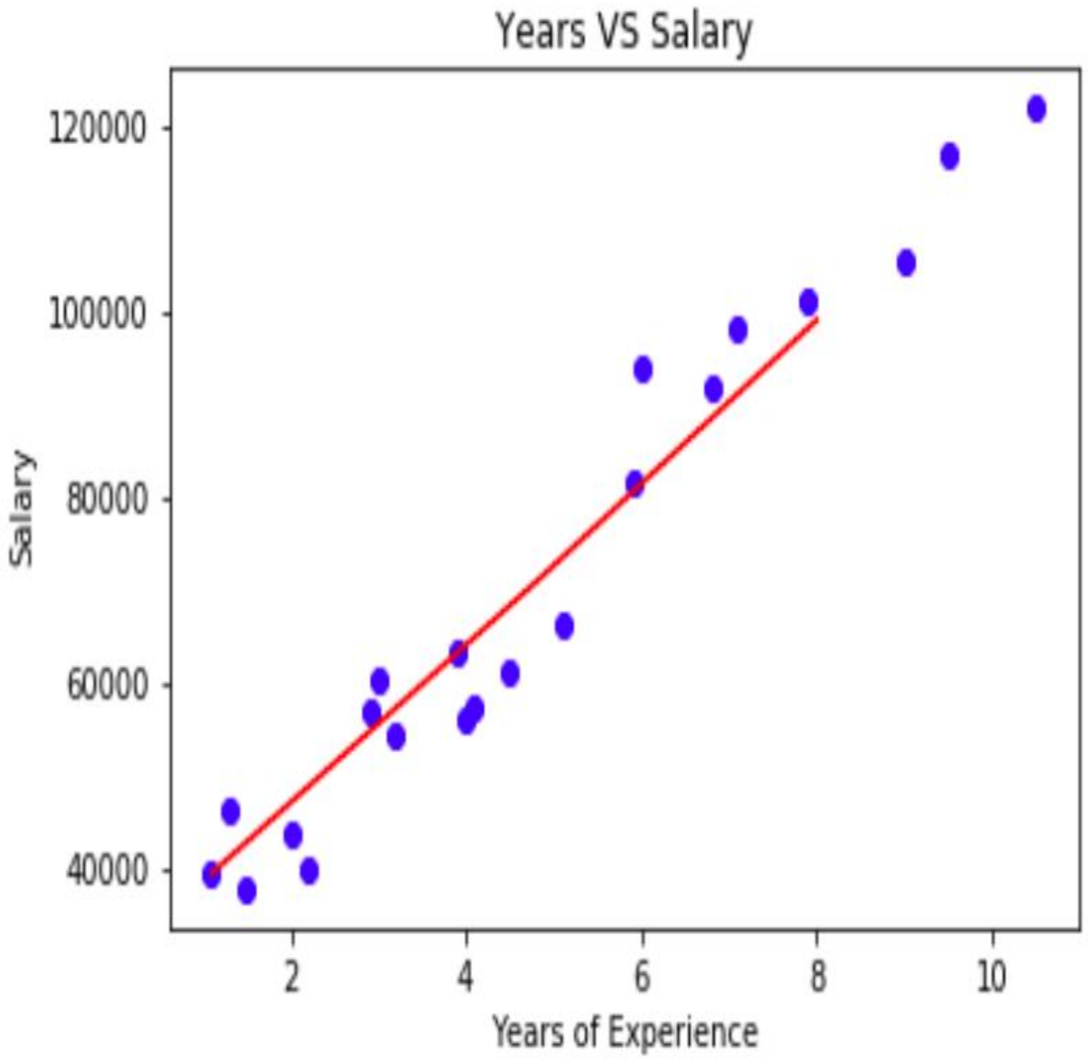
Scatter plot Experience VS Salary

The best fit line for the data is the one which produces least error or least square approximation error among all regression line’s that can be drawn. This method of finding best fit line is called Least Square Approximation Method.

Now let’s get started by drawing an regression line among the data points in the scatter plot using mean of independent and dependent vectors. Draw a line across the point such that the point will be the mean of two vectors.

```
Xm ( Mean) = Sum of all experience values / Total number of
experience values
Ym (Mean) = Sum of all salary values / Total number of salary
values
```

Now plot a line which will be our assumed regression line over the data ponts in the scatter plot.



Regression line plotted using mean

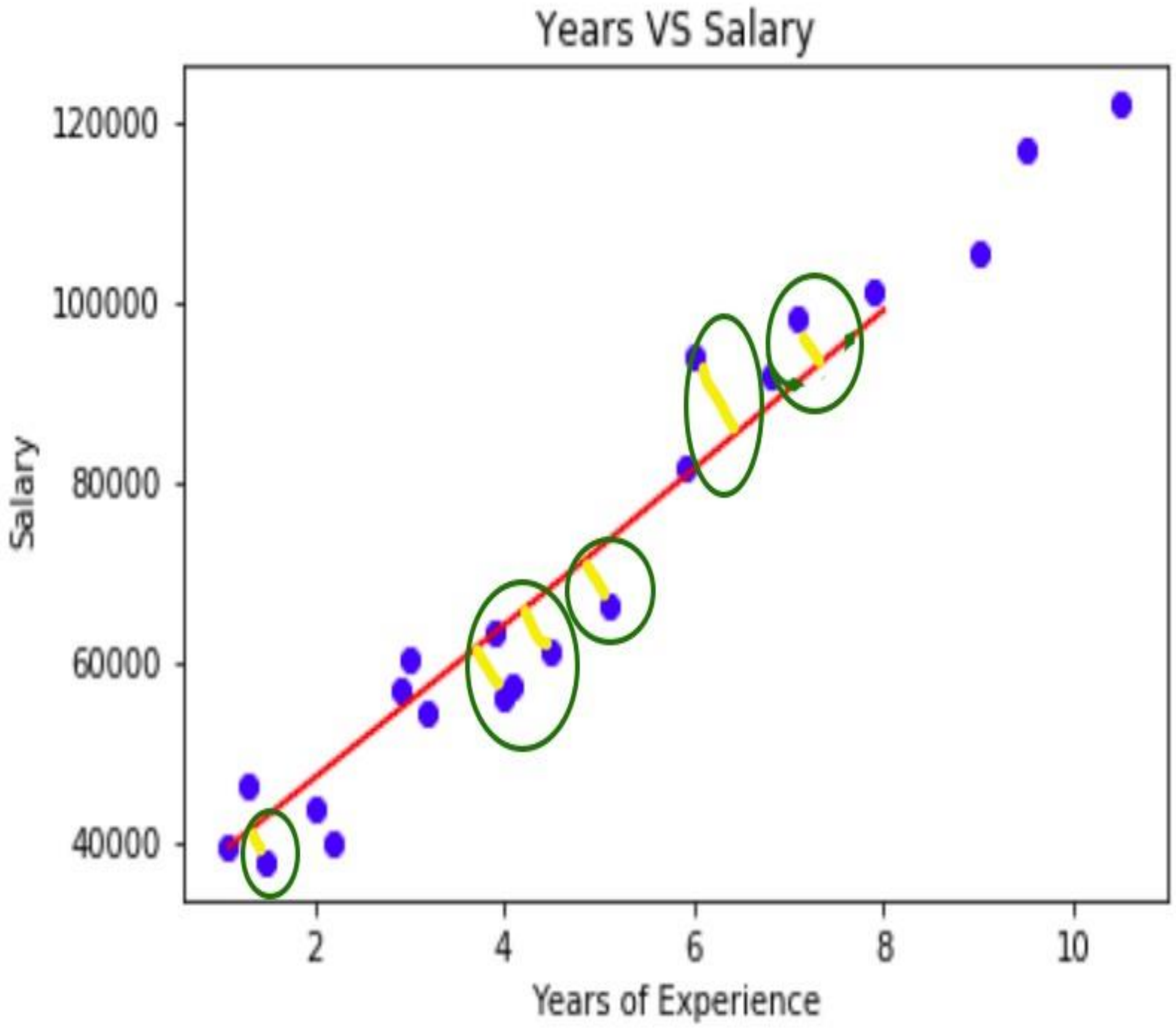
From the above plot, we can observe that the regression line is some what far from some data points. This whole process is an iterable one and will be continued until the best fit line which is having the least square approximation distance is obtained.

The values corresponding to the original value on the regression line are called predicted values. The Least Approximation distance can be calculated as follows.

Actualvalue	Predicted
10000	12000=10000-12000=abs(-2000)2000(error)
150000	145000=15000-145000=500

(2000+500)2

$$\min \sum (y_i - \hat{y}_i)^2$$



Distance Approximation among original value and predicted value

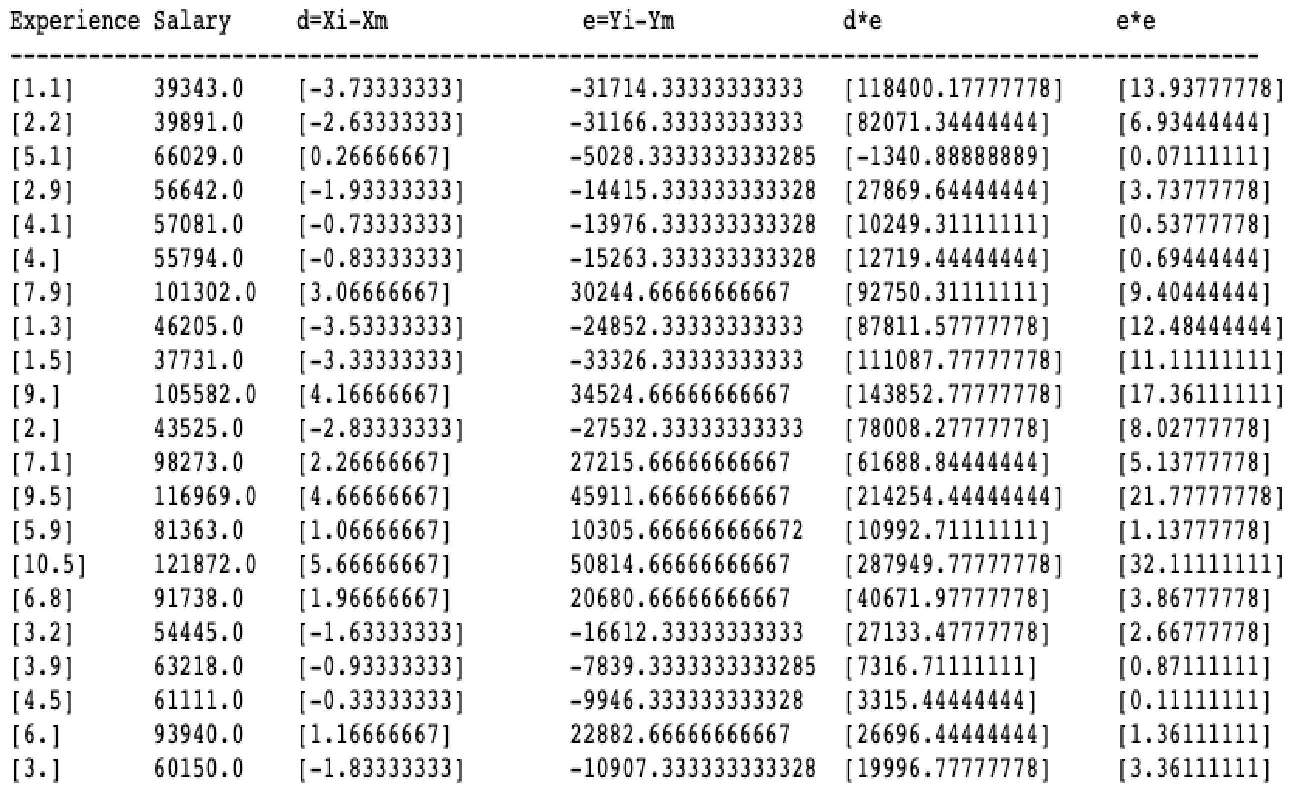
To calculate the regression line, for which the approximation distance is minimum when compared to the regression line now.

The slope or multiplier of new regression line can be calculated as follows:

$$\text{slope} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

where Xi, Yi are the values of independent vector and dependent vectors. They are subtracted from their corresponding mean's and the slope of new regression line is calculated.

Let's calculate the slope for new regression line. You can view the whole calculation in the following table.



```
Experience Salary     d=Xi-Xm              e=Yi-Ym             d*e                   e*e
------------------------------------------------------------------------------------------
[1.1]      39343.0    [-3.73333333]        -31714.33333333333   [118400.17777778]    [13.93777778]
[2.2]      39891.0    [-2.63333333]        -31166.33333333333   [82071.34444444]     [6.93444444]
[5.1]      66029.0    [0.26666667]         -5028.3333333333285  [-1340.88888889]     [0.07111111]
[2.9]      56642.0    [-1.93333333]        -14415.333333333328  [27869.64444444]     [3.73777778]
[4.1]      57081.0    [-0.73333333]        -13976.333333333328  [10249.31111111]     [0.53777778]
[4.]       55794.0    [-0.83333333]        -15263.333333333328  [12719.44444444]     [0.69444444]
[7.9]      101302.0   [3.06666667]         30244.66666666667    [92750.31111111]     [9.40444444]
[1.3]      46205.0    [-3.53333333]        -24852.33333333333   [87811.57777778]     [12.48444444]
[1.5]      37731.0    [-3.33333333]        -33326.33333333333   [111087.77777778]    [11.11111111]
[9.]       105582.0   [4.16666667]         34524.66666666667    [143852.77777778]    [17.36111111]
[2.]       43525.0    [-2.83333333]        -27532.33333333333   [78008.27777778]     [8.02777778]
[7.1]      98273.0    [2.26666667]         27215.66666666667    [61688.84444444]     [5.13777778]
[9.5]      116969.0   [4.66666667]         45911.66666666667    [214254.44444444]    [21.77777778]
[5.9]      81363.0    [1.06666667]         10305.666666666672   [10992.71111111]     [1.13777778]
[10.5]     121872.0   [5.66666667]         50814.66666666667    [287949.77777778]    [32.11111111]
[6.8]      91738.0    [1.96666667]         20680.66666666667    [40671.97777778]     [3.86777778]
[3.2]      54445.0    [-1.63333333]        -16612.33333333333   [27133.47777778]     [2.66777778]
[3.9]      63218.0    [-0.93333333]        -7839.3333333333285  [7316.71111111]      [0.87111111]
[4.5]      61111.0    [-0.33333333]        -9946.333333333328   [3315.44444444]      [0.11111111]
[6.]       93940.0    [1.16666667]         22882.66666666667    [26696.44444444]     [1.36111111]
[3.]       60150.0    [-1.83333333]        -10907.333333333328  [19996.77777778]     [3.36111111]
```

Hence the summation of d*e and e*e are 1463496.36666667 and 3.366111111. The slope of the new regression cab be calculated from the above value's and it is 0.00016808.

I have performed all the operation using python code. Have a look at it.

```
Xm = np.mean(X_train)
Ym = np.mean(y_train)
sum1 = 0
sum2 = 0
print('Experience Salary     d=Xi-Xm              e=Yi-Ym
d*e                 e*e')
print('----------------------------------------------------------
-----------------------------------')
for pos in range(0, len(X_train)):
    d = (X_train[pos] - Xm)
    e = (y_train[pos] - Ym)
    sum1 = sum1 + d*e
    sum2 = sum2 = d*d
    print(f'{str(X_train[pos]):{10}} {str(y_train[pos]):{10}}
{str(X_train[pos]-Xm):{20}} {str(y_train[pos]-Ym):20}
{str(d*e):{20}} {str(d*d):{20}}')
```

From the slope, we can calculate the y-intercept or bias by substituting x and y as zero in y-mx+c equation.

The obtained line equation is the new regression line and this process is continued for all regression line that cab be possibly drawn in our scatter plot. The regression line with minimum least square approximaton error is called best fit line.

Don’t worry, python’s scikit learn library does all these hectic work for us.

R-square Regression Analysis

To check how efficient is our model fits to the data, we can use the R-square Regression analysis method. This method is also called Coefficient of determination. Higher the R-square value, higher the efficiency of our model(value near 1). But not all R-square value models are bad, it is based on the problem statement.

We can find the R-square value of the model in the following way.

$$R^2 = \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$$

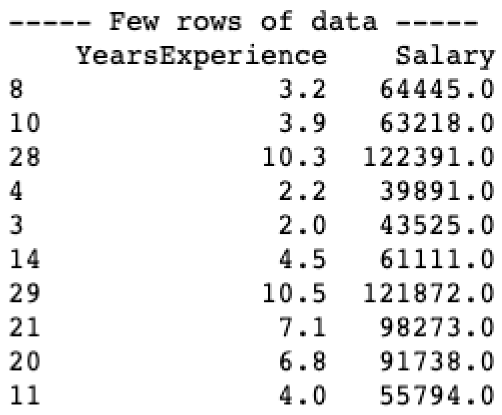
where Yp is the predicted dependent variable, y is the actual or original dependent variable and Ym is the mean of the dependent variable. Like in the Least square Approximation method we can calculate R-square value.

Let's implement Simple Linear Regression on Salaries data. Firstly, import all the necessary libraries and load the dataset.

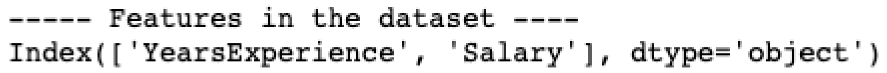
```
# Importing the librariesimport numpy as np
import pandas as pd
import matplotlib.pyplot as plt# Loading the datasetdf =
pd.read_csv('Salary_Data.csv')
```

Now go through the data or perform some EDA (Exploratory Data Analysis), to understand and get familiar with the dataset.

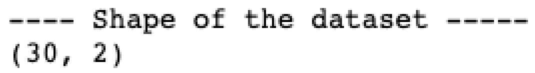
```
# Viewing few rows of dataprint('----- Few rows of data -----')
print(df.sample(10))print('\n\n')
print('----- Features in the dataset ----')
print(df.columns)print('\n\n')
print('---- Shape of the dataset -----')
print(df.shape)
```



```
----- Few rows of data -----
    YearsExperience    Salary
8               3.2   64445.0
10              3.9   63218.0
28             10.3  122391.0
4               2.2   39891.0
3               2.0   43525.0
14              4.5   61111.0
29             10.5  121872.0
21              7.1   98273.0
20              6.8   91738.0
11              4.0   55794.0
```



```
----- Features in the dataset ----
Index(['YearsExperience', 'Salary'], dtype='object')
```



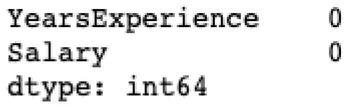
```
---- Shape of the dataset -----
(30, 2)
```

Some insights about the datset

There are about 30 observations of data with two columns namely YearsExperience and Salary in our dataset. Our problem statement is to predict the salary based upon the experience(in years) he/she has. So YearsExperience is the independent variable and Salary becomes the dependent variable as per our problem statement.

Let’s have a look over our dataset for any missing values in the dataset.

```
# Check for null valuesdf.isnull().sum()
```



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

Insights about Missing values in the dataset

Hurrah..!! No missing values are present in the dataset. So no need of any data preprocessing, let's jump directly into splitting our dataset into independent and dependent vectors.

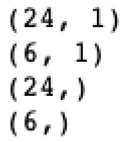
```
# Converting dataset to dependent and independant vectors#
YearsExperience
X = df.iloc[:, :-1].values# Salary
y = df.iloc[:, 1].values
```

Now split the data into training and test data using scikit learn's train_test_split method. I have spliced data in a way that 80 percent is training set and 20 percent is test set.

```
# Splitting the dataset into testing and training set'sfrom
sklearn.model_selection import train_test_splitX_train, X_test,
y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Some insights about the dimensions of training and test data after splitting them.

```
# Dimensions of datset after splitting into testing and training
set'sprint(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

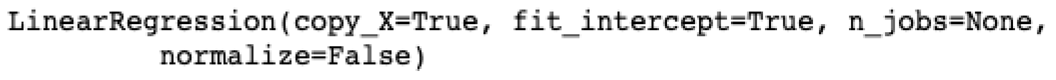


```
(24, 1)
(6, 1)
(24,)
(6,)
```

Dimension of the resultant datsets

Now the test and train dataset's are ready. Let's start importing the scikit learn's LinearRegression model and instantiate it.

```
# Fitting Simple Linear Regression model to the training datafrom
sklearn.linear_model import LinearRegression# Instantiating
LinearRegression Modellinear_regression = LinearRegression()#
Fitting to the training datalinear_regression.fit(X_train, y_train)
```



```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
        normalize=False)
```

Output showing our Linear Regression model is trained

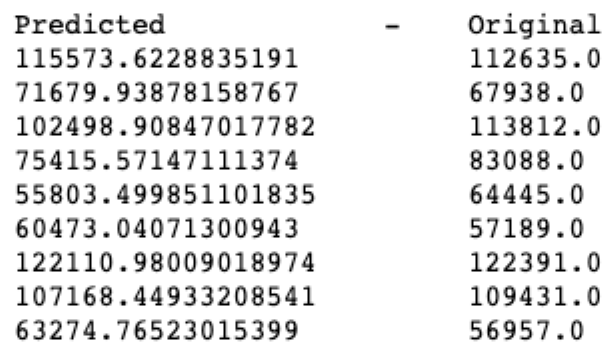
The LinearRegression Model that was instantiated was fitted to our training data. Means a Regression line (a best fit line) with minimum least square approximation distance was identified. So from that regression line, our model starts predicting the output.

So start passing the test data to the model, to see the predictions of salary that our model predicts.

```
# Predicting dependent variable using independant
variablepredictions = linear_regression.predict(X_test)
```

Our Model has predicted the salaries of persons with respect to experience. Let’s view the predicted values and the original values together.

```
# Lets view predicted and original salaries
print('Predicted            -    Original')
for pos in range(0, len(predictions)):
    print(f'{predictions[pos]:<{25}}  {y_test[pos]:<{15}}')
```



```
Predicted              -    Original
115573.6228835191           112635.0
71679.93878158767           67938.0
102498.90847017782          113812.0
75415.57147111374           83088.0
55803.499851101835          64445.0
60473.04071300943           57189.0
122110.98009018974          122391.0
107168.44933208541          109431.0
63274.76523015399           56957.0
```

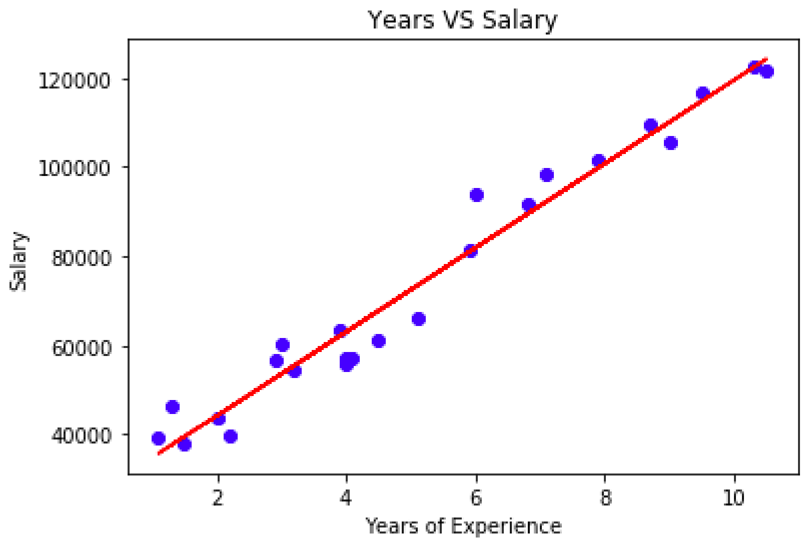
Predicted and original values

In some cases, the predicted values are very close to the original values and in some cases the predicted values are some what far but no too much from the original values. This is because our Regression Model learns the correlation among the variables by expressing as a straight line. So not all data points passed through the regression lines, due to outliers and some factors. So these causes these differences between and original target values. So that's the reason that Linear Regression is not 100 percent accurate.

Let's visualize the relationship between the independent variables with both predicted and original values along with the Regression line to get clear idea.

Now let's plot a scatter plot between Experience and Salary of training dataset and along with Regression line.

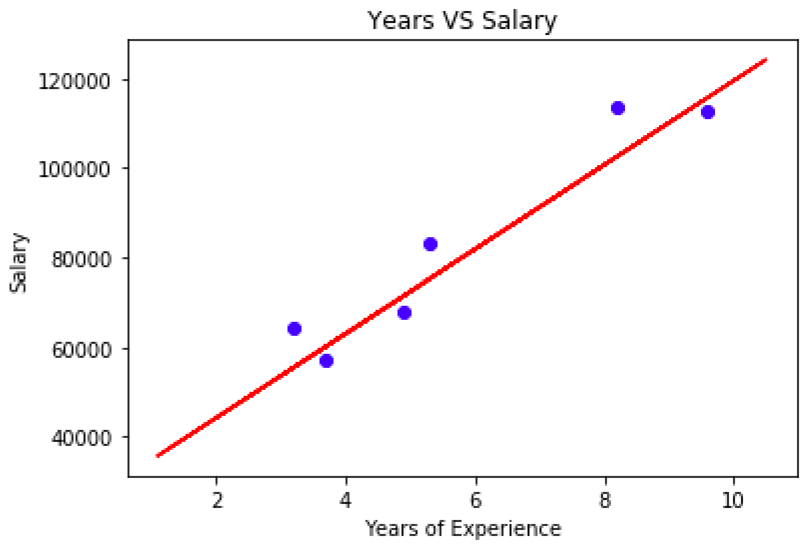
```
# Training data VS Regression line
# Regression line is drawn using predicted values for training
setplt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, linear_regression.predict(X_train), color='red')
plt.title('Years VS Salary')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



Original and Predicted values of training set

Since it's the best fit line, mostly every data point is very close to the Regression line. Now let's plot a scatter plot between

Experience and Salary of test dataset and along with Regression line.

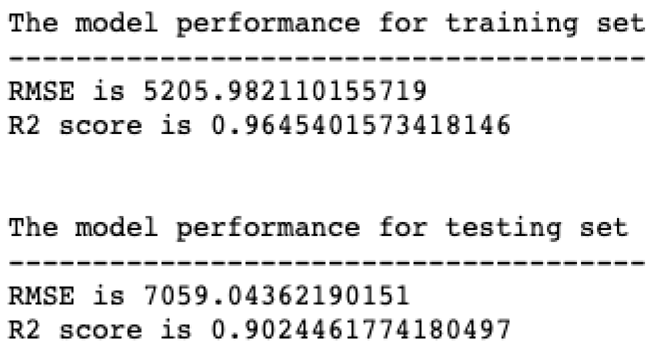


Original vs predicted values of test data

Even the Regression line is very close to mostly every point in the test dataset.

Mean Square error and R-square regression can be performed in the following way.

```
#import librariesfrom sklearn.metrics import
mean_squared_error,r2_score# model evaluation for training
sety_train_predict = linear_regression.predict(X_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
r2 = r2_score(y_train, y_train_predict)
print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")# model evaluation for testing sety_test_predict =
linear_regression.predict(X_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
r2 = r2_score(y_test, y_test_predict)print("The model performance
for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```



```
The model performance for training set
--------------------------------------
RMSE is 5205.982110155719
R2 score is 0.9645401573418146


The model performance for testing set
--------------------------------------
RMSE is 7059.04362190151
R2 score is 0.9024461774180497
```

mean squared error and r2 error for train and test data