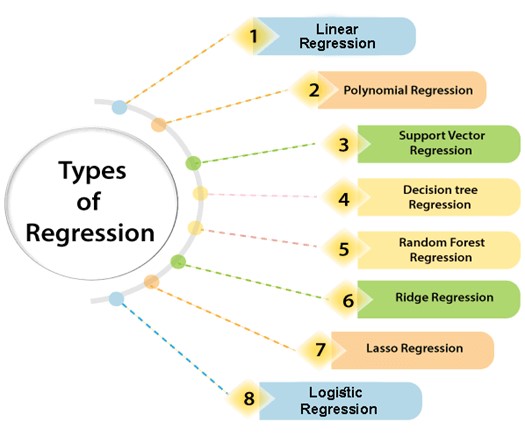
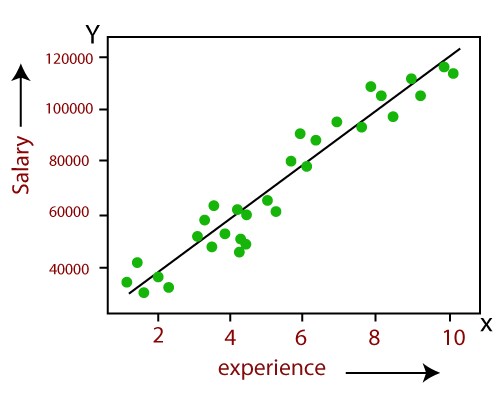
**EXPERIMENT – 03**

Aim: BUILD A MODEL USING LINEAR REGRESSION ALGORITHM ON ANY DATASET

**Linear Regression**

:

* Linear regression is a statistical regression method which is used for predictive analysis.
* It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
* It is used for solving the regression problem in machine learning.
* Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
* If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression.
* The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of the year of experience.
* Below is the mathematical equation for Linear regression:
* Y=aX+b
* Here, Y = dependent variable (target variable),
* X= Independent variable(predictor variable),
* a and b are the linear coefficients

Some popular applications of linear regression are:

* Analyzing trends and sales estimates
* Salary forecasting
* Real estate prediction

The linear regression model provides a sloped straight line representing the relationship between the variables.

Consider the below image:

* Mathematically, we can represent a linear regression as:
* y= a0+a1x+ ε
* Here,
* Y= Dependent Variable (Target Variable)
* X= Independent Variable (predictor Variable)
* a0= intercept of the line (Gives an additional degree of freedom)
* a1 = Linear regression coefficient (scale factor to each input value).
* ε = random error
* The values for x and y variables are training datasets for Linear Regression model representation.

# **Implementation of Simple Linear Regression Algorithm using Python**

Problem Statement example for Simple Linear Regression:

* Here we are taking a dataset that has two variables: salary

(dependent variable) and experience (Independent variable).

The goals of this problem is:

* We want to find out if there is any correlation between these two variables
* We will find the best fit line for the dataset.
* How the dependent variable is changing by changing the independent variable.

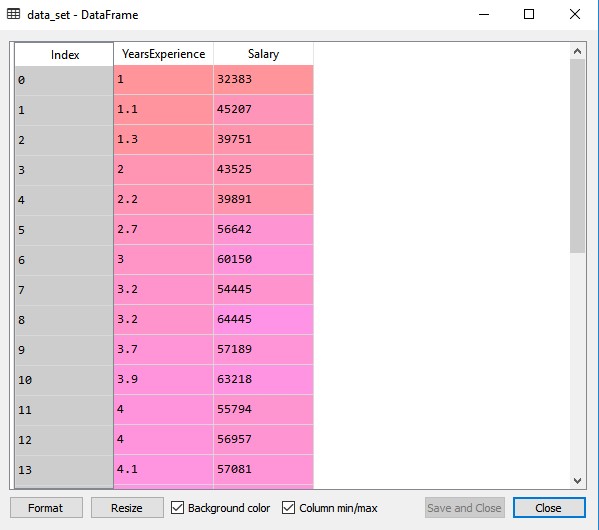
Step-1: Data Pre-processing

* The first step for creating the Simple Linear Regression model is data pre-processing.
* First, we will import the three important libraries, which will help us for loading the dataset, plotting the graphs, and creating the Simple Linear Regression model.
* **import** numpy as nm

**import** matplotlib.pyplot as mtp

**import** pandas as pd

Next, we will load the dataset into our code:



data\_set= pd.read\_csv('Salary\_Data.cs v’)

data\_set

By executing the above line of code (Shift+ENTER), we can read the dataset on our screen.

The output shows the dataset, which has two variables: Salary and Experience.

After that, we need to extract the dependent and independent variables from the given dataset.

The independent variable is years of experience, and the dependent variable is salary. Below is code for it:

x= data\_set.iloc[:, :-1].values

y= data\_set.iloc[:, 1].values

In the above lines of code, for x variable, we have taken -1 value since we want to remove the last column from the dataset. For y variable, we have taken 1 value as a parameter, since we want to extract the second column and indexing starts from the zero.

By executing the above line of code, we will get the output for xand y variable as:

* Next, we will split both variables into the test set and training set. We have 30 observations, so we will take 20 observations for the training set and 10 observations for the test set. We are splitting our dataset so that we can train our model using a training dataset and then test the model using a test dataset. The code for this is given below:
* # Splitting the dataset into training and test set.

from sklearn.model\_selection **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 1/3,

random\_s tate=0)

print(x\_train)

print(x\_test)

print(y\_train)

print(y\_test)

By executing the above code, we will get x-test, x-train and y-test, y-train dataset.

* For simple linear Regression, we will not use Feature Scaling. Because Python libraries take care of it for some cases, so we don't need to perform it here. Now, our dataset is well prepared to work on it and we are going to start building a Simple Linear Regression model for the given problem.

Step-2: Fitting the Simple Linear Regression to the Training Set:

* Now the second step is to fit our model to the training dataset. To do so, we will import the LinearRegression class of the linear\_model library from the scikit learn. After importing the class, we are going to create an object of the class named as a regressor. The code for this is given below

#Fitting the Simple Linear Regression model to the training dataset

from sklearn.linear\_model **import** LinearRegression

regressor= LinearRegression()

regressor.fit(x\_train, y\_train)

* In the above code, we have used a fit() method to fit our Simple Linear Regression object to the training set. In the fit() function, we have passed the x\_train and y\_train, which is our training dataset for the dependent and an independent variable. We have fitted our regressor object to the training set so that the model can easily learn the correlations between the predictor and target variables. After executing the above lines of code, we will get the below output.

Step: 3. Prediction of test set result:

* Dependent (salary) and an independent variable (Experience). So, now, our model is ready to predict the output for the new observations. In this step, we will provide the test dataset (new observations) to the model to check whether it can predict the correct output or not.
* We will create a prediction vector y\_pred, and x\_pred, which will contain predictions of test dataset, and prediction of training set respectively.

#Prediction of Test and Training set result y\_pred=regressor.predict(x\_test)

x\_pred= regressor.predict(x\_train)

On executing the above lines of code, two variables named y\_pred and x\_pred will generate in the variable explorer options that contain salary predictions for the training set and test set.

* You can check the variable by clicking on the variable explorer option in the IDE, and also compare the result by comparing values from y\_pred and y\_test. By comparing these values, we can check how good our model is performing.

Step: 4. visualizing the Training set results:

mtp.scatter(x\_train, y\_train, color="green")

mtp.plot(x\_train, x\_pred, color="red")

mtp.title("Salary vs Experience (Training Dataset)") mtp.xlabel("Years of Experience")

mtp.ylabel("Salary(In Rupees)")

mtp.show()

* Output:
* By executing the above lines of code, we will get the below graph plot as an output.
* In the above plot, we can see the real values observations in green dots and predicted values are covered by the red regression line. The regression line shows a correlation between the dependent and independent variable.
* The good fit of the line can be observed by calculating the difference between actual values and predicted values. But as we can see in the above plot, most of the observations are close to the regression line, hence our model is good for the training set.
* Step: 5. visualizing the Test set results:
* In the previous step, we have visualized the performance of our model on the training set. Now, we will do the same for the Test set. The complete code will remain the same as the above code, except in this, we will use x\_test, and y\_test instead of x\_train and y\_train.
* Here we are also changing the color of observations and regression line to differentiate between the two plots, but it is optional.

#visualizing the Test set results

mtp.scatter(x\_test, y\_test, color="blue") mtp.plot(x\_train, x\_pred, color="red")

mtp.title("Salary vs Experience (Test Dataset)")

mtp.xlabel("Years of Experience") mtp.ylabel("Salary(In Rupees)") mtp.show()

* Output:
* By executing the above line of code, we will get the output as:
* In the plot, there are observations given by the blue color, and prediction is given by the red regression line. As we can see, most of the observations are close to the regression line, hence we can say our Simple Linear Regression is a good model and able to make good predictions.
* We can calculate the MSE and R-square values using the following code:
* y\_pred = regressor.predict**(**X\_test**)**
* print**(**'R2 score: %.2f' % r2\_score**(**y\_test,y\_pred**))**
* # Priniting R2 Score
* print**(**'Mean squared Error :',mean\_squared\_error**(**y\_test,y\_pred**))**
* # Priniting the mean error

OUTPUT:

C:\Users\admin\Downloads\4h.PNG

EXPERIMENT-4

AIM: Build a model using Polynomial regression on any dataset

Theory:

Polynomial Regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modelled as an nth-degree polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y | x)

While in simple linear regression models the relationship is a straight line, polynomial regression allows for more flexibility by fitting a polynomial equation to the data.

When the relationship between the variables is better represented by a curve rather than a straight line, polynomial regression can capture the non-linear patterns in the data.

When the relationship is non-linear, a polynomial regression model introduces higher-degree polynomial terms.

The general form of a polynomial regression equation of degree n is:

y=0+1x+2x²+3x³......nxⁿ

where,

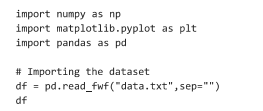
y is the dependent variable.

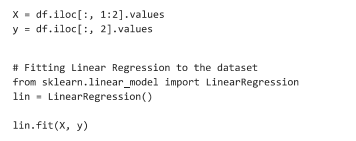
x is the independent variable.

0,1,2,3,n…..are the coefficients of the polynomial terms.

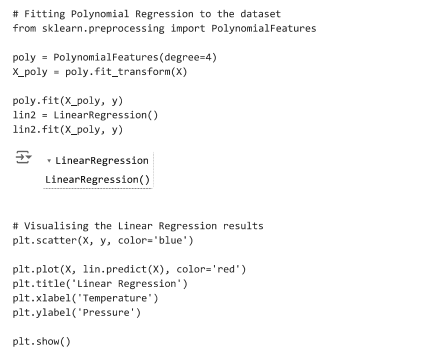
n is the degree of the polynomial.

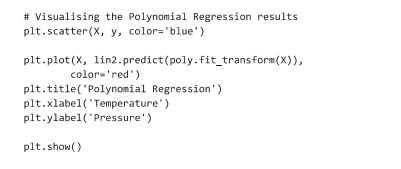
The choice of the polynomial degree (n) is a crucial aspect of polynomial regression. A higher degree allows the model to fit the training data more closely, but it may also lead to overfitting, especially if the degree is too high. Therefore, the degree should be chosen based on the complexity of the underlying relationship in the data.

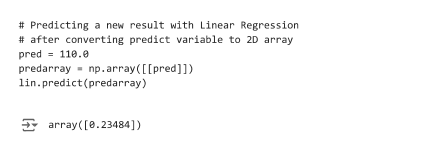


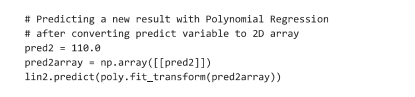










s