Parul® University
Vadodara, Gujarat
NAAC GRADE A++

Information and
Communication Technology

Chapter 2: Supervised Learning

Dr. Vinod Patidar
Associate Professor
Computer Science and Engineering

Content

- Supervised Learning
- Linear and Non-Linear Examples
- Multi-Class & Multi-Label Classification
- Linear Regression
- Multi-linear Regression
- Naïve Bayes Classifier
- Decision Trees
- ID3
- CART
- Error Bounds

Introduction to Supervised Learning

- Supervised learning is the types of machine learning in which machines are trained using well "labeled" training data, and on basis of that data, machines predict the output.
- The labeled data means some input data is already tagged with the correct output.
- In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.
- Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y).**
- In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

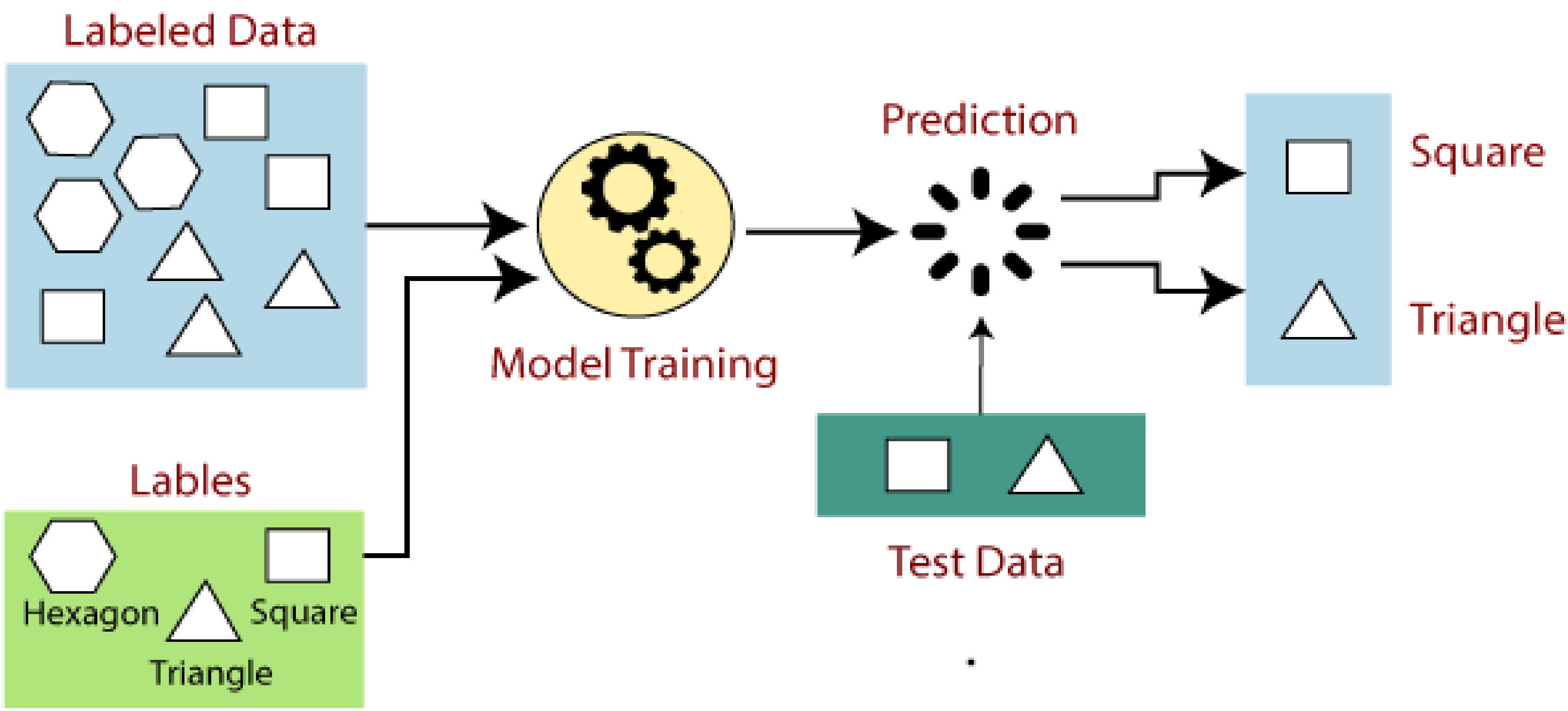
Key Points:

- Supervised learning involves training a model using labeled data.
- Labeled data consists of input features and their corresponding output labels or target values.
- The algorithm learns from the labeled data to make predictions or decisions.
- The goal is to find a mapping function that can generalize well to new, unseen data.
- Supervised learning is used for tasks like classification, regression, and prediction.

How Supervised Learning Works?

In supervised learning, models are trained using labeled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



How Supervised Learning Works?

Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labeled as a **Square**.
- If the given shape has three sides, then it will be labeled as a **triangle**.
- If the given shape has six equal sides then it will be labeled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

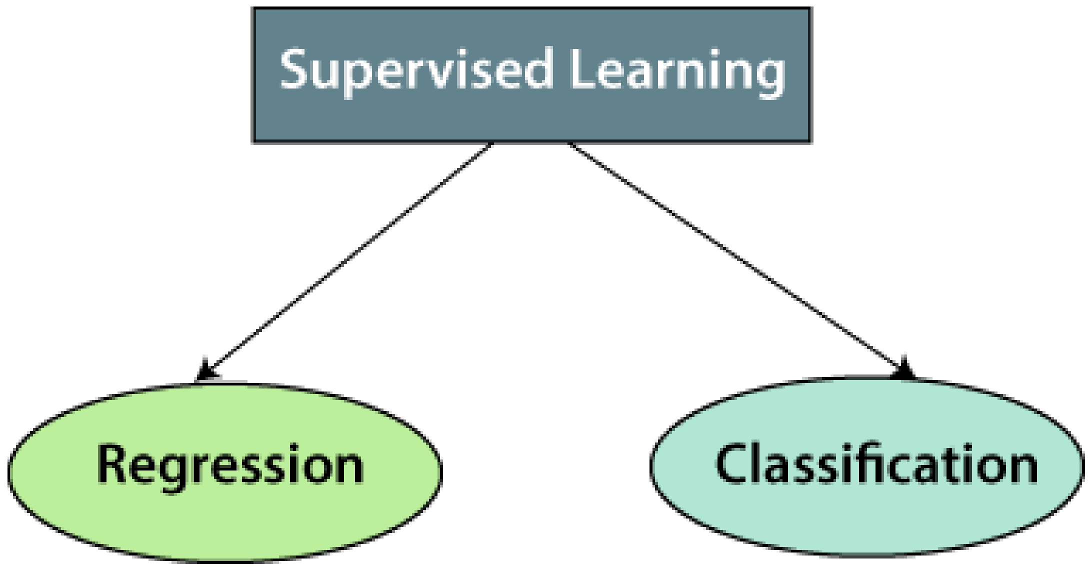
Steps Involved in Supervised Learning

- First Determine the type of training dataset
- Collect/Gather the labeled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- **Determine the input features of the training dataset**, which should have enough knowledge so that the model can accurately predict the output.
- **Determine the suitable algorithm for the model**, such as support vector machine, decision tree, etc.
- **Execute the algorithm on the training dataset**. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- **Evaluate the accuracy of the model by providing the test set**. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning

Supervised learning can be further classified into two main categories:

•



Types of supervised Machine learning Algorithms

1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

Types of supervised Machine learning Algorithms

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of Supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

Role of Labeled Data in Training :

Labeled data plays a crucial role in supervised learning as it forms the foundation for training machine learning models.

The availability of labeled data allows the algorithm to learn patterns and relationships between input features and their corresponding output labels. Here are some key points highlighting the role of labeled data in training:

1. **Providing Ground Truth:** Labeled data provides the ground truth or correct answers for the learning algorithm. It establishes the relationship between input features and their associated output labels, serving as a reference for the model to learn from.

Role of Labeled Data in Training :

2. Model Training:

Labeled data is used to train the machine learning model. During the training process, the algorithm analyzes the labeled examples, identifies patterns, and adjusts its internal parameters to make accurate predictions or decisions.

3. Supervised Learning:

Labeled data is essential for supervised learning, where the algorithm learns from labeled examples to predict outputs for unseen data. By observing the labeled data, the model can understand the underlying patterns and generalize that knowledge to make predictions on new, unlabeled instances.

Role of Labeled Data in Training :

4. Evaluating Model Performance:

Labeled data enables the evaluation of the model's performance. By comparing the predicted outputs of the model with the true labels in the labeled dataset, we can measure the accuracy, precision, recall, or other metrics to assess how well the model is performing.

5. Iterative Improvement:

Labeled data allows for iterative improvement of the model. By training the model, evaluating its performance, and analyzing prediction errors, we can refine the model, adjust its parameters, and enhance its predictive capabilities.

6. Generalization:

Labeled data helps the model generalize from the training set to new, unseen data. Through exposure to various labeled examples, the model can learn patterns and relationships that hold true across different instances, enabling it to make accurate predictions on previously unseen data.

Regression Analysis in Machine learning:

- Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.
- More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.
- It predicts continuous/real values such as **temperature, age, salary, price,** etc.

- understand the concept of regression analysis using the example

Regression Analysis in Machine learning:

Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:

Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

the company wants to do the advertisement of $200 in the year 2019 and wants to know the prediction about the sales for this year. So to solve such type of prediction problems in machine learning we need regression analysis.

Regression Analysis in Machine learning:

Example:

Some examples of regression can be as:

- Prediction of rain using temperature and other factors
- Determining Market trends
- Prediction of road accidents due to rash driving.

Terminologies Related to the Regression Analysis

Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.

Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.

Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.

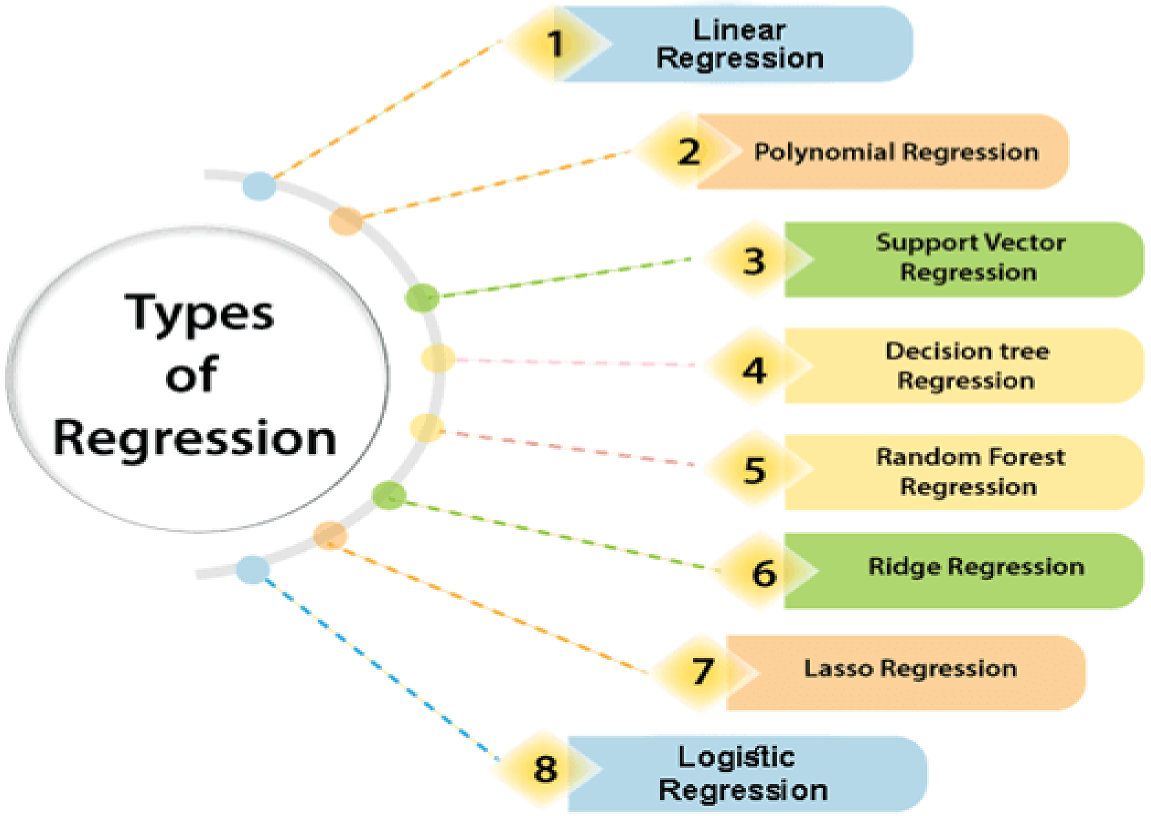
Why do we use Regression Analysis:

- Regression analysis helps in the prediction of a continuous variable.
- There are various scenarios in the real world **where we need some future predictions such as weather condition, sales prediction, marketing trends, etc.**, for such case we need some technology which can make predictions more accurately. So for such case we need Regression analysis which is a statistical method and used in machine learning and data science.

Below are some other reasons for using Regression analysis:

- Regression estimates the relationship between the target and the independent variable.
- It is used to find the trends in data.
- It helps to predict real/continuous values.
- By performing the regression, we can confidently determine the most important factor, the least important factor, and how each factor is affecting the other factors

Types of Regression:



Types of Regression:

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

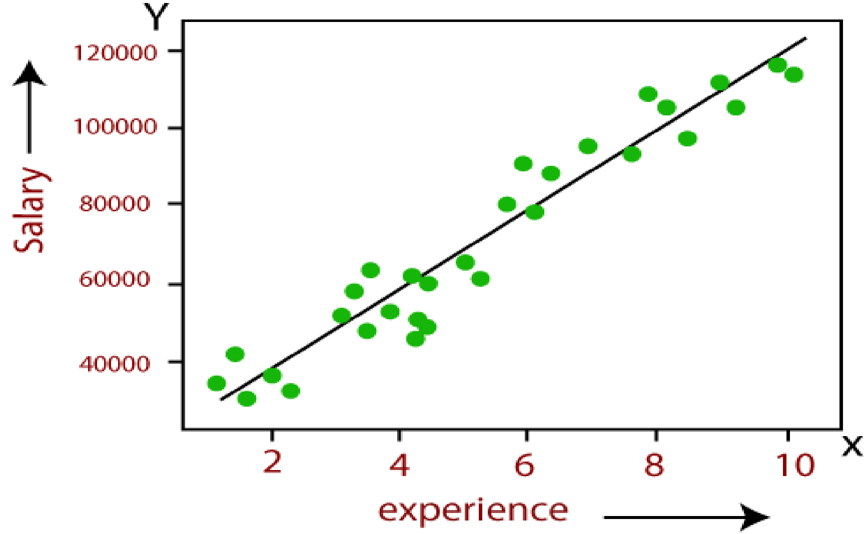
- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression:**

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**.

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**.



Linear Regression:

Below is the mathematical equation for Linear regression:

$$Y = aX + b$$

Here,

Y = dependent variables (target variables)

X= Independent variables (predictor variables)

a and b are the linear coefficients

Linear Regression:

Linear Regression Equation

$$Y = a + bx$$

$$a = \frac{[(\Sigma y)(\Sigma x^2) - (\Sigma y)(\Sigma xy)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

$$b = \frac{[n(\Sigma xy) - (\Sigma x)(\Sigma y)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

Linear Regression:

Applications of linear regression are:

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

Linear Regression:

Solved Questions on Linear Regression

Question 1: Find the linear regression equation for the given data:

x	y
3	8
9	6
5	4
3	2

Solution:

Calculating intercept and slope value.

x	y	x^2	xy
3	8	9	24
9	6	81	54
5	4	25	20
3	2	9	6
$\sum x = 20$	$\sum y = 20$	$\sum x^2 = 124$	$\sum xy = 104$

Linear Regression:

Using formula,

$$a = \frac{\sum y \sum x^2 - \sum x \sum xy}{n(\sum x^2) - (\sum x)^2}$$

a = {20 (124) – 20 (104)} / {4 (124) – 400}

a = 400/96 = 4.17

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

b = {4 (104) – 20 (20)} / {4 (124) – 400}

b = 16/96 = 0.166

So, linear regression equation is, y=a+bx => y = 4.17 + 0.166x

Types of Linear Regression:

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**

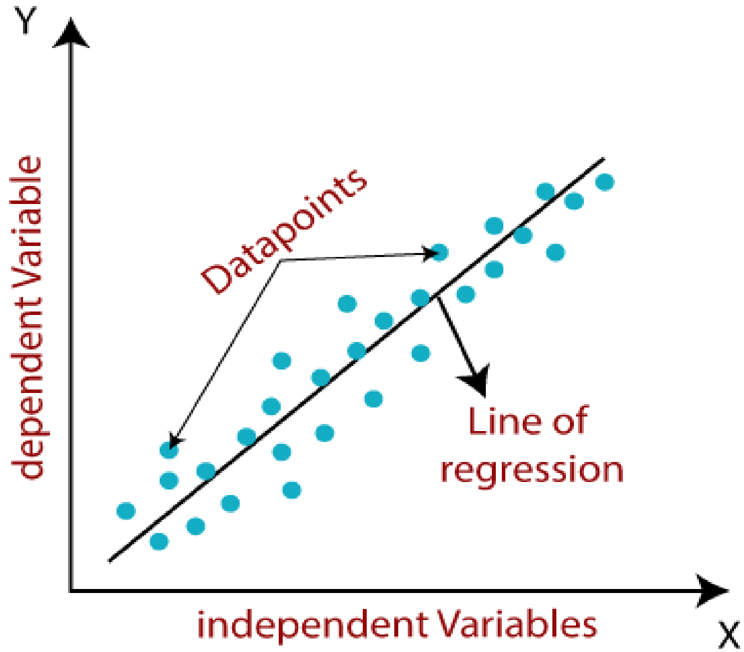
  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- **Multiple Linear regression:**

If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line:

A linear line showing the relationship between the dependent and independent variables is called a **regression line**

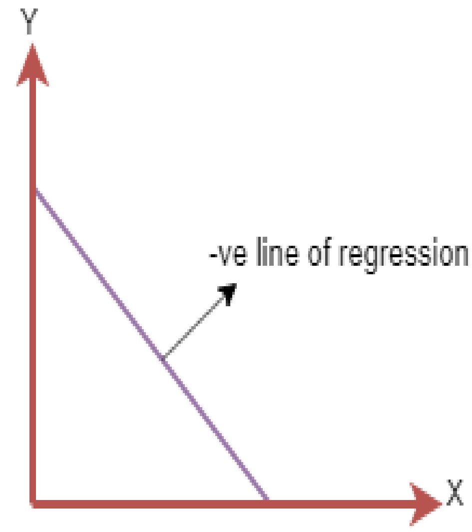


Linear Regression Line:

A regression line can show two types of relationship:

Negative Linear Relationship:

If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship



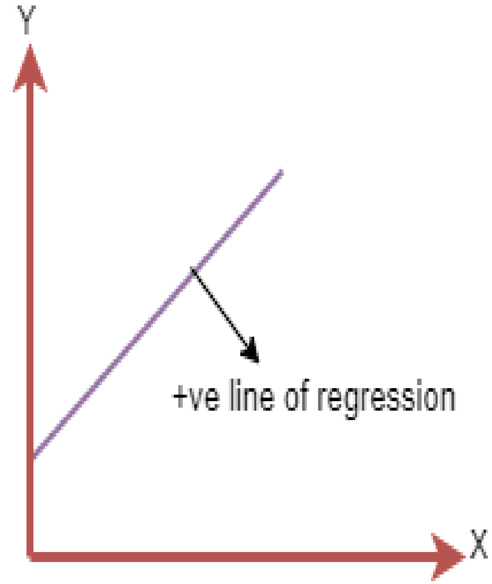
The line of equation will be: $\mathbf{Y= -a_0+a_1X}$

Linear Regression Line:

A regression line can show two types of relationship:

Positive Linear Relationship:

If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

Linear Regression :

Mathematically, we can represent a linear regression as:

$y= a_0+a_1x+ \varepsilon$

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.

Linear Regression :

Finding the best fit line:

- When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized.
- The best fit line will have the least error.
- The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use **cost function**.

Linear Regression :

Cost function-

- The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the **cost function** is used to estimate the values of the coefficient for the best fit line.

- Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.

- We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis function**.

Linear Regression :

Cost function-

For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

It can be written as:

For the above linear equation, MSE can be calculated as:

$$\text{MSE}= 1\frac{1}{N}\sum_{i=1}^{n}(y_i \ - (a_1x_i + a_0))^2$$

Where,

N=Total number of observation

Yi = Actual value

$(a1x_i+a_0)$= Predicted value.

Linear Regression :

Residuals:

The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function will be low.

Linear Regression :

Gradient Descent:

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Linear Regression :

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**. It can be achieved by **R-squared method.**

Linear Regression :

R-squared method:
R-squared is a statistical method that determines the goodness of fit.
It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
It is also called a **coefficient of determination,** or **coefficient of multiple determination** for multiple regression.

It can be calculated from the below formula:

$$\text{R-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$$

Simple Linear Regression in Machine Learning :

- **S**imple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.
- The key point in Simple Linear Regression is that the ***dependent variable must be a continuous/real value***. However, the independent variable can be measured on continuous or categorical values.

Simple Linear Regression in Machine Learning :

Simple Linear regression algorithm has mainly two objectives:

Model the relationship between the two variables. Such as the relationship between Income and expenditure, experience and Salary, etc.

Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

Simple Linear Regression in Machine Learning :

The Simple Linear Regression model can be represented using the below equation:

$y= a_0+a_1x+ \varepsilon$

Where,
a0= It is the intercept of the Regression line (can be obtained putting x=0)
a1= It is the slope of the regression line, which tells whether the line is increasing or decreasing.
ε = The error term. (For a good model it will be negligible)

Implementation of Simple Linear Regression Algorithm:

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.**

Implementation of Simple Linear Regression Algorithm:

To implement the Simple Linear regression model in machine learning using Python, we need to follow the below steps:

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
```

Implementation of Simple Linear Regression Algorithm:

Next, we will load the dataset into our code:

data_set= pd.read_csv('Salary_Data.csv')

data_set - DataFrame

Index	YearsExperience	Salary
0	1	32383
1	1.1	45207
2	1.3	39751
3	2	43525
4	2.2	39891
5	2.7	56642
6	3	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4	55794
12	4	56957
13	4.1	57081

Format Resize Background color Column min/max Save and Close Close

Implementation of Simple Linear Regression Algorithm:

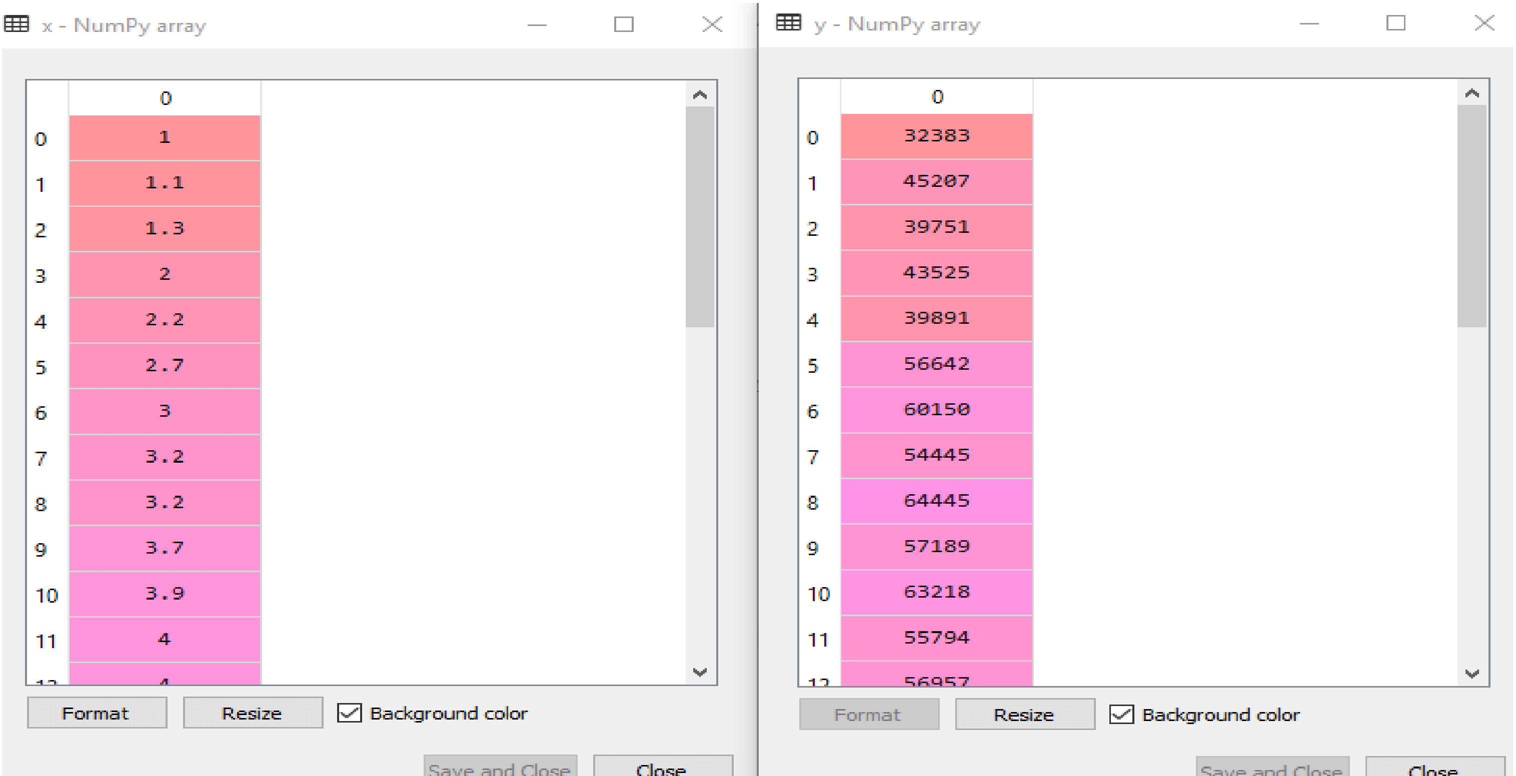
After that, we need to **extract the dependent and independent variables from the given dataset**. The independent variable is years of experience, and the dependent variable is salary. Below is code for it:

```
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

Implementation of Simple Linear Regression Algorithm:

By executing the above slide of code, we will get the output for X and Y variable as:



Implementation of Simple Linear Regression Algorithm:

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:

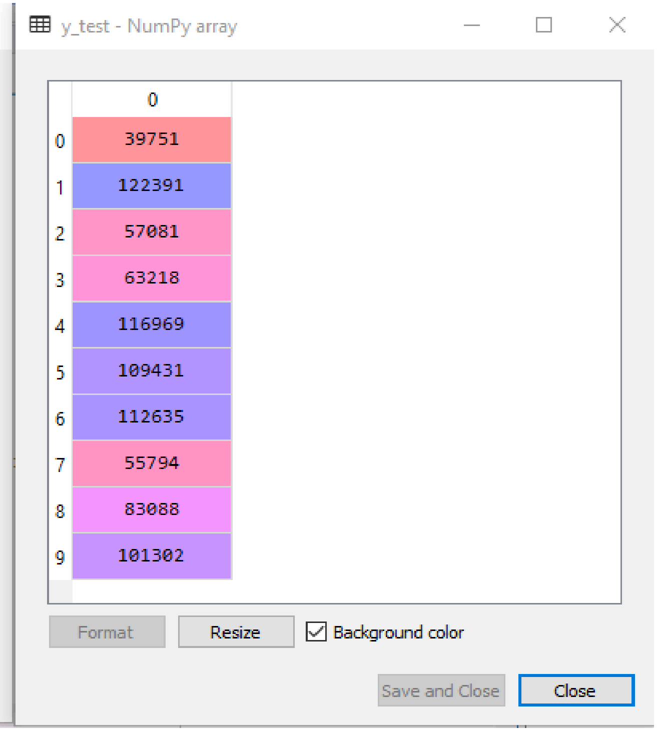
```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)
```

Implementation of Simple Linear Regression Algorithm:

By executing the above slide code, we will get x-test, x-train and y-test, y-train dataset. Consider the below images:

Test-dataset:

	0
0	1.3
1	10.3
2	4.1
3	3.9
4	9.5
5	8.7
6	9.6
7	4
8	5.3
9	7.9



	0
0	39751
1	122391
2	57081
3	63218
4	116969
5	109431
6	112635
7	55794
8	83088
9	101302

Implementation of Simple Linear Regression Algorithm:

Training Dataset:

x_train - NumPy array

	0
0	2.7
1	5.1
2	3.2
3	4.5
4	8.2
5	6.8
6	1.1
7	10.5
8	3
9	2.2
10	5.8
11	6
12	3.7

Format Resize Background color
Save and Close Close

y_train - NumPy array

	0
0	56642
1	66029
2	64445
3	61111
4	113812
5	91738
6	45207
7	121872
8	60150
9	39891
10	81363
11	93940
12	57189

Format Resize Background color
Save and Close Close

Implementation of Simple Linear Regression Algorithm:

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:

Implementation of Simple Linear Regression Algorithm:

```
#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.

Output:

```
Out[7]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

Implementation of Simple Linear Regression Algorithm:

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)
```

Implementation of Simple Linear Regression Algorithm:

Step: 4. visualizing the Training set results:

- Now in this step, we will visualize the training set result. To do so, we will use the scatter() function of the pyplot library, which we have already imported in the pre-processing step. The **scatter () function** will create a scatter plot of observations.
- In the x-axis, we will plot the Years of Experience of employees and on the y-axis, salary of employees. In the function, we will pass the real values of training set, which means a year of experience x_train, training set of Salaries y_train, and color of the observations. Here we are taking a green color for the observation, but it can be any color as per the choice.
- Now, we need to plot the regression line, so for this, we will use the **plot() function** of the pyplot library. In this function, we will pass the years of experience for training set, predicted salary for training set x_pred, and color of the line.

Implementation of Simple Linear Regression Algorithm:

Next, we will give the title for the plot. So here, we will use the **title()** function of the **pyplot** library and pass the name ("Salary vs Experience (Training Dataset)". After that, we will assign labels for x-axis and y-axis using **xlabel() and ylabel() function**.

Finally, we will represent all above things in a graph using show(). The code is given below:

```
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()
```

Implementation of Simple Linear Regression Algorithm:

Output:

By executing the above lines of code, we will get the below graph plot as an output.

Implementation of Simple Linear Regression Algorithm:

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()
```

Implementation of Simple Linear Regression Algorithm:

Output:

By executing the above line of code, we will get the output as:

Multiple Linear Regression:

- Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable
- Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable.

Example:

- Prediction of CO_2 emission based on engine size and number of cylinders in a car.

Multiple Linear Regression:

Key points about MLR:

- For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- Each feature variable must model the linear relationship with the dependent variable.
- MLR tries to fit a regression line through a multidimensional space of data-points.

Multiple Linear Regression:

MLR equation:
In Multiple Linear Regression, the target variable(Y) is a linear combination of multiple predictor variables x_1, x_2, x_3, ...,x_n. Since it is an enhancement of Simple Linear Regression, so the same is applied for the multiple linear regression equation, the equation becomes:

$$Y = b0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots\ldots\ldots\ldots + b_nx_n$$

Where,
Y= Output/Response variable
b_0, b_1, b_2, b_3 , b_n....= Coefficients of the model.
x_1, x_2, x_3, x_4,...= Various Independent/feature variable

Implementation of Multiple Linear Regression model :

To implement MLR using Python, we have below problem:

Problem Description:

We have a dataset of **50 start-up companies**. This dataset contains five main information: **R&D Spend, Administration Spend, Marketing Spend, State, and Profit for a financial year**.

- Our goal is to create a model that can easily **determine which company has a maximum profit,** and which is the **most affecting factor for the profit of a company**.

Since we need to find the Profit, so it is the dependent variable, and the other four variables are independent variables. Below are the main steps of deploying the MLR model:

1.Data Pre-processing Steps

2.Fitting the MLR model to the training set

3.Predicting the result of the test set

Implementation of Multiple Linear Regression model :

Step-1: Data Pre-processing Step:
The very first step is data pre-processing, which we have already discussed previous class.
This process contains the below steps:

Importing libraries: Firstly we will import the library which will help in building the model. Below is the code for it:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

Implementation of Multiple Linear Regression model :

Importing dataset: Now we will import the dataset(50_CompList), which contains all the variables. Below is the code for it:

```
#importing datasets
data_set= pd.read_csv('50_CompList.csv')
```

Output: We will get the dataset as:

data_set - DataFrame

Index	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349	136898	471784	New York	192262
1	162598	151378	443899	California	191792
2	153442	101146	407935	Florida	191050
3	144372	118672	383200	New York	182902
4	142107	91391.8	366168	Florida	166188
5	131877	99814.7	362861	New York	156991
6	134615	147199	127717	California	156123
7	130298	145530	323877	Florida	155753
8	120543	148719	311613	New York	152212
9	123335	108679	304982	California	149760
10	101913	110594	229161	Florida	146122
11	100672	91790.6	249745	California	144259
12	93863.8	127320	249839	Florida	141586
13	91992.4	135495	252665	California	134307

Format | Resize | Background color | Column min/max | Save and Close | Close

above output, we can clearly see that there are five variables, in which four variables are continuous and one is categorical variable.

Implementation of Multiple Linear Regression model :

Extracting dependent and independent Variables:

```
#Extracting Independent and dependent Variable
x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 4].values
```

Implementation of Multiple Linear Regression model :

Encoding Dummy Variables:

As we have one categorical variable (State), which cannot be directly applied to the model, so we will encode it. To encode the categorical variable into numbers, we will use the **LabelEncoder** class. But it is not sufficient because it still has some relational order, which may create a wrong model. So in order to remove this problem, we will use **OneHotEncoder**, which will create the dummy variables. Below is code for it:

```
#Catgorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_x= LabelEncoder()
x[:, 3]= labelencoder_x.fit_transform(x[:,3])
onehotencoder= OneHotEncoder(categorical_features= [3])
x= onehotencoder.fit_transform(x).toarray()
```

Implementation of Multiple Linear Regression model :

Encoding Dummy Variables:

Here we are only encoding one independent variable, which is state as other variables are continuous.

Output:- →

x - NumPy array

	0	1	2	3	4	5
0	0	0	1	165349	136898	471784
1	1	0	0	162598	151378	443899
2	0	1	0	153442	101146	407935
3	0	0	1	144372	118672	383200
4	0	1	0	142107	91391.8	366168
5	0	0	1	131877	99814.7	362861
6	1	0	0	134615	147199	127717
7	0	1	0	130298	145530	323877
8	0	0	1	120543	148719	311613
9	1	0	0	123335	108679	304982
10	0	1	0	101913	110594	229161
11	1	0	0	100672	91790.6	249745
12	0	1	0	93863.8	127320	249839
13	1	0	0	91992.4	135495	252665

Format Resize ☑ Background color

Implementation of Multiple Linear Regression model :

As we can see in the above output, the state column has been converted into dummy variables (0 and 1). **Here each dummy variable column is corresponding to the one State**. We can check by comparing it with the original dataset. The first column corresponds to the **California State**, the second column corresponds to the **Florida State**, and the third column corresponds to the **New York State**.

Now, we are writing a single line of code just to avoid the dummy variable trap:

```
#avoiding the dummy variable trap:
x = x[:, 1:]
```

If we do not remove the first dummy variable, then it may introduce multicollinearity in the model.

Implementation of Multiple Linear Regression model :

x - NumPy array

	0	1	2	3	4
0	0	1	165349	136898	471784
1	0	0	162598	151378	443899
2	1	0	153442	101146	407935
3	0	1	144372	118672	383200
4	1	0	142107	91391.8	366168
5	0	1	131877	99814.7	362861
6	0	0	134615	147199	127717
7	1	0	130298	145530	323877
8	0	1	120543	148719	311613
9	0	0	123335	108679	304982
10	1	0	101913	110594	229161
11	0	0	100672	91790.6	249745
12	1	0	93863.8	127320	249839

Format | Resize | Background color

Save and Close | Close

Implementation of Multiple Linear Regression model :

Now we will split the dataset into training and test set. 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= 0.2, random_state=0)
```

Implementation of Multiple Linear Regression model :

Step: 2- Fitting our MLR model to the Training set:
Now, we have well prepared our dataset in order to provide training, which means we will fit our regression model to the training set. It will be similar to as we did in [Simple Linear Regression]() model. The code for this will be:

```
#Fitting the MLR model to the training set:
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
```

Out[9]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

Implementation of Multiple Linear Regression model :

Step: 3- Prediction of Test set results:
The last step for our model is checking the performance of the model. We will do it by predicting the test set result. For prediction, we will create a **y_pred** vector. Below is the code for it:

```
#Predicting the Test set result;
y_pred= regressor.predict(x_test)
```

By executing the above lines of code, a new vector will be generated under the variable explorer option. We can test our model by comparing the predicted values and test set values.

Implementation of Multiple Linear Regression model :

We can also check the score for training dataset and test dataset. Below is the code for it:

```
print('Train Score: ', regressor.score(x_train, y_train))
print('Test Score: ', regressor.score(x_test, y_test))
```

Output: The score is:

Train Score: 0.9501847627493607 Test Score: 0.9347068473282446

The above score tells that our model is 95% accurate with the training dataset and 93% accurate with the test dataset.

Application of Multiple Linear Regression:

There are mainly two applications of Multiple Linear Regression:

1. Effectiveness of Independent variable on prediction:

2. Predicting the impact of changes:

Differences Between Linear and Multilinear Regression:

Number of Independent Variables:

- **Linear Regression**: Involves one independent variable.
- **Multilinear Regression**: Involves multiple independent variables.

Complexity:

- **Linear Regression**: Simpler, easier to visualize and interpret.
- **Multilinear Regression**: More complex, requires more data to estimate multiple parameters.

Use Cases:

- **Linear Regression**: Suitable for simple scenarios where the outcome is influenced by a single factor.
- **Multilinear Regression**: Suitable for more complex scenarios where the outcome is influenced by multiple factors.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/