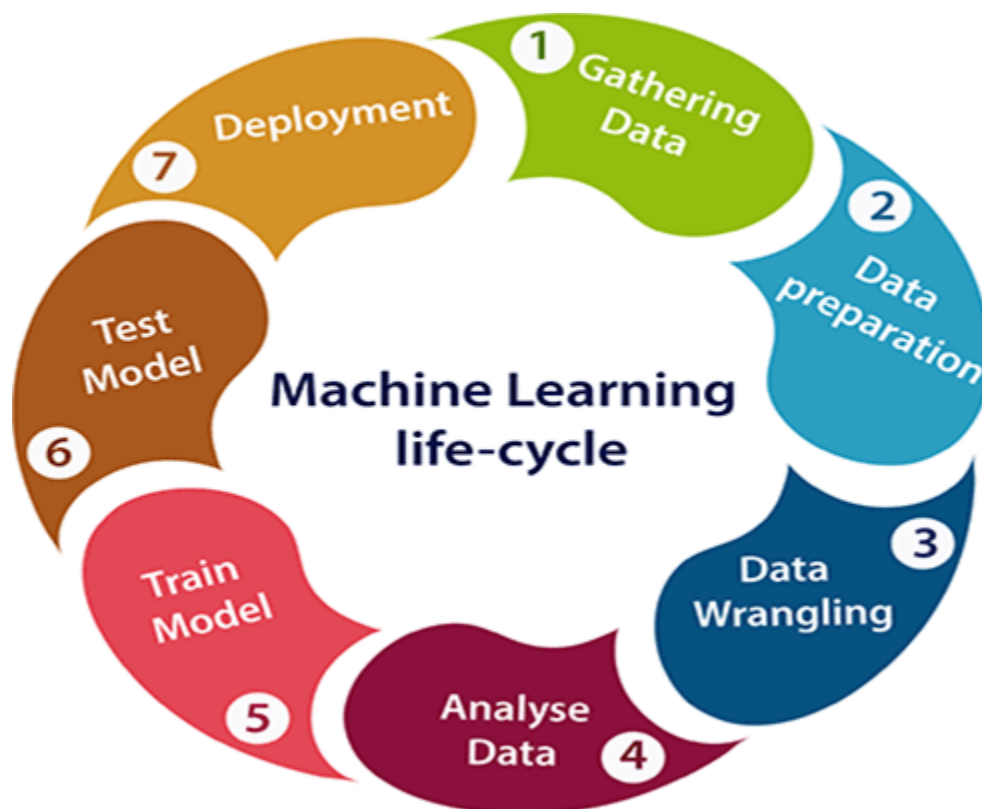


Machine learning Life cycle

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:

- **Gathering Data**
- **Data preparation**
- **Data Wrangling**
- **Analyse Data**
- **Train the model**
- **Test the model**
- **Deployment**



The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

1. Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**

By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

- **Data exploration:**

It is used to understand the nature of data that we have to work with. We need

to understand the characteristics, format, and quality of data.

A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

- **Data pre-processing:**

Now the next step is preprocessing of data for its analysis.

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
 - **Building models**
 - **Review the result**
-

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification, Regression, Cluster analysis, Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Supervised Machine Learning

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled 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.

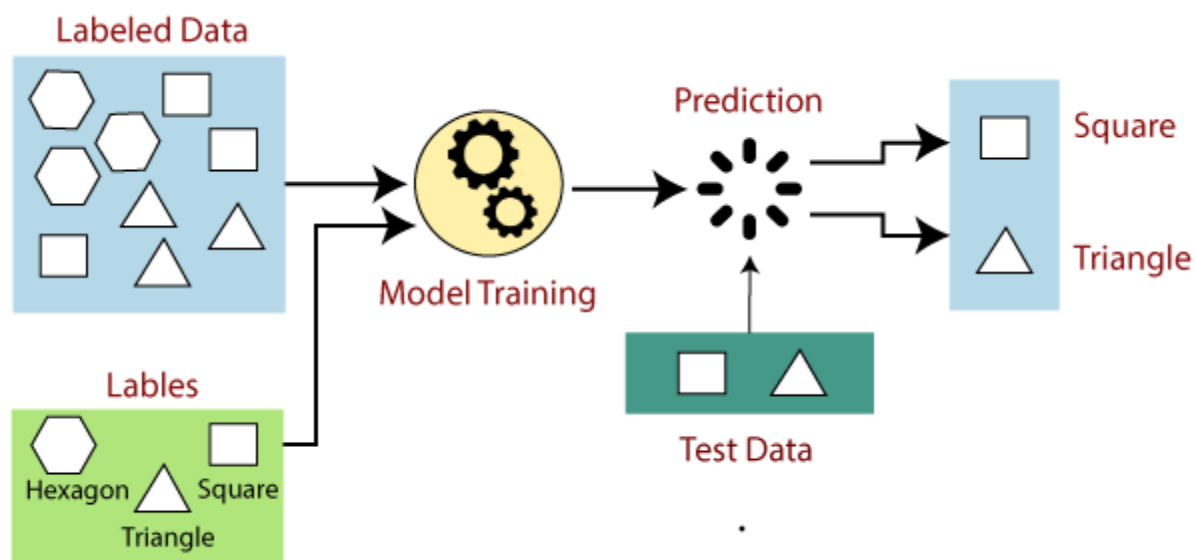
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled 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:



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 labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

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

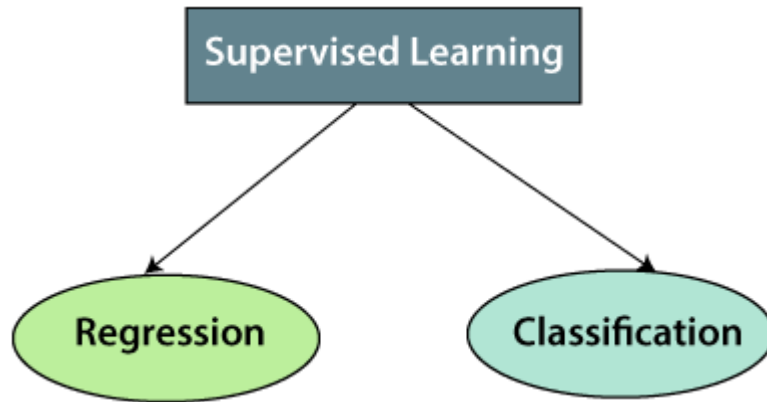
The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled 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 Algorithms:

Supervised learning can be further divided into two types of problems:



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

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.

Unsupervised Machine Learning

In the previous topic, we learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.**

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image

features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.



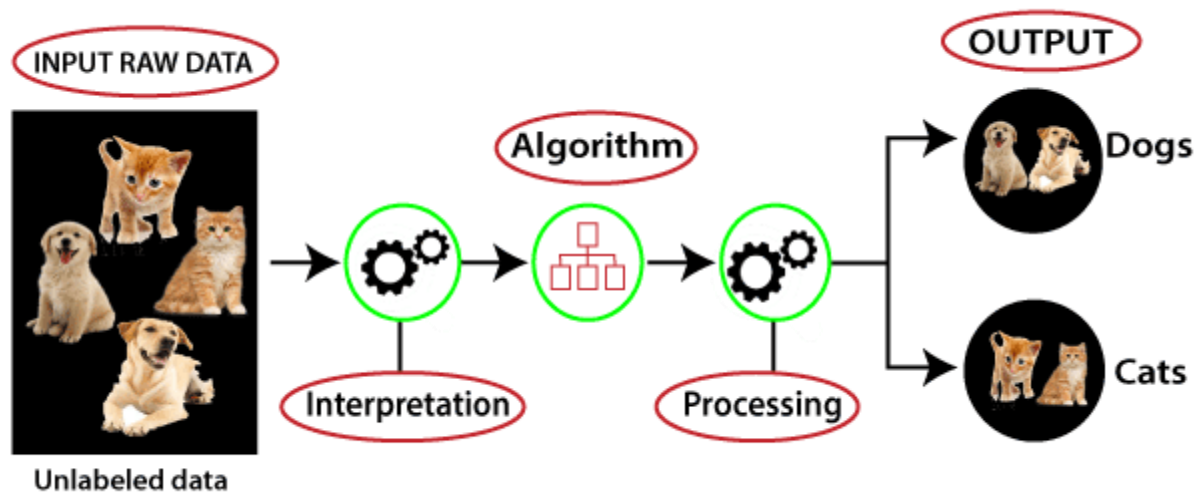
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

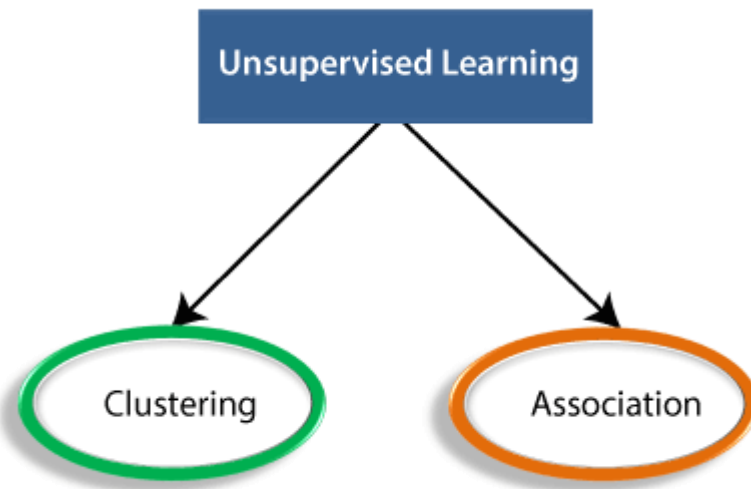


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering:** Clustering is a method of grouping the objects into clusters such that objects with most similarities remain in a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchical clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principal Component Analysis**

- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

Advantages of Unsupervised Learning

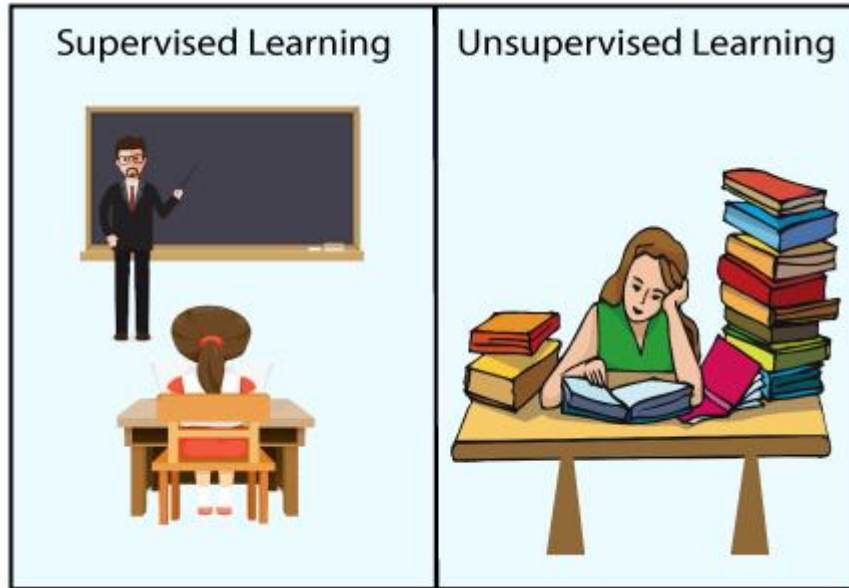
- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Difference between Supervised and Unsupervised Learning

Supervised and Unsupervised learning are the two techniques of machine learning. But both the techniques are used in different scenarios and with different datasets. Below the explanation of both learning methods along with their difference table is given.



Supervised Machine Learning:

Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: **Classification** and **Regression**.

Learn more [Supervised Machine Learning](#)

Example: Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.

Unsupervised Machine Learning:

Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial Intelligence as in this, we first train the model for	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a

Learn more [Unsupervised Machine Learning](#)

each data, and then only it can predict the correct output.	child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

Unsupervised learning can be used for two types of problems: **Clustering** and **Association**.

Example: To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

The main differences between Supervised and Unsupervised learning are given below:

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.

ADVERTISEMENT

We can understand the concept of regression analysis using the below example:

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	??

Now, 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 is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, ***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."*** The distance between datapoints and line tells whether a model has captured a strong relationship or not.

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.
- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

Why do we use Regression Analysis?

As mentioned above, 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:

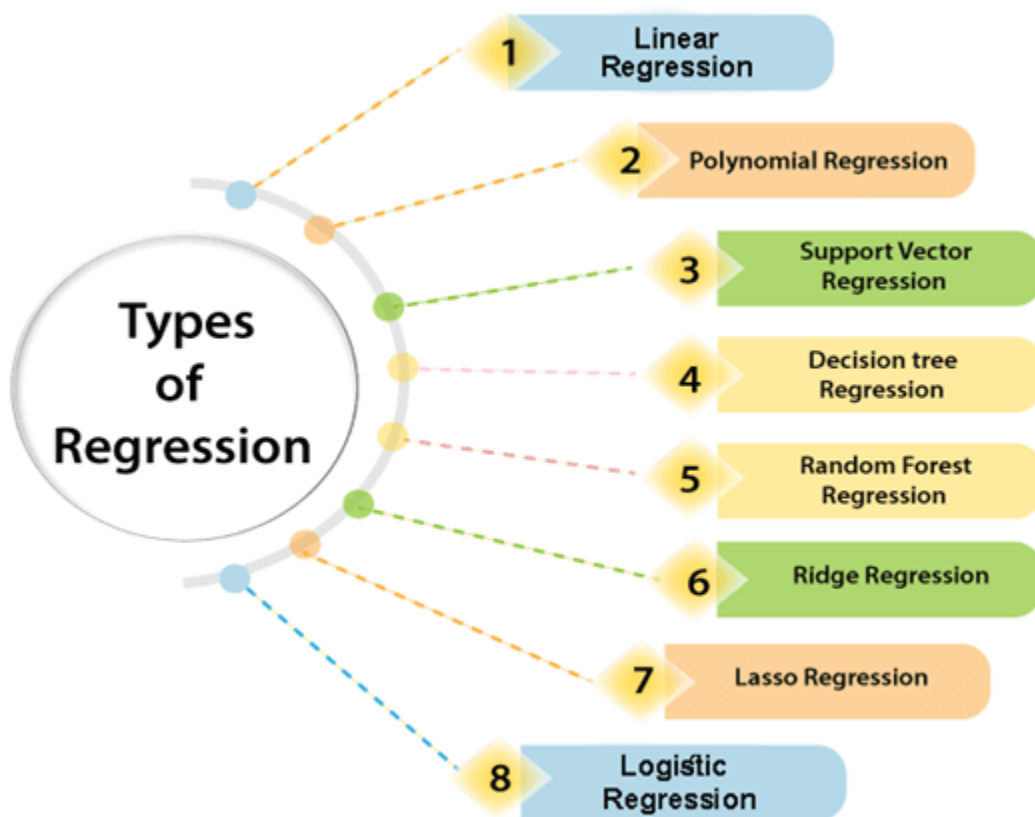
ADVERTISEMENT

- 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

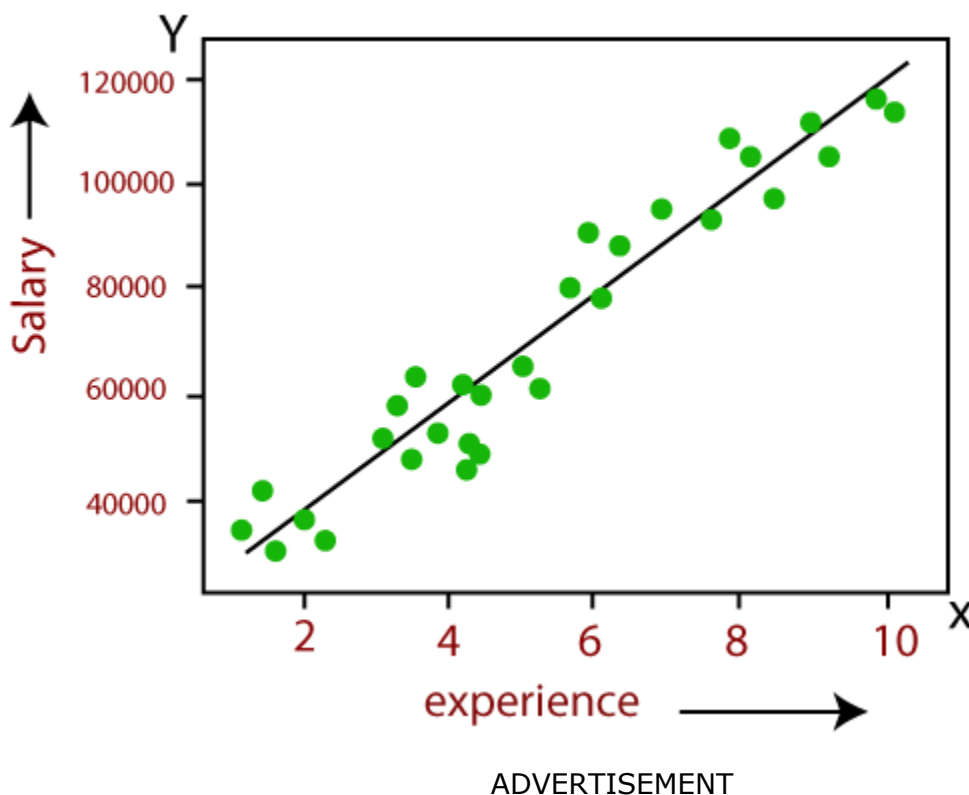
There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important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

1. $Y = aX + b$

**Here, Y = dependent variables (target variables),
X= Independent variables (predictor variables),
a and b are the linear coefficients**

Some popular applications of linear regression are:

- **Analyzing trends and sales estimates**
- **Salary forecasting**
- **Real estate prediction**
- **Arriving at ETAs in traffic.**

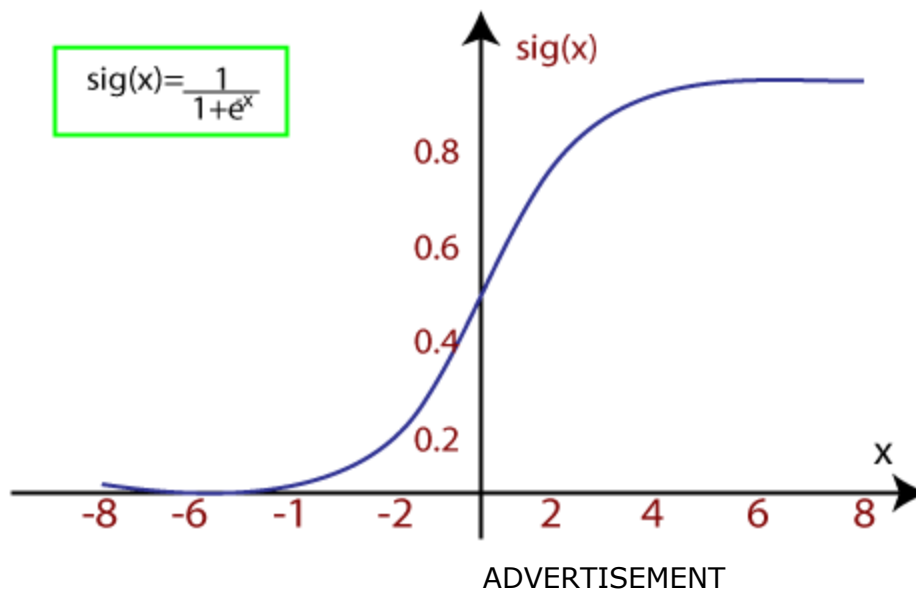
Logistic Regression:

- Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In **classification problems**, we have dependent variables in a binary or discrete format such as 0 or 1.
- Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.
- It is a predictive analysis algorithm which works on the concept of probability.
- Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used.
- Logistic regression uses **sigmoid function** or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- $f(x)$ = Output between the 0 and 1 value.
- x = input to the function
- e = base of natural logarithm.

When we provide the input values (data) to the function, it gives the S-curve as follows:



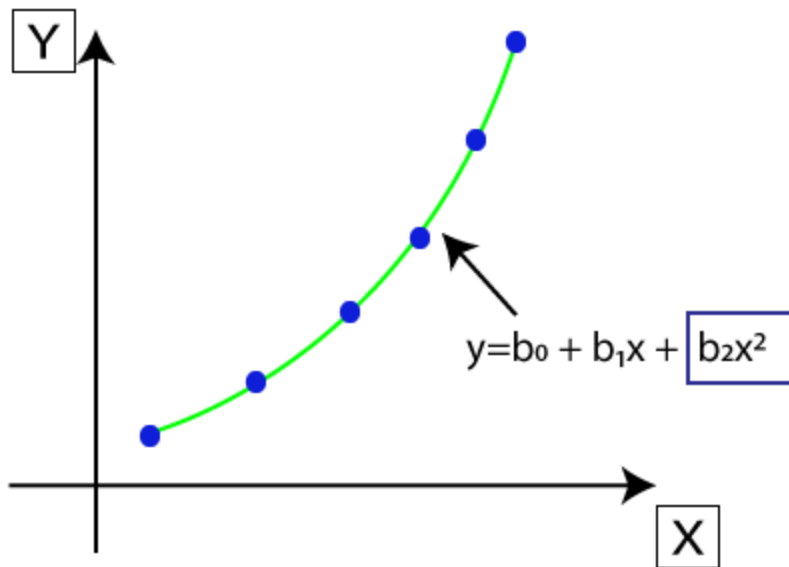
- It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0.

There are three types of logistic regression:

- **Binary(0/1, pass/fail)**
- **Multi(cats, dogs, lions)**
- **Ordinal(low, medium, high)**

Polynomial Regression:

- Polynomial Regression is a type of regression which models the **non-linear dataset** using a linear model.
- It is similar to multiple linear regression, but it fits a non-linear curve between the value of x and corresponding conditional values of y.
- Suppose there is a dataset which consists of datapoints which are present in a non-linear fashion, so for such case, linear regression will not best fit to those datapoints. To cover such datapoints, we need Polynomial regression.
- **In Polynomial regression, the original features are transformed into polynomial features of given degree and then modeled using a linear model.** Which means the datapoints are best fitted using a polynomial line.



- The equation for polynomial regression also derived from linear regression equation that means Linear regression equation $Y = b_0 + b_1x$, is transformed into Polynomial regression equation $Y = b_0 + b_1x + b_2x^2 + b_3x^3 + \dots + b_nx^n$.
- Here Y is the **predicted/target output**, **b_0, b_1, \dots, b_n are the regression coefficients**. x is our **independent/input variable**.
- The model is still linear as the coefficients are still linear with quadratic

Note: This is different from Multiple Linear regression in such a way that in Polynomial regression, a single element has different degrees instead of multiple variables with the same degree.

Support Vector Regression:

Support Vector Machine is a supervised learning algorithm which can be used for regression as well as classification problems. So if we use it for regression problems, then it is termed as Support Vector Regression.

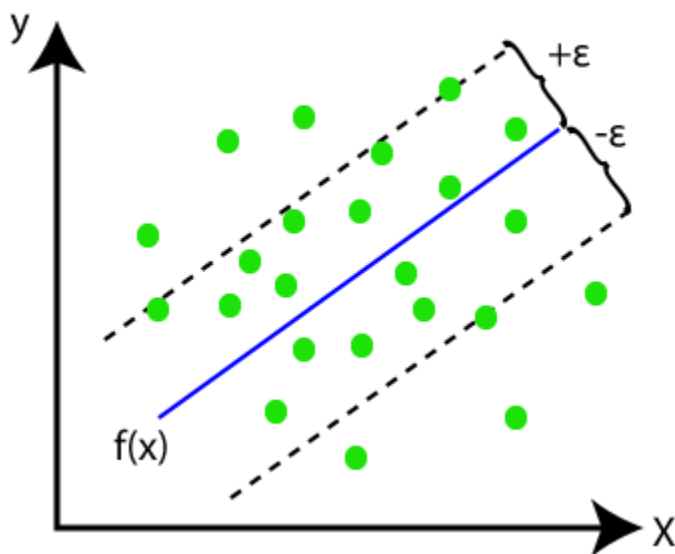
ADVERTISEMENT

Support Vector Regression is a regression algorithm which works for continuous variables. Below are some keywords which are used in **Support Vector Regression**:

- **Kernel:** It is a function used to map a lower-dimensional data into higher dimensional data.

- **Hyperplane:** In general SVM, it is a separation line between two classes, but in SVR, it is a line which helps to predict the continuous variables and cover most of the datapoints.
- **Boundary line:** Boundary lines are the two lines apart from hyperplane, which creates a margin for datapoints.
- **Support vectors:** Support vectors are the datapoints which are nearest to the hyperplane and opposite class.

In SVR, we always try to determine a hyperplane with a maximum margin, so that maximum number of datapoints are covered in that margin. ***The main goal of SVR is to consider the maximum datapoints within the boundary lines and the hyperplane (best-fit line) must contain a maximum number of datapoints.*** Consider the below image:

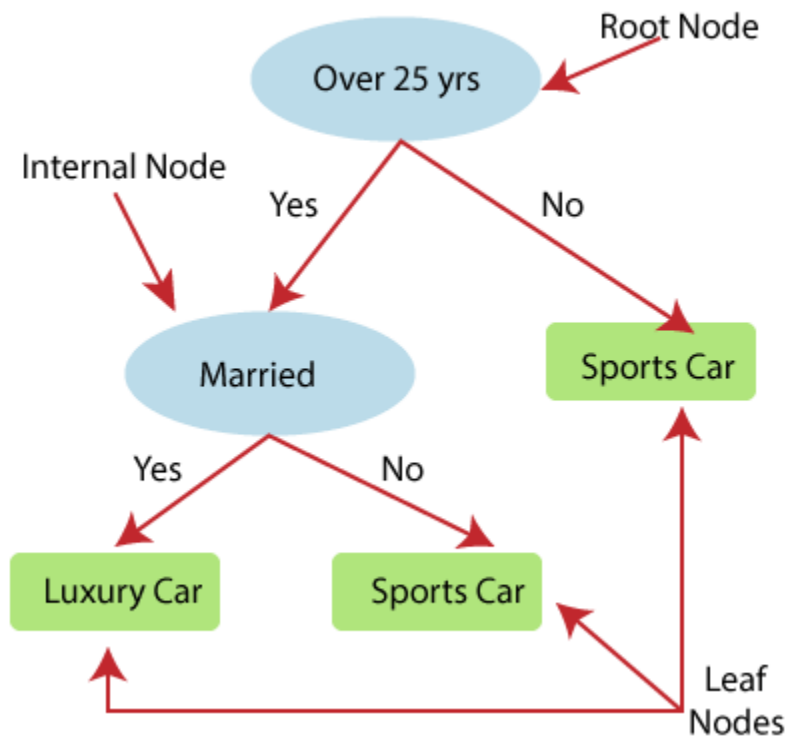


Here, the blue line is called hyperplane, and the other two lines are known as boundary lines.

Decision Tree Regression:

- Decision Tree is a supervised learning algorithm which can be used for solving both classification and regression problems.
- It can solve problems for both categorical and numerical data

- Decision Tree regression builds a tree-like structure in which each internal node represents the "test" for an attribute, each branch represent the result of the test, and each leaf node represents the final decision or result.
- A decision tree is constructed starting from the root node/parent node (dataset), which splits into left and right child nodes (subsets of dataset). These child nodes are further divided into their children node, and themselves become the parent node of those nodes. Consider the below image:

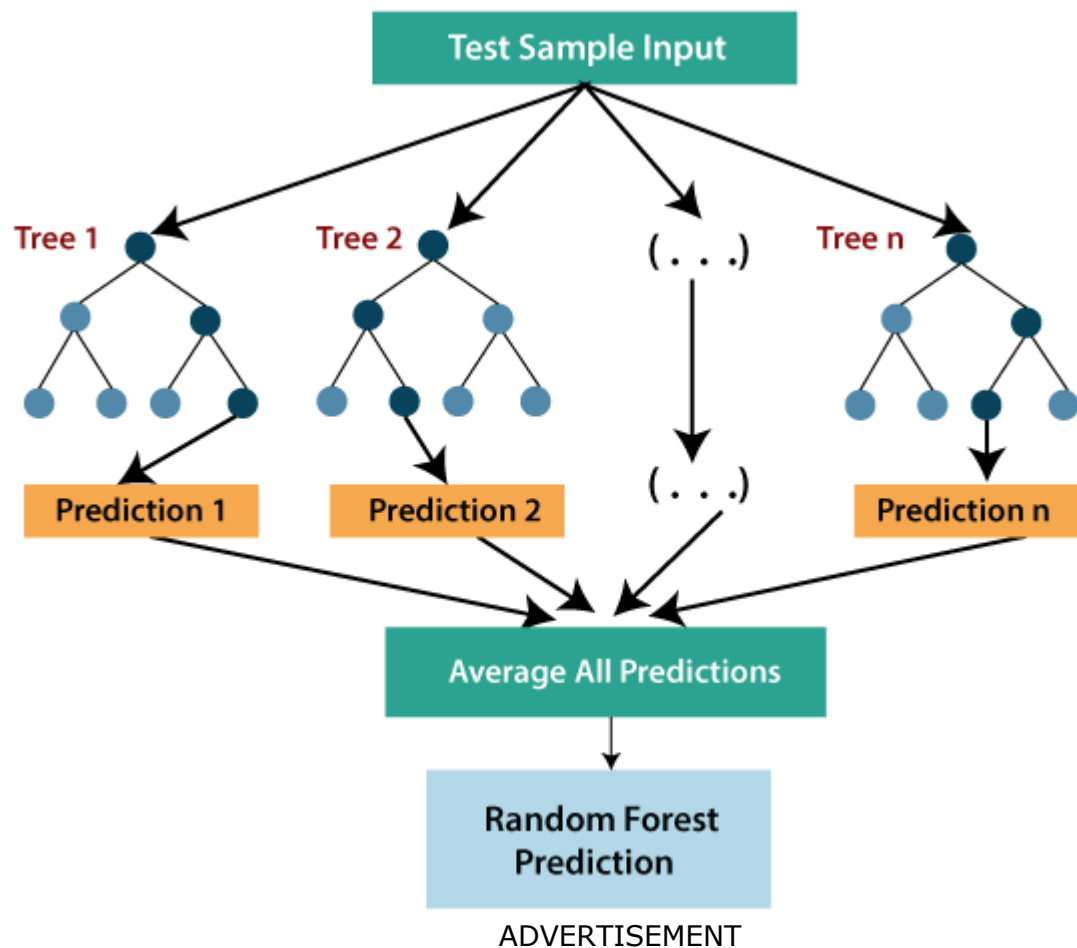


Above image showing the example of Decision Tree regression, here, the model is trying to predict the choice of a person between Sports cars or Luxury car.

- Random forest is one of the most powerful supervised learning algorithms which is capable of performing regression as well as classification tasks.
- The Random Forest regression is an ensemble learning method which combines multiple decision trees and predicts the final output based on the average of each tree output. The combined decision trees are called as base models, and it can be represented more formally as:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \dots$$

- Random forest uses **Bagging or Bootstrap Aggregation** technique of ensemble learning in which aggregated decision tree runs in parallel and do not interact with each other.
- With the help of Random Forest regression, we can prevent Overfitting in the model by creating random subsets of the dataset.



Ridge Regression:

- Ridge regression is one of the most robust versions of linear regression in which a small amount of bias is introduced so that we can get better long term predictions.
- The amount of bias added to the model is known as **Ridge Regression penalty**. We can compute this penalty term by multiplying with the lambda to the squared weight of each individual features.

- The equation for ridge regression will be:

$$L(x, y) = \text{Min}(\sum_{i=1}^n (y_i - w_i x_i)^2 + \lambda \sum_{i=1}^n (w_i)^2)$$

- A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.
- Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as **L2 regularization**.
- It helps to solve the problems if we have more parameters than samples.

Lasso Regression:

- Lasso regression is another regularization technique to reduce the complexity of the model.
- It is similar to the Ridge Regression except that penalty term contains only the absolute weights instead of a square of weights.
- Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0.
- It is also called as **L1 regularization**. The equation for Lasso regression will be:

$$L(x, y) = \text{Min}(\sum_{i=1}^n (y_i - w_i x_i)^2 + \lambda \sum_{i=1}^n |w_i|)$$

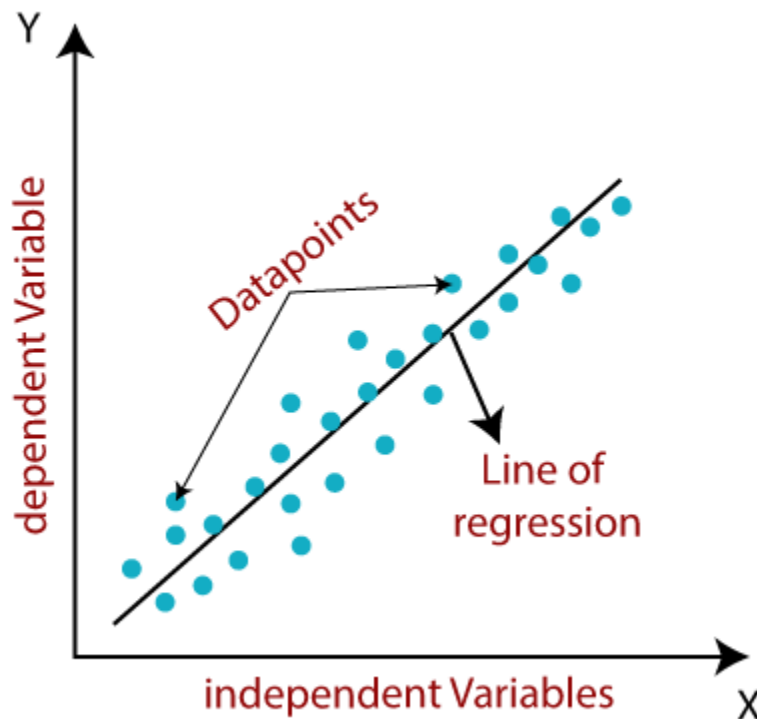
Linear Regression in Machine Learning

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price**, etc.

ADVERTISEMENT

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

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:

$$y = a_0 + a_1x + \varepsilon$$

Here,

ADVERTISEMENT

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a_0 = intercept of the line (Gives an additional degree of freedom)

a_1 = 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.

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:**

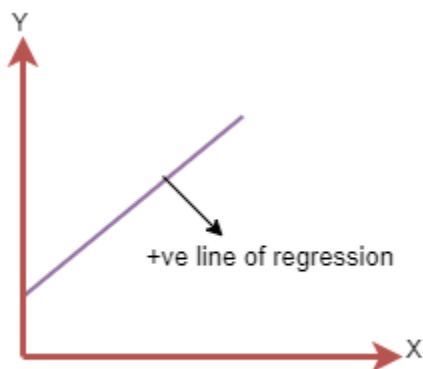
If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**

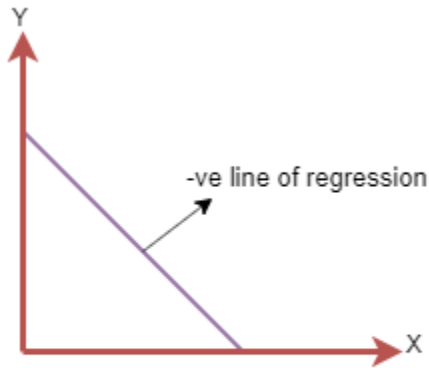
If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y = a_0 + a_1X$

- **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: $Y = -a_0 + a_1x$

ADVERTISEMENT

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.

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**.

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:

$$MSE = \frac{1}{N} \sum_{i=1}^n (y_i - (a_1 x_i + a_0))^2$$

Where,

ADVERTISEMENT

N=Total number of observation

Y_i = Actual value

(a₁x_i+a₀)= Predicted value.

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.

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.

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 below method:

1. 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:

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

Assumptions of Linear Regression

Below are some important assumptions of Linear Regression. These are some formal checks while building a Linear Regression model, which ensures to get the best possible result from the given dataset.

ADVERTISEMENT

- **Linear relationship between the features and target:**
Linear regression assumes the linear relationship between the dependent and independent variables.
- **Small or no multicollinearity between the features:**
Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.
- **Homoscedasticity Assumption:**
Homoscedasticity is a situation when the error term is the same for all the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.
- **Normal distribution of error terms:**
Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.
It can be checked using the **q-q plot**. If the plot shows a straight line without any deviation, which means the error is normally distributed.
- **No autocorrelations:**
The linear regression model assumes no autocorrelation in error terms. If there

will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

Simple Linear Regression in Machine Learning

Simple 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.

ADVERTISEMENT

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 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 Model:

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

$$y = a_0 + a_1x + \epsilon$$

Where,

a_0 = It is the intercept of the Regression line (can be obtained putting $x=0$)

a_1 = 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 using Python

Problem Statement example for Simple Linear Regression:

ADVERTISEMENT

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 dependent variable.**

In this section, we will create a Simple Linear Regression model to find out the best fitting line for representing the relationship between these two variables.

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](#). We have already done it earlier in this tutorial. But there will be some changes, which are given in the below steps:

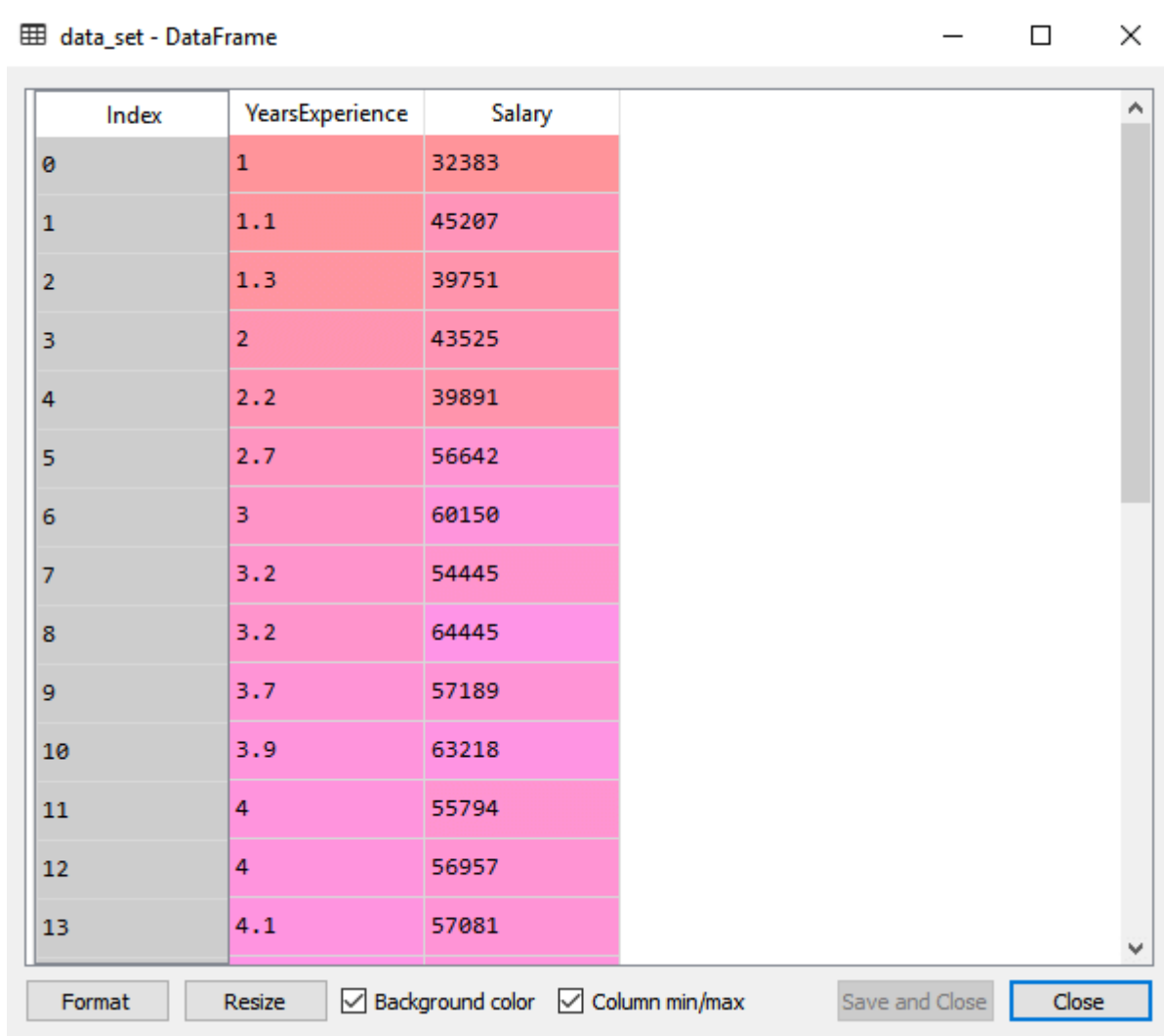
- First, we will import the three important libraries, which will help us for loading the dataset, plotting the graphs, and creating the Simple Linear Regression model.

1. **import** numpy as nm
2. **import** matplotlib.pyplot as mtp
3. **import** pandas as pd

- Next, we will load the dataset into our code:

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

By executing the above line of code (ctrl+ENTER), we can read the dataset on our Spyder IDE screen by clicking on the variable explorer option.



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

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

Note: In Spyder IDE, the folder containing the code file must be saved as a working directory, and the dataset or csv file should be in the same folder.

- 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:

1. `x= data_set.iloc[:, :-1].values`
2. `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.

ADVERTISEMENT

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

	0
0	1
1	1.1
2	1.3
3	2
4	2.2
5	2.7
6	3
7	3.2
8	3.2
9	3.7
10	3.9
11	4
12	4

	0
0	32383
1	45207
2	39751
3	43525
4	39891
5	56642
6	60150
7	54445
8	64445
9	57189
10	63218
11	55794
12	56957

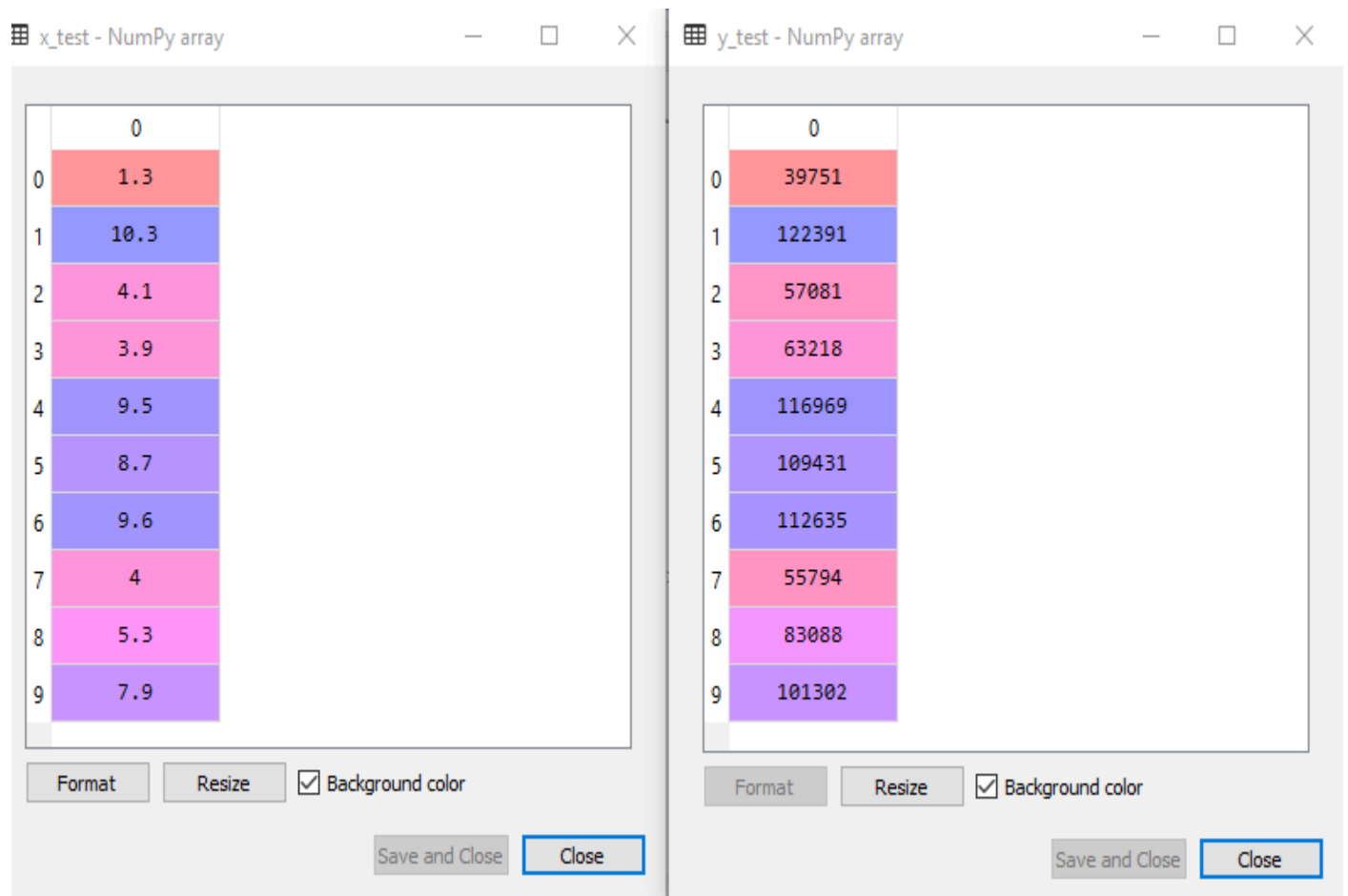
In the above output image, we can see the X (independent) variable and Y (dependent) variable has been extracted from the given dataset.

- 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:

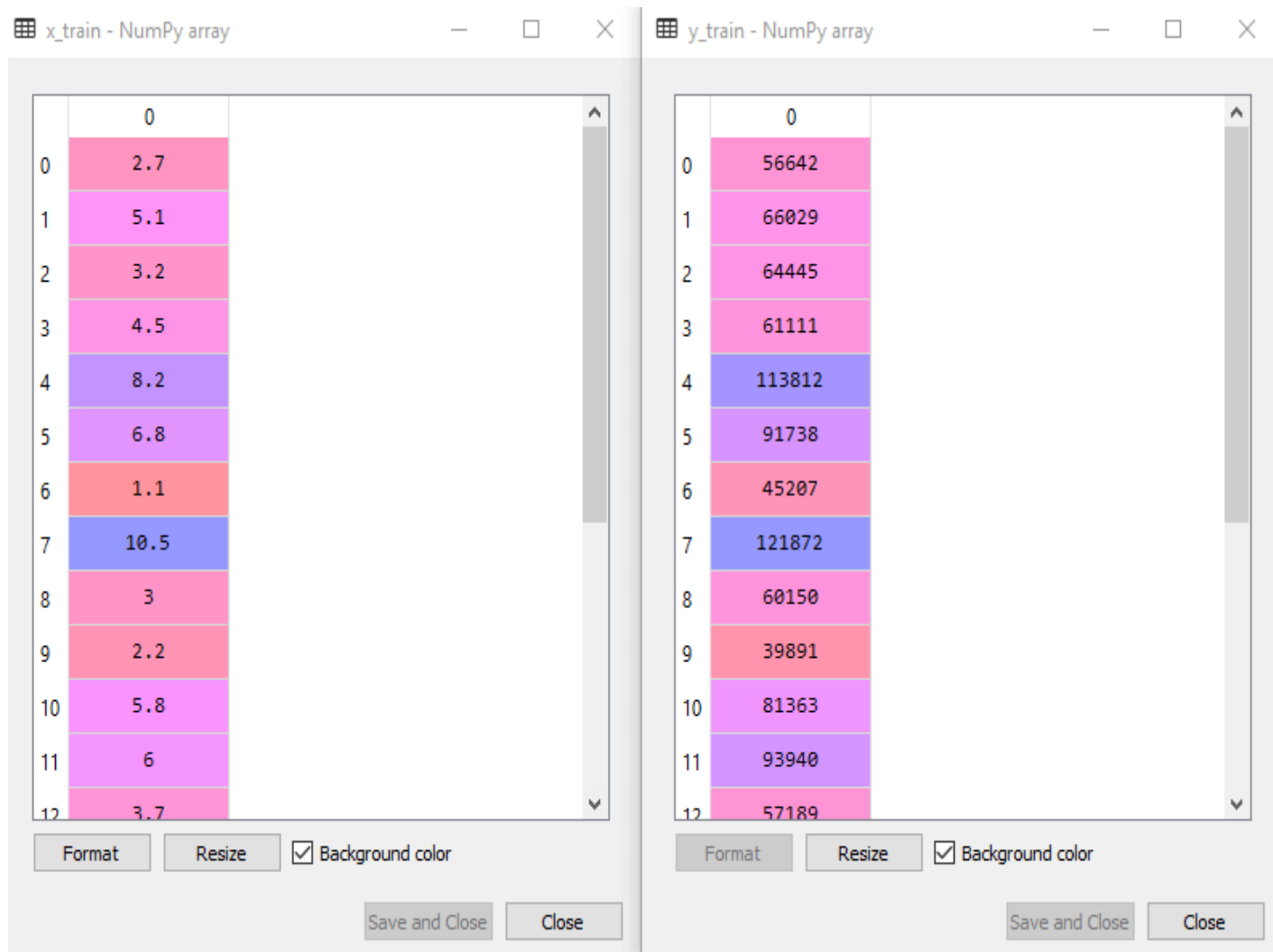
1. # Splitting the dataset into training and test set.
2. from sklearn.model_selection **import** train_test_split
3. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)

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

Test-dataset:



Training 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:

1. #Fitting the Simple Linear Regression model to the training dataset
2. from sklearn.linear_model **import** LinearRegression
3. regressor= LinearRegression()

4. 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:

ADVERTISEMENT

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

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.

1. #Prediction of Test and Training set result
2. y_pred= regressor.predict(x_test)
3. 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.

Output:

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:

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.

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:

ADVERTISEMENT

1. `mtp.scatter(x_train, y_train, color="green")`
2. `mtp.plot(x_train, x_pred, color="red")`
3. `mtp.title("Salary vs Experience (Training Dataset)")`
4. `mtp.xlabel("Years of Experience")`
5. `mtp.ylabel("Salary(In Rupees)")`
6. `mtp.show()`

Output:

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

Multiple Linear Regression

In the previous topic, we have learned about Simple Linear Regression, where a single Independent/Predictor(X) variable is used to model the response variable (Y). But there may be various cases in which the response variable is affected by more than one predictor variable; for such cases, the Multiple Linear Regression algorithm is used.

ADVERTISEMENT

Moreover, Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. We can define it as:

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.

Example:

Prediction of CO₂ emission based on engine size and number of cylinders in a car.

Some 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.

MLR equation:

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

1. $Y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$ (a)

Where,

ADVERTISEMENT

Y= Output/Response variable

$b_0, b_1, b_2, b_3, b_n, \dots$ = Coefficients of the model.

$x_1, x_2, x_3, x_4, \dots$ = Various Independent/feature variable

Assumptions for Multiple Linear Regression:

- A **linear relationship** should exist between the Target and predictor variables.
- The regression residuals must be **normally distributed**.
- MLR assumes little or **no multicollinearity** (correlation between the independent variable) in data.

Implementation of Multiple Linear Regression model using Python:

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**

Step-1: Data Pre-processing Step:

The very first step is [data pre-processing](#), which we have already discussed in this tutorial. 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:

1. `# importing libraries`
2. `import numpy as nm`
3. `import matplotlib.pyplot as mtp`
4. `import pandas as pd`

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

1. `#importing datasets`
2. `data_set= pd.read_csv('50_BlueList.csv')`

Output: We will get the dataset as:

ADVERTISEMENT

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

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

○ **Extracting dependent and independent Variables:**

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

Output:

Out[5]:

```
array([[165349.2, 136897.8, 471784.1, 'New York'],
```

```
[162597.7, 151377.59, 443898.53, 'California'],
[153441.51, 101145.55, 407934.54, 'Florida'],
[144372.41, 118671.85, 383199.62, 'New York'],
[142107.34, 91391.77, 366168.42, 'Florida'],
[131876.9, 99814.71, 362861.36, 'New York'],
[134615.46, 147198.87, 127716.82, 'California'],
[130298.13, 145530.06, 323876.68, 'Florida'],
[120542.52, 148718.95, 311613.29, 'New York'],
[123334.88, 108679.17, 304981.62, 'California'],
[101913.08, 110594.11, 229160.95, 'Florida'],
[100671.96, 91790.61, 249744.55, 'California'],
[93863.75, 127320.38, 249839.44, 'Florida'],
[91992.39, 135495.07, 252664.93, 'California'],
[119943.24, 156547.42, 256512.92, 'Florida'],
[114523.61, 122616.84, 261776.23, 'New York'],
[78013.11, 121597.55, 264346.06, 'California'],
[94657.16, 145077.58, 282574.31, 'New York'],
[91749.16, 114175.79, 294919.57, 'Florida'],
[86419.7, 153514.11, 0.0, 'New York'],
[76253.86, 113867.3, 298664.47, 'California'],
[78389.47, 153773.43, 299737.29, 'New York'],
[73994.56, 122782.75, 303319.26, 'Florida'],
[67532.53, 105751.03, 304768.73, 'Florida'],
[77044.01, 99281.34, 140574.81, 'New York'],
[64664.71, 139553.16, 137962.62, 'California'],
[75328.87, 144135.98, 134050.07, 'Florida'],
[72107.6, 127864.55, 353183.81, 'New York'],
[66051.52, 182645.56, 118148.2, 'Florida'],
[65605.48, 153032.06, 107138.38, 'New York'],
[61994.48, 115641.28, 91131.24, 'Florida'],
[61136.38, 152701.92, 88218.23, 'New York'],
[63408.86, 129219.61, 46085.25, 'California'],
[55493.95, 103057.49, 214634.81, 'Florida'],
[46426.07, 157693.92, 210797.67, 'California'],
[46014.02, 85047.44, 205517.64, 'New York'],
[28663.76, 127056.21, 201126.82, 'Florida'],
[44069.95, 51283.14, 197029.42, 'California'],
[20229.59, 65947.93, 185265.1, 'New York'],
[38558.51, 82982.09, 174999.3, 'California'],
[28754.33, 118546.05, 172795.67, 'California'],
[27892.92, 84710.77, 164470.71, 'Florida'],
[23640.93, 96189.63, 148001.11, 'California'],
[15505.73, 127382.3, 35534.17, 'New York'],
[22177.74, 154806.14, 28334.72, 'California'],
[1000.23, 124153.04, 1903.93, 'New York'],
[1315.46, 115816.21, 297114.46, 'Florida'],
[0.0, 135426.92, 0.0, 'California'],
[542.05, 51743.15, 0.0, 'New York'],
[0.0, 116983.8, 45173.06, 'California']], dtype=object)
```

As we can see in the above output, the last column contains categorical variables which are not suitable to apply directly for fitting the model. So we need to encode this variable.

ADVERTISEMENT

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:

1. #Categorical data
2. from sklearn.preprocessing **import** LabelEncoder, OneHotEncoder
3. labelencoder_x= LabelEncoder()
4. x[:, 3]= labelencoder_x.fit_transform(x[:,3])
5. onehotencoder= OneHotEncoder(categorical_features= [3])
6. x= onehotencoder.fit_transform(x).toarray()

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

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**.

Note: We should not use all the dummy variables at the same time, so it must be 1 less than the total number of dummy variables, else it will create a dummy variable trap.

- 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.

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

As we can see in the above output image, the first column has been removed.

- Now we will split the dataset into training and test set. The code for this is given below:

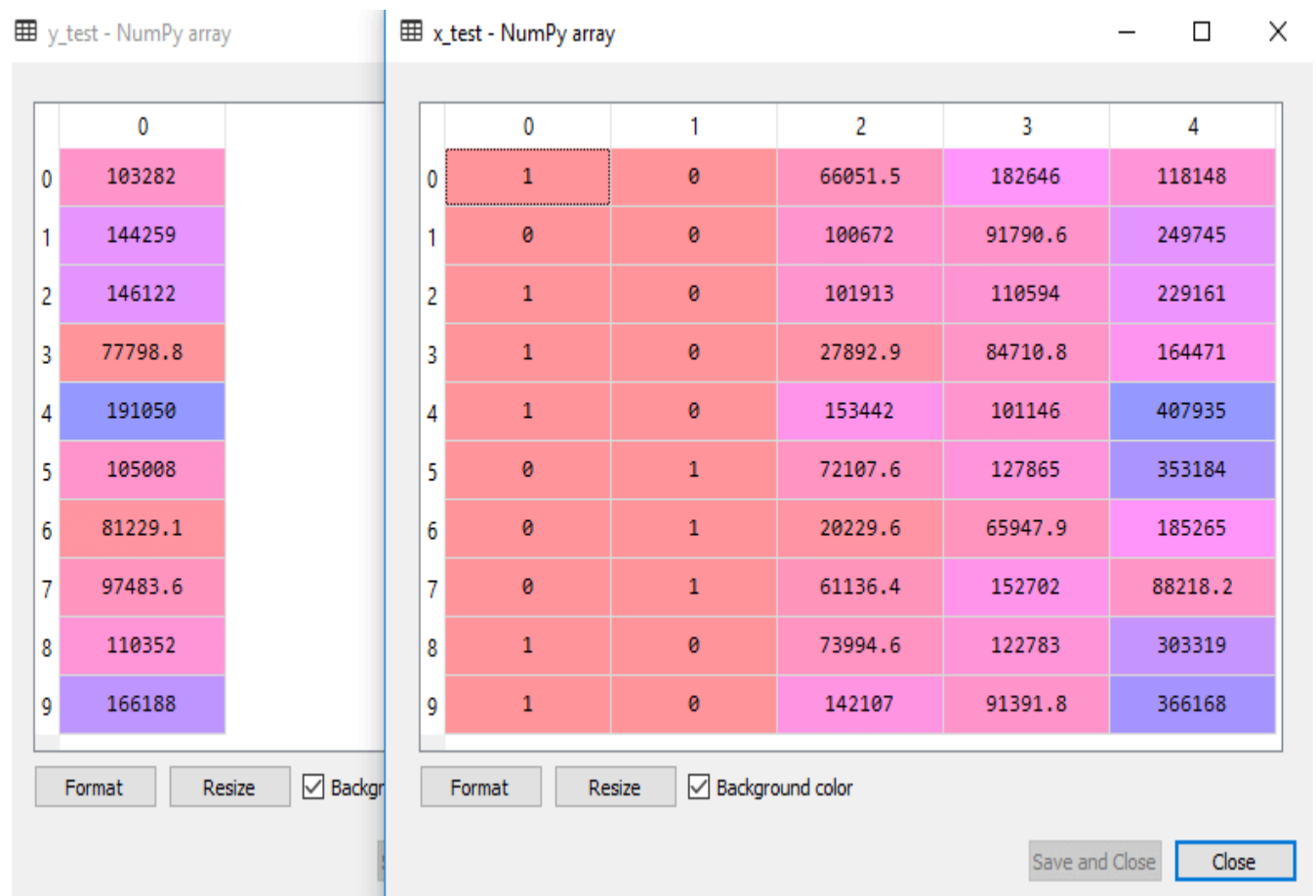
- # Splitting the dataset into training and test set.
- from sklearn.model_selection **import** train_test_split
- x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)

The above code will split our dataset into a training set and test set.

ADVERTISEMENT

Output: The above code will split the dataset into training set and test set. You can check the output by clicking on the variable explorer option given in Spyder IDE. The test set and training set will look like the below image:

Test set:



The image shows two side-by-side windows in the Spyder IDE variable explorer. The left window is titled 'y_test - NumPy array' and displays a 1D array of 10 values. The right window is titled 'x_test - NumPy array' and displays a 2D array of 10 rows and 5 columns. Both windows have a 'Background color' checkbox checked. The right window also has 'Save and Close' and 'Close' buttons at the bottom right.

	0
0	103282
1	144259
2	146122
3	77798.8
4	191050
5	105008
6	81229.1
7	97483.6
8	110352
9	166188

	0	1	2	3	4
0	1	0	66051.5	182646	118148
1	0	0	100672	91790.6	249745
2	1	0	101913	110594	229161
3	1	0	27892.9	84710.8	164471
4	1	0	153442	101146	407935
5	0	1	72107.6	127865	353184
6	0	1	20229.6	65947.9	185265
7	0	1	61136.4	152702	88218.2
8	1	0	73994.6	122783	303319
9	1	0	142107	91391.8	366168

Training set:

	0	1	2	3	4
0	1	0	55493.9	103057	214635
1	0	1	46014	85047.4	205518
2	1	0	75328.9	144136	134050
3	0	0	46426.1	157694	210798
4	1	0	91749.2	114176	294920
5	1	0	130298	145530	323877
6	1	0	119943	156547	256513
7	0	1	1000.23	124153	1903.93
8	0	1	542.05	51743.2	0
9	0	1	65605.5	153032	107138
10	0	1	114524	122617	261776
11	1	0	61994.5	115641	91131.2

	0
0	96778.9
1	96479.5
2	105734
3	96712.8
4	124267
5	155753
6	132603
7	64926.1
8	35673.4
9	101005
10	129917
11	99937.6

Note: In MLR, we will not do feature scaling as it is taken care by the library, so we don't need to do it manually.

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:

1. #Fitting the MLR model to the training set:
2. from sklearn.linear_model **import** LinearRegression
3. regressor= LinearRegression()
4. regressor.fit(x_train, y_train)

Output:


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

Now, we have successfully trained our model using the training dataset. In the next step, we will test the performance of the model using the test dataset.

Step: 3- Prediction of Test set results:

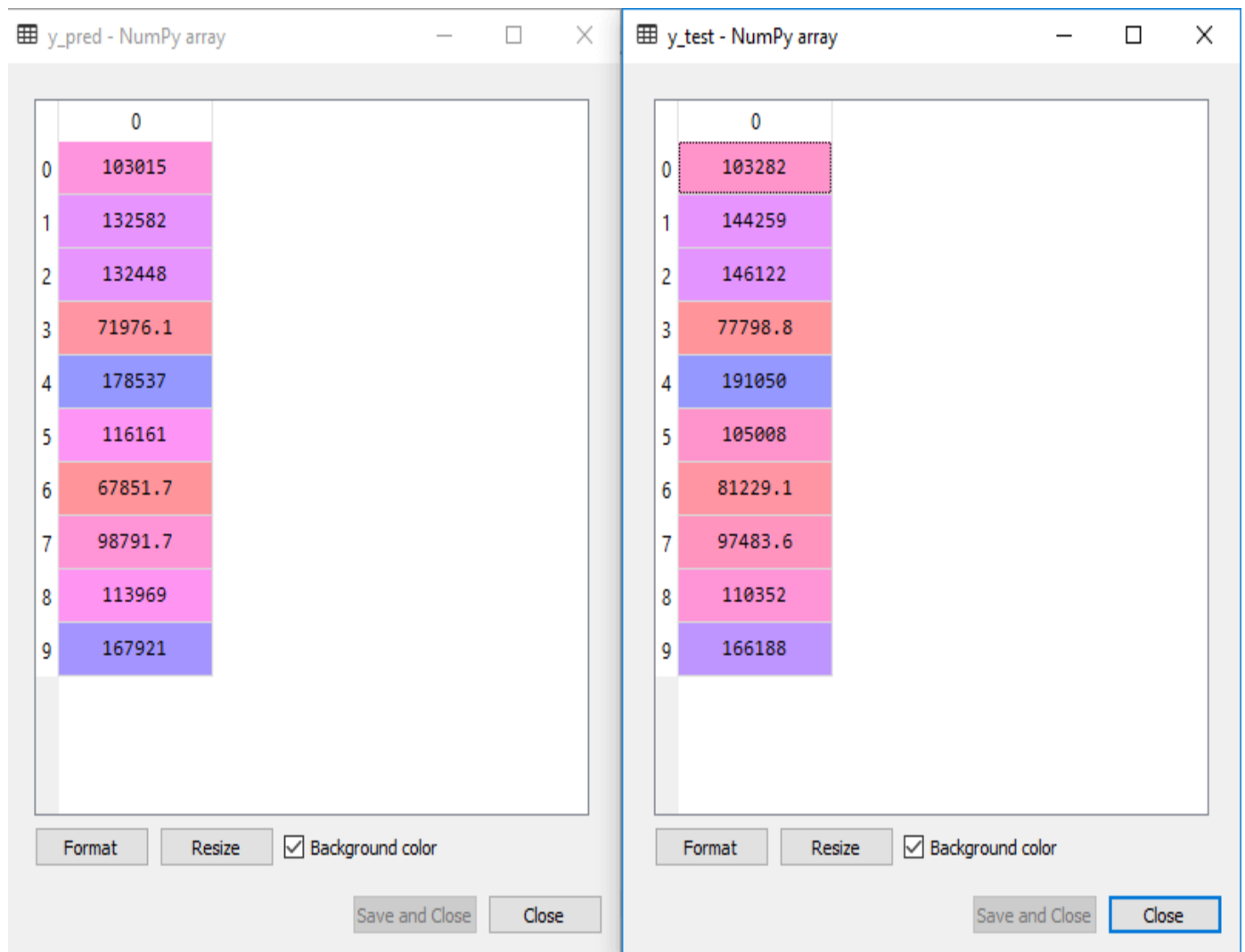
The last step for our model is checking the performance of the model. We will do it by predicting the test set result. For prediction, we will create a **y_pred** vector. Below is the code for it:

1. #Predicting the Test set result;
2. 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.

Output:

ADVERTISEMENT



In the above output, we have predicted result set and test set. We can check model performance by comparing these two value index by index. For example, the first index has a predicted value of **103015\$** profit and test/real value of **103282\$** profit. The difference is only of **267\$**, which is a good prediction, so, finally, our model is completed here.

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

1. `print('Train Score: ', regressor.score(x_train, y_train))`
2. `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.

Note: In the next topic, we will see how we can improve the performance of the model using the **Backward Elimination** process.

Applications of Multiple Linear Regression:

There are mainly two applications of Multiple Linear Regression:

- Effectiveness of Independent variable on prediction:
- Predicting the impact of changes:



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.

1. `#visualizing the Test set results`
2. `mtp.scatter(x_test, y_test, color="blue")`
3. `mtp.plot(x_train, x_pred, color="red")`
4. `mtp.title("Salary vs Experience (Test Dataset)")`
5. `mtp.xlabel("Years of Experience")`
6. `mtp.ylabel("Salary(In Rupees)")`
7. `mtp.show()`

Output:

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



In the above 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.