Supervised Learning:
In supervised learning, the model learns from labeled data, meaning each input data point is paired with the correct output. The goal is to learn a mapping from input variables to output variables.
The model tries to approximate the relationship between input and output based on the labeled training data.
Common examples include regression and classification tasks.
Examples of supervised learning algorithms include:
Linear Regression
Logistic Regression
Support Vector Machines (SVM)
Decision Trees
Random Forests
Neural Networks (when trained with labeled data)

Unsupervised Learning:
In unsupervised learning, the model learns from unlabeled data, where there are no predefined labels for the input data.
The goal is to discover hidden patterns or structures in the data.
Unsupervised learning is often used for clustering, dimensionality reduction, and density estimation.
Examples of unsupervised learning algorithms include:
K-Means Clustering
Hierarchical Clustering
Principal Component Analysis (PCA)
t-Distributed Stochastic Neighbor Embedding (t-SNE)
Autoencoders
Generative Adversarial Networks (GANs)
These algorithms form the backbone of many machine learning applications, each with its strengths and weaknesses depending on the nature of the data and the problem at hand.

Reinforcement learning (RL) is another key paradigm in machine learning, distinct from both supervised and unsupervised learning. Here's an overview:

Reinforcement Learning:
In reinforcement learning, an agent learns to make decisions by interacting with an environment. The agent learns from feedback in the form of rewards or punishments received from its actions. The goal of the agent is to learn a policy, a mapping from states to actions, that maximizes cumulative rewards over time.
RL is often formulated as a Markov Decision Process (MDP), where the agent observes the current state, selects an action, receives a reward, and transitions to a new state according to the environment's dynamics.
RL algorithms aim to optimize the agent's policy through exploration (trying out different actions) and exploitation (taking the best-known actions based on the current policy).
Reinforcement learning is commonly used in scenarios such as game playing, robotics, recommendation systems, and autonomous vehicle control.
Some key concepts and algorithms in reinforcement learning include:

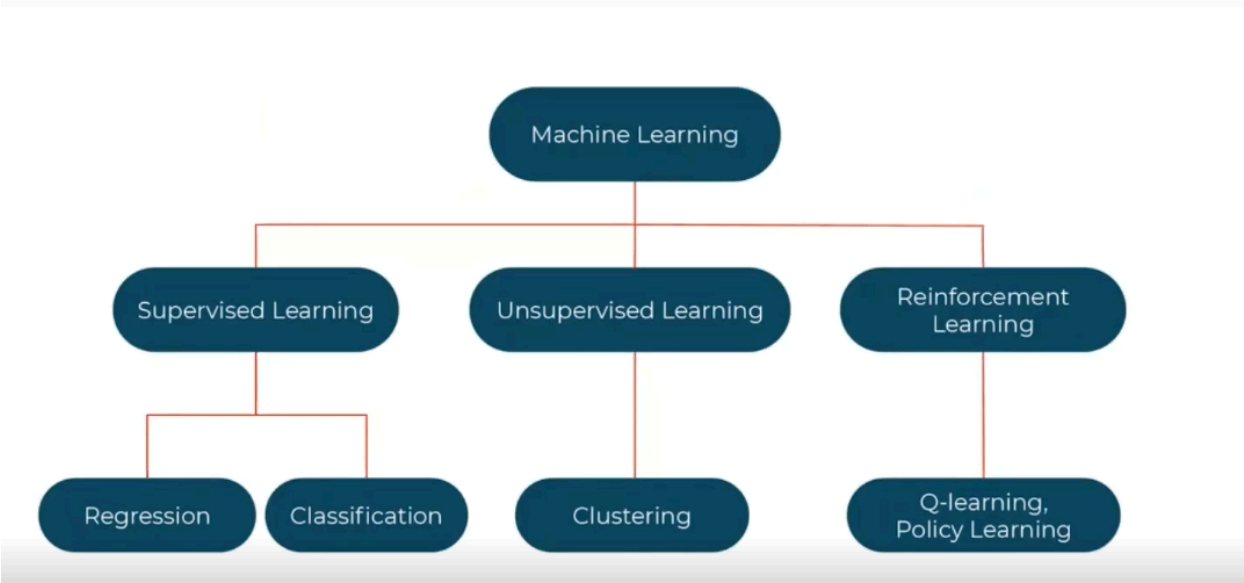
Policy: The strategy the agent uses to determine its actions in different states.
Value Function: Measures the expected cumulative reward the agent can achieve from a given state or state-action pair.
Q-Learning: A model-free RL algorithm that learns the value of taking a particular action in a particular state.
Deep Q-Networks (DQN): An extension of Q-learning that uses deep neural networks to approximate the Q-value function.
Policy Gradient Methods: Directly optimize the agent's policy by adjusting its parameters in the direction that increases expected rewards.
Actor-Critic Methods: Combines value-based and policy-based methods by training both a policy (actor) and a value function (critic) simultaneously.

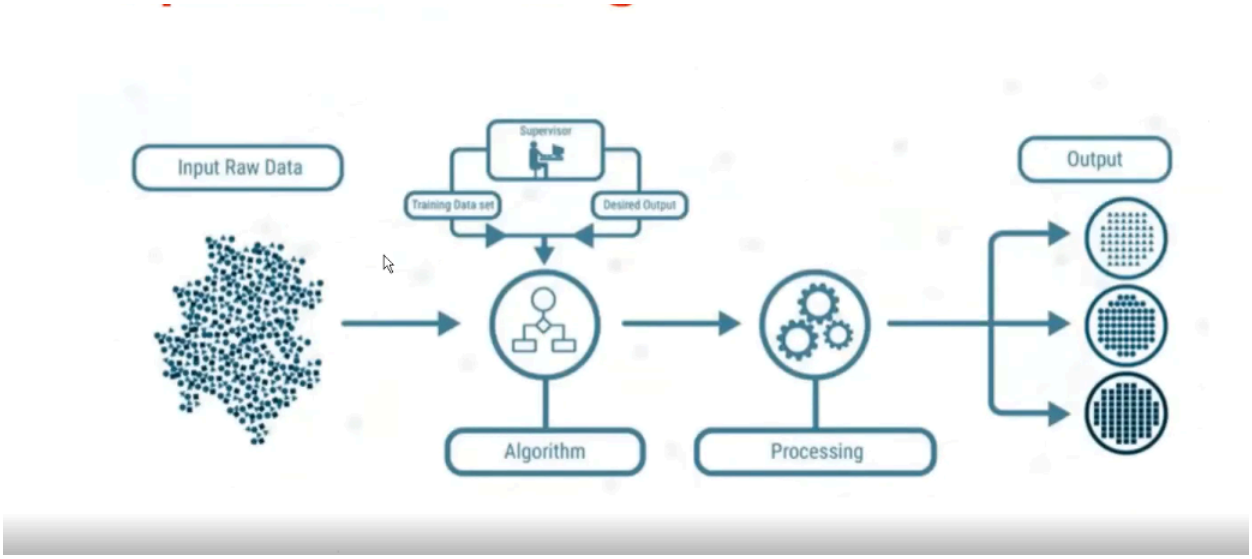
Machine Learning - Learning any process by which a system improves performance from experience.
Machine Learning is concerned with computer program that automatically improve their performance through experience.

Revisiting Bird Classification

Features/Input Variables | Dependent Variable

Has Wings	Can Fly	Has Backbone	Has Chitin	Bird or Not
Yes	Yes	Yes	No	Bird
Yes	Yes	Yes	No	Bird
Yes	Yes	No	Yes	Not Bird





Processing- training
Example o/p from above dataset- (a)bird (b) not bird

X > Input data
Y > Output data

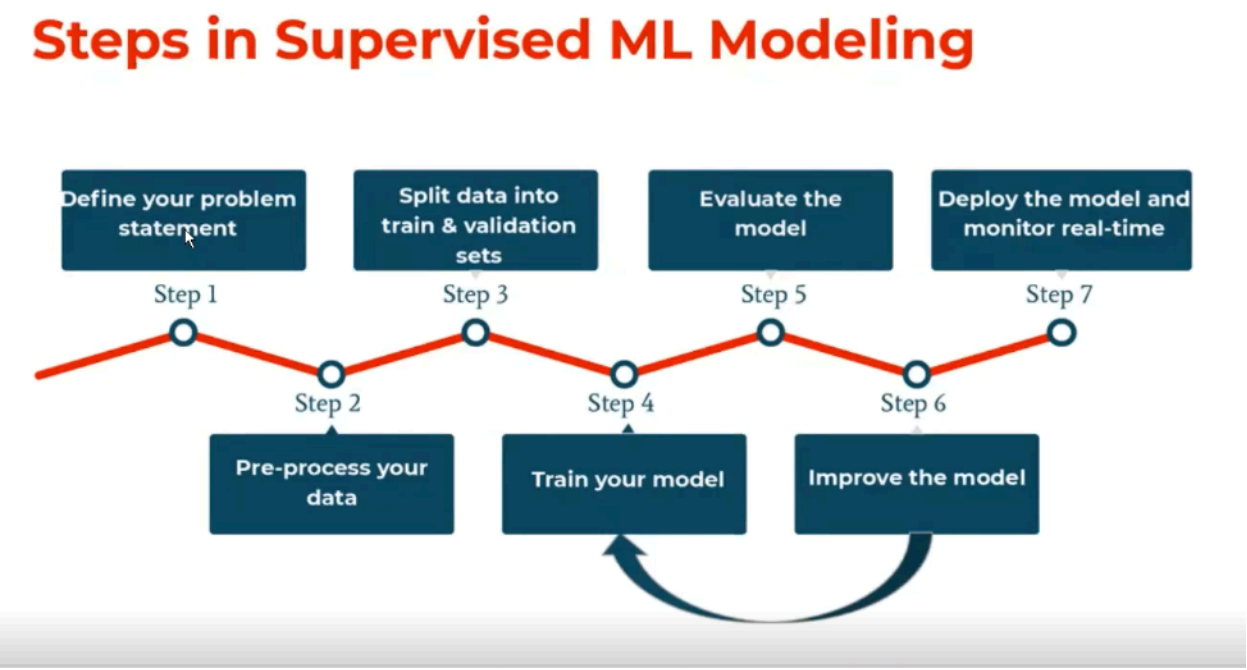
Supervised Learning

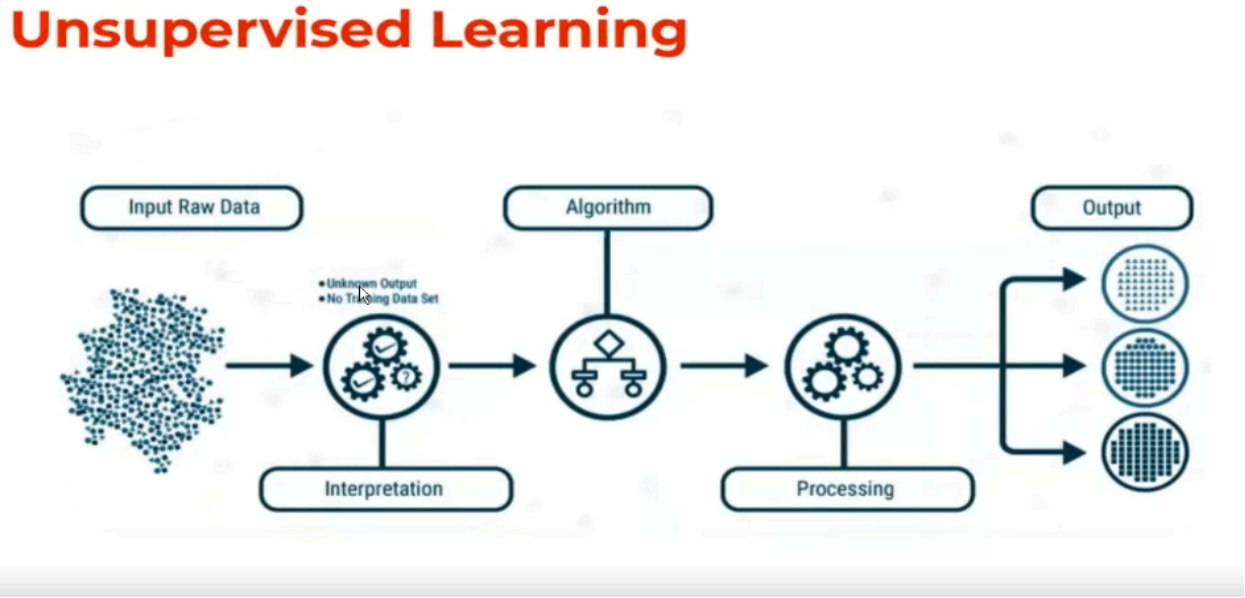
- Regression - Dependent variables can take continuous value.

Example - price of a house($1000, $500.50), stock market.

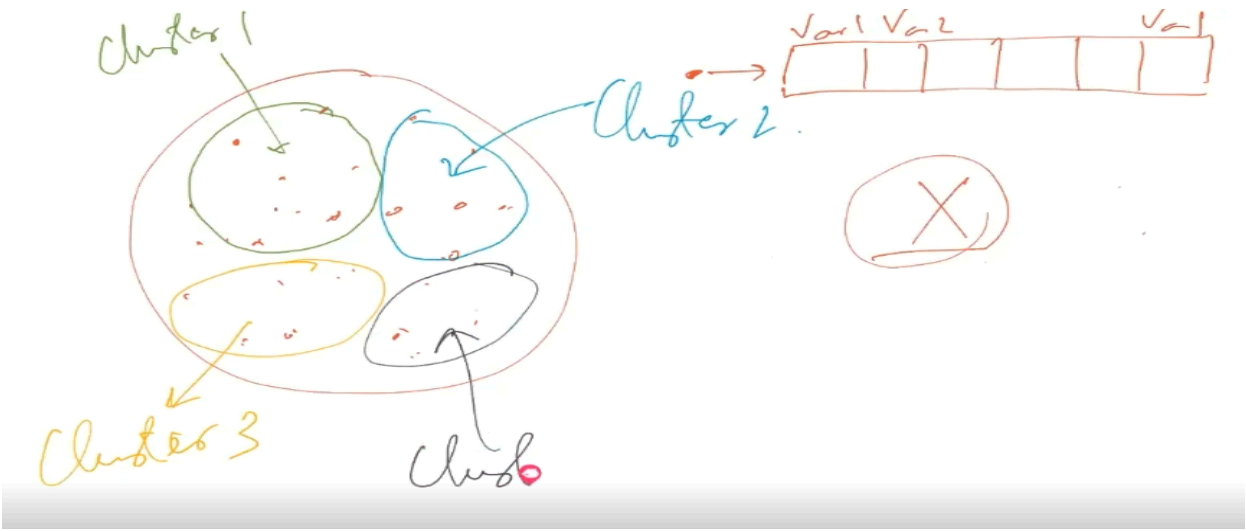
- Classification - Dependent variables can take only fixed set of categories/levels/factors/lables.

Example- Credit card fraud detection.





Here dependent variable is unknown.
o/p will be some different group of observations.
Grouping them based on their behaviors.

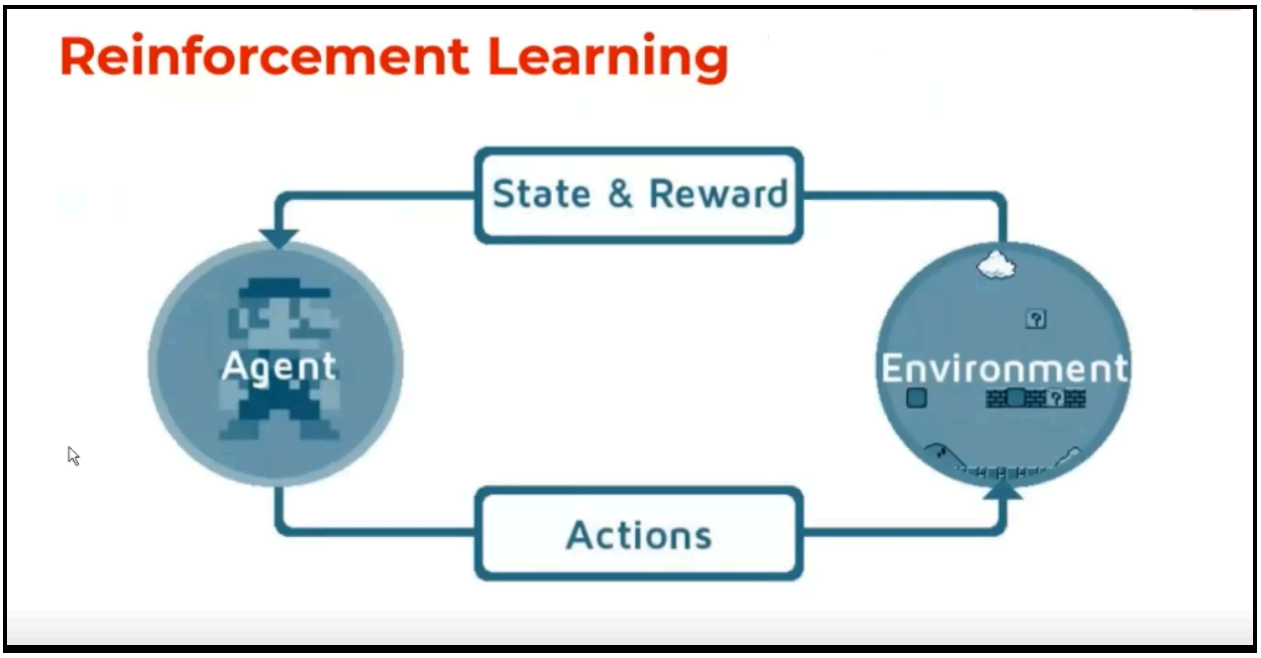


Datasets in Unsupervised Learning

Feature 1	Feature 2	.....	Feature K

No dependent variable available

Example - A retail company might use unsupervised clustering to segment their customers into groups based on their purchase history. The company could use a clustering algorithm to identify groups of customers who have similar shopping habits, such as the frequency of their purchases, the types of products they buy, and the amount of money they spend.Once the company has identified these customer segments, they can use this information to target their marketing campaigns more effectively. For example, the company could send targeted emails to each customer segment with offers that are tailored to their specific interests



Linear Regression
Power of two things (a) relation i/p - o/p
(b) future prediction

ML - finding patterns in data
Function maps i/p - o/p
f(x) = y
Obj: find that function

ML - model to capture deterministic part
Data = deterministic + Random
deterministic = mathematical function

deterministic - mathematical relationships that govern the problem that the model is trying to solve. For example, in a model that is trying to predict the weather, the deterministic part of the model might represent the relationship between the current temperature, humidity, and wind speed and the future weather conditions.

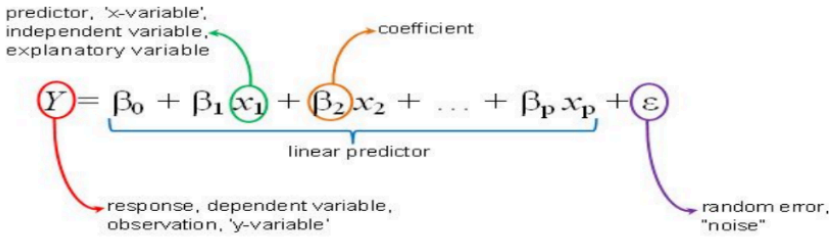
Linear Regression - y = mx + c

2D - Line
3D - Plane
More than 3D - Hyperplane

Linear regression is one of the most basic types of regression in supervised machine learning. The linear regression model consists of a predictor variable and a dependent variable related linearly to each other. We try to find the relationship between independent variable (input) and a corresponding dependent variable (output).

Hypothesis of Linear Regression

The linear regression model can be represented by the following equation:



Y is the actual value
β_0 is the bias term.
$\beta_1,\ldots,\beta p$ are the model parameters
$x_1, x_2,\ldots,xp$ are the feature values.

It is a mathematical equation that is used to predict the value of the dependent variable (Y) based on the values of the independent variables (X1, X2, ..., Xp)

$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta pXp + \varepsilon$

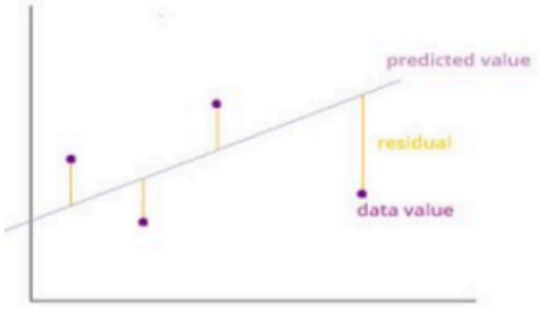
The term $\beta 0$ is the intercept of the regression line, and the terms $\beta 1$, $\beta 2$, ..., βp are the slope coefficients of the regression line. The term ε is the error term, and it represents the random variation that is not explained by the regression line.
The hypothesis of linear regression is used to fit a linear regression line to a dataset of observations. The line is fitted by minimizing the sum of squared errors (SSE). The SSE is a measure of the distance between the predicted values and the actual values. The lower the SSE, the better the fit of the linear regression line.
Here is an example of how the hypothesis of linear regression can be used. Let's say that we have a dataset of houses, and we want to predict the price of a house based on the number of bedrooms. The hypothesis of linear regression can be used to fit a linear regression line to the dataset. The line can then be used to predict the price of a house for a given number of bedrooms.

How do we determine the best fit line?

The best fit line is considered to be the line for which the error between the predicted values and the observed values is minimum.
It is also called the regression line and the errors are also known as residuals.
It can be visualized by the vertical lines from the observed data value to the regression line.

$$\min(SSE) = \sum_{i=1}^{n} (\text{actual output} - \text{predicted output})^2$$

SSE : Sum square error



The lower the SSE, the better the fit of the linear regression model.

```
SSE = (actual output - predicted output)^2
```

The term `actual output` refers to the actual value of the output variable, and the term `predicted output` refers to the value of the output variable that is predicted by the linear regression model. The ^2 operator means that the difference between the actual output and the predicted output is squared.
The SSE equation is minimized during the training of the linear regression model. This is done by adjusting the coefficients of the linear regression model until the SSE is minimized.

Here is an example of how the SSE equation can be used. Let's say that we have a linear regression model that predicts the price of a house based on the number of bedrooms. The model has been trained on a dataset of houses, and the SSE for the model is 100. This means that the average difference between the predicted prices and the actual prices is 100.
If we were to improve the model, we could try to reduce the SSE. This could be done by adding more features to the model, such as the size of the house or the location of the house. By adding more features, the model would be able to make more accurate predictions, and the SSE would be reduced.

- Anything we can estimated is estimated value and we will put a hat. Ex- y^ - predicted value
- True value ex- y
- Our mathematical model should very close to reality.

Without error term

Linear regression equation without error term gives predicted output.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_p X_p$$

By varying different parameter values (β_p) , predicted value can be determined and value which gives minimum SSE will produce the best fit line.

SSE : $\sum_{i=1}^{n} (y - \hat{y})^2$

Note: Predicted value can have positive or negative error. If we don't square the error, then the positive and negative points will cancel each other out. Absolute function(|Actual- Pred|) can be one option but it is not differentiable at Actual = Pred

When I get the minimum SSE value, my mathematical model will be very close to reality.

Derivative = 0

Setting these partial derivatives equal to zero yields the following equations for the minimizing values β_0 and β_1

$$\sum_{i=1}^{n} y_i = n\beta_o + \beta_1 \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} x_i y_i = \beta_o \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2$$

These equations are known as the **normal equations.**

contd..

If we let

$$\bar{y} = \sum_i \frac{y_i}{n}$$

$$\bar{x} = \sum_i \frac{x_i}{n}$$

then we can write the first normal equation as $\beta_0 = \bar{y} - \beta_1 \bar{x}$

Substituting this value of β_0 into the second normal equation yields

$$\sum_i x_i y_i = \left(\bar{y} - \beta_1 \bar{x}\right) \cdot n\bar{x} + \beta_1 \sum_i x_i^2$$

Or

$$\beta_1 \left(\sum_i x_i^2 - n\bar{x}^2 \right) = \sum_i x_i y_i - n \cdot \bar{x}\,\bar{y}$$

$$\beta_1 = \frac{\left(\sum_i x_i y_i - n \cdot \bar{x} \cdot \bar{y}\right)}{\left(\sum_i x_i^2 - n \cdot (\bar{x})^2\right)}$$

$$\beta_1 = \frac{n\sum_i x_i y_i - \sum_i x \sum_i y}{n\sum_i x_i^2 - \left(\sum_i x_i\right)^2}$$

Normal equation/ Closed form equation ensure us that we will get global minima solution.
Gradient Descent / search algorithm is a Iterative solution. It will give you a approximate value. If the value is 10. Gradient descent value will be 10.1/9.9.

In linear regression, the closed-form solution is the equation that can be used to calculate the coefficients of the linear model. The coefficients are the values that are multiplied by the independent variables to get the predicted value of the dependent variable
The closed-form solution for linear regression is

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

where:

- $\hat{\beta}$ is the vector of coefficients
- X is the matrix of independent variables
- y is the vector of dependent variables
- ^{T} is the transpose operator

The closed-form solution can be solved algebraically. This means that there is no need to use numerical methods to find the solution. The closed-form solution is often preferred over numerical solutions because it is more precise and faster to compute.
However, the closed-form solution for linear regression can only be used if the number of independent variables is less than or equal to the number of observations. If there are more independent variables than observations, then the closed-form solution does not exist. In this case, numerical methods must be used to find an approximate solution for the coefficients

Example: Sam found how many **hours of sunshine** vs how many **ice creams** were sold at the shop from Monday to Friday:

"x" Hours of Sunshine	"y" Ice Creams Sold
2	4
3	5
5	7
7	10
9	15

Let us find the best **m** (slope) and **b** (y-intercept) that suits that data

$$y = mx + b$$

Step 1: For each (x,y) calculate x^2 and xy:

x	y	x^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135

Step 2: Sum x, y, x^2 and xy (gives us Σx, Σy, Σx^2 and Σxy):

x	y	x^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135
Σx: 26	**Σy: 41**	**Σx^2: 168**	**Σxy: 263**

Also **N** (number of data values) = **5**

Step 3: Calculate Slope **m**:

$$\mathbf{m} = \frac{N\,\Sigma(xy) - \Sigma x\,\Sigma y}{N\,\Sigma(x^2) - (\Sigma x)^2}$$

$$= \frac{5 \times 263 - 26 \times 41}{5 \times 168 - 26^2}$$

$$= \frac{1315 - 1066}{840 - 676}$$

$$= \frac{249}{164} = 1.5183...$$

Step 4: Calculate Intercept **b**:

$$\mathbf{b} = \frac{\Sigma y - m\,\Sigma x}{N}$$

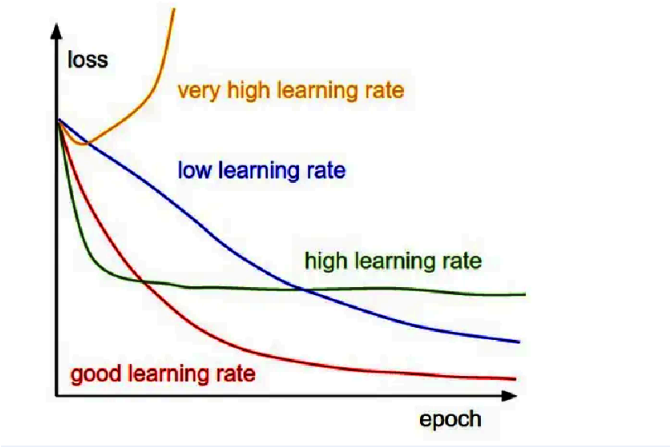
$$= \frac{41 - 1.5183 \times 26}{5}$$

$$= 0.3049...$$

Step 5: Assemble the equation of a line:

$$y = mx + b$$

$$y = 1.518x + 0.305$$



- Gradient Descent is an optimization algorithm that helps machine learning models to find out paths to a minimum value using repeated steps.

Gradient descent is used to minimize a function so that it gives the lowest output of that function. This function is called the Loss Function.

The gradient descent equation is an iterative algorithm that is used to find the minimum of a function. The function is typically represented as a graph, and the gradient descent algorithm moves down the graph until it reaches the minimum point

The gradient descent equation is:

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

where:

- x_n is the current point
- x_{n+1} is the next point
- α is the learning rate
- $\nabla f(x_n)$ is the gradient of the function at the point x_n

$\nabla f(x_n)$ is pronounced as "del f of x n

The gradient of the function is a vector that points in the direction of the steepest ascent of the function. The learning rate controls how much the algorithm moves in the direction of the gradient.

The gradient descent algorithm starts at a random point and then moves in the direction of the gradient until it reaches the minimum point. The algorithm is guaranteed to converge to the minimum point if the learning rate is small enough.
Gradient descent is a simple and efficient algorithm that can be used to find the minimum of a function. However, it can be slow to converge if the function is complex. In these cases, other optimization algorithms, such as Newton's method, may be more efficient.

Convergence- refers to the process of the gradient descent algorithm approaching a local minimum of the loss function. Convergence typically refers to the algorithm reaching a point where further iterations do not significantly change the solution.

Gradient Ascent: If we want to maximize a function, we move in the direction of the gradient.
Gradient Descent: If we want to minimize a function, we move in the direction opposite to the gradient.

The term "gradient" refers to the vector of partial derivatives of a scalar-valued function (often referred to as the "loss function" or "cost function") with respect to each of its parameters.
Mathematically, if we have a function $f(w)$ that depends on a vector of parameters w = $(w1,w2,\ldots,wn)$ the gradient of f with respect to w denoted by $\nabla f(w)$ or $\partial f/\partial w$, is a vector where each component is the partial derivative of f with respect to the corresponding parameter wi.

In the context of gradient descent optimization, the gradient of the loss function represents the direction of the steepest ascent (the direction in which the function increases the most rapidly) at a particular point in the parameter space.

Therefore, in gradient descent, the algorithm iteratively updates the parameters in the direction opposite to the gradient of the loss function. This means that if we want to minimize the loss function, we move in the direction opposite to the gradient, taking steps proportional to the negative of the gradient. This process continues iteratively until convergence is reached, meaning that the gradient becomes close to zero or the updates to the parameters become negligible.

In summary, the gradient in gradient descent is a crucial concept that guides the optimization process by indicating the direction of the greatest increase of the loss function, allowing the algorithm to search for the optimal parameters that minimize the loss.

- Check Bias-Variance Tradeoff code in github and notes

- L1 and L2 regularization are techniques used in machine learning to address the issue of overfitting.

L1 - using absolute value.
L2 - because we are using square.

L1 Regularization (Lasso) with MSE-
L1 regularization adds the sum of the absolute values of the coefficients to the MSE loss function. This encourages sparsity(sparsity refers to a situation where many of the model's parameters (coefficients) are zero or close to zero.), potentially setting some coefficients to exactly zero.

Loss Function with L1 Regularization:

$$\text{Loss}_{L1} = \text{MSE} + \lambda \sum_{j=1}^{p} |\beta_j|$$

Where:

- **MSE:** Mean Squared Error
- λ: Regularization parameter controlling the strength of the penalty
- β_j: Model coefficients

p: The total number of features (or predictors) in the model. It indicates how many coefficients
j: An index that ranges from 1 to p, representing each feature's coefficient in the sum.

L2 Regularization (Ridge) with MSE-
L2 regularization adds the sum of the squared values of the coefficients to the MSE loss function. This encourages smaller coefficients, reducing the risk of overfitting.

Loss Function with L2 Regularization:

$$\text{Loss}_{L2} = \text{MSE} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Where:

- **MSE:** Mean Squared Error
- λ: Regularization parameter controlling the strength of the penalty
- β_j: Model coefficients

p: The total number of features (or predictors) in the model. It indicates how many coefficients
j: An index that ranges from 1 to p, representing each feature's coefficient in the sum.