**Machine learning**

Machine learning is a field of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computers to perform tasks without being explicitly programmed for those tasks. The primary goal of machine learning is to create systems that can learn from data and make predictions or decisions based on that learning.

1. **Supervised Learning:** In supervised learning, the algorithm is trained on a labelled dataset, which means that the input data is paired with corresponding output labels. The algorithm learns the mapping from inputs to outputs, and once trained, it can make predictions on new, unseen data.
2. **Unsupervised Learning:** Unsupervised learning deals with unlabelled data, where the algorithm tries to find patterns, relationships, or structures in the data without explicit guidance on the output. Clustering and dimensionality reduction are common tasks in unsupervised learning.

**Supervised Learning:**

**Supervised learning algorithms** are trained on labeled data, meaning that the input data is accompanied by a corresponding output value that the algorithm is trying to predict. For example, a supervised learning algorithm could be trained to predict the price of a house based on its size, location, and other features. Common supervised learning algorithms include linear regression, decision trees, and support vector machines.

**Unsupervised learning algorithms,** on the other hand, are trained on unlabeled data, meaning that there is no corresponding output value that the algorithm is trying to predict. Instead, these algorithms are designed to find patterns or structure in the data on their own. For example, an unsupervised learning algorithm could be used to group similar products together based on their features. Common unsupervised learning algorithms include clustering, principal component analysis, and autoencoders.

**Reinforcement learning algorithms** are a bit different in that they are designed to learn by interacting with an environment and receiving feedback in the form of rewards or punishments. These algorithms are often used in robotics, gaming, and other applications where an agent needs to learn how to make decisions based on its environment. Common reinforcement learning algorithms include Q-learning, policy gradient methods, and deep reinforcement learning.

Overall, machine learning algorithms are an essential tool for data analysis and prediction, and their applications are only continuing to expand as technology advances.

There are many machine learning algorithms, and new ones are constantly being developed. Here is a list of some of the most common and widely used machine learning algorithms:

1. Linear regression
2. Multivariate repression
3. Polynomial regression
4. Logistic regression
5. Decision trees
6. Random forests
7. Support vector machines
8. Naive Bayes
9. K-Nearest Neighbors
10. K-means clustering
11. Hierarchical clustering
12. Reinforcement learning

This is not an exhaustive list, and there are many other machine learning algorithms that can be used for different purposes and applications.

The main goal of supervised learning is to build a predictive model that can accurately generalize to new, unseen data. There are two primary types of supervised learning tasks:

1. **Regression:** In regression tasks, the goal is to predict a continuous output or target variable. Examples include predicting house prices, temperature, or stock prices.

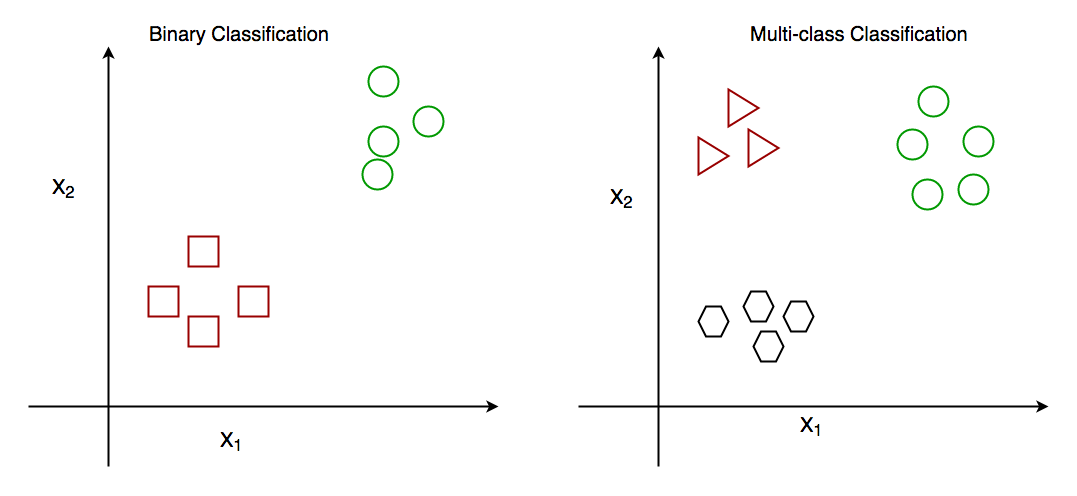
**2. Classification:** In classification tasks, the goal is to predict a categorical output or label. Examples include spam detection, image classification, and sentiment analysis.

**Classification Algorithms**

[**Classification**](https://www.geeksforgeeks.org/getting-started-with-classification/) is the process of finding or discovering a model or function which helps in separating the data into multiple categorical classes i.e. discrete values. In classification, data is categorized under different labels according to some parameters given in the input and then the labels are predicted for the data.

* In a classification task, we are supposed to predict discrete target variables(class labels) using independent features.
* In the classification task, we are supposed to find a [decision boundary](https://www.geeksforgeeks.org/ml-decision-function/) that can separate the different classes in the target variable.

The derived mapping function could be demonstrated in the form of “IF-THEN” rules. The classification process deal with problems where the data can be divided into binary or multiple discrete labels. Let’s take an example, suppose we want to predict the possibility of the winning of a match by Team A on the basis of some parameters recorded earlier. Then there would be two labels Yes and No.



*Binary Classification and Multiclass Classification*

**Comparison between Classification and Regression**

|  |  |  |
| --- | --- | --- |
|  | **Classification** | **Regressor** |
| **1.** | In this problem statement, the target variables are discrete. | In this problem statement, the target variables are continous. |
| **2.** | Problems like [Spam Email Classification](https://www.geeksforgeeks.org/detecting-spam-emails-using-tensorflow-in-python/), [Disease prediction](https://www.geeksforgeeks.org/disease-prediction-using-machine-learning/) like problems are solved using Classification Algorithms. | Problems like [House Price Prediction](https://www.geeksforgeeks.org/house-price-prediction-using-machine-learning-in-python/), [Rainfall Prediction](https://www.geeksforgeeks.org/ml-rainfall-prediction-using-linear-regression/) like problems are solved using regression Algorithms. |
| **3.** | In this algorithm, we try to find the best possible decision boundary which can separate the two classes with the maximum possible separation. | In this algorithm, we try to find the best-fit line which can represent the overall trend in the data. |
| **4.** | [Evaluation metrics](https://www.geeksforgeeks.org/metrics-for-machine-learning-model/) like Precision, Recall, and F1-Score are used here to evaluate the performance of the classification algorithms. | Evaluation metrics like [Mean Squared Error,](https://www.geeksforgeeks.org/python-mean-squared-error/) [R2-Score](https://www.geeksforgeeks.org/ml-r-squared-in-regression-analysis/), and  [MAPE](https://www.geeksforgeeks.org/how-to-calculate-mape-in-python/) are used here to evaluate the performance of the regression algorithms. |
| **5.** | Here we face the problems like [binary Classification](https://www.geeksforgeeks.org/getting-started-with-classification/) or [Multi-Class Classification](https://www.geeksforgeeks.org/multiclass-classification-using-scikit-learn/) problems. | Here we face the problems like [Linear Regression](https://www.geeksforgeeks.org/ml-linear-regression/) models as well as non-linear models. |

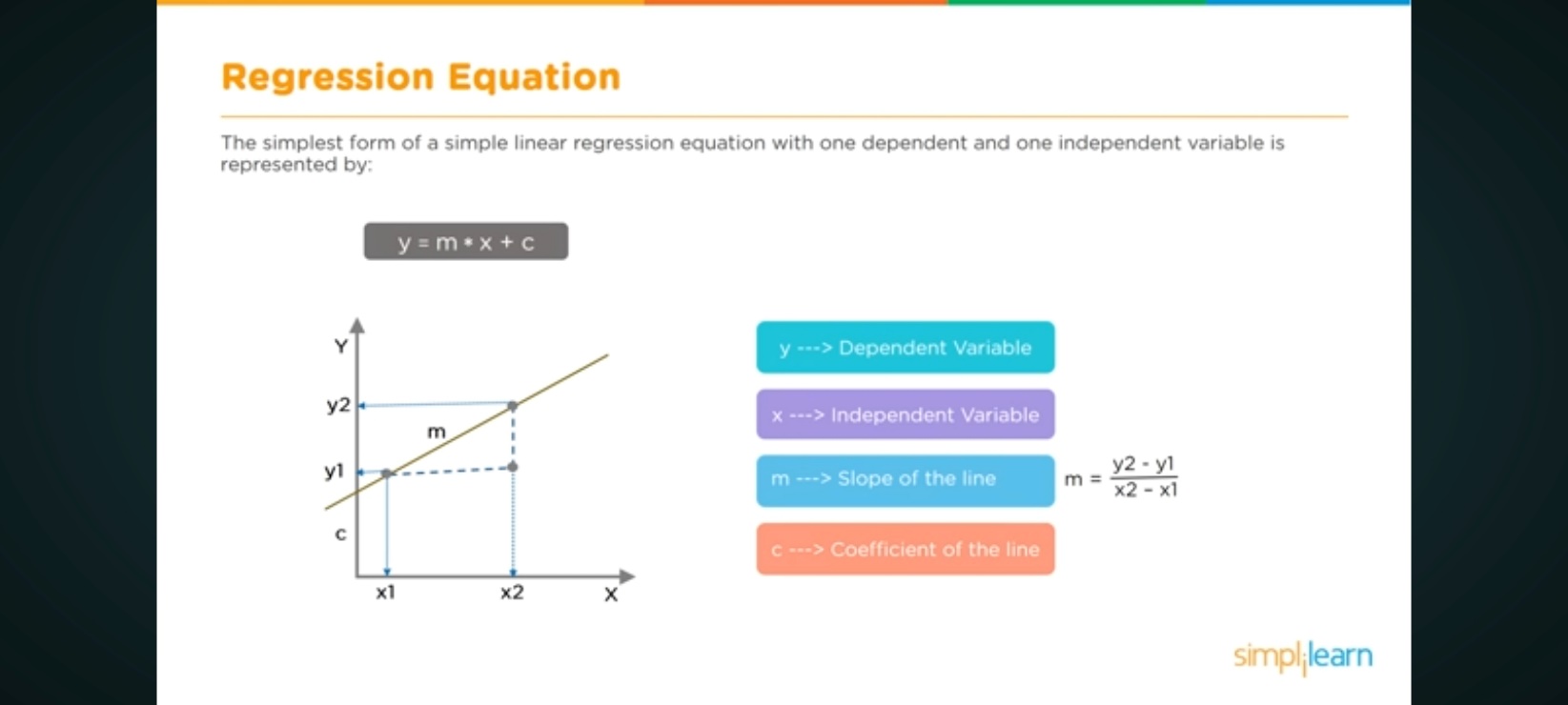
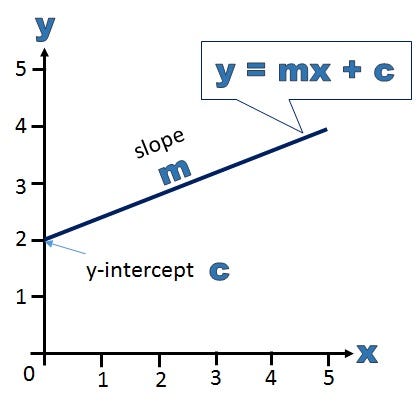
Linear Regression is an algorithm that belongs to supervised Machine Learning. It tries to apply relations that will predict the outcome of an event based on the independent variable data points. The relation is usually a straight line that best fits the different data points as close as possible. The output is of a continuous form, i.e., numerical value. For example, the output could be revenue or sales in currency, the number of products sold, etc. In the above example, the independent variable can be single or multiple.

**Regression**

It involves modelling the relationship between a dependent variable (also known as the target or response variable) and one or more independent variables (also known as predictors or features). The primary goal of regression is to understand and quantify the relationship between variables and to make predictions based on that understanding.

**Simple Linear Regression**

* Simple Linear Regression is a statistical method used to model the relationship between a single independent variable (predictor) and a dependent variable (response).
* The relationship is assumed to be linear, meaning it can be represented by a straight line.
  + - **Linearity:** There must be linear relationship between the dependent variable and the independent variables.
    - **Data appropriateness**
    - **Data independence:** The observations must be independent of each other.
    - **Data distribution: T**he residuals / errors should be normally distributed.
    - **Homoscedasticity:** The variance of the residuals should be constant across all levels of the independent variables.
* The purpose of simple linear regression is to find the best-fitting line that minimizes the difference between the predicted and observed values of the dependent variable.



The simple linear regression model is represented by the equation:

y=mx + c

where:

* *y* is the dependent variable.
* *x* is the independent variable.
* *m* is the slope of the line, representing the rate of change in *y* for a one-unit change in *x*.
* *c* is the y-intercept, which is the value of *y* when *x* is 0.

The goal in simple linear regression is to estimate the values of *m* and *c* that minimize the sum of squared differences between the observed values (*y*) and the values predicted by the line (*mx*+*c*).

The formula for the slope (*m*) is given by:

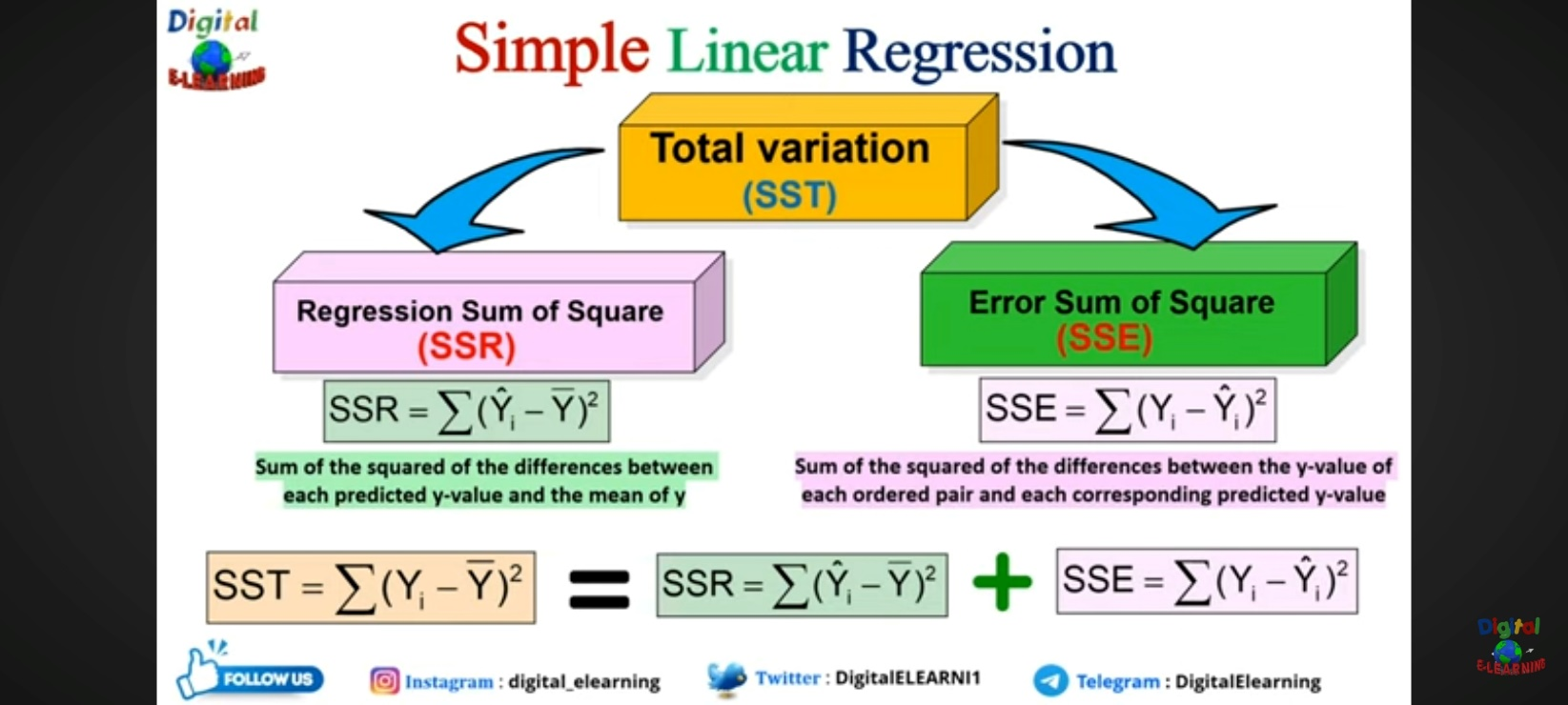
m=n(∑x\*y) −(∑x) (∑y) ​/n(∑x2) −(∑x)2

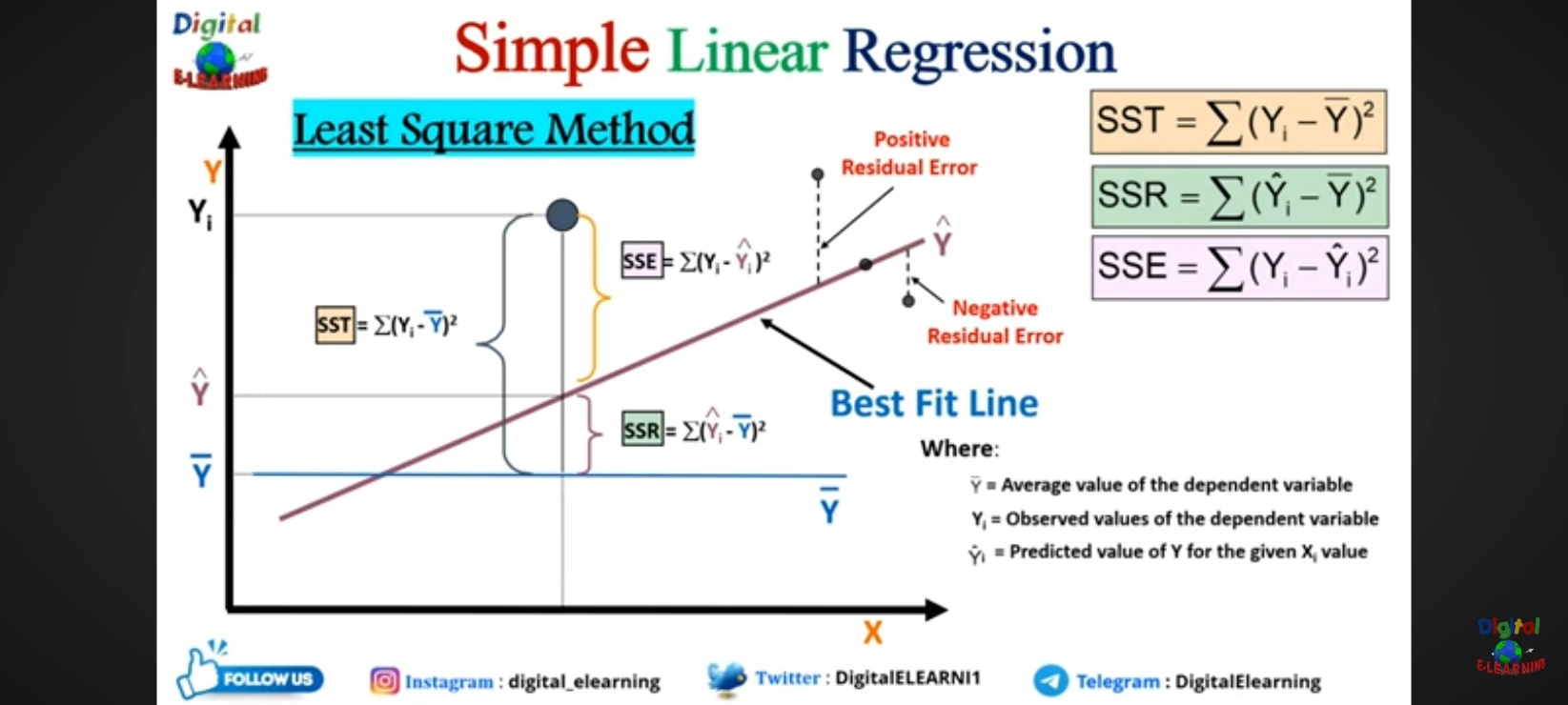
where:

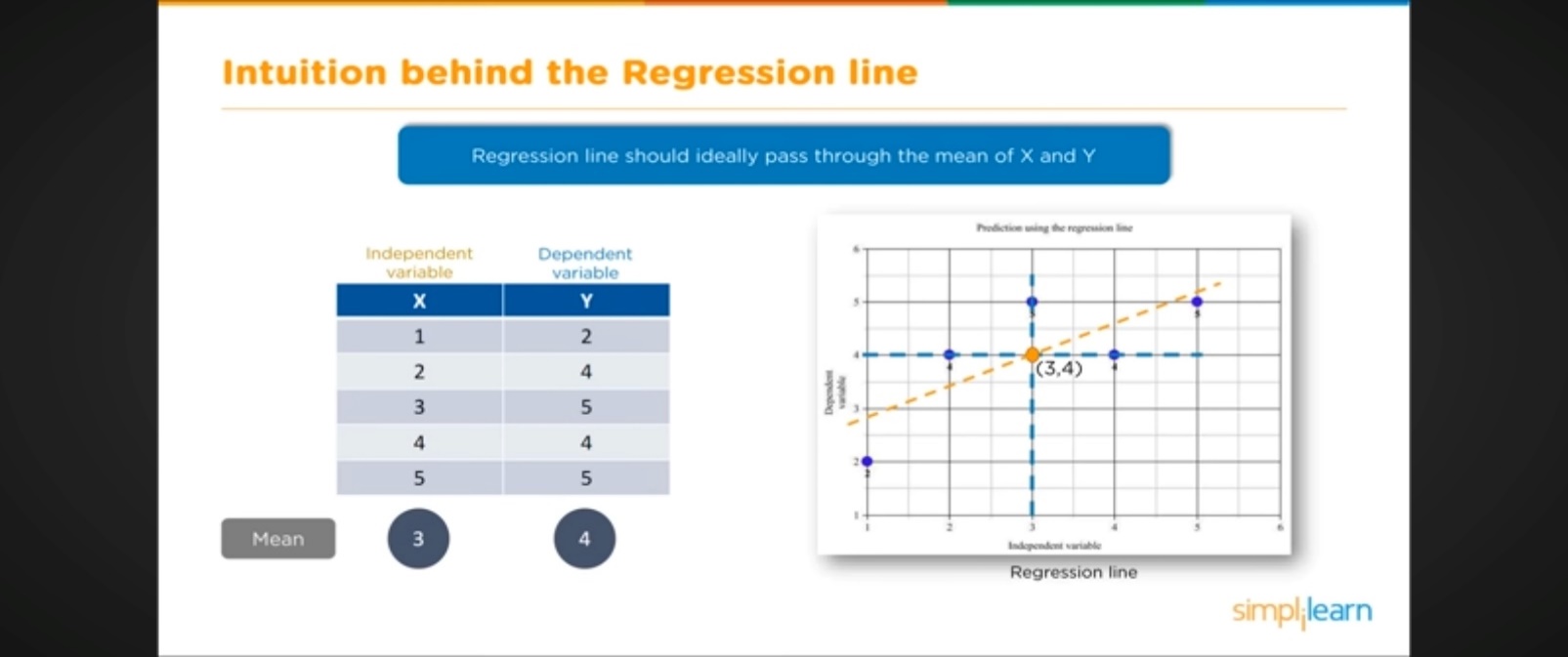
* *n* is the number of data points.
* ∑ denotes summation.
* *x* and *y* are the respective means of the independent and dependent variables.

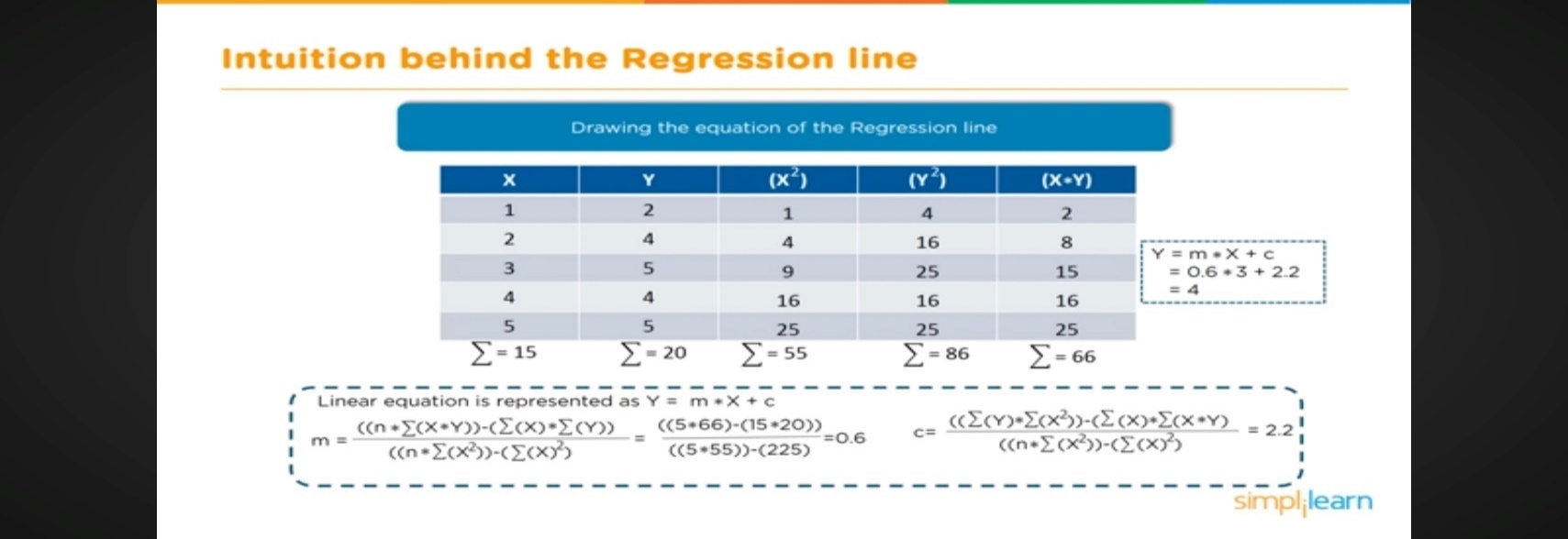
The formula for the y-intercept (*c*) is given by:

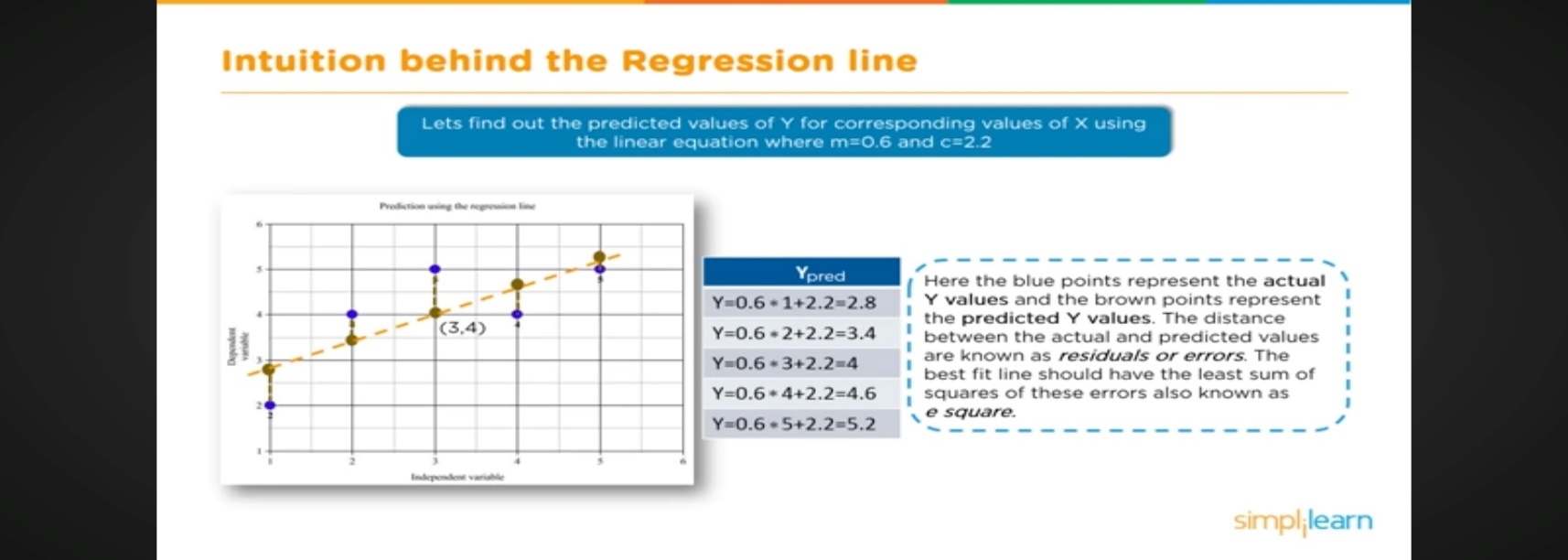
C = (∑x) (∑y) ​/ n

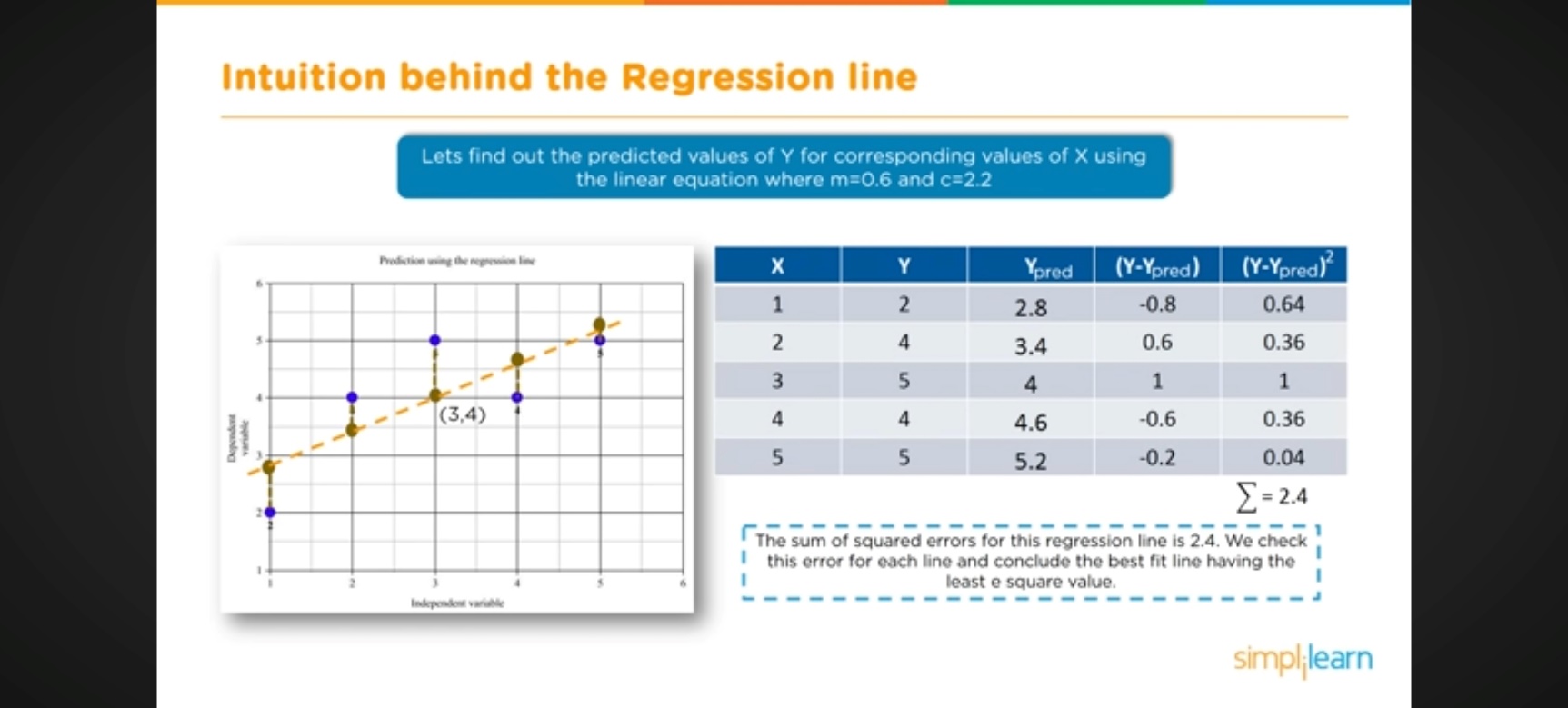












**Advantages**:

* Simple and easy to implement.
* Interpretable results.
* Computationally efficient.

**Disadvantages**:

* Assumes a linear relationship between the dependent and independent variables.
* Sensitive to outliers.
* May not perform well with non-linear data.

**Ordinary Least Squares Method: Concepts & Examples**

Ordinary least squares (OLS) is a technique used in linear regression model to find the best-fitting line for a set of data points by minimizing the residuals (the differences between the observed and predicted values). It does so by estimating the coefficients of the linear regression model by minimizing the sum of the squared differences between the observed values of the dependent variable and the predicted values from the model.

**Assumptions of OLS Method**

The OLS method relies on several assumptions to be valid. The following is the list of key assumptions:

1. Linearity: There must be linear relationship between the dependent variable and the independent variables.
2. Independence: The observations must be independent of each other.
3. Homoscedasticity: The variance of the residuals should be constant across all levels of the independent variables.
4. Normality: The residuals / errors should be normally distributed.
5. No multicollinearity: The independent variables should not be highly correlated with each other.

**Evaluating OLS Results**

OLS provides us with estimates of the coefficients of a linear regression model, but it’s important to evaluate how well the model fits the data. In this section, we will discuss different methods for evaluating OLS results such as some of the following:

* Residual analysis
* R-squared and adjusted R-squared
* F-statistics

**Linear Regression Models & Problems Examples**

* Forecasting sales: Organizations often use linear regression models to forecast future sales. This can be helpful for things like budgeting and planning. Algorithms such as Amazon’s item-to-item collaborative filtering are used to predict what customers will buy in the future based on their past purchase history.
* Cash forecasting: Many businesses use linear regression to forecast how much cash they’ll have on hand in the future. This is important for things like managing expenses and ensuring that there is enough cash on hand to cover unexpected costs.
* Analyzing survey data: Linear regression can also be used to analyze survey data. This can help businesses understand things like customer satisfaction and product preferences. For example, a company might use linear regression to figure out how likely people are to recommend their product to others.
* Stock predictions: A lot of businesses use linear regression models to predict how stocks will perform in the future. This is done by analyzing past data on stock prices and trends to identify patterns.
* Predicting consumer behavior: Businesses can use linear regression to predict things like how much a customer is likely to spend. Regression models can also be used to predict consumer behavior. This can be helpful for things like targeted marketing and product development. For example, Walmart uses linear regression to predict what products will be popular in different regions of the country.

**Polynomial Regression**

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

**What is a Polynomial Regression?**

* Inspection of residuals. If we try to fit a linear model to curved data, a scatter plot of residuals (Y-axis) on the predictor (X-axis) will have patches of many positive residuals in the middle. Hence in such a situation, it is not appropriate.
* An assumption in the usual multiple linear regression analysis is that all the independent variables are independent.

**Assumptions:**

* Linearity: The relationship between the independent and dependent variables is linear.
* Independence: Observations are independent of each other.
* Homoscedasticity: Constant variance of the errors.
* Normality: The errors are normally distributed.

**Difference between linear regression and Polynomial Regression**

** Model Form:**

* **Linear Regression:** Models a straight-line relationship.
* **Polynomial Regression:** Models a curved, polynomial relationship.

** Complexity:**

* **Linear Regression:** Simpler model with fewer parameters.
* **Polynomial Regression:** More complex, can fit a wider range of data shapes but introduces more parameters.

** Overfitting:**

* **Linear Regression:** Less prone to overfitting because of its simplicity.
* **Polynomial Regression:** More prone to overfitting, especially with high-degree polynomials, because it can fit the noise in the data.

** Interpretability:**

* **Linear Regression:** Easier to interpret as the relationship between variables is straightforward.
* **Polynomial Regression:** Harder to interpret due to the complexity of the polynomial terms.

** Flexibility:**

* **Linear Regression:** Limited to linear relationships.
* **Polynomial Regression:** More flexible, can model a variety of relationships by adjusting the degree of the polynomial.

**Why Polynomial Regression?**

Polynomial regression is a type of regression analysis used in statistics and machine learning when the relationship between the independent variable (input) and the dependent variable (output) is not linear. While simple linear regression models the relationship as a straight line, polynomial regression allows for more flexibility by fitting a polynomial equation to the data.

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

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

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

Y = a+b1x+b2x2+b3x3+b4x4 ………. +*n x*n

Where :

* y is the dependent variable.
* x is the independent variable.
* ​ 0,1, ……., *n* are the coefficients of the polynomial terms.
* n is the degree of the polynomial.
* represents the error term.

1. The choice of the polynomial degree (n) is a crucial aspect of polynomial regression. A higher degree allows the model to fit the training data more closely, but it may also lead to overfitting, especially if the degree is too high. Therefore, the degree should be chosen based on the complexity of the underlying relationship in the data.
2. The polynomial regression model is trained to**find the coefficients** that minimize the difference between the predicted values and the actual values in the training data.
3. Once the model is trained, it can be used to make predictions on new, unseen data.

**Overfitting Vs Under-fitting**

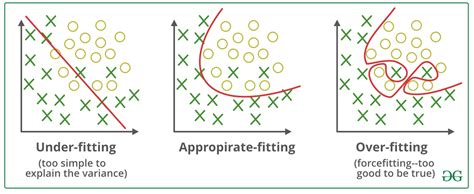
While dealing with the polynomial regression one thing that we face is the problem of [overfitting](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/) this happens because while we increase the order of the polynomial regression to achieve better and better performance model gets overfit on the data and does not perform on the new data points.

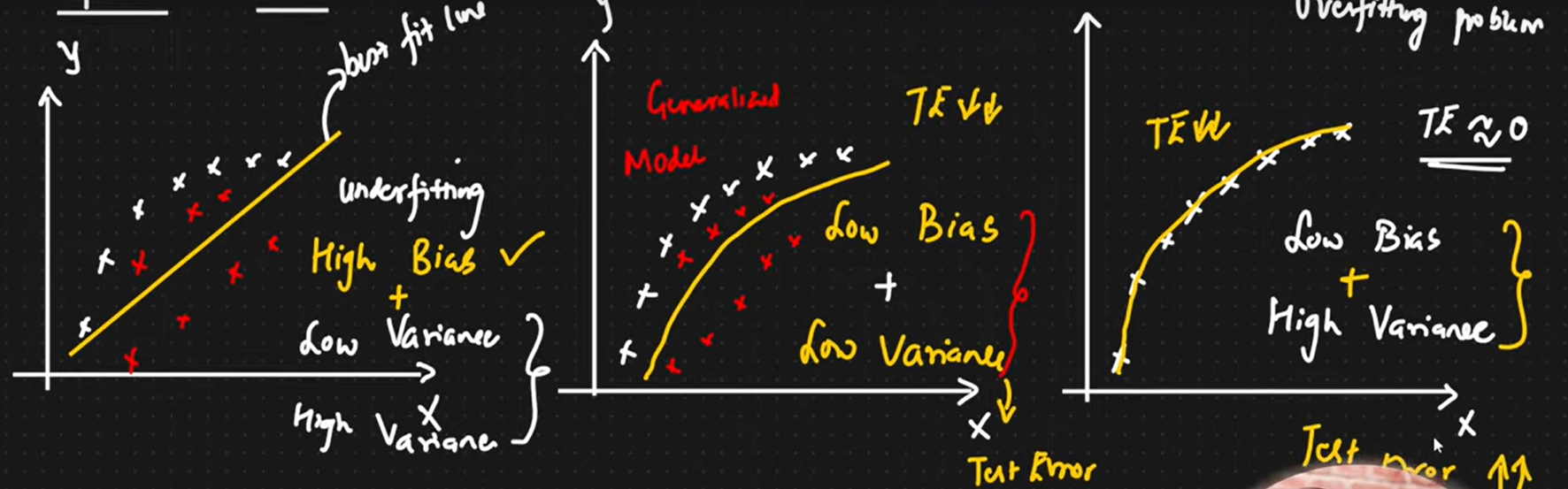
**Overfitting**:

* **Definition**: Overfitting occurs when a model learns the training data too well, capturing not only the underlying pattern but also the noise and random fluctuations present in the data.
* **Cause**: It happens when the model is too complex or has too many parameters relative to the amount of training data.

**Under fitting**:

* **Definition**: Under fitting occurs when a model is too simplistic to capture the complexities of the data. It fails to learn the training data effectively.
* **Cause**: It results from using an overly simple model or insufficient features.







**Regularization**

Regularization in machine learning refers to techniques used to prevent overfitting by adding additional constraints or penalties to the model. Overfitting occurs when a model learns the noise in the training data instead of the actual underlying patterns, resulting in poor generalization to new, unseen data. Regularization helps to improve the generalization ability of the model.

**Common Types of Regularization**

1. **L1 Regularization (Lasso)**
   * **Definition:** Adds the sum of the absolute values of the coefficients to the loss function.
   * **Loss Function:** J(w)=(0.5)∗∑(y−h(y))2+∑|w|
   * **Effect:** Can shrink some coefficients to zero, effectively performing feature selection.
2. **L2 Regularization (Ridge)**
   * **Definition:** Adds the sum of the squared values of the coefficients to the loss function.
   * **Loss Function:** J(w)=(0.5)∗∑(y−h(y))2+∑|w|2
   * **Effect:** Shrinks the coefficients but does not set them to zero, helping to reduce the variance of the model.
3. **Elastic Net**
   * **Definition:** Combines both L1 and L2 regularization.
   * **Effect:** Provides a balance between feature selection (L1) and coefficient shrinkage (L2).

**Regularization Parameter (λ\lambdaλ)**

The regularization parameter λ\ controls the strength of the penalty. A higher λ\ increases the penalty, leading to smaller coefficients and potentially underfitting. A lower λ reduces the penalty, which might lead to overfitting. The value of λ is typically chosen through cross-validation.

**Impact on Model Training**

* **Bias-Variance Tradeoff:** Regularization increases bias (simplifies the model) but reduces variance (model's sensitivity to small fluctuations in the training set).
* **Training Process:** Regularization is added to the loss function during model training. The optimization algorithm (e.g., gradient descent) will then minimize the regularized loss function.

**Application of Polynomial Regression**

 Growth **Rate of Tissues**:

Polynomial regression is used to model the non-linear growth patterns of biological tissues over time. By fitting a polynomial equation to the data, it can capture the complex, accelerated growth phases that a simple linear model cannot.

 Progression **of Disease Epidemics**:

Polynomial regression helps in understanding the non-linear spread of an epidemic by modeling the number of infected individuals over time. This approach can reveal critical inflection points and exponential growth phases in the disease progression.

 Distribution **of Carbon Isotopes in Lake Sediments**:

Polynomial regression is applied to model the varying concentration of carbon isotopes at different sediment depths. This method captures the non-linear historical changes in environmental conditions, aiding in climate studies and paleo environmental reconstructions.

**Advantages:**

* Simple and easy to implement.
* Interpretable results.
* Computationally efficient.

**Disadvantages:**

* Assumes a linear relationship between the dependent and independent variables.
* Sensitive to outliers.
* May not perform well with linear data.

**Multivariate Regression**

Multivariate regression refers to the statistical technique that establishes a relationship between multiple data variables. It estimates a linear equation that facilitates the analysis of multiple dependent or outcome variables depending on one or more predictor variables at different points in time.

**Assumptions**

The validity and reliability of the multivariate regression findings depend upon the following four assumptions:

1. **Linearity**: The correlation between the predictor and outcome variables is linear.
2. **Independence**: The observations are autonomous of each other, i.e., the value of the other independent variable should not influence the value of the independent variables.
3. **Homoscedasticity**: The **variance** of the errors (residuals) is even across all levels of the explanatory variables. This ensures that the spread of residuals is the same for all predicted values.
4. **Normality**: The residuals (differences between observed and predicted values) should be normally distributed, ensuring that statistical inferences about **regression** coefficients are valid.

**Formula**

The multivariate regression equation is represented as follows:

***Y = β0 + β1X1+ βkXk + residual***

* Y represents the dependent variable.
* β0​ is the intercept.
* β1​,β2​,…,βk​ are the coefficients for the respective independent variables X1​,X2​,…,Xk​.
* Residual represents the error term

**Advantages And Disadvantages**

The researchers need to consider the following pros and cons before conducting the multivariate regression analysis:

**Advantages**

Some of the benefits of this model are discussed below:

* **Better Comprehends Relationships**: Unlike simple linear regression, which considers only one predictor, multivariate regression can account for interactions and interdependencies among various predictors, capturing complex relationships between these variables.
* **Reliable Predictions**: By including multiple predictors, the model might provide more accurate estimations than simple regression models, leading to a better fit for the data.
* **Correlation, Strength, and Direction**: Multivariate regression can help identify which explanatory variables significantly influence the dependent variable, establishing a correlation and quantifying the direction and strength of these correlations.

**Disadvantages**

The various limitations of this regression technique are as follows:

* **Difficult to Interpret**: Multivariate regression can be challenging to interpret, especially for individuals unfamiliar with [**statistical analyses**](https://www.wallstreetmojo.com/statistical-analysis/), due to multiple predictors.
* **Complex Calculations**: Since this model incorporates multiple variables, its computation involves complex mathematical calculations.
* **Extensive Data Requirement**: Multivariate regression requires a larger [**sample size**](https://www.wallstreetmojo.com/sample-size-formula/) than simple regression. Small sample sizes can result in unreliable parameter estimates and low statistical power.
* **Overfitting**: It occurs when the model fits the training data too closely, capturing noise rather than the underlying pattern.

**Multivariate Regression vs Multiple Regression**

Multivariate regression and [**multiple regression**](https://www.wallstreetmojo.com/multiple-regression-formula/) are terms that are often used interchangeably, but they are two different methods, as discussed below:

| **Basis** | **Multivariate Regression** | **Multiple Regression** |
| --- | --- | --- |
| Meaning | A regression analysis method involving multiple variables, both dependent and independent variables | A statistical measure that facilitates the analysis of the association between a dependent variable and two or more independent variables |
| Purpose | It determines how a set of predictors impacts multiple related outcome variables. | It is used to understand how multiple predictors influence a single outcome variable. |
| Number of Dependent Variables | Several | Single |
| Outcome | Provides insights into how a set of predictors influences multiple related outcomes simultaneously, revealing patterns among the dependent variables | Identifies individual effects of predictors on the dependent variable while controlling for other predictors |

**Application:**

** Economics and Finance:**

* **Predicting stock prices**: Multivariate regression can help predict stock prices based on variables like interest rates, inflation, company earnings, and market indices.
* **GDP forecasting**: By analyzing factors like investment, government expenditure, and consumer spending, multivariate regression can forecast a country's GDP.

** Marketing and Sales**:

* **Sales prediction**: Businesses can predict future sales by analyzing variables such as advertising spend, pricing, seasonality, and customer demographics.
* **Customer behavior analysis**: Companies can understand how factors like product features, price, customer age, and income level affect purchasing decisions.

** Healthcare:**

* **Predicting disease outcomes**: Medical professionals can predict patient outcomes (e.g., risk of heart disease) based on variables like age, BMI, blood pressure, cholesterol levels, and lifestyle factors.
* **Health costs estimation**: By using variables like patient demographics, medical history, and treatment plans, multivariate regression can help predict healthcare costs.

** Education:**

* **Student performance**: Universities and schools can predict student performance based on multiple factors such as attendance, study hours, socio-economic background, and previous academic scores.
* **Dropout rates**: Institutions may identify factors like family income, student engagement, and academic performance that contribute to dropout rates.

**Introduction to Decision Tree in Machine Learning**

Decision Tree in machine learning is a part of classification algorithm which also provides solutions to the regression problems using the classification rule; its structure is like the flowchart where each of the internal nodes represents the test on a feature (e.g., whether the random number is greater than a number or not), each leaf node is used to represent the class label( results that need to be computed after taking all the decisions) and the branches represents conjunction of features that lead to the class labels.

Decision Tree in [Machine Learning has got](https://www.educba.com/what-is-machine-learning/) a wide field in the modern world. There are a lot of algorithms in ML which is utilized in our day-to-day life. One of the important algorithms is the Decision Tree used for classification and a solution for regression problems. As it is a predictive model, Decision Tree Analysis is done via an algorithmic approach where a data set is split into subsets as per conditions. The name itself says it is a tree-like model in the form of if-then-else statements. The deeper is the tree and more are the nodes, the better is the model.

**Types of decision tree**

Decision Tree is a tree-like graph where sorting starts from the root node to the leaf node until the target is achieved. It is the most popular one for decision and classification based on supervised algorithms. It is constructed by recursive partitioning where each node acts as a test case for some attributes and each edge, deriving from the node, is a possible answer in the test case. Both the root and leaf nodes are two entities of the algorithm.

small example as follows:

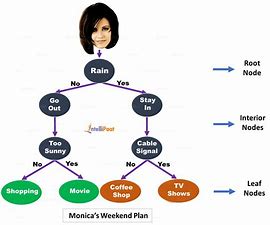


Fig 1.0

There are two types of Decision Trees:

1. **Classification Trees:**The above example is a categorial based Classification Tree.
2. **Regression Trees**: In this type of algorithm, the decision or result is continuous. It has got a single numerical output with more inputs or predictors.

In the Decision tree, the typical challenge is to identify the attribute at each node. The process is called attribute selection and has some measures to use in order to identify the attribute.

**a. Information Gain (IG)**

Information Gain measures how much information an individual feature gives about the class. It acts as the main key to construct a Decision Tree. An attribute with the highest Information Gain splits first. So, the Decision Tree always maximizes the Information Gain. When we use a node to partition the instances into smaller subsets, then the entropy changes.

**Entropy:** It is the measure of uncertainty or impurity in a random variable. Entropy decides how a Decision Tree splits the data into subsets.

The equation for Information Gain and entropy are as follows:

**Information Gain=** entropy(parent)- [weighted average\*entropy(children)]

**Entropy:** **∑p(X)log p(X)**

P(X) here is the fraction of examples in a given class.

**b. Gini Index**

Gini Index is a metric that decides how often a randomly chosen element would be incorrectly identified. It clearly states that attribute with a low Gini Index is given first preference.

**Gini = 1- ∑Pi2**

**Split creation**

1. To create a split, first, we need to calculate the Gini score.
2. The data is split using a list of rows having an index of an attribute and a split value of that attribute. After the right and left dataset is found, we can get the split value by the Gini score from the first part. Now, the split value will be the decider where the attribute will reside.
3. The next part is evaluating all the splits. The best possible value is calculated by evaluating the cost of the split. The best split is used as a node of the Decision Tree.

**Decision Tree Terminologies**

Some of the common Terminologies used in Decision Trees are as follows:

* **Root Node:** It is the topmost node in the tree, which represents the complete dataset. It is the starting point of the decision-making process.
* **Decision/Internal Node**: A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
* **Leaf/Terminal Node:** A node without any child nodes that indicates a class label or a numerical value.
* **Branch/Sub-Tree:** A subsection of the decision tree starts at an internal node and ends at the leaf nodes.
* **Parent Node:** The node that divides into one or more child nodes.
* **Child Node:**The nodes that emerge when a parent node is split.
* **Impurity**: A measurement of the target variable’s homogeneity in a subset of data. It refers to the degree of randomness or uncertainty in a set of examples. The **Gini index** and **entropy** are two commonly used impurity measurements in decision trees for classifications task
* **Information Gain:** Information gain is a measure of the reduction in impurity achieved by splitting a dataset on a particular feature in a decision tree. The splitting criterion is determined by the feature that offers the greatest information gain, It is used to determine the most informative feature to split on at each node of the tree, with the goal of creating pure subsets

### Image result for Decision Tree Terminologies

### Building a tree – decision tree in machine learning:

There are two steps to building a Decision Tree.

#### **1. Terminal node creation**

While creating the terminal node, the most important thing is to note whether we need to stop growing trees or proceed further. The following ways can be used for this:

* Maximum tree depth: When the tree reaches the maximum number of nodes, execution stops there.
* Minimum node records: It can be defined as a minimum of patterns that a node requires. Then we can stop adding terminal nodes immediately we get those minimum node records.

#### **2. Recursive splitting**

Once, the node is created, we can create a child node recursively by splitting the data set and calling the same function multiple times.

**Prediction**

After a tree is built, the prediction is done using a recursive function. The same prediction process is followed again with left or right child nodes and so on.

### Advantages and disadvantages of decision tree

Below are given some advantages and disadvantages:

#### **Advantages**

The decision tree has some advantages in Machine Learning as follows:

* **Comprehensive:** It takes consideration of each possible outcome of a decision and traces each node to the conclusion accordingly.
* **Specific:** Decision Trees assign a specific value to each problem, decision, and outcome(s). It reduces uncertainty and ambiguity and also increases clarity.
* **Simplicity:** Decision Tree is one of the easier and reliable algorithms as it has no complex formulae or data structures. Only simple statistics and maths are required for calculation.
* **Versatile:** Decision Trees can be manually constructed using maths and as well be used with other computer programs.

#### **Disadvantages**

The decision tree has some disadvantages in Machine Learning as follows:

* Decision trees are less appropriate for estimation and financial tasks where we need an appropriate value(s).
* It is an error-prone classification algorithm as compared to other computational algorithms.
* It is computationally expensive. At each node, the candidate split must be sorted before ascertaining the best. There are other alternatives which many business entities follow for financial tasks as Decision Tree is too expensive for evaluation.
* While working with continuous variables, Decision Tree is not fit as the best solution as it tends to lose information while categorizing variables.
* It is sometimes unstable as small variations in the data set might lead to the formation of a new tree.

**Application:**

**1. Finance**

* **Credit Scoring:** Predicting whether a customer will default on a loan based on factors like income, credit history, and employment status.

**2. Healthcare**

* **Diagnosis:** Assisting in medical diagnosis by predicting diseases or conditions based on symptoms, medical history, and test results.

**3. Marketing**

* **Customer Segmentation:** Segmenting customers based on demographic data, purchasing behavior, and preferences.

### 4. ****Environmental Science****

* **Species Classification:** Classifying species based on habitat characteristics, environmental factors, and biological traits.

### 5. ****Education****

* **Student Performance Prediction:** Predicting student performance based on demographics, past academic records, and learning behaviors.

## Random Forest Algorithm

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.

Before understanding the working of the random forest algorithm in machine learning, we must look into the ensemble learning technique. **Ensemble**simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

**What are Ensemble Learning models?**

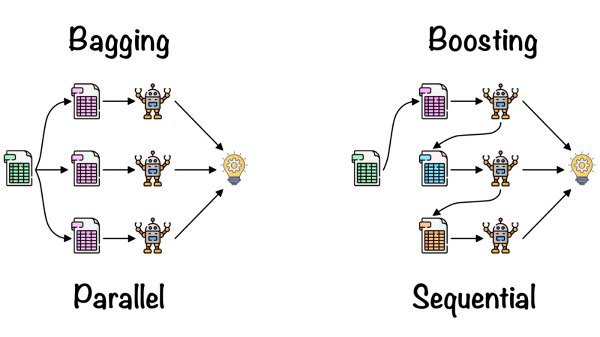
Ensemble learning models work just like a group of diverse experts teaming up to make decisions – think of them as a bunch of friends with different strengths tackling a problem together. Picture it as a group of friends with different skills working on a project. Each friend excels in a particular area, and by combining their strengths, they create a more robust solution than any individual could achieve alone.

Similarly, in ensemble learning, different models, often of the same type or different types, team up to enhance predictive performance. It’s all about leveraging the collective wisdom of the group to overcome individual limitations and make more informed decisions in various machine learning tasks. Some popular ensemble models include- AdaBoost, Random Forest, Bagging, Voting etc.

Ensemble uses two types of methods:

1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example,  Random Forest.

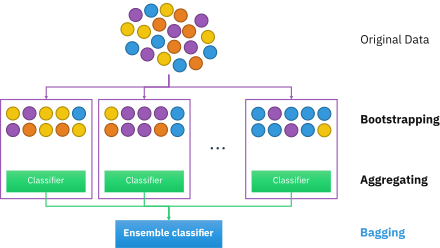
2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example,  ADA BOOST, XG BOOST.



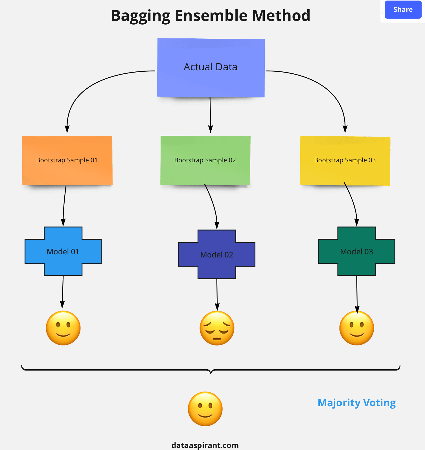
As mentioned earlier, Random forest works on the Bagging principle. Now let’s dive in and understand bagging in detail.

#### **Bagging**

Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forest. Bagging chooses a random sample/random subset from the entire data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as **row sampling**. This step of row sampling with replacement is called**bootstrap**. Now each model is trained independently, which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting, is known as **aggregation**.



Now let’s look at an example by breaking it down with the help of the following figure. Here the bootstrap sample is taken from actual data (Bootstrap sample 01, Bootstrap sample 02, and Bootstrap sample 03) with a replacement which means there is a high possibility that each sample won’t contain unique data. The model (Model 01, Model 02, and Model 03) obtained from this bootstrap sample is trained independently. Each model generates results as shown. Now the Happy emoji has a majority when compared to the Sad emoji. Thus based on majority voting final output is obtained as Happy emoji.



#### **Boosting**

Boosting is one of the techniques that use the concept of ensemble learning. A boosting algorithm combines multiple simple models (also known as weak learners or base estimators) to generate the final output. It is done by building a model by using weak models in series.

There are several boosting algorithms; AdaBoost was the first really successful boosting algorithm that was developed for the purpose of binary classification. AdaBoost is an abbreviation for Adaptive Boosting and is a prevalent boosting technique that combines multiple “weak classifiers” into a single “strong classifier.” There are Other Boosting techniques. For more, you can visit

#### **Assumptions for Random Forest:**

#### Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

#### There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

#### The predictions from each tree must have very low correlations.

#### **Steps Involved in Random Forest Algorithm**

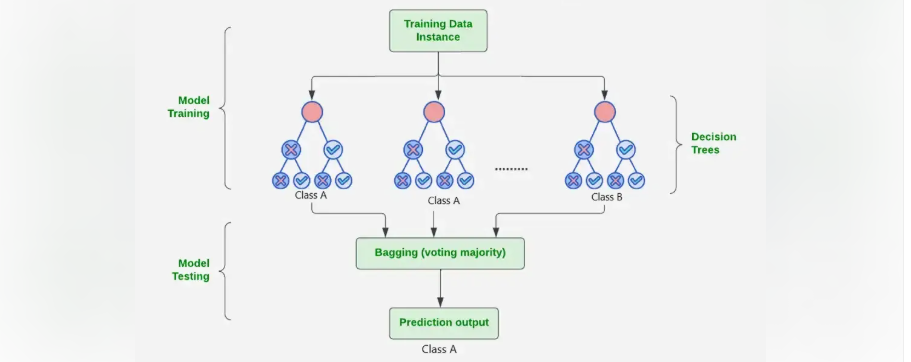
Step 1: In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.

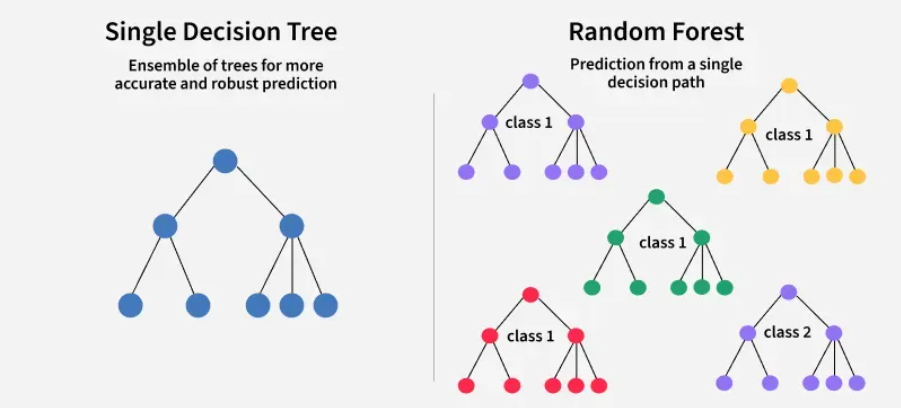
Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on **Majority Voting or Averaging**for Classification and regression, respectively.

For example:  consider the fruit basket as the data as shown in the figure below. Now n number of samples are taken from the fruit basket, and an individual decision tree is constructed for each sample. Each decision tree will generate an output, as shown in the figure. The final output is considered based on majority voting. In the below figure, you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.





## Important Features of Random Forest

* **Diversity:**Not all attributes/variables/features are considered while making an individual tree; each tree is different.
* **Immune to the curse of dimensionality:** Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization:**Each tree is created independently out of different data and attributes. This means we can fully use the CPU to build random forests.
* **Train-Test split:**In a random forest, we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability:**Stability arises because the result is based on majority voting/ averaging.

**Why use Random Forest?**

Below are some points that explain why we should use the Random Forest algorithm:

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing

**Applications of Random Forest in Real-World Scenarios**

Some of the widely used real-world application of Random Forest is discussed below:

1. Finance Wizard: Imagine Random Forest as our financial superhero, diving into the world of credit scoring. Its mission? To determine if you’re a credit superhero or, well, not so much. With a knack for handling financial data and sidestepping overfitting issues, it’s like having a guardian angel for robust risk assessments.
2. Health Detective: In healthcare, Random Forest turns into a medical Sherlock Holmes. Armed with the ability to decode medical jargon, patient records, and test results, it’s not just predicting outcomes; it’s practically assisting doctors in solving the mysteries of patient health.
3. Environmental Guardian: Out in nature, Random Forest transforms into an environmental superhero. With the power to decipher satellite images and brave noisy data, it becomes the go-to hero for tasks like tracking land cover changes and safeguarding against potential deforestation, standing as the protector of our green spaces.
4. Digital Bodyguard: In the digital realm, Random Forest becomes our vigilant guardian against online trickery. It’s like a cyber-sleuth, analyzing our digital footsteps for any hint of suspicious activity. Its ensemble approach is akin to having a team of cyber-detectives, spotting subtle deviations that scream “fraud alert!” It’s not just protecting our online transactions; it’s our digital bodyguard.

**XGBoost Algorithm**

1. Introduction

* XGBoost stands for Extreme Gradient Boosting.
* It is an optimized implementation of Gradient Boosting, designed for efficiency, scalability, and high performance.
* Belongs to the ensemble learning family → combines multiple weak models (decision trees) to create a strong predictive model.
* Widely used in Kaggle competitions, industry projects, and ML research due to its accuracy and speed.

2. Key Features

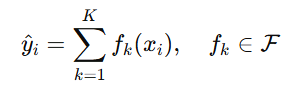
* Uses decision trees as base learners.
* Builds trees sequentially → each new tree corrects the errors of the previous trees.
* Supports regularization (L1 & L2) → prevents overfitting.
* Handles missing values automatically.
* Efficient for large datasets (parallel & distributed computing support).

3. How XGBoost Works

1. Initialize model with an initial prediction (often mean of target values for regression).
2. Fit first tree on the residual errors (difference between prediction and actual).
3. Add new trees sequentially, each tree trained to minimize residuals from previous predictions.
4. Combine predictions: final output is the sum of predictions from all trees.
5. Stop condition: number of trees reached, or error improvement falls below a threshold.

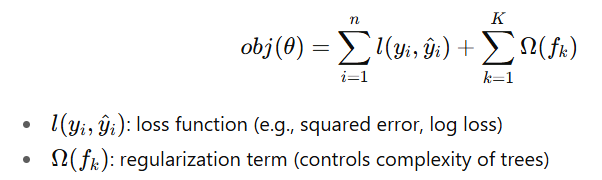
4. Mathematics Behind XGBoost

4.1 Prediction Function

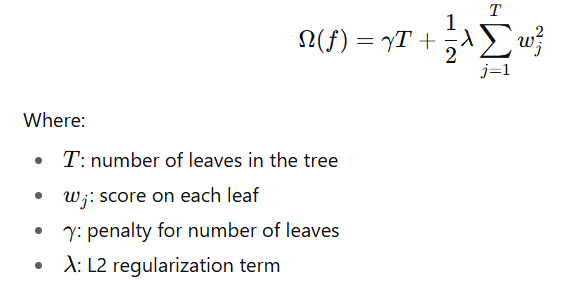
  
Where:

* = predicted value for the (*i*th) data point
* K = number of trees
* fk(xi) = prediction of the kth tree

4.2 Objective Function



4.3 Regularization Term



5. What Makes XGBoost “Extreme”?

5.1 Preventing Overfitting

* Learning rate ((\eta)) → scales contribution of each tree.
* Regularization → L1 (LASSO), L2 (Ridge).
* Pruning → removes branches that don’t reduce error.
* Combination of shrinkage, regularization, pruning = robust models.

5.2 Tree Construction

* Builds trees level-wise (breadth-first), unlike traditional depth-first.
* Uses best split search to minimize loss at each level.
* Prioritizes important features early.

5.3 Handling Missing Data

* Sparse-aware → missing values treated as separate category.
* Learns the “default direction” for missing values.

5.4 Cache & Hardware Optimization

* Efficient CPU cache usage.
* Block structure for parallel computation.

5.5 Approximate Greedy Algorithm

* Approximates best split for large datasets.
* Efficient and scalable without major accuracy loss.

6. Advantages of XGBoost

✅ High accuracy (top choice in competitions).  
✅ Scalable (handles millions of records).  
✅ Regularization built-in → avoids overfitting.  
✅ Handles missing values automatically.  
✅ Parallelized computation → very fast.  
✅ Flexible → regression, classification, ranking problems.

7. Disadvantages of XGBoost

❌ Computationally expensive for very large models.  
❌ Difficult to interpret compared to simpler models.  
❌ Memory-intensive (not ideal for resource-limited systems).  
❌ Sensitive to noise/outliers.  
❌ Overfitting risk on small datasets with too many trees.

8. Applications of XGBoost

* Finance → fraud detection, risk modeling.
* Healthcare → disease prediction.
* Marketing → customer churn prediction, recommendation systems.
* Competitions → Kaggle, DrivenData, etc.
* General ML tasks → classification, regression, ranking.

**Naive Bayes**

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Naive Bayes assumes that the features used in the model are independent of each other. This simplification allows for easy coding and quick predictions.

The name “naive” comes from this assumption of independence.

To start with, let us consider a dataset.

Here is a tabular representation of our dataset.

|  | **Outlook** | **Temperature** | **Humidity** | **Windy** | **Play Golf** |
| --- | --- | --- | --- | --- | --- |
| 0 | Rainy | Hot | High | False | No |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |

**1. Frequency Table – Outlook vs Play**

| **Outlook** | **Yes** | **No** | **Total** |
| --- | --- | --- | --- |
| Sunny | 2 | 1 | 3 |
| Overcast | 1 | 0 | 1 |
| Rainy | 0 | 2 | 2 |
| **Total** | 3 | 3 | 6 |

**2. Frequency Table – Temperature vs Play**

| **Temperature** | **Yes** | **No** | **Total** |
| --- | --- | --- | --- |
| Hot | 1 | 2 | 3 |
| Mild | 1 | 0 | 1 |
| Cool | 1 | 1 | 2 |
| **Total** | 3 | 3 | 6 |

**3. Frequency Table – Humidity vs Play**

| **Humidity** | **Yes** | **No** | **Total** |
| --- | --- | --- | --- |
| High | 2 | 2 | 4 |
| Normal | 1 | 1 | 2 |
| **Total** | 3 | 3 | 6 |

**4. Frequency Table – Windy vs Play**

| **Windy** | **Yes** | **No** | **Total** |
| --- | --- | --- | --- |
| False | 3 | 2 | 5 |
| True | 0 | 1 | 1 |
| **Total** | 3 | 3 | 6 |

**Assumption of Naive Bayes**

The fundamental Naive Bayes assumption is that each feature makes an:

* **Feature independence:** This means that when we are trying to classify something, we assume that each feature (or piece of information) in the data does not affect any other feature.
* **Continuous features are normally distributed:** If a feature is continuous, then it is assumed to be normally distributed within each class.
* **Discrete features have multinomial distributions:** If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
* **Features are equally important:** All features are assumed to contribute equally to the prediction of the class label.
* **No missing data:** The data should not contain any missing values.

**Bayes’ Theorem**

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

P(A∣B) =P(B∣A) \* P(A)​/P(B)

Where:

* (P(A|B)) is the probability of event A Given B.
* (P(B|A)) is the probability of event B given A.
* (P(A)) is the prior probability of event A.
* (P(B)) is the prior probability of event B.

where A and B are events and P(B) ≠ 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

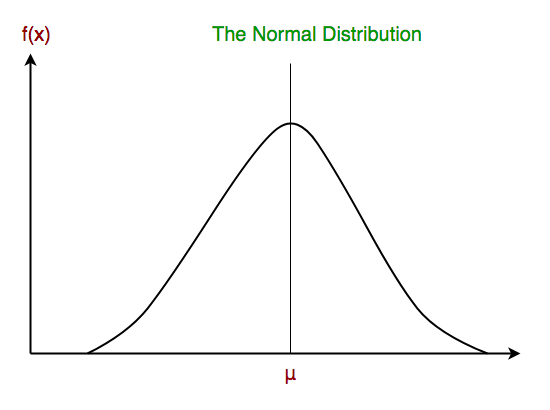
The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(A| B).

Now, we discuss one of such classifiers here.

**Types of Naive Bayes Classifiers**

1. **Gaussian Naive Bayes:** Assumes that the features follow a Gaussian distribution.

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution). When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

1. 
2. The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:
3. Now, we look at an implementation of Gaussian Naive Bayes classifier using scikit-learn.
4. **Multinomial Naive Bayes:** Used for discrete count data, typically used in text classification.
5. **Bernoulli Naive Bayes:** Assumes binary/Boolean features, often used for binary text classification (presence/absence of words).

**Example: Gaussian Naive Bayes**

Consider a simple example with Gaussian Naive Bayes for classifying whether a person will buy a computer based on their age and salary.

**Advantages of Naive Bayes Classifier:**

1. **Simple to Implement**:
   1. Naive Bayes is a straightforward algorithm that is easy to implement. It doesn’t require complex computations or extensive training time.
   2. It can be used for both binary and multiple-class classification tasks.
2. **Handles Missing Data Well**:
   1. Naive Bayes is useful for handling missing data. It considers only the present data and ignores missing data, maintaining accuracy.
3. **Fast and Scalable**:
   1. The algorithm is fast and scalable, making it suitable for large datasets.
   2. It can be parallelized to run on multiple processors or clusters.
4. **Performs Well in Text Classification**:
   1. Naive Bayes excels in text classification tasks, such as sentiment analysis or spam filtering.
   2. It handles high-dimensional data and categorical features common in natural language processing.
5. **Works Well with Small Datasets**:
   1. Naive Bayes performs well with limited training data.
   2. It’s suitable for applications like fraud detection or medical diagnosis.
6. **Requires Less Training Data**:
   * Compared to other algorithms like decision trees or neural networks, Naive Bayes needs less training data.

**Disadvantages of Naive Bayes Classifier:**

1. **Assumption of Feature Independence**:
   * Naive Bayes assumes that features are independent, which may not hold true in some cases.
   * If features are highly correlated, it can lead to suboptimal results.
2. **Zero Probability for Unseen Categories**:
   * If a categorical variable in the test data wasn’t present in the training data, Naive Bayes assigns it zero probability.
   * This limitation affects predictions for unseen categories.

## Applications of Naive Bayes

## Text Classification:

## Spam Filtering: Classifying emails as spam or not spam based on word occurrences.

## Sentiment Analysis: Determining sentiment (positive/negative) of text based on word features.

## Document Categorization: Classifying documents into predefined categories based on content.

## Medical Diagnosis:

## Disease Prediction: Predicting diseases based on symptoms and medical test results.

## Recommendation Systems:

## Product Recommendations: Recommending products to users based on their previous interactions and features.

## Image Recognition:

## Face Recognition: Identifying faces in images based on pixel values and patterns.

## K-Nearest Neighbor(KNN) Algorithm for Machine Learning

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dog’s images and based on the most similar features it will put it in either cat or dog category.

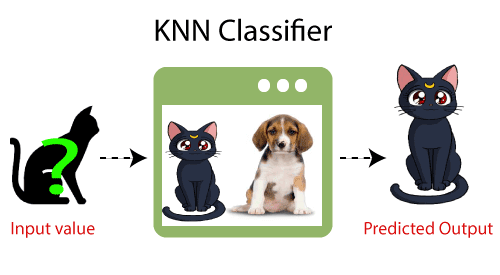
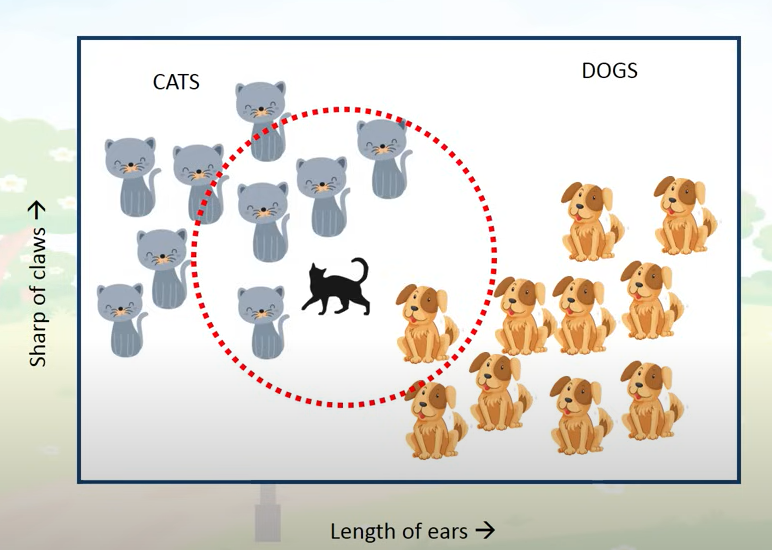


Fig: 1.0



## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

K = **√n**

**n = no of row or entries**



Fig:1.1

## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbour is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

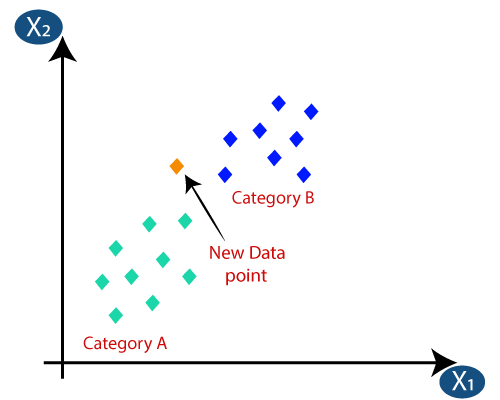
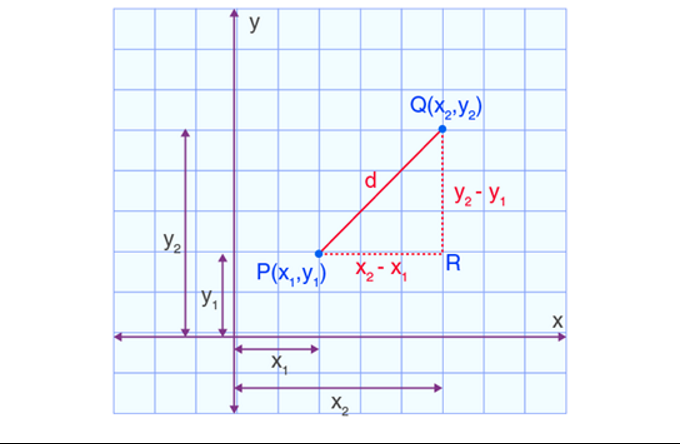


Fig:1.2

* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



**Distance Metrics**

* Euclidean distance
* Manhattan Distance
* Minkowski Distance:
* Cosine similarity

**Euclidean distance**

Hypotenuse2 = Base2 + Perpendicular2

PQ2 = PR2 + QR2

Therefore, d2 = (x2 – x1)2 + (y2 – y1)2

**d =√ [(x2– x1)2 + (y2– y1)2]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Weight (x2)** | **Height (y2)** | **Class** | **Euclidean Distance** |
| 51 | 167 | Underweight | 5 |
| 62 | 182 | Normal | 13.89 |
| 69 | 176 | Normal |  |
| 64 | 173 | Normal |  |
| 65 | 172 | Normal |  |
| 56 | 174 | Underweight |  |
| 58 | 169 | Normal |  |
| 57 | 173 | Normal |  |
| 55 x1 | 170 y1 | Normal |  |

Ex

**1) d =√ [(51– 55)2 + (167– 170)2] = 5**

**2) d =√ [(62– 55)2 + (182– 170)2] = 13.89**

**K = 3**

| **Weight** | **Height** | **Class** | **Distance** |
| --- | --- | --- | --- |
| 51 | 167 | Underweight | 5.00 |
| 62 | 182 | Normal | 13.89 |
| 69 | 176 | Normal | 15.23 |
| 64 | 173 | Normal | 9.49 |
| 65 | 172 | Normal | 10.20 |
| 56 | 174 | Underweight | 4.12 |
| 58 | 169 | Normal | 3.16 |
| 57 | 173 | Normal | 3.61 |

Fig:1.3

* By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

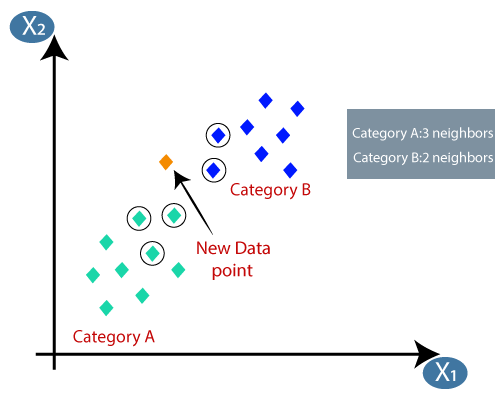


Fig:1.4

* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

# **Why Value of K is Important?**

K affects the accuracy of our model and the number of nearest neighbors from which out new data is classified is highly significant very Larger value for K could lead to overfitting and very small value for k could lead to underfitting. So, Choosing the right value for K is very important

## How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

## Choosing the Value of k

## Small (k) values: Can lead to a model that is too sensitive to noise in the data (overfitting).

## Large (k) values: Can lead to a model that is too generalized, potentially missing important patterns (under fitting).

## Optimal (k) value: Typically found using cross-validation.

In machine learning, the optimal value of k (often used in K-Nearest Neighbors or clustering algorithms) is typically found using cross-validation. Cross-validation is a method that divides the dataset into multiple subsets (folds) to evaluate the model's performance with different values of k and reduce the risk of overfitting.

Here's how cross-validation helps find the optimal kkk value:

1. **Divide the Data**: Split the dataset into training and validation sets (using, e.g., k-fold cross-validation).
2. **Evaluate Across k Values**: Train the model with different values of kkk on each training set and evaluate it on the corresponding validation set.
3. **Select Optimal k**: After comparing performance across multiple folds, select the kkk value that yields the best average score across all validation sets. This score could be accuracy, precision, recall, or another relevant metric based on the problem.

## Advantages of KNN

## Simplicity: Easy to understand and implement.

## No Training Phase: KNN is a lazy learner, meaning it does not require a training phase.

## Adaptability: Works well with non-linear data distributions.

## Disadvantages of KNN

## Computationally Expensive: High memory and computation cost, especially with large datasets, as it requires storing all data points and calculating distances for each prediction.

## Sensitive to Irrelevant Features: Performance can degrade if irrelevant or redundant features are included in the dataset.

## Imbalanced Data: May perform poorly with imbalanced datasets as the majority class can dominate the predictions.

## Draw back

## Optimize the TC

## Varience of KNN

## - KD tree

## - ball tree

## Applications of KNN

## Pattern Recognition:

## Handwriting recognition

## Face recognition

## Image classification

## Medical Diagnosis:

## Predicting disease outcomes based on patient features

## Classifying medical images

## Recommendation Systems:

## Recommending products or content based on user preferences and behaviors

## Finance:

## Stock price prediction

## Credit scoring and fraud detection

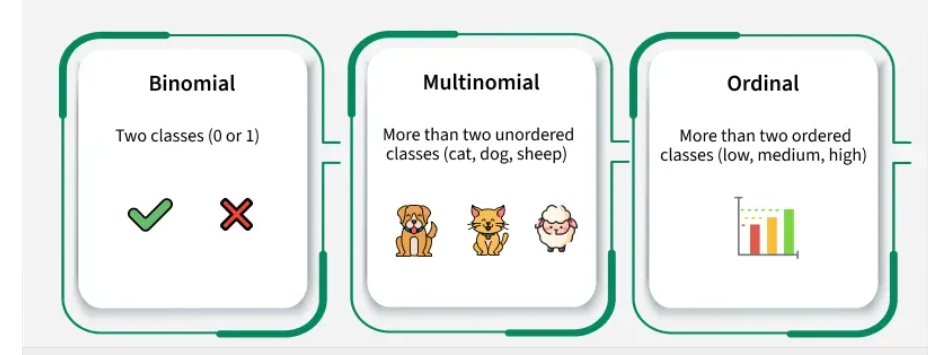
## Marketing:

## Customer segmentation and targeting

## Predicting customer churn

**Logistic Regression in Machine Learning**

Logistic Regression is a supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1. In this article, we will see the basics of logistic regression and its core concepts.



**Why "Regression" in the Name?**

Despite the name, logistic regression is used for **classification**, not regression. It's called "regression" because it uses a **linear model** internally, but applies a **logistic (sigmoid) function** to the output to get a **probability between 0 and 1**.

**Types of Logistic Regression**

Logistic regression can be classified into three main types based on the nature of the dependent variable:

1. **Binomial Logistic Regression**: This type is used when the dependent variable has only two possible categories. Examples include Yes/No, Pass/Fail or 0/1. It is the most common form of logistic regression and is used for binary classification problems.
2. **Multinomial Logistic Regression**: This is used when the dependent variable has three or more possible categories that are not ordered. For example, classifying animals into categories like "cat," "dog" or "sheep." It extends the binary logistic regression to handle multiple classes.

**Assumptions of Logistic Regression**

Understanding the assumptions behind logistic regression is important to ensure the model is applied correctly, main assumptions are:

1. **Independent observations**: Each data point is assumed to be independent of the others means there should be no correlation or dependence between the input samples.
2. **Binary dependent variables**: It takes the assumption that the dependent variable must be binary, means it can take only two values. For more than two categories SoftMax functions are used.
3. **Linearity relationship between independent variables and log odds**: The model assumes a linear relationship between the independent variables and the log odds of the dependent variable which means the predictors affect the log odds in a linear way.
4. **No outliers**: The dataset should not contain extreme outliers as they can distort the estimation of the logistic regression coefficients.
5. **Large sample size**: It requires a sufficiently large sample size to produce reliable and stable results.

**Understanding Sigmoid Function**

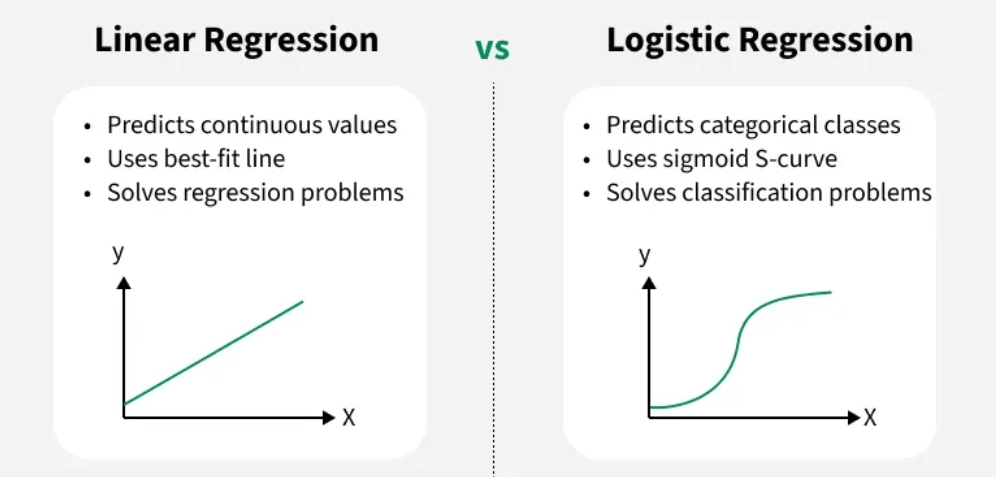
1. The sigmoid function is a important part of logistic regression which is used to convert the raw output of the model into a probability value between 0 and 1.

2. This function takes any real number and maps it into the range 0 to 1 forming an "S" shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.

3. In logistic regression, we use a threshold value usually 0.5 to decide the class label.

* If the sigmoid output is same or above the threshold, the input is classified as Class 1.
* If it is below the threshold, the input is classified as Class 0.

This approach helps to transform continuous input values into meaningful class predictions.



**How does Logistic Regression work?**

Logistic regression model transforms the [linear regression](https://www.geeksforgeeks.org/ml-linear-regression/) function continuous value output into categorical value output using a sigmoid function which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

we use the [sigmoid function](https://www.geeksforgeeks.org/derivative-of-the-sigmoid-function/) where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

**The Logistic Function (Sigmoid)**

The **sigmoid function** maps any real value to the range (0, 1):

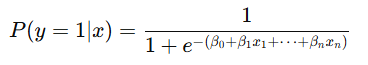
σ(z)=1/1+e−z

Where:

* z=β0+β1x1+β2x2+⋯+βnxn
* This z is the **linear combination** of input features.

The output σ(z) is the **probability** that the instance belongs to the **positive class (label = 1)**.

**Logistic Regression Equation**



You can **threshold** the probability (e.g., at 0.5) to make a prediction:

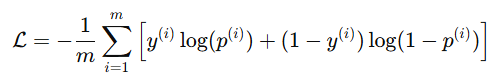
* If P≥0.5, predict class 1
* Else, predict class 0



As shown above the sigmoid function converts the continuous variable data into the probability i.e between 0 and 1.

**Loss Function: Binary Cross-Entropy**

To train logistic regression, we minimize the **binary cross-entropy loss**:



Where:

* y(i)is the true label
* p(i) is the predicted probability

This loss punishes wrong predictions more when the model is confident.

**Terminologies involved in Logistic Regression**

Here are some common terms involved in logistic regression:

1. **Independent Variables:**These are the input features or predictor variables used to make predictions about the dependent variable.
2. **Dependent Variable**: This is the target variable that we aim to predict. In logistic regression, the dependent variable is categorical.
3. **Logistic Function**: This function transforms the independent variables into a probability between 0 and 1 which represents the likelihood that the dependent variable is either 0 or 1.
4. **Odds**: This is the ratio of the probability of an event happening to the probability of it not happening. It differs from probability because probability is the ratio of occurrences to total possibilities.
5. **Log-Odds (Logit)**: The natural logarithm of the odds. In logistic regression, the log-odds are modeled as a linear combination of the independent variables and the intercept.
6. **Coefficient**: These are the parameters estimated by the logistic regression model which shows how strongly the independent variables affect the dependent variable.
7. **Intercept**: The constant term in the logistic regression model which represents the log-odds when all independent variables are equal to zero.

**How to Evaluate Logistic Regression Model?**

Evaluating the logistic regression model helps assess its performance and ensure it generalizes well to new, unseen data. The following metrics are commonly used:

1. **Accuracy:** Accuracy provides the proportion of correctly classified instances.  
   Accuracy=TruePositives+TrueNegatives/Total
2. **Precision:** Precision focuses on the accuracy of positive predictions.  
   Precision=True Positives/TruePositives+FalsePositives
3. **Recall (Sensitivity or True Positive Rate):** Recall measures the proportion of correctly predicted positive instances among all actual positive instances.  
   Recall=TruePositives/TruePositives+FalseNegatives

**F1 Score:**F1 score is the harmonic mean of precision and recall.  
F1Score=2∗Precision∗Recall/Precision+Recall

**Differences Between Linear and Logistic Regression**

Logistic regression and linear regression differ in their application and output. Here's a comparison:

| **Linear Regression** | **Logistic Regression** |
| --- | --- |
| Linear regression is used to predict the continuous dependent variable using a given set of independent variables. | Logistic regression is used to predict the categorical dependent variable using a given set of independent variables. |
| It is used for solving regression problem. | It is used for solving classification problems. |
| In this we predict the value of continuous variables | In this we predict values of categorical variables |
| In this we find best fit line. | In this we find S-Curve. |
| Least square estimation method is used for estimation of accuracy. | Maximum likelihood estimation method is used for Estimation of accuracy. |
| The output must be continuous value, such as price, age etc. | Output must be categorical value such as 0 or 1, Yes or no etc. |
| It required linear relationship between dependent and independent variables. | It not required linear relationship. |

**Advantages of Logistic Regression**

1. **Simple and Easy to Implement**
   * It’s intuitive and easy to implement compared to more complex models.
2. **Computationally Efficient**
   * Logistic regression is not resource-intensive and works well with smaller datasets.
3. **Works Well with Linearly Separable Classes**
   * Very effective when the relationship between the independent variables and the log-odds of the dependent variable is linear.
4. **Probability Estimates**
   * Provides class probabilities (between 0 and 1), which is useful in risk assessment and ranking.
5. **No Need for Feature Scaling**
   * Not heavily affected by feature scaling (though regularization might require it).
6. **Interpretable Model**
   * Coefficients are easy to understand, making it a good choice when model explainability is important.

**Disadvantages of Logistic Regression**

1. **Assumes Linearity in the Logit**
   * Assumes a linear relationship between independent variables and the log odds, which may not always hold.
2. **Not Suitable for Complex Relationships**
   * Can't capture non-linear relationships unless transformed features or polynomial terms are added.
3. **Sensitive to Irrelevant Features**
   * Including too many irrelevant or correlated features can reduce model performance.
4. **Limited to Binary or Multinomial Outputs**
   * Basic logistic regression is for binary classification; multinomial logistic regression is more complex.
5. **Prone to Overfitting**
   * Especially if the number of features is much larger than the number of observations (high-dimensional data).
6. **Cannot Handle Missing Values**
   * Logistic regression doesn’t natively handle missing data.

**Applications of Logistic Regression**

1. **Medical Diagnosis**
   * Predicting whether a patient has a disease (e.g., cancer, diabetes) based on symptoms or test results.
2. **Credit Scoring**
   * Assessing whether a person is likely to default on a loan (good/bad credit risk).
3. **Marketing and Customer Retention**
   * Predicting whether a customer will buy a product or churn.
4. **Political Forecasting**
   * Predicting election outcomes (win/loss) based on campaign data.
5. **Fraud Detection**
   * Determining whether a transaction is fraudulent or not.
6. **Social Science Research**
   * Analysing factors that influence binary outcomes like employment status, voting behavior, etc.

**Support Vector Machine (SVM)**

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks. While it can be applied to regression problems, SVM is best suited for classification tasks. The primary objective of the SVM algorithm is to identify the optimal hyperplane in an N-dimensional space that can effectively separate data points into different classes in the feature space. The algorithm ensures that the margin between the closest points of different classes, known as support vectors, is maximized.

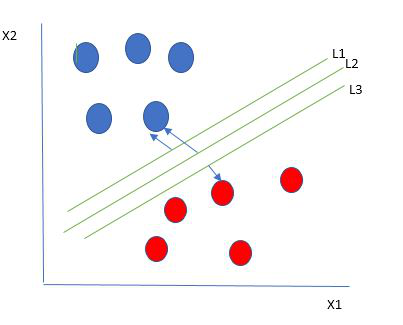
The dimension of the hyperplane depends on the number of features. For instance, if there are two input features, the hyperplane is simply a line, and if there are three input features, the hyperplane becomes a 2-D plane. As the number of features increases beyond three, the complexity of visualizing the hyperplane also increases.

Consider two independent variables, x1 and x2, and one dependent variable represented as either a blue circle or a red circle.

* In this scenario, the hyperplane is a line because we are working with two features (x1 and x2).
* There are multiple lines (or hyperplanes) that can separate the data points.
* The challenge is to determine the best hyperplane that maximizes the separation margin between the red and blue circles.

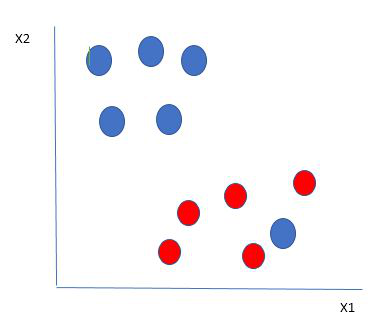
**How does Support Vector Machine Algorithm Work?**

One reasonable choice for the **best hyperplane** in a **Support Vector Machine (SVM)** is the one that maximizes the **separation margin** between the two classes. The **maximum-margin hyperplane**, also referred to as the **hard margin**, is selected based on maximizing the distance between the hyperplane and the nearest data point on each side.



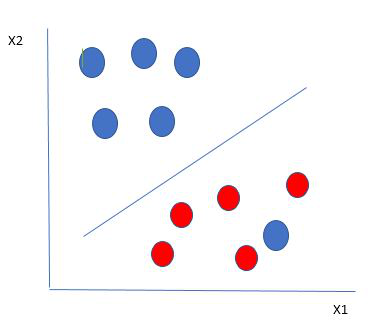
Multiple hyperplanes separate the data from two classes

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2. Let’s consider a scenario like shown below



Selecting hyperplane for data with outlier

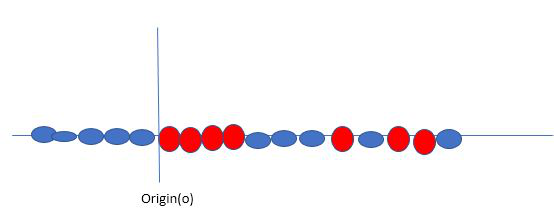
Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



Hyperplane which is the most optimized one

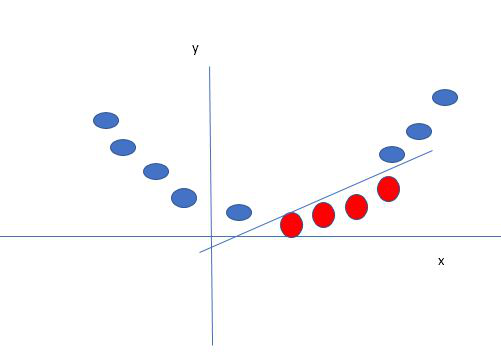
So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**. When there is a soft margin to the data set, the SVM tries to minimize (1/margin+∧(∑penalty)). Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?



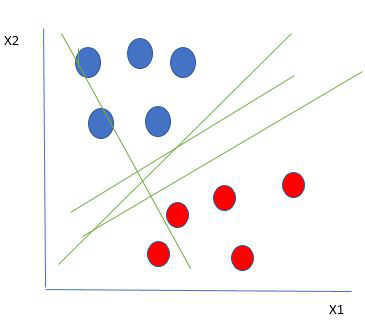
Original 1D dataset for classification

Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**. We call a point xi on the line and we create a new variable yi as a function of distance from origin o.so if we plot this we get something like as shown below



Mapping 1D data to 2D to become able to separate the two classes

In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.



**1. Mathematical Formulation**

**Mathematics Behind SVM**

### Goal:

Find a hyperplane:

wT x + b = 0

**Where:**

* w is the weight vector (normal to the hyperplane),
* b is the bias/intercept,
* x is the input feature vector.

**a. Objective Function:**

For a binary classification problem, the SVM aims to solve the following optimization problem:

minimize (1/2) ||w||²

Subject to:



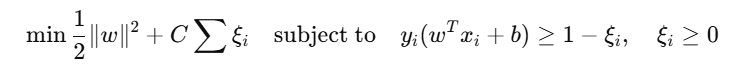
Where:

w: Weight vector (normal to the hyperplane)  
b: Bias term  
xi: Feature vector for the i-th data point  
yi: Class label (+1 or -1) for the i-th data point

**b. Soft Margin (For Non-Linearly Separable Data):**

To handle misclassifications, a slack variable ξi is introduced:

Subject to:



Here, C is a regularization parameter that balances maximizing the margin and minimizing classification errors.

**Types of Support Vector Machine**

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

* **Linear SVM:**Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
* **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space

**What is a Kernel in SVM?**

A kernel is a function that transforms input data into a higher-dimensional space so that a non-linear problem becomes linearly separable.

You don’t compute the transformation manually. Instead, kernels compute the dot product in high-dimensional space implicitly.

This is called the “Kernel Trick”.

**Why Kernels Are Needed**

* Real-world data is often not linearly separable.
* Linear SVM fails if there's no clear straight line or hyperplane to separate classes.
* Kernels solve this by mapping data into higher dimensions where separation is possible.

**Common SVM Kernel Types**

**1. Linear Kernel**

K (x, x′) =xT x′

* Simple dot product.
* Used when data is linearly separable.

Fast and works well with high-dimensional sparse data (e.g., text classification).

**2. Polynomial Kernel**

K (x, x′) = (xT x′ + c) d

* Maps input space into a polynomial feature space.
* Good for data with curved boundaries.

Parameters:

* c: trade-off between high-order and low-order terms (default = 1)
* d: degree of the polynomial

**3. RBF (Radial Basis Function) / Gaussian Kernel**

K (x, x′) = exp (−γ ∥ x − x′ ∥2)

* Maps points into infinite-dimensional space.
* Great for complex non-linear problems.

Parameter:

* γ: controls the influence of a single training example.
  + High γ: closer fit, might overfit
  + Low γ: smoother decision boundary

**4. Sigmoid Kernel**

K (x, x′) = tanh (α xT x′ + c)

* Inspired by neural networks.
* Rarely used in practice; unstable in many scenarios.

**Visual Intuition**

| Kernel | Use When... | Decision Boundary |
| --- | --- | --- |
| Linear | Data is linearly separable | Straight line |
| Polynomial | Data has curved boundaries | Curved polynomial |
| RBF/Gaussian | Data is complex and not linearly separable | Flexible/curvy |
| Sigmoid | Experimental or specific tasks | Similar to neural nets |

**Support Vector Machine Terminology**

* **Hyperplane**: The **hyperplane** is the decision boundary used to separate data points of different classes in a feature space. For **linear classification**, this is a linear equation represented as wx+b=0.
* **Support Vectors**: **Support vectors** are the closest data points to the hyperplane. These points are critical in determining the hyperplane and the margin in **Support Vector Machine (SVM)**.
* **Margin**: The **margin** refers to the distance between the **support vector** and the hyperplane. The primary goal of the SVM algorithm is to maximize this margin, as a wider margin typically results in better classification performance.
* **Kernel**: The **kernel** is a mathematical function used in SVM to map input data into a higher-dimensional feature space. This allows the SVM to find a hyperplane in cases where data points are not linearly separable in the original space. Common **kernel functions** include linear, polynomial, radial basis function (RBF), and sigmoid.
* **Hard Margin**: A **hard margin** refers to the maximum-margin hyperplane that perfectly separates the data points of different classes without any misclassifications.
* **Soft Margin**: When data contains **outliers** or is not perfectly separable, SVM uses the **soft margin** technique. This method introduces a **slack variable** for each data point to allow some misclassifications while balancing between maximizing the margin and minimizing violations.
* **Hinge Loss**: The **hinge loss** is a common loss function in SVMs. It penalizes misclassified points or margin violations and is often combined with a regularization term in the objective function.

**Advantages of Support Vector Machine (SVM)**

1. **High-Dimensional Performance**: SVM excels in high-dimensional spaces, making it suitable for **image classification** and **gene expression analysis**.
2. **Nonlinear Capability**: Utilizing **kernel functions** like **RBF** and **polynomial**, SVM effectively handles **nonlinear relationships**.
3. **Outlier Resilience**: The **soft margin** feature allows SVM to ignore outliers, enhancing robustness in **spam detection** and **anomaly detection**.
4. **Binary and Multiclass Support**: SVM is effective for both **binary classification** and **multiclass classification**, suitable for applications in **text classification**.
5. **Memory Efficiency**: SVM focuses on **support vectors**, making it memory efficient compared to other algorithms.

**Disadvantages of Support Vector Machine (SVM)**

1. **Slow Training**: SVM can be slow for large datasets, affecting performance in **SVM in data mining** tasks.
2. **Parameter Tuning Difficulty**: Selecting the right **kernel** and adjusting parameters like **C** requires careful tuning, impacting **SVM algorithms**.
3. **Noise Sensitivity**: SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.
4. **Limited Interpretability**: The complexity of the **hyperplane** in higher dimensions makes SVM less interpretable than other models.
5. **Feature Scaling Sensitivity**: Proper **feature scaling** is essential; otherwise, SVM models may perform poorly.

**Clustering**

Clustering is a type of unsupervised machine learning technique used to group a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). The goal of clustering is to discover natural groupings within a dataset, revealing hidden patterns and structures without prior knowledge of the group labels.

**Key concepts in clustering include:**

1. Similarity and Dissimilarity: Clustering algorithms rely on measures of similarity (or dissimilarity) between data points. Common measures include Euclidean distance, Manhattan distance, and cosine similarity.
2. Cluster Centroid: The central point of a cluster, often used in partitioning methods like K-means. It represents the mean position of all the points in the cluster.
3. Cluster Density: In density-based methods, clusters are defined as areas of higher density of data points, separated by areas of lower density.
4. Cluster Hierarchy: In hierarchical clustering, clusters are arranged in a tree-like structure, with smaller clusters nested within larger ones.
5. Cluster Membership: In traditional (hard) clustering, each data point belongs to exactly one cluster. In fuzzy clustering, each data point can belong to multiple clusters with varying degrees of membership.

**Applications of Clustering:**

* Market Segmentation: Grouping customers based on purchasing behavior.
* Image Segmentation: Dividing an image into regions for analysis.
* Document Clustering: Organizing documents into thematic groups.
* Anomaly Detection: Identifying outliers that do not fit into any cluster.
* Genomics: Grouping genes or proteins with similar expression patterns.

**Examples of Clustering Algorithms:**

* K-Means: Partitions data into K clusters by minimizing the sum of squared distances between data points and their cluster centroids.
* Hierarchical Clustering: Builds a tree of clusters using either a bottom-up (agglomerative) or top-down (divisive) approach.
* DBSCAN: Identifies clusters based on the density of data points, with the ability to detect noise points.
* Gaussian Mixture Models: Assumes data is generated from a mixture of several Gaussian distributions and uses the EM algorithm for clustering.

**Here are some common types of clustering algorithms:**

1. **Partitioning Methods**
   * K-Means Clustering: Divides the data into K clusters, where each cluster is represented by the mean of its points (centroid).
   * K-Medoids Clustering: Similar to K-means, but uses medoids (most centrally located points) instead of centroids.
2. **Hierarchical Methods**
   * Agglomerative Clustering: A bottom-up approach where each data point starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
   * Divisive Clustering: A top-down approach where all data points start in one cluster, and splits are performed recursively as one moves down the hierarchy.
3. **Density-Based Methods**
   * DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Forms clusters based on the density of points in a region, with noise points (outliers) identified as points in low-density regions.
   * OPTICS (Ordering Points To Identify the Clustering Structure): Similar to DBSCAN but can identify clusters with varying densities.

**K-Means Clustering**

**What is K-Means Clustering?**

K-Means Clustering is an unsupervised machine learning algorithm used to partition a dataset into a specified number of clusters (K). Each cluster is characterized by its centroid, which is the average of all the data points assigned to that cluster. The primary goal is to group similar data points together while keeping different clusters as distinct as possible.

**Objective of K-Means Clustering**

The objective is to partition the data into K clusters such that:

* Data points within each cluster are as similar to each other as possible.
* Data points in different clusters are as dissimilar as possible.

**How K-Means Clustering Works**

1. Initialization:
   * Randomly select K data points as the initial centroids (cluster centers).
2. Assignment Step:
   * Assign each data point to the nearest centroid based on the Euclidean distance.
3. Update Step:
   * Calculate the new centroid for each cluster by taking the average of all data points assigned to that cluster.
4. Iteration:
   * Repeat the Assignment and Update steps until the centroids no longer change or the changes are minimal.

**Distance Metric**

**Euclidean Distance:**

Hypotenuse2 = Base2 + Perpendicular2

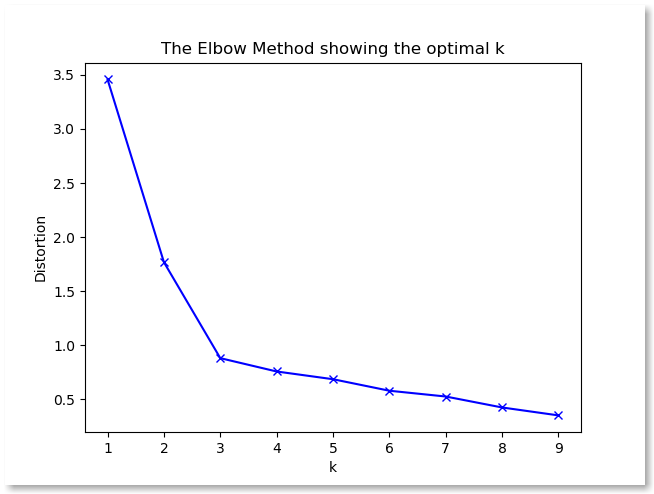
PQ2 = PR2 + QR2

Therefore, d2 = (x2 – x1)2 + (y2 – y1)2

**d =√ [(x2– x1)2 + (y2– y1)2]**

**Choosing the Number of Clusters**

* **Elbow Method:** Plot the sum of squared distances from each point to its assigned cluster centroid and look for an "elbow" point where the rate of decrease sharply slows.
* The best k value is Where the sudden decrease in the SSD



* **Silhouette Score:** Measures how similar a data point is to its own cluster compared to other clusters. A higher silhouette score indicates better-defined clusters.

**K-Means Algorithm Steps**

1. Initialize:
   * Choose K initial centroids randomly.
2. Repeat for a given number of iterations or until convergence:
   * Assign each data point to the nearest centroid.
   * Update centroids by calculating the mean of all data points assigned to each centroid.
3. End:
   * Final clusters are formed based on the final positions of the centroids.
4. Initialize K centroids randomly
5. Repeat until convergence or a fixed number of iterations:
6. For each data point: Assign it to the nearest centroid
7. For each centroid: Recalculate its position as the mean of the data points assigned to it

**Advantages of K-Means Clustering**

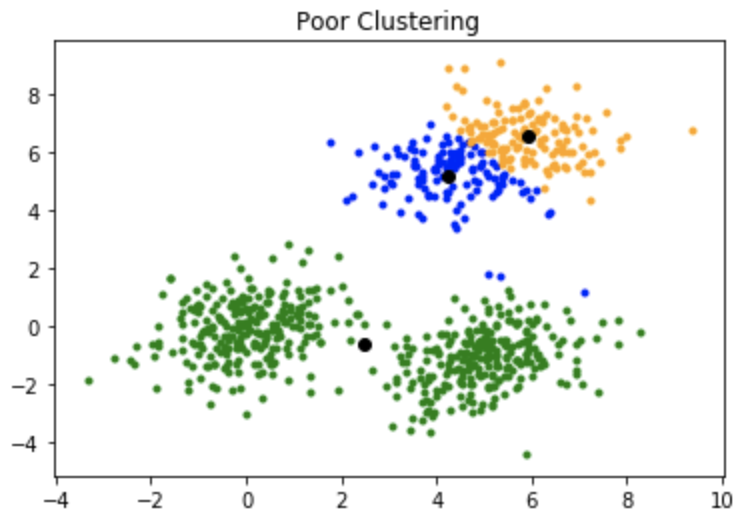
* Efficiency: K-Means is computationally efficient, making it suitable for large datasets.
* Simplicity: The algorithm is straightforward and easy to implement.

**Disadvantages of K-Means Clustering**

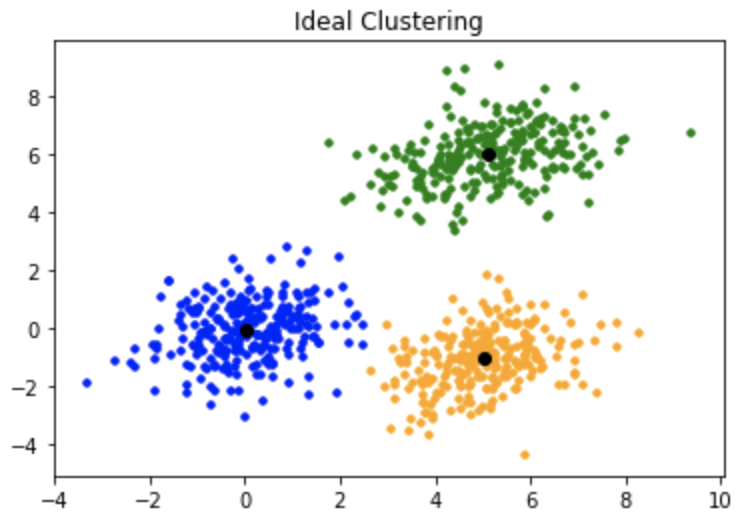
* Initial Centroid Sensitivity: Different initializations can lead to different results. Poor initialization can result in suboptimal clustering.
* Number of Clusters (K): The number of clusters K must be specified in advance, which may not always be obvious.
* Sensitivity to Noise: Outliers or noisy data points can affect the clustering results.
* Assumption of Spherical Clusters: K-Means assumes clusters are spherical and equally sized, which may not fit all datasets.

### ****K-means algorithm**** drawback****:****

One disadvantage of the K-means algorithm is that it is sensitive to the initialization of the centroids or the mean points. So, if a centroid is initialized to be a “far-off” point, it might just end up with no points associated with it, and at the same time, more than one cluster might end up linked with a single centroid. Similarly, more than one centroids might be initialized into the same cluster resulting in poor clustering. For example, consider the images shown below.   
A poor initialization of centroids resulted in poor clustering.



This is how the clustering should have been: 



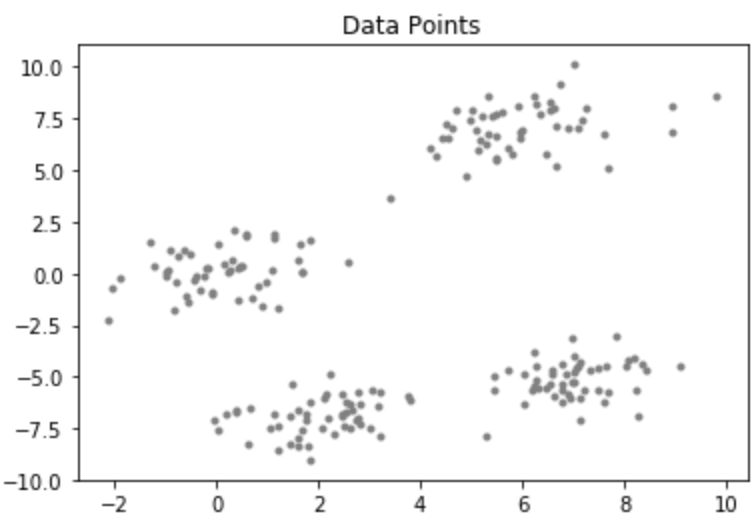
### ****K-mean++:****

To overcome the above-mentioned drawback, we use K-means++. This algorithm ensures a smarter initialization of the centroids and improves the quality of the clustering. Apart from initialization, the rest of the algorithm is the same as the standard K-means algorithm. That is K-means++ is the standard K-means algorithm coupled with a smarter initialization of the centroids.

#### **Initialization algorithm:**

The steps involved are:

1. Randomly select the first centroid from the data points.
2. For each data point compute its distance from the nearest, previously chosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its distance from the nearest, previously chosen centroid. (i.e. the point having maximum distance from the nearest centroid is most likely to be selected next as a centroid)
4. Repeat steps 2 and 3 until k centroids have been sampled



### Applications of K-Means Clustering

1. **Market Segmentation:**
   * Group customers based on purchasing behavior, demographics, or other features to target marketing strategies.
2. **Document Clustering:**
   * Organize a large collection of documents into clusters based on content similarity.
3. **Image Compression:**
   * Reduce the number of colors in an image by clustering pixels with similar colors.
4. **Anomaly Detection:**
   * Identify unusual data points by their distance from cluster centroids.
5. **Genetic Clustering:**
   * Group genes with similar expression patterns for gene function prediction or disease diagnosis.

* **Medical Imaging**: Segmenting different regions in medical images (e.g., MRI, CT scans) to identify and analyse different tissues or abnormalities.

**Principal Component Analysis (PCA) in Machine Learning**

Having too many features in data can cause problems like overfitting (good on training data but poor on new data), slower computation, and lower accuracy. This is called the [curse of dimensionality](https://www.geeksforgeeks.org/videos/curse-of-dimensionality-in-machine-learning/), where more features exponentially increase the data needed for reliable results.

Principal Component Analysis (PCA) is a **dimensionality reduction** technique used in machine learning and statistics. It transforms a dataset with many correlated features into a smaller set of uncorrelated features, called **principal components**, while retaining as much variance as possible.

**Why Use PCA?**

1. **Curse of Dimensionality** – When dealing with high-dimensional data, many algorithms suffer from inefficiency and overfitting.
2. **Feature Reduction** – PCA helps in reducing the number of features while preserving important information.
3. **Noise Reduction** – By removing less significant components, PCA reduces the effect of noise in the dataset.
4. **Data Visualization** – It enables visual representation of high-dimensional data in 2D or 3D.

**How PCA Works?**

PCA follows these steps:

1. **Standardize the Data**
   * Since PCA is affected by scale, we normalize the dataset so that all features have zero mean and unit variance.
2. **Compute the Covariance Matrix**
   * The covariance matrix captures the relationships between different features.
3. **Compute the Eigenvalues and Eigenvectors**
   * Eigenvectors determine the direction of the new feature space (principal components).
   * Eigenvalues represent the amount of variance carried by each principal component.
4. **Select the Top k Principal Components**
   * Choose the number of components (k) that explain most of the variance.
5. **Transform the Data**
   * The original data is projected onto the new feature space defined by the selected principal components.

**Mathematical Representation**

Let x be a dataset with n samples and p features.

1. **Compute Mean and Standardize the Data:**

X′=X−μ

where μ is the mean of each feature.

1. **Compute Covariance Matrix C:**



1. **Compute Eigenvalues λ and Eigenvectors V:**



The eigenvectors define the new axes, and eigenvalues represent the variance captured.

1. **Select k Principal Components**
   * Sort eigenvalues in descending order and select the top k.
2. **Transform the Data:**



where Vk contains the top k eigenvectors.

**Choosing the Right Number of Components**

* The explained variance ratio tells how much variance each principal component retains.
* A common method is to select the number of components where the cumulative explained variance is around **95%**.
* **Scree Plot**: A plot of eigenvalues helps in deciding the number of principal components.

**Advantages of PCA**

* Reduces computational cost.
* Removes multicollinearity among features.
* Enhances visualization in lower dimensions.

**Limitations of PCA**

* May lead to loss of interpretability.
* Assumes that principal components are linear combinations of original features.
* PCA is sensitive to outliers.

**Applications of PCA**

* Image Compression
* Face Recognition
* Feature Extraction in Machine Learning
* Noise Filtering

**Recommendation System**

The goal for a recommendation system is to extract information from data about the relationship existing between users and products. One of the common usage is to take the products that the user already likes and and answer the question "What other products can be recommended to the user?"

There are multiple ways of answering the above question. These are the three main types algorithms being used for recommendation systems, to try answering different aspects of recommendations:

**Recommendation System in Python**

** Purpose of Data Science and AI:**

* Both Data Science (DS) and Artificial Intelligence (AI) focus on generating business value through data and programming skills.

** Impact of Recommendation Systems:**

* Companies like Netflix, Amazon, and Uber Eats have revolutionized user experiences with personalized recommendations.
* Recommendation algorithms enhance user experience by providing tailored options based on individual interests and preferences.

 Role **of Python**:

* Python is a crucial tool for developing and implementing recommendation systems.
* It offers a flexible and robust environment for creating state-of-the-art recommendation algorithms.

** Applications of Recommendation Systems:**

* Websites and platforms use recommendation systems to predict user preferences and suggest relevant content.
* Common applications include social media, streaming services, and e-commerce platforms.

** Benefits:**

* Recommendation systems improve user experience, increase engagement, and drive business growth.

**Recommender System is of different types:**

* **Content-Based Recommendation:**It is supervised machine learning used to induce a classifier to discriminate between interesting and uninteresting items for the user.
* **Collaborative Filtering:**Collaborative Filtering recommends items based on similarity measures between users and/or items. The basic assumption behind the algorithm is that users with similar interests have common preferences.

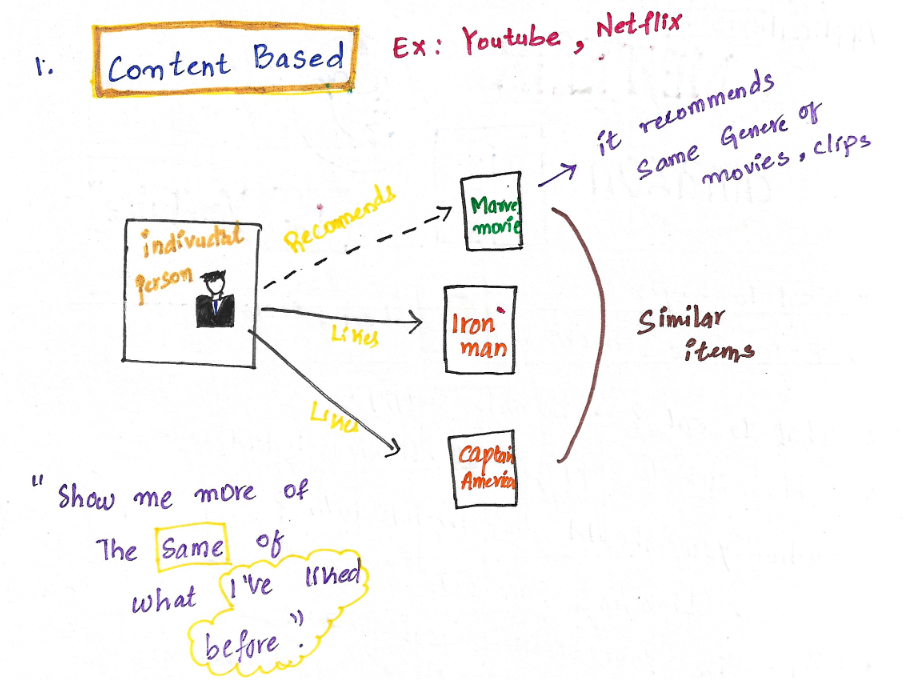
**Content Based Filtering**

♣   Used to find products with "similar" attributes. Example: if a person likes movies from action genere then recommend other books from action genere.

♣   The movie will have different attributes, such as, direction, cast, cinematography, story etc. A field specialist will rate all these attributes on a fixed scale.

♣   Based on similarity between these attributes, we can recommend movies to a new user once they have picked up a few movies that they liked.

♣   One major drawback of this method is getting a field expert to rate movies on their attributes. It is time consuming and inefficient.



**Content-Based Recommendation System**

Content-based systems recommend items to the customer similar to previously high-rated items by the customer. It uses the features and properties of the item. From these properties, it can calculate the similarity between the items.

In a content-based recommendation system, first, we need to create a profile for each item, which represents the properties of those items. The user profiles are inferred for a particular user. We use these user profiles to recommend the items to the users from the catalog.

**Item profile**

In a content-based recommendation system, we need to build a profile for each item, which contains the important properties of each item. For Example, If the movie is an item, then its actors, director, release year, and genre are its important properties, and for the document, the important property is the type of content and set of important words in it.

Let’s have a look at how to create an item profile. First, we need to perform the TF-IDF vectorizer, here TF (term frequency) of a word is the number of times it appears in a document and The IDF (inverse document frequency) of a word is the measure of how significant that term is in the whole corpus.

**TF-IDF Vectorizer**

* **Term Frequency(TF)**:

 Definition: Measures how often a term or word appears in a specific document within a text corpus.

 Purpose: Helps rank terms in a document based on their frequency, indicating their relative significance or value within that document.

 Use **Cases**: Useful in text analysis tasks like information retrieval, document classification, and sentiment analysis to identify important terms.

* **Inverse-document Frequency(IDF):**

 Definition: Evaluates the importance of a term within a set of documents by measuring its uniqueness or rarity across the entire corpus.

 Calculation: IDF is computed as the reciprocal of the fraction of documents containing the term, often logarithmized. Common terms have lower IDF values, while rare terms have higher values.

 Purpose: Helps highlight terms that are more unique to certain documents rather than being too common across all documents.

 Use Cases: Important for document ranking, categorization, and text mining to better represent and retrieve information from large datasets.

**Term Frequency-Inverse Document Frequency (TF-IDF):**

* Definition: A numerical statistic that combines TF and IDF to assess the significance of a term within a document relative to a corpus of documents.
* Calculation: TF-IDF is computed by multiplying TF and IDF scores, producing a weighted score that reflects both the term’s frequency in the document and its rarity in the corpus.
* Purpose: Enhances the representation of terms in text analysis, making it easier to identify important terms in documents relative to a large collection of documents.

**User profile**

The user profile is a vector that describes the user preference. During the creation of the user’s profile, we use a utility matrix that describes the relationship between user and item. From this information, the best estimate we can decide which item the user likes, is some aggregation of the profiles of those items.

**Advantages and Disadvantages**

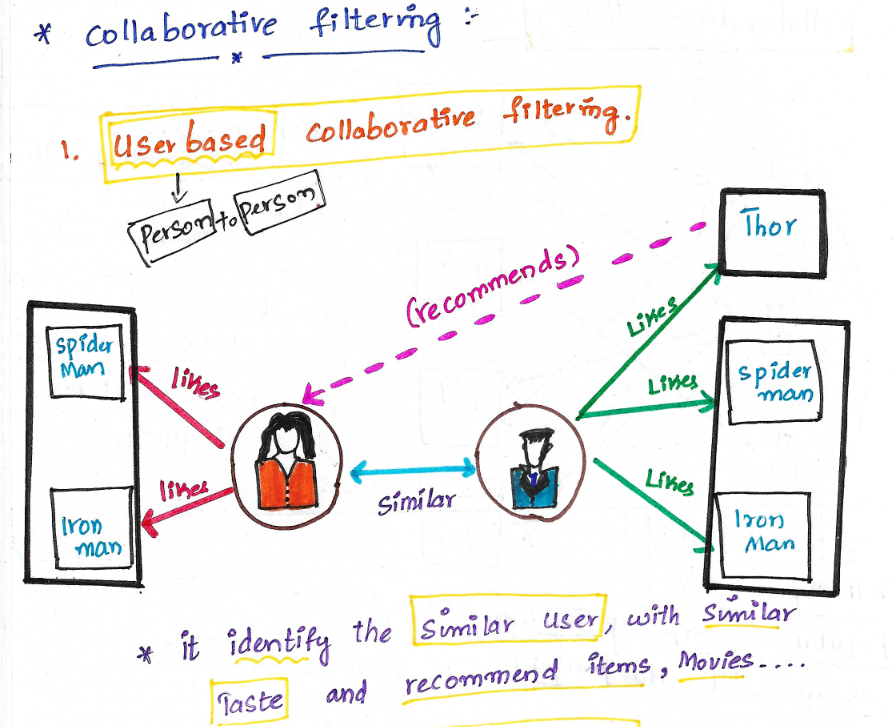
* **Advantages:**
  + No need for data on other users when applying to similar users.
  + Able to recommend to users with unique tastes.
  + Able to recommend new & popular items
  + Explanations for recommended items.
* **Disadvantages**:
  + Finding the appropriate feature is hard.
  + Doesn’t recommend items outside the user profile.

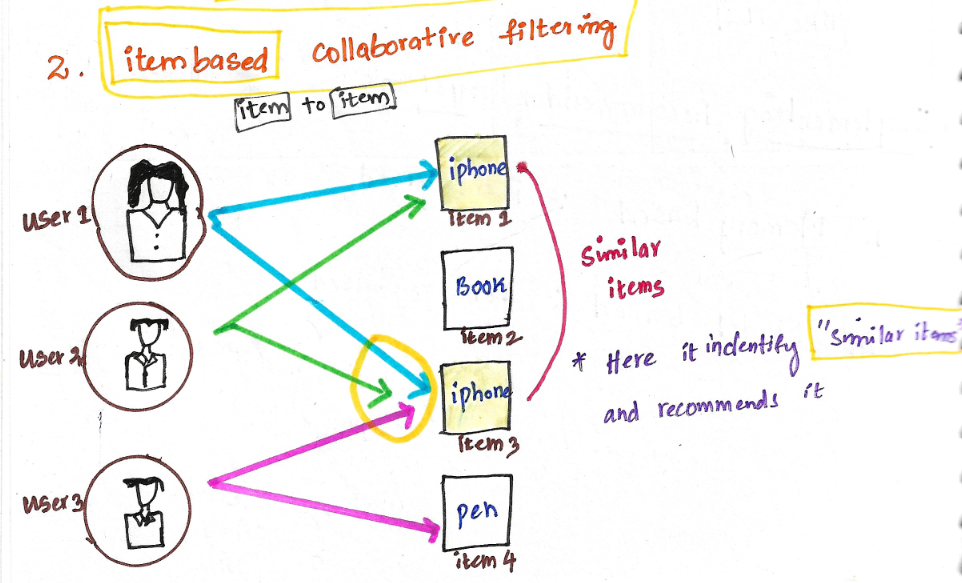
**Collaborative Filtering**

♣   This is somewhat indirect way of recommending, but is the most used method in the industry. Here we find products liked by "similar" users (they have the same interest as the active user) and recommend them to the active user.

♣   For an active user, who has rated some movies highly in the system; we find other users who have rated these movies in a similar manner (they might have rated other movies as well). These users have commen interests, so we can use other user's ratings as a guide for the likeliness of movie recommendation to the active user.

♣   Here the assumption is that the movies that "similar" users like are similar to each other. In this method also we are measuring similarity of products, however here the similarity is measured indirectly through similarity of users.

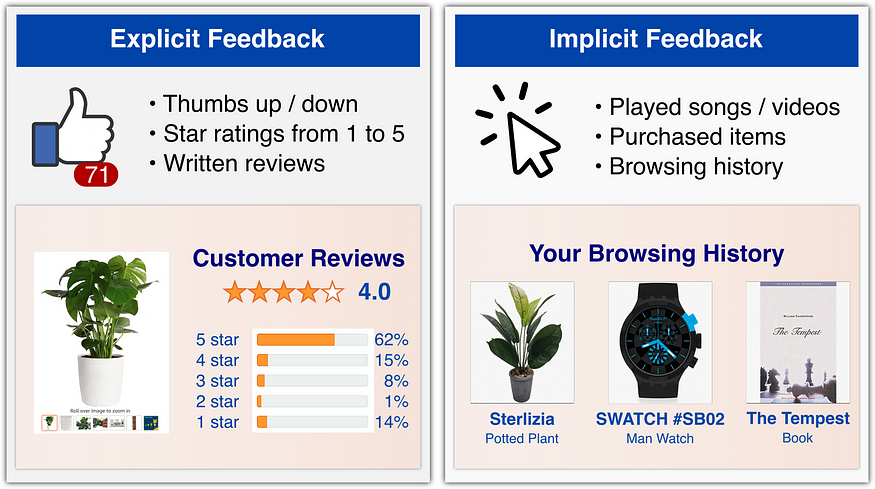




Collaborative filtering is based on the idea that similar people (based on the data) generally tend to like similar things. It predicts which item a user will like based on the item preferences of other similar users.

Collaborative filtering uses a user-item matrix to generate recommendations. This matrix contains the values that indicate a user’s preference towards a given item. These values can represent either explicit feedback (direct user ratings) or implicit feedback (indirect user behavior such as listening, purchasing, watching).

* **Explicit Feedback:**The amount of data that is collected from the users when they choose to do so. Many of the times, users choose not to provide data for the user. So, this data is scarce and sometimes costs money.  For example, ratings from the user.
* **Implicit Feedback:**In implicit feedback, we track user behavior to predict their preference.



**Example:**

* Consider a user x, we need to find another user whose rating are similar to x’s rating, and then we estimate x’s rating based on another user.

|  | **M\_1** | **M\_2** | **M\_3** | **M\_4** | **M\_5** | **M\_6** | **M\_7** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | 4 |  |  | 5 | 1 |  |  |
| **B** | 5 | 5 | 4 |  |  | 5 |  |
| **C** |  |  |  | 2 | 4 |  |  |
| **D** |  | 3 |  |  |  |  | 3 |

* Let’s create a matrix representing different user and movies:
* Consider two users x, y with rating vectors rx and ry. We need to decide a similarity matrix to calculate similarity b/w sim(x,y). THere are many methods to calculate similarity such as: Jaccard similarity, cosine similarity and pearson similarity. Here, we use centered cosine similarity/ pearson similarity, where we normalize the rating by subtracting the mean:

|  | **M\_1** | **M\_2** | **M\_3** | **M\_4** | **M\_5** | **M\_6** | **M\_7** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | 2/3 |  |  | 5/3 | -7/3 |  |  |
| **B** | 1/3 | 1/3 | -2/3 |  |  |  |  |
| **C** |  |  |  | -5/3 | 1/3 | 4/3 |  |
| **D** |  | 0 |  |  |  |  | 0 |

* Here, we can calculate similarity: For ex: sim(A,B) = cos(rA, rB) = 0.09 ; sim(A,C) = -0.56. sim(A,B) > sim(A,C).

**Rating Predictions**

Let rx be the vector of user x’s rating. Let N be the set of k similar users who also rated item i. Then we can calculate the prediction of user x and item i by using following formula:

**Advantages and Disadvantages**

* **Advantages:**
  + No need for the domain knowledge because embedding are learned automatically.
  + Capture inherent subtle characteristics.
* **Disadvantages**:
  + Cannot handle fresh items due to cold start problem.
  + Hard to add any new features that may improve quality of model

**Cosine Similarity**

The cosine similarity (or cosine distance when subtracted from 1) measures the cosine of the angle between two non-zero vectors in a multi-dimensional space. It is a metric often used in text analysis and recommendation systems to assess the similarity between two items based on their attributes.

**How Cosine Similarity Works**

1. **Vector Representation:**
   * Each item (e.g., a movie's genre) is represented as a vector in a multi-dimensional space.
   * For example, if you have a genre vector, each dimension represents a different genre, and the value at each dimension indicates the presence or absence (or degree) of that genre.
2. **Cosine of the Angle:**
   * The cosine similarity calculates the cosine of the angle between two vectors.
   * If the vectors are identical, the angle between them is 0 degrees, and the cosine similarity is 1.
   * If the vectors are orthogonal (completely dissimilar), the angle is 90 degrees, and the cosine similarity is 0.

