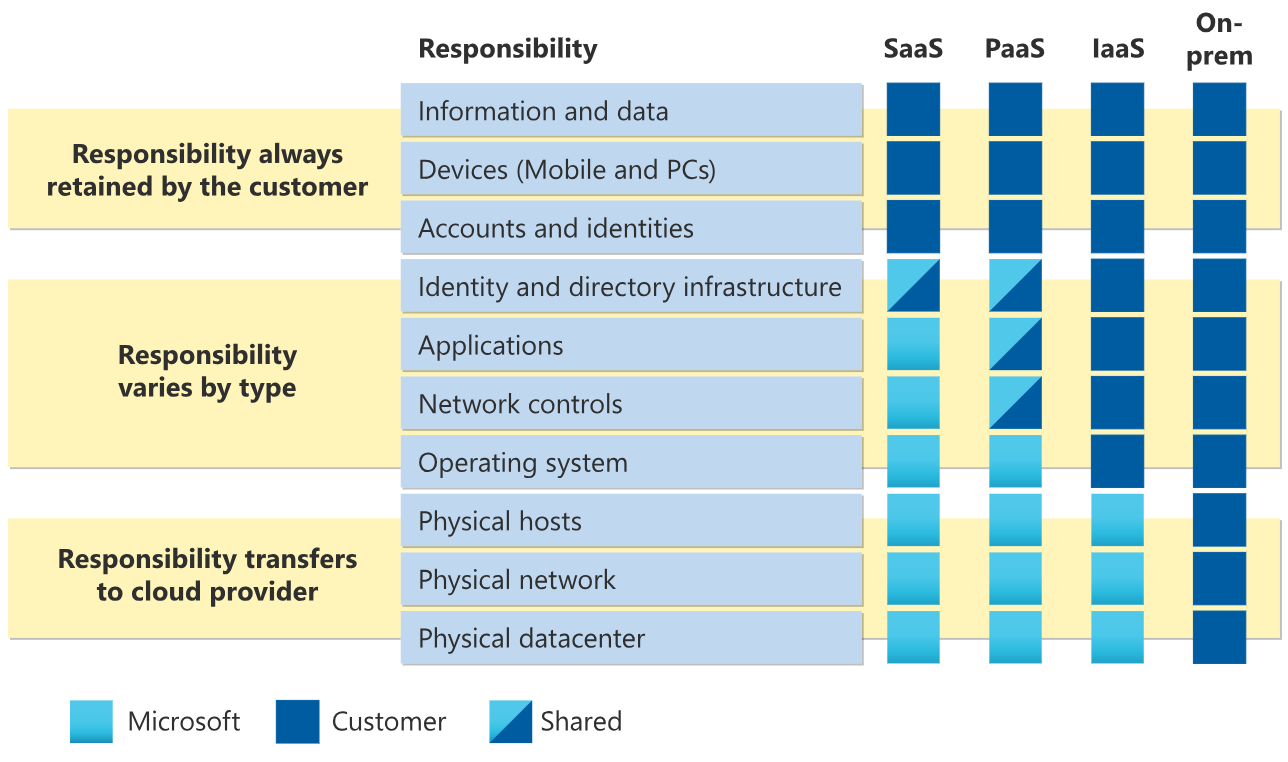
**What is cloud computing**

Cloud computing is the delivery of computing services over the internet. Computing services include common IT infrastructure such as virtual machines, storage, databases, and networking. Cloud services also expand the traditional IT offerings to include things like **Internet of Things (IoT), machine learning (ML), and artificial intelligence (AI).**



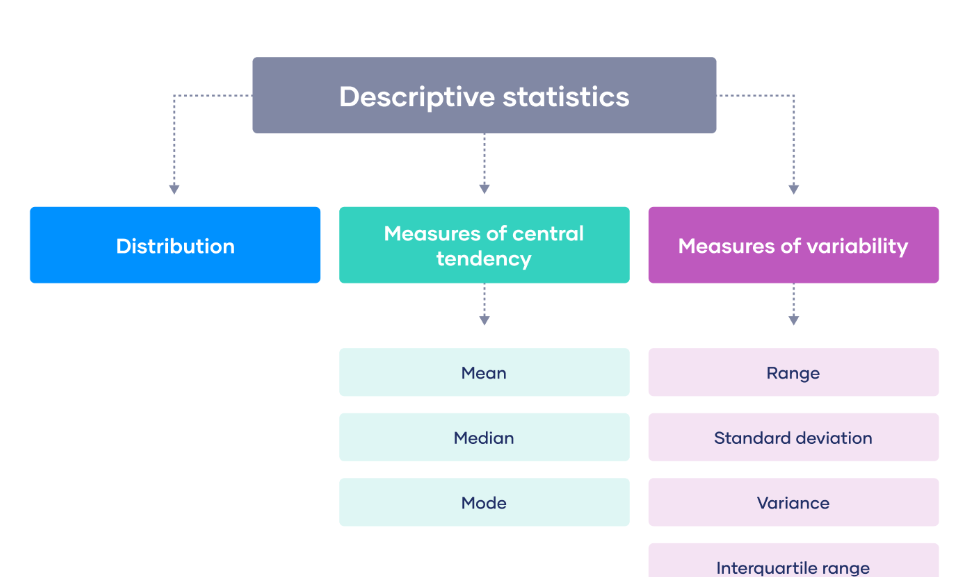
This consumption-based model has many benefits, including:

* No upfront costs.
* No need to purchase and manage costly infrastructure that users might not use to its fullest potential.
* The ability to pay for more resources when they're needed.
* The ability to stop paying for resources that are no longer needed.

# Descriptive Statistics

**Descriptive statistics** summarize and organize characteristics of a data set. A data set is a collection of responses or observations from a sample or entire population.

In quantitative research, after collecting data, the first step of statistical analysis is to describe characteristics of the responses, such as the average of one variable (e.g., age), or the relation between two variables (e.g., age and creativity).



Using descriptive statistics, you can report characteristics of your data:

* The **distribution**concerns the frequency of each value.
* The **central tendency** concerns the averages of the values.
* The **variability** concerns how spread out the values are.

# Inferential Statistics

While descriptive statistics summarize the characteristics of a data set, **inferential statistics** help you come to conclusions and make predictions based on your data.

When you have collected data from a sample, you can use inferential statistics to understand the larger population from which the sample is taken.

Inferential statistics have two main uses:

* making **estimates** about populations
* **testing hypotheses** to draw conclusions about populations

**What is EDA?**

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

EDA helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

**SQL- BASICS**

SQL is a standard language for accessing and manipulating databases.

## **What is SQL?**

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases

## **What Can SQL do?**

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

# SQL Syntax

Most of the actions you need to perform on a database are done with SQL statements.

SQL statements consists of keywords that are easy to understand.

## **Database Tables**

A database most often contains one or more tables.

## **Keep in Mind That...**

* SQL keywords are NOT case sensitive: select is the same as SELECT

## **Semicolon after SQL Statements?**

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

## **Some of The Most Important SQL Commands**

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

## **Types of SQL Commands**

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

# 

### **1. Data Definition Language (DDL)**

* DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
* All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

* CREATE
* ALTER
* DROP
* TRUNCATE

# **CREATE** It is used to create a new table in the database.

# CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE)

# **DROP:** It is used to delete both the structure and record stored in the table.

# DROP TABLE table\_name;

# **c. ALTER:** It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

1. ALTER TABLE STU\_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU\_DETAILS MODIFY (NAME VARCHAR2(20));

**d. TRUNCATE:** It is used to delete all the rows from the table and free the space containing the table.

**Syntax:**

1. TRUNCATE TABLE table\_name;

### **2. Data Manipulation Language**

* DML commands are used to modify the database. It is responsible for all form of changes in the database.
* The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
* INSERT
* UPDATE
* DELETE

**a. INSERT:** The INSERT statement is a SQL query. It is used to insert data into the row of a table.

**Syntax:**

1. INSERT INTO TABLE\_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, .... valueN);

OR

1. INSERT INTO TABLE\_NAME
2. VALUES (value1, value2, value3, .... valueN);
3. **UPDATE:** This command is used to update or modify the value of a column in the table.
4. UPDATE students
5. SET User\_Name = 'Sonoo'
6. WHERE Student\_Id = '3'

**c. DELETE:** It is used to remove one or more row from a table.

DELETE FROM table\_name [WHERE condition]

### **3. Data Control Language**

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

* Grant
* Revoke

**a. Grant:** It is used to give user access privileges to a database.

**Example**

1. GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

**b. Revoke:** It is used to take back permissions from the user.

**Example**

1. REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

### **4. Transaction Control Language**

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

* COMMIT
* ROLLBACK
* SAVEPOINT

1. **Commit:** Commit command is used to save all the transactions to the database.

DELETE FROM CUSTOMERS

WHERE AGE = 25;

COMMIT;

**b. Rollback:** Rollback command is used to undo transactions that have not already been saved to the database.

**Syntax:**

1. ROLLBACK;

**Example:**

DELETE FROM CUSTOMERS

WHERE AGE = 25;

ROLLBACK;

**c. SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

**Syntax:**

1. SAVEPOINT SAVEPOINT\_NAME;

### **5. Data Query Language**

DQL is used to fetch the data from the database.

It uses only one command:

* SELECT

**a. SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

**Syntax:**

1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;

# SQL SELECT Statement

# The SELECT statement is used to select data from a database.

## **Syntax**

# SELECT column1, column2, ... FROM table\_name;

# SELECT CustomerName, City FROM Customers;

# SQL SELECT DISTINCT Statement

# The SELECT DISTINCT statement is used to return only distinct (different) values.

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

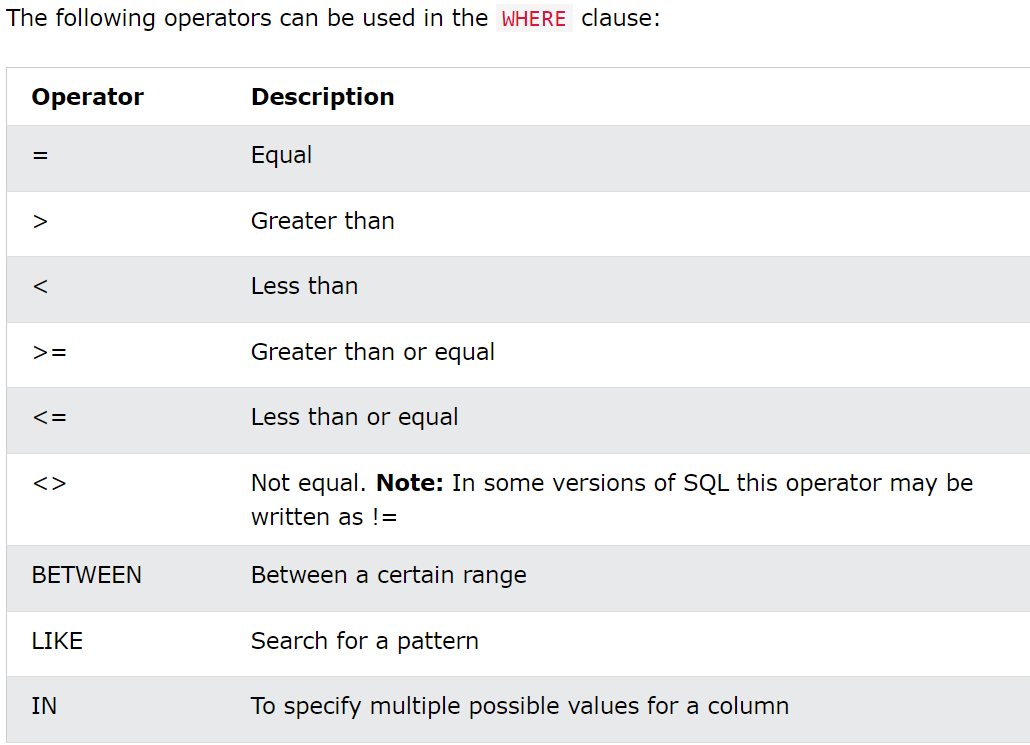
SELECT DISTINCT Country FROM Customers;

## **The SQL WHERE Clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;



SELECT \* FROM Employee

WHERE Salary BETWEEN 50,000 AND 60,000;

SELECT \* FROM Employee

WHERE City LIKE 's%';

SELECT \* FROM Customers

WHERE City IN ('Paris','London');

## **The SQL ORDER BY**

The ORDER BY keyword is used to sort the result-set in ascending or descending order

SELECT \* FROM Employee  
ORDER BY Salary

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

## **ORDER BY Several Columns**

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

SELECT \* FROM Customers  
ORDER BY Country, CustomerName;

## **Using Both ASC and DESC**

The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CustomerName" column:

# SQL AND Operator

The WHERE clause can contain one or many AND operators.

The AND operator is used to filter records based on more than one condition

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

SELECT \*  
FROM Customers  
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';

## **AND vs OR**

The AND operator displays a record if all the conditions are TRUE.

The OR operator displays a record if any of the conditions are TRUE.

## All Conditions Must Be True

SELECT \* FROM Customers  
WHERE Country = 'Germany'  
AND City = 'Berlin'  
AND PostalCode > 1200;

## **Combining AND and OR**

You can combine the AND and OR operators.

SELECT \* FROM Customers  
WHERE Country = 'Spain' AND (CustomerName LIKE 'G%' OR CustomerName LIKE 'R%');

## **The SQL OR Operator**

The WHERE clause can contain one or more OR operators.

The OR operator is used to filter records based on more than one condition

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

# SQL NOT Operator

The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

SELECT \* FROM Customers  
WHERE NOT Country = 'Spain';

## NOT LIKE

SELECT \* FROM Customers  
WHERE CustomerName NOT LIKE 'A%';

## NOT BETWEEN

SELECT \* FROM Customers  
WHERE CustomerID NOT BETWEEN 10 AND 60;

## NOT IN

SELECT \* FROM Customers  
WHERE City NOT IN ('Paris', 'London');

## NOT Greater Than

SELECT \* FROM Customers  
WHERE NOT CustomerID > 50;

## NOT Less Than

SELECT \* FROM Customers  
WHERE NOT CustomerId < 50;

# SQL INSERT INTO Statement

### **INSERT INTO Syntax**

It is possible to write the INSERT INTO statement in two ways:

* 1. Specify both the column names and the values to be inserted:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

* 1. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

## Insert Data Only in Specified Columns

INSERT INTO Customers (CustomerName, City, Country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

## Insert Multiple Rows

It is also possible to insert multiple rows in one statement.

To insert multiple rows of data, we use the same INSERT INTO statement, but with multiple values:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES  
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'),  
('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306', 'Norway'),  
('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA', 'UK');

# SQL NULL Values

## **What is a NULL Value?**

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

## **How to Test for NULL Values?**

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

### IS NULL Syntax

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

### IS NOT NULL Syntax

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

# SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## UPDATE Multiple Records

UPDATE Customers  
SET ContactName='Juan'  
WHERE Country='Mexico';

## **The SQL DELETE Statement**

The DELETE statement is used to delete existing records in a table.

## SQL DELETE Example

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

## Delete All Records

DELETE FROM table\_name;

DELETE FROM Customers;

## Delete a Table

To delete the table completely, use the DROP TABLE statement:

DROP TABLE Customers;

# SQL TOP, LIMIT, FETCH FIRST or ROWNUM Clause

## **The SQL SELECT TOP Clause**

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

SELECT TOP 3 \* FROM Customers;

## **LIMIT**

SELECT \* FROM Customers  
LIMIT 3;

## **FETCH FIRST**

SELECT \* FROM Customers  
FETCH FIRST 3 ROWS ONLY;

## **SQL TOP PERCENT Example**

SELECT TOP 50 PERCENT \* FROM Customers;

## **ADD a WHERE CLAUSE**

SELECT TOP 3 \* FROM Customers  
WHERE Country='Germany';

## **ADD the ORDER BY Keyword**

SELECT TOP 3 \* FROM Customers  
ORDER BY CustomerName DESC;

# SQL Aggregate Functions

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

The most commonly used SQL aggregate functions are:

* MIN() - returns the smallest value within the selected column
* MAX() - returns the largest value within the selected column
* COUNT() - returns the number of rows in a set
* SUM() - returns the total sum of a numerical column
* AVG() - returns the average value of a numerical column

**SQL MIN() and MAX() Functions**

SELECT MIN(Price)  
FROM Products;

SELECT MAX(Price)  
FROM Products;

## **Set Column Name (Alias)**

When you use MIN() or MAX(), the returned column will not have a descriptive name. To give the column a descriptive name, use the AS keyword:

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

## **Use MIN() with GROUP BY**

Here we use the MIN() function and the GROUP BY clause, to return the smallest price for each category in the Products table:

SELECT MIN(Price) AS SmallestPrice, CategoryID  
FROM Products  
GROUP BY CategoryID;

# SQL GROUP BY Statement

## **The SQL GROUP BY Statement**

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

### **GROUP BY Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;

# SQL HAVING Clause

# The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

### **HAVING Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;

# SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5 ORDER BY COUNT(CustomerID) DESC;

## **The SQL COUNT() Function**

The COUNT() function returns the number of rows that matches a specified criterion.

SELECT COUNT(\*)  
FROM Products;

## **Syntax**

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

## **Specify Column**

You can specify a column name instead of the asterix symbol (\*).

If you specify a column name instead of (\*), NULL values will not be counted.

SELECT COUNT(ProductName)  
FROM Products;

## **Add a WHERE Clause**

SELECT COUNT(ProductID)  
FROM Products  
WHERE Price > 20;

## **Ignore Duplicates**

You can ignore duplicates by using the DISTINCT keyword in the COUNT() function.

If DISTINCT is specified, rows with the same value for the specified column will be counted as one.

SELECT COUNT(DISTINCT Price)  
FROM Products;

## **Use an Alias**

Give the counted column a name by using the AS keyword.

SELECT COUNT(\*) AS [Number of records]  
FROM Products;

## **Use COUNT() with GROUP BY**

Here we use the COUNT() function and the GROUP BY clause, to return the number of records for each category in the Products table:

SELECT COUNT(\*) AS [Number of records], CategoryID  
FROM Products  
GROUP BY CategoryID

## **The SQL SUM() Function**

The SUM() function returns the total sum of a numeric column.

SELECT SUM(Quantity)  
FROM OrderDetails;

## **Syntax**

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

## **Add a WHERE Clause**

SELECT SUM(Quantity)  
FROM OrderDetails  
WHERE ProductId = 11;

## Use an Alias

Give the summarized column a name by using the AS keyword.

SELECT SUM(Quantity) AS total  
FROM OrderDetails;

## **Use SUM() with GROUP BY**

SELECT OrderID, SUM(Quantity) AS [Total Quantity]  
FROM OrderDetails  
GROUP BY OrderID;

## **SUM() With an Expression**

The parameter inside the SUM() function can also be an expression.

SELECT SUM(Quantity \* 10)  
FROM OrderDetails;

# SQL AVG() Function

# SELECT AVG(Price) FROM Products;

## **Syntax**

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

## **Add a WHERE Clause**

SELECT AVG(Price)  
FROM Products  
WHERE CategoryID = 1;

## **Use an Alias**

SELECT AVG(Price) AS [average price]  
FROM Products;

# SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign % represents zero, one, or multiple characters
* The underscore sign \_ represents one, single character

# SELECT \* FROM Customers WHERE CustomerName LIKE 'a%';

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

## **The \_ Wildcard**

The \_ wildcard represents a single character.

It can be any character or number, but each \_ represents one, and only one, character

SELECT \* FROM Customers  
WHERE city LIKE 'S\_TN\_';

## **The % Wildcard**

The % wildcard represents any number of characters, even zero characters.

SELECT \* FROM Customers  
WHERE city LIKE '%L%';

## **Starts With**

To return records that starts with a specific letter or phrase, add the % at the end of the letter or phrase.

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'La%';

## **Ends With**

To return records that ends with a specific letter or phrase, add the % at the beginning of the letter or phrase.

SELECT \* FROM Customers  
WHERE CustomerName LIKE '%a';

## **Contains**

To return records that contains a specific letter or phrase, add the % both before and after the letter or phrase.

SELECT \* FROM Customers  
WHERE CustomerName LIKE '%or%'

## **Combine Wildcards**

Any wildcard, like % and \_ , can be used in combination with other wildcards.

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a\_\_%'

## **Without Wildcard**

If no wildcard is specified, the phrase has to have an exact match to return a result.

SELECT \* FROM Customers  
WHERE Country LIKE 'Spain';

**What is EDA?**

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

EDA helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

# Statistics - Populations and Samples

**Population**: Everything in the group that we want to learn about.

**Sample**: A part of the population.

For good statistical analysis, sample is **representative** of the population.

The sample is used to make conclusions about the whole population. If the sample is not similar enough to the whole population, the conclusions could be useless.

## **Parameters and Statistics**

**Parameter**: A number that describes something about the whole **population**.

**Sample statistic**: A number that describes something about the **sample**.

The parameters are the key things we want to learn about. The parameters are usually unknown.

Sample statistics gives us **estimates** for parameters.

There will always be some **uncertainty** about how accurate estimates are. More certainty gives us more useful knowledge.

# Inferential Statistics

While descriptive statistics summarize the characteristics of a data set, **inferential statistics** help you come to conclusions and make predictions based on your data.

When you have collected data from a sample, you can use inferential statistics to understand the larger population from which the sample is taken.

Inferential statistics have two main uses:

* making **estimates** about populations
* **testing hypotheses** to draw conclusions about populations

## **Statistical Inference**

Using data analysis and statistics to make conclusions about a [population](https://www.w3schools.com/statistics/statistics_populations_and_samples.php) is called statistical inference.

The main types of statistical inference are:

* Estimation
* Hypothesis testing

## **Estimation**

Statistics from a sample are used to estimate population [parameters](https://www.w3schools.com/statistics/statistics_parameters_and_statistics.php).

The most likely value is called a **point estimate**.

There is **always** uncertainty when estimating.

The uncertainty is often expressed as **confidence intervals** defined by a likely lowest and highest value for the parameter.

## **Hypothesis Testing**

**Hypothesis testing** is a method to check if a claim about a population is true. More precisely, it checks how likely it is that a hypothesis is true is based on the sample data.

There are different types of hypothesis testing.

The steps of the test depends on:

* Type of data (categorical or numerical)
* If you are looking at:
  + A single group
  + Comparing one group to another
  + Comparing the same group before and after a change

## **Probability Distributions**

Statistical inference methods rely on probability calculation and probability distributions.

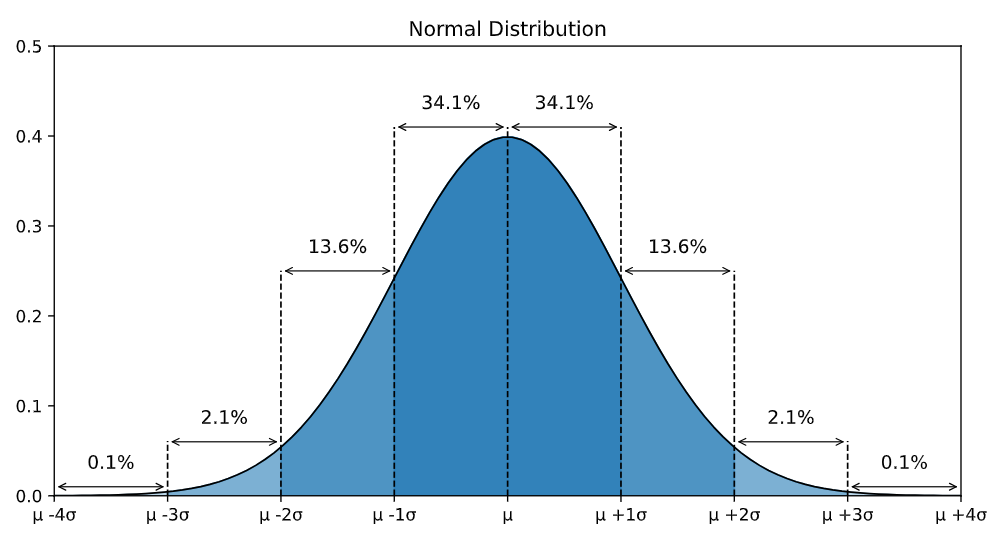
## **Normal Distribution**

The normal distribution is described by the mean (�) and the standard deviation (�).

The normal distribution is often referred to as a 'bell curve' because of it's shape:

* Most of the values are around the center (�)
* The median and mean are equal
* It has only one mode
* It is symmetric, meaning it decreases the same amount on the left and the right of the center

The area under the curve of the normal distribution represents probabilities for the data.



* Roughly 68.3% of the data is within 1 standard deviation of the average (from μ-1σ to μ+1σ)
* Roughly 95.5% of the data is within 2 standard deviations of the average (from μ-2σ to μ+2σ)
* Roughly 99.7% of the data is within 3 standard deviations of the average (from μ-3σ to μ+3σ)

The standard deviation describes how spread out the normal distribution is

## **Probability Distributions**

Probability distributions are functions that calculates the probabilities of the outcomes of random variables.

Typical examples of random variables are coin tosses and dice rolls.

**Machine Learning**

Machine Learning is making the computer learn from studying data and statistics.

Machine Learning is a step into the direction of artificial intelligence (AI).

Machine Learning is a program that analyses data and learns to predict the outcome.

## **Data Set**

A data set is any collection of data. It can be anything from an array to a complete database.

## **Data Types**

We can split the data types into three main categories:

* **Numerical**
* **Categorical**
* **Ordinal**

**Numerical** data are numbers, and can be split into two numerical categories:

* **Discrete Data**  
  - counted data that are limited to integers. Example: The number of cars passing by, No of coins or Rupees

Discrete data is data that only occurs in certain intervals. Essentially it is data that is not continuous.

* **Continuous Data**  
  - measured data that can be any number. Example: The price of an item, or the size of an item,Height,Weight,Distance,Volume

**Categorical** data are **values** that **cannot be measured up against each other**. Example: a color value, or any yes/no values,gender,group

**Ordinal** data are like **categorical data**, but can be **measured up against each other**

**Interval,age category,education level**

Frequency distribution,mode,median,range

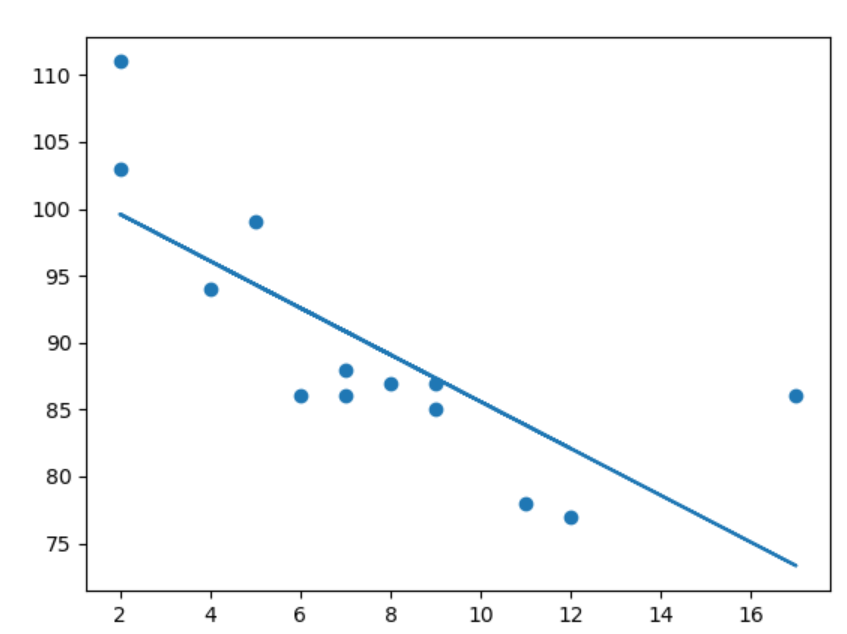
## **Regression**

The term regression is used when you try to find the relationship between variables.

## **Linear Regression**

Linear regression uses the relationship between the data-points to draw a straight line through all them.

This line can be used to predict future values.



import matplotlib.pyplot as plt  
from scipy import stats  
  
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]  
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
slope, intercept, r, p, std\_err = stats.linregress(x, y)  
  
def myfunc(x):  
  return slope \* x + intercept  
  
mymodel = list(map(myfunc, x))  
  
plt.scatter(x, y)  
plt.plot(x, mymodel)  
plt.show()

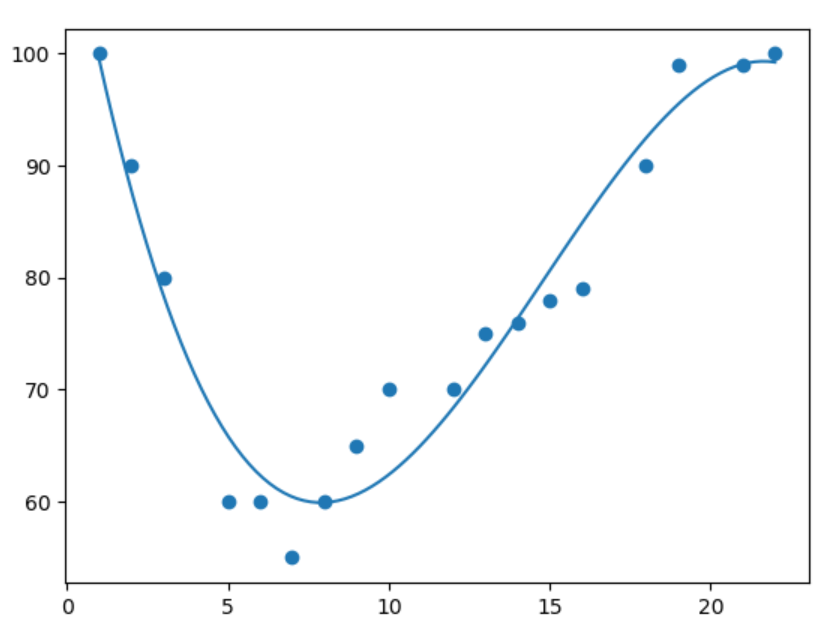
speed = myfunc(9)

print(speed)

## **Polynomial Regression**

If your data points clearly will not fit a linear regression (a straight line through all data points), it might be ideal for polynomial regression.

Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.



import numpy  
import matplotlib.pyplot as plt  
  
x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]  
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]  
  
mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))  
  
myline = numpy.linspace(1, 22, 100)  
  
plt.scatter(x, y)  
plt.plot(myline, mymodel(myline))  
plt.show()

speed = mymodel(15)  
print(speed)

## **Multiple Regression**

Multiple regression is like linear regression, but with more than one independent value, meaning that we try to predict a value based on **two or more** variables.

# Machine Learning - Train/Test

## **Evaluate Your Model**

In Machine Learning we create models to predict the outcome of certain events

## **What is Train/Test**

Train/Test is a method to measure the accuracy of your model.

It is called Train/Test because you split the data set into two sets: a training set and a testing set.

80% for training, and 20% for testing.

You train the model using the training set.

You test the model using the testing set.

Train the model means create the model.

Test the model means test the accuracy of the model.

# Machine Learning - Decision Tree

## **Decision Tree**

A Decision Tree is a Flow Chart, and can help you make decisions based on previous experience.

## **What is a confusion matrix?**

## **What is a confusion matrix?**

It is a table that is used in classification problems to assess where errors in the model were made.

The rows represent the actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.

## **Creating a Confusion Matrix**

Confusion matrixes can be created by predictions made from a logistic regression.

import matplotlib.pyplot as plt  
import numpy  
from sklearn import metrics  
  
actual = numpy.random.binomial(1,.9,size = 1000)  
predicted = numpy.random.binomial(1,.9,size = 1000)  
  
confusion\_matrix = metrics.confusion\_matrix(actual, predicted)  
  
cm\_display = metrics.ConfusionMatrixDisplay(confusion\_matrix = confusion\_matrix, display\_labels = [False, True])  
  
cm\_display.plot()  
plt.show()



* **True positives (TP):** occur when the model accurately predicts a positive data point.
* **True negatives (TN)**: occur when the model accurately predicts a negative data point.
* **False positives (FP)**: occur when the model predicts a positive data point incorrectly.
* **False** **negatives (FN)**: occur when the model mispredicts a negative data point.

The Confusion Matrix created has four different quadrants:

False Negative (Top-Left Quadrant)  
False Positive (Top-Right Quadrant)  
True Negative (Bottom-Left Quadrant)  
True Positive (Bottom-Right Quadrant)

True means that the values were accurately predicted, False means that there was an error or wrong prediction.

Now that we have made a Confusion Matrix, we can calculate different measures to quantify the quality of the model.

## **Created Metrics**

The matrix provides us with many useful metrics that help us to evaluate out classification model.

The different measures include: Accuracy, Precision, Sensitivity (Recall), Specificity, and the F-score, explained below.

## **Accuracy**

Accuracy measures how often the model is correct.

### **How to Calculate**

(True Positive + True Negative) / Total Predictions

Accuracy = metrics.accuracy\_score(actual, predicted)

## **Precision**

Of the positives predicted, what percentage is truly positive?

### **How to Calculate**

True Positive / (True Positive + False Positive)

Precision does not evaluate the correctly predicted negative cases:

Precision = metrics.precision\_score(actual, predicted)

## **Sensitivity (Recall)**

Of all the positive cases, what percentage are predicted positive?

Sensitivity (sometimes called Recall) measures how good the model is at predicting positives.

This means it looks at true positives and false negatives (which are positives that have been incorrectly predicted as negative).

### **How to Calculate**

True Positive / (True Positive + False Negative)

Sensitivity is good at understanding how well the model predicts something is positive:

Sensitivity\_recall = metrics.recall\_score(actual, predicted)

## **Specificity**

How well the model is at prediciting negative results?

Specificity is similar to sensitivity, but looks at it from the persepctive of negative results.

### **How to Calculate**

True Negative / (True Negative + False Positive)

Since it is just the opposite of Recall, we use the recall\_score function, taking the opposite position label:

Specificity = metrics.recall\_score(actual, predicted, pos\_label=0)

## **F-score**

F-score is the "harmonic mean" of precision and sensitivity.

It considers both false positive and false negative cases and is good for imbalanced datasets.

### **How to Calculate**

2 \* ((Precision \* Sensitivity) / (Precision + Sensitivity))

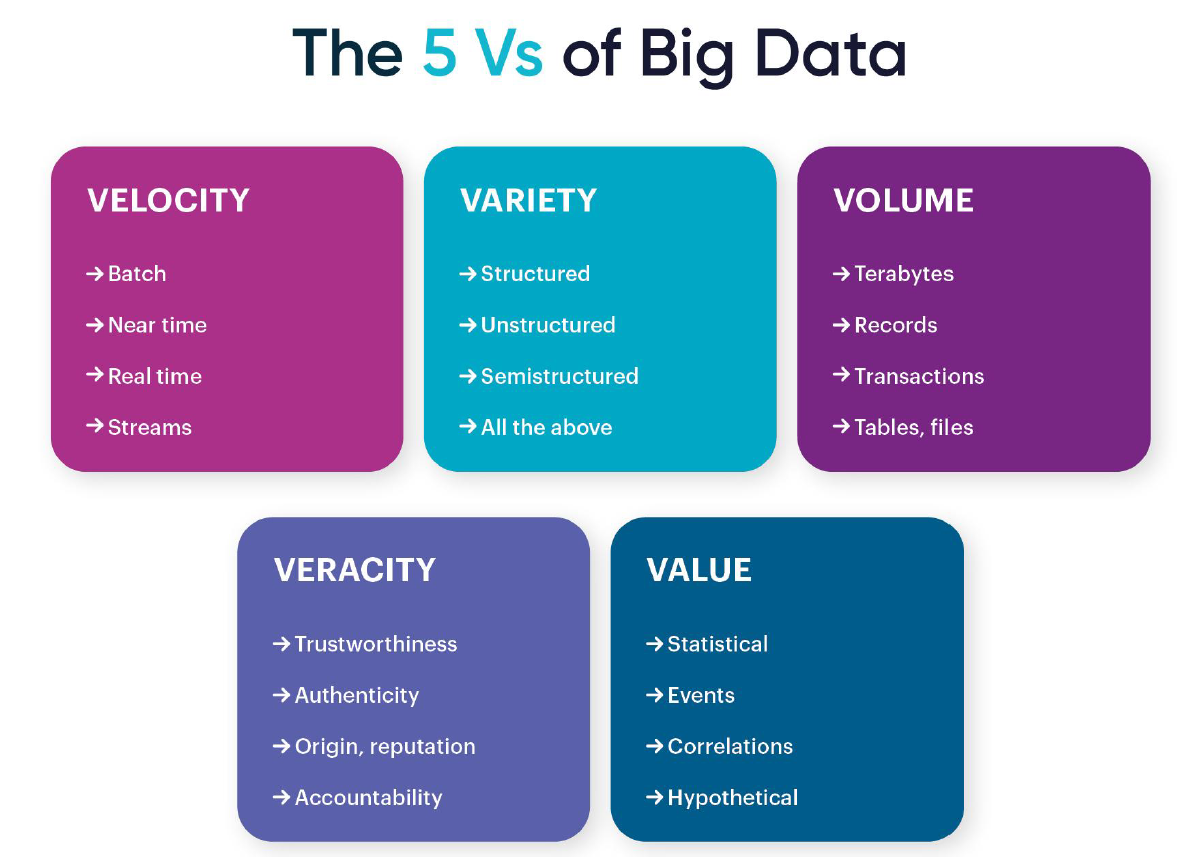
This score does not take into consideration the True Negative values:

F1\_score = metrics.f1\_score(actual, predicted)

print({"Accuracy":Accuracy,"Precision":Precision,"Sensitivity\_recall":Sensitivity\_recall,"Specificity":Specificity,"F1\_score":F1\_score})

**What is BigData?**

* Big Data is a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently



**What is Cluster?**

A cluster, in the context of computing, refers to a group of computers or servers that work together and can

be viewed as a single system. These computers, known as nodes, interact with each other to accomplish a

common goal. This setup is used to improve performance and availability over that provided by a single

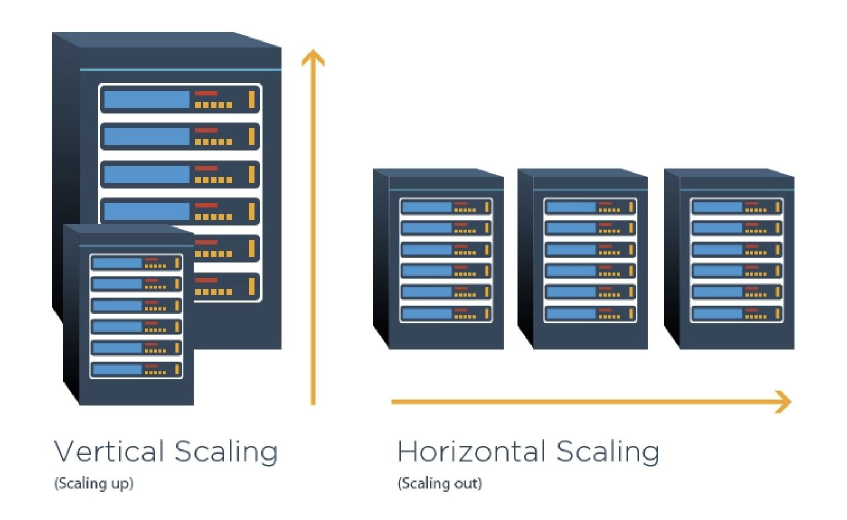
computer, while typically being much more cost-effective and scalable than a single computer of comparable speed or availability.

**Vertical Scaling vs Horizontal Scaling**

● **Vertical** Scaling, also known as scaling up, involves increasing the capacity of a single server, such as

using a more powerful CPU, adding more RAM, or increasing disk space.

● **Horizontal** Scaling, also known as scaling out, involves adding more servers to a system and distributing the load across multiple servers.



**Hadoop**

Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity

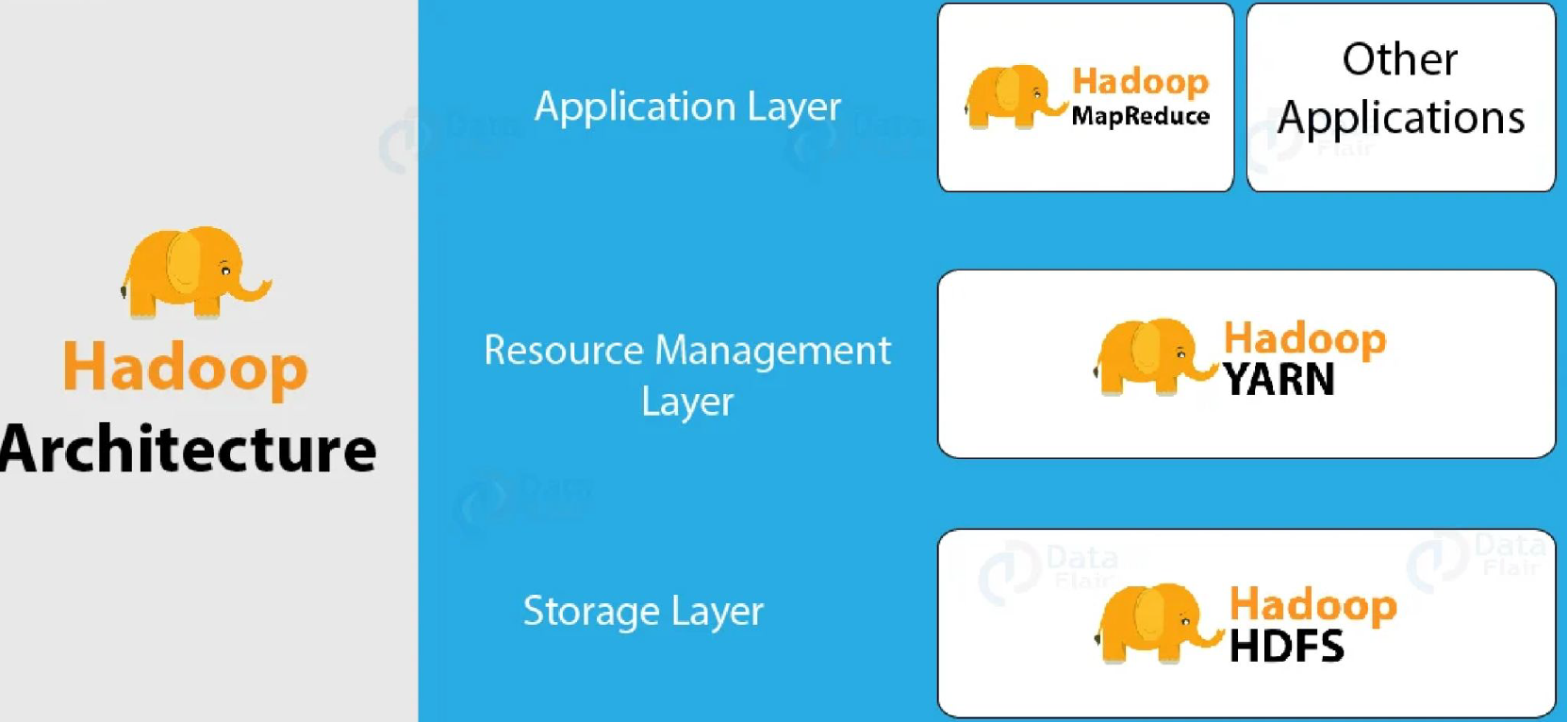
hardware. Essentially, it accomplishes two tasks: massive data storage and faster processing.

It was developed by the Apache Software Foundation and is based on two main components:

● **Hadoop Distributed File System (HDFS):** This is the storage component of Hadoop, designed to hold large amounts of data,potentially in the range of petabytes or even exabytes. The data is distributed across multiple nodes in the cluster, providing high availability and fault tolerance.

● **Map-Reduce:** This is the processing component of Hadoop, which provides a software framework for writing applications that process large amounts of data in parallel. MapReduce operations are divided into two stages: the **Map stage**, which sorts and filters the data, and the **Reduce stage**, which summarizes the data.

● **Yet Another Resource Negotiator (YARN):** This is the resource management layer in Hadoop. Introduced in Hadoop 2.0, YARN decouples the programming model from the resource management infrastructure, and it oversees and manages the compute resources in the clusters.



**Properties of Hadoop**

● **Scalability**: Can store and distribute large data sets across many servers.

● **Cost-effectiveness:** Designed to run on inexpensive, commodity hardware.

● **Flexibility**: Can handle any type of data, structured or unstructured.

● **Fault Tolerance**: Data is automatically replicated to other nodes in the cluster.

● **Data Locality**: Processes data on or near the node where it's stored, reducing network I/O.

● **Simplicity**: Provides a simple programming model (MapReduce) for processing data.

● **Open-source**: Freely available to use and modify with a large community of contributors.

**HDFS**

HDFS, the Hadoop Distributed File System, follows a master-slave architecture. The two main components of HDFS are

the NameNode (the master) and the DataNode (the slave).

**NameNode**: The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file

system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself.

● The NameNode manages the file system namespace. It maintains the file system tree and the metadata for all the

files and directories in the tree.

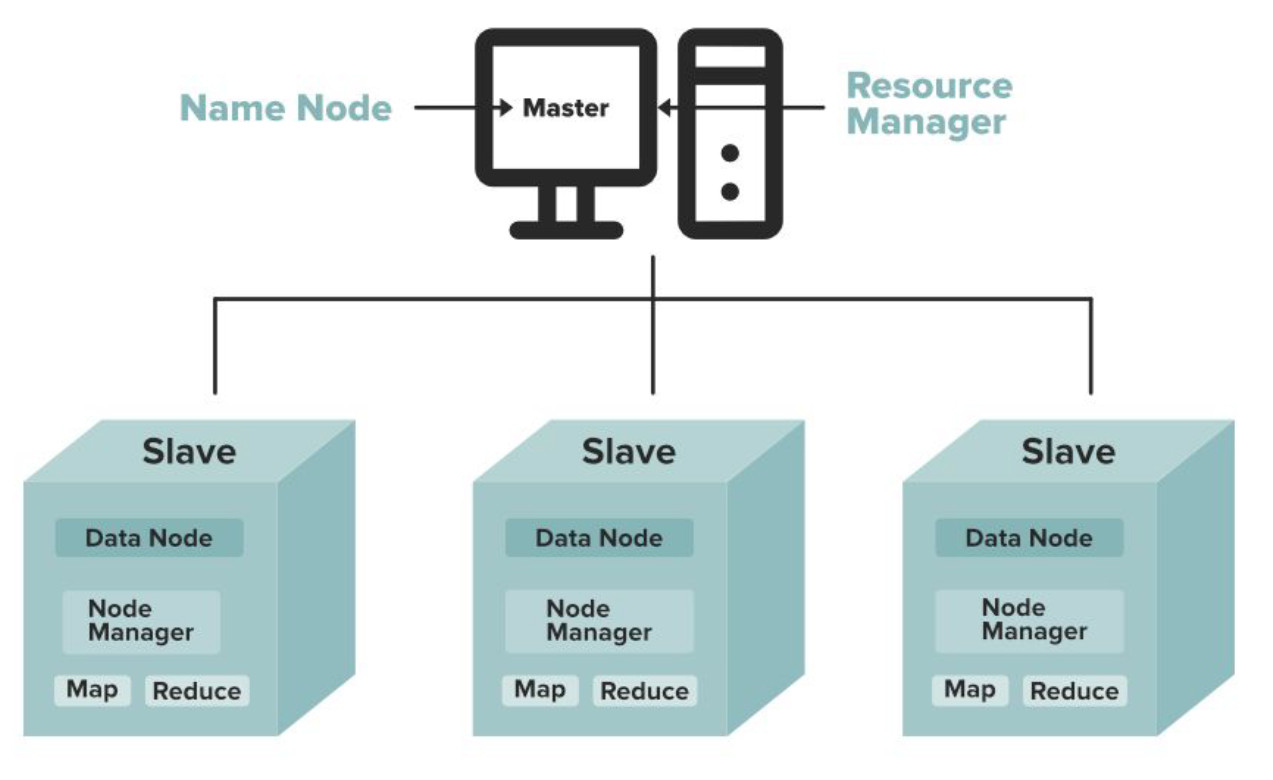
● The NameNode also knows the DataNodes on which all the blocks for a given file are located.

**DataNode:** The DataNodes are responsible for serving read and write requests from the file system’s clients. They also

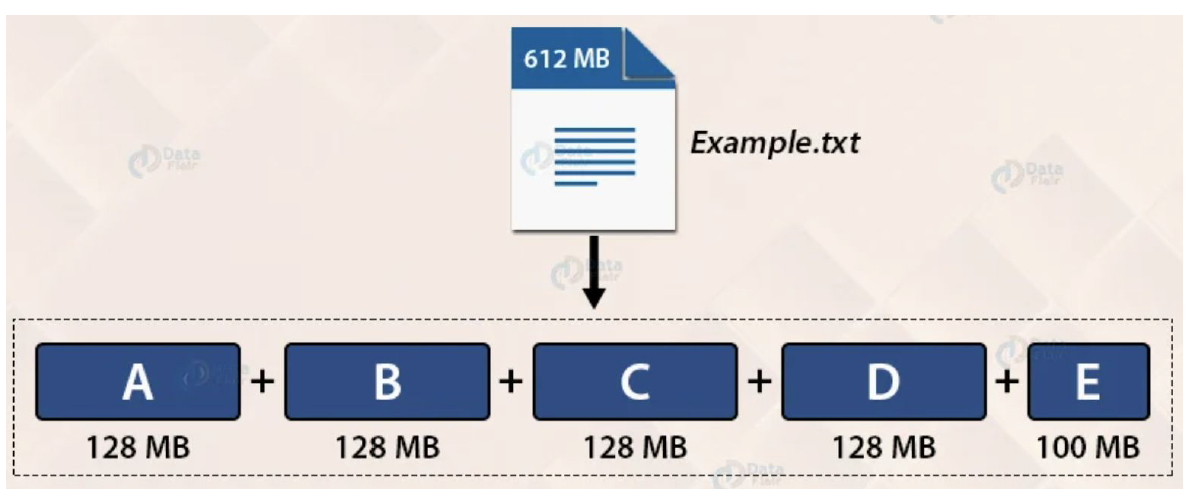
perform **block** creation, deletion, and replication upon instruction from the NameNode.

● The DataNodes manage the storage attached to the nodes that they run on.

● DataNodes regularly report back to the NameNode with lists of blocks that they are storing.



**Block:** Block is nothing but the smallest unit of storage on a computer system. It is the smallest contiguous storage allocated to a file. In Hadoop, we have a default block size of **128MB** or **256MB**.



**Problems with Hadoop Map-Reduce?**

1. ***Batch Processing:*** Hadoop and MapReduce are designed for batch processing, making them unfit for real-time or near real-time processing such as streaming data.

2. ***Complexity:*** Hadoop has a steep learning curve and its setup, configuration, and maintenance can be complex and time-consuming.

3. ***Data Movement:*** Hadoop's architecture can lead to inefficiencies and network congestion when dealing with smaller data sets.

4. ***Fault Tolerance:*** While Hadoop has data replication for fault tolerance, it can lead to inefficient storage use and doesn't cover application-level failures.

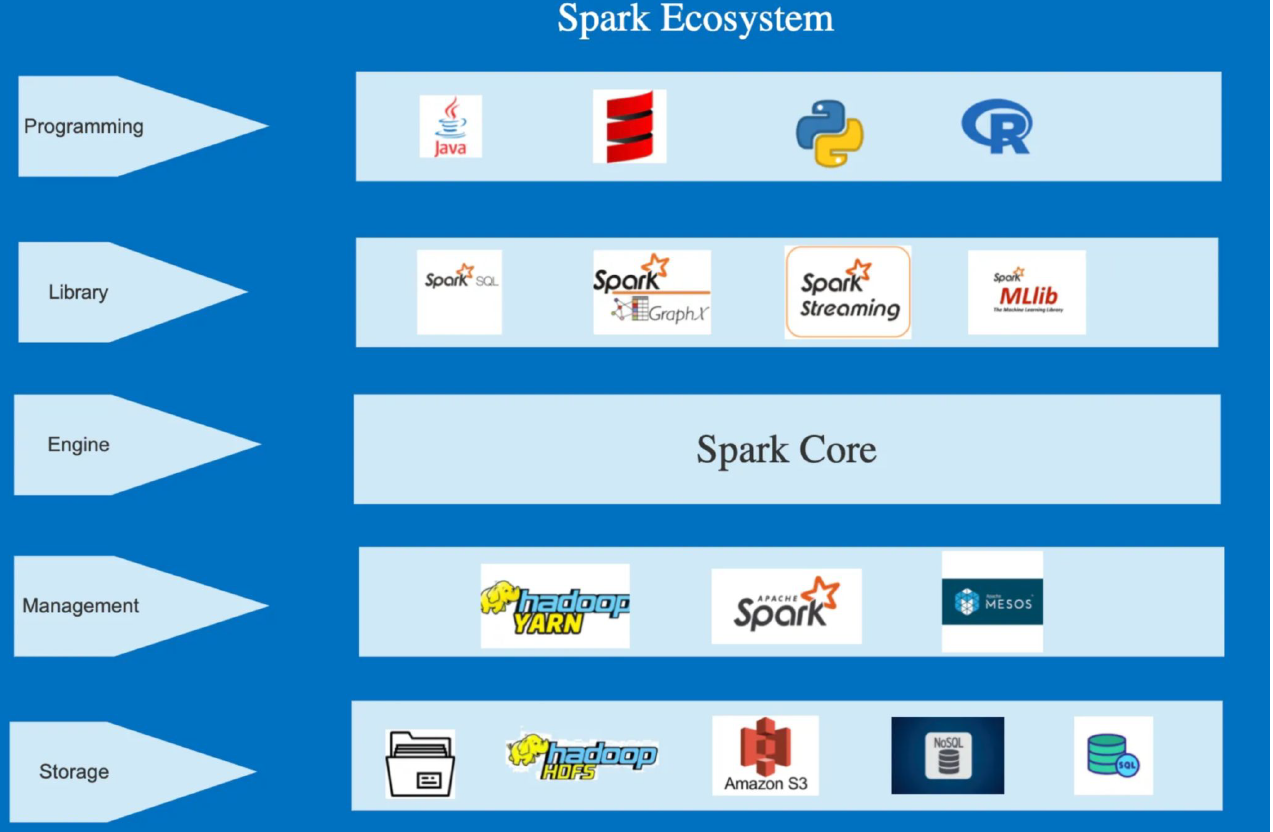
5. ***No Support for Interactive Processing:*** MapReduce doesn't support interactive processing, making it unsuitable for tasks needing back-and-forth communication.

6. ***Not Optimal for Small Files:*** Hadoop is less effective with many small files, as it's designed to handle large data files

**What is Apache Spark?**

Apache Spark is an open-source, distributed computing system designed for big data processing and analytics. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Spark is known for its speed, ease of use, and versatility in handling multiple types of data workloads, including batch processing,real-time data streaming, machine learning, and interactive queries.





**RDD in Spark**

RDDs are the building blocks of any Spark application. RDDs Stands for:

● **Resilient**: Fault tolerant and is capable of rebuilding data on failure

● **Distributed**: Distributed data among the multiple nodes in a cluster

● **Dataset**: Collection of partitioned data with valuesResilient Distributed Datasets (RDD) are a core abstraction in Apache Spark. Here are some key points about RDDs

and their properties:

1. **Fundamental Data Structure:** RDD is the fundamental data structure of Spark, which allows it to efficiently

operate on large-scale data across a distributed environment.

2. **Immutability**: Once an RDD is created, it cannot be changed. Any transformation applied to an RDD creates anew RDD, leaving the original one untouched.

3. **Resilience**: RDDs are fault-tolerant, meaning they can recover from node failures. This resilience is providedthrough a feature known as lineage, a record of all the transformations applied to the base data.

**3. Lazy Evaluation**: RDDs follow a lazy evaluation approach, meaning transformations on RDDs are not

executed immediately, but computed only when an action (like count, collect) is performed. This leads to optimized

computation.

4. **Partitioning**: RDDs are partitioned across nodes in the cluster, allowing for parallel computation on

separate portions of the dataset.

5. **In-Memory Computation**: RDDs can be stored in the memory of worker nodes, making them readily

available for repeated access, and thereby speeding up computations.

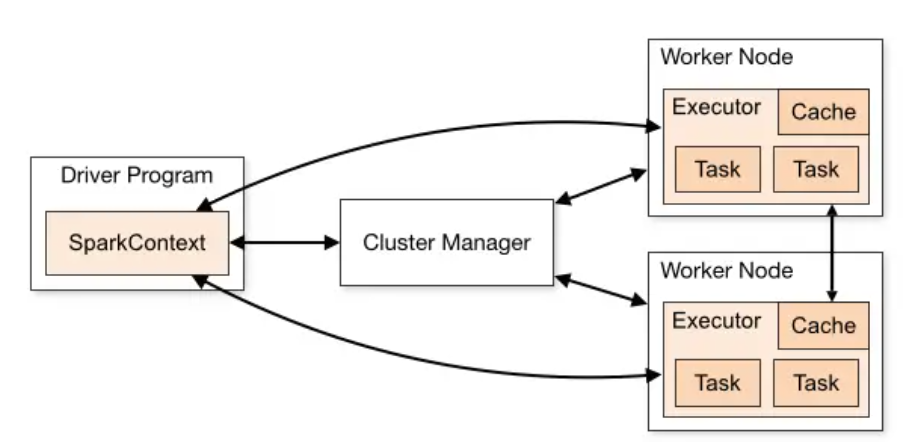
6. **Distributed Nature**: RDDs can be processed in parallel across a Spark cluster, contributing to the overall

speed and scalability of Spark.

7. **Persistence**: Users can manually persist an RDD in memory, allowing it to be reused across parallel

operations. This is useful for iterative algorithms and fast interactive use.

8. **Operations**: Two types of operations can be performed on RDDs - transformations (which create a newRDD) and actions (which return a value to the driver program or write data to an external storage system).

****

df=spark.read.format("csv").option("header","true").load("abfss://spark\_practice@onelake.dfs.fabric.microsoft.com/Demo\_Spark.Lakehouse/Files/Apr2021.csv")

# df now is a Spark DataFrame containing CSV data from "abfss://spark\_practice@onelake.dfs.fabric.microsoft.com/Demo\_Spark.Lakehouse/Files/Apr2021.csv".

display(df)

## **Logistic Regression**

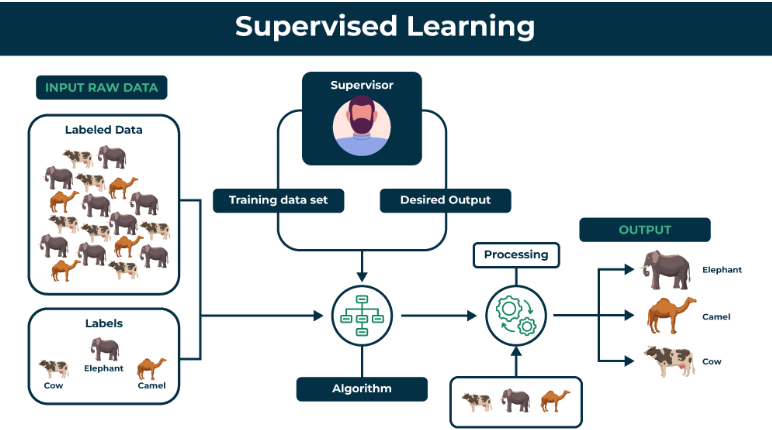
Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.

In the simplest case there are two outcomes, which is called binomial, an example of which is predicting if a tumor is malignant or benign. Other cases have more than two outcomes to classify, in this case it is called multinomial. A common example for multinomial logistic regression would be predicting the class of an iris flower between 3 different species.

Here we will be using basic logistic regression to predict a binomial variable. This means it has only two possible outcomes.

**Supervised Learning**

In [supervised learning](https://www.geeksforgeeks.org/supervised-machine-learning/), the machine is trained on a **set of labeled data**, which means that the **input data is paired with the desired output**. The machine then learns to **predict the output for new input data**. Supervised learning is often used for tasks such as **classification, regression, and object detection.** Labeled data is **data that has been tagged with a correct answer or classification.**



**Key Points:**

* Supervised learning involves training a machine from labeled data.
* Labeled data consists of examples with the correct answer or classification.
* The machine learns the relationship between inputs (fruit images) and outputs (fruit labels).
* The trained machine can then make predictions on new, unlabeled data.

### **1- Regression**

Regression is a type of supervised learning that is used to **predict continuous values,** such as **house prices, stock prices,** or customer churn. Regression algorithms learn a function that maps from the **input features to the output value.**

Some common regression algorithms include:

* Linear Regression
* Polynomial Regression
* Support Vector Machine Regression
* Decision Tree Regression
* Random Forest Regression

### **2- Classification**

Classification is a type of supervised learning that is used to predict **categorical values**, such as whether a **customer will churn or not, whether an email is spam or not, or whether a medical image shows a tumor or not.** Classification algorithms **learn a function that maps from the input features to a probability distribution over the output classes.**

Some common classification algorithms include:

* Logistic Regression
* Support Vector Machines
* Decision Trees
* Random Forests
* Naive Baye

### **Advantages of Supervised learning**

* Supervised learning **allows collecting data** and **produces data output from previous experiences**.
* Helps to **optimize performance** criteria with the help of experience.
* Supervised machine learning helps to **solve** various types of **real-world computation problems**.
* It performs **classification** and **regression** tasks.
* It allows **estimating or mapping** the **result to a new sample.**
* We have **complete control** over **choosing the number of classes we want in the training data.**

### **Disadvantages of Supervised learning**

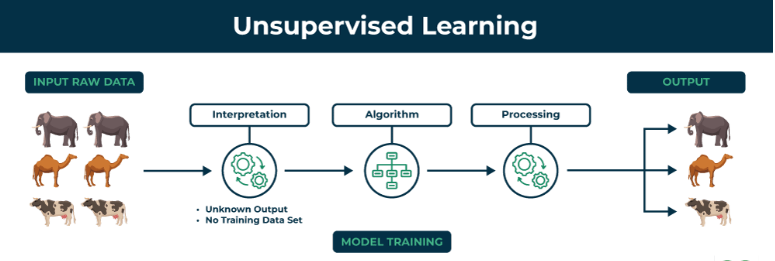
* Classifying **big data** can be challenging.
* Training for supervised learning **needs a lot of computation time.** So, it requires a lot of time.
* Supervised learning **cannot handle all complex tasks** in Machine Learning.
* Computation time is vast for supervised learning.
* It **requires** a **labelled data set**.
* It **requires** a **training process.**

## **What is Unsupervised learning?**

Unsupervised learning is a type of machine learning that **learns from unlabeled data.** This means that the **data does not have any pre-existing labels or categories.** The **goal** of unsupervised learning is to **discover patterns and relationships in the data without any explicit guidance.**

 Here the task of the machine is to **group unsorted information according to similarities, patterns, and differences without any prior training of data.**

In unsupervised learning, the **machine is trained on a set of unlabeled data**, which means that the **input data is not paired with the desired output**. The machine then learns to find patterns and relationships in the data. Unsupervised learning is often used for tasks such as **clustering, dimensionality reduction, and anomaly detection**

****

**Key Points**

* Unsupervised learning allows the model to **discover patterns and relationships in unlabeled data.**
* Clustering algorithms **group similar data points together** based on their **inherent characteristics**.
* **Feature extraction** captures **essential information from the data**, enabling the model to make meaningful distinctions.
* **Label association assigns categories to the clusters** based on the **extracted patterns and characteristics**

## **Types of Unsupervised Learning**

Unsupervised learning is classified into two categories of algorithms:

* **Clustering**: A clustering problem is where you want to discover the **inherent groupings** in the data, such as grouping customers by purchasing behavior.

**Clustering Types:-**

1. Hierarchical clustering
2. K-means clustering
3. Principal Component Analysis
4. Singular Value Decomposition
5. Independent Component Analysis
6. Gaussian Mixture Models (GMMs)
7. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

* **Association**: An **association rule** learning problem is where you want to **discover rules that describe large portions of your data**, such as people that buy X also tend to buy Y.

Some common association rule learning algorithms include:

* Apriori Algorithm
* Eclat Algorithm
* FP-Growth Algorithm

### **Advantages** **of Unsupervised learning**

* It does **not require** training **data to be labeled**.
* **Dimensionality reduction** can be **easily accomplished** using unsupervised learning.
* Capable of **finding previously unknown patterns** in data.
* Unsupervised learning can help you **gain insights from unlabeled data** that you might not have been able to get otherwise.
* Unsupervised learning is **good at finding patterns and relationships in data without being told what to look for**. This can help you learn new things about your data.

### **Disadvantages of Unsupervised learning**

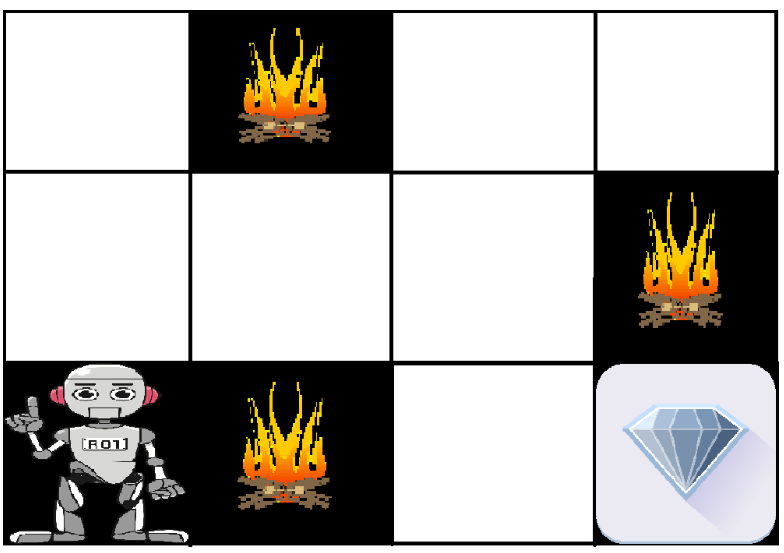
* **Difficult to measure accuracy or effectiveness** due to **lack of predefined answers** during training.
* The results often have **lesser accuracy**.
* The **user needs to spend time interpreting and label the classes which follow that classification**.
* Unsupervised learning can be **sensitive to data quality, including missing values, outliers, and noisy data.**
* Without labeled data, it can be **difficult to evaluate the performance of unsupervised learning models, making it challenging to assess their effectiveness.**

# Reinforcement learning

# Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

In RL, the data is accumulated from machine learning systems that use a **trial-and-error method**. Data is not part of the input that we would find in supervised or unsupervised machine learning.

Reinforcement learning uses algorithms that learn from outcomes and decide which action to take next.





* Input: The input should be an initial state from which the model will start
* Output: There are many possible outputs as there are a variety of solutions to a particular problem
* Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
* The model keeps continues to learn.
* The best solution is decided based on the maximum reward.
* **Advantages of Reinforcement learning**
* 1. Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques.
* 2. The model can correct the errors that occurred during the training process.
* 3. In RL, training data is obtained via the direct interaction of the agent with the environment
* 4. Reinforcement learning can handle environments that are non-deterministic, meaning that the outcomes of actions are not always predictable. This is useful in real-world applications where the environment may change over time or is uncertain.
* 5. Reinforcement learning can be used to solve a wide range of problems, including those that involve decision making, control, and optimization.
* 6. Reinforcement learning is a flexible approach that can be combined with other machine learning techniques, such as deep learning, to improve performance.
* **Disadvantages of Reinforcement learning**
* 1. Reinforcement learning is not preferable to use for solving simple problems.
* 2. Reinforcement learning needs a lot of data and a lot of computation
* 3. Reinforcement learning is highly dependent on the quality of the reward function. If the reward function is poorly designed, the agent may not learn the desired behavior.
* 4. Reinforcement learning can be difficult to debug and interpret. It is not always clear why the agent is behaving in a certain way, which can make it difficult to diagnose and fix problems.

import gym

import numpy as np

# Define the Q-table and learning rate

q\_table = np.zeros((state\_size, action\_size))

alpha = 0.8

gamma = 0.95

# Train the Q-Learning algorithm

for episode in range(num\_episodes):

state = env.reset()

done = False

while not done:

# Choose an action

action = np.argmax(

q\_table[state, :] + np.random.randn(1, action\_size) \* (1. / (episode + 1)))

# Take the action and observe the new state and reward

next\_state, reward, done, \_ = env.step(action)

# Update the Q-table

q\_table[state, action] = (1 - alpha) \* q\_table[state, action] + \

alpha \* (reward + gamma \* np.max(q\_table[next\_state, :]))

state = next\_state

# Test the trained Q-Learning algorithm

state = env.reset()

done = False

while not done:

# Choose an action

action = np.argmax(q\_table[state, :])

# Take the action

state, reward, done, \_ = env.step(action)

env.render()

# Natural Language Processing

Natural language processing (NLP) is a **subfield of Artificial Intelligence (AI)**. This is a widely used technology for **personal assistants** that are used in **various business fields/areas**. This technology works on the **speech provided by the user breaks** it down for **proper understanding and processes it accordingly**.

Knowledge representation, logical reasoning, and constraint satisfaction were the emphasis of AI applications in NLP. Here first **it was applied to semantics and later to grammar**.

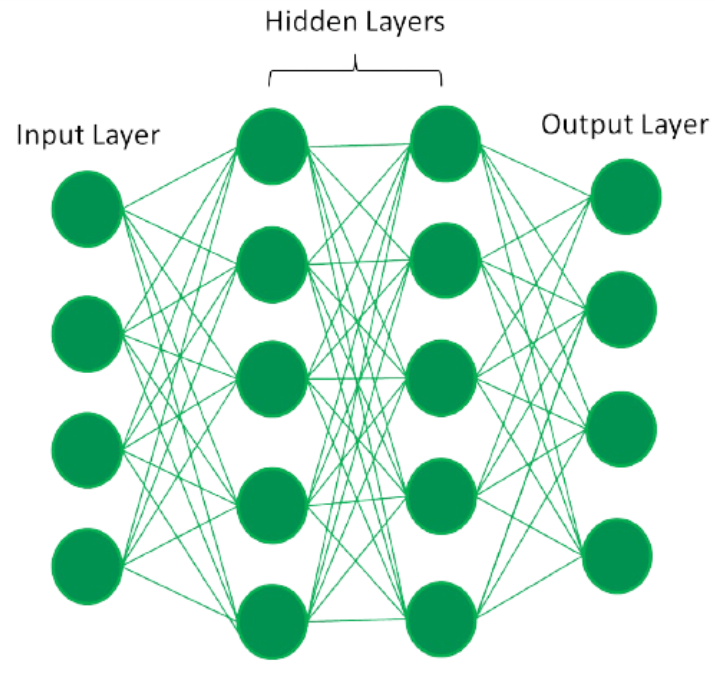
Natural Language Processing (NLP) is a field that **combines computer science, linguistics, and machine learning to study** how **computers and humans communicate in natural language.** The goal of NLP is for computers to be able to interpret and generate human language. This not only improves the efficiency of work done by humans but also helps in interacting with the machine. NLP bridges the gap of interaction between humans and electronic devices.

NLP is used in a wide range of applications, including machine translation, sentiment analysis, speech recognition, chatbots, and text classification. Some common techniques used in NLP include:

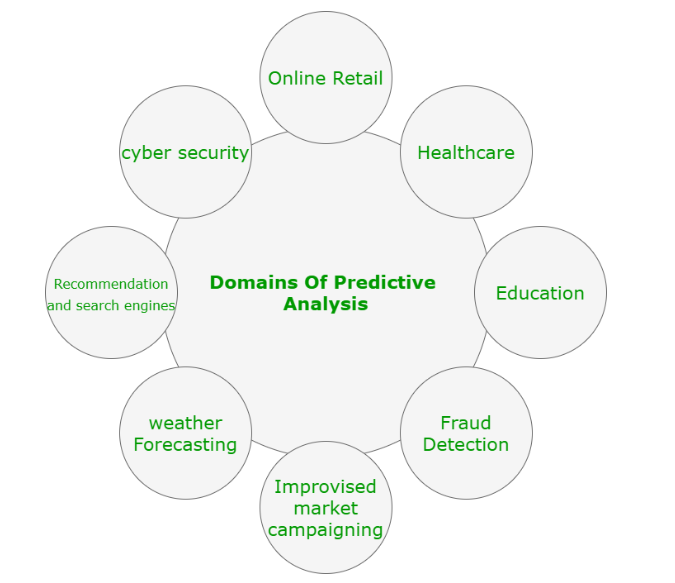
1. Tokenization: the process of breaking text into individual words or phrases.
2. Part-of-speech tagging: the process of labeling each word in a sentence with its grammatical part of speech.
3. Named entity recognition: the process of identifying and categorizing named entities, such as people, places, and organizations, in text.
4. Sentiment analysis: the process of determining the sentiment of a piece of text, such as whether it is positive, negative, or neutral.
5. Machine translation: the process of automatically translating text from one language to another.
6. Text classification: the process of categorizing text into predefined categories or topics.

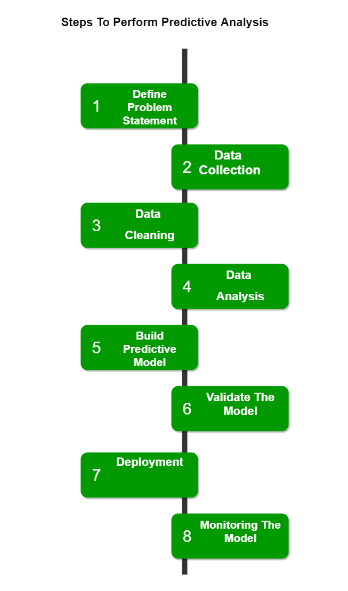
**Deep Learning**

Deep learning is a **branch of machine learning** which is **based on artificial neural networks**. It is capable of **learning complex patterns and relationships within data**. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is **based on artificial neural networks (ANNs)** also known as **deep neural networks (DNNs)**. These neural networks are **inspired by the structure and function of the human brain’s biological neurons**, and they are **designed to learn from large amounts of data.**



# Step by Step Predictive Analysis – Machine Learning





# Time Series Analysis

Time series visualization and analytics empower users to graphically represent time-based data, enabling the identification of trends and the tracking of changes over different periods. This data can be presented through various formats, such as line graphs, gauges, tables, and more.

The utilization of time series visualization and analytics facilitates the extraction of insights from data, enabling the generation of forecasts and a comprehensive understanding of the information at hand. Organizations find substantial value in time series data as it allows them to analyze both real-time and historical metrics.

### Basic Time Series Concepts

* **Trend:** A [trend](https://www.geeksforgeeks.org/what-is-a-trend-in-time-series/)represents the general direction in which a time series is moving over an extended period. It indicates whether the values are increasing, decreasing, or staying relatively constant.
* **Seasonality:** [Seasonality](https://www.geeksforgeeks.org/seasonality-detection-in-time-series-data/) refers to recurring patterns or cycles that occur at regular intervals within a time series, often corresponding to specific time units like days, weeks, months, or seasons.
* **Moving average:**The [moving average](https://www.geeksforgeeks.org/program-find-simple-moving-average/)method is a common technique used in time series analysis to smooth out short-term fluctuations and highlight longer-term trends or patterns in the data. It involves calculating the average of a set of consecutive data points, referred to as a “window” or “rolling window,” as it moves through the time series
* **Noise:** Noise, or random fluctuations, represents the irregular and unpredictable components in a time series that do not follow a discernible pattern. It introduces variability that is not attributable to the underlying trend or seasonality.
* **Differencing:**Differencing is used to make the difference in values of a specified interval. By default, it’s one, we can specify different values for plots. It is the most popular method to remove trends in the data.
* **Stationarity:**A [stationary time series](https://www.geeksforgeeks.org/how-to-check-if-time-series-data-is-stationary-with-python/) is one whose statistical properties, such as mean, variance, and autocorrelation, remain constant over time.
* **Order:**The order of differencing refers to the number of times the time series data needs to be differenced to achieve stationarity.
* **Autocorrelation**[: Autocorrelation](https://www.geeksforgeeks.org/autocorrelation/), is a statistical method used in time series analysis to quantify the degree of similarity between a time series and a lagged version of itself.
* **Resampling**: [Resampling](https://www.geeksforgeeks.org/how-to-resample-time-series-data-in-python/)is a technique in time series analysis that involves changing the frequency of the data observations. It’s often used to transform the data to a different frequency (e.g., from daily to monthly) to reveal patterns or trends more clearly.

### Types of Time Series Data

Time series data can be broadly classified into two sections:

**1. Continuous Time Series Data:**Continuous time series data involves measurements or observations that are recorded at regular intervals, forming a seamless and uninterrupted sequence. This type of data is characterized by a continuous range of possible values and is commonly encountered in various domains, including:

* Temperature Data: Continuous recordings of temperature at consistent intervals (e.g., hourly or daily measurements).
* Stock Market Data: Continuous tracking of stock prices or values throughout trading hours.
* Sensor Data: Continuous measurements from sensors capturing variables like pressure, humidity, or air quality.

2. **Discrete Time Series Data:**Discrete time series data, on the other hand, consists of measurements or observations that are limited to specific values or categories. Unlike continuous data, discrete data does not have a continuous range of possible values but instead comprises distinct and separate data points. Common examples include:

* Count Data: Tracking the number of occurrences or events within a specific time period.
* Categorical Data: Classifying data into distinct categories or classes (e.g., customer segments, product types).
* Binary Data: Recording data with only two possible outcomes or states.



**import** numpy as np

**import** pandas as pd

**import** matplotlib.pyplot as plt

**import** seaborn as sns

**from** matplotlib **import** gridspec

# ****Code : Loading the Data****

# data = pd.read\_csv("credit.csv")

# ****Code : Understanding the Data****

data.head()

**Code : Describing the Data**

fraud **=** data[data['Class'] **==** 1]

valid **=** data[data['Class'] **==** 0]

outlierFraction **=** len(fraud)**/**float(len(valid))

**print**(outlierFraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] **==** 1])))

print('Valid Transactions: {}'.format(len(data[data['Class'] **==** 0])))

print(data.shape)

print(data.describe())

**Code : Imbalance in the data**

fraud **=** data[data['Class'] **==** 1]

valid **=** data[data['Class'] **==** 0]

outlierFraction **=** len(fraud)**/**float(len(valid))

**print**(outlierFraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] **==** 1])))

print('Valid Transactions: {}'.format(len(data[data['Class'] **==** 0])))

**Code : Print the amount details for Fraudulent Transaction**

print(“Amount details of the fraudulent transaction”)

fraud.Amount.describe()

**Code : Print the amount details for Normal Transaction**

print(“details of valid transaction”)

valid.Amount.describe()

**Code : Plotting the Correlation Matrix**

corrmat **=** data.corr()

fig **=** plt.figure(figsize **=** (12, 9))

sns.heatmap(corrmat, vmax **=** .8, square **=** True)

plt.show()

# ****Code : Separating the X and the Y values****

# dividing the X and the Y from the dataset

X **=** data.drop(['Class'], axis **=** 1)

Y **=** data["Class"]

**print**(X.shape)

print(Y.shape)

# getting just the values for the sake of processing

# (its a numpy array with no columns)

xData **=** X.values

yData **=** Y.values

**Training and Testing Data Bifurcation**

# Using Scikit-learn to split data into training and testing sets

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

# Split the data into training and testing sets

xTrain, xTest, yTrain, yTest = train\_test\_split(

xData, yData, test\_size = 0.2, random\_state = 42)

**Code : Building a Random Forest Model using scikit learn**

# Building the Random Forest Classifier (RANDOM FOREST)

from sklearn.ensemble import RandomForestClassifier

# random forest model creation

rfc = RandomForestClassifier()

rfc.fit(xTrain, yTrain)

# predictions

yPred = rfc.predict(xTest)

**Code : Building all kinds of evaluating parameters**

# Evaluating the classifier

# printing every score of the classifier

# scoring in anything

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.metrics import precision\_score, recall\_score

from sklearn.metrics import f1\_score, matthews\_corrcoef

from sklearn.metrics import confusion\_matrix

n\_outliers = len(fraud)

n\_errors = (yPred != yTest).sum()

print("The model used is Random Forest classifier")

acc = accuracy\_score(yTest, yPred)

print("The accuracy is {}".format(acc))

prec = precision\_score(yTest, yPred)

print("The precision is {}".format(prec))

rec = recall\_score(yTest, yPred)

print("The recall is {}".format(rec))

f1 = f1\_score(yTest, yPred)

print("The F1-Score is {}".format(f1))

MCC = matthews\_corrcoef(yTest, yPred)

print("The Matthews correlation coefficient is{}".format(MCC))

**Code : Visualizing the Confusion Matrix**

# printing the confusion matrix

LABELS **=** ['Normal', 'Fraud']

conf\_matrix **=** confusion\_matrix(yTest, yPred)

plt.figure(figsize **=**(12, 12))

sns.heatmap(conf\_matrix, xticklabels **=** LABELS,

            yticklabels **=** LABELS, annot **=** True, fmt **=**"d");

plt.title("Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

