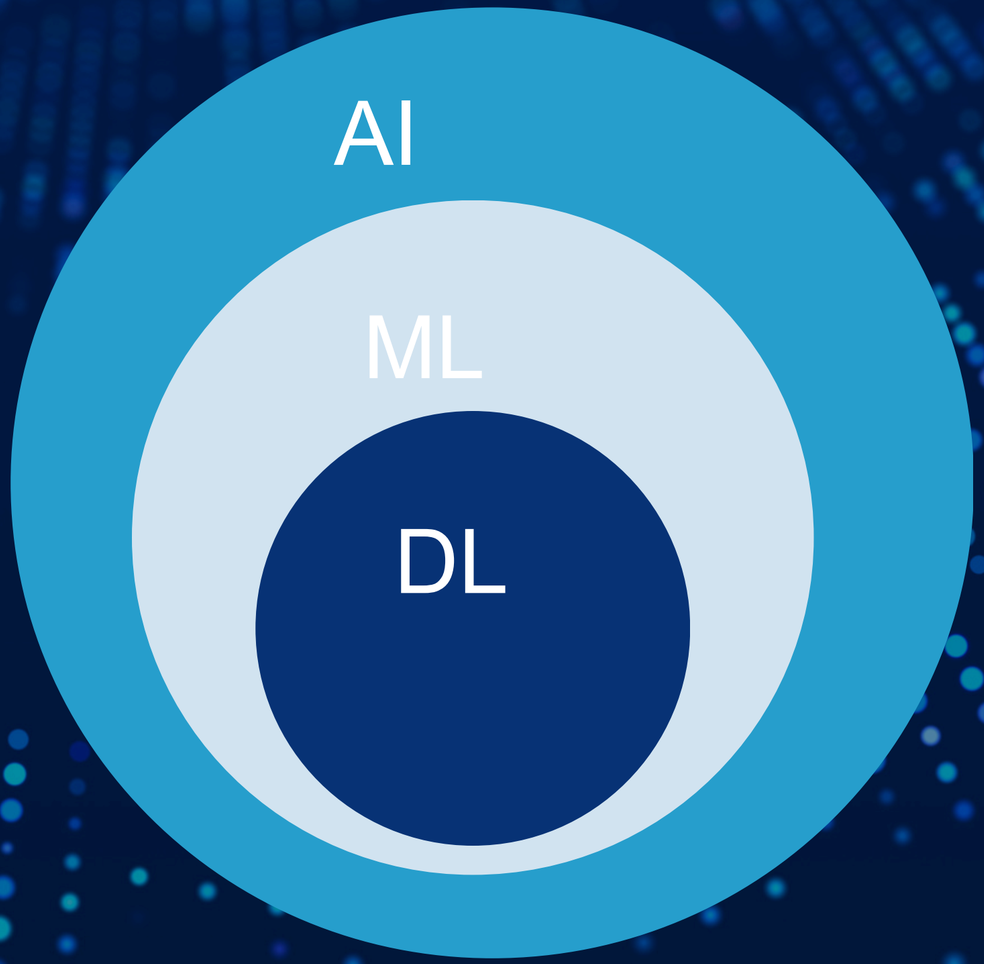
ML SIG

DAY 1

Understanding Basic Definitions



AI :The field of creating machines that can think, reason, and perform tasks like humans.

ML :A subset of AI that enables machines to learn from data and improve their performance without explicit programming

DL :A specialized branch of ML that uses artificial neural networks to process large amounts of data and make complex decisions

Overview Of Machine Learning

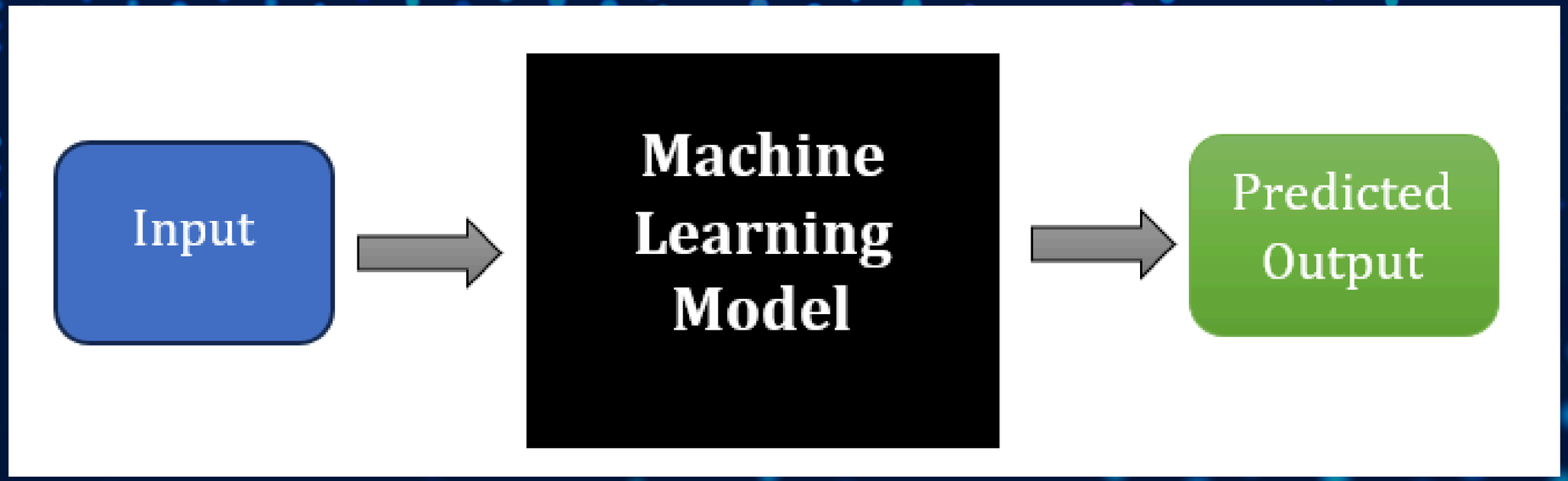
- subset of Artificial Intelligence
- process of training a piece of software (called a model)
- Example-
  Transportation: Self-driving cars and Traffic Prediction with Machine Learning

TEACHABLE LEARNING

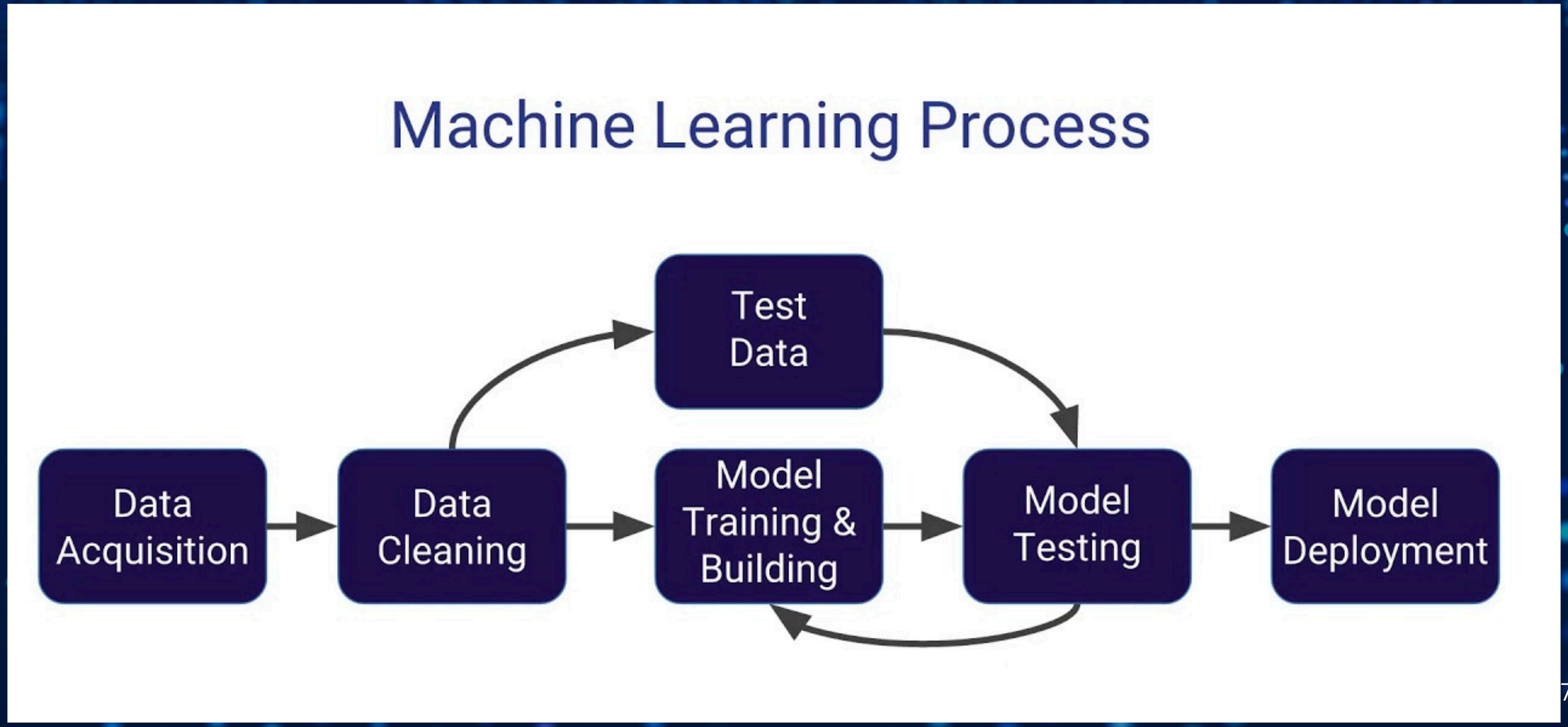
https://teachablemachine.withgoogle.com/

What is Machine Learning Model

- Machine Learning model is a program
- Learns from data
- Makes predictions or decisions
- without being explicitly programmed

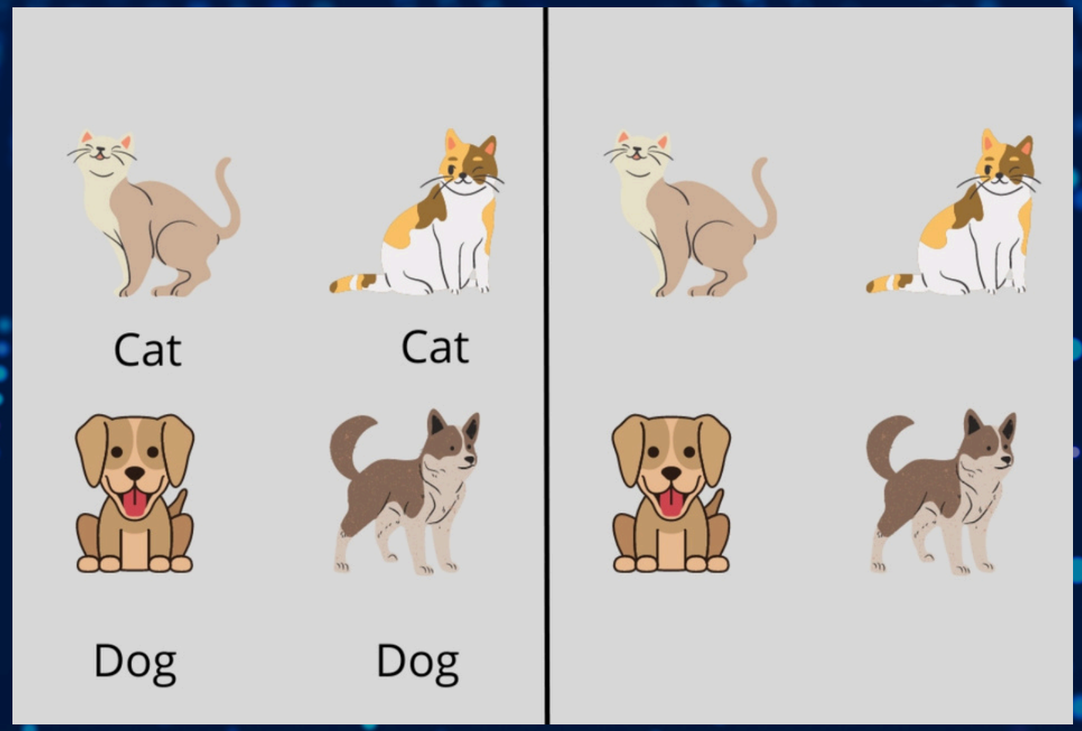


Steps to build ML Model



Basic Terminologies

Labelled And Unlabelled Data



- **Labelled Data-** Includes one or more features and a label.

- **Unlabelled Data -** Includes one or more features but no label

Ex- To predict the value of a house
Features: Bedrooms, Bathrooms, Balcony, Area, Property Age

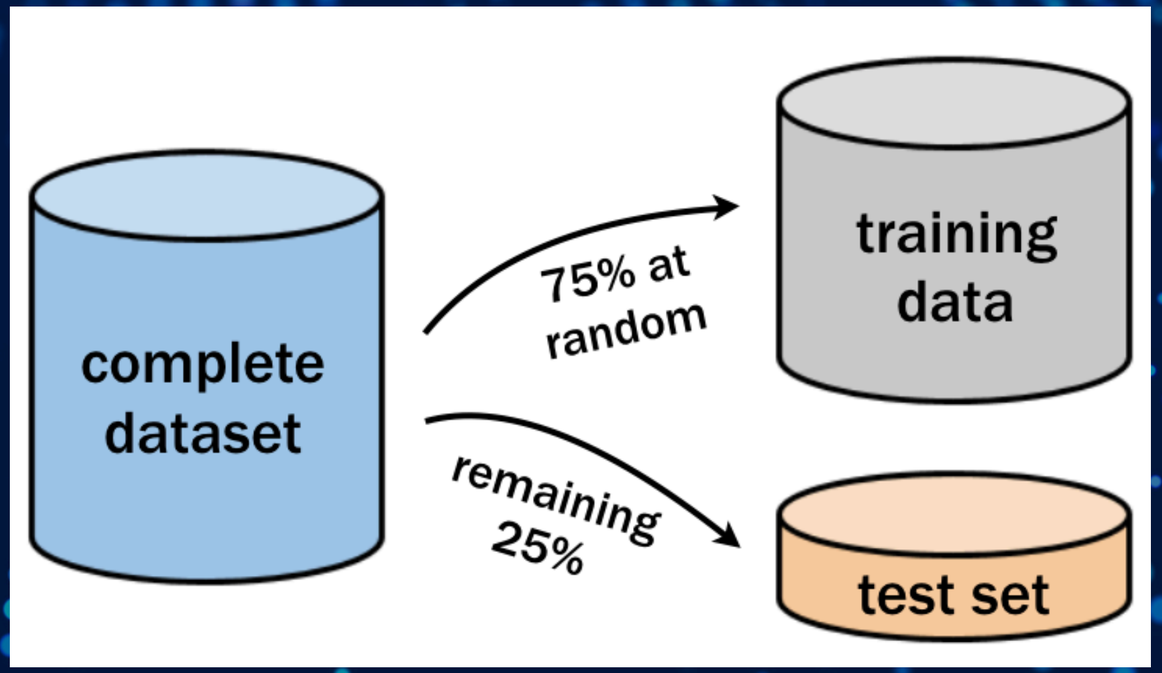
- Labelled Data-

Label: Property Price

- Unlabelled Data-

Label: None (no target variable, such as Property Price)

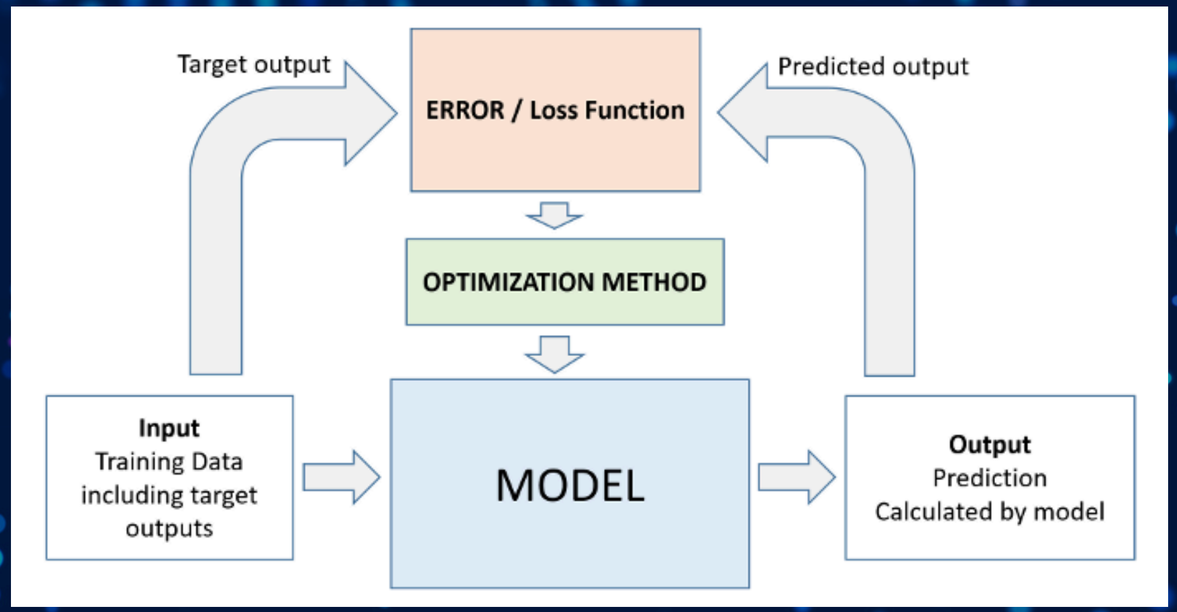
Train And Test Data



- **Train Data:** The dataset used to teach a machine learning model by helping it learn patterns.
  Ex:- Study Material & Practice Questions that a student use to study

- **Test Data:** A separate dataset used to check how well the trained model makes predictions on new, unseen data.
  Ex:- Final Exam having new set of questions

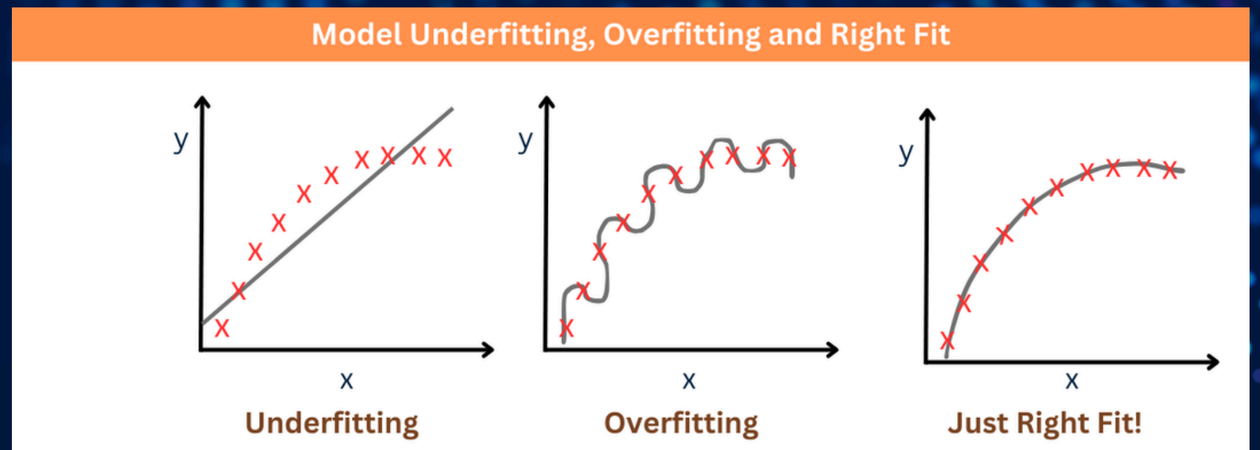
Loss Function And Optimization



- Loss Function- A measure of error between predicted and actual values. Lower loss function better accuracy.
  Ex:- Like making tea with guessed sugar—error is the difference from the perfect taste.

- Optimization - The process of minimizing the loss function by adjusting model parameters.
  Ex:-Adjusting the sugar amount to get the perfect taste.

Overfitting Underfitting and Generalized Model



Bias	High	Low	Low
Variance	Low	High	Low

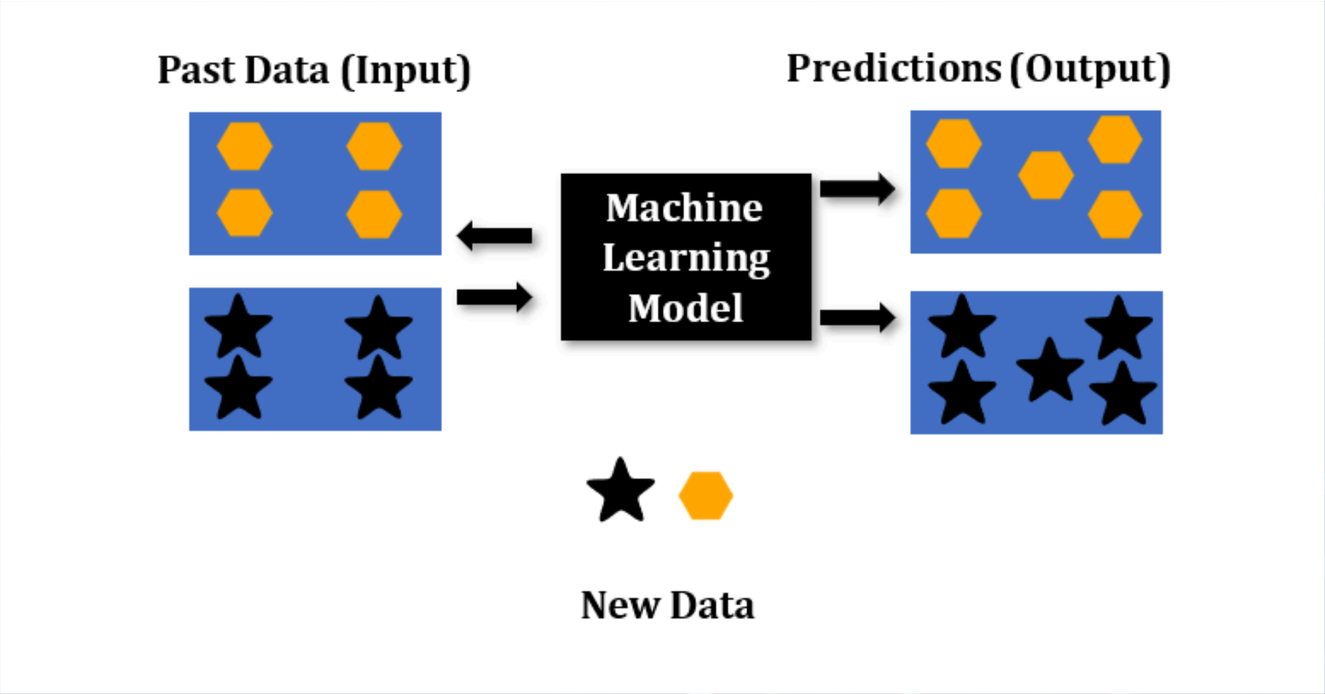
- Underfitting:-Too simple; fails to learn patterns.
  Ex:- barely study and fail to answer even basic questions.

- Generalized Model:- Balances learning and works well on new data.
  Ex:- understand concepts and can solve both familiar and new questions.

- Overfitting: Learns too much, including noise; fails on new data.
  Ex:-memorize PYQs but struggle with new questions.

Types Of Machine Learning

Supervised Machine Learning

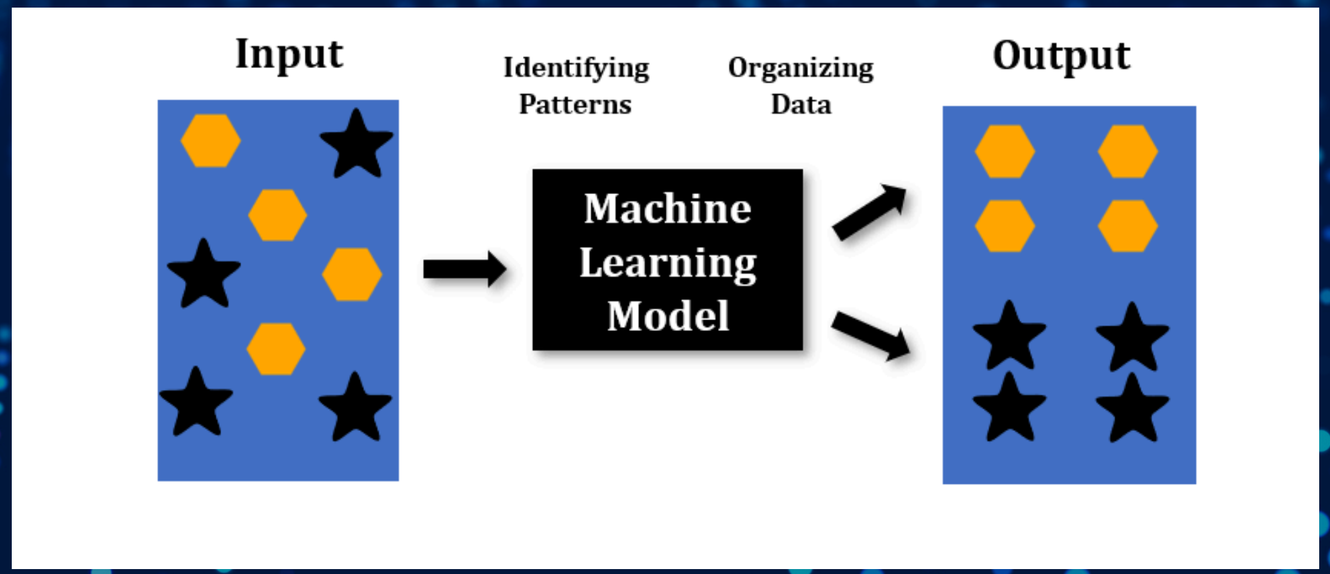


- In supervised learning, the machine is enabled to predict based on labeled data fed to it.
- Models are trained on labeled examples and make predictions on unlabelled examples.

Following are the two types of Supervised learning:

- Classification
- Regression

Unsupervised Machine Learning

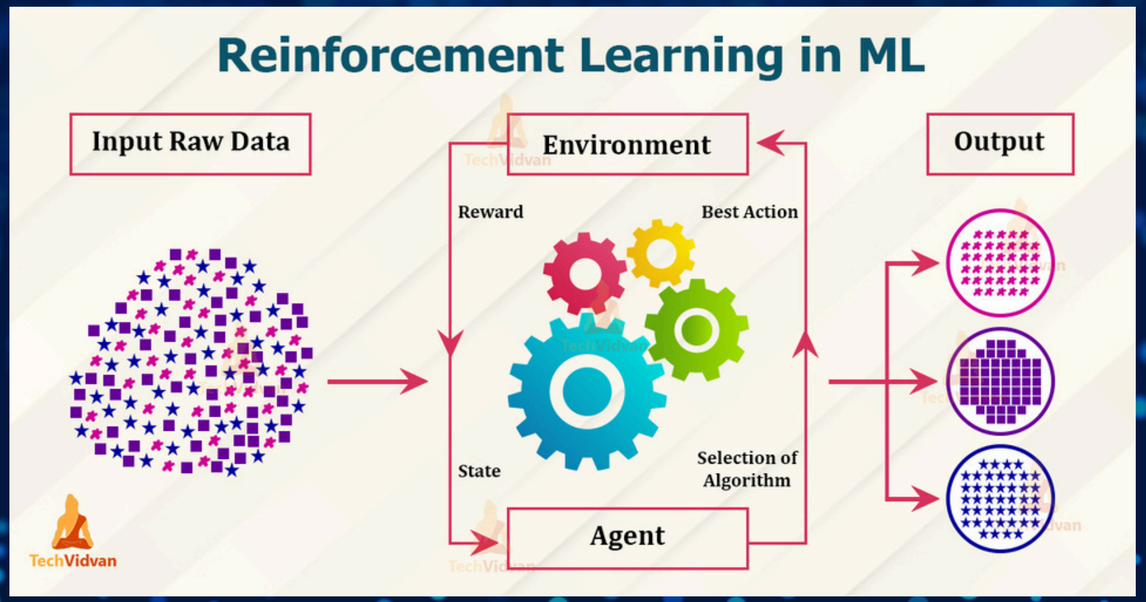


- The model identifies patterns and relationships in the input without any prior knowledge about the data.
- Model is given unlabeled data.
- unsupervised learning allows identifying and discovering patterns and insights on their own. This process is without explicit instruction.

Types of unsupervised machine learning:-

- Clustering
- Association

Reinforcement Learning



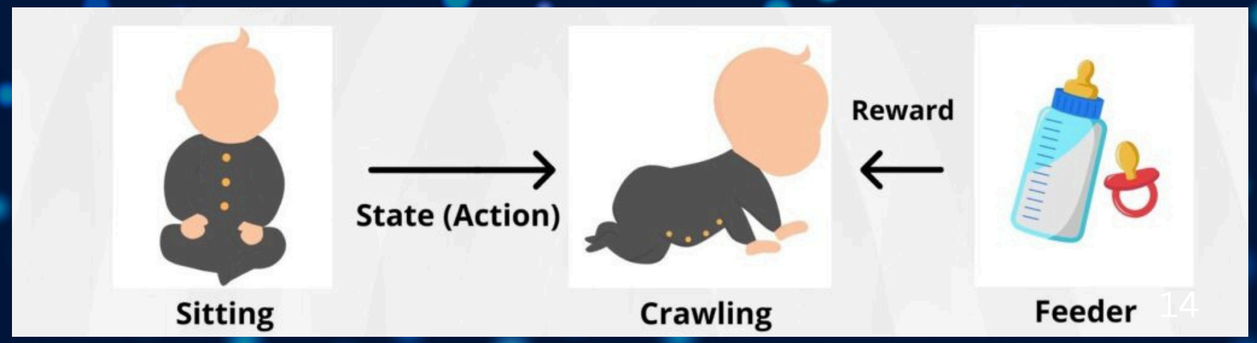
- ML model learns based on the rewards it received (or penalties) for its previous actions.

Process Of Reinforcement Learning

1. Exploration: The agent tries different actions to see their outcomes.
2. Exploitation: Knowledge gained is used to maximize rewards.
3. Learning: Strategy is updated based on feedback.

Components of reinforcement learning:-

1. Agent: The learner
2. Environment: Surrounding
3. Actions
4. Rewards: Feedback from the environment



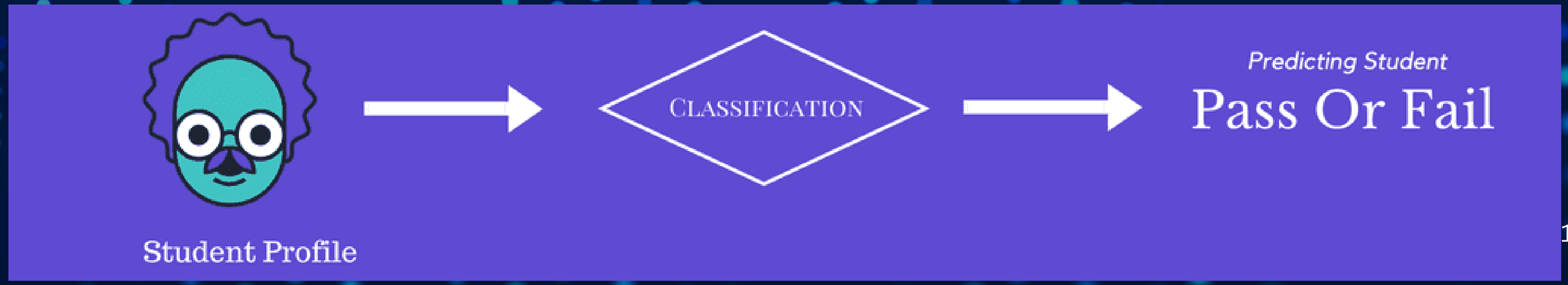
Types Of Supervised Learning

Classification

- In classification the goal is to predict a categorical output variable based on one or more input features.

- Ex:- 1. Spam vs. Not Spam email classification
  2. Cancer diagnosis based on tumor characteristics

- Algorithms used- Decision Trees, K-Nearest Neighbors, Support Vector Machines. Neural Networks are also used

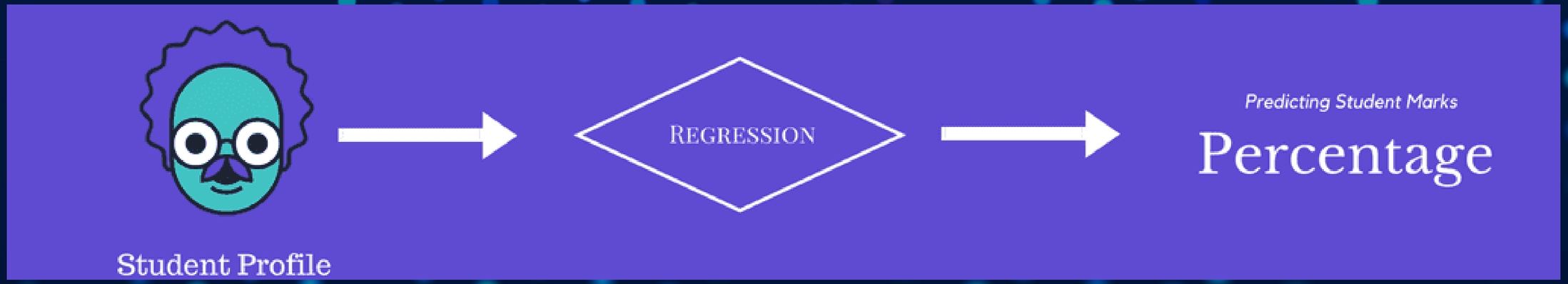


Regression

- In regression model, the goal is to predict a continuous output variable based on one or more input features.

- Ex:- 1. Age Prediction: based on height, weight, health, and other factors.
  2.House Price Prediction: based on number of rooms, bathrooms, balcony, etc.

- Types:- 1. Linear Regression
  2. Logistic Regression
  3. Ordinal Regression
  4. Poisson Regression



PANDAS

What is Pandas ?

Pandas is an open-source Python library for data analysis and manipulation.

It provides fast, flexible, and easy-to-use tools for handling structured data.

The Key features of Pandas are :
1] Data Cleaning & Preprocessing
2] Filtering & Sorting
3] Aggregation & Analysis

Works with:
- CSV
- Excel
- SQL
- JSON

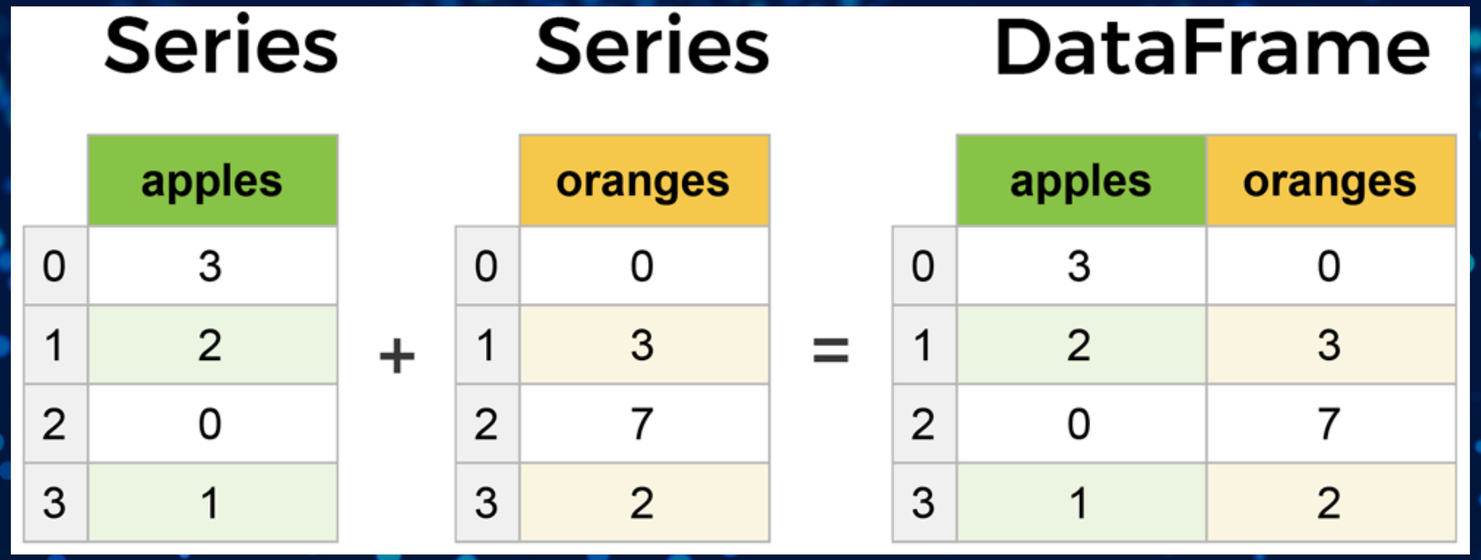
Series Vs Dataframe

Series :

- One column with an index
- Single data type
- pd.Series([10, 20, 30])
-

DataFrame :

- Multiple columns with row & column indexes
- Multiple data types
- pd.DataFrame({'A': [10, 20], 'B': [30, 40]})



Data Importing and Exporting

Importing data

Data can be imported from and exported to :

- CSV
- Excel
- JSON
- SQL
- URI's , etc

Syntax : pd.read_csv("data.csv")

pd.to_csv("data.csv" , index = false)

```
#Exporting data frames
car_sales.to_csv("exported-car-sales.csv" , index = False)
#car_sale.to_excel("exported-car-sales.excel")  this is to export file as excel
```

```
exported_car_sales = pd.read_csv("exported-car-sales.csv" )
```

Exploring and Understanding Data

Viewing First and Last Rows of dataset

- The first and last few rows of a dataset can be accesed by the method df.head() , df.tail()
- By default it prints 5 rows
- You can print as many rows as you want

```
# You can also print as many rows as you want by using
car_sales.head(7)
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
1	Honda	Red	87899.0	4.0	$5,000
2	Toyota	Blue	NaN	3.0	$7,000
3	BMW	Black	11179.0	5.0	$22,000
4	Nissan	White	213095.0	4.0	$3,500
5	Toyota	Green	NaN	4.0	$4,500
6	Honda	NaN	NaN	4.0	$7,500

```
car_sales.head() #Returns first five rows
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
1	Honda	Red	87899.0	4.0	$5,000
2	Toyota	Blue	NaN	3.0	$7,000
3	BMW	Black	11179.0	5.0	$22,000
4	Nissan	White	213095.0	4.0	$3,500

Exploring and Understanding Data

Checking Column names , dtypes , missing values & summary statistics

- **View Column Names**: Use df.columns to list all column names.
- **Check Data Types**: Use df.dtypes to identify numeric and categorical data.
- **Detect Missing Values**: Use df.isnull().sum() to count missing values in each column.
- **Get Summary Statistics** : df.describe() provides key metrics like mean, median, min, max, and standard deviation for numerical columns.

Indexing & Selecting Data

How to select Columns ?

- To select a single column use :
  df[“col_Name”]
- To select multiple columns use :
  df[[“col1” , “col2”]]

How to select Rows ?

- Similar to list slicing in Python.:
  df[start_index : end_index]
  (It does not include the end_index)
- Another way is by using loc , iloc

```
car_sales["Make"]  #Prints the column having name make
```

```
0    Toyota
1     Honda
2    Toyota
3       BMW
4    Nissan
5    Toyota
6     Honda
7     Honda
8    Toyota
9       NaN
Name: Make, dtype: object
```

Indexing & Selecting Data

.loc and .iloc

.loc[]

- It is used to select rows by labels (index names).
- You can also select specific columns at the same time.

```
#.loc & .iloc
animals = pd.Series (["cat" , "Dog" , "bird" , "Panda" , "snake" ] , index = [0,3,9,8,3])
animals
```

```
animals.loc[3]
```

```
3      Dog
3    snake
dtype: object
```

.iloc[]

- .iloc[] is used to select rows and columns based on index positions (integer-based indexing).
- It follows zero-based indexing

```
animals.iloc[:3]
```

```
0     cat
3     Dog
9    bird
dtype: object
```

Data Cleaning & Preprocessing

Handling Missing Values

- Fill missing values fillna()
- Drop missing values dropna()

```
car_sales_missing["Odometer (KM)"].fillna(car_sales_missing["Odometer (KM)"].mean())
```

Removing Duplicates

- Identify duplicates duplicated()
- Remove duplicates drop_duplicates()

```
car_sales.duplicated().sum()  # Count duplicate rows
```

```
car_sales.drop_duplicates(inplace=True)
```

Data Cleaning & Preprocessing

Why Change Data Types?

- Ensures correct calculations
- Reduces memory usage
- Helps in better data analysis

Changing Data Types in Pandas

1] Numeric Data Conversions

2]Date & Time Conversions

3]Categorical Data Conversions

4]Boolean Data Conversions

```
# Ensure the 'Price' column is treated as strings first
car_sales['Price'] = car_sales['Price'].astype(str)

# Now you can use the .str.replace() method
car_sales['Price'] = car_sales['Price'].astype(int)
```

Filtering & Conditional Selection

Why Use Filtering & Conditional Selection?

- Extract relevant data based on conditions
- Helps in targeted analysis
- Enables better decision-making

To filter the data we can :

- Use comparison operators: >, <, ==, !=, etc.
- Use logical operators: & (AND), | (OR), ~ (NOT).
- Apply multiple conditions to refine data selection

```
car_sales[car_sales["Odometer (KM)"] >100000]
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
4	Nissan	White	213095.0	4.0	$3,500

Applying Functions to Data

1] Using .apply() for Custom Transformations

It allows user to apply diiferent function to dataFrames

2]Using lambda function for quick transformation

It is faster way to apply simpler function

Benefits of .apply() & lambda

- Flexible – Works on rows & columns
- Efficient – Reduces loops
- Powerful – Enables custom transformations

```
car_sales['Price'] = car_sales['Price'].apply(lambda x:f"${x}")
```

```
car_sales
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043.0	4.0	$$4,000
1	Honda	Red	87899.0	4.0	$$5,000
2	Toyota	Blue	NaN	3.0	$$7,000
3	BMW	Black	11179.0	5.0	$$22,000
4	Nissan	White	213095.0	4.0	$$3,500
5	Toyota	Green	NaN	4.0	$$4,500
6	Honda	NaN	NaN	4.0	$$7,500
7	Honda	Blue	NaN	NaN	$nan
8	Toyota	White	31600.0	NaN	$nan
9	NaN	White	31600.0	4.0	$$9,700

Sorting & Ordering Data

Why Sort Data?

- Organizes data for better analysis
- Makes it easier to identify trends
- Helps in ranking & comparisons

Syntax : df.sort_values("col_name")

It sorts values in ascending order by default

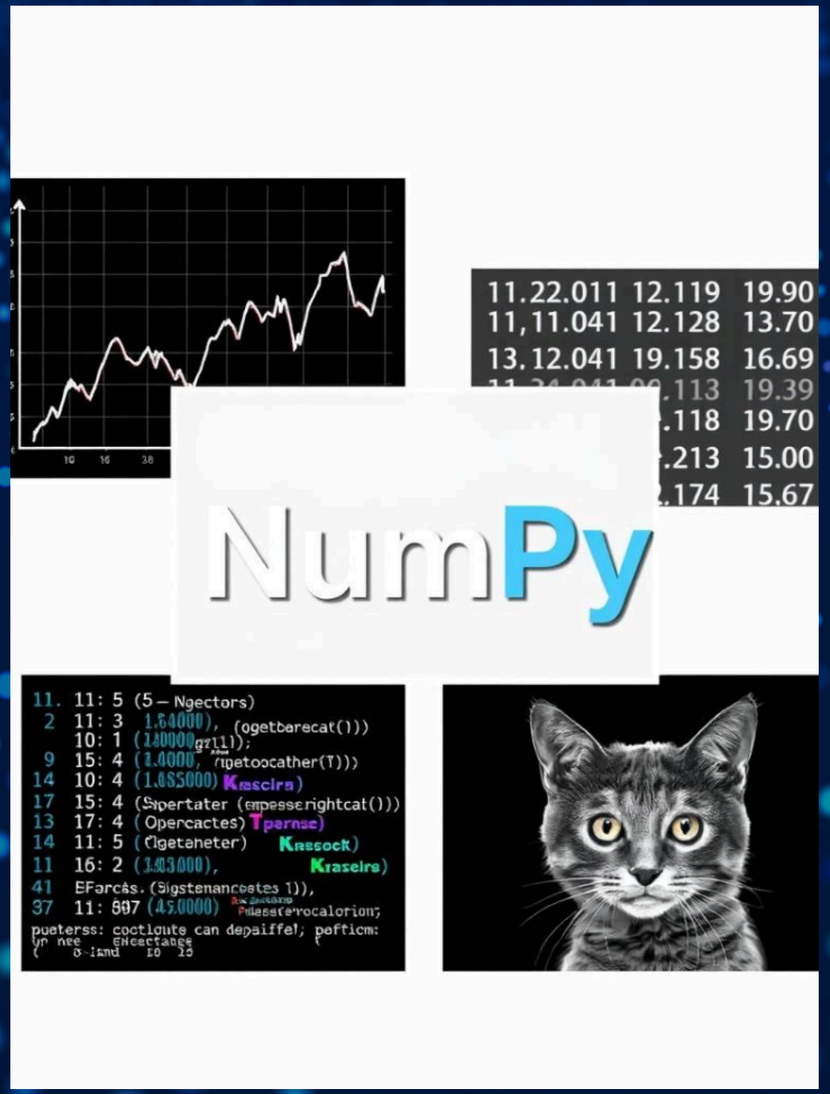
Benefits of Sorting

- Improves readability – Easier to find insights
- Enhances efficiency – Quick data access
- Works with .groupby() & filtering

NumPy

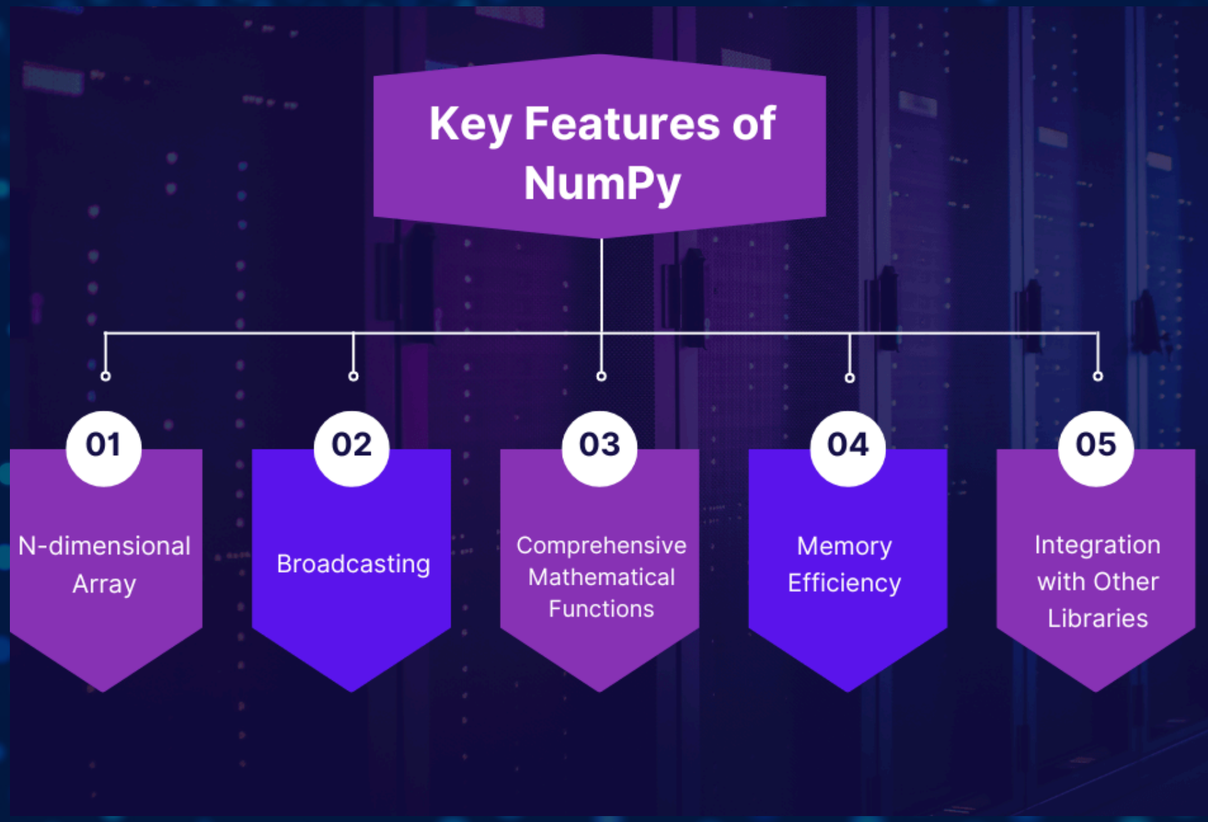
What Is NumPy

- NumPy (Numerical Python) is a library for numerical computing in Python.
- Developed to overcome the limitations of Python lists in terms of performance and memory usage.
- Core functionality: Efficient handling of large arrays and matrices, with a collection of mathematical functions.
- Provides a high-performance N-dimensional array object (ndarray) for data manipulation.



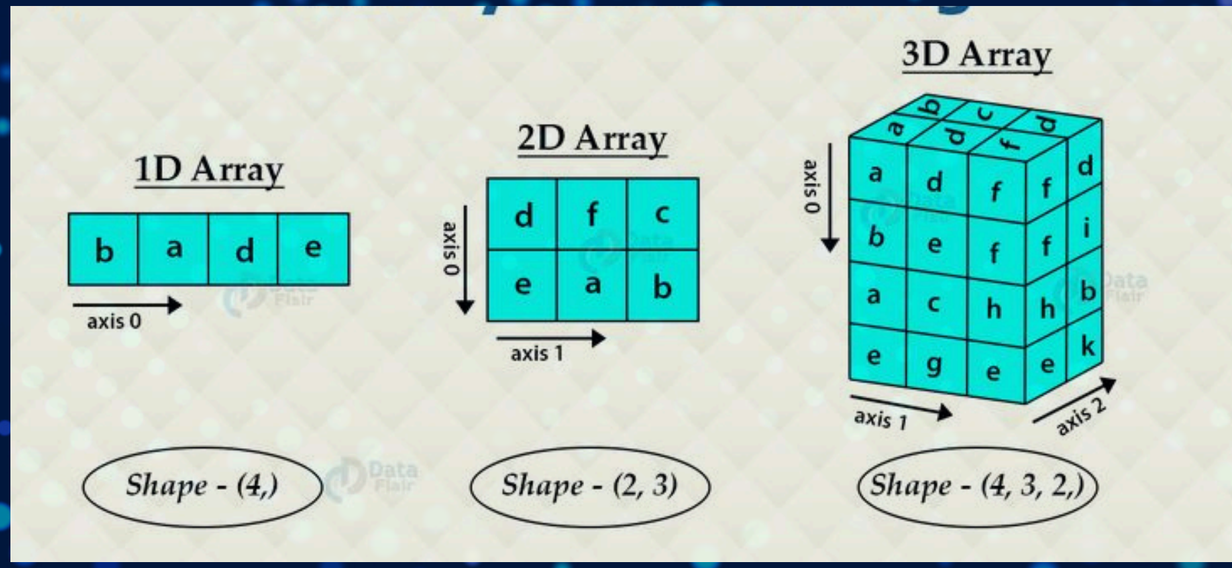
Key Features of NumPy

- Fast array operations due to its C++ based implementation.
- N-dimensional arrays (ndarrays) that can hold data of the same type.
- Efficient memory storage for large data.
- Mathematical operations: Built-in functions for scientific and mathematical computing (e.g., addition, mean, standard deviation, etc.)



Understanding the ndarray

- The ndarray is a multidimensional array (can be 1D, 2D, 3D, etc.).
- Homogeneous data types: All elements in the array must be of the same type.
- Provides an efficient way to store large datasets in memory.



Creating NumPy Arrays

- **Creating arrays from lists:** Converts a Python list into a NumPy array.
- **Creating arrays from ranges:** equivalent to range() in Python
- **Arrays filled with zeros:** Creates an array of a specific shape with all elements set to zero.
- **Arrays filled with ones:** Creates an array of a specific shape with all elements set to one.
- **Creating arrays with random numbers:** np.random.rand(2, 3)

NumPy

```
numpy.ones()
numpy.zeros()
numpy.array()
numpy.arange()
```

Create an Array

Indexing, Slicing, and Reshaping

- Indexing 1D arrays: arr[0] accesses the first element.
- Indexing 2D arrays: arr[0, 1] accesses the element at the first row and second column.
- Slicing: Extracting subsets of arrays, e.g., arr[1:3] (index 1 to 2).
- Reshaping arrays: Changing the shape of an array, e.g., arr.reshape((2, 2))

```
arr = np.array([1, 2, 3, 4, 5])
print(arr[0])  # Indexing

arr = np.array([[1, 2], [3, 4]])
print(arr[1, 1])  # Indexing 2D array

arr = np.array([1, 2, 3, 4, 5])
print(arr[1:4])  # Slicing

arr = np.array([1, 2, 3, 4, 5])
arr_reshaped = arr.reshape((5, 1))  # Reshaping
print(arr_reshaped)
```

Mathematical & Statistical Operations

- Arithmetic operations: arr1 + arr2, arr1 - arr2, arr * 2.
- Element-wise operations: np.sqrt(arr), np.exp(arr) for mathematical functions.
- Statistical operations: np.mean(arr), np.median(arr), np.std(arr) for mean, median, and standard deviation.
- Correlation: np.corrcoef(arr1, arr2) calculates correlation between two arrays.

```
arr = np.array([1, 2, 3, 4, 5])
print(np.mean(arr))  # Mean
print(np.median(arr))  # Median
print(np.std(arr))  # Standard deviation

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
print(np.corrcoef(arr1, arr2))  # Correlation
```

Matrix Operations & Linear Algebra

- Dot product: np.dot(arr1, arr2) calculates the dot product of two arrays.
- Matrix multiplication: arr1 @ arr2 (or np.matmul(arr1, arr2)) for matrix multiplication.
- Determinant: np.linalg.det(arr) calculates the determinant of a matrix.
- Inverse: np.linalg.inv(arr) computes the inverse of a matrix.
- Eigenvalues: np.linalg.eigvals(arr) computes the eigenvalues of a matrix.

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

print(np.dot(arr1, arr2))  # Dot product

print(np.linalg.det(arr1))  # Determinant
print(np.linalg.inv(arr1))  # Inverse
```

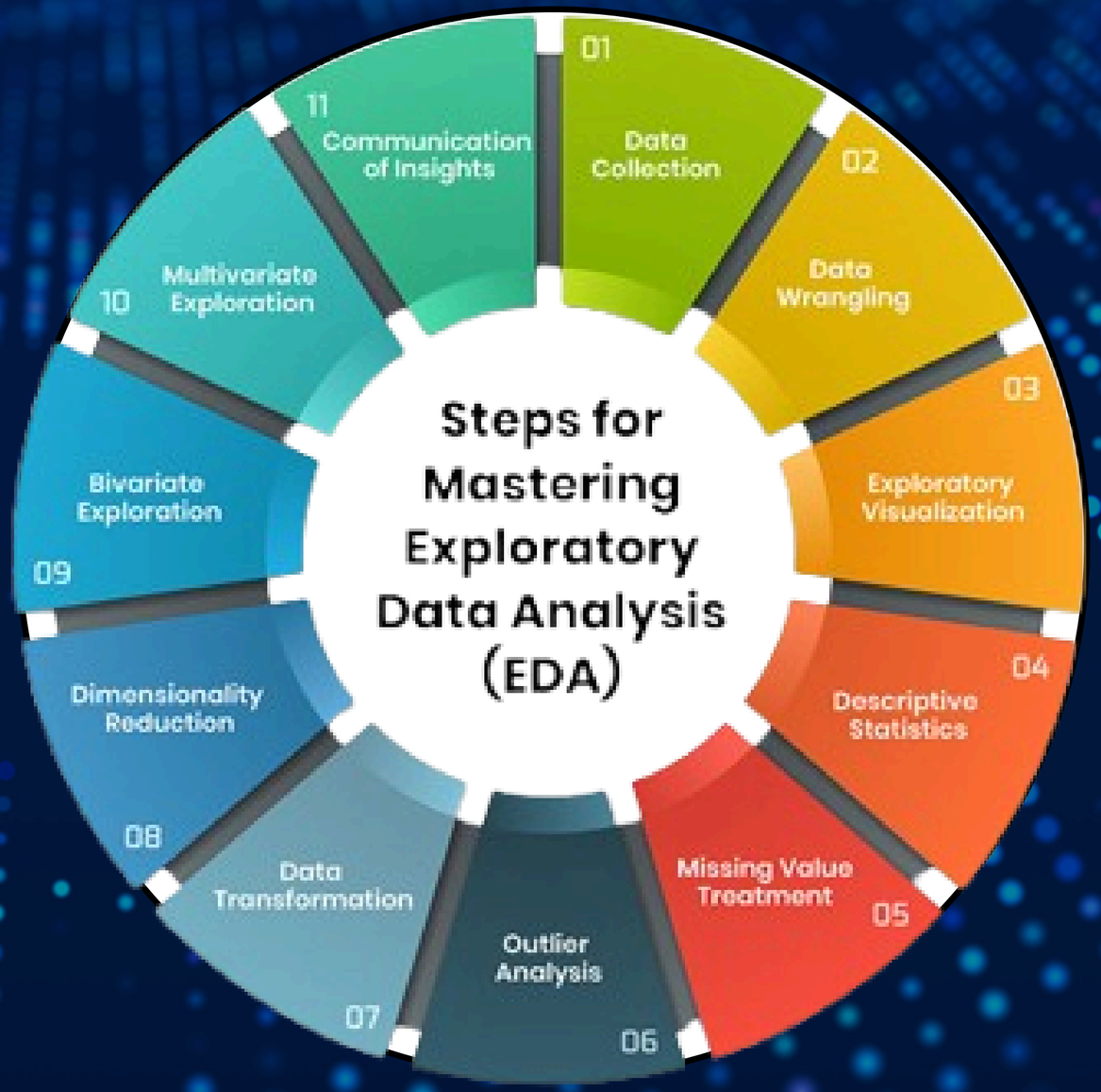
Broadcasting in NumPy

- Broadcasting allows NumPy to perform operations on arrays of different shapes.
- It automatically adjusts smaller arrays to match the shape of larger ones for element-wise operations.
- Smaller arrays are broadcast over larger arrays to make them compatible.
- Examples: Adding a scalar to an array, adding a 1D array to a 2D array

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([[1], [2], [3]])

# Broadcasting will add the 1D array t
result = arr1 + arr2
print(result)
```

EDA



Exploratory Data Analysis (EDA) is an important first step in data science projects. It involves looking at and visualizing data to understand its main features, find patterns, and discover how different parts of the data are connected.

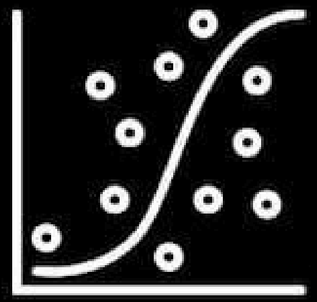
Types of Visualizations in EDA

Univariate analysis

When the data consists of only one type of variable, it is a univariate analysis. For example - Histograms.

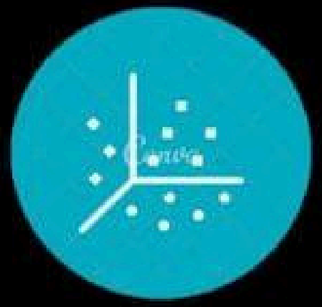
Bivariate analysis

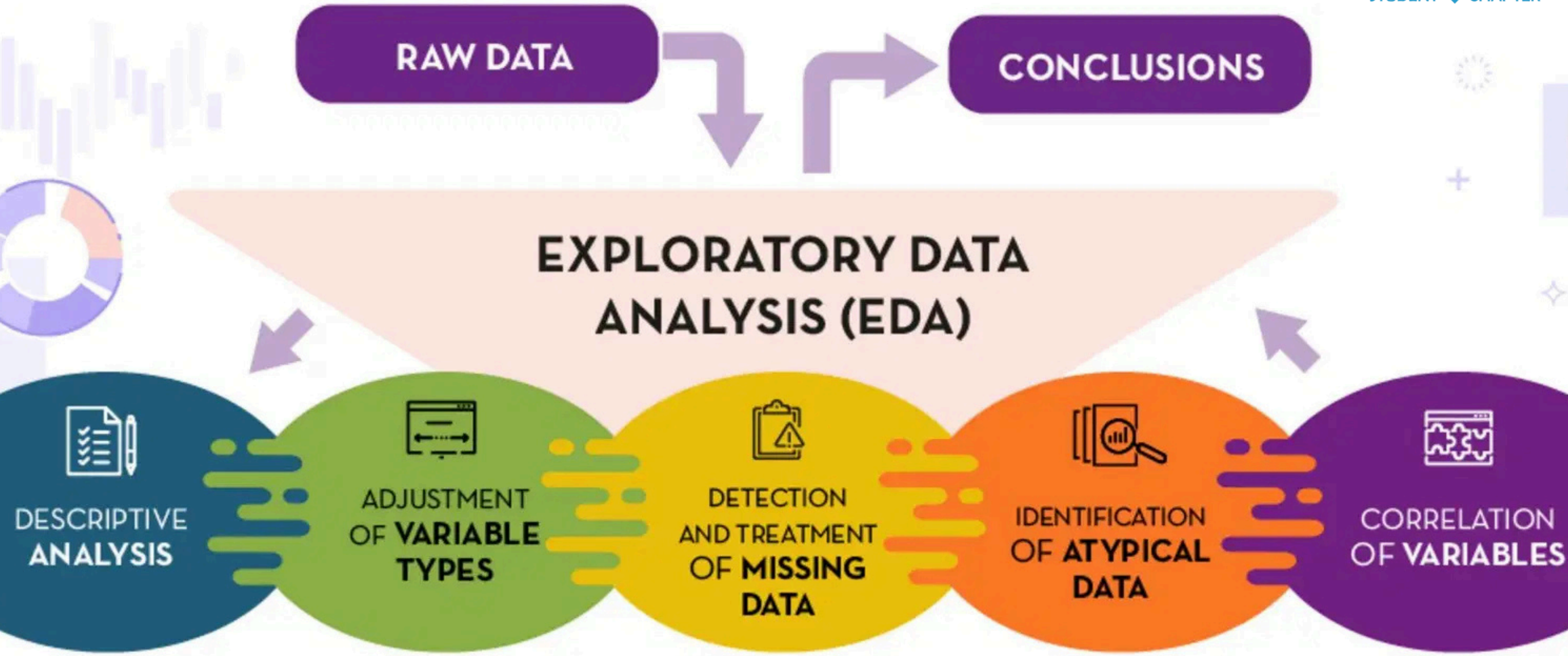
When the data consists of two types of variables, it is a by-variate analysis. For example - Scatter Plot.



Multivariate analysis

When the data consists of three or more types of variables, it is a multivariate analysis. For example - Cluster Analysis.




RAW DATA
CONCLUSIONS
EXPLORATORY DATA ANALYSIS (EDA)
DESCRIPTIVE ANALYSIS
ADJUSTMENT OF VARIABLE TYPES
DETECTION AND TREATMENT OF MISSING DATA
IDENTIFICATION OF ATYPICAL DATA
CORRELATION OF VARIABLES

Types of Descriptive Analysis

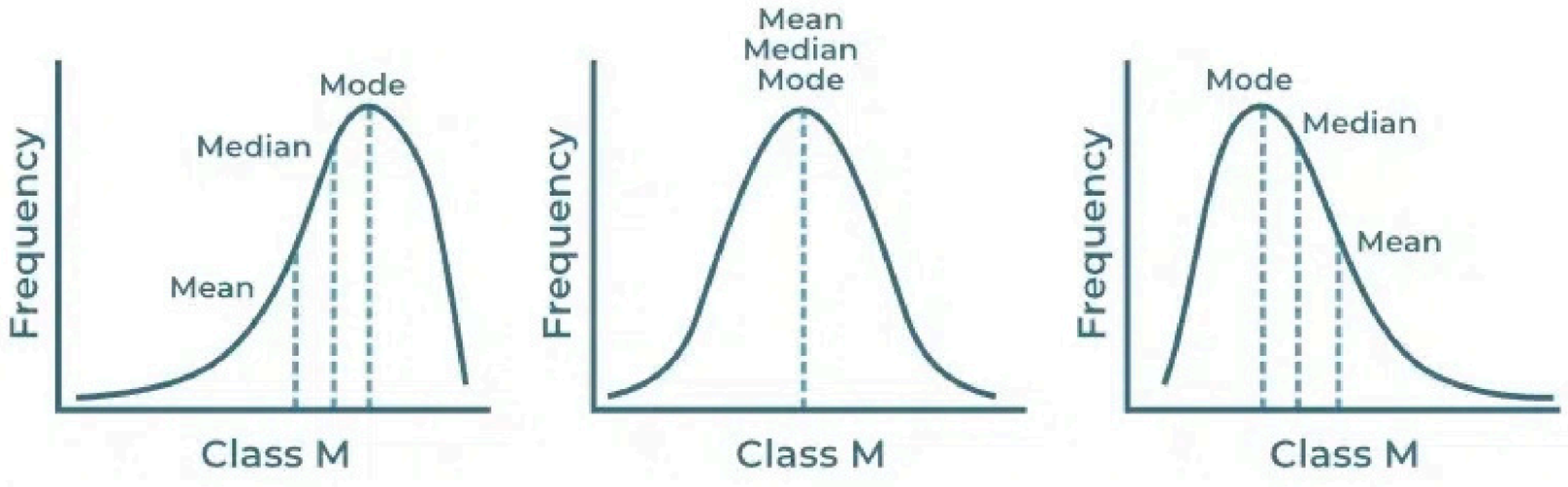
01 Measures of Frequency

02 Central Tendency

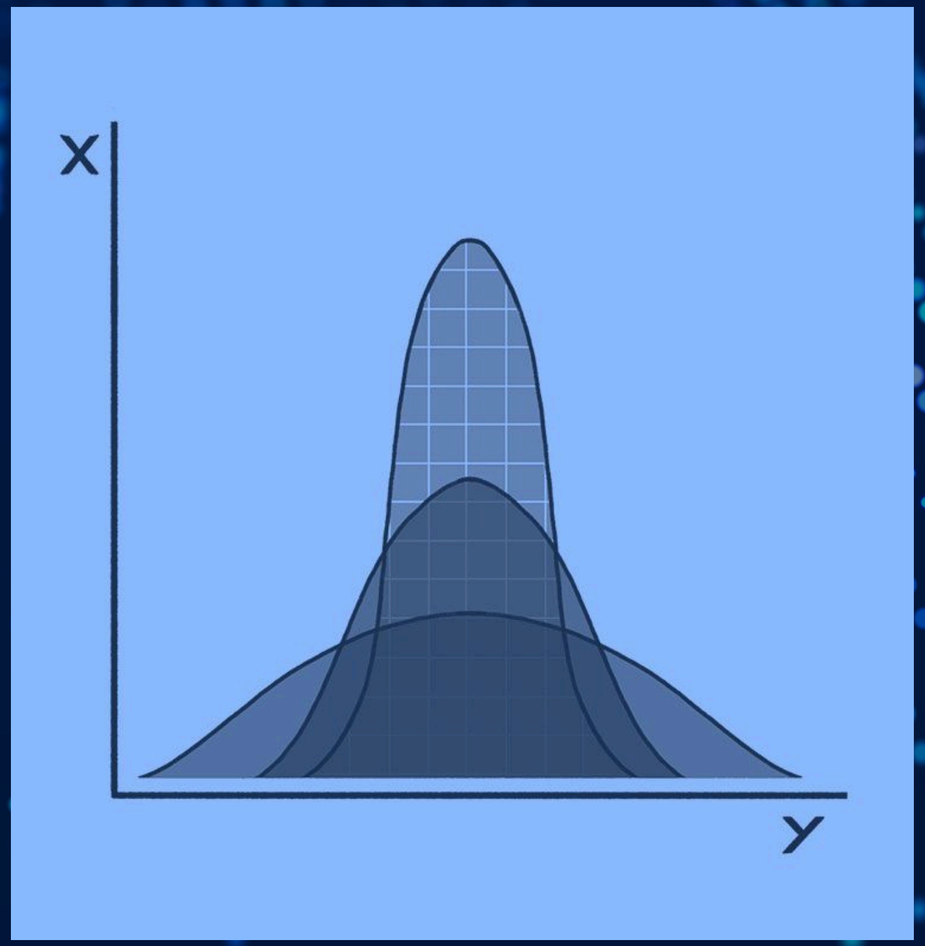
03 Measure of Dispersion

04 Variability

Measures Of Central Tendency



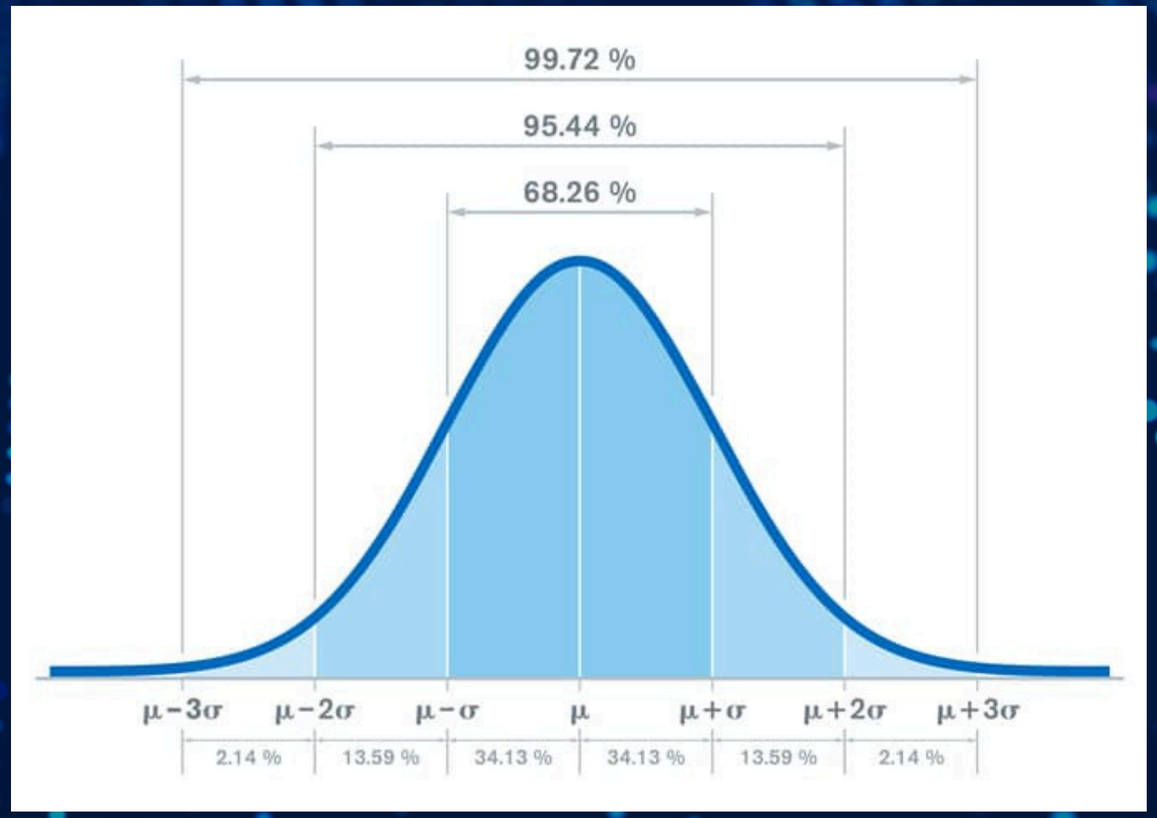
Variance



Variance is defined as an average squared deviation from the mean.

$$\sigma^2 = \frac{\sum (xi - \bar{x})^2}{N}$$

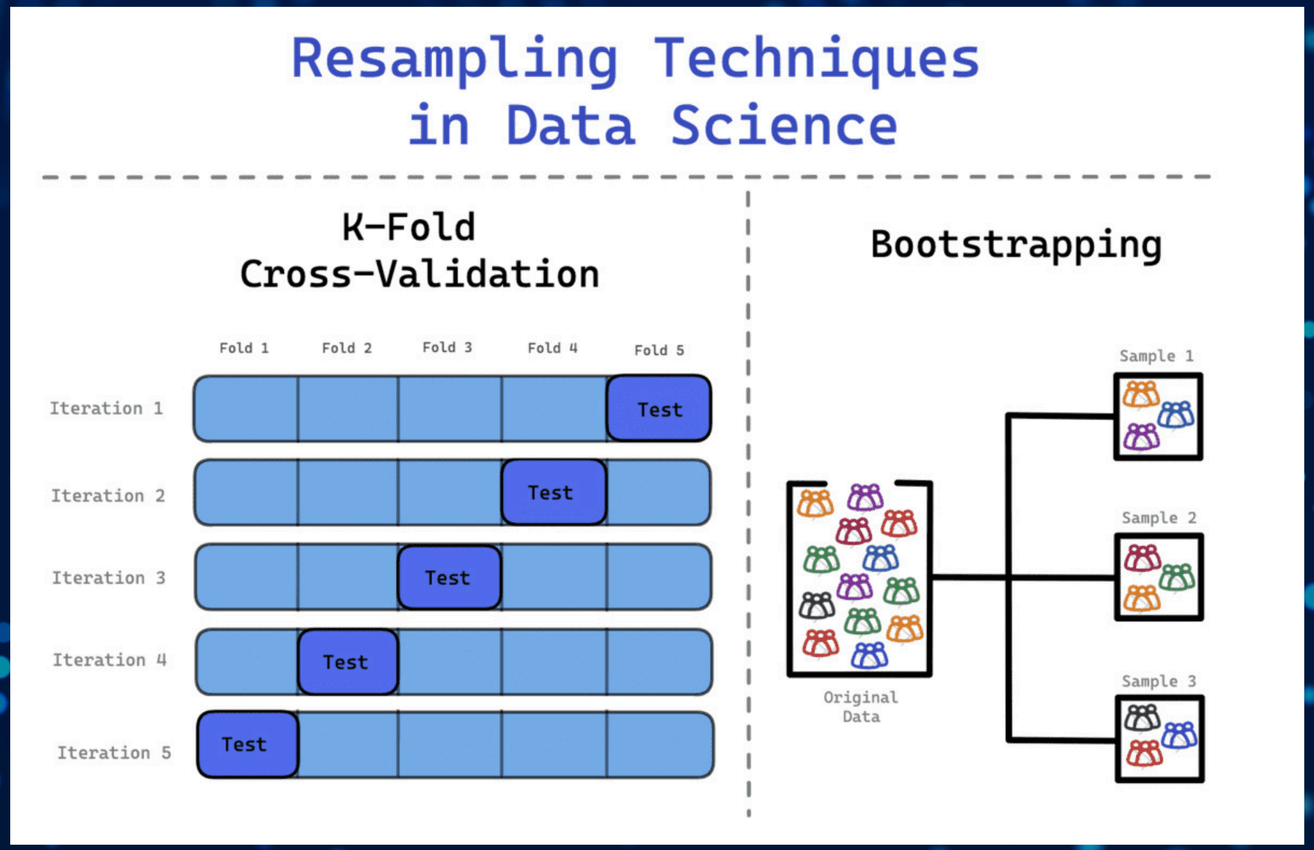
Standard Deviation



Standard deviation is widely used to measure the extent of variation or dispersion in data.

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

Resampling



Resampling is changing the frequency of time intervals and time normalization standardizing data lengths for easier comparison.

Cross-Validation is used to estimate the test error associated with a model to evaluate its performance by dividing observations into k size

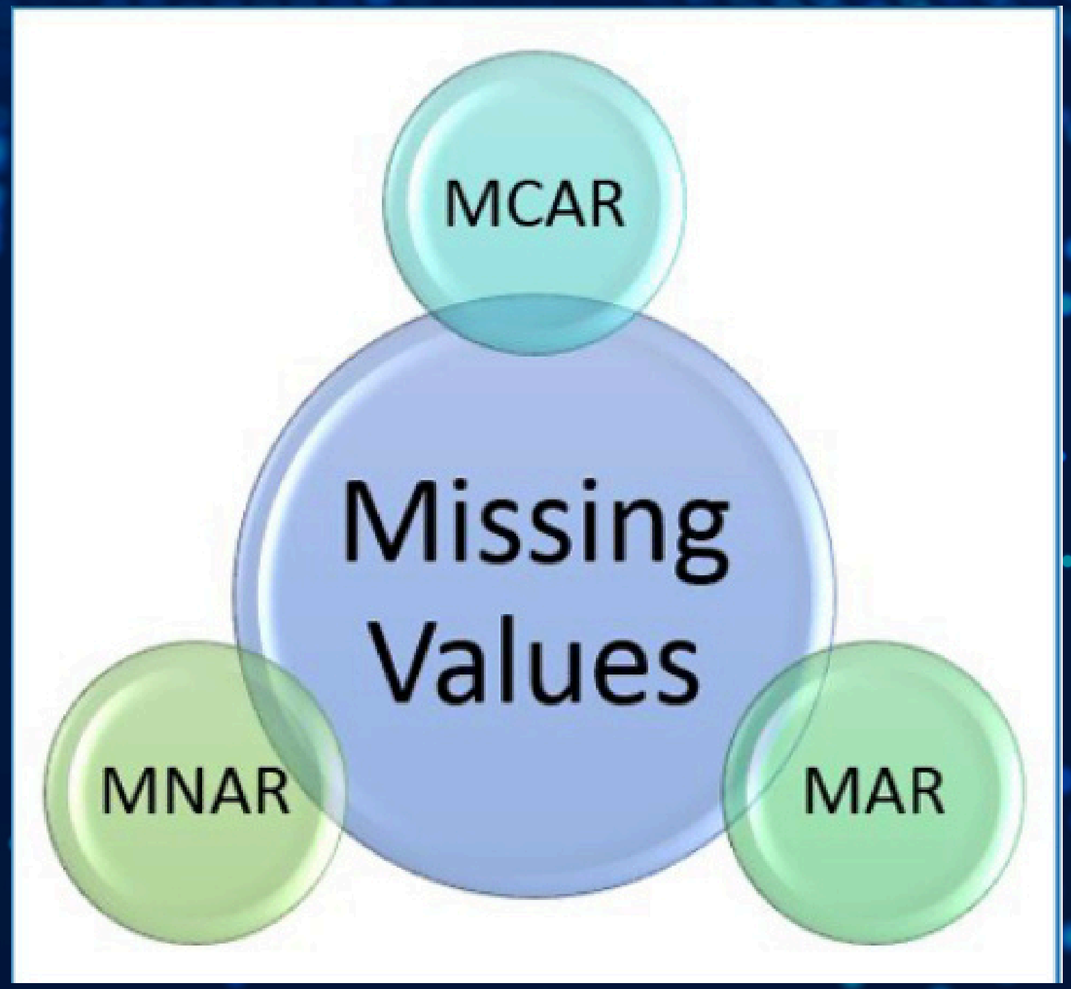
Bootstrap is when multiple subsets of the original training data are created by randomly sampling with replacement, which are then used to train individual decision trees

Missing Values

	School ID	Name	Address	City	Subject	Marks	Rank	Grade
0	101.0	Alice	123 Main St	Los Angeles	Math	85.0	2	B
1	102.0	Bob	456 Oak Ave	New York	English	92.0	1	A
2	103.0	Charlie	789 Pine Ln	Houston	Science	78.0	4	C
3	NaN	David	101 Elm St	Los Angeles	Math	89.0	3	B
4	105.0	Eva	NaN	Miami	History	NaN	8	D
5	106.0	Frank	222 Maple Rd	NaN	Math	95.0	1	A
6	107.0	Grace	444 Cedar Blvd	Houston	Science	80.0	5	C
7	108.0	Henry	555 Birch Dr	New York	English	88.0	3	B

Missing values are data points that are absent for a specific variable in a dataset. They can be represented in various ways, such as blank cells, null values, or special symbols like “NA” or “unknown.” These missing data points pose a significant challenge in data analysis and can lead to inaccurate or biased results.

Missing Values



1. Mean Imputation
2. Median Imputation
3. Mode Imputation
4. Interpolation
5. Last Observation Carried Forward
6. Dropping Missing Values

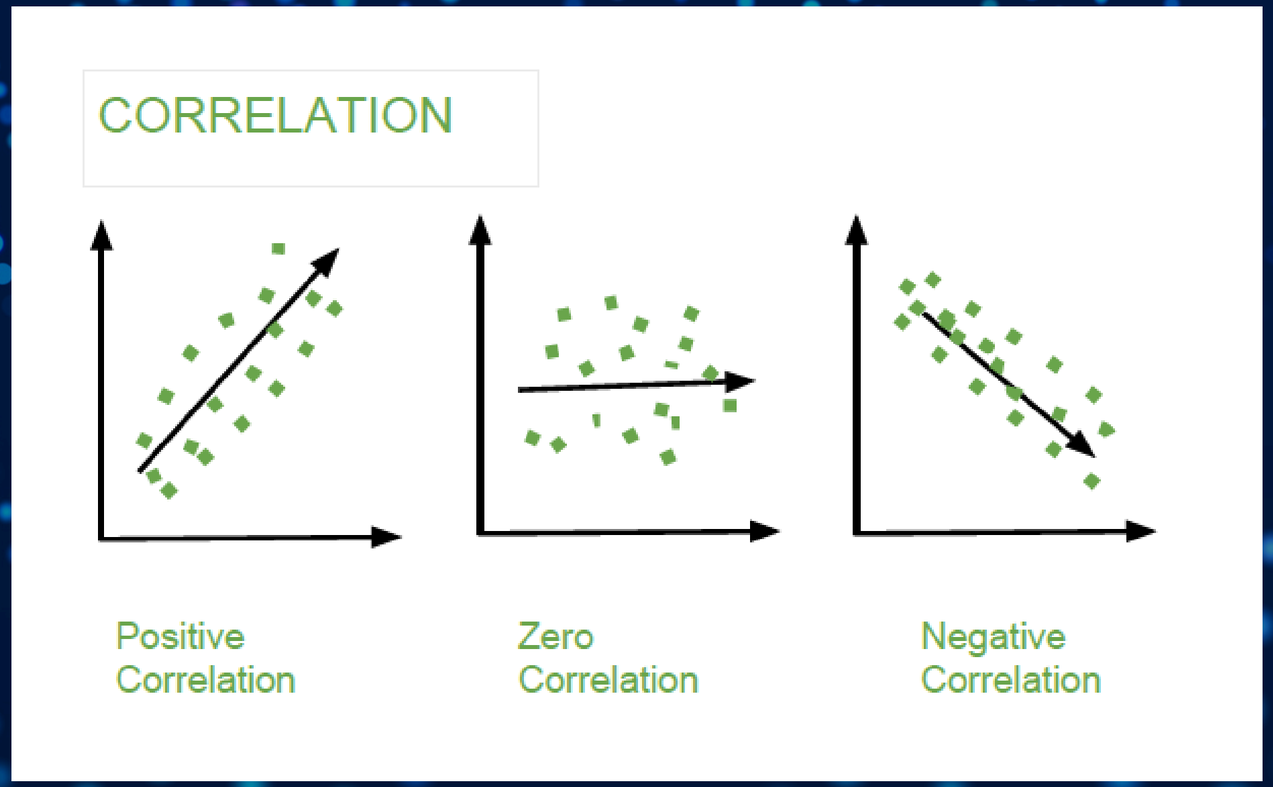
Atypical Values

Atypical data, also known as outliers, are observations that deviate significantly from the rest of the dataset.

$$Z = \frac{x - \mu}{\sigma}$$

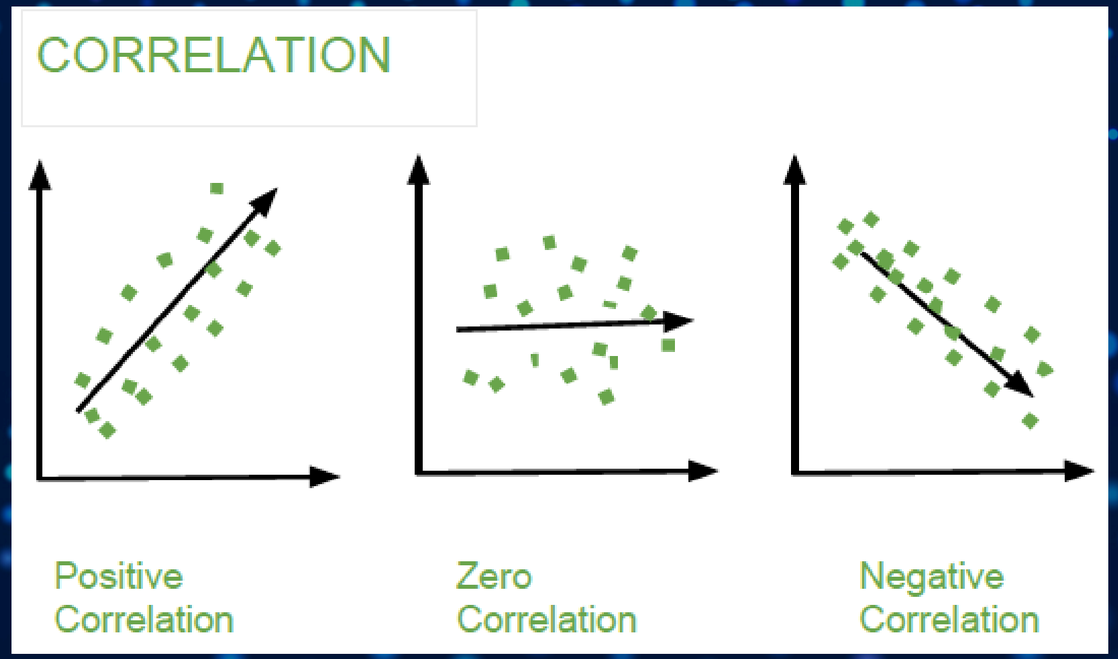
Correlation Analysis

Correlation analysis is a statistical technique for determining the strength of a link between two variables. It is used to detect patterns and trends in data and to forecast future occurrences.



Correlation Analysis

Correlation analysis is a statistical technique for determining the strength of a link between two variables. It is used to detect patterns and trends in data and to forecast future occurrences.



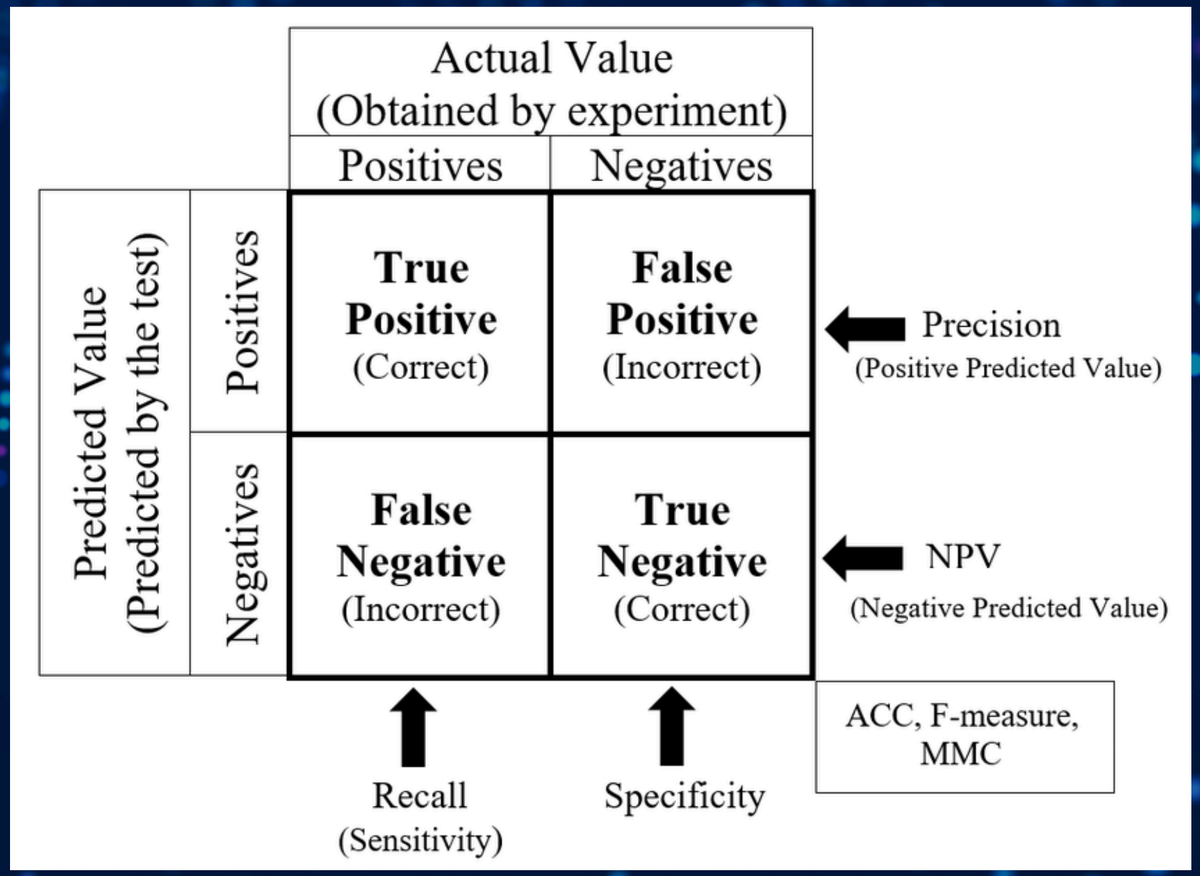
$$Corr(x, y) = \frac{\sum_{i=1}^{n}(x_i - x')\,(y_i - y')}{\sqrt{\sum_{i=1}^{n}(x_i - x')^2 \sum_{i=1}^{n}(y_i - y')^2}}$$

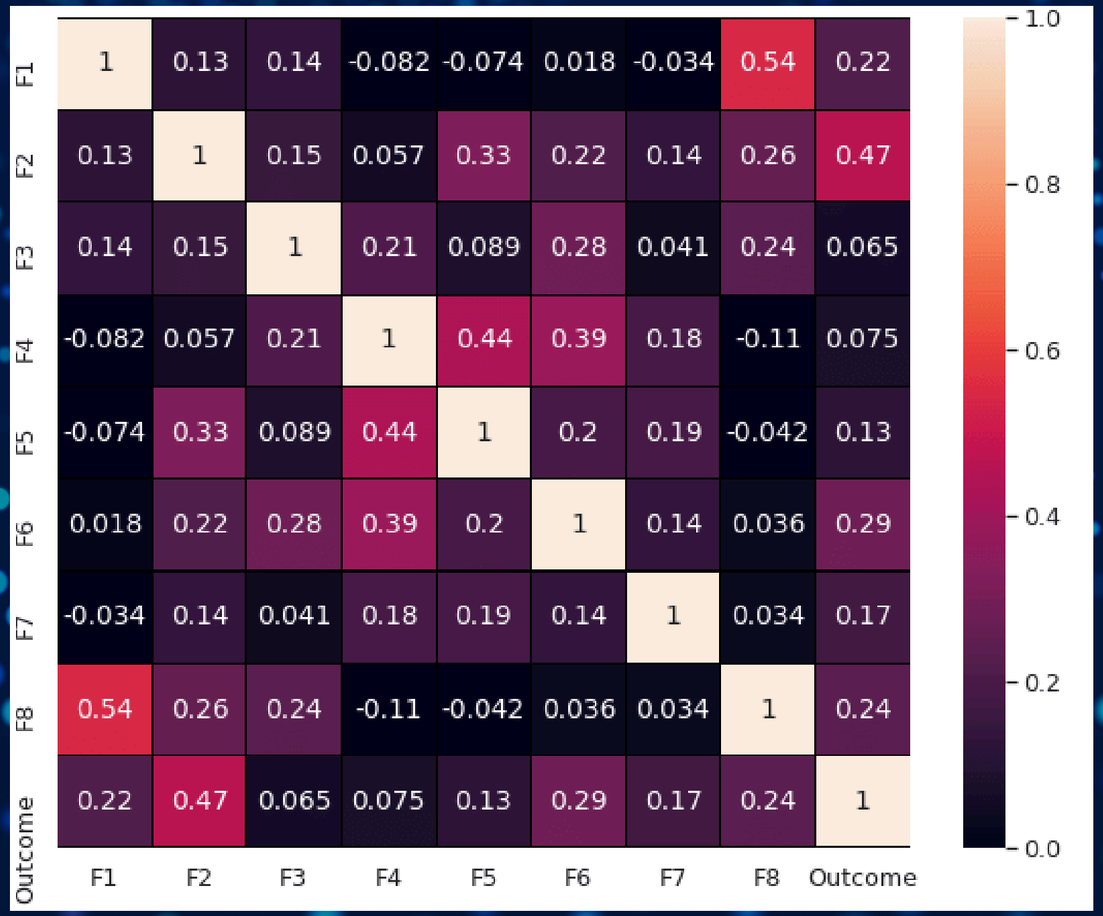
Covariance

Covariance is a statistical measure that indicates the direction of the linear relationship between two variables. It assesses how much two variables change together from their mean values.

$$Covri(x, y) = \frac{\sum_{i=1}^{n}(x_i - x')\,(y_i - y')}{n}$$

Confusion Matrix





THANK YOU