NARSIMHA REDDY ENGINEERING COLLEGE

An Autonomous Institution| Affiliated to JNTUH | Approved by AICTE
Accredited by NBA & NAAC with 'A' Grade

NRCM your roots to success. Estd.2007

Department of Computer Science and Engineering

Machine Learning and its applications (25CS405)

Prepared by
Dr.K.Anuradha
Dr.A.Jayanthi
Mrs.P.Revathy
Mr.Naresh

Unit – I (part 1)

Introduction to Machine Learning

MACHINE LEARNING - UNIT1

Introduction:

- Introduction to machine learning
- Supervised learning, Unsupervised learning, Semi-supervised learning
- Reinforcement learning
- Deep learning, Concept learning using find-S algorithm.

Feature Engineering:

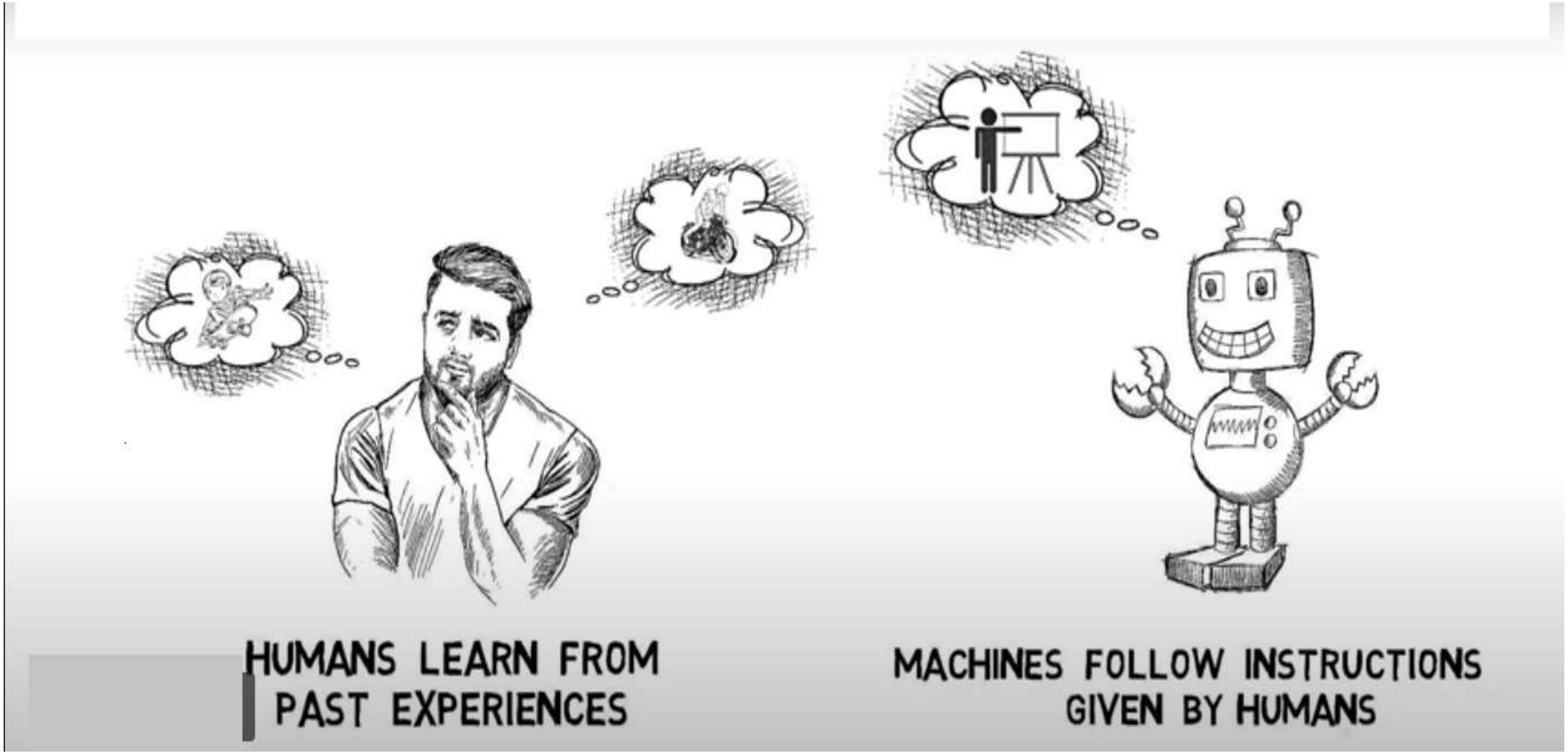
- Feature Selection using Filter, Wrapper, Embedded methods
- Feature normalization using min-max normalization,

  z-score normalization, and constant factor normalization

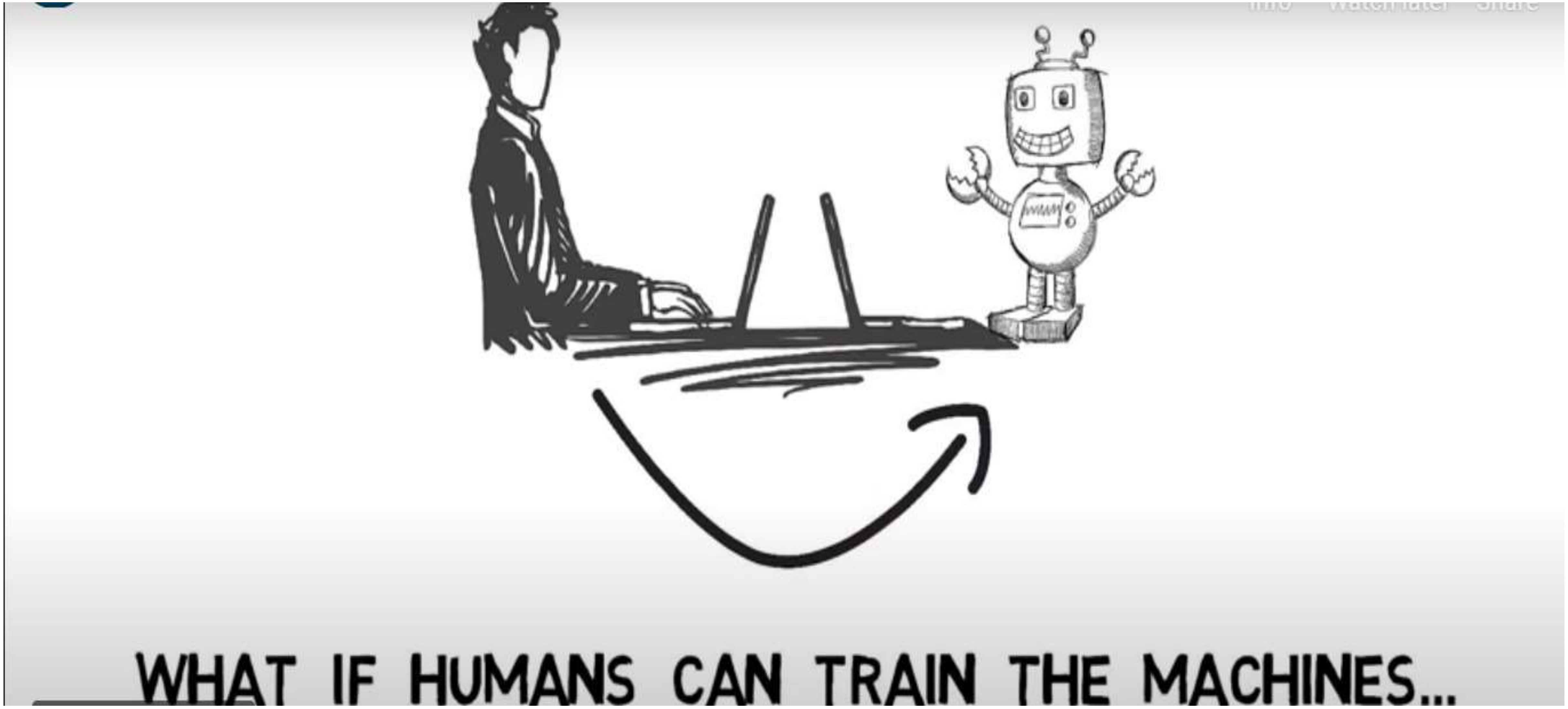
Introduction to Dimensionality Reduction:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA) techniques.

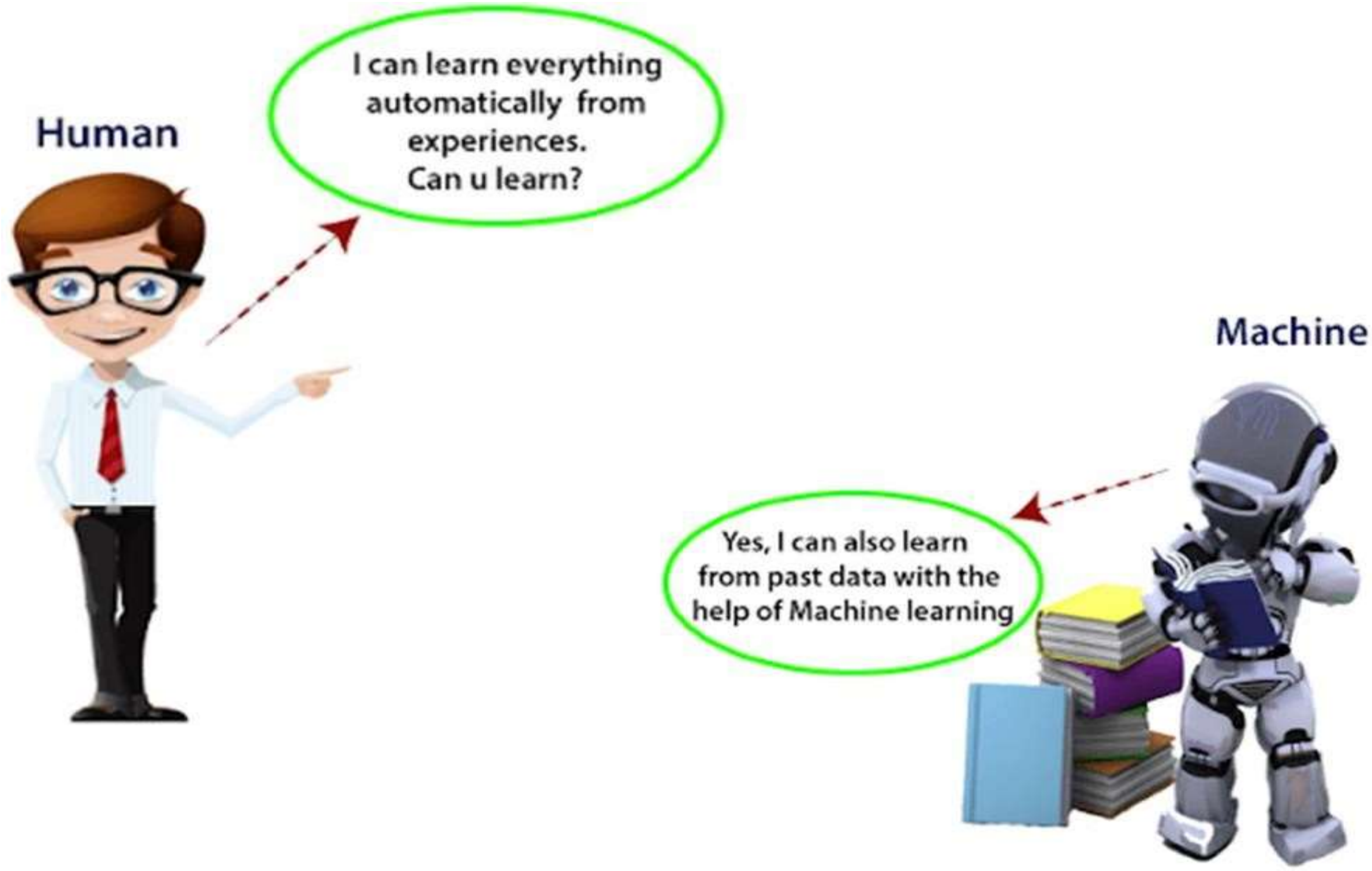
Machine Learning


HUMANS LEARN FROM PAST EXPERIENCES
MACHINES FOLLOW INSTRUCTIONS GIVEN BY HUMANS

Machine Learning



Machine Learning


Human
I can learn everything automatically from experiences. Can u learn?
Machine
Yes, I can also learn from past data with the help of Machine learning

Definition of Machine Learning

- **Definition by Tom Mitchell**
- It says that a computer program is said to learn from experience ‘E’ with respect to some class of tasks ‘T’ and performance measure ‘P’, if its performance at tasks in T as measured by P improves with experience E.
- Example: Hand writing Recognition problem

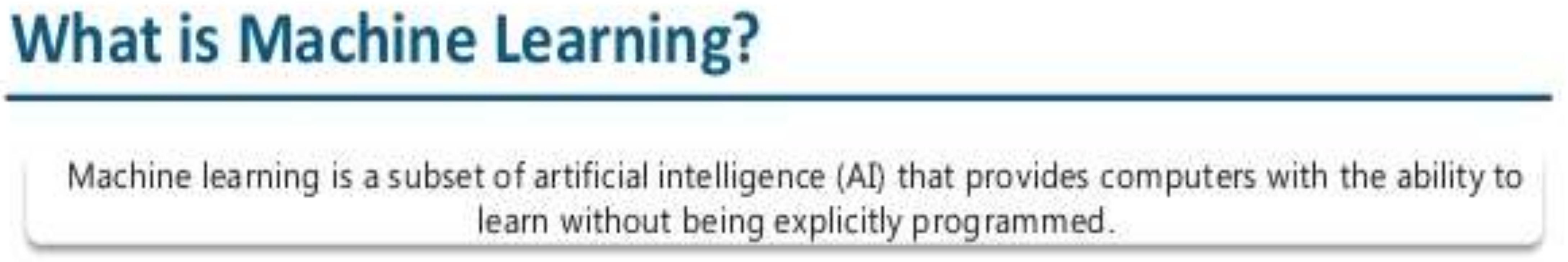
OR

- **Definition by Arthur Samuel**
- Machine learning is a "Field of study that gives computers the ability to learn without being explicitly programmed“
- In machine learning, algorithms are trained to find patterns and correlations in large data sets and to make the best decisions and predictions based on that analysis.

OR

- Machine Learning is the ability of systems to learn from data, identify patterns, and enact lessons from that data without human interaction or with minimal human interaction.
- Machine learning makes day-to-day and repetitive work much easier!

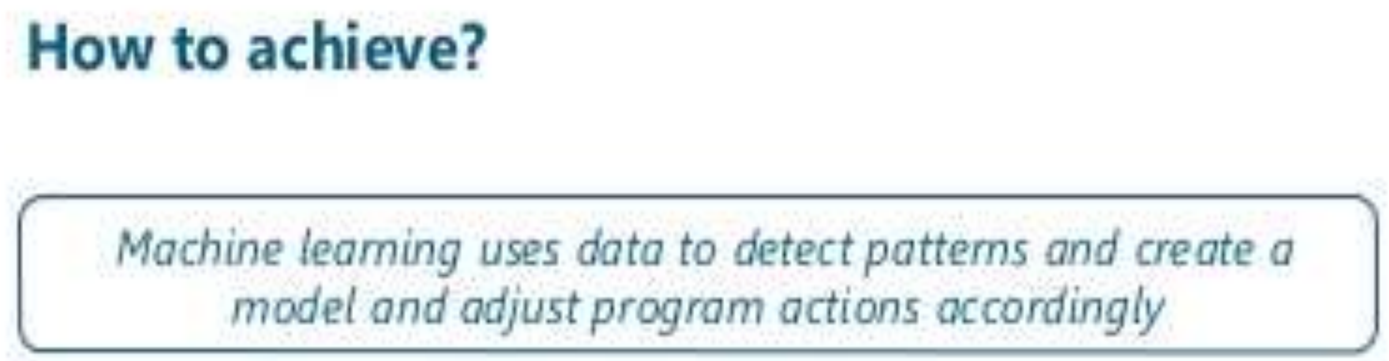
Machine Learning - Definition



What is Machine Learning?

Machine learning is a subset of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed.

The major focus of ML is to allow computer systems learn from experience without being explicitly programmed or human intervention.



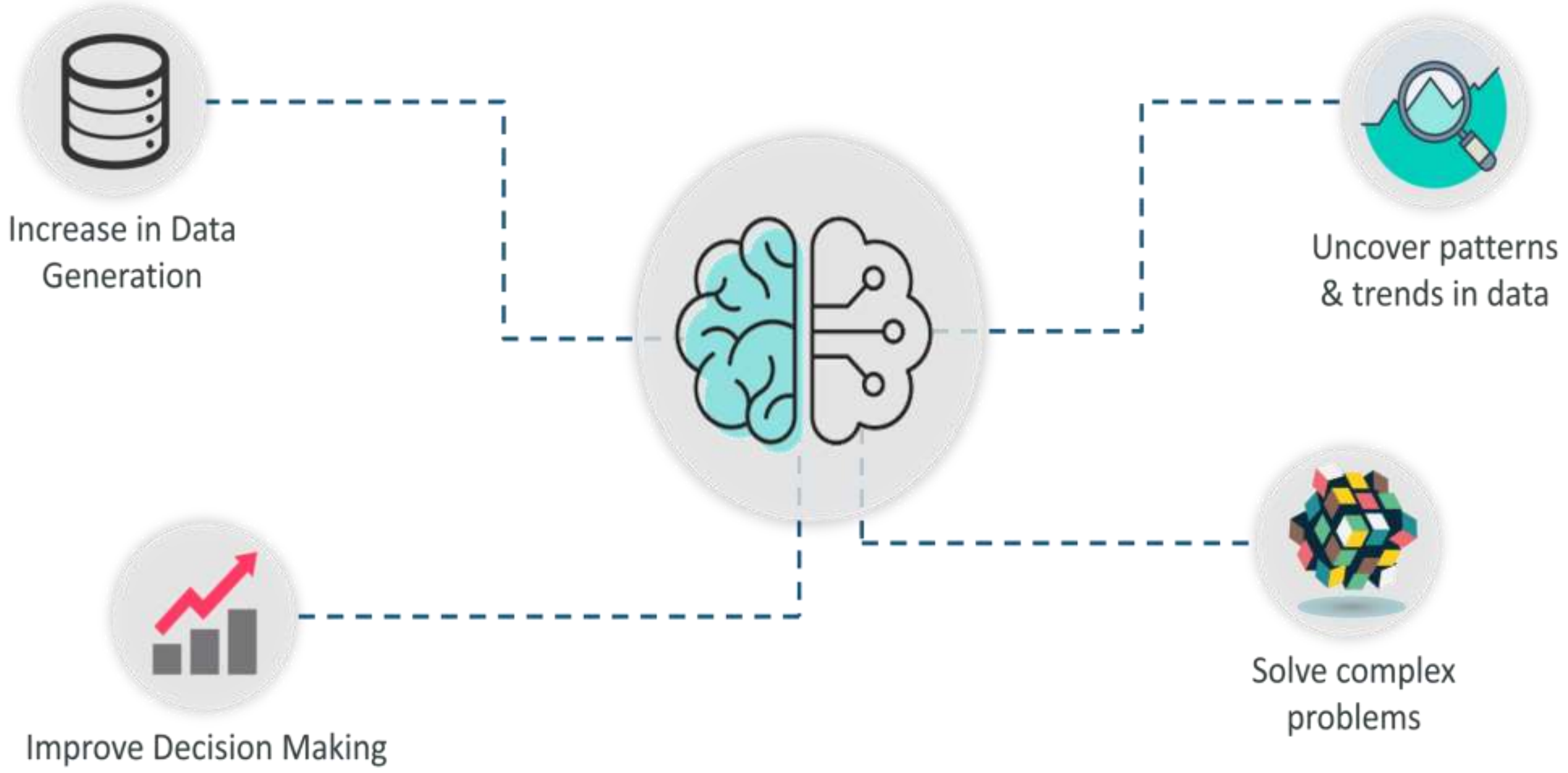
How to achieve?

Machine learning uses data to detect patterns and create a model and adjust program actions accordingly

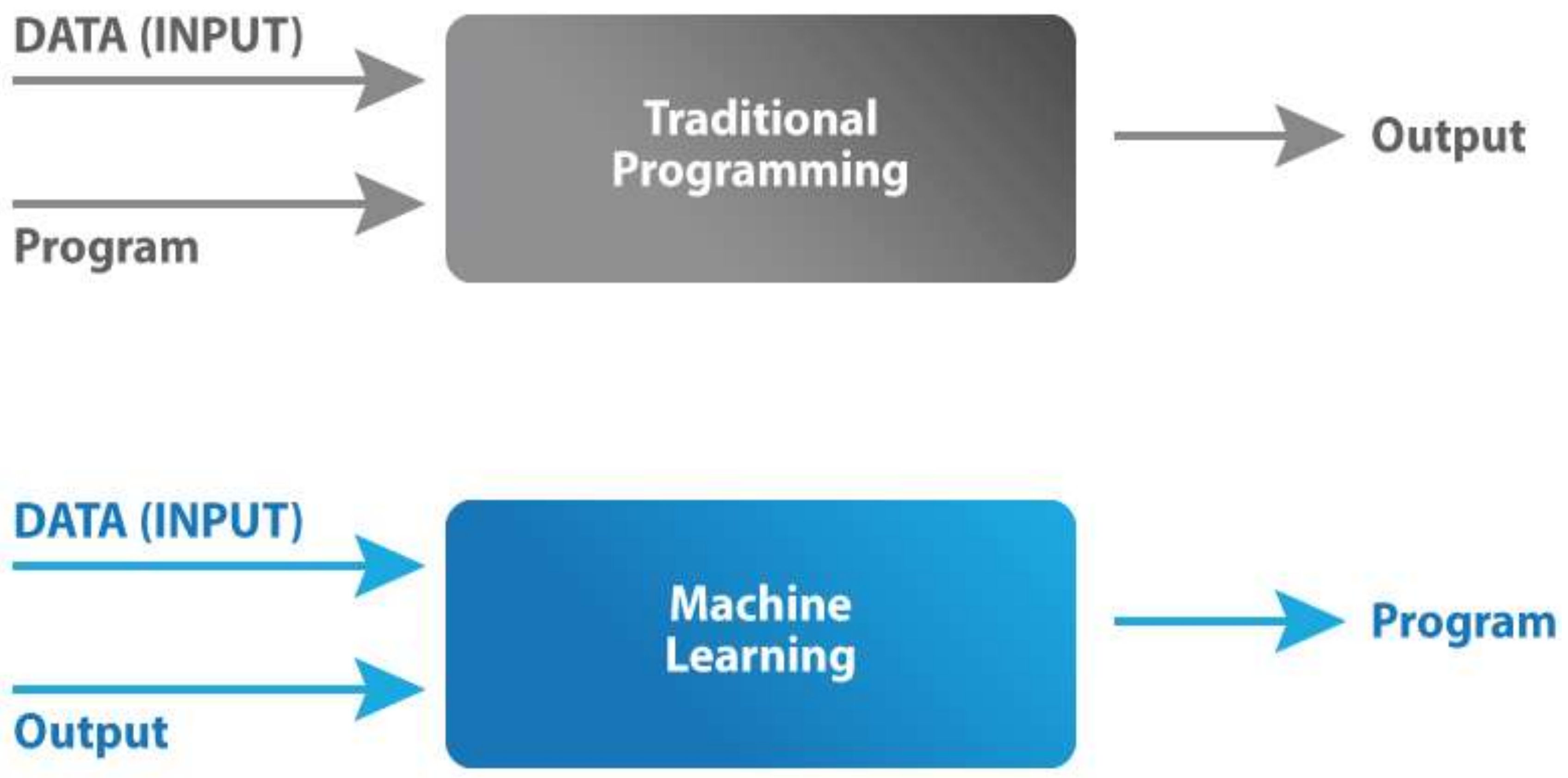
Why Machine Learning

Analysing huge sensor data and predicting the outcome e.g in Forecasting systems is not possible by manual calculations

Why Machine Learning


Increase in Data Generation
Uncover patterns & trends in data
Solve complex problems
Improve Decision Making

Traditional Programming Vs. Machine Learning



How is Machine Learning different from traditional programming?

Program

```
if (eyes == 2) & (legs == 4) & (tail ==  1)...  then
   print "Cat"
```

Input

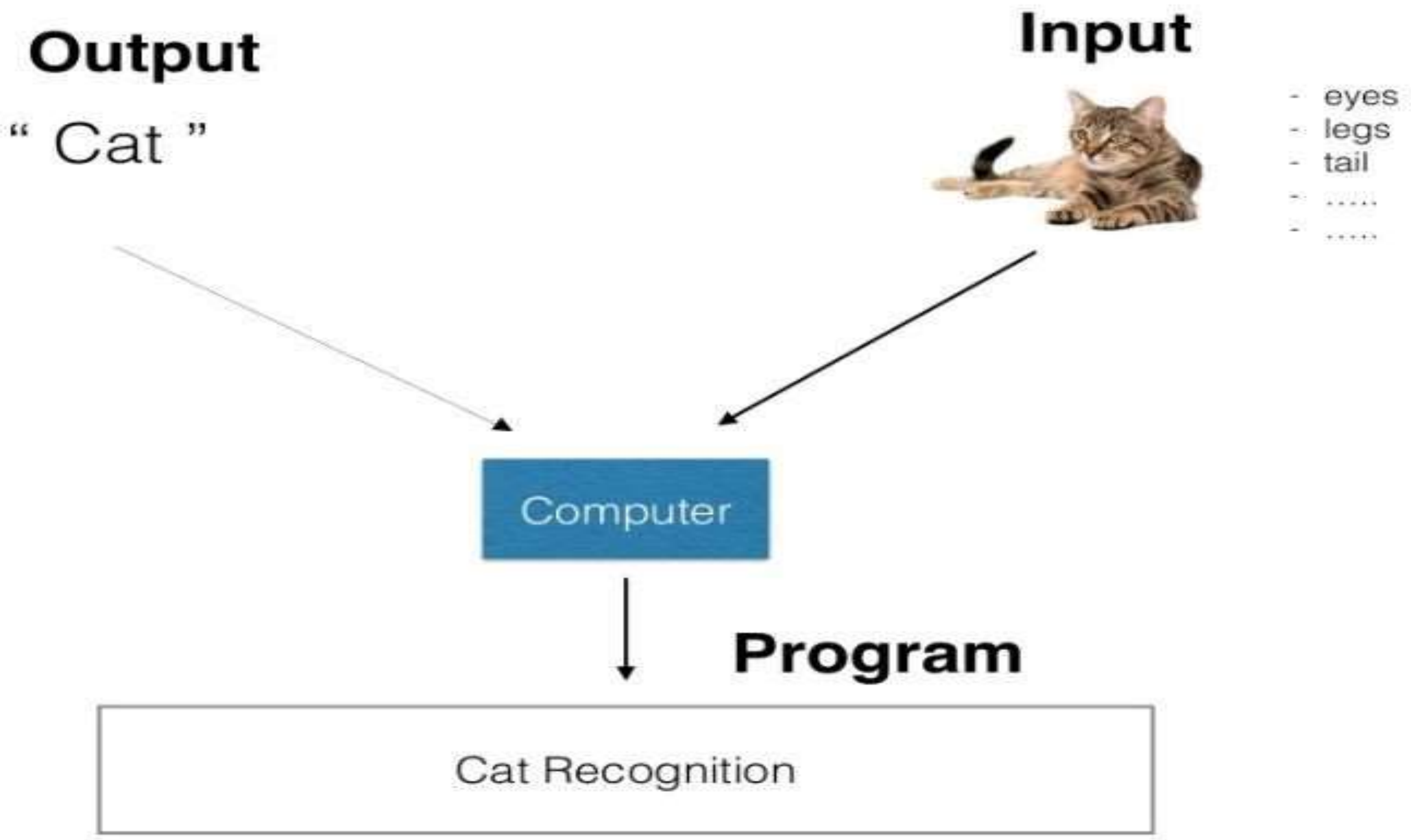
- eyes
- legs
- tail
- .....
- .....

Computer

" Cat " **Output**

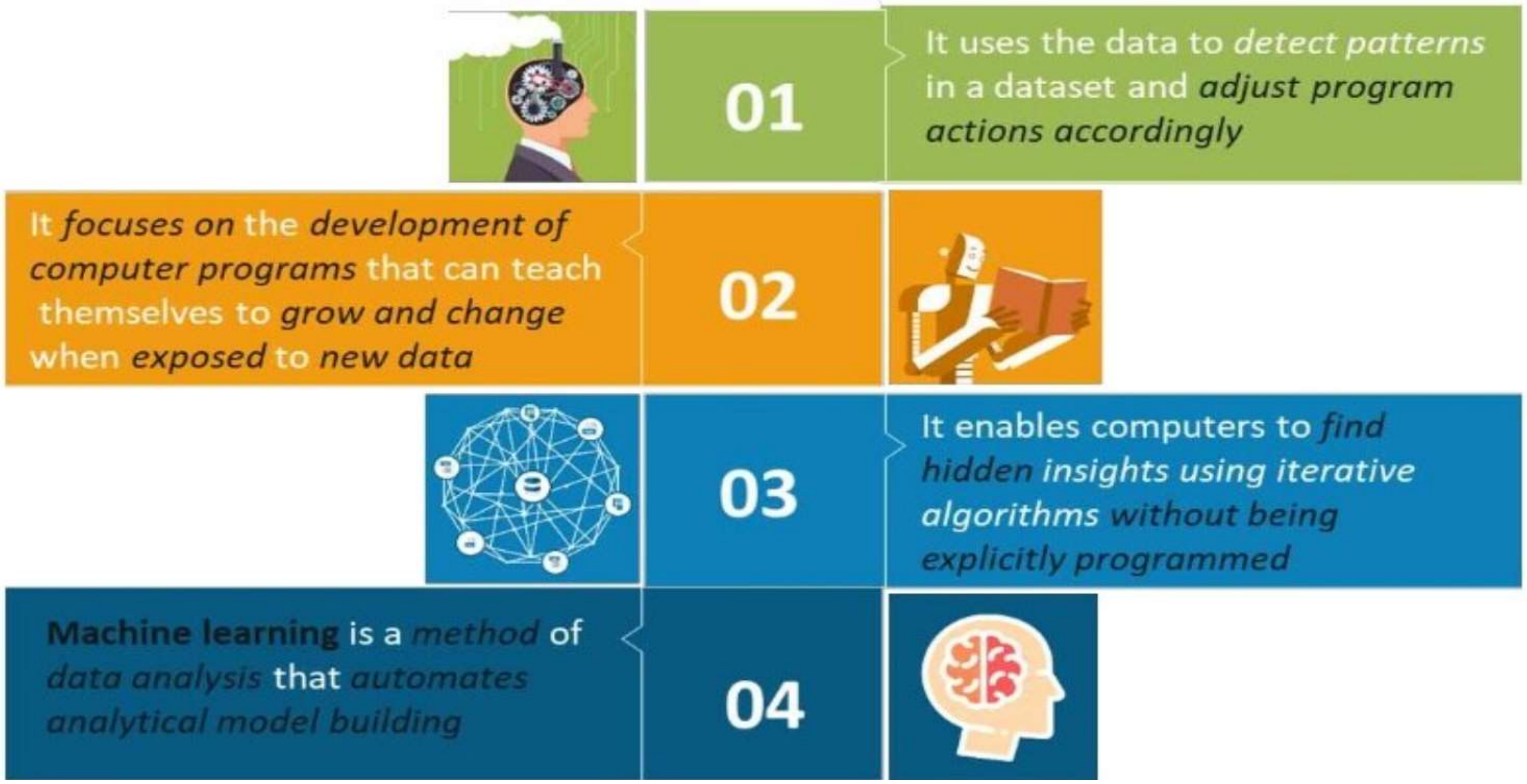
Traditional programming

How is Machine Learning different from traditional programming?



Machine Learning

Features of Machine Learning



01

It uses the data to *detect patterns* in a dataset and *adjust program actions accordingly*

02

It *focuses on* the *development of computer programs* that can teach themselves to *grow and change* when *exposed* to *new data*

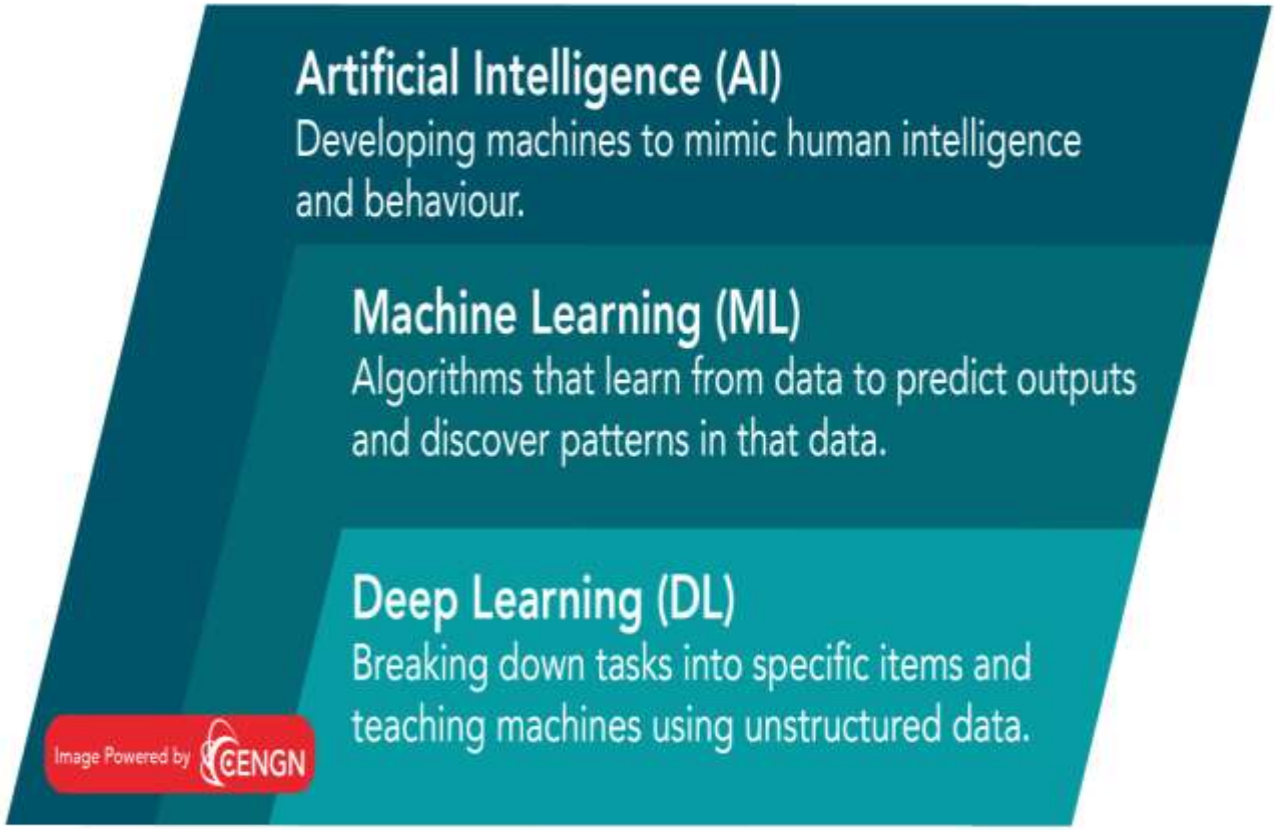
03

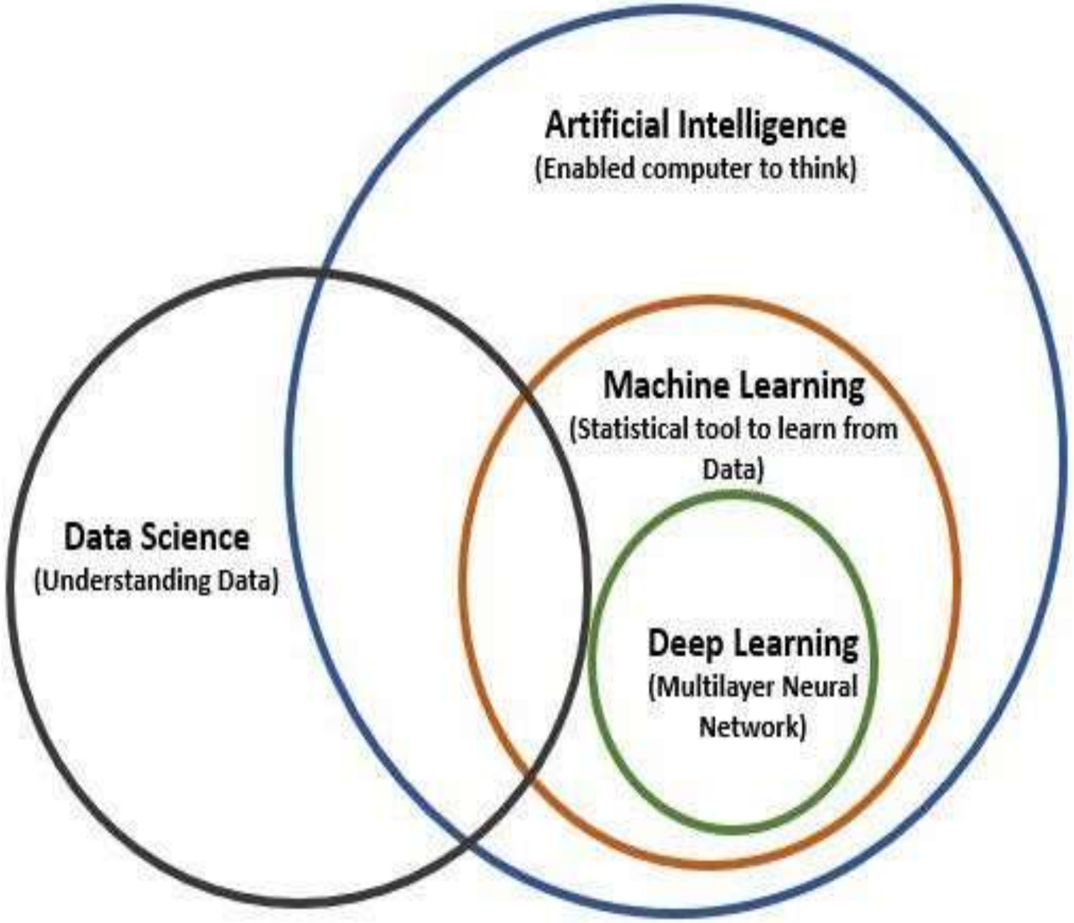
It enables computers to *find hidden insights using iterative algorithms* without being *explicitly programmed*

04

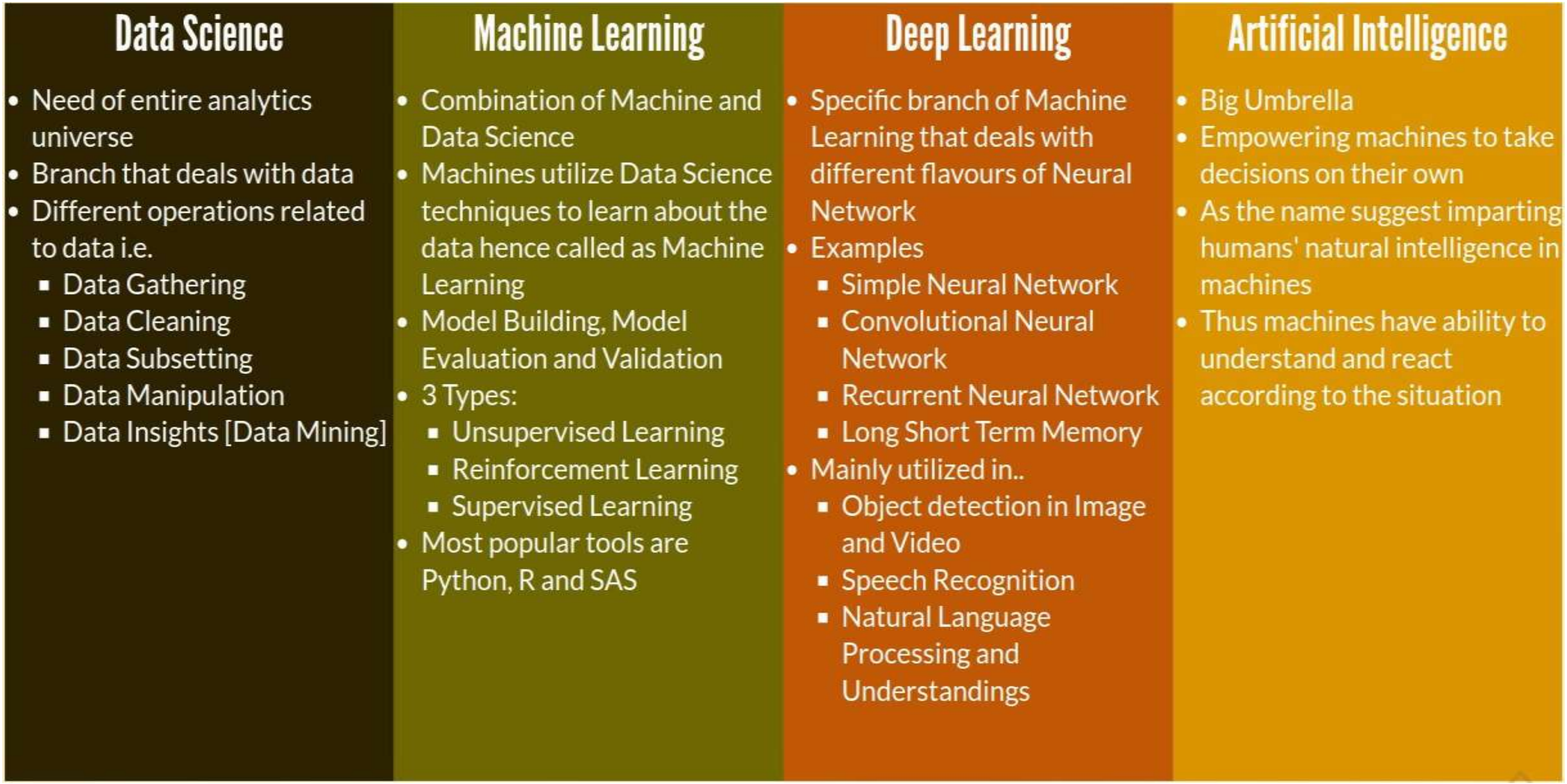
Machine learning is a *method* of *data analysis* that *automates analytical model building*

Relationship between AI, ML, DL and DS


Artificial Intelligence (AI)
Developing machines to mimic human intelligence and behaviour.
Machine Learning (ML)
Algorithms that learn from data to predict outputs and discover patterns in that data.
Deep Learning (DL)
Breaking down tasks into specific items and teaching machines using unstructured data.
Image Powered by CENGN


Artificial Intelligence
(Enabled computer to think)
Machine Learning
(Statistical tool to learn from Data)
Data Science
(Understanding Data)
Deep Learning
(Multilayer Neural Network)

All the technologies at a glance………



Data Science

- Need of entire analytics universe
- Branch that deals with data
- Different operations related to data i.e.
  - Data Gathering
  - Data Cleaning
  - Data Subsetting
  - Data Manipulation
  - Data Insights [Data Mining]

Machine Learning

- Combination of Machine and Data Science
- Machines utilize Data Science techniques to learn about the data hence called as Machine Learning
- Model Building, Model Evaluation and Validation
- 3 Types:
  - Unsupervised Learning
  - Reinforcement Learning
  - Supervised Learning
- Most popular tools are Python, R and SAS

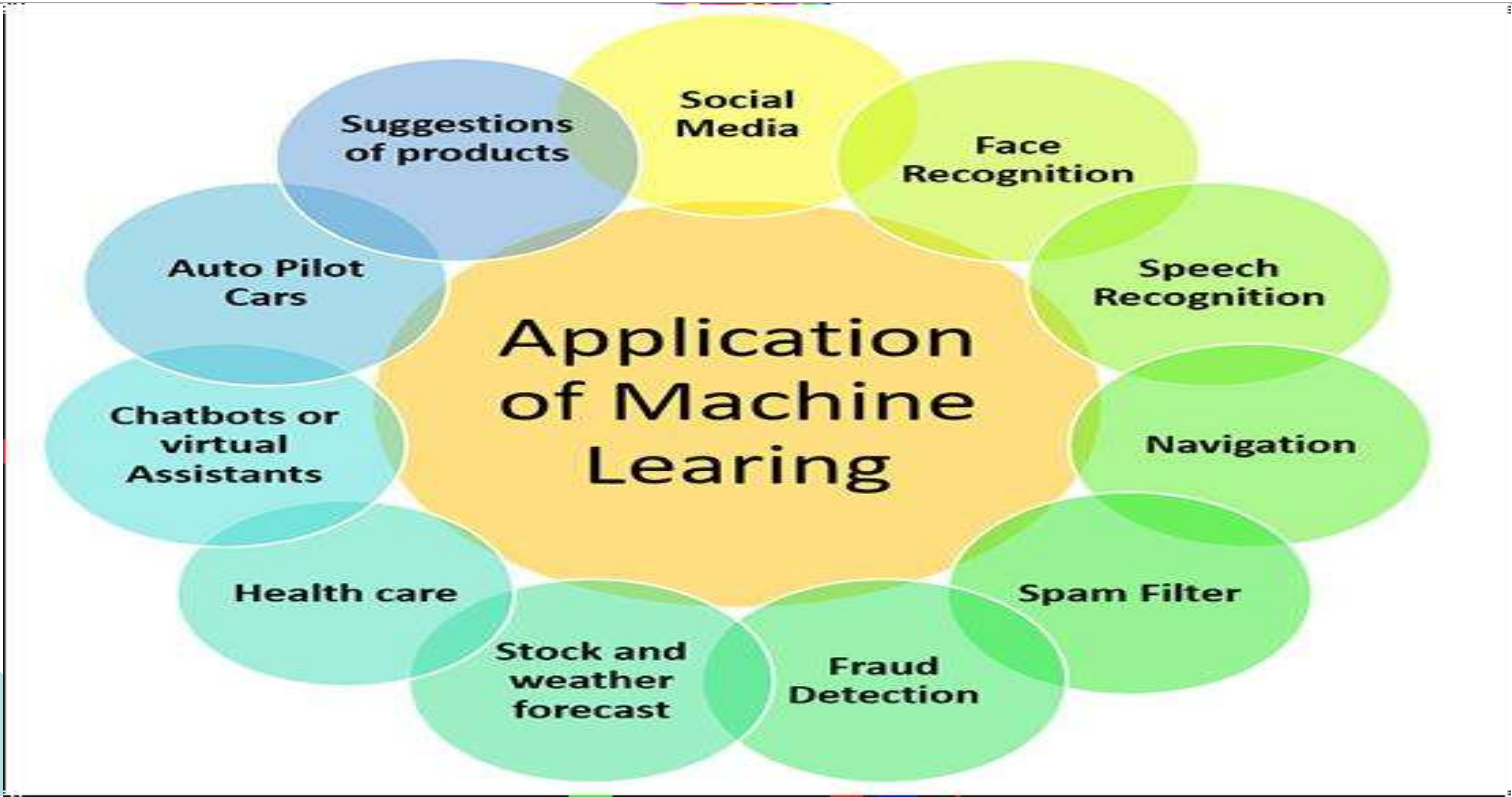
Deep Learning

- Specific branch of Machine Learning that deals with different flavours of Neural Network
- Examples
  - Simple Neural Network
  - Convolutional Neural Network
  - Recurrent Neural Network
  - Long Short Term Memory
- Mainly utilized in..
  - Object detection in Image and Video
  - Speech Recognition
  - Natural Language Processing and Understandings

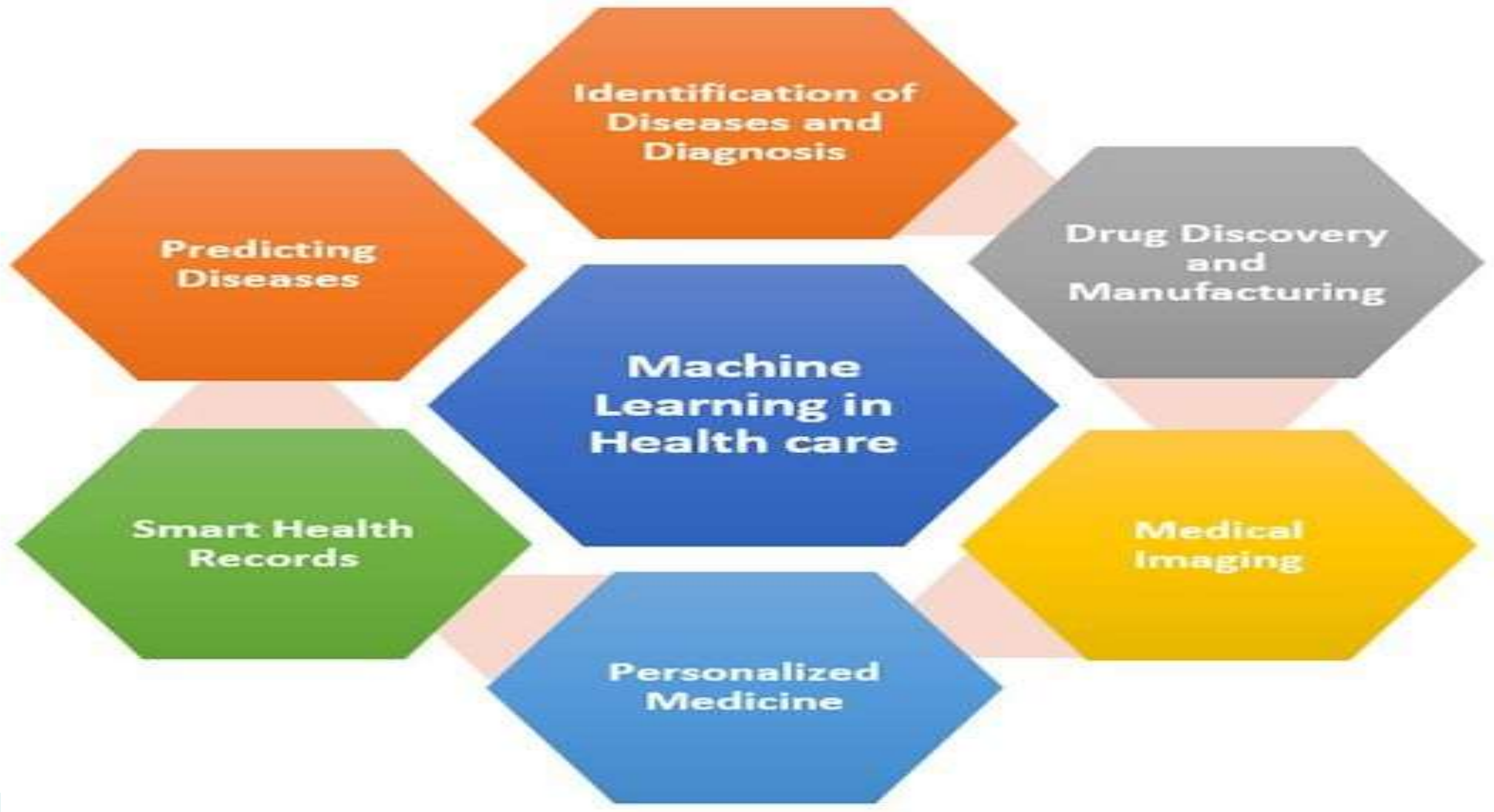
Artificial Intelligence

- Big Umbrella
- Empowering machines to take decisions on their own
- As the name suggest imparting humans' natural intelligence in machines
- Thus machines have ability to understand and react according to the situation

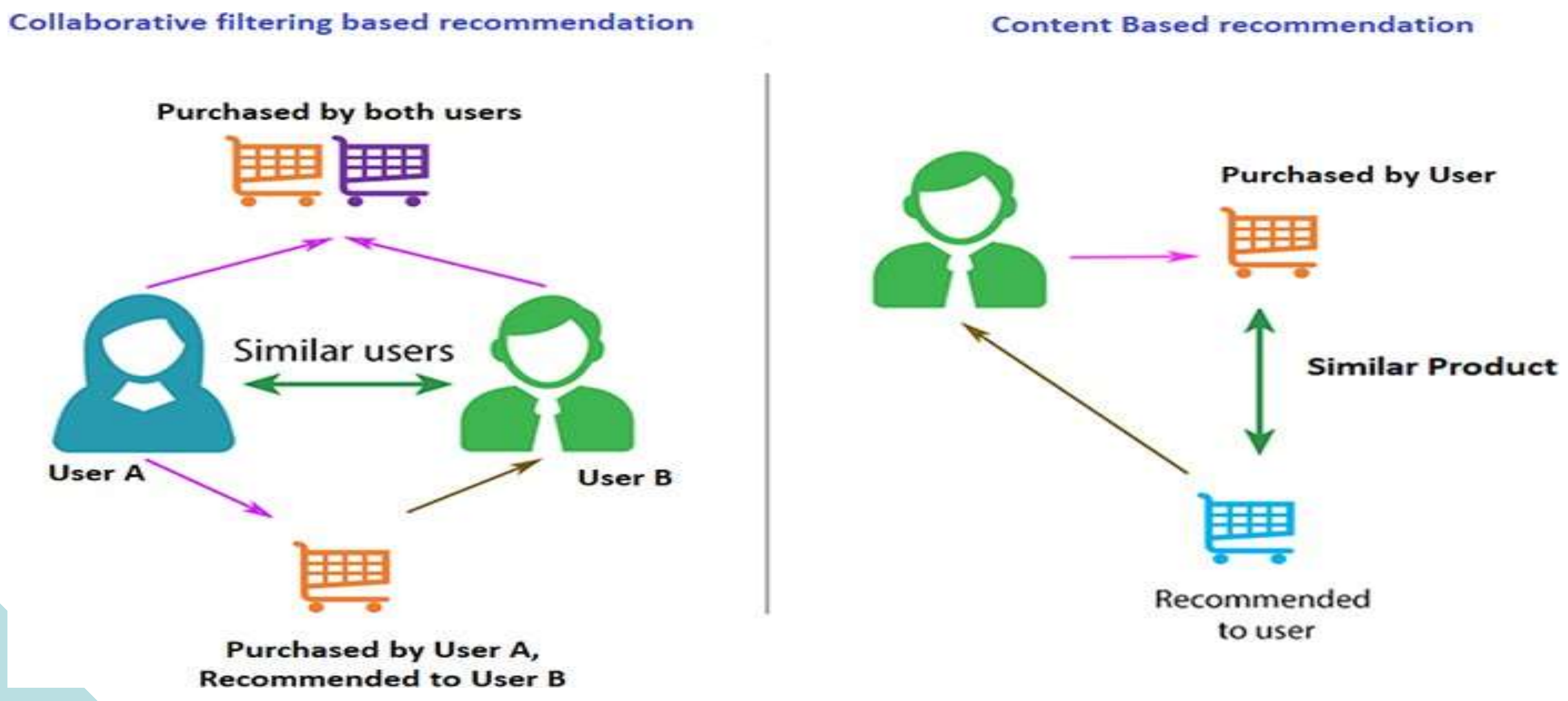
Applications of Machine Learning



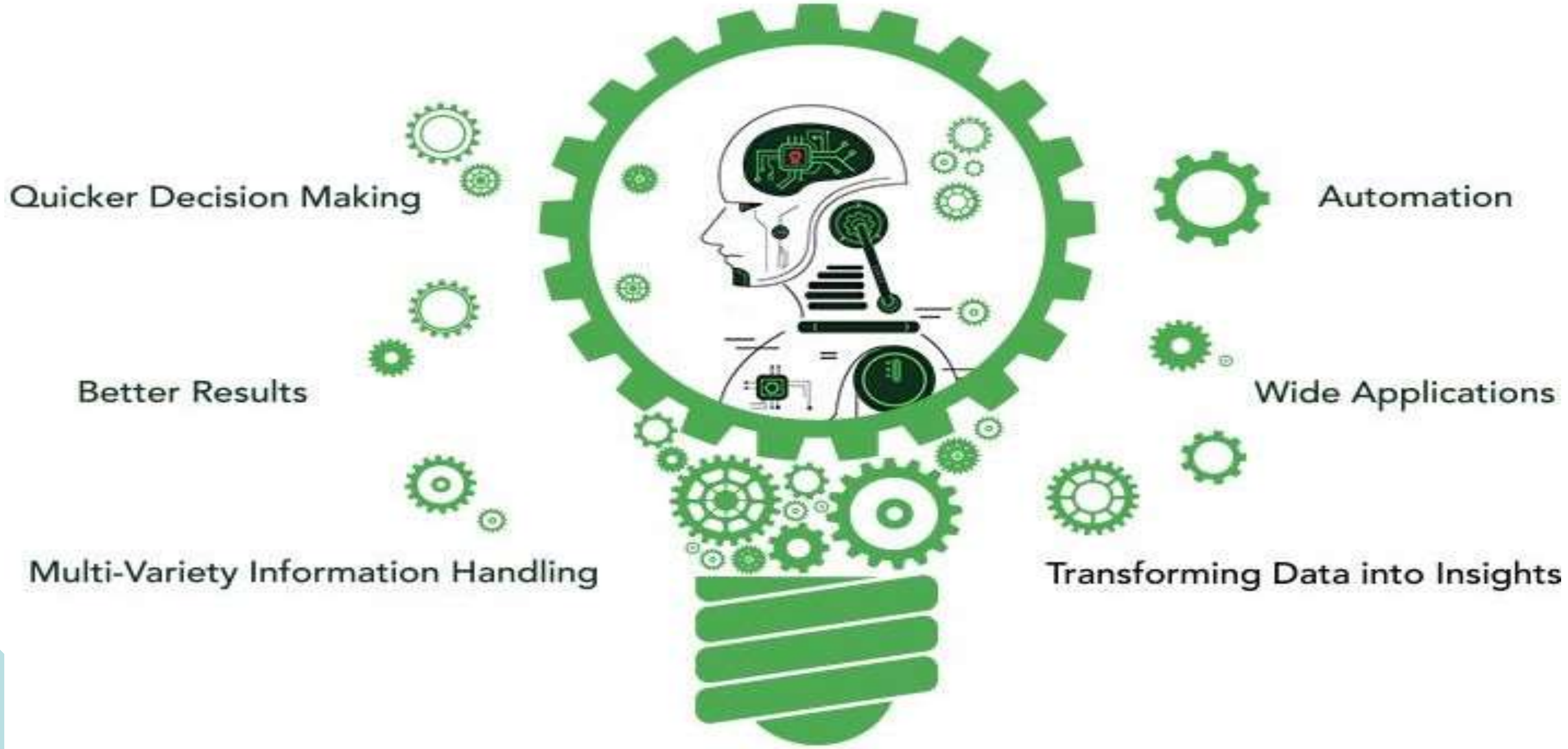
Applications of Machine Learning – Health Care


Identification of Diseases and Diagnosis
Predicting Diseases
Drug Discovery and Manufacturing
Machine Learning in Health care
Smart Health Records
Medical Imaging
Personalized Medicine

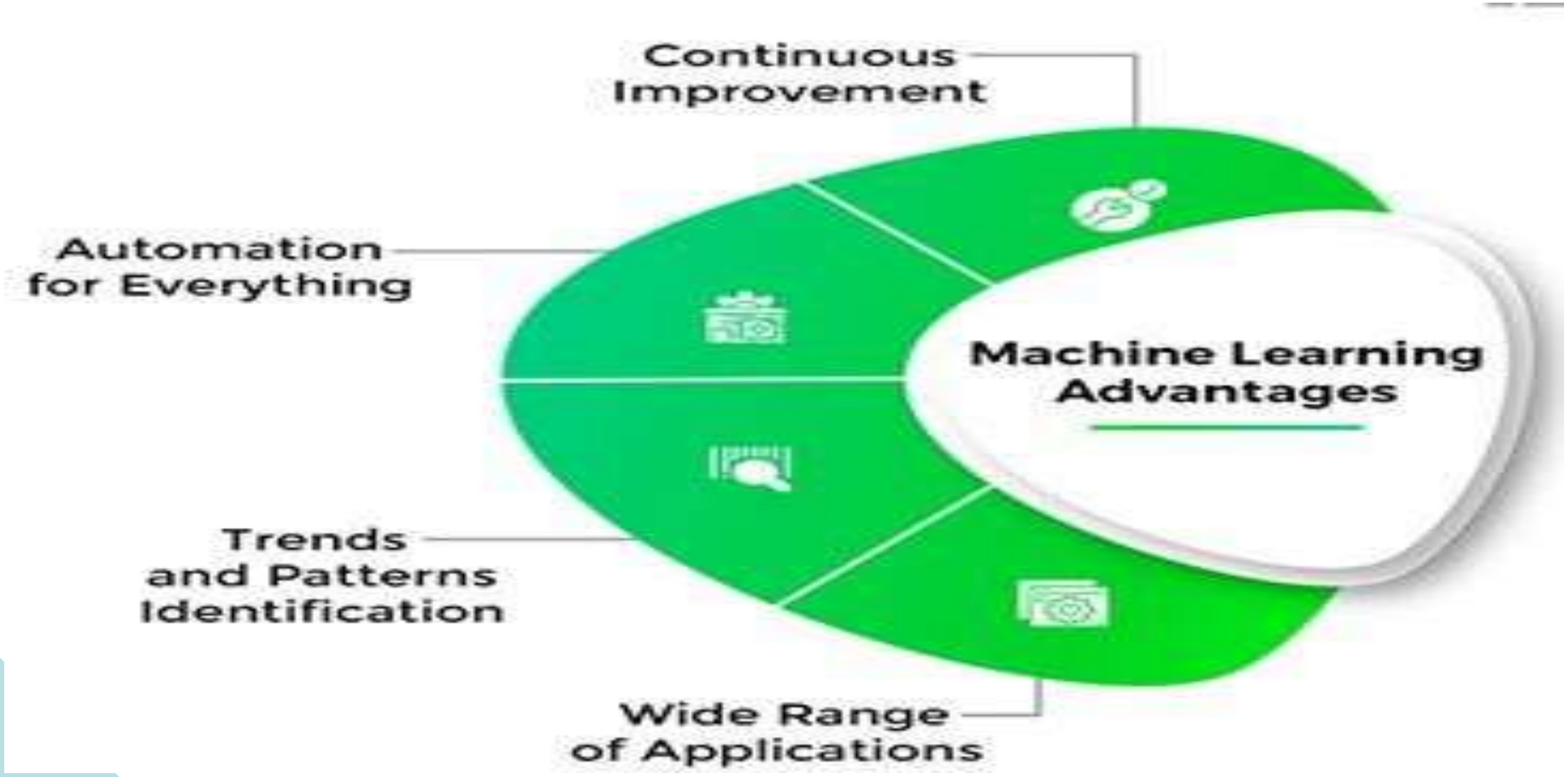
Applications of Machine Learning – Recommendations


Collaborative filtering based recommendation
Purchased by both users
Similar users
User A
User B
Purchased by User A, Recommended to User B
Content Based recommendation
Purchased by User
Similar Product
Recommended to user

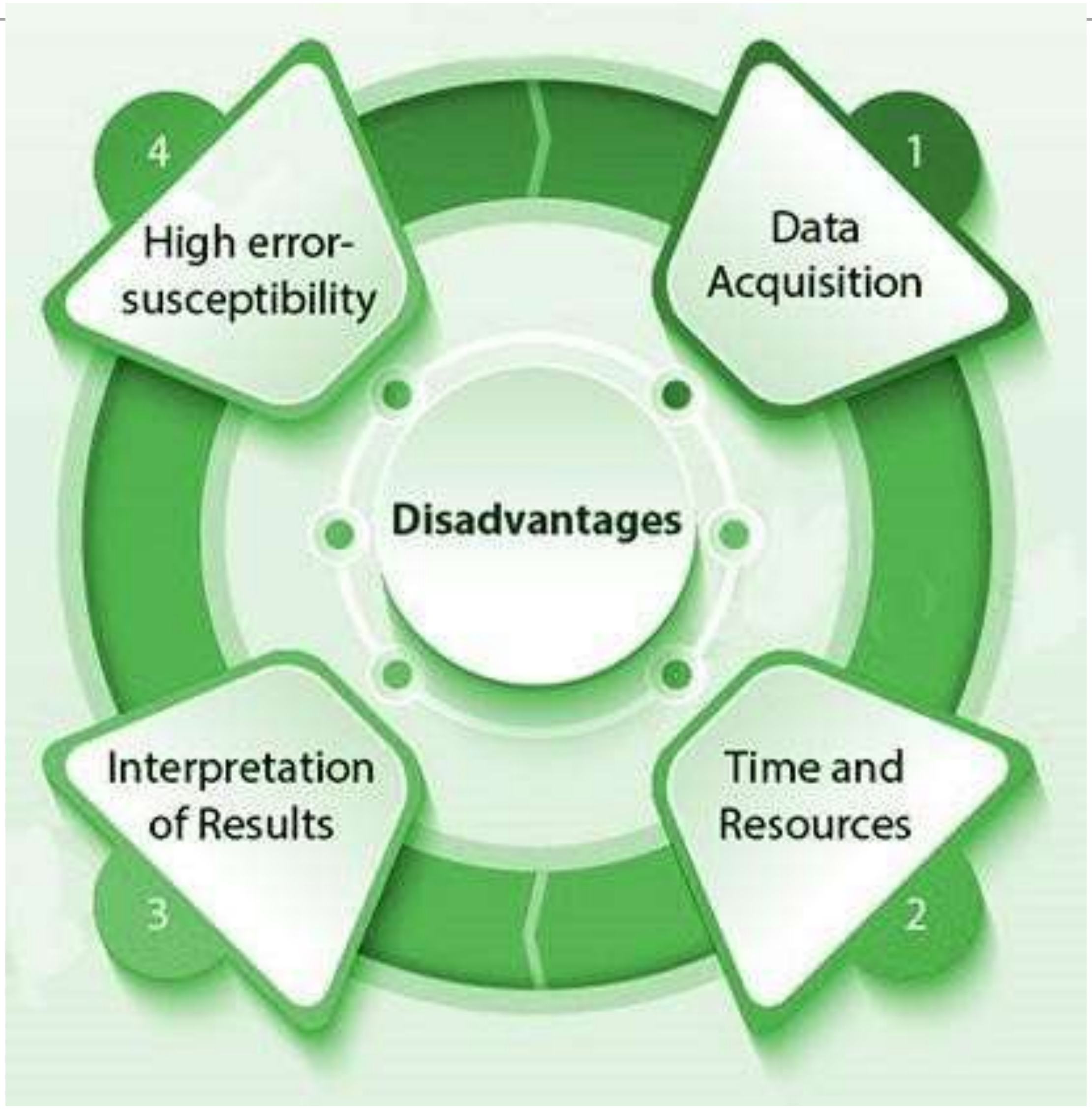
Need for Machine Learning


Quicker Decision Making
Better Results
Multi-Variety Information Handling
Automation
Wide Applications
Transforming Data into Insights

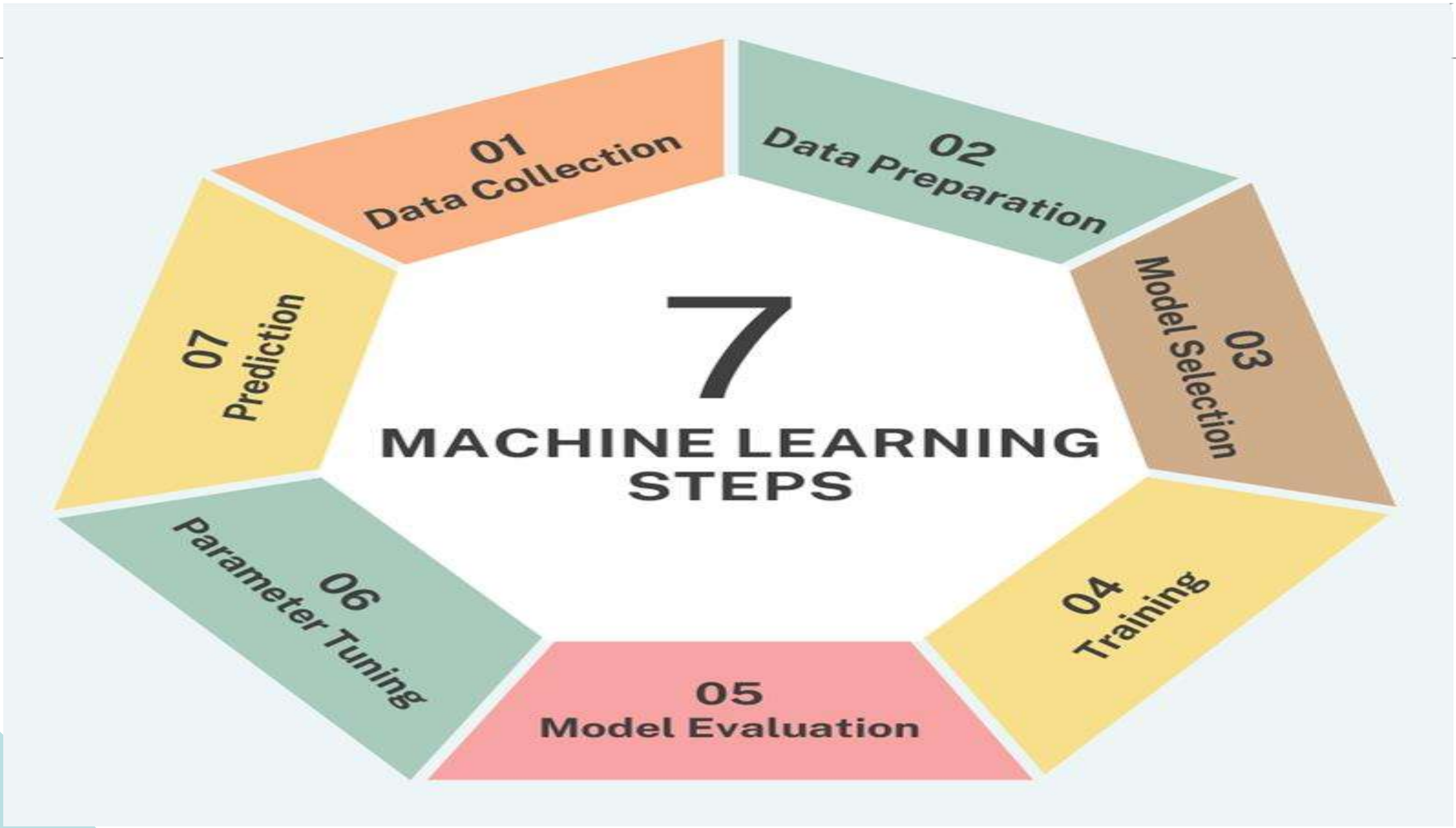
Advantages of Machine Learning


Continuous Improvement
Automation for Everything
Machine Learning Advantages
Trends and Patterns Identification
Wide Range of Applications

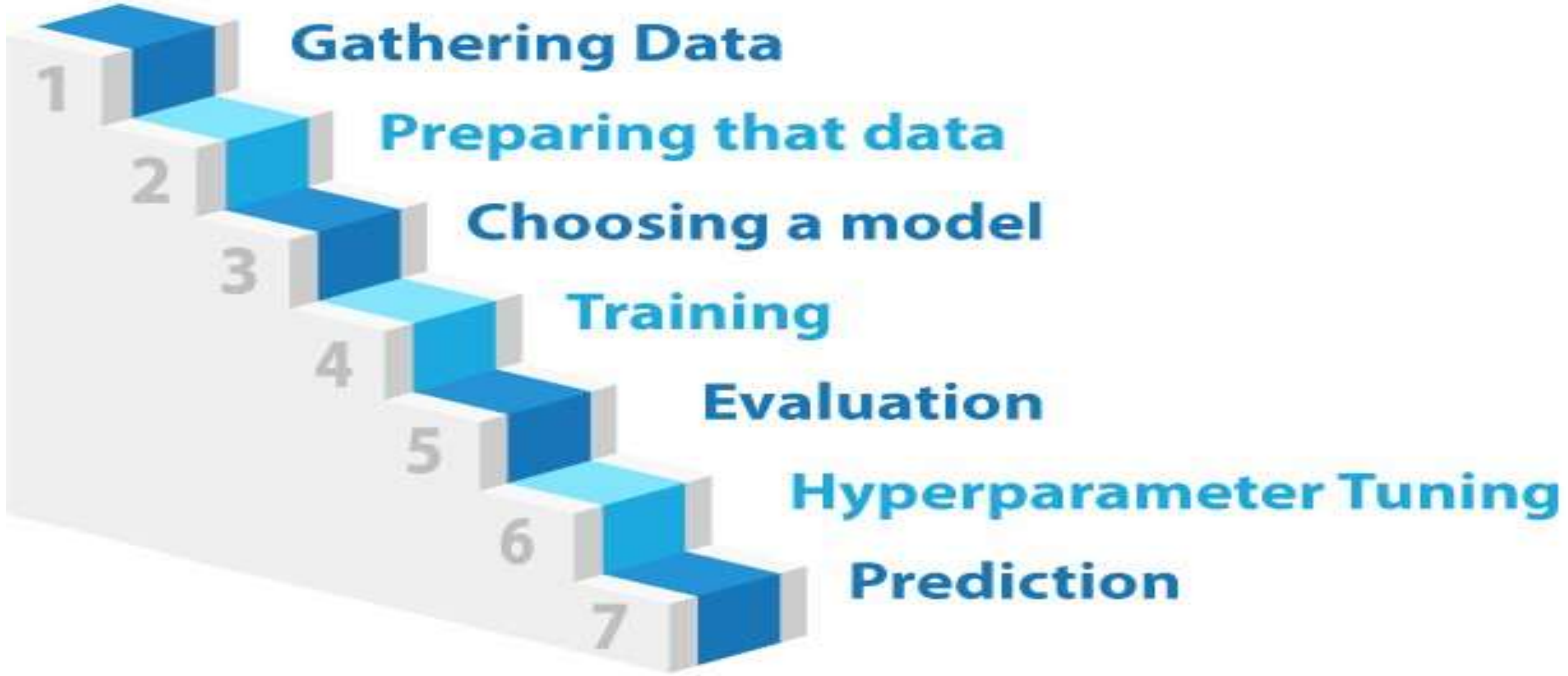
Disadvantages of Machine Learning



Machine Learning Steps



7 steps of Machine Learning

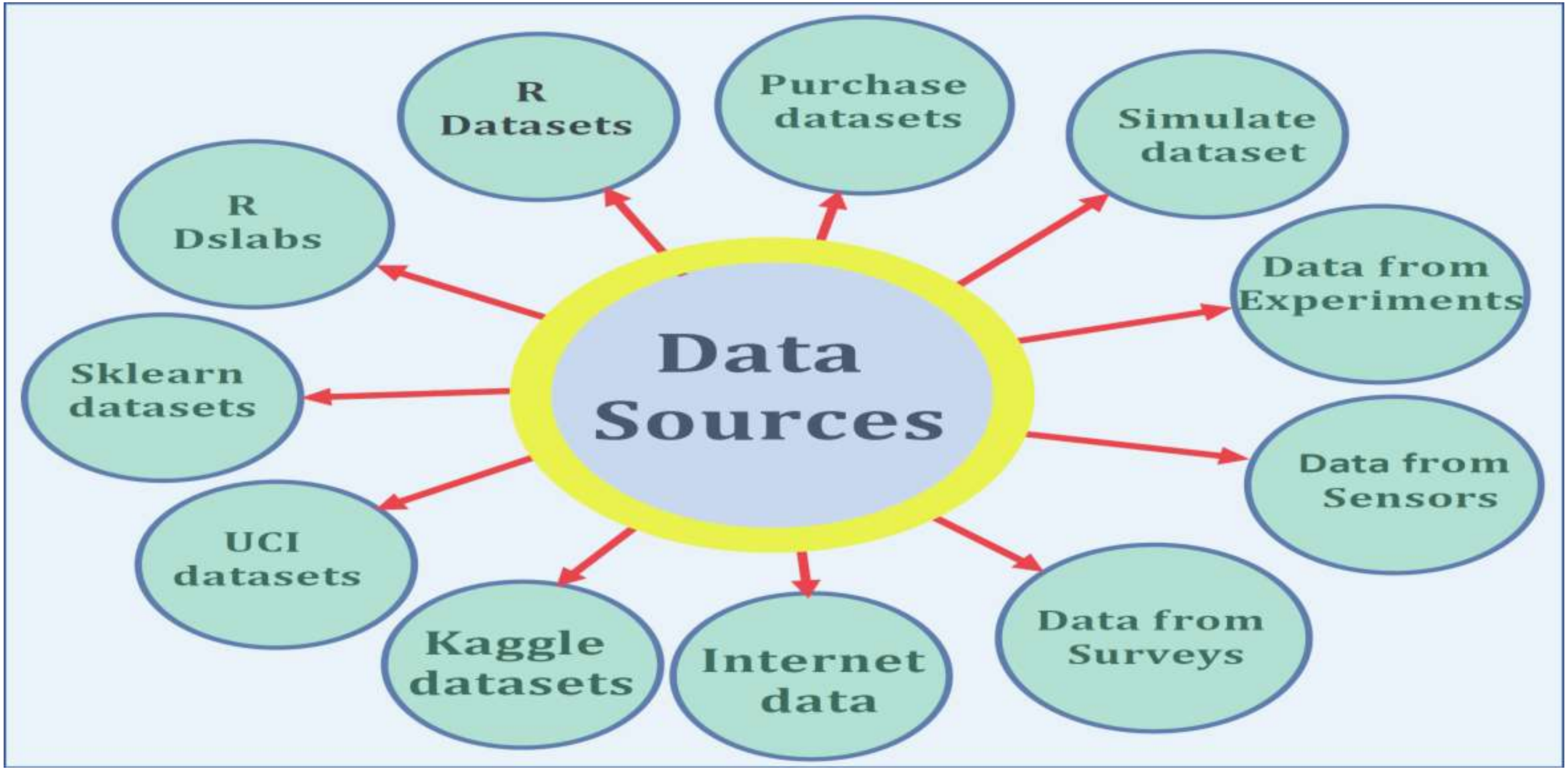


Machine Learning – Step1

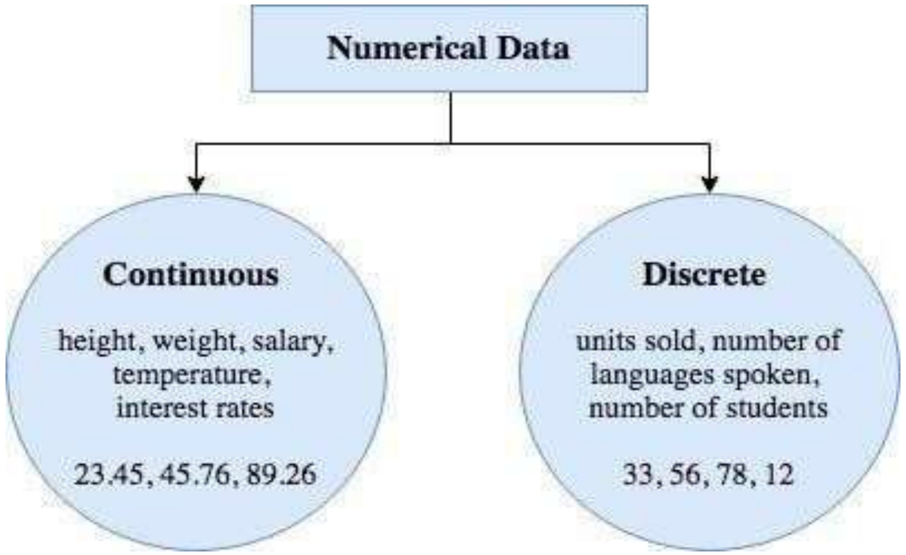
Gathering the Data

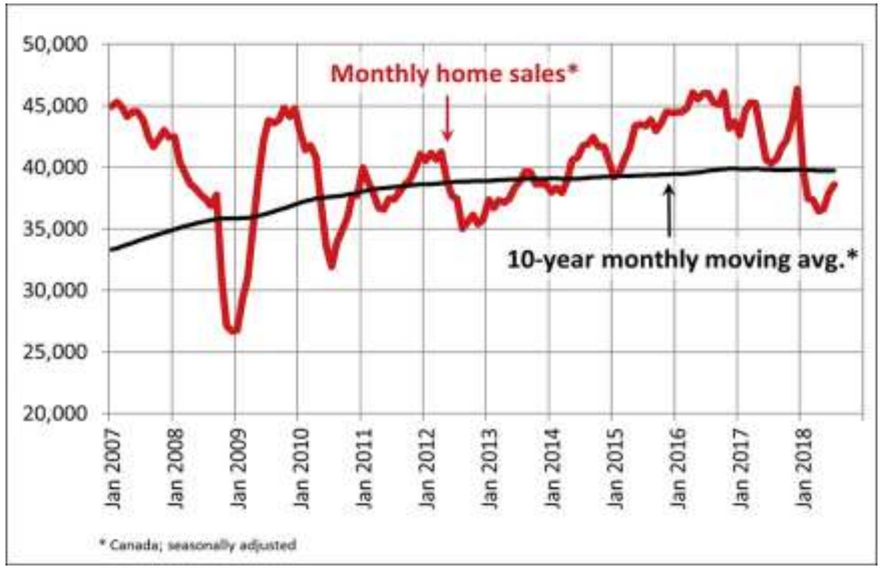
- Gathering Data:
- It can be any unprocessed fact, value, text, sound or picture that is not being interpreted and analyzed.
- Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence.
- Without data, we can’t train any model and all modern research and automation will go in vain. Big Enterprises are spending lots of money just to gather as much certain data as possible.
- Data is typically divided into two types: labeled and unlabeled.
- The data used in machine learning is typically numerical or categorical.

Machine Learning Steps – 1. Data Collection

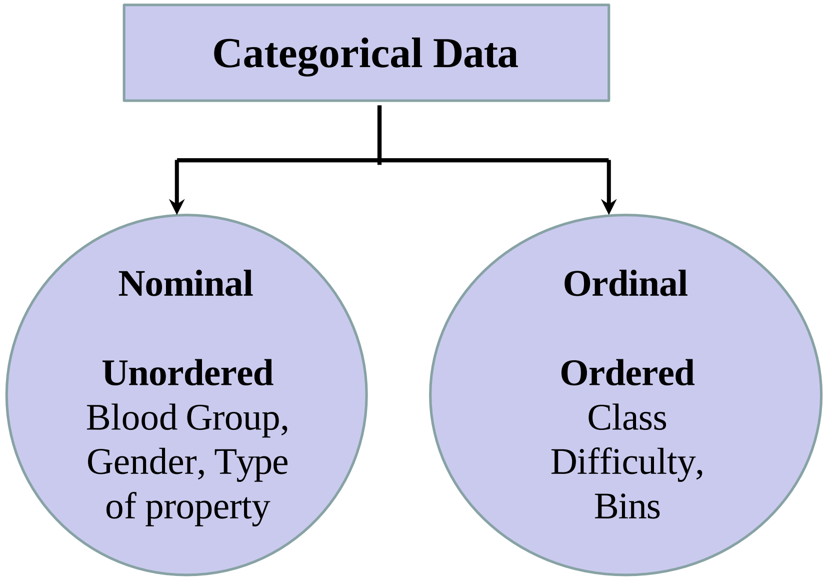


Types of Data

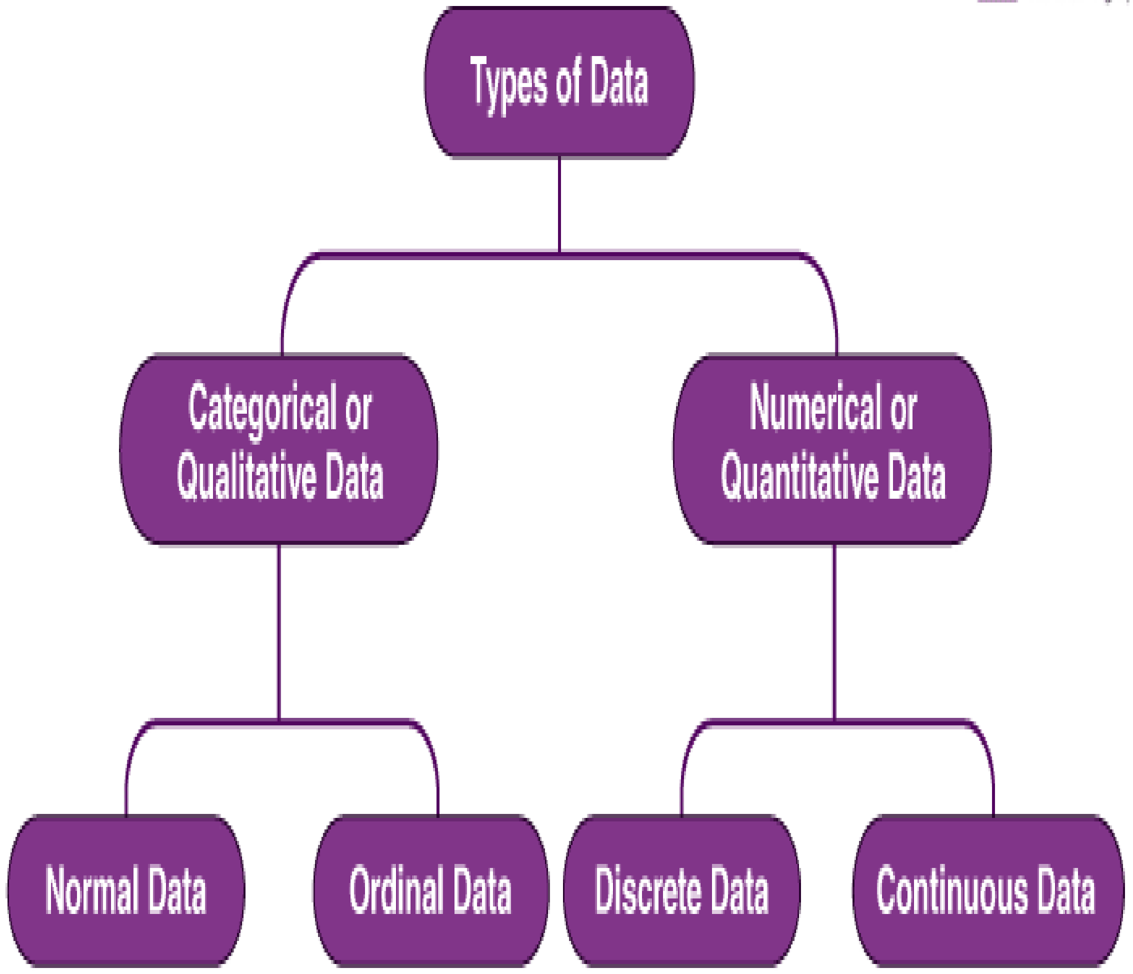

Numerical Data
Continuous
height, weight, salary, temperature, interest rates
23.45, 45.76, 89.26
Discrete
units sold, number of languages spoken, number of students
33, 56, 78, 12

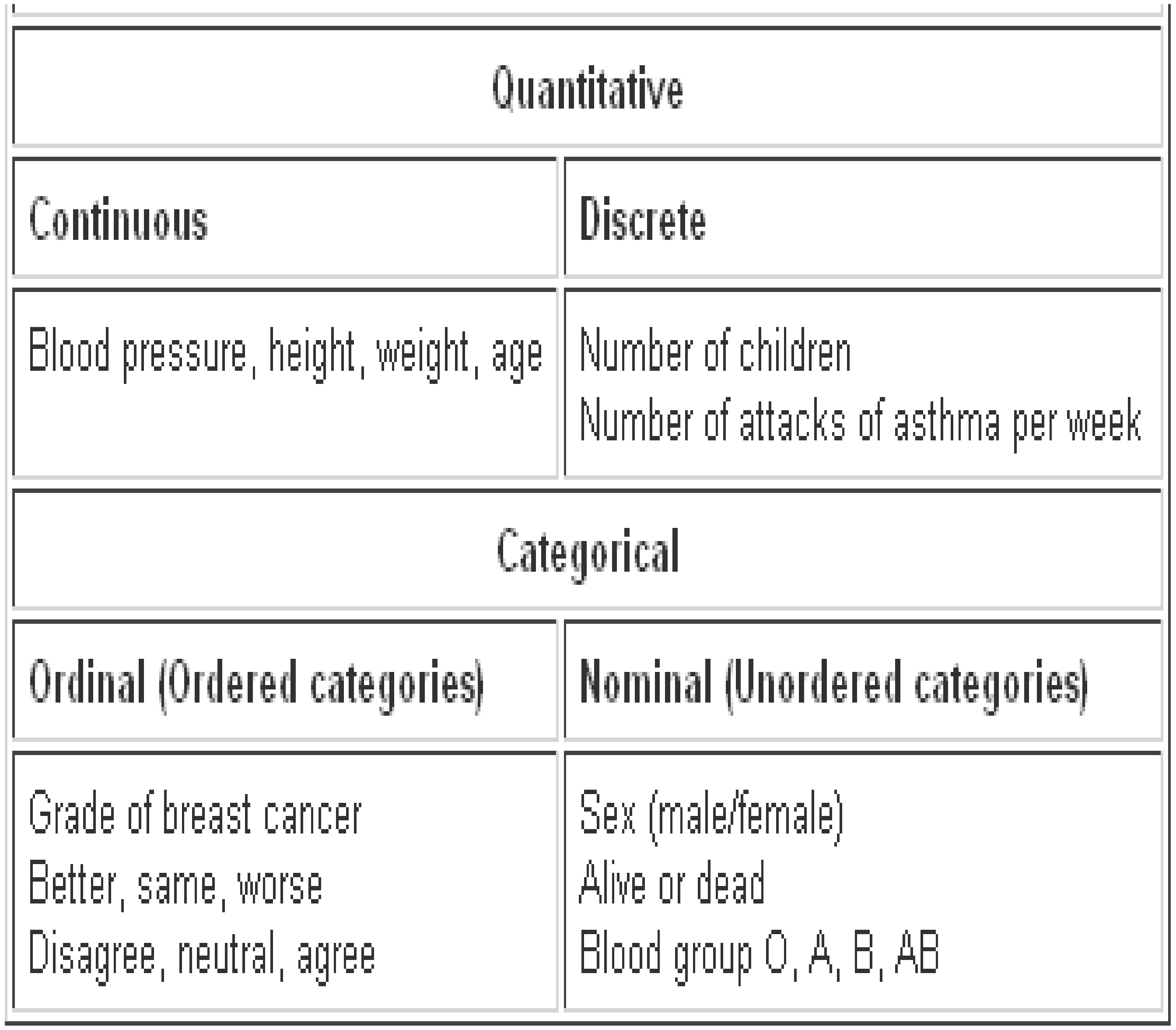


Numerical Data	Categorical Data
Time Series Data	Text



Types of Data





Quantitative	
Continuous	Discrete
Blood pressure, height, weight, age	Number of children Number of attacks of asthma per week
Categorical	
Ordinal (Ordered categories)	Nominal (Unordered categories)
Grade of breast cancer Better, same, worse Disagree, neutral, agree	Sex (male/female) Alive or dead Blood group O, A, B, AB

Types of Data

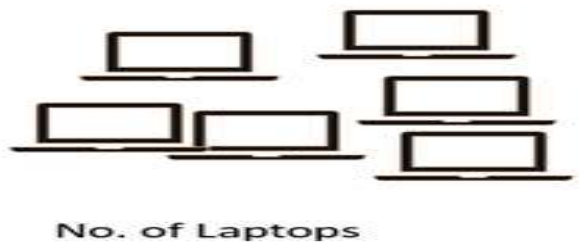
1. Quantitative Data Type:

This Type Of Data Type Consists Of Numerical Values. Anything Which Is Measured By Numbers. E.G., Profit, Quantity Sold, Height, Weight, Temperature, Etc.

This is again of two types **Discrete Data type and Continuous Data type**

A.) Discrete Data Type (counting process)

The Numeric Data Which Have Discrete Values Or Whole Numbers. This Type Of Variable Value If Expressed In Decimal Format Will Have No Proper Meaning. Their Values Can Be Counted.
Ex : No. Of Cars You Have, No. Of Marbles In Containers, Students In A Class etc



Types of Data

1. Quantitative Data Type:

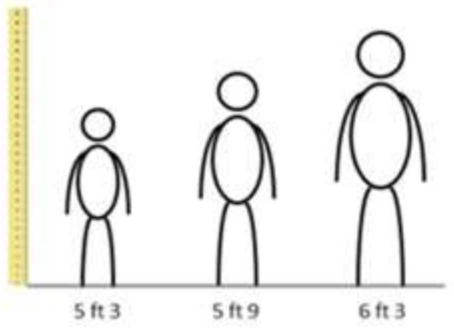
B.) Continuous Data Type (measuring process)

The Numerical Measures Which Can Take The Value Within A Certain Range.

This Type Of Variable Value If Expressed In Decimal Format Has True Meaning.

Their Values Can Not Be Counted But Measured. The Value Can Be Infinite

E.G.: – Height, Weight, Time, Area, Distance, Measurement Of Rainfall etc.



Height

Time

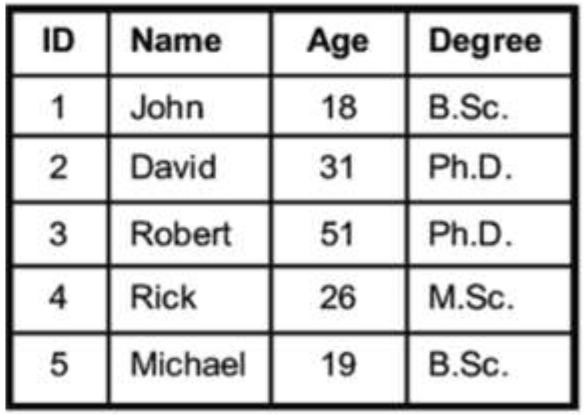
Types of Data

2. Qualitative Data Type:

These Are The Data Types That Cannot Be Expressed In Numbers. This Describes Categories Or Groups And Is Hence Known As The Categorical Data Type.

This Can Be Divided Into **Structured Data and Unstructured Data:**

A. Structured Data:
This Type Of Data Is Either Number Or Words. This Can Take Numerical Values But Mathematical Operations Cannot Be Performed On It. This Type Of Data Is Expressed In Tabular Format.
E.G.) Sunny=1, Cloudy=2, Windy=3 Or Binary Form Data Like 0 Or1, Good Or Bad, Etc.

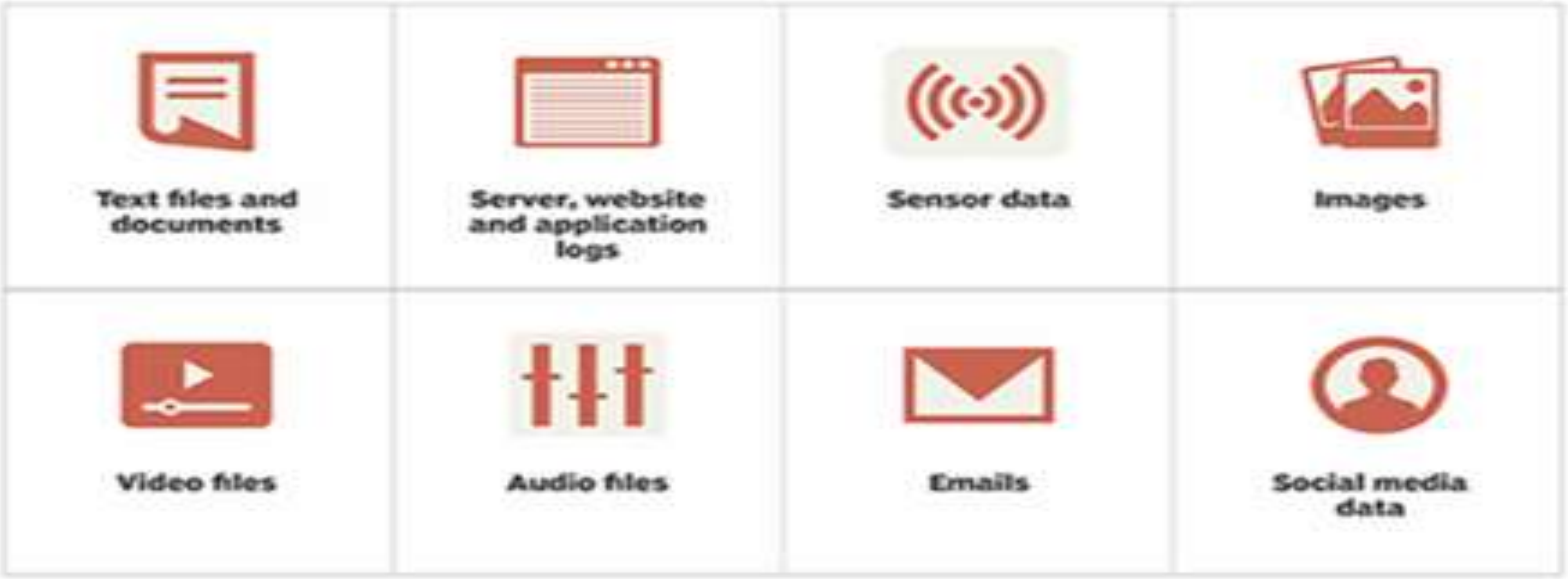


ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

Types of Data

B. Unstructured Data:

This Type Of Data Does Not Have The Proper Format And Therefore Known As Unstructured Data. This Comprises Textual Data, Sounds, Images, Videos, Etc.



Types of Data

Besides This, There Are Also Other Types Refer As Data Types Preliminaries Or Data Measures:-

1. **Nominal**
2. **Ordinal**
3. **Interval**
4. **Ratio**

These Can Also Be Refer Different Scales Of Measurements.

I. Nominal Data Type:
This Is In Use To Express Names Or Labels Which Are Not Order Or Measurable.
E.G., Male Or Female (Gender), Race, Country, Etc.

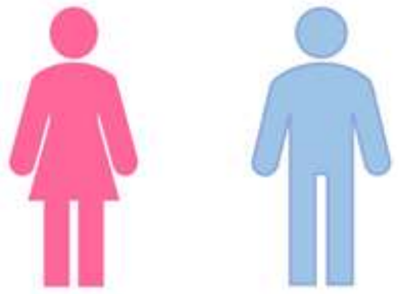


Fig: Gender (Female, Male), An Example Of Nominal Data Type

Types of Data

II. Ordinal Data Type:

This Is Also A Categorical Data Type Like Nominal Data But Has Some Natural Ordering Associated With It. E.G., Likert Rating Scale, Shirt Sizes, Ranks, Grades, Etc.

Good
Average
Poor

Hot
Hotter
Hottest

III. Interval Data Type:

This Is Numeric Data Which Has Proper Order And The Exact Zero Means The True Absence Of A Value Attached. Here Zero Means Not A Complete Absence But Has Some Value. This Is The Local Scale.
Ex : Temperature Measured In Degree Celsius, Time, Sat Score, Credit Score, PH, Etc. Difference Between Values Is Familiar. In This Case, There Is No Absolute Zero. Absolute

Types of Data

IV. Ratio Data Type:

This Quantitative Data Type Is The Same As The Interval Data Type But Has The Absolute Zero. Here Zero Means Complete Absence And The Scale Starts From Zero. This Is The Global Scale.

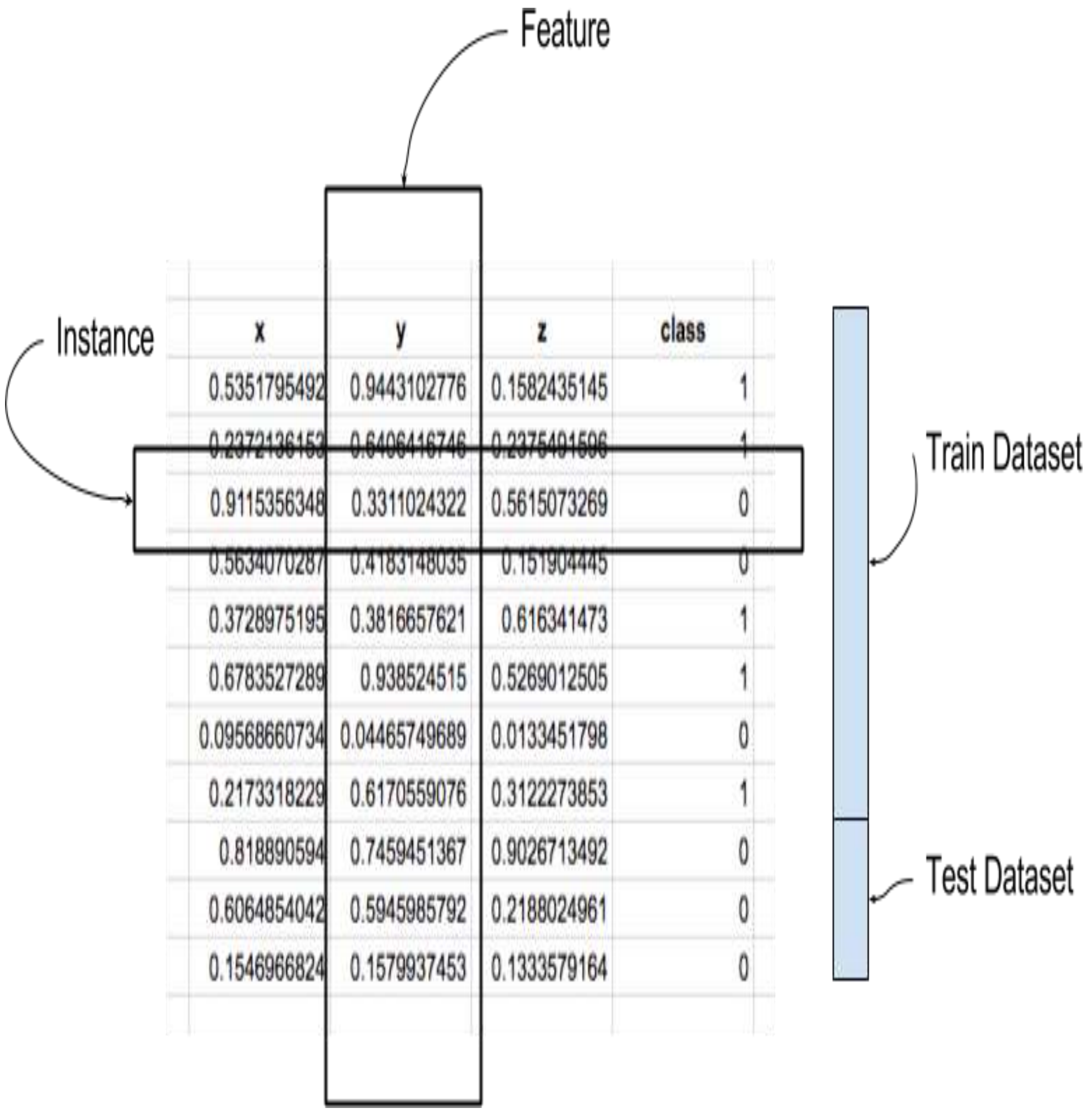
E.G., Temperature In Kelvin, Height, Weight, Etc.

Fig: Weight, An Example Of Ratio Data Type

Dataset

- **A data set** is an organized collection of data. They are generally associated with a unique body of work and typically cover one topic at a time.
- **Rows/**instances/observations/records/samples/ objects/predictors.
- **Columns/**features/attributes/variables/fields and characteristics.
- Each dataset has one or more independent / input variables and one dependent variable /output variable.

- Independent variables - Input variable/predictor variable.
- Dependent variable - Output variable/target variable/response variable.



x	y	z	class
0.5351795492	0.9443102776	0.1582435145	1
0.2372136153	0.6406416746	0.2375401596	1
0.9115356348	0.3311024322	0.5615073269	0
0.5634070287	0.4183148035	0.151904445	0
0.3728975195	0.3816657621	0.616341473	1
0.6783527289	0.938524515	0.5269012505	1
0.09568660734	0.04465749689	0.0133451798	0
0.2173318229	0.6170559076	0.3122273853	1
0.818890594	0.7459451367	0.9026713492	0
0.6064854042	0.5945985792	0.2188024961	0
0.1546966824	0.1579937453	0.1333579164	0

Types of Datasets

- 1.Data set consists of only numerical attributes
- 2.Data set consists of only categorical attributes
- 3.Data set consists of both numerical and categorical attributes
- Dataset1: Dataset2: Dataset3:

Dataset1:

age	income	height	weight
20	12000	6.3	30
40	15000	5.2	70
35	20000	5.6	65
60	100000	5.4	59

Dataset2:

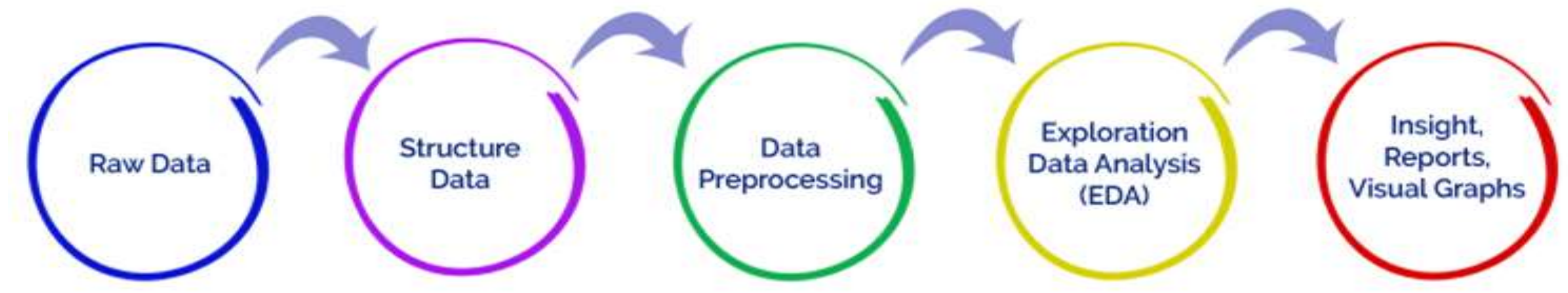
age	income	studen t
youth	Fair	Yes
youth	Good	No
senior	excellent	Yes
middle	Good	Yes
senior	Fair	No
middle	good	no

Dataset3:

age	income	Credit rating
youth	12000	Yes
senior	15000	No
middle	20000	Yes
youth	100000	Yes

Machine Learning – Step2

Preparation of Data


Raw Data
Structure Data
Data Preprocessing
Exploration Data Analysis (EDA)
Insight, Reports, Visual Graphs

- **When the data is gathered from different sources it is collected in Raw format which is not feasible for the analysis.**
- **Raw data converted into Structured data (The data in specific format, such as table format)**
- **Data Preprocessing - is a technique that is used to convert the structured data into a clean data set.**
- **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**

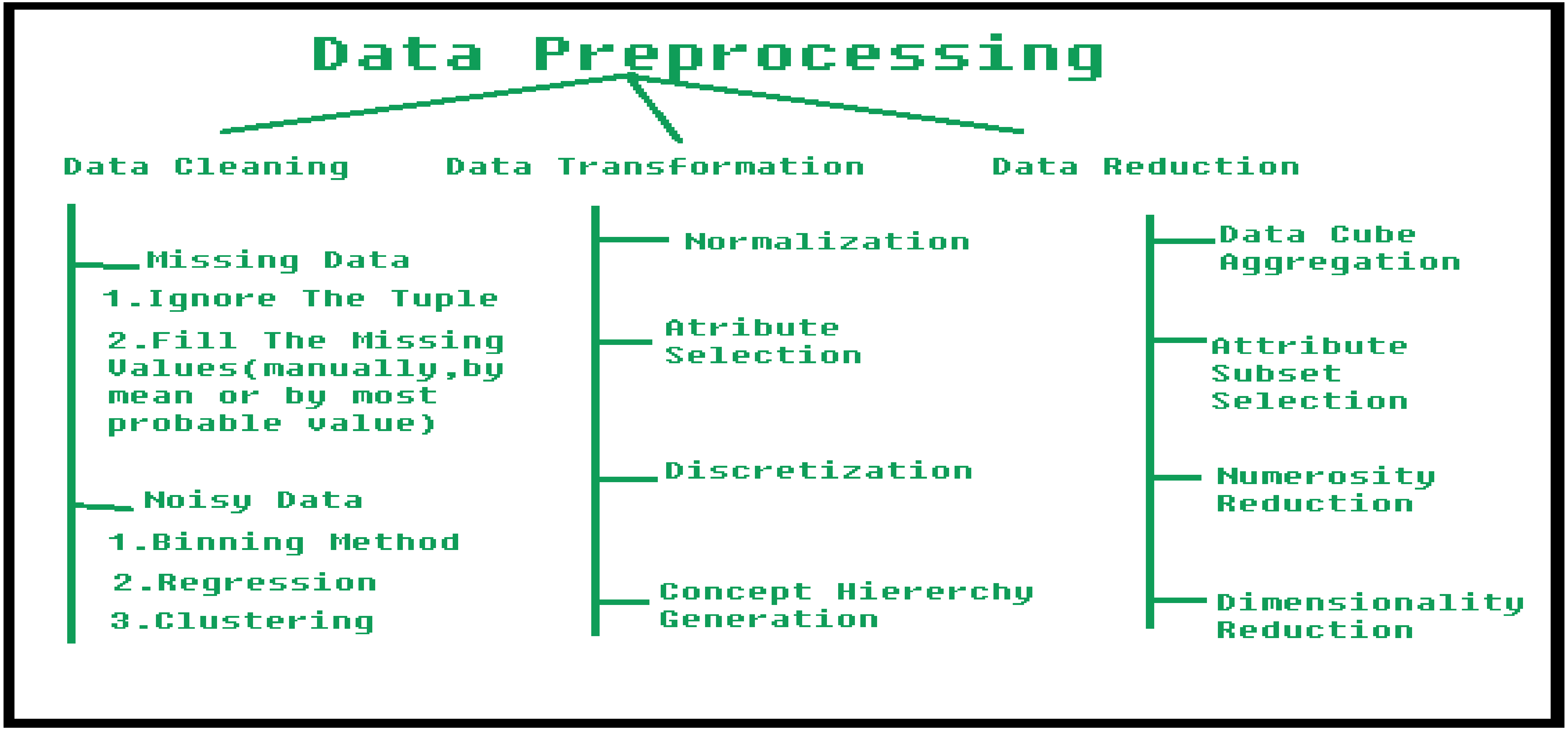
Machine Learning – Step2

Preparation of Data

- **Few essential tasks when working with data in the Data preparation step.**
- **Data cleaning: This task includes the identification of errors and making corrections or improvements to those errors.**
- **Feature Selection: We need to identify the most important or relevant input data variables for the model.**
- **Data Transformation: Data transformation involves converting raw data into a well-suitable format for the model.**
- **Feature Engineering: Feature engineering involves deriving new variables from the available dataset.**
- **Dimensionality Reduction: The dimensionality reduction process involves converting higher dimensions into lower dimension features without changing the information.**

Machine Learning – Step2

Preparation of Data



Machine Learning – Step2

Preparation of Data

- There are four stages of data processing: **cleaning, integration, reduction and transformation**.

1. Data cleaning: or cleansing/scrubbing is the process of cleaning datasets by accounting for missing values, removing outliers, correcting inconsistent data points, and smoothing noisy data. In essence, the motive behind data cleaning is to offer complete and accurate samples for machine learning models.

The following are some methods used to solve the problem of noise:

•**Regression:** Regression analysis can help determine the variables that have an impact. This will enable you to work with only the essential features instead of analyzing large volumes of data. Both linear regression and multiple linear regression can be used for smoothing the data.

•**Binning:** Binning methods can be used for a collection of sorted data. They smoothen a sorted value by looking at the values around it. The sorted values are then divided into "bins," which means sorting data into smaller segments of the same size. There are different techniques for

Machine Learning – Step2

Preparation of Data

2． **Data Integration:-** Since data is collected from various sources, data integration is a crucial part of data preparation. Integration may lead to several inconsistent and redundant data points, ultimately leading to models with inferior accuracy.

- Here are some approaches to integrate data:

- **Data consolidation:** Data is physically brought together and stored in a single place. Having all data in one place increases efficiency and productivity. This step typically involves using data warehouse software.
- **Data virtualization:** In this approach, an interface provides a unified and real-time view of data from multiple sources. In other words, data can be viewed from a single point of view.
- **Data propagation:** Involves copying data from one location to another with the help of specific applications. This process can be synchronous or asynchronous and is usually event-driven.

Machine Learning – Step2

Preparation of Data

3. Data Reduction - There are two segments of dimensionality reduction:

Feature selection (selecting a subset of the variables)--try to find a subset of the original set of features. This allows us to get a smaller subset that can be used to visualize the problem using data modeling

Feature extraction (extracting new variables from the data)---reduces the data in a high-dimensional space to a lower-dimensional space, or in other words, space with a lesser number of dimensions.

- The following are some ways to perform dimensionality reduction:

- Principal component analysis (PCA): A statistical technique used to extract a new set of variables from a large set of variables. The newly extracted variables are called principal components. This method works only for features with numerical values.
- High correlation filter: A technique used to find highly correlated features and remove them; otherwise, a pair of highly correlated variables can increase the multicollinearity in the dataset.
- Missing values ratio: This method removes attributes having missing values more than a specified threshold.
- Low variance filter: Involves removing normalized attributes having variance less than a threshold value as minor changes in data translate to less

Machine Learning - Step2

Preparation of Data

4. Data transformation is the process of converting data from one format to another. It involves methods for transforming data into appropriate formats that the computer can learn efficiently from.

The following are some strategies for data transformation.

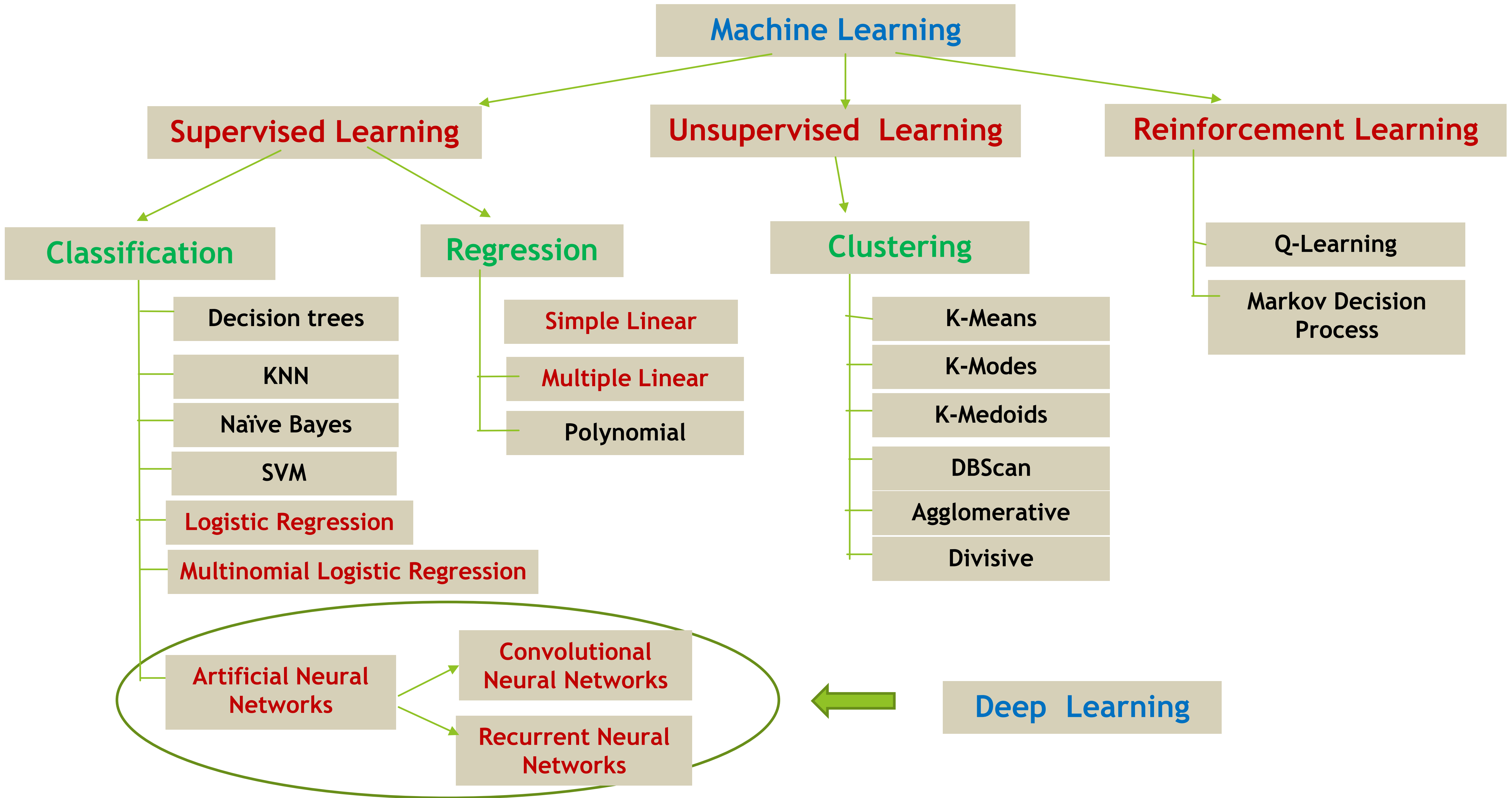
Smoothing - This statistical approach is used to remove noise from the data with the help of algorithms. It helps highlight the most valuable features in a dataset and predict patterns. It also involves eliminating outliers from the dataset to make the patterns more visible.

Aggregation - Aggregation refers to pooling data from multiple sources and presenting it in a unified format for data mining or analysis.

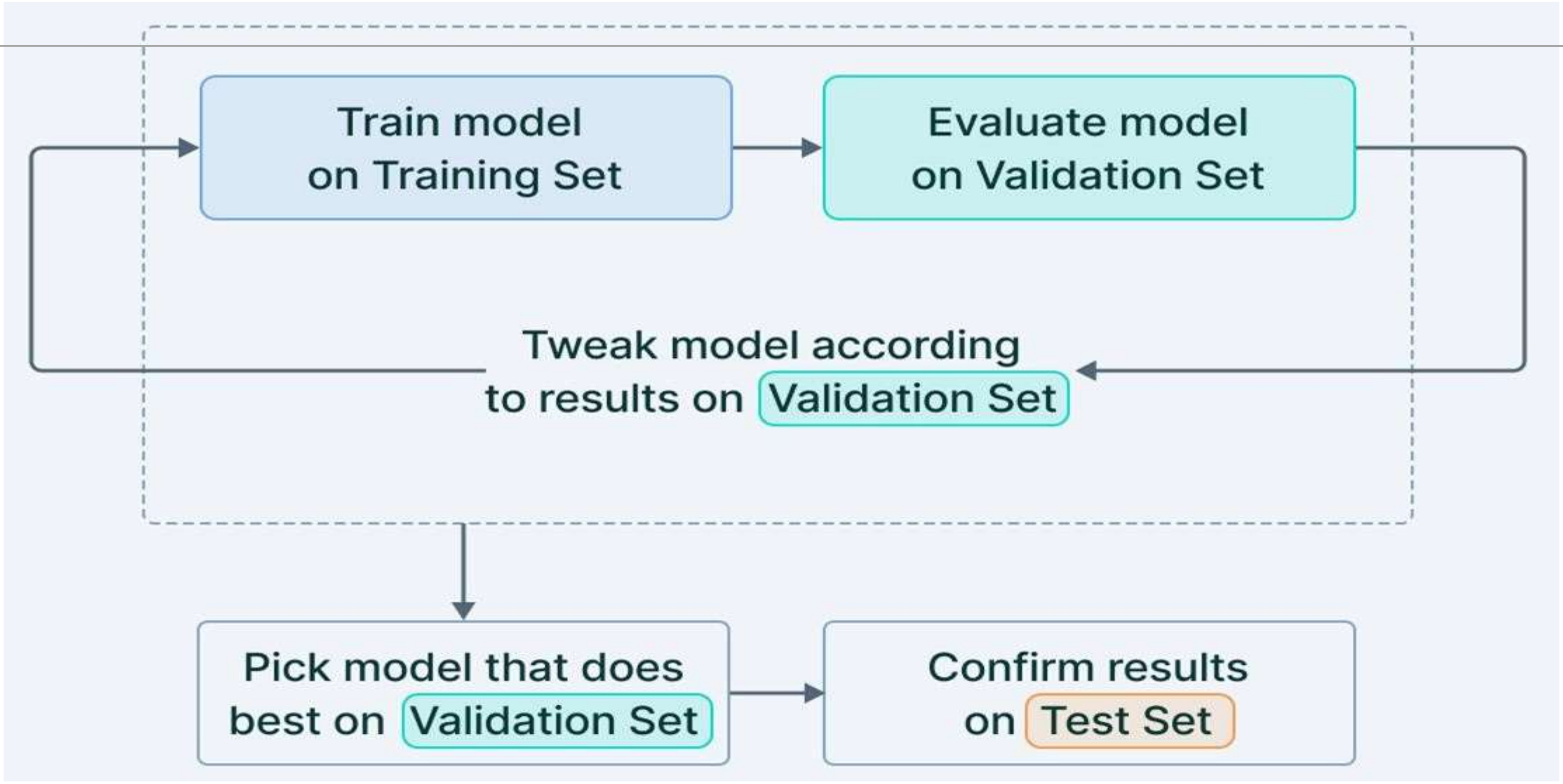
Discretization - Discretization involves converting continuous data into sets of smaller intervals. For example, it’s more efficient to place people in categories such as “teen,” “young adult,” “middle age,” or “senior” than using continuous age values.

Generalization- Generalization involves converting low-level data features into high-level data features. For instance, categorical attributes such as home address can be generalized to higher-level definitions such as city or state.

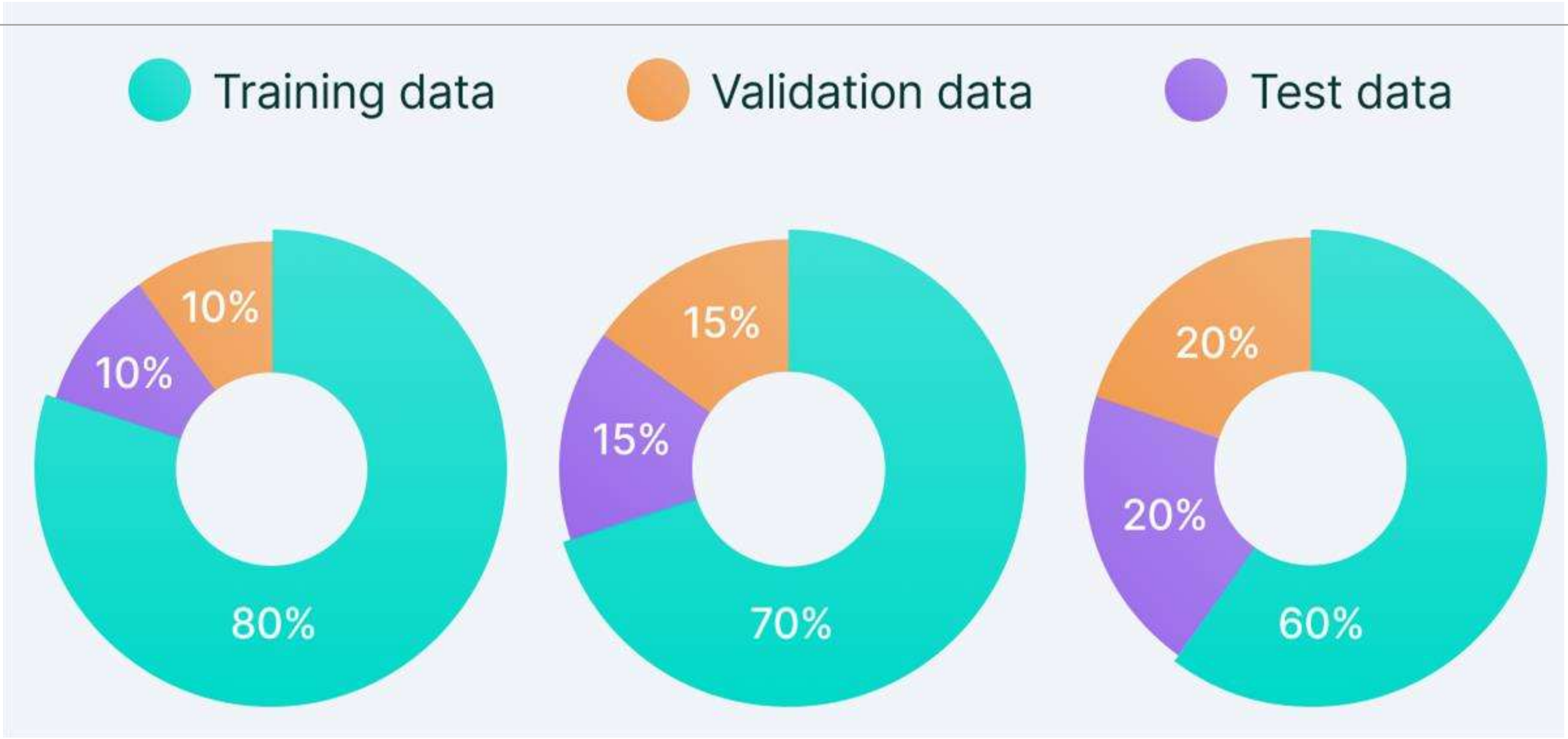
Machine Learning – Step 3 (Choosing a Model)



Machine Learning Steps – 4. Training


Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Machine Learning Steps – 4. Training



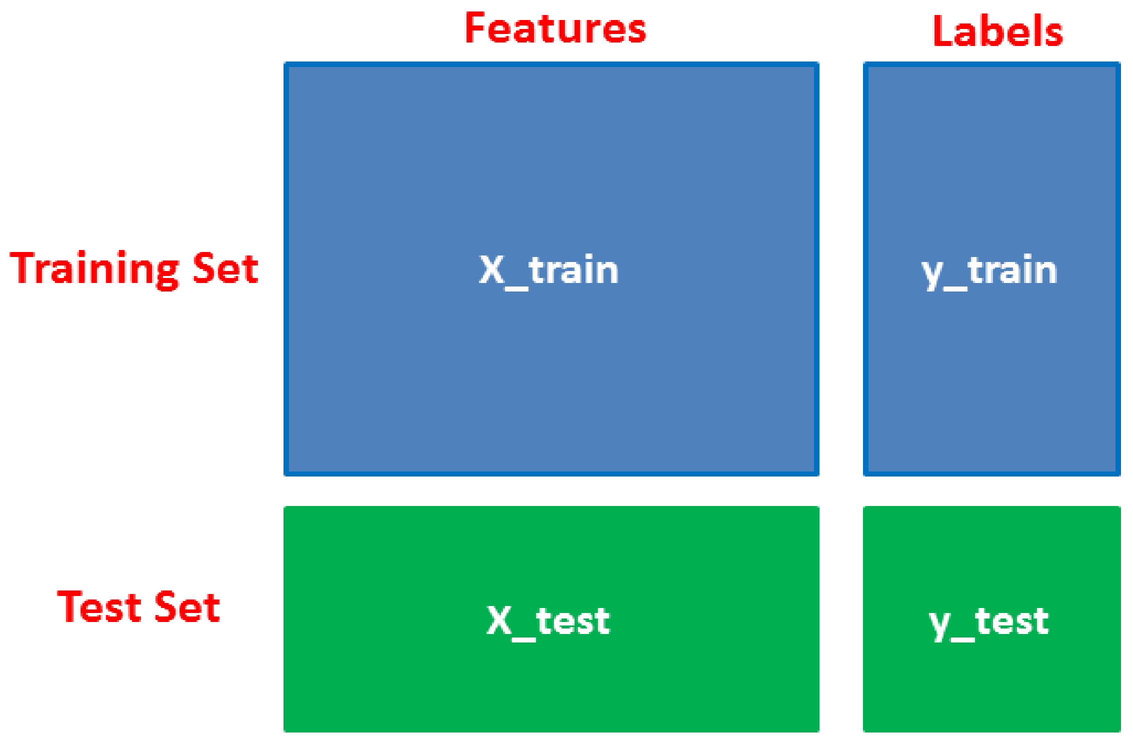
Machine Learning – Step4

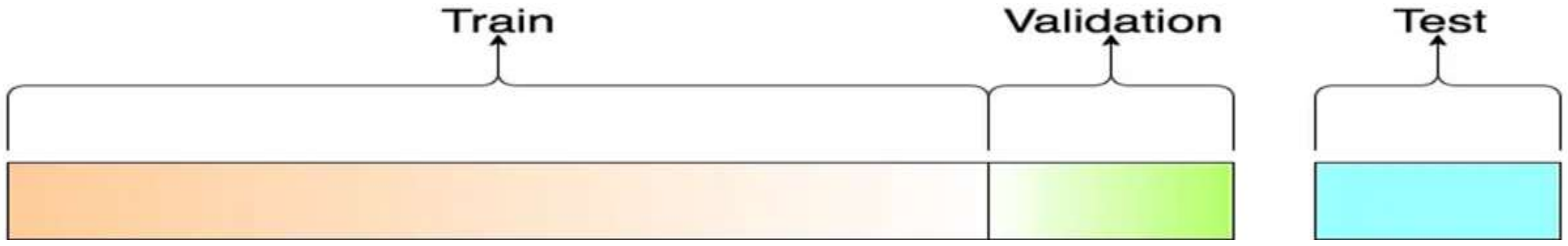
Training the Model

- In machine learning, a fundamental task is the development of algorithm models that analyze scenarios and make predictions. During this process Dataset split into different ratio.
- Dataset split mainly depends on 2 things –

(i) the no. of samples (rows) in our data.

(ii) actual model being training.



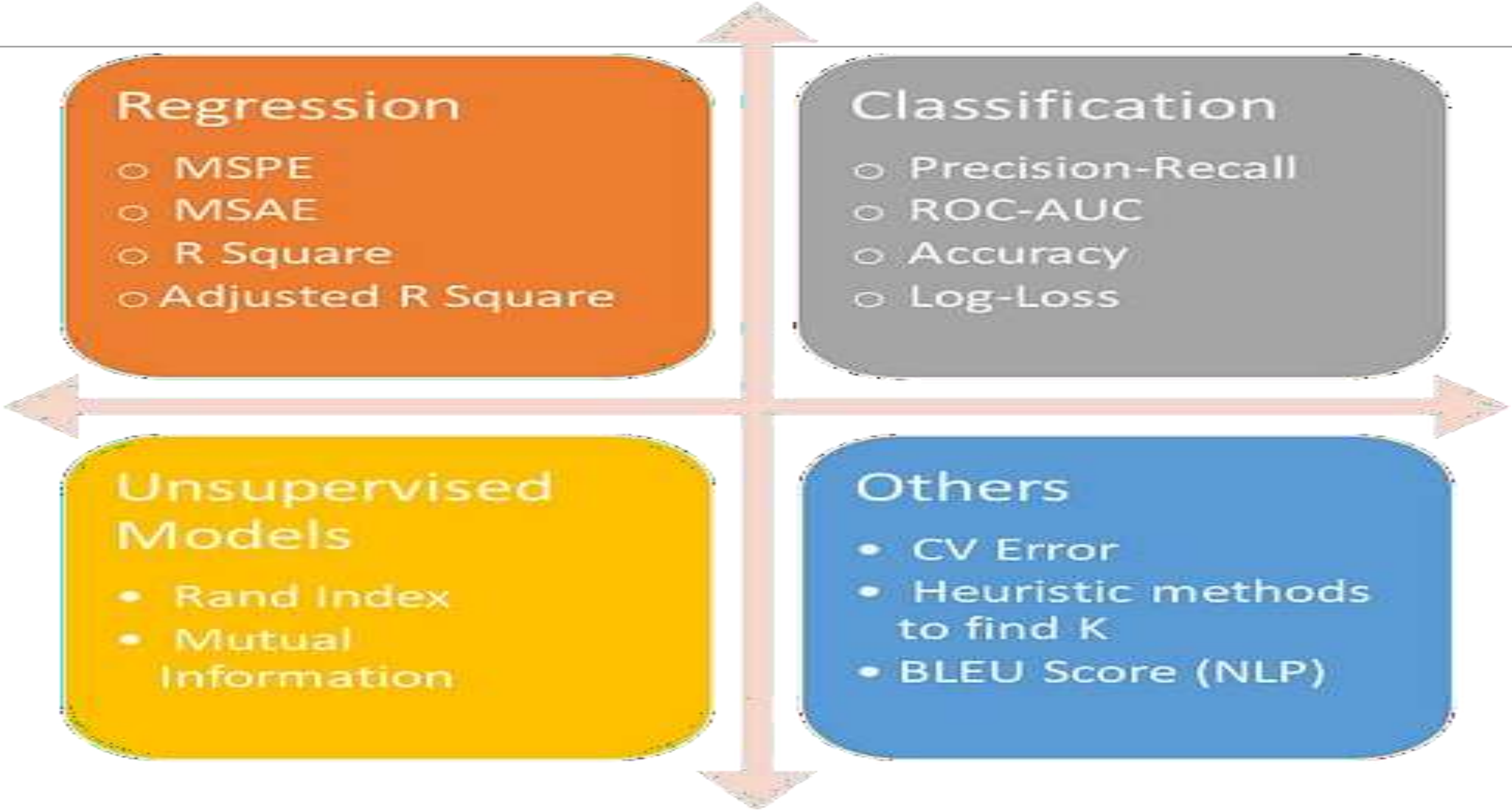


Machine Learning – Step4

Training the Model

- We can split the data set into three sets: a Training set , Validation and Testing set.
- 70%/80% for training, and 30%/20% for testing.(it depends on the given data)
- *Train* the model means ***create*** the model.
- *Test* the model means **test** the accuracy of the model.
- The fundamental purpose for splitting the dataset is to assess how effective will the trained model be in generalizing to new data.
- This split can be achieved by using train_test_split function of scikit-learn.
- The ideal split is **70%** training and **30%** testing set.
- Validation set is approximately **10-15%** of the total data available and part of training set but this can change depending upon the number of hyperparameters

Machine Learning Steps – 5. Model Evaluation



Regression

- MSPE
- MSAE
- R Square
- Adjusted R Square

Classification

- Precision-Recall
- ROC-AUC
- Accuracy
- Log-Loss

Unsupervised Models

- Rand Index
- Mutual Information

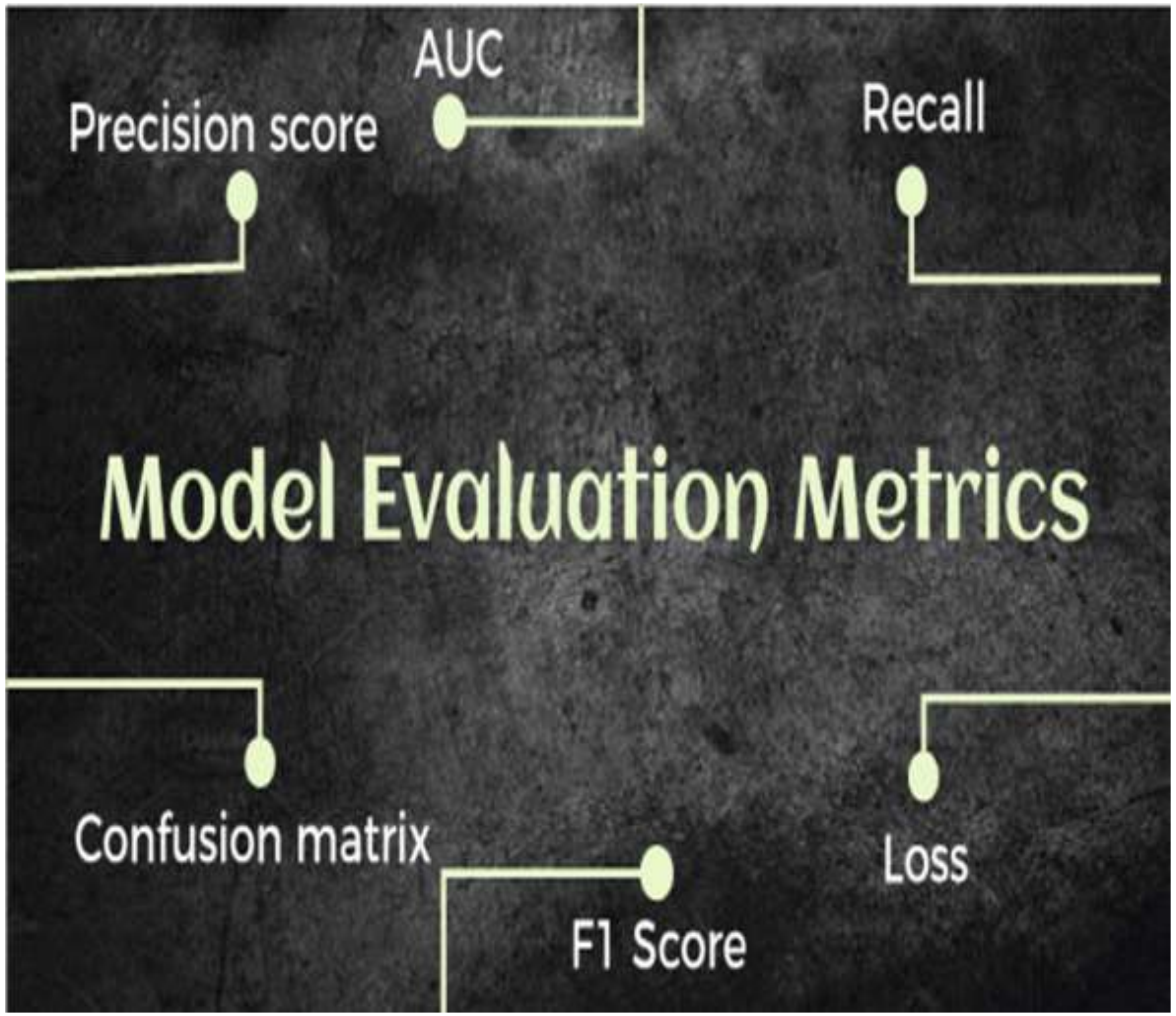
Others

- CV Error
- Heuristic methods to find K
- BLEU Score (NLP)

Machine Learning – Step5

Performance Evaluation

- In machine learning, each task or problem is divided into classification and regression . Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. In this topic, we will discuss metrics used for classification and regression tasks.



Machine Learning – Step 5

Performance Evaluation

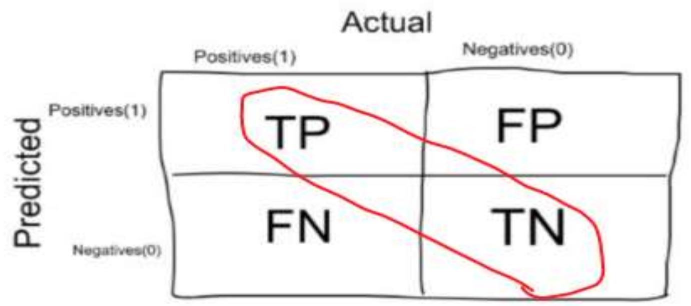
Performance Metrics for Classification

1. Confusion Matrix:

- A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.
- A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).

2. Accuracy- It can be determined as the number of correct predictions to the total number of predictions. Accuracy for the matrix can be calculated by taking average of the values lying across the "main diagonal".

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total number of predictions}}$$



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Machine Learning – Step5

Performance Evaluation

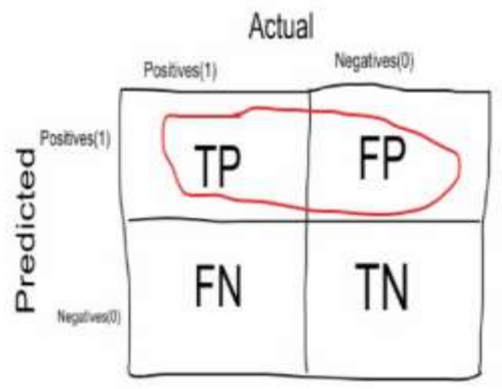
Performance Metrics for Classification

3. Precision:-It is the number of correct positive results divided by the number of positive results predicted by classifier

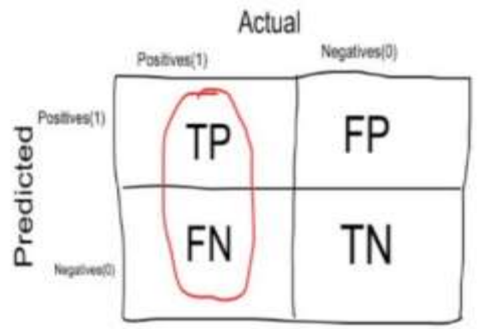
4. Recall : It is the number of correct positive results divided by the number of all relevant samples

5. F-Score: or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.

6.AUC(Area Under the Curve)-ROC: Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. The curve is plotted between two parameters, **True Positive Rate (TPR), False Positive Rate (FPR)**



$$\text{Precision} = \frac{TP}{TP + FP}$$



$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

Machine Learning – Step5

Performance Evaluation

Performance Metrics for Regression

- Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models.

1. Mean Absolute Error--MAE is one of the simplest metrics, which measures the absolute difference between actual and predicted values, where absolute means taking a number as Positive.

- Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

$$\boldsymbol{MAE = 1/N \sum |Y - Y'|}$$

2. Mean Squared Error--It measures the average of :ted values and the actual value given by the model.

$$\boldsymbol{MSE = 1/N \sum (Y - Y')^2}$$

3. R2 Score--R squarec :termination, which is another popular metric used for Regression model evaluation. The R-squared metric enables us to compare our model with a constant baseline to determine the performance of the model. To select the constant baseline, we need to take the mean of the data and draw the line at the mean.

$$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$$

Machine Learning – Step6
Hyperparameter Tuning

Parameters:

- A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data. They are required by the model when making predictions. They are estimated or learned from data.
- Parameters are key to machine learning algorithms. They are the part of the model that is learned from historical training data.
- Some examples of model parameters include:

- The **weights** in an artificial neural network.
- The **support vectors** in a support vector machine.
- The **coefficients** in a linear regression or logistic regression.

Machine Learning – Step6

Hyperparameter Tuning

Hyperparameters

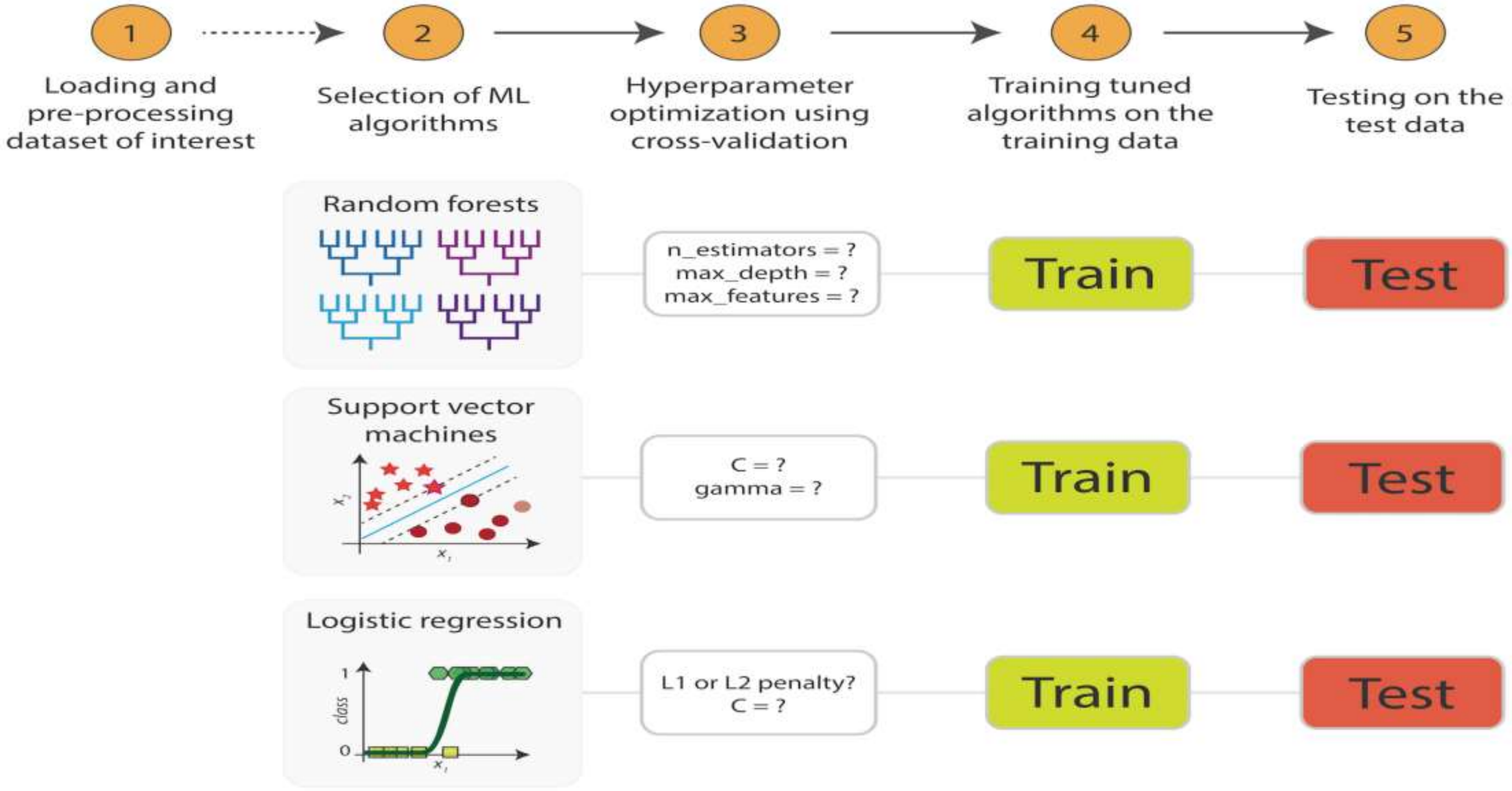
- A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.
- They are often used in processes to help estimate model parameters. They are often specified by the practitioner. They are often tuned for a given predictive modeling problem.
- Some examples of model hyperparameters include:
- The learning rate for training a neural network.
- The C and sigma hyperparameters for support vector machines.
- The k in k-nearest neighbors.

Machine Learning – Step6

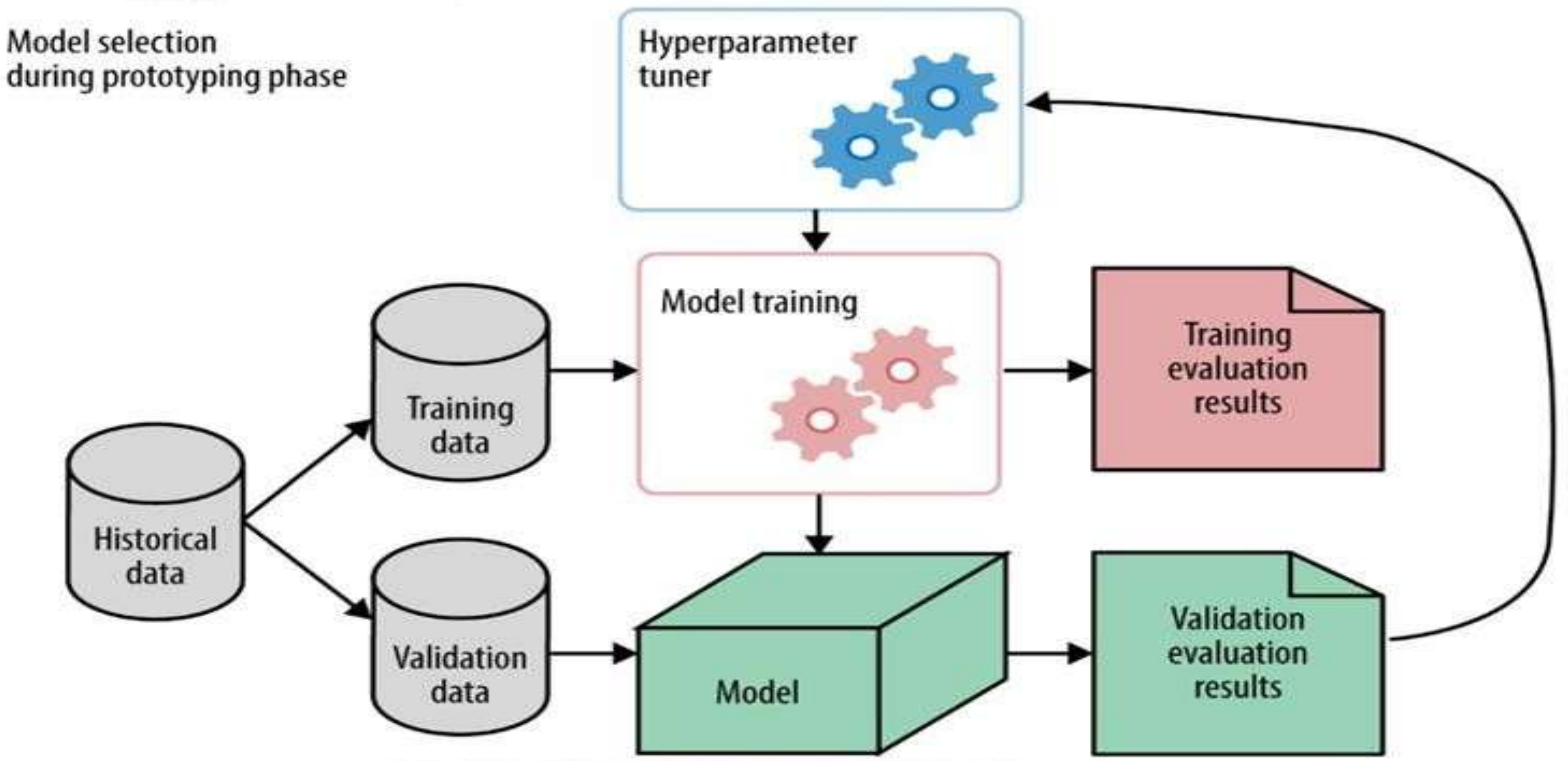
Hyperparameter Tuning

- Hyperparameters are adjustable parameters that let you control the model training process. For example, with neural networks, you decide the number of hidden layers and the number of nodes in each layer. Model performance depends heavily on hyperparameters.
- Hyperparameter tuning, also called hyperparameter optimization, is the process of finding the configuration of hyperparameters that results in the best performance. The process is typically computationally expensive and manual.
- we are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning.

Hyperparameter Tuning

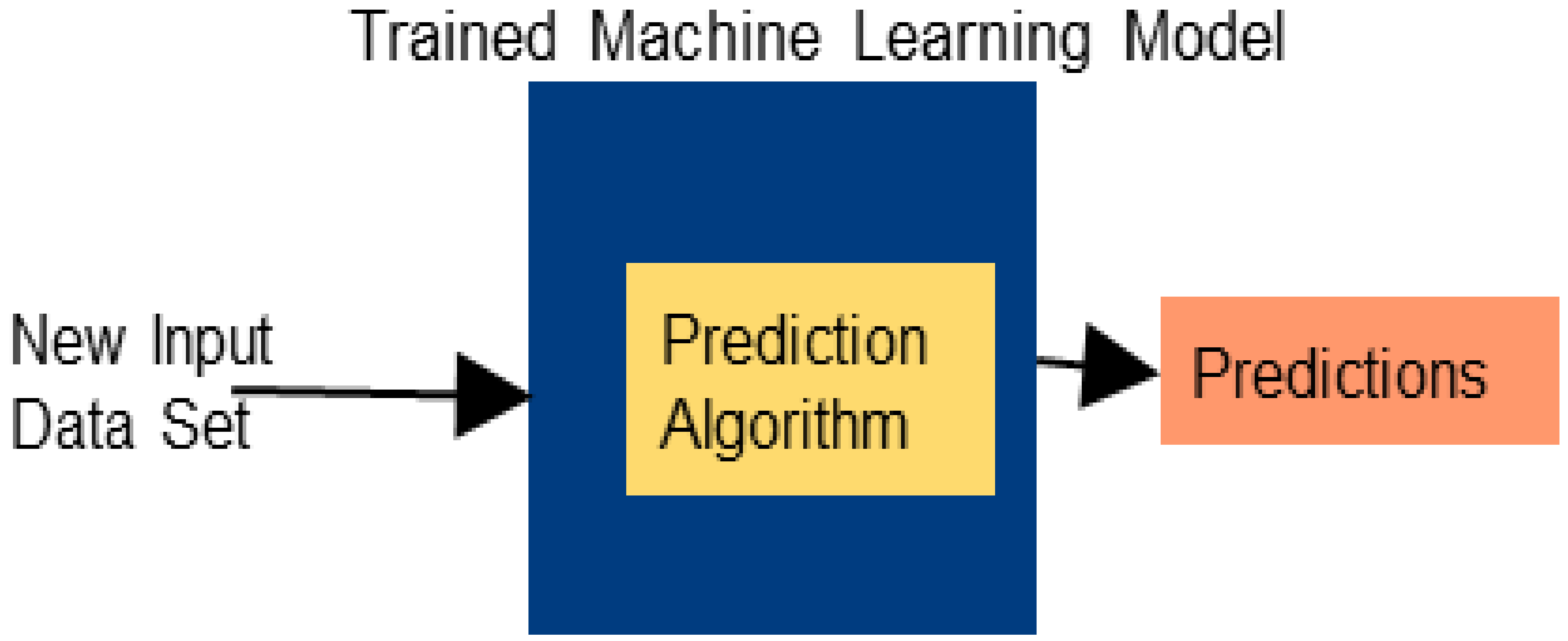


Machine Learning Steps – 6. HyperParameter Tuning

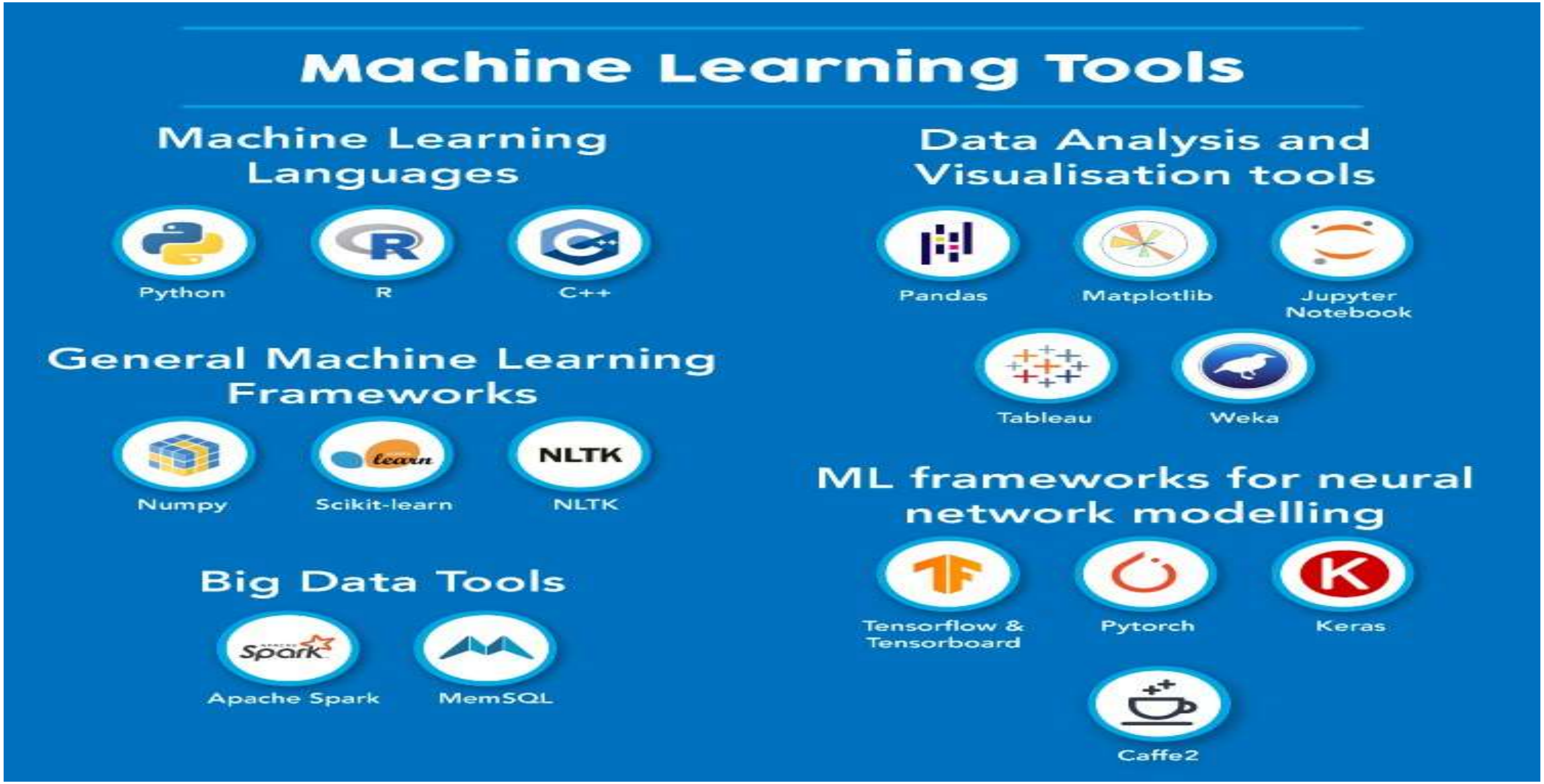


Machine Learning – Step7

Prediction



Machine Learning Tools



Machine Learning Tools

Machine Learning Languages

- Python
- R
- C++

General Machine Learning Frameworks

- Numpy
- Scikit-learn
- NLTK

Big Data Tools

- Apache Spark
- MemSQL

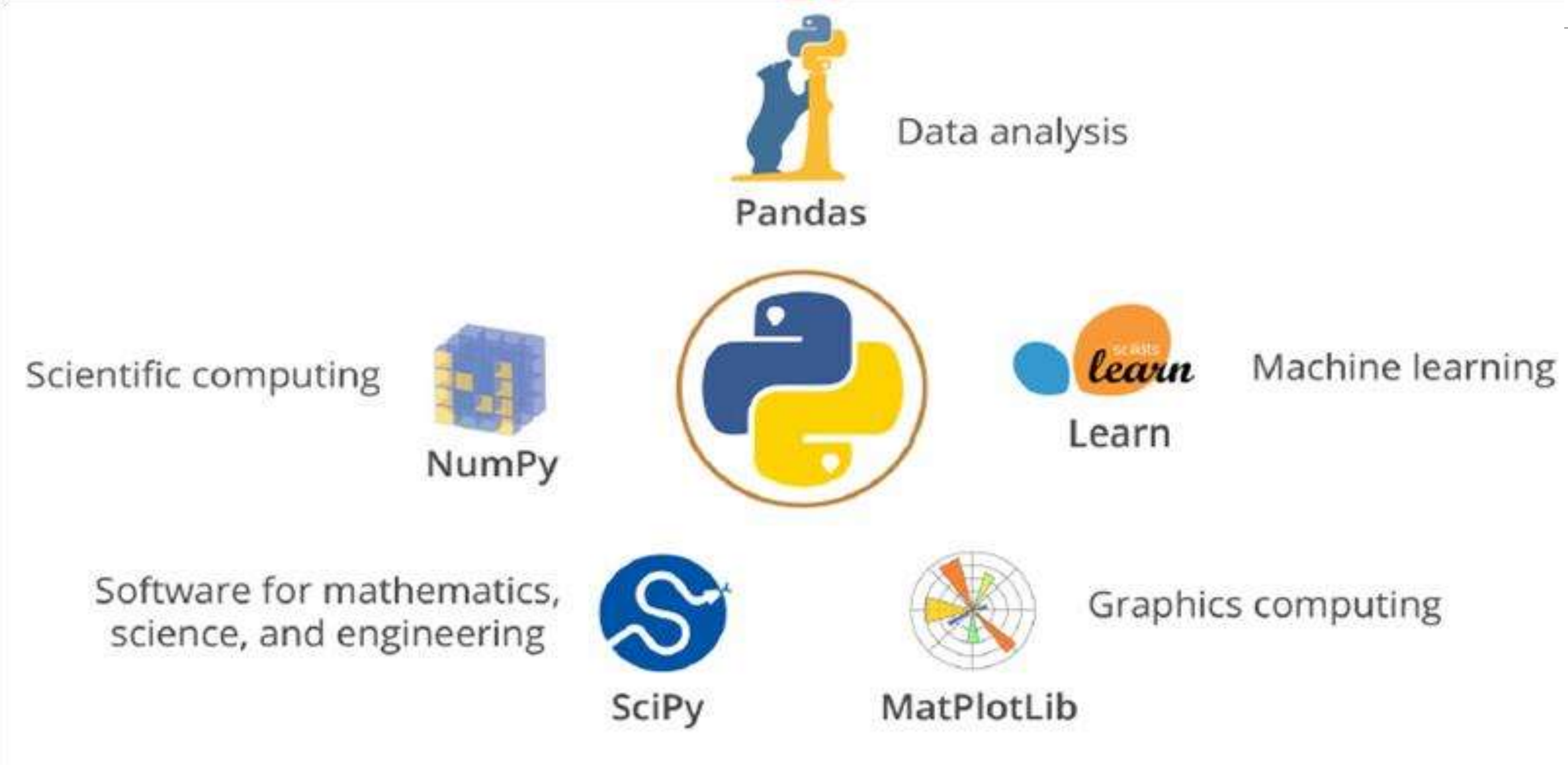
Data Analysis and Visualisation tools

- Pandas
- Matplotlib
- Jupyter Notebook
- Tableau
- Weka

ML frameworks for neural network modelling

- Tensorflow & Tensorboard
- Pytorch
- Keras
- Caffe2

Python Tools and Technologies



Types of Machine Learning Models

Types of Machine Learning Models

- A Machine Learning Model is chosen based on the expected outcome and the input data provided.

- **Supervised machine learning:**

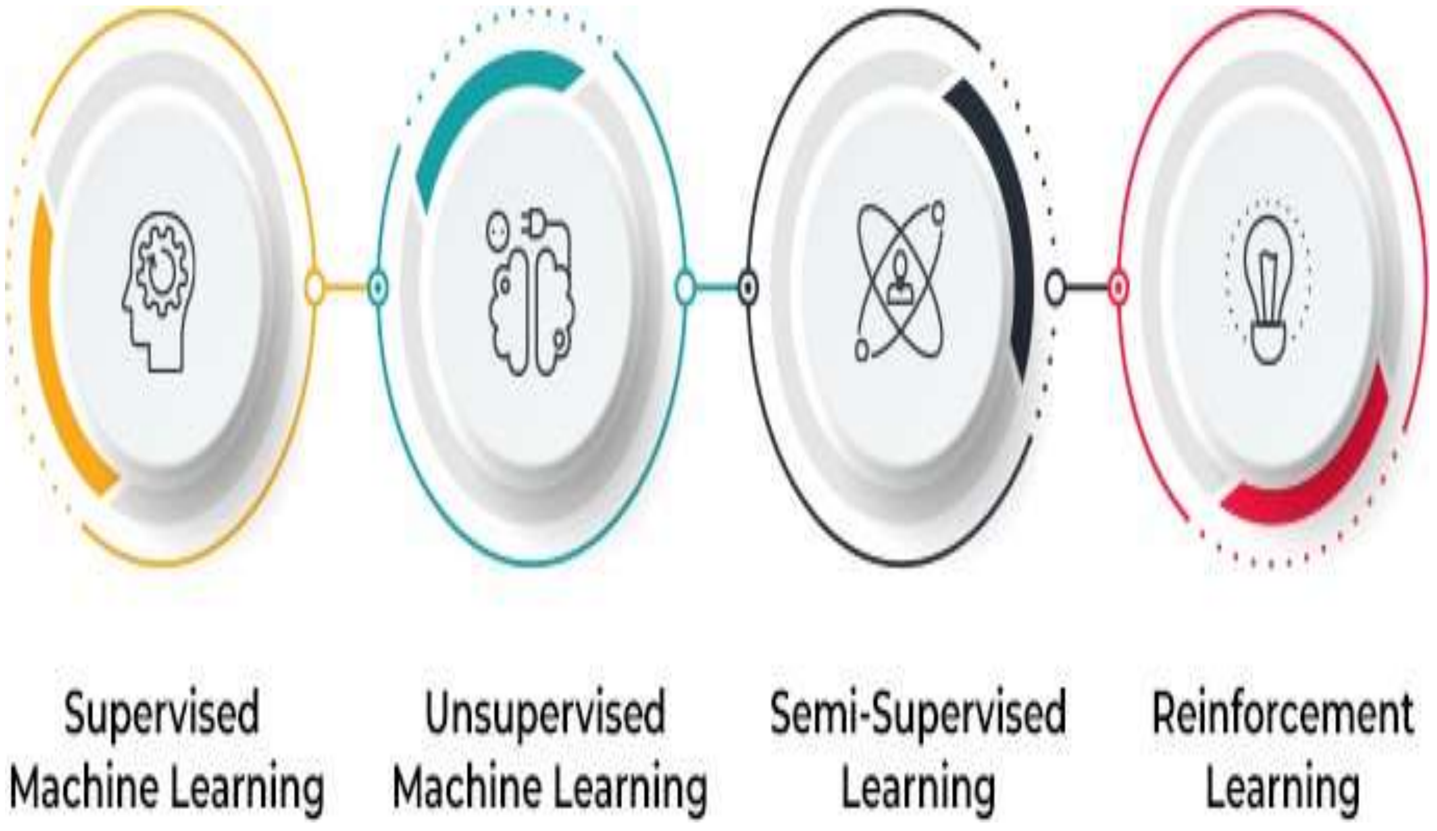
User supervises the machine while training it to work on its own. This requires labeled training data

- **Unsupervised learning:**

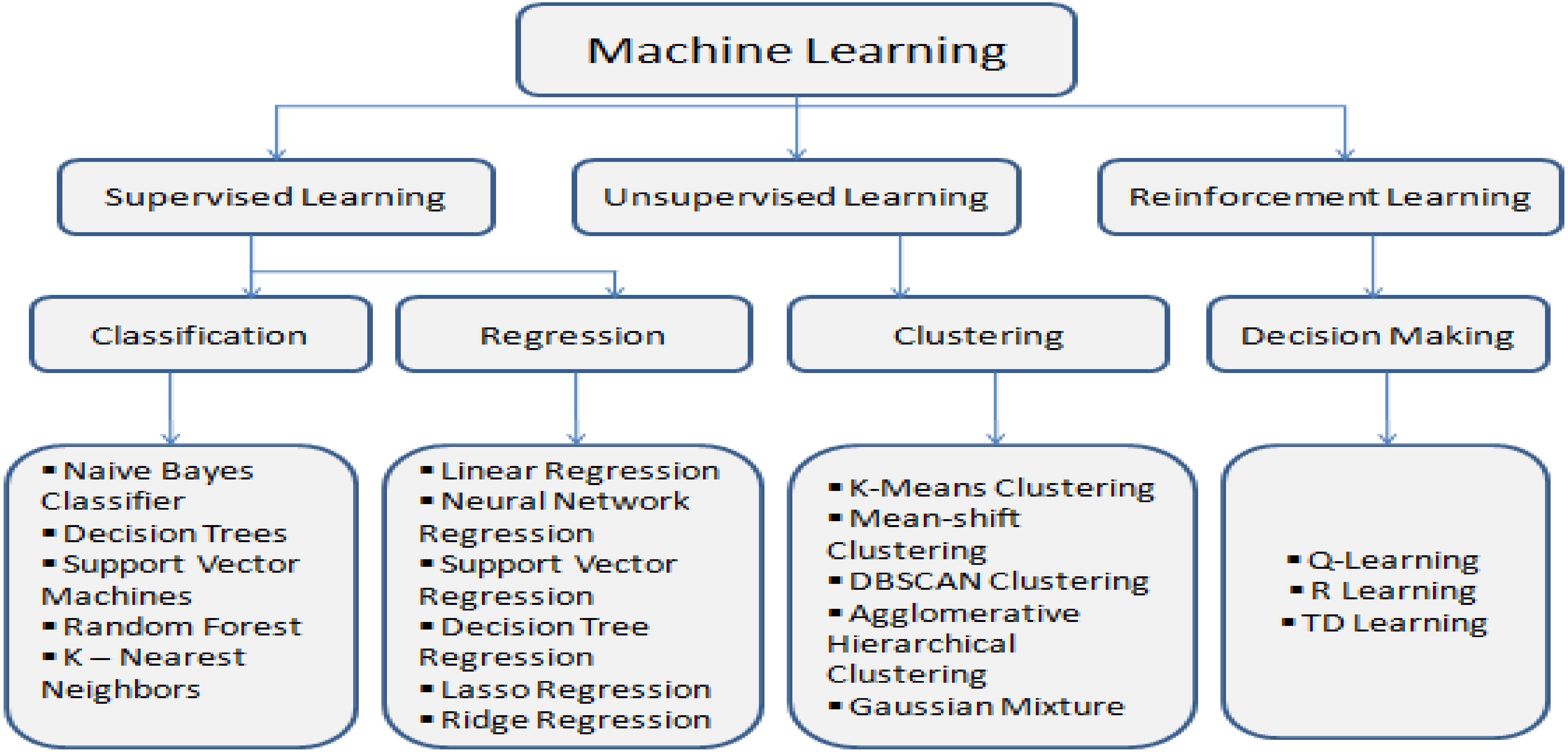
There is training data, but it won't be labeled

- **Reinforcement learning:**

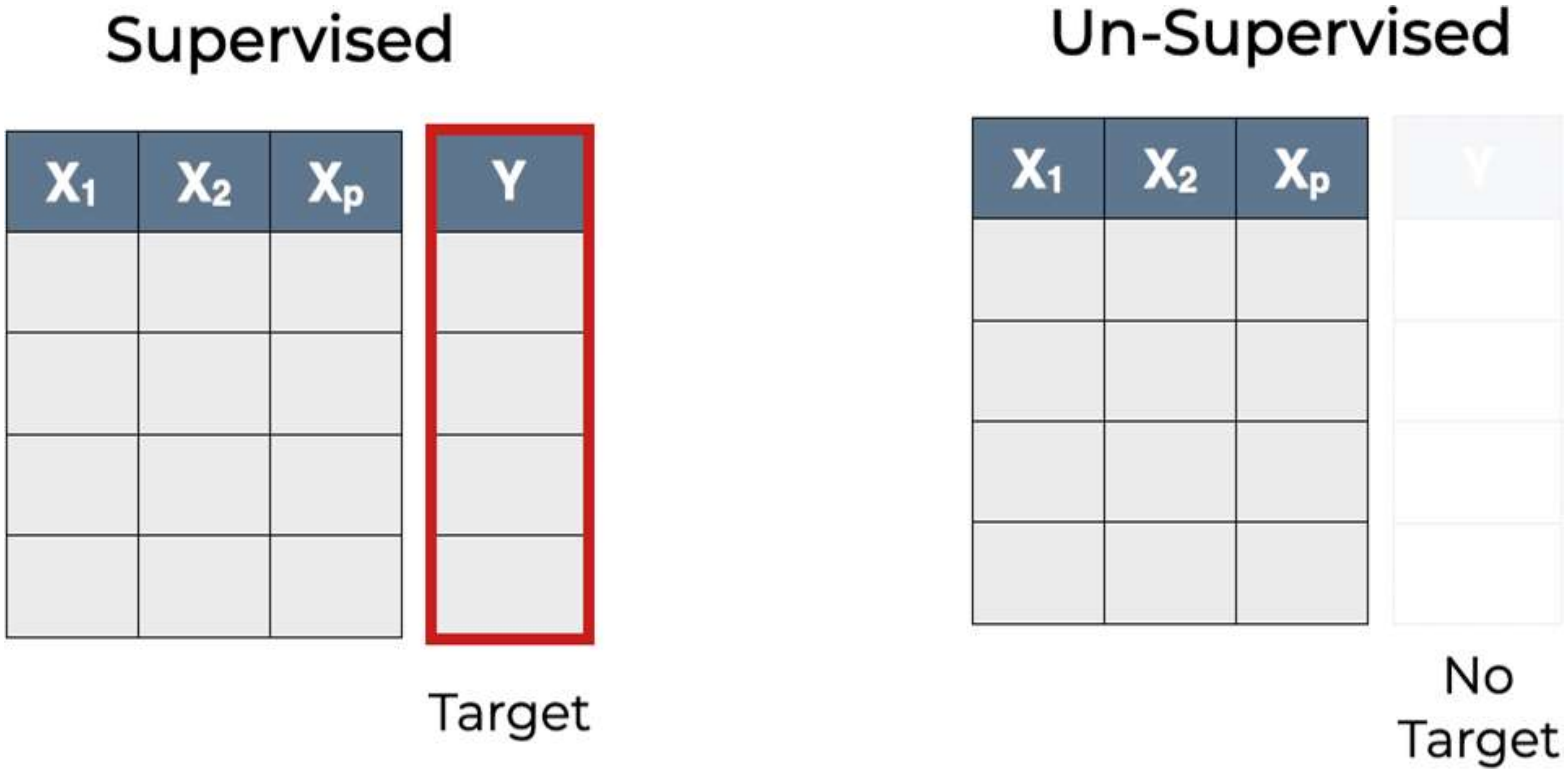
The system learns on its own



Types of Machine Learning



Supervised Vs. Unsupervised Learning



Supervised Learning

- Supervised learning is a type of machine learning that uses labeled data to train machine learning models. In labeled data, the output is already known. The model just needs to map the inputs to the respective outputs.
- Supervised learning algorithms take labeled inputs and map them to the known outputs, which means you already know the target variable.
- An example of supervised learning is to train a system that identifies the image of an animal.
- Supervised learning algorithms are generally used for solving classification and regression problems.
  - Classification -- Predicts a Class Label (Categorical)
  - Regression -- Predicts a Class Label (Numerical)

Supervised Learning

- These models predict the output with respect to the sample data that is provided.
- These are also called **predictive models**.
- The goal is to learn a mapping from inputs 'X' to outputs 'y', given a labeled set of input-output pairs.

$$D = \{(X_i, y_i)\}, \text{ for i=1 to N}$$

D → Training Dataset

X_i → Input

y_i → Output (Categorical or Nominal)

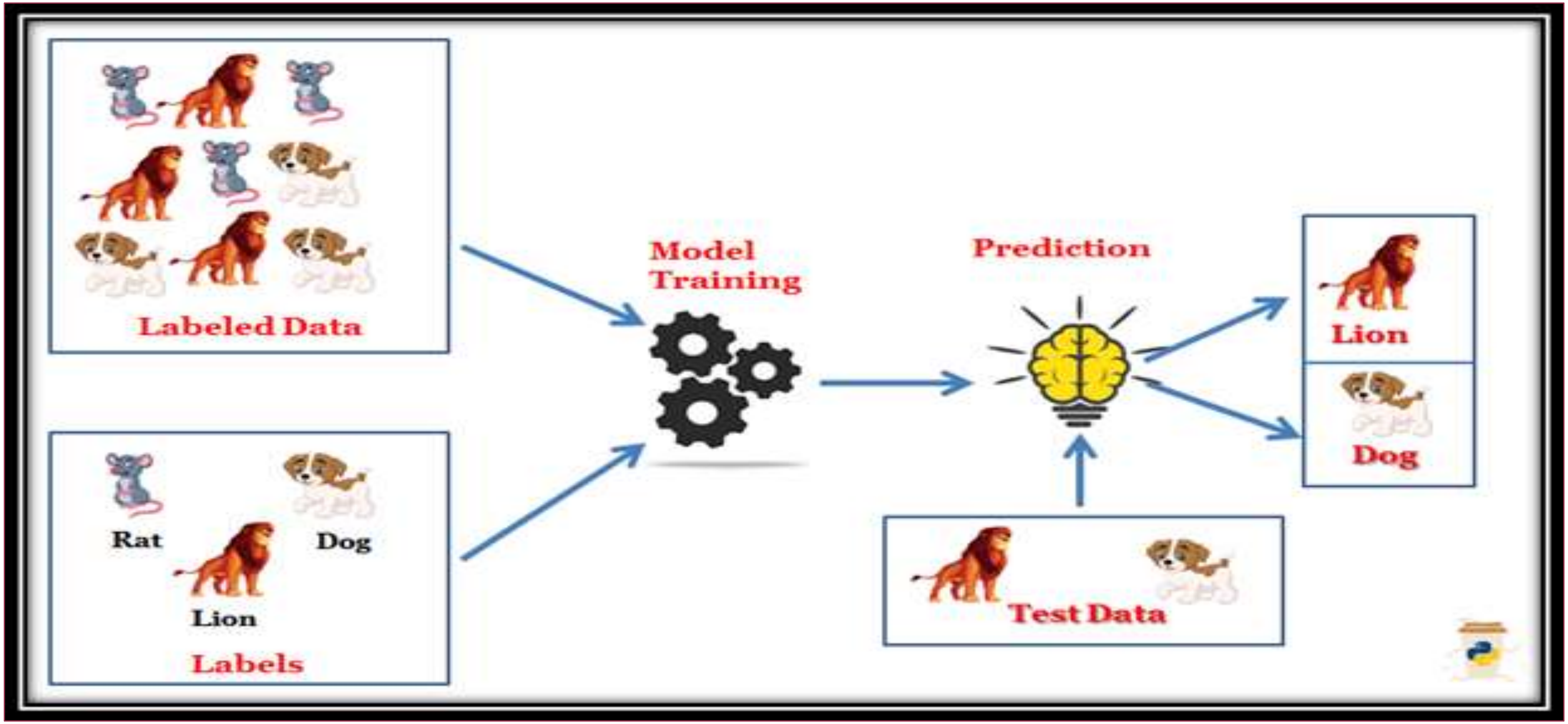
N → Number of Training examples

Eg: D_{dim}= {#rooms, #floor,area,amenities}

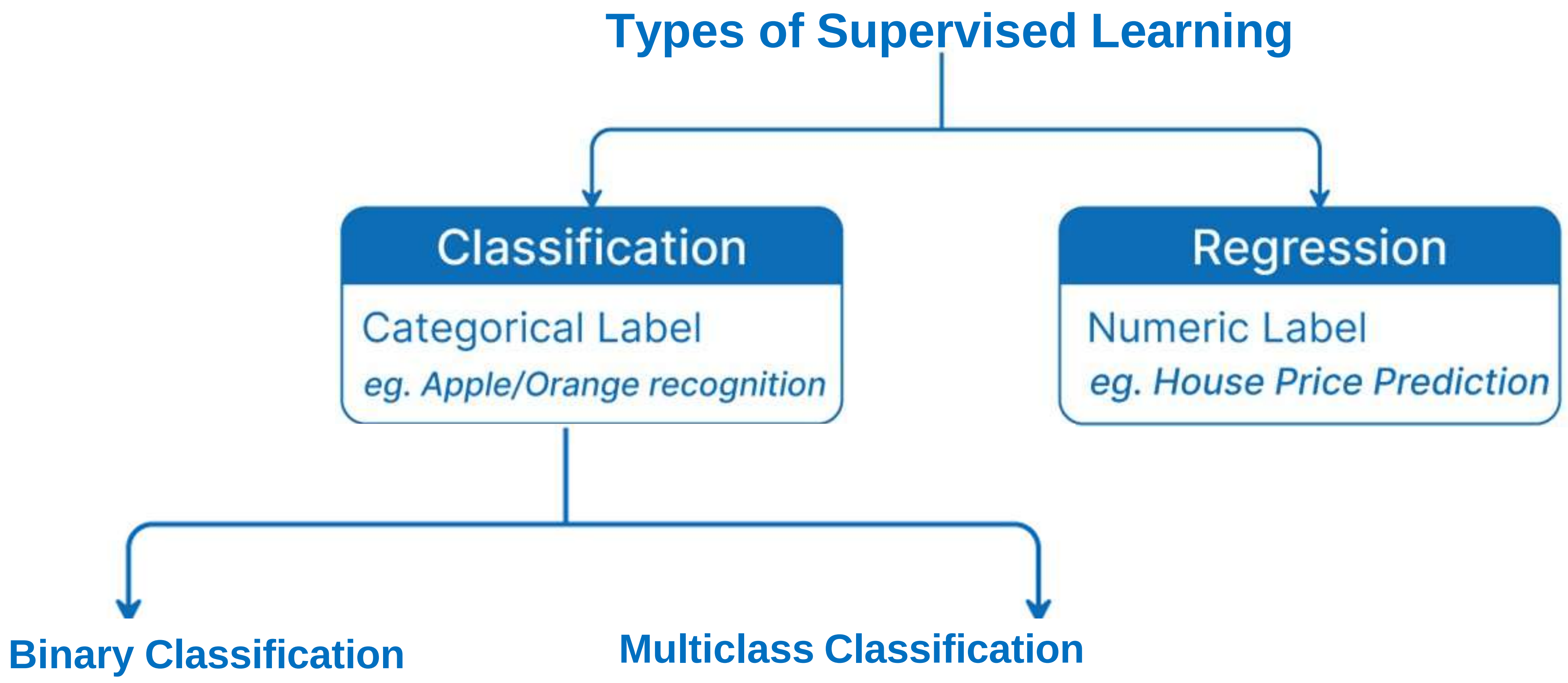
X_i = {2,1,1020sft,Yes}

y_i = {60 Lakhs}

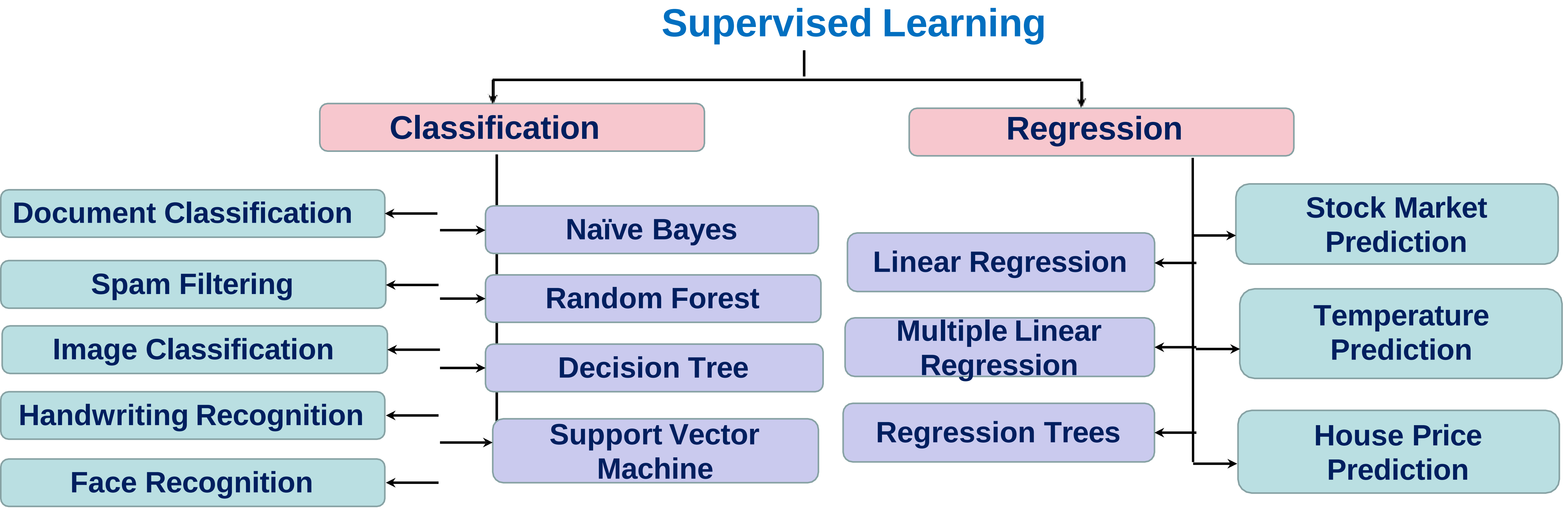
Supervised Learning



Supervised Learning


Types of Supervised Learning
Classification
Categorical Label
eg. Apple/Orange recognition
Regression
Numeric Label
eg. House Price Prediction
Binary Classification
Multiclass Classification

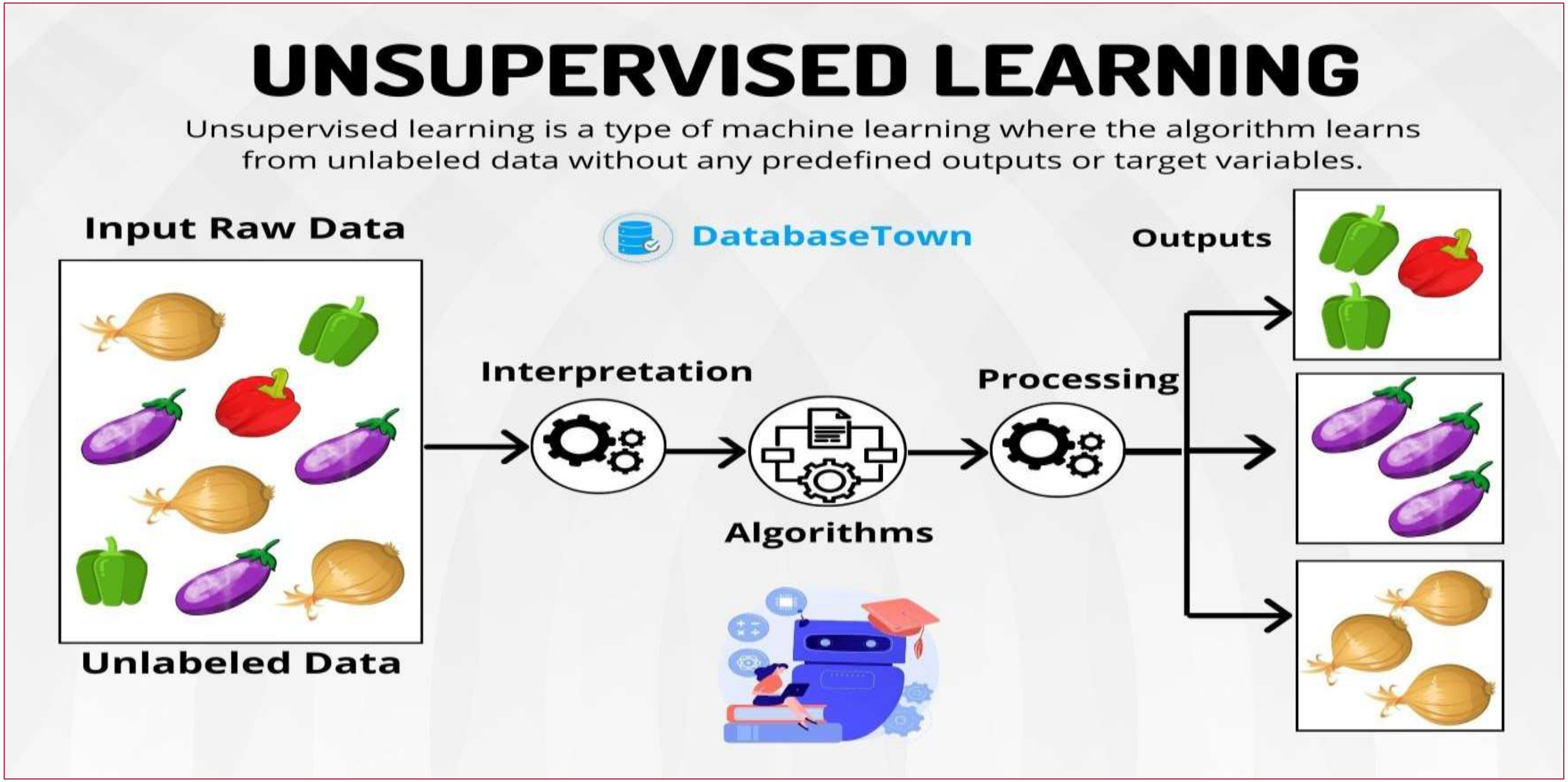
Supervised Learning – Applications & Algorithms



Unsupervised Learning

- Unsupervised learning is a type of machine learning that uses unlabeled data to train machines. Unlabeled data doesn't have a fixed output variable.
- The model learns from the data, discovers the patterns and features in the data, and returns the output.
- Ex:- Flipkart uses this model to find and recommend products that are well suited for you.
- Unsupervised learning finds patterns and understands the trends in the data to discover the output. So, the model tries to label the data based on the features of the input data.
- The training process used in unsupervised learning techniques does not need any supervision to build models. They learn on their own and predict the output.
- Unsupervised learning can be further grouped into types:
  - Clustering
  - Association

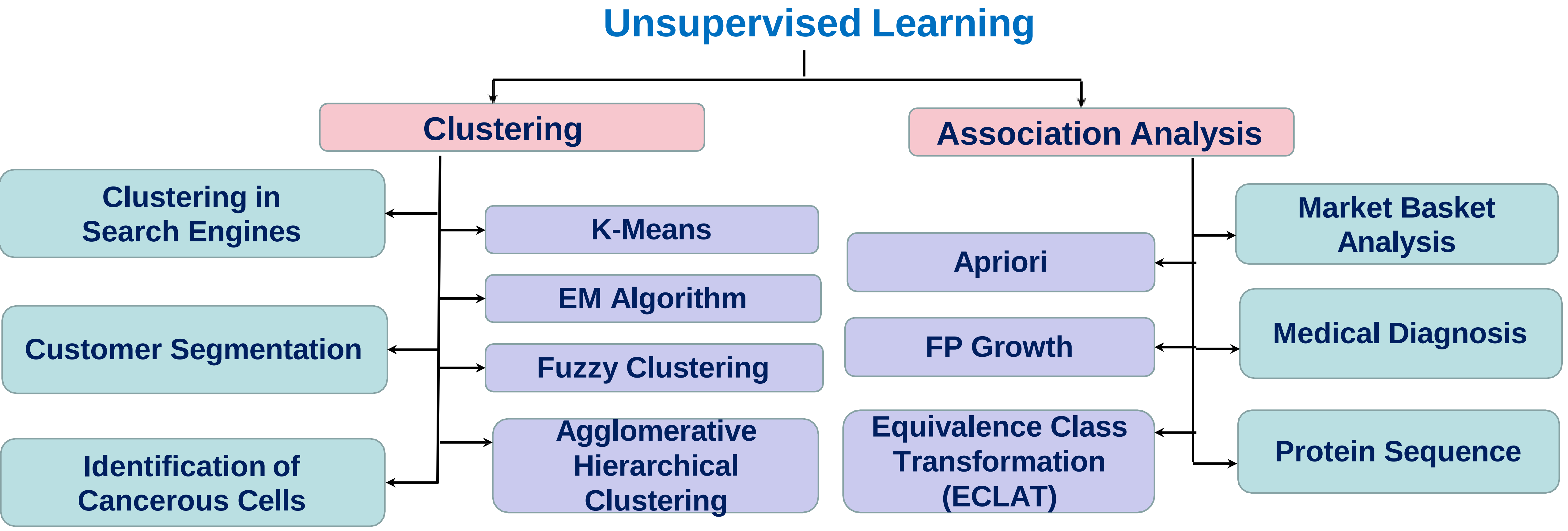
Unsupervised Learning


UNSUPERVISED LEARNING
Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.
Input Raw Data
DatabaseTown
Outputs
Interpretation
Processing
Algorithms
Unlabeled Data

Unsupervised Learning

- The goal is to discover interesting patterns in the data, called as knowledge discovery.
- It performs the following tasks:
  - Exploratory Data Analysis
  - Clustering
  - Dimensionality Reduction
  - Feature Ranking

Unsupervised Learning – Applications & Algorithms

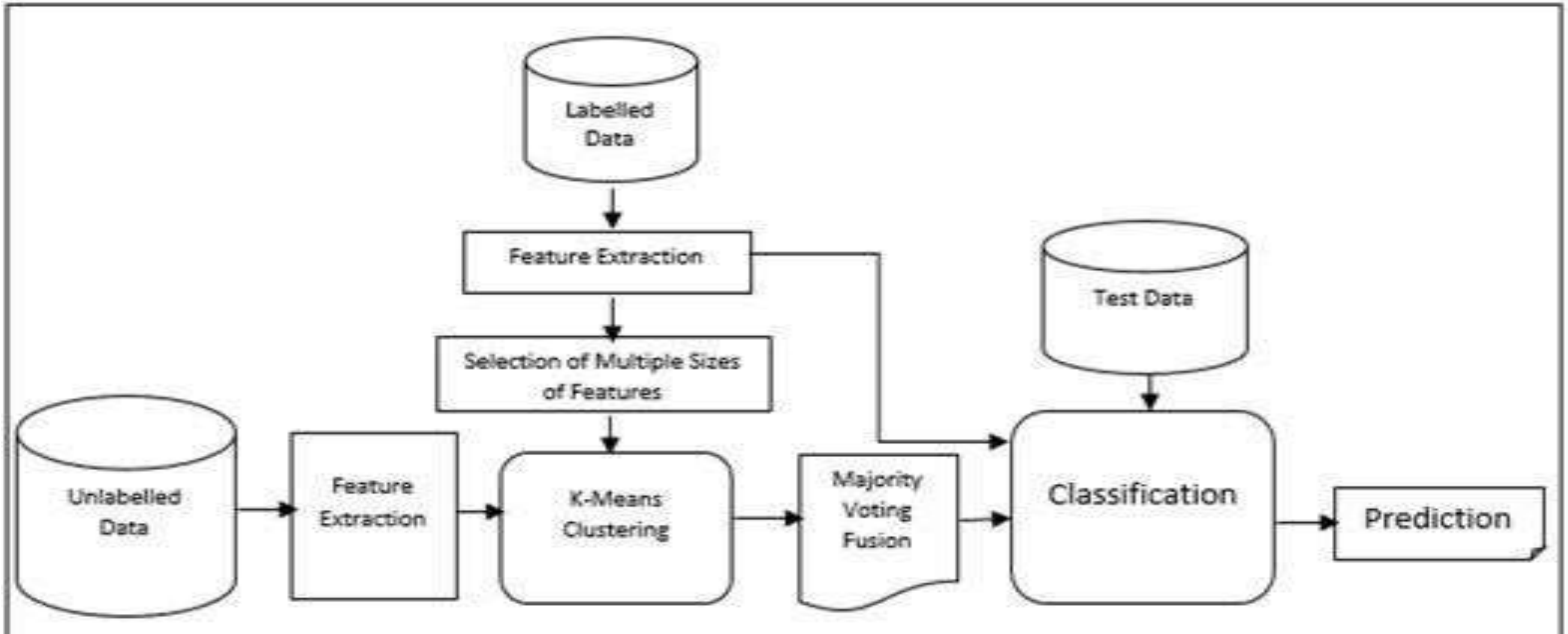


Semi-supervised Learning

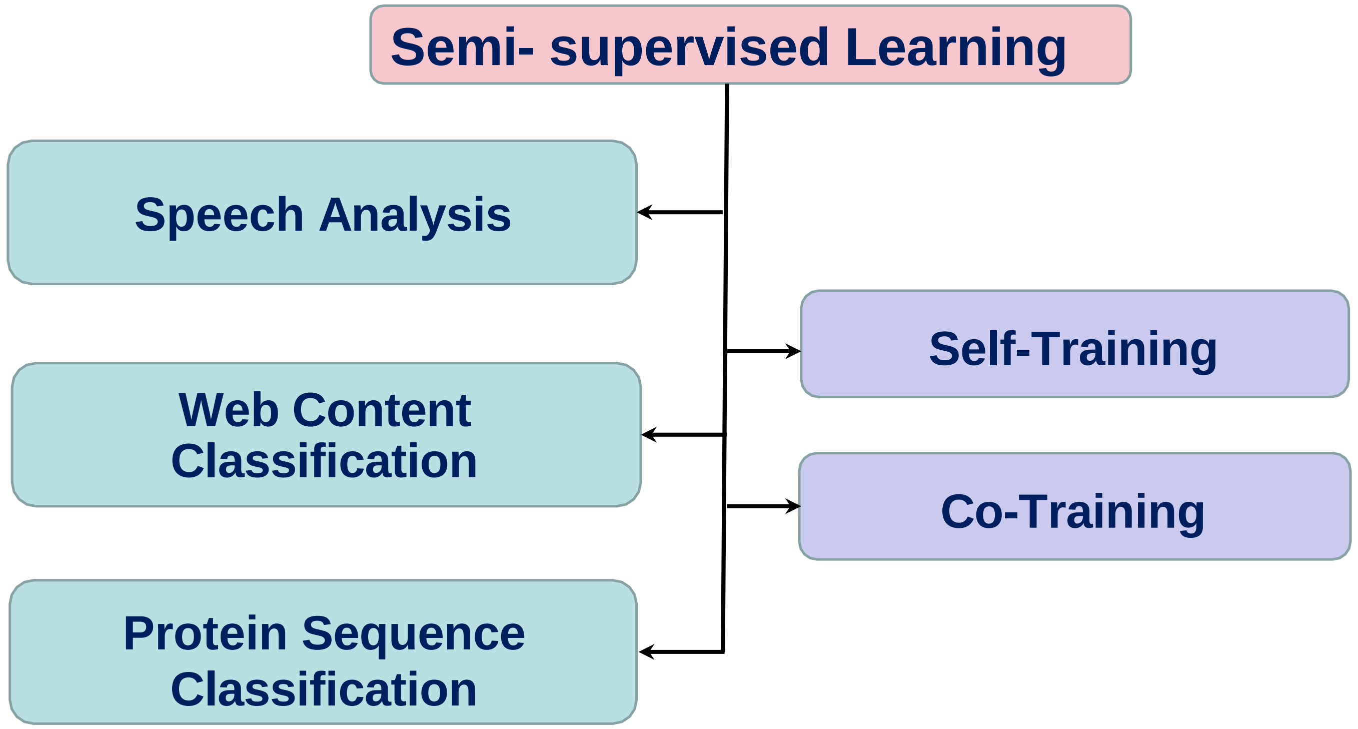
- Semi-supervised learning (SSL) is a machine learning technique that uses a small portion of labeled data and lots of unlabeled data to train a predictive model.
- Semi-supervised learning is a branch of machine learning that combines supervised and unsupervised learning by using both labeled and unlabeled data to train artificial intelligence (AI) models for classification and regression tasks.
- Semi-supervised learning methods are especially relevant in situations where obtaining a sufficient amount of labeled data is prohibitively difficult or expensive, but large amounts of unlabeled data are relatively easy to acquire. In such scenarios, neither fully supervised nor unsupervised learning methods will provide adequate solutions.
- Semi-supervised learning bridges supervised learning and unsupervised learning techniques to solve their key challenges. With semi-supervised learning, you train an initial model on a few labeled samples and then iteratively apply the model to a large dataset.

Semi-supervised Learning

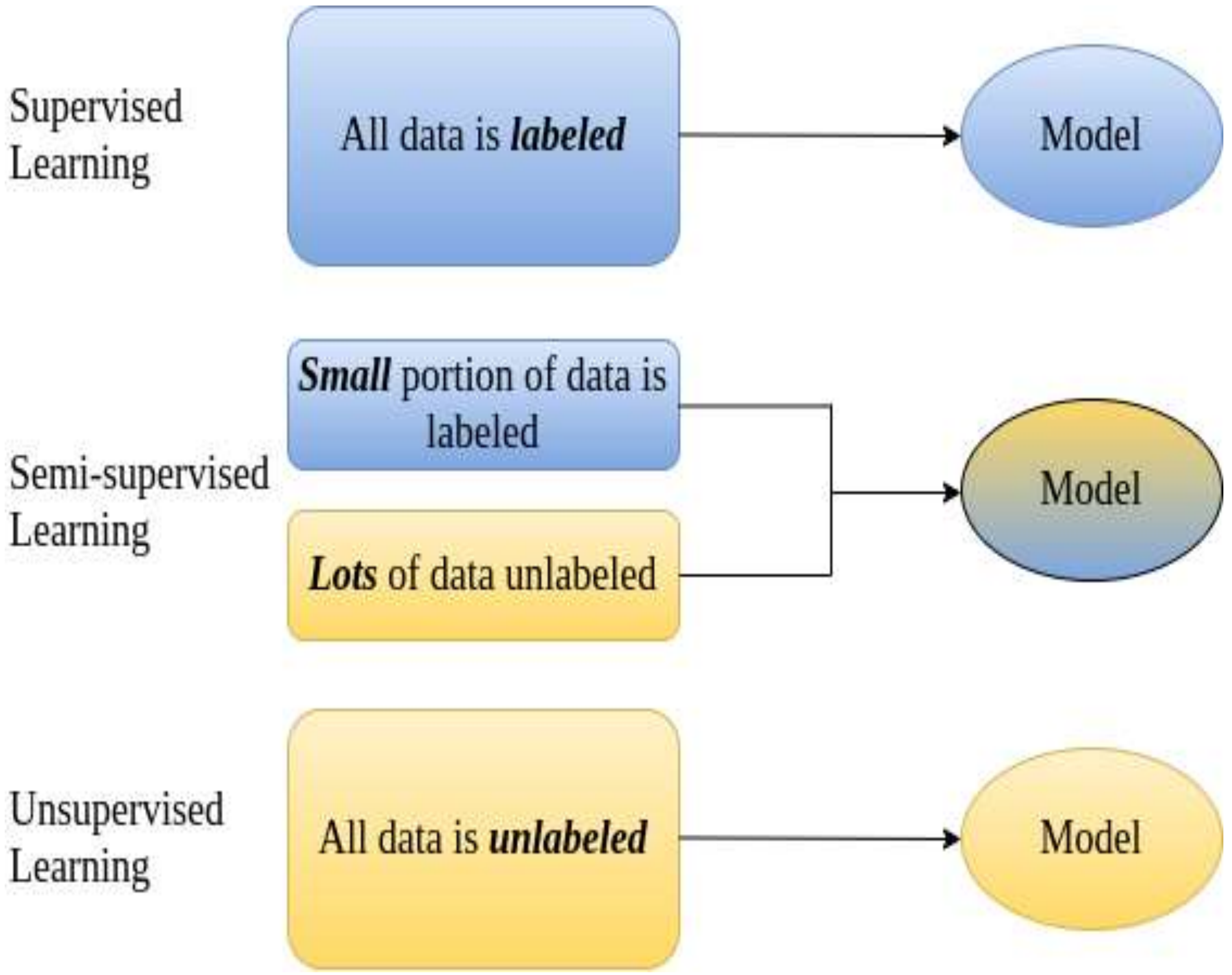
- Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data.



Semi-supervised Learning: Applications & Algorithms


Semi- supervised Learning
Speech Analysis
Web Content Classification
Protein Sequence Classification
Self-Training
Co-Training

supervised / semi-supervised / unsupervised Learning



Examples:-

- Image Classification
- Sentiment Analysis
- Document Clustering
- Medical Diagnosis
- Customer Segmentation
- Social Network Analysis
- Satellite Image Segmentation

Reinforcement Learning

- Reinforcement learning is a type of Machine Learning that trains a model to return an optimum solution for a problem by taking a sequence of decisions by itself.
- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions.
- For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.

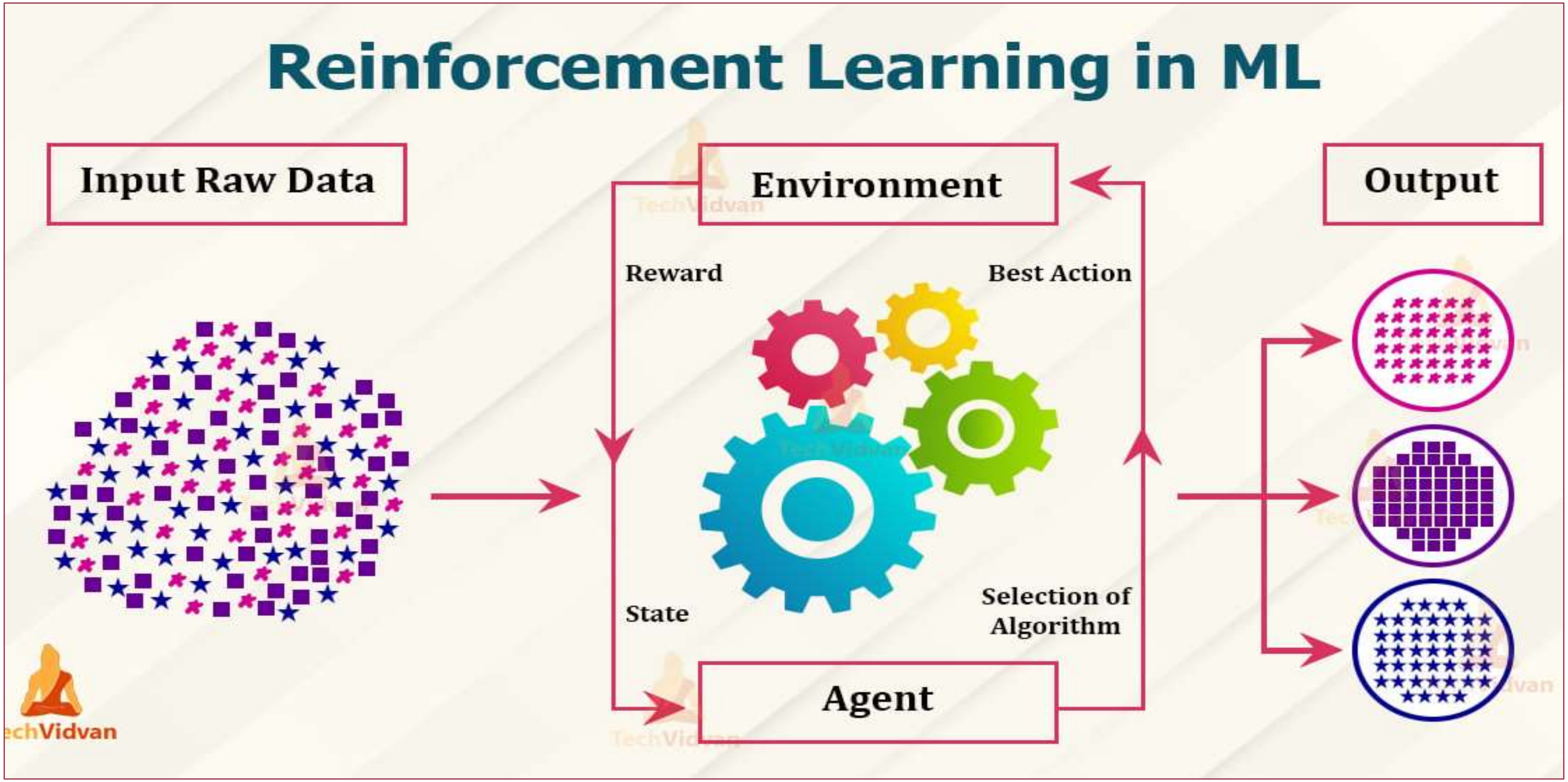
Reinforcement Learning

- Reinforcement Learning solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way.
- Hence, we can say that *"**Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that**."* How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- **Ex:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback

Reinforcement Learning

- Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that.
- The agent is not instructed about the environment and what actions need to be taken. It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action. The agent may get a delayed reward.
- The following two steps are used to train the target:
  - For training, the agent is provided with the contextual information about the environment and choices. With this knowledge, the agent interacts with the environment and make decisions.
  - The agent is provided a reward/ feedback based on how well the agent has made the decision resulted in achieving the desired goal.

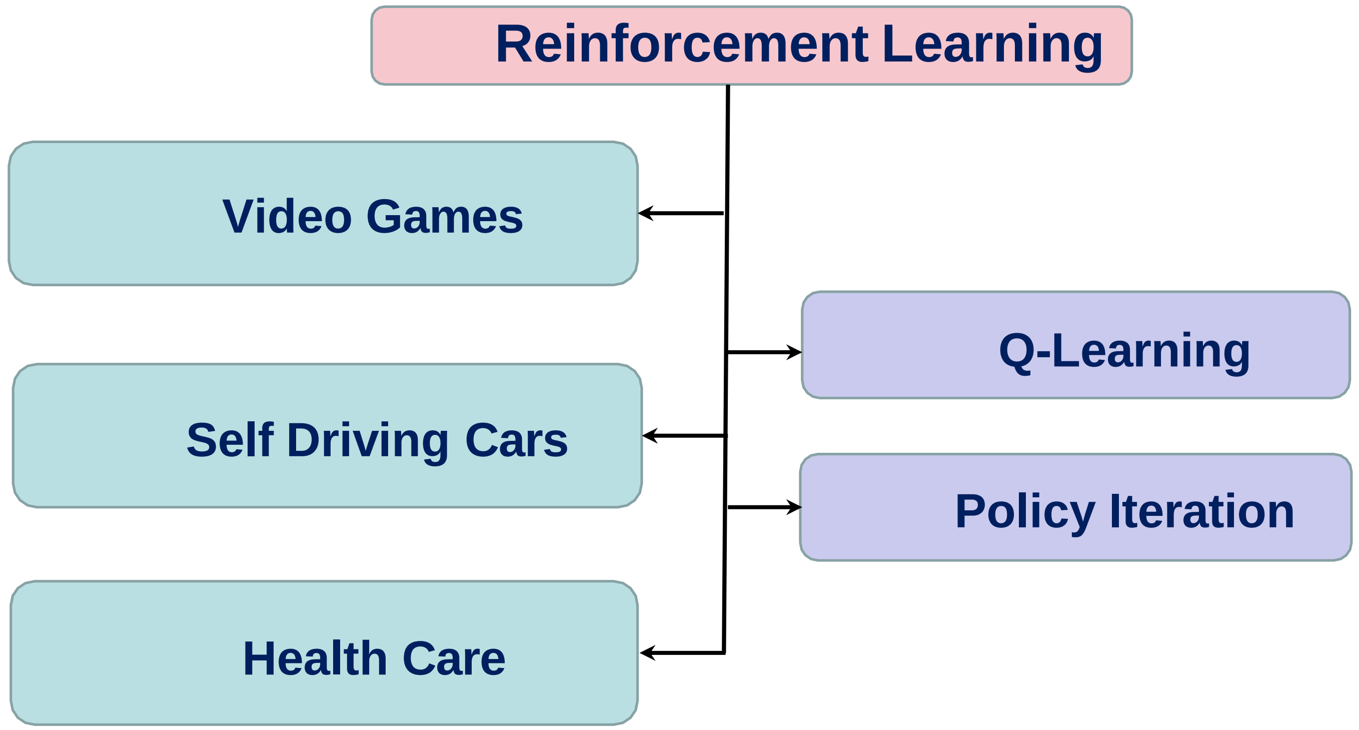
Reinforcement Learning



Reinforcement Learning

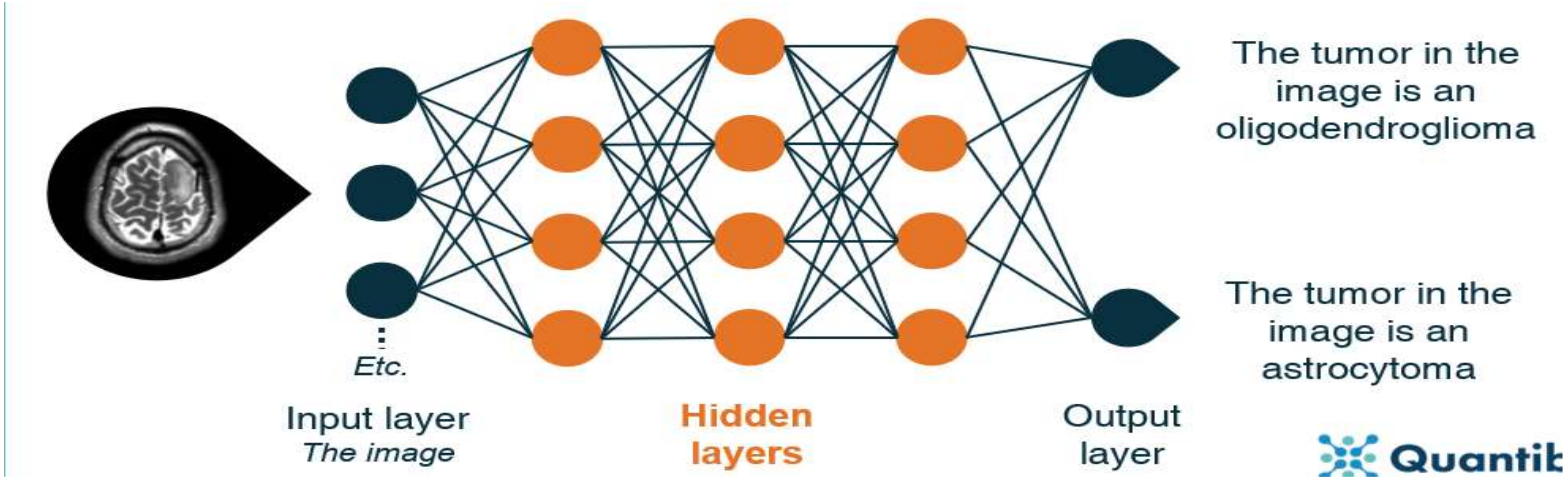
•Agent():	An entity that can perceive/explore the environment and act upon it.
•Environment():	A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
•Action():	Actions are the moves taken by an agent within the environment.
•State():	State is a situation returned by the environment after each action taken by the agent.
•Reward():	A feedback returned to the agent from the environment to evaluate the action of the agent.
•Policy():	Policy is a strategy applied by the agent for the next action based on the current state.
•Value():	It is expected long-term retuned with the discount factor and opposite to the short-term reward.
•Q-value():	It is mostly similar to the value, but it takes one additional parameter as a current action(a)

Reinforcement Learning: Applications & Algorithms



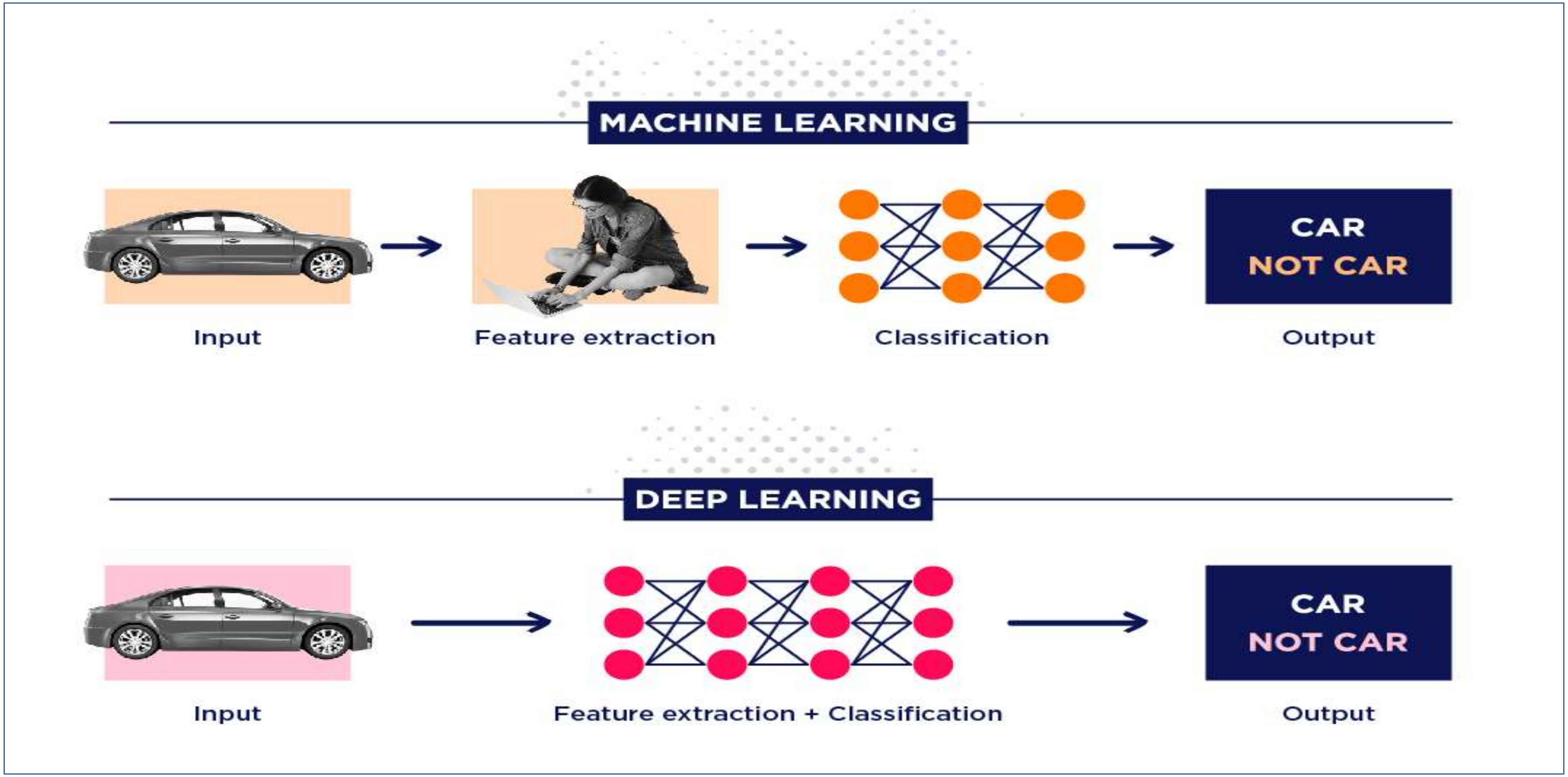
Deep Learning

- **Deep Learning** is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. it makes use of deep neural networks.
- **Deep learning** mimics the network **of neurons in a brain.**
- It is a subset of Machine Learning, these algorithms are constructed with connected layers.



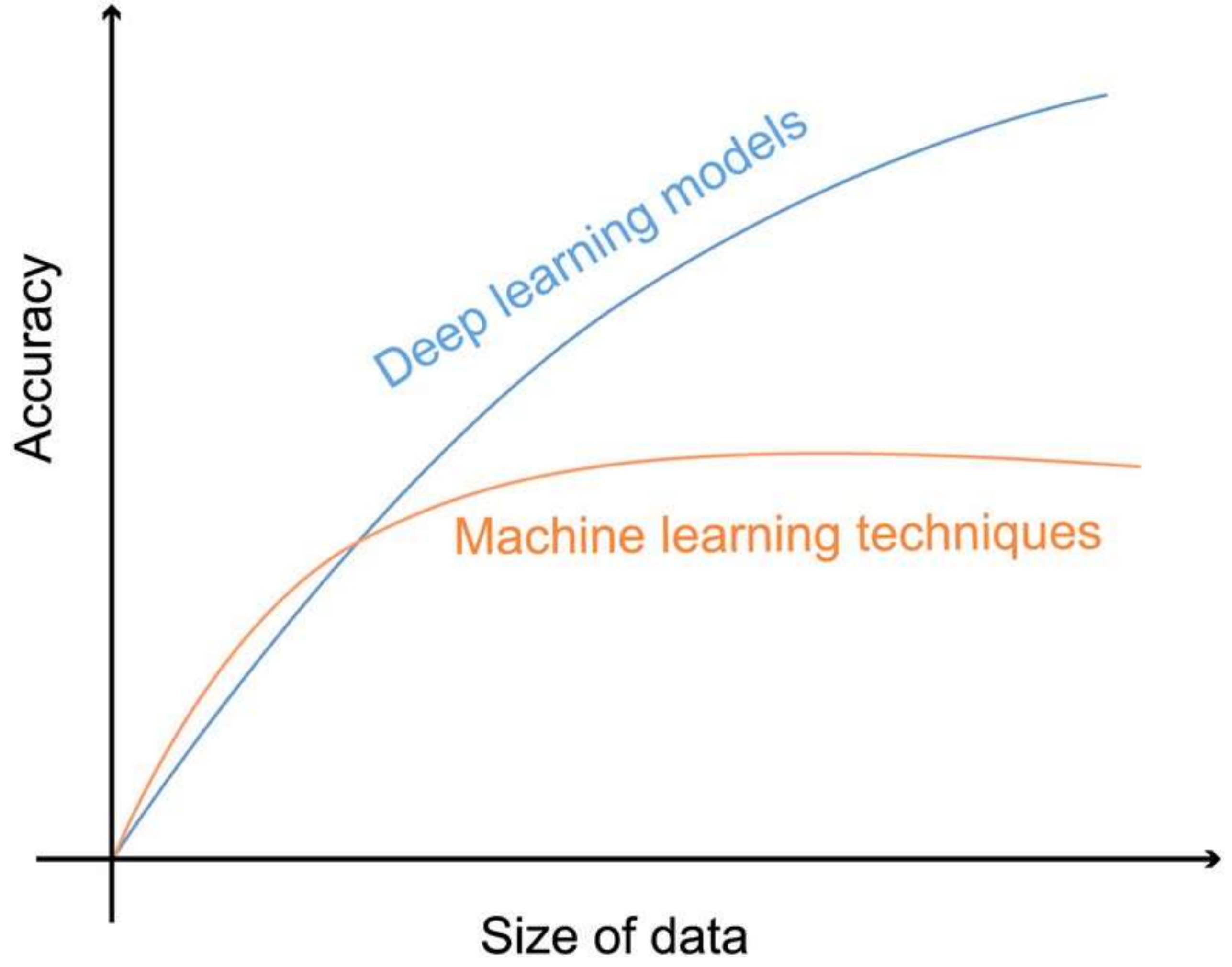
Etc.
Input layer
The image
Hidden layers
Output layer
The tumor in the image is an oligodendroglioma
The tumor in the image is an astrocytoma
Quantib

Deep Learning



Selection of Machine Learning Algorithms

- The selection of Machine Learning algorithms depend on the following parameters:
  - Type of Data
  - Expected Outcome
  - Accuracy
  - Training time
  - Linearity
  - Number of Parameters
  - Number of Features



Concept Learning

- In Machine Learning, concept learning can be termed as "*a problem of searching through a predefined space of potential hypothesis for the hypothesis that best fits the training examples" – Tom Mitchell.*

- Most of human learning is based on past instances or experiences.

- For ex, we can identify any type of vehicle based on a certain set of features like make, model, etc., that are defined over a large set of features.

- These special features differentiate the set of cars, trucks, etc from the larger set of vehicles, are known as concepts.

- Similarly, machines can also learn from concepts to identify whether an object belongs to a specific category or not.

Concept Learning

- Any algorithm that supports concept learning requires the following:

  - Training Data
  - Target Concept
  - Actual Data Objects

- In order to understand Find-S algorithm, we have a basic idea of the following concepts as well:

  - Concept Learning
  - General Hypothesis
  - Specific Hypothesis

Concept Learning using find-S algorithm

General Hypothesis

Hypothesis, in general, is an explanation for something. The general hypothesis basically states the general relationship between the major variables.
For example, a general hypothesis for ordering food would be *I want a burger.*
G = { '?', '?', '?', …..'?'}

Specific Hypothesis

The specific hypothesis fills in all the important details about the variables given in the general hypothesis. The more specific details into the example given above would be *I want a cheeseburger with a chicken pepper on filling with a lot of lettuce.* S = {'Φ','Φ','Φ', ……,'Φ'}

Concept Learning using find-S algorithm

The **Find-S algorithm** operates on a hypothesis space to find a general hypothesis that accurately represents the target concept based on labeled training data.

The Find-S algorithm follows the steps given below:

1. Initialize ‘h’ to the most specific hypothesis.

2. The Find-S algorithm only considers the positive examples and eliminates negative examples.

3. For each positive example, the algorithm checks for each attribute in the example. If the attribute value is the same as the hypothesis value, the algorithm moves on without any changes.

4. But if the attribute value is different than the hypothesis value, the algorithm changes it to ‘?’.

Concept Learning using find-S algorithm

1.The process starts with initializing ‘h’ with the most specific hypothesis, generally, it is the first positive example in the data set.

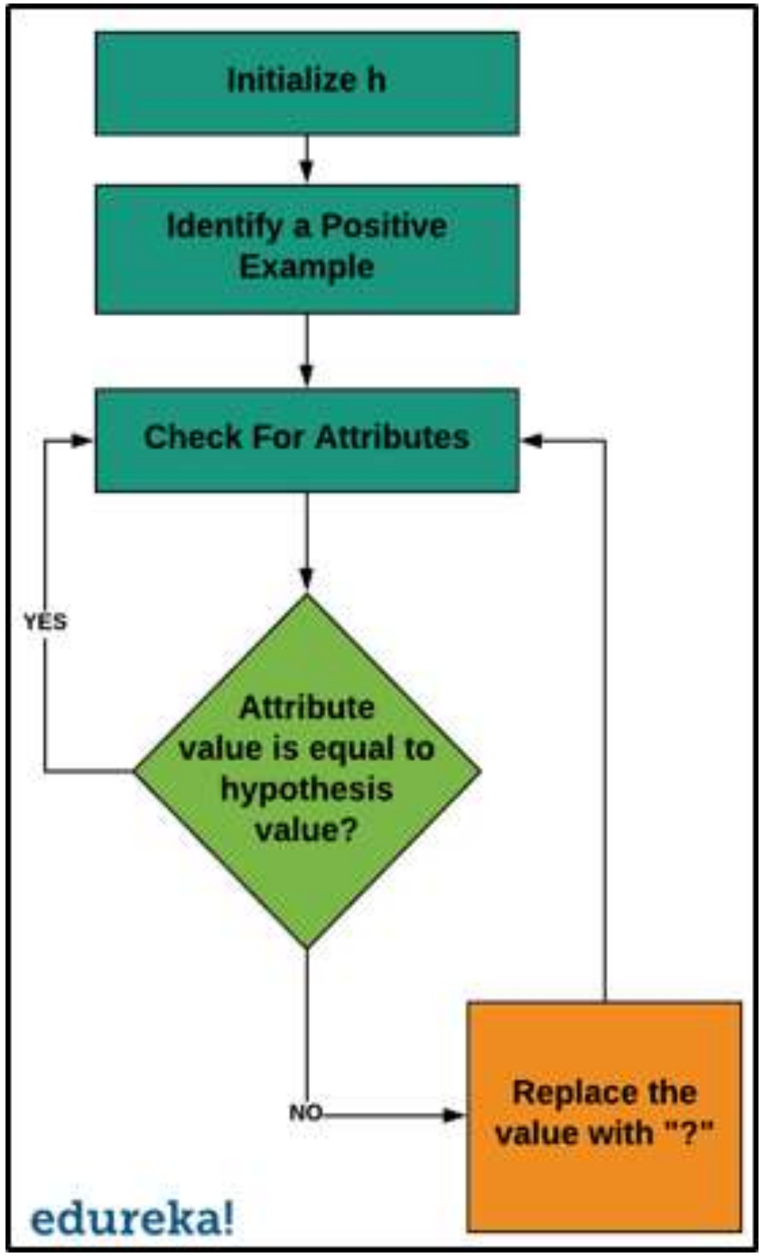
2.We check for each positive example. If the example is negative, we will move on to the next example but if it is a positive example we will consider it for the next step.

3. We will check if each attribute in the example is equal to the hypothesis value.

4. **If the value matches, then no changes are made.**

5. **If the value does not match, the value is changed to ‘?’.**

6.We do this until we reach the last positive example in the data set.


Initialize h
Identify a Positive Example
Check For Attributes
YES
Attribute value is equal to hypothesis value?
NO
Replace the value with "?"
edureka!

Symbols to represent concepts and operations

Ø (Empty Set) – This symbol represents the absence of any specific value or attribute. It is often used to initialize the hypothesis as the most specific concept.

? (Don't Care) – The question mark symbol represents a "don't care" or "unknown" value for an attribute. It is used when the hypothesis needs to generalize over different attribute values that are present in positive examples.

Positive Examples (+) – The plus symbol represents positive examples, which are instances labeled as the target class or concept being learned.

Negative Examples (-) – The minus symbol represents negative examples, which are instances labeled as non-target classes or concepts that should not be covered by the hypothesis.

Hypothesis (h) – The variable h represents the hypothesis, which is the learned concept or generalization based on the training data. It is refined iteratively throughout the algorithm.

Concept Learning using find-S algorithm

Ex:- How to implement it to a smaller data set with a bunch of examples to decide if a person wants to go for a walk. **The concept of this particular problem will be on what days does a person likes to go on walk.**

Time	Weather	Temperature	Company	Humidity	Wind	Goes
Morning	Sunny	Warm	Yes	Mild	Strong	Yes
Evening	Rainy	Cold	No	Mild	Normal	No
Morning	Sunny	Moderate	Yes	Normal	Normal	Yes
Evening	Sunny	Cold	Yes	High	Strong	Yes

There are six attributes and a final attribute that defines the **positive** or **negative** example. In this case, **yes is a positive example**, which means the person will go for a walk.

Concept Learning using find-S algorithm

The Specific Hypothesis is:

h = {‘Morning’, ‘Sunny’, ‘Warm’, ‘Yes’, ‘Mild’, ‘Strong’}

This is our initial hypothesis, and now we will consider each example one by one, but only the positive examples.

h = {‘Morning’, ‘Sunny’, ‘?’, ‘Yes’, ‘?’, ‘?’}

h = {‘?’, ‘Sunny’, ‘?’, ‘Yes’, ‘?’, ‘?’}

We replaced all the different values in the specific hypothesis to get a resultant general hypothesis.

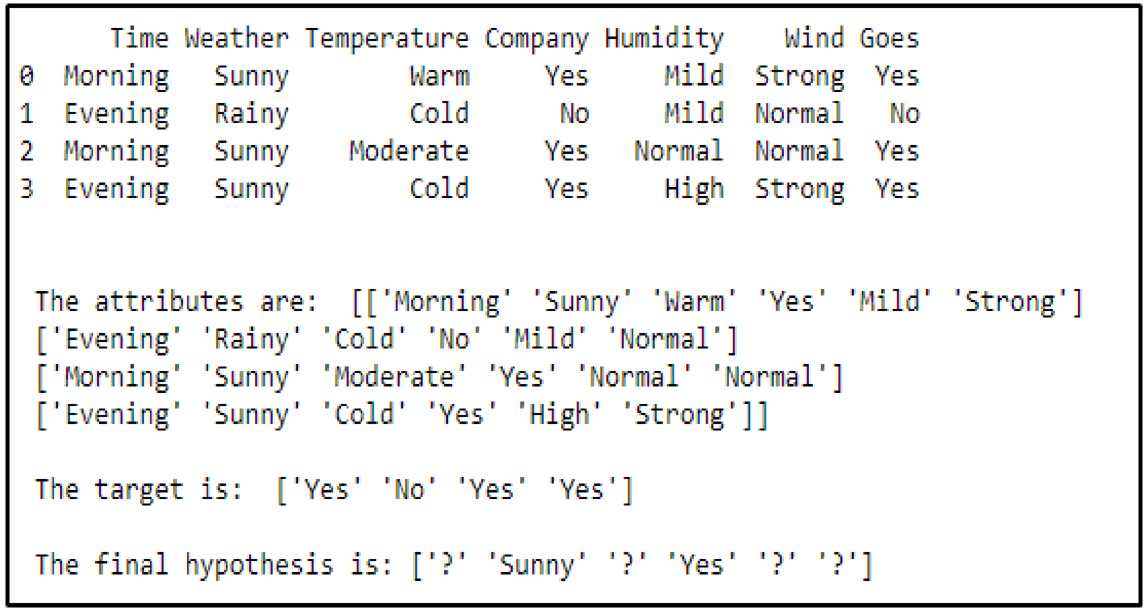
Final hypothesis h = {‘?’, ‘Sunny’, ‘?’, ‘Yes’, ‘?’, ‘?’}

Conclusion: If weather is sunny and company is yes then person will go for a walk irrespective of the values of the attributes time, temparature, humidity and wind.

Implementation of Find-S algorithm

```
import pandas as pd
import numpy as np

# to read the data in the csv file
data = pd.read_csv("data.csv")
print(data,"n")

# making an array of all the attributes
d = np.array(data)[: , -1]
print("n The attributes are: ",d)

 # segragating the target that has positive and negative
examples
target = np.array(data)[ : , -1]
print("n The target is: ", target)

# training function to implement find-s algorithm
def train(c,t):
    for i, val in enumerate(t):
        if val == "Yes":
            specific_hypothesis = c[i].copy()
            break
```

```
            for i, val in enumerate(c):
        if t[i] == "Yes":
            for x in range(len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass
    return specific_hypothesis

# obtaining the final hypothesis
print("n The final hypothesis is:", train(d,target))
```



```
      Time Weather Temperature Company Humidity    Wind Goes
0  Morning   Sunny        Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning   Sunny    Moderate     Yes   Normal  Normal  Yes
3  Evening   Sunny        Cold     Yes     High  Strong  Yes


 The attributes are:  [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

 The target is:  ['Yes' 'No' 'Yes' 'Yes']

 The final hypothesis is: ['?' 'Sunny' '?' 'Yes' '?' '?']
```

Thank you

Unit - I(part 2)

Introduction to Machine Learning

MACHINE LEARNING - UNIT 1

Introduction:

- Introduction to machine learning
- Supervised learning, Unsupervised learning, Semi-supervised learning
- Reinforcement learning
- Deep learning, Concept learning using find-S algorithm.

Feature Engineering:

- Feature Selection using Filter, Wrapper, Embedded methods
- Feature normalization using min-max normalization
- z-score normalization and constant factor normalization

Introduction to Dimensionality Reduction:

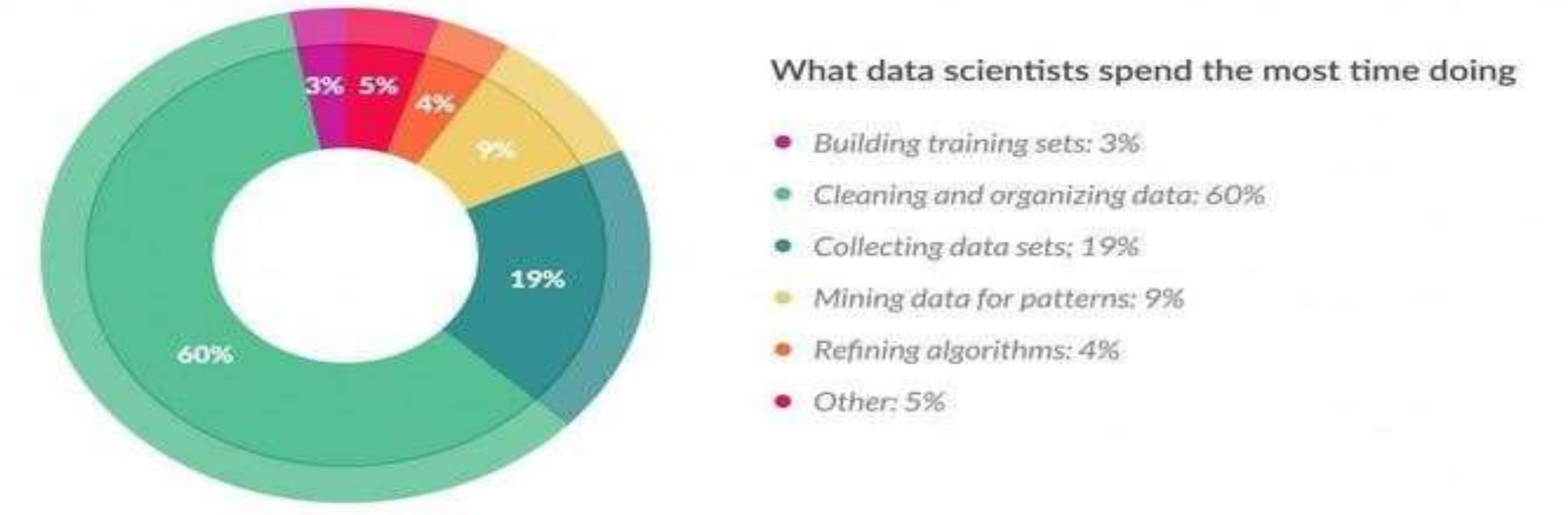
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA) techniques.

Feature Engineering

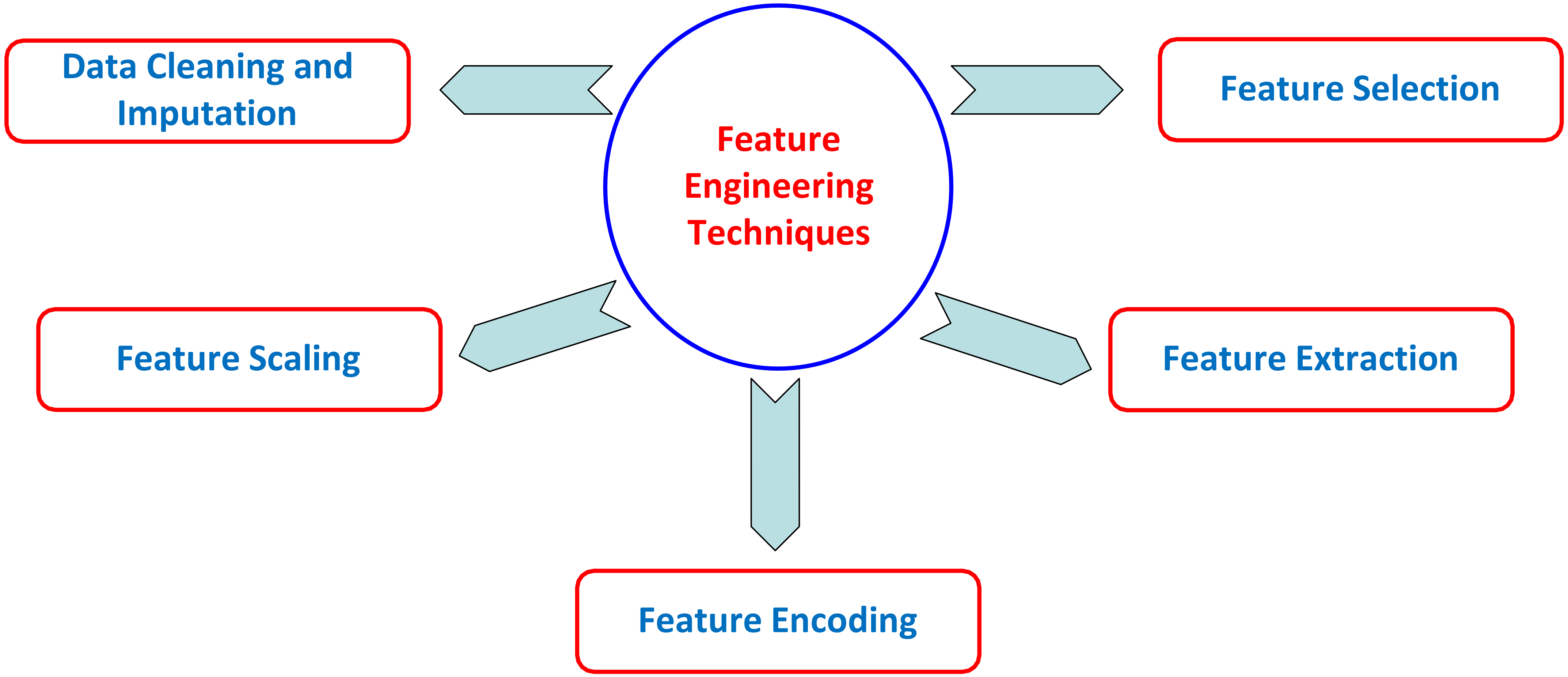
Feature Engineering

Feature Engineering

- Feature engineering is a machine learning method that uses data to create new variables that are not included in the training set.
- It can generate new features for both supervised and unsupervised



Feature Engineering - Techniques



Data Cleaning & Imputation

- The raw data may contain missing values, unstructured data, incorrect inputs, and outliers.
- Addressing missing values and inconsistencies in the data ensures that the information used for training the model is reliable and consistent.
- Missing values can be handled using the following methods:

  - **Delete Columns**
    - Delete entire column that is irrelevant
    - Delete columns by checking a specified threshold

  - **Impute missing values for Continuous Variables**
    - Filling up the missing value with some value of the corresponding feature
    - Can use mean, median or mode of the values in the feature with missing values

  - **Impute missing values for Categorical Variables**
    - Drop rows with missing values
    - Assign new category to missing values
    - Impute with most frequent value for that column

  - **Prediction Imputation**
    - Apply linear regression among column with missing value and rest of the columns

Feature Scaling

- **Standardizing the range of numerical features ensures that all features contribute equally to the model's training process, preventing any single feature from dominating the analysis.**

1. Standardization

Standard scaler / Z-score normalization

2. Normalization

Min - max normalization

Mean normalization

Constant factor normalization

Robust scaling

Feature Encoding

Categorical features, such as colors or names, need to be encoded into numerical values to be compatible with machine learning algorithms.

Common techniques include:

- **One-hot encoding : Makes use of Dummy variables, Used for independent variables**
- **Label encoding : Used for Dependent Variable**

Feature Extraction

- Extracting features from raw data can involve techniques like dimensionality reduction, which reduces the number of features while preserving the most important information.
- Dimensionality Reduction is one of the feature extraction methods.

Feature Extraction Methods

1. Principal Component Analysis(PCA)
2. Linear Discriminant Analysis(LDA)

Feature Selection

Selecting the most relevant and informative features can reduce the complexity of the model and improve its performance by eliminating irrelevant or redundant features.

The feature selection approaches are of three types:

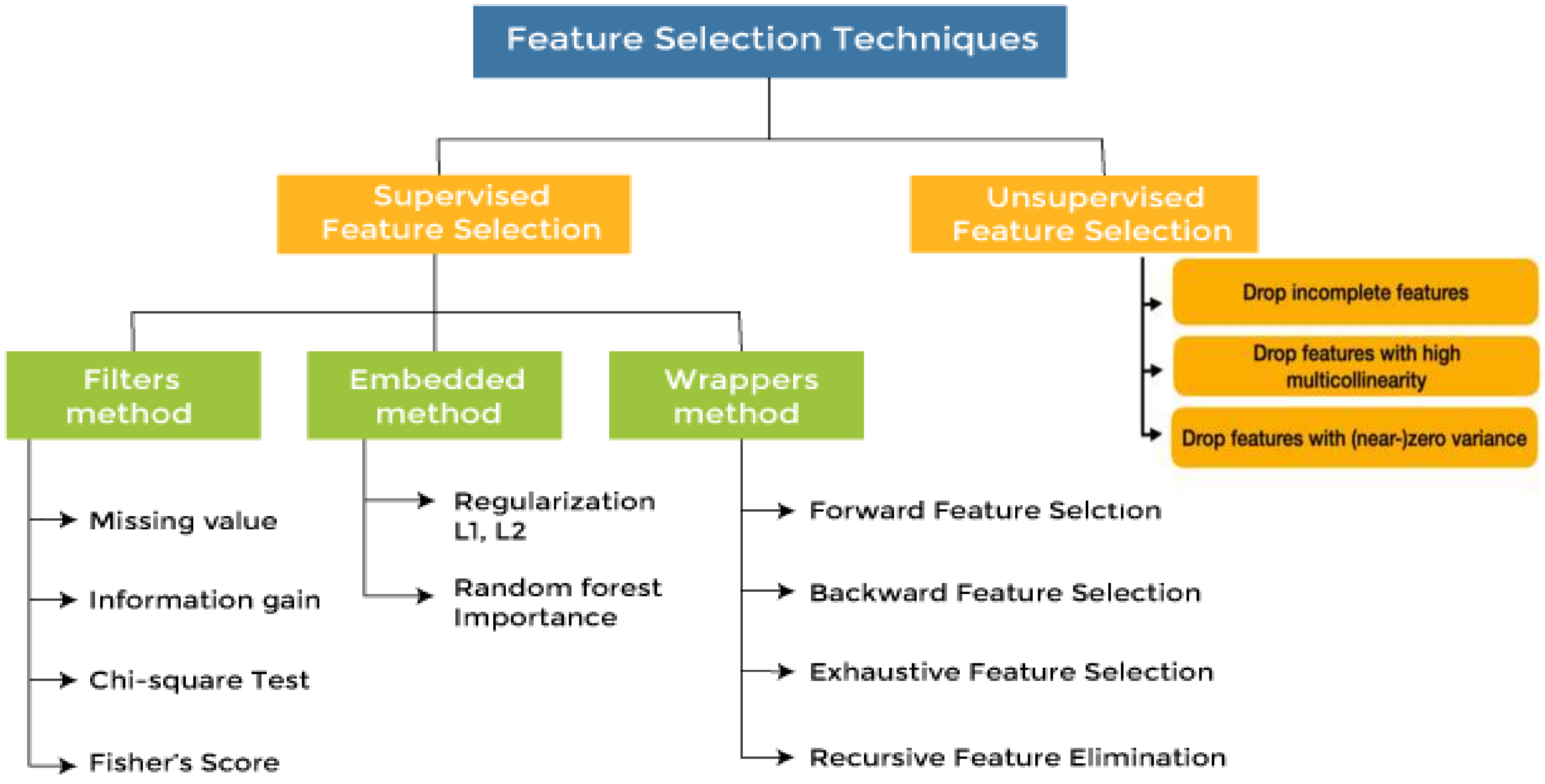
- **Filter based Methods**
- **Wrapper based Methods**
- **Embedded Approach**

Feature Selection

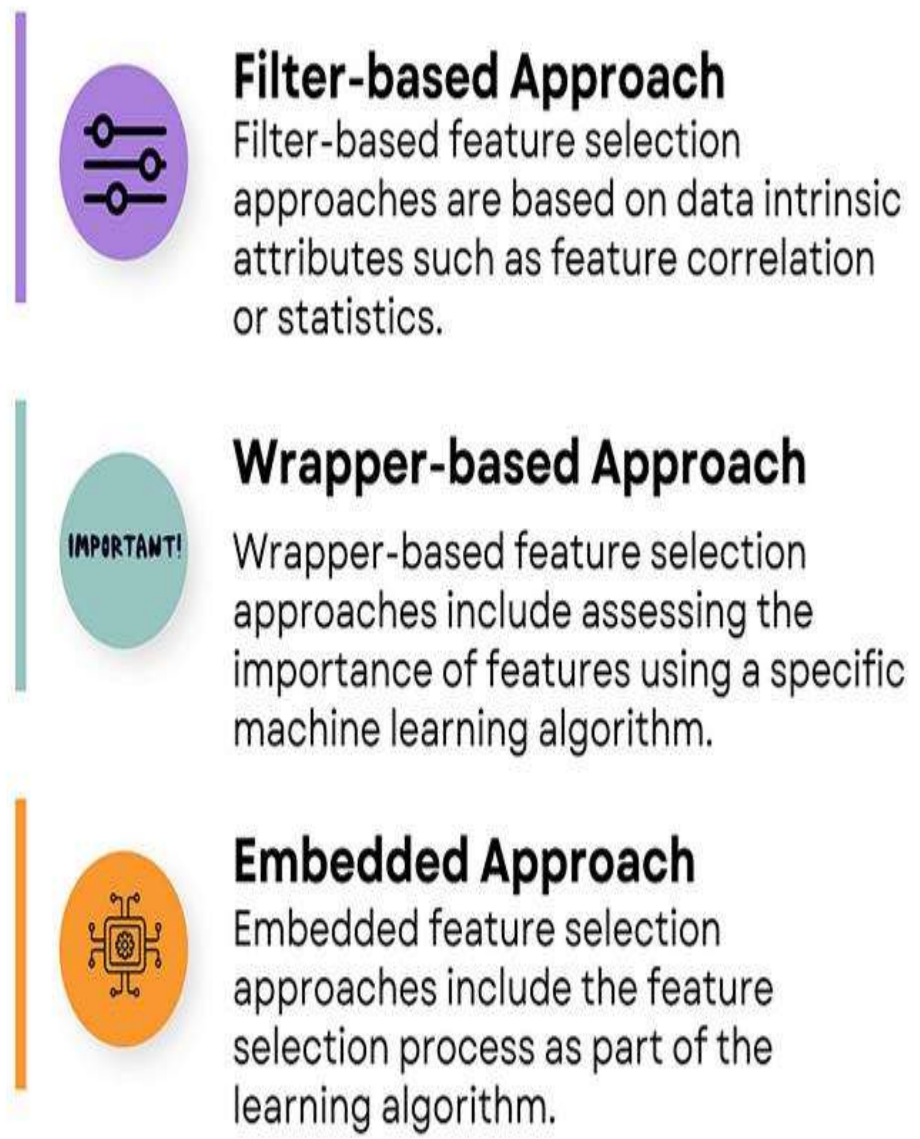
Feature Selection

- It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building.
- Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.
- Some benefits of using feature selection in machine learning:
  - It helps in avoiding the curse of dimensionality.
  - It helps in the simplification of the model so that it can be easily interpreted by the researchers.
  - It reduces the training time.
  - It reduces overfitting hence enhance the generalization.

Feature Selection Techniques


Feature Selection Techniques
Supervised Feature Selection
Unsupervised Feature Selection
Drop incomplete features
Drop features with high multicollinearity
Drop features with (near-)zero variance
Filters method
Embedded method
Wrappers method
Missing value
Information gain
Chi-square Test
Fisher's Score
Regularization L1, L2
Random forest Importance
Forward Feature Selction
Backward Feature Selection
Exhaustive Feature Selection
Recursive Feature Elimination

Feature Selection Methods(Supervised Feature Selection)



Filter-based Approach

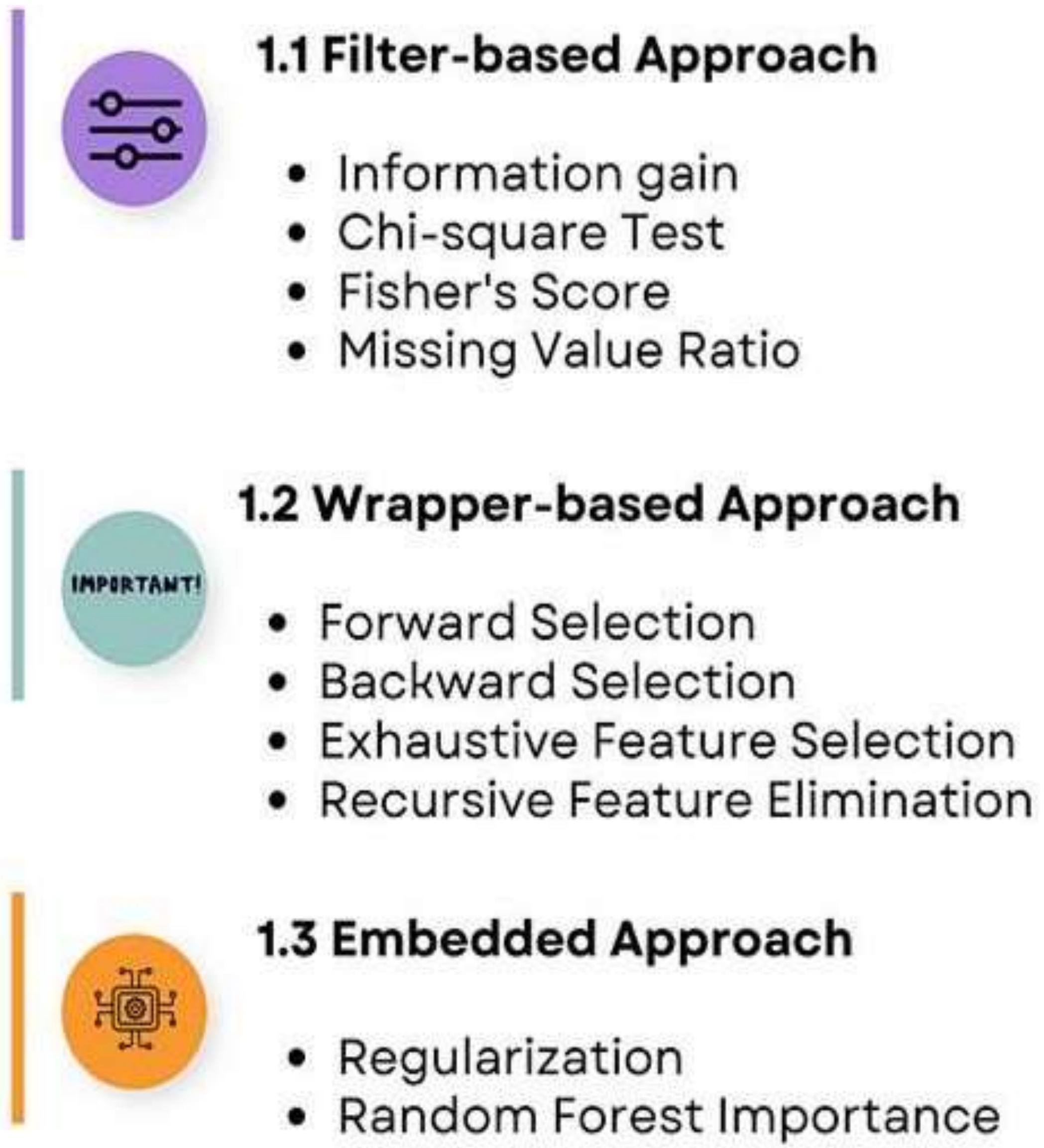
Filter-based feature selection approaches are based on data intrinsic attributes such as feature correlation or statistics.

Wrapper-based Approach

Wrapper-based feature selection approaches include assessing the importance of features using a specific machine learning algorithm.

Embedded Approach

Embedded feature selection approaches include the feature selection process as part of the learning algorithm.



1.1 Filter-based Approach

- Information gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio

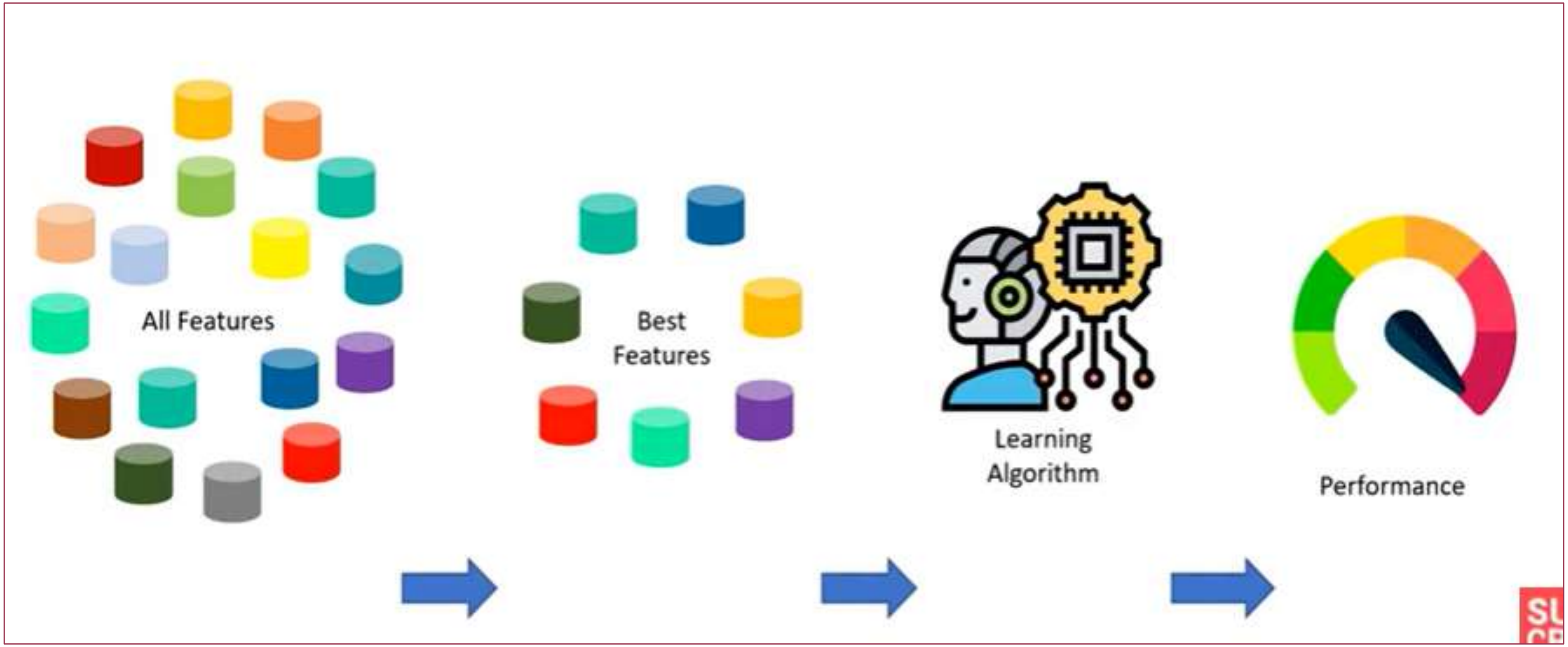
1.2 Wrapper-based Approach

- Forward Selection
- Backward Selection
- Exhaustive Feature Selection
- Recursive Feature Elimination

1.3 Embedded Approach

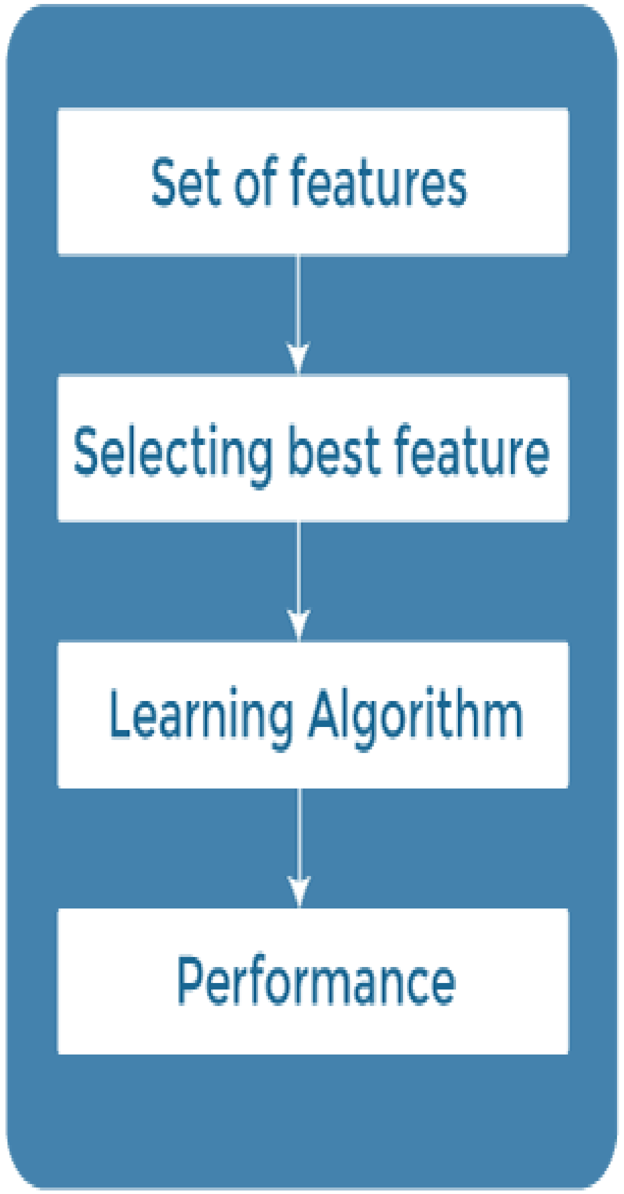
- Regularization
- Random Forest Importance

Filter Methods



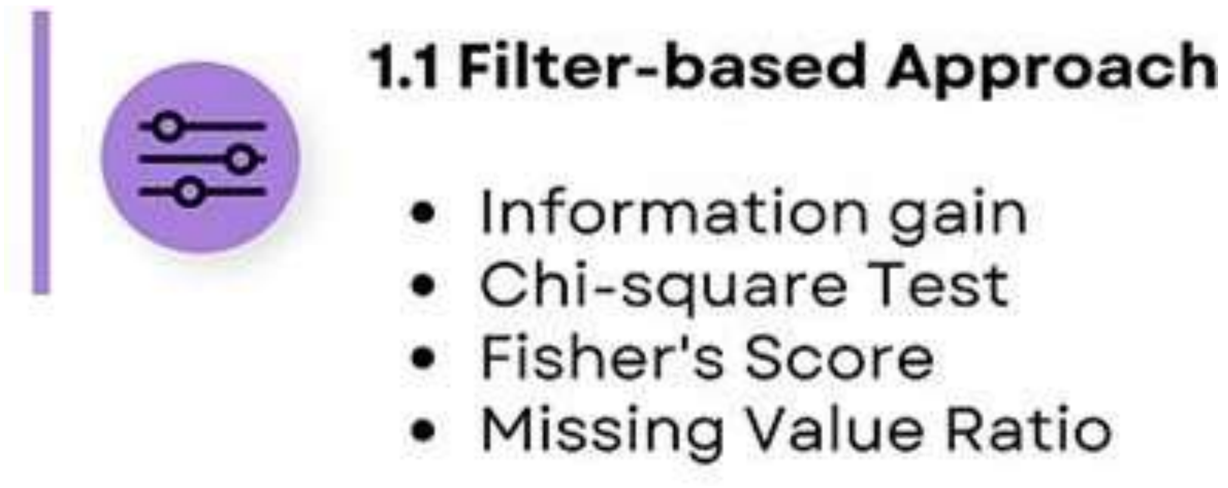
Filter Methods

- **In Filter Methods,** features are selected on the basis of statistical measures. This method select features from the dataset irrespective of the use of any machine learning algorithm. The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.
- Features are dropped based on their relation to the output, or how they are correlating to the output. We use correlation to check if the features are positively or negatively correlated to the output labels and drop features accordingly.
- The advantage of using filter methods is that it needs low computational time and does not overfit the data.



Filter Based Feature Selection Techniques

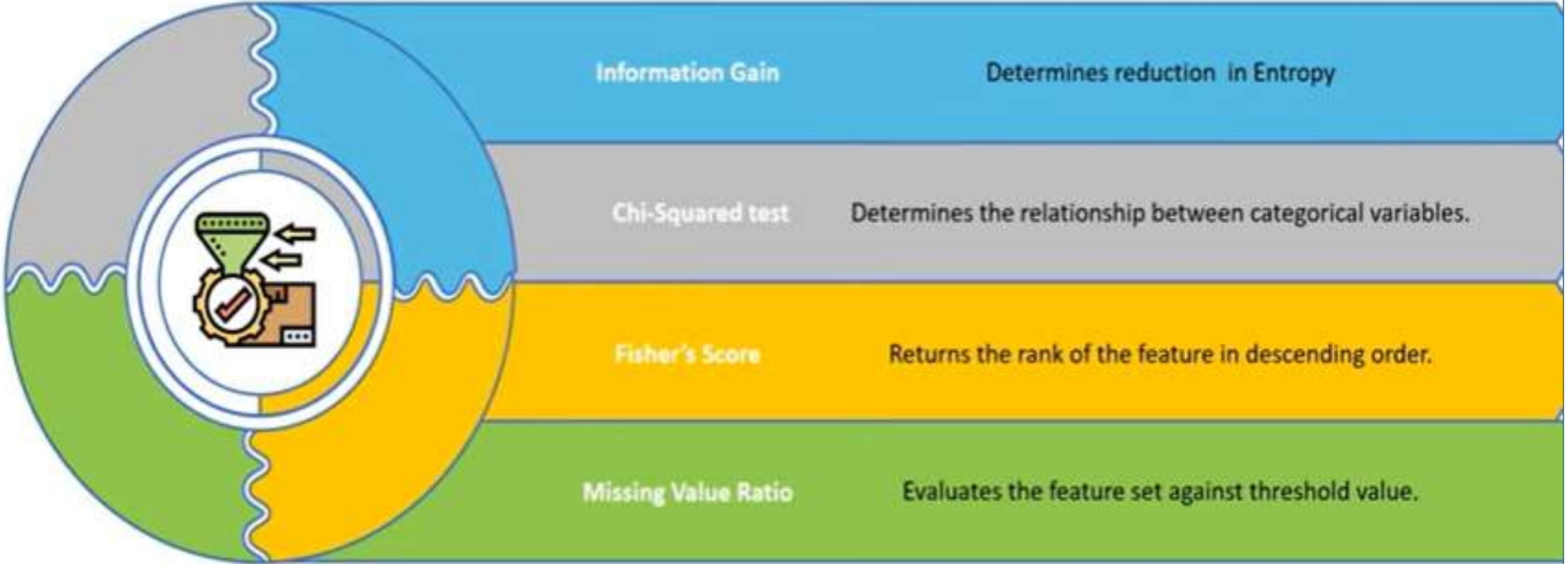
- In Filter Method, features are selected on the basis of statistics as measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.
- The filter method filters out the irrelevant features and redundant columns from the dataset by using different metrics through ranking.
- The advantage of using filter methods is that it needs low computational time and does not overfit the data.



1.1 Filter-based Approach

- Information gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio

Filter Method Techniques



Information Gain	Determines reduction in Entropy
Chi-Squared test	Determines the relationship between categorical variables.
Fisher's Score	Returns the rank of the feature in descending order.
Missing Value Ratio	Evaluates the feature set against threshold value.

Filter Based Techniques – Information Gain

- Information gain determines the reduction in entropy while transforming the dataset.

- It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.
- Information Entropy for a dataset with C classes:

$$E = -\sum_{i}^{C} p_i \log_2 p_i$$

- **Information Gain = Entropy before split - Entropy after split**
- Higher Information Gain = More Entropy removed

Filter Based Techniques – Chi Square Test

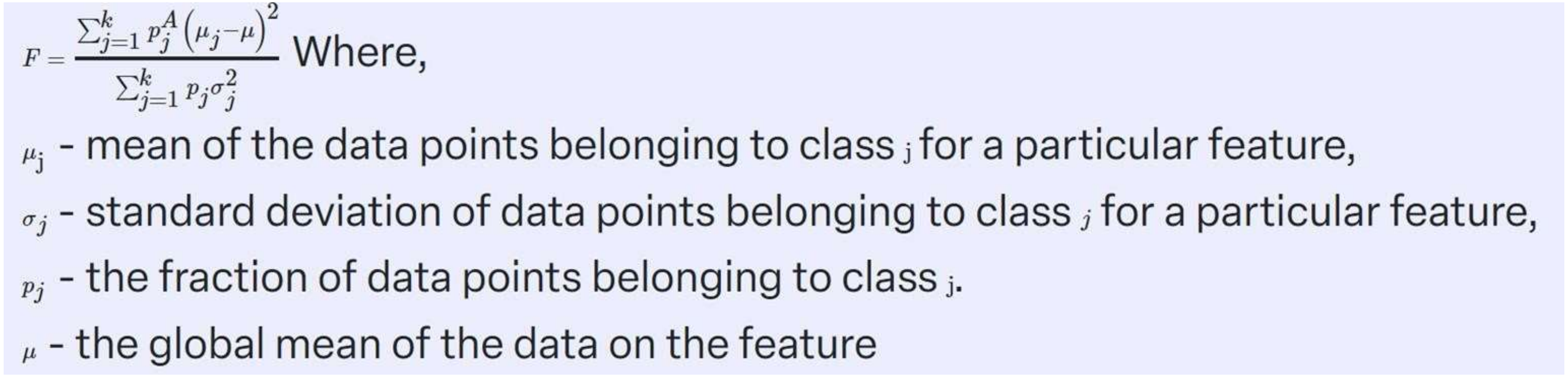
- Chi-square test is a technique to determine the relationship between the categorical variables.
- The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

$$X^2 = \sum \frac{(Observed\ value - Expected\ value)^2}{Expected\ value}$$

- Higher the Chi-Square value, the feature is more dependent on the response and it can be selected for model training.

Filter Based Techniques – Fisher’s Score

- Fisher's score is one of the popular supervised technique of feature selection.
- Fisher's Score is calculated as the ratio of between-class and within-class variance.
- It returns the rank of the variable on the fisher's criteria in descending order.



$$F = \frac{\sum_{j=1}^{k} p_j^A \left(\mu_j - \mu\right)^2}{\sum_{j=1}^{k} p_j \sigma_j^2}$$ Where,

μ_j - mean of the data points belonging to class $_j$ for a particular feature,

σ_j - standard deviation of data points belonging to class $_j$ for a particular feature,

p_j - the fraction of data points belonging to class $_j$.

μ - the global mean of the data on the feature

- Feature with Highest fisher score is identified to be the relevant variable.

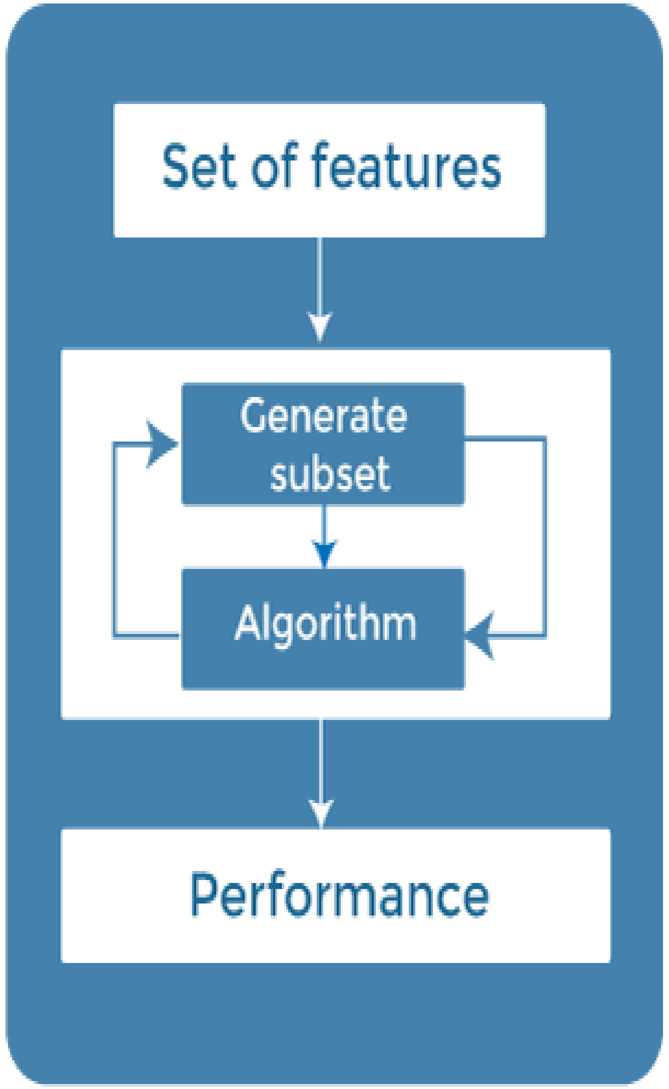
Filter Based Techniques – Missing Value Ratio

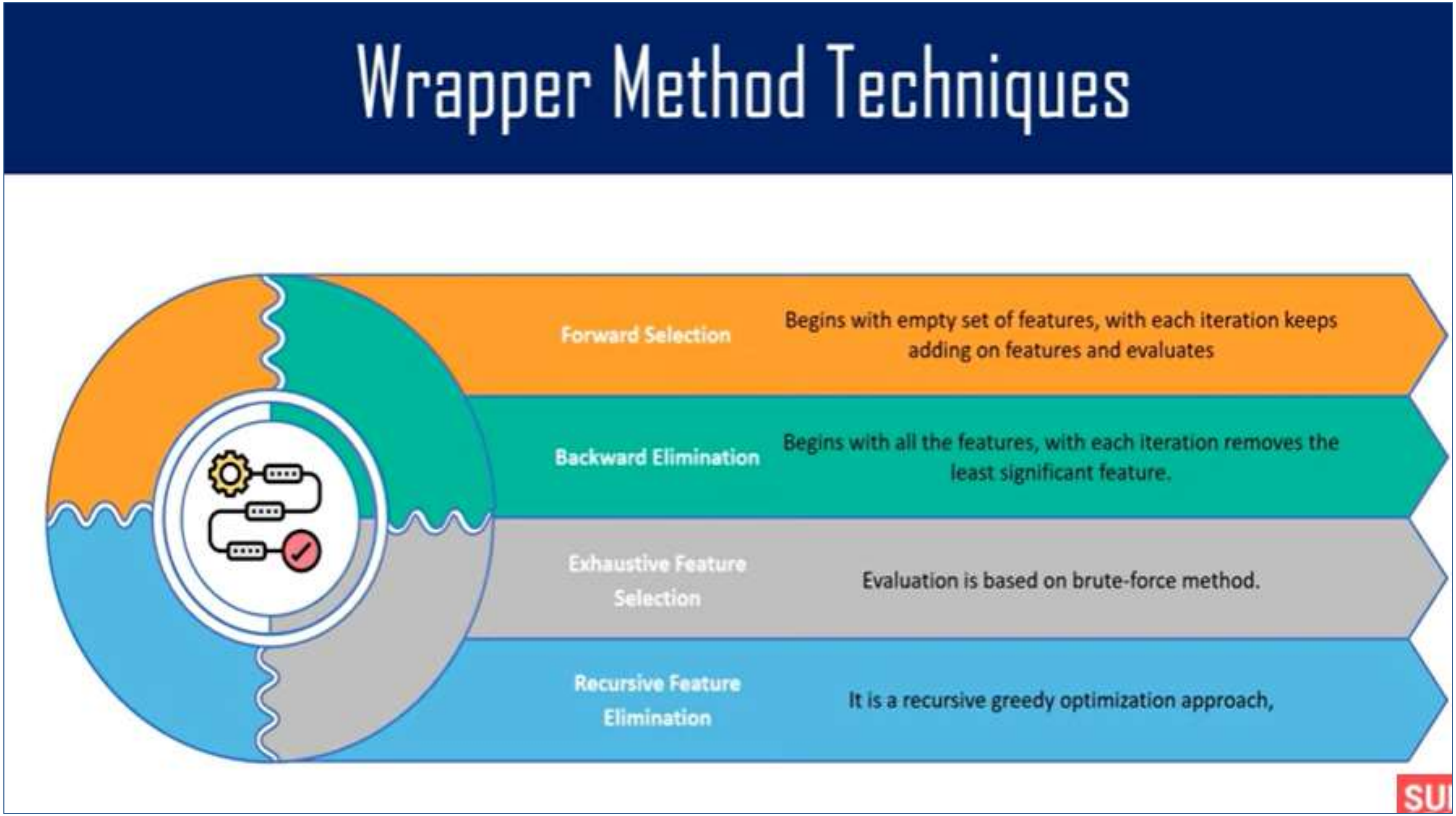
- **Missing Value Ratio:**

The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.

Wrapper Methods

- Wrapper methods, also referred to as greedy algorithms, train the algorithm by using a subset of features in an iterative manner. Based on the conclusions made from training in prior to the model, addition and removal of features takes place.
- Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features has been achieved.
- The main advantage of wrapper methods over the filter methods is that they provide an optimal set of features for training the model, thus resulting in better accuracy than the filter methods but are computationally more expensive.



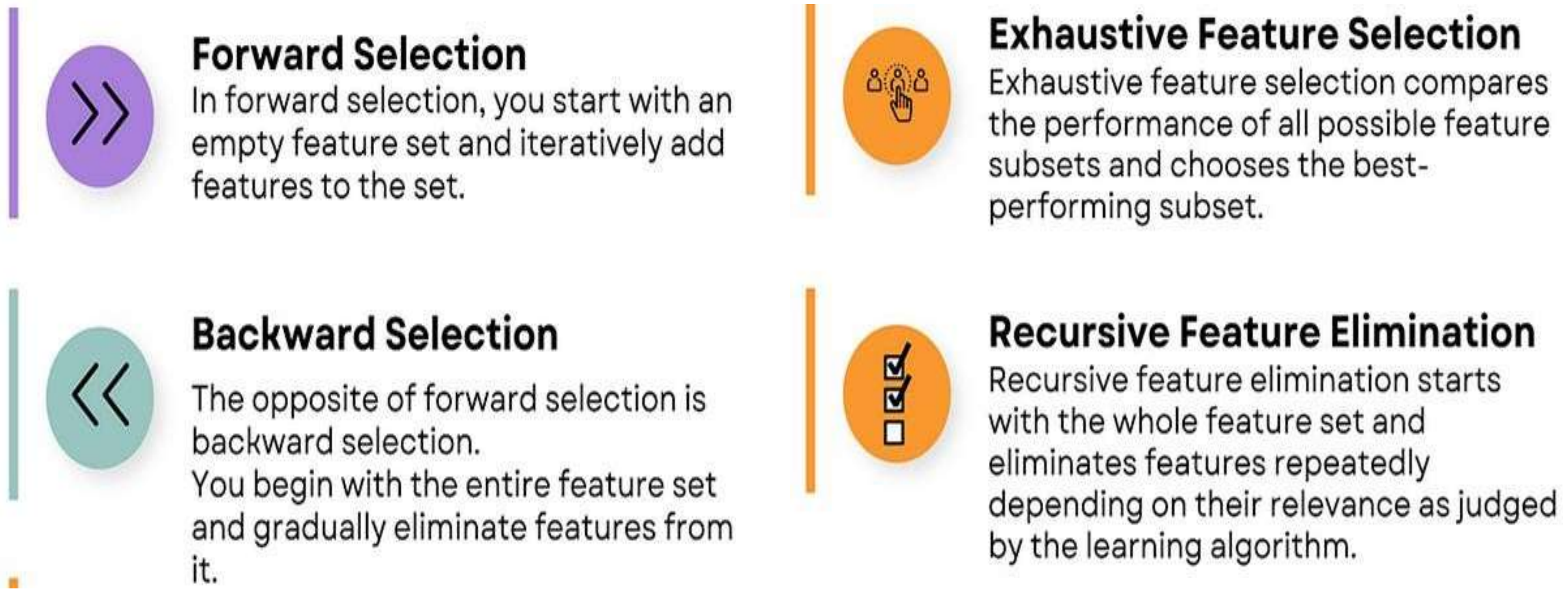


Wrapper Method Techniques

Forward Selection	Begins with empty set of features, with each iteration keeps adding on features and evaluates
Backward Elimination	Begins with all the features, with each iteration removes the least significant feature.
Exhaustive Feature Selection	Evaluation is based on brute-force method.
Recursive Feature Elimination	It is a recursive greedy optimization approach,

Wrapper Based Feature Selection Techniques

- In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



Forward Selection

In forward selection, you start with an empty feature set and iteratively add features to the set.

Exhaustive Feature Selection

Exhaustive feature selection compares the performance of all possible feature subsets and chooses the best-performing subset.

Backward Selection

The opposite of forward selection is backward selection.
You begin with the entire feature set and gradually eliminate features from it.

Recursive Feature Elimination

Recursive feature elimination starts with the whole feature set and eliminates features repeatedly depending on their relevance as judged by the learning algorithm.

Feature Selection – Wrapper Methods

- **Forward selection** – This method is an iterative approach where we initially start with an empty set of features and keep adding a feature which best improves our model after each iteration. The stopping criterion is till the addition of a new variable does not improve the performance of the model.
- **Backward elimination** – This method is also an iterative approach where we initially start with all features and after each iteration, we remove the least significant feature. The stopping criterion is till no improvement in the performance of the model is observed after the feature is removed.
- **Bi-directional elimination** – This method uses both forward selection and backward elimination technique simultaneously to reach one unique solution.

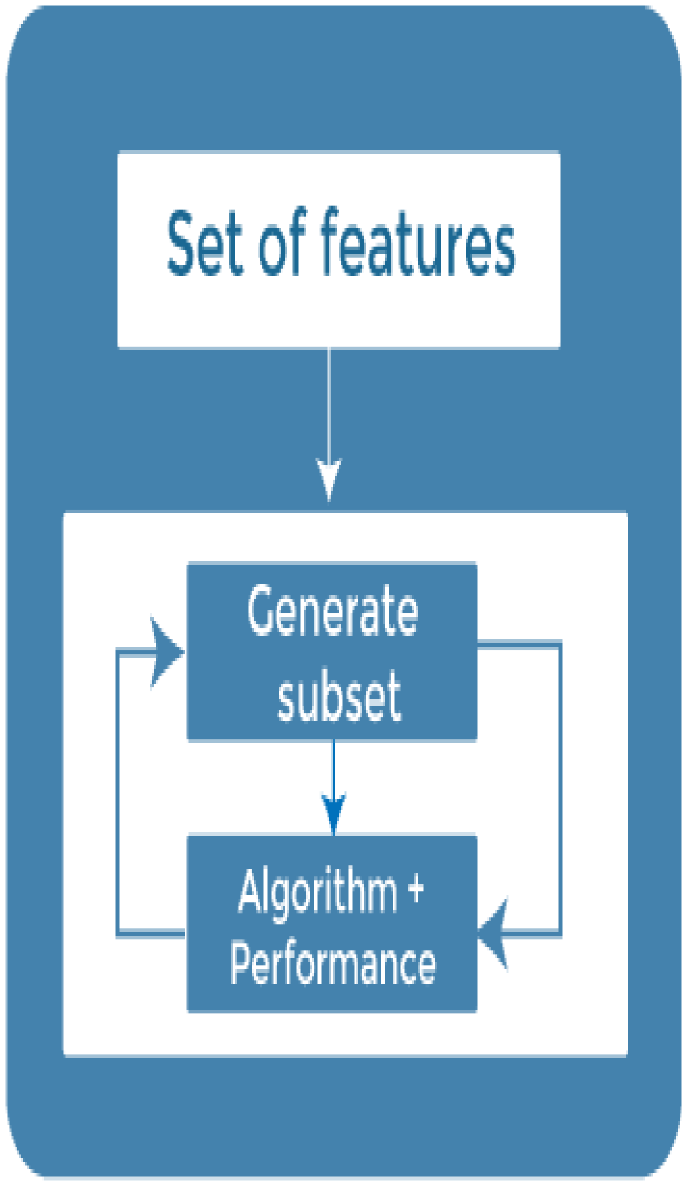
Feature Selection – Wrapper Methods

- **Exhaustive selection –** This technique is considered as the brute force approach for the evaluation of feature subsets. It creates all possible subsets and builds a learning algorithm for each subset and selects the subset whose model's performance is best.

- **Recursive elimination –** This greedy optimization method selects features by recursively considering the smaller and smaller set of features. The estimator is trained on an initial set of features and their importance is obtained using feature_importance_attribute. The least important features are then removed from the current set of features till we are left with the required number of features.

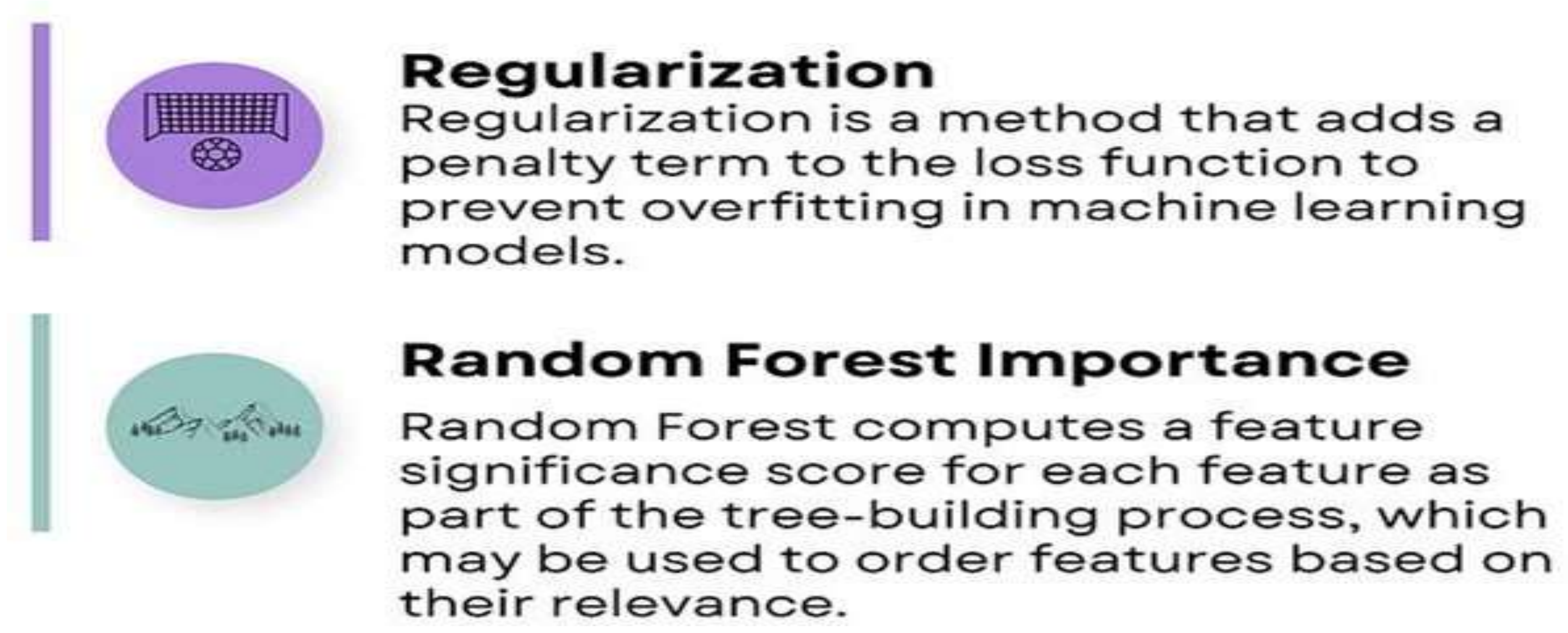
Embedded Methods

- In embedded methods, the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods.
- Embedded methods encounter the drawbacks of filter and wrapper methods and merge their advantages.
- Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost.
- These methods are faster like those of filter methods and more accurate than the filter methods.
- These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration.



Embedded Feature Selection Techniques

- Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost.
- These are fast processing methods similar to the filter methods but more accurate than the filters method.



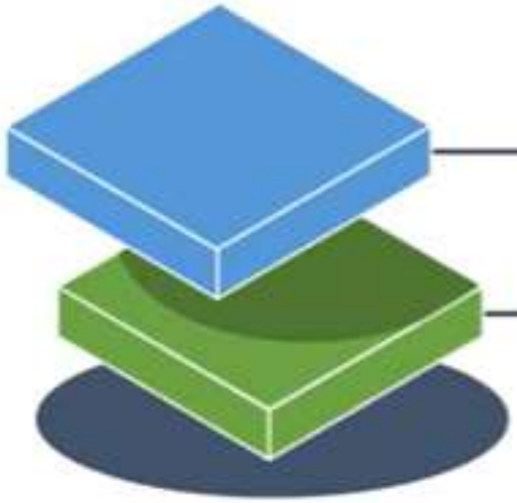
Regularization

Regularization is a method that adds a penalty term to the loss function to prevent overfitting in machine learning models.

Random Forest Importance

Random Forest computes a feature significance score for each feature as part of the tree-building process, which may be used to order features based on their relevance.

Embedded Method Techniques



Regularization Techniques like L1, L2 or Elastic Nets

Random Forest, Tress based methods for feature selection

Feature Selection – Embedded Methods

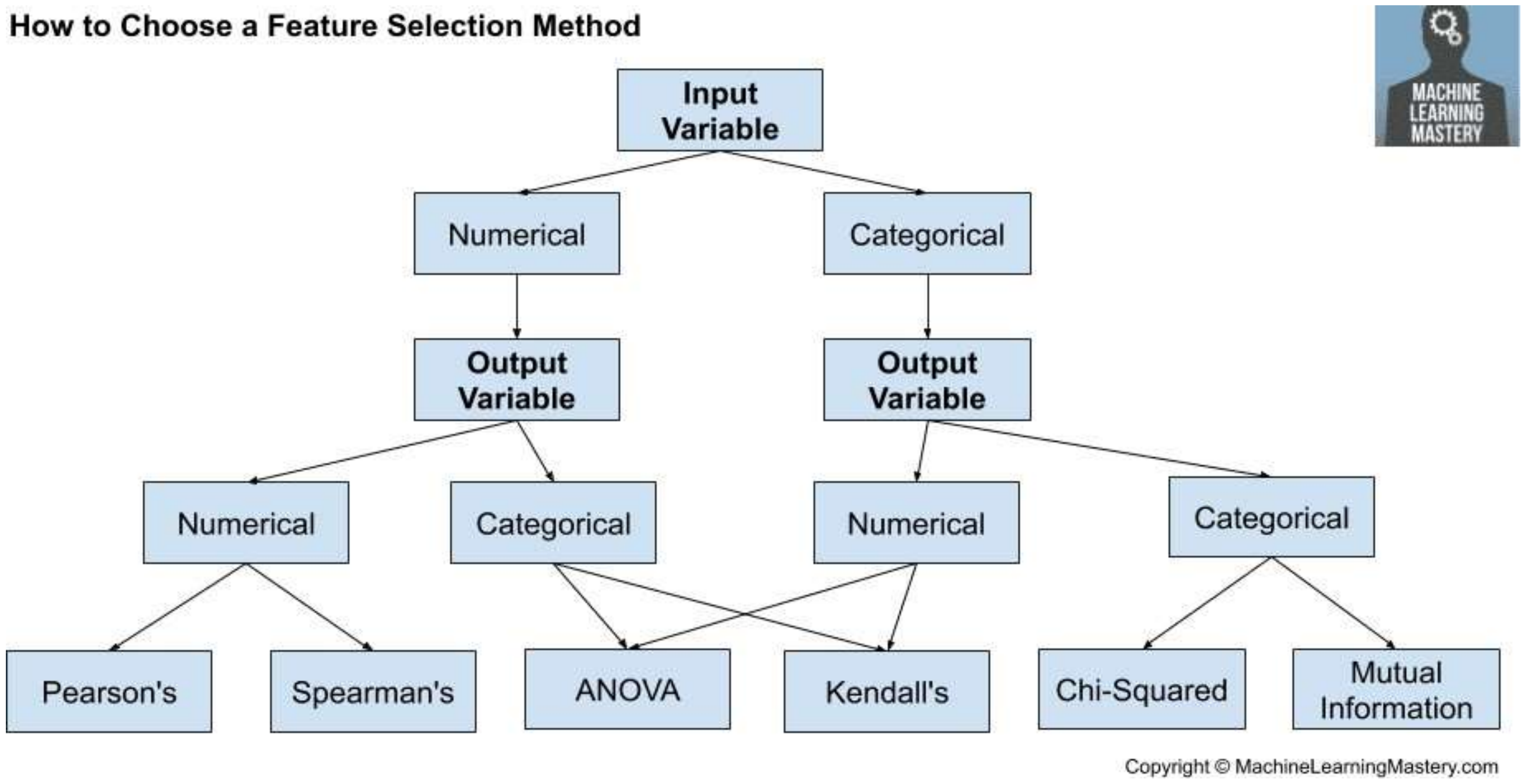
- **Regularization –** This method adds a penalty to different parameters of the machine learning model to avoid over-fitting of the model. This approach of feature selection uses Lasso (L1 regularization) and Elastic nets (L1 and L2 regularization). The penalty is applied over the coefficients, thus bringing down some coefficients to zero. The features having zero coefficient can be removed from the dataset.

- **Tree-based methods –** Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the impurity (Gini impurity) over all the trees. The remaining nodes create a subset of the most important features.

Unsupervised Feature Selection Methods

- Unsupervised feature selection methods do not make use of any labels. In other words, they don't need access to the target variable of the machine learning model.
- How can we claim a feature to be unimportant for the model without analyzing its relation to the model's target? in some cases, this is possible. We might want to discard the features with:
- Zero or near-zero variance. Features that are (almost) constant provide little information to learn from and thus are irrelevant.
- Many missing values. While dropping incomplete features is not the preferred way to handle missing data, it is often a good start, and if too many entries are missing, it might be the only sensible thing to do since such features are likely inconsequential.
- High multicollinearity; multicollinearity means a strong correlation between different features, which might signal redundancy issues.

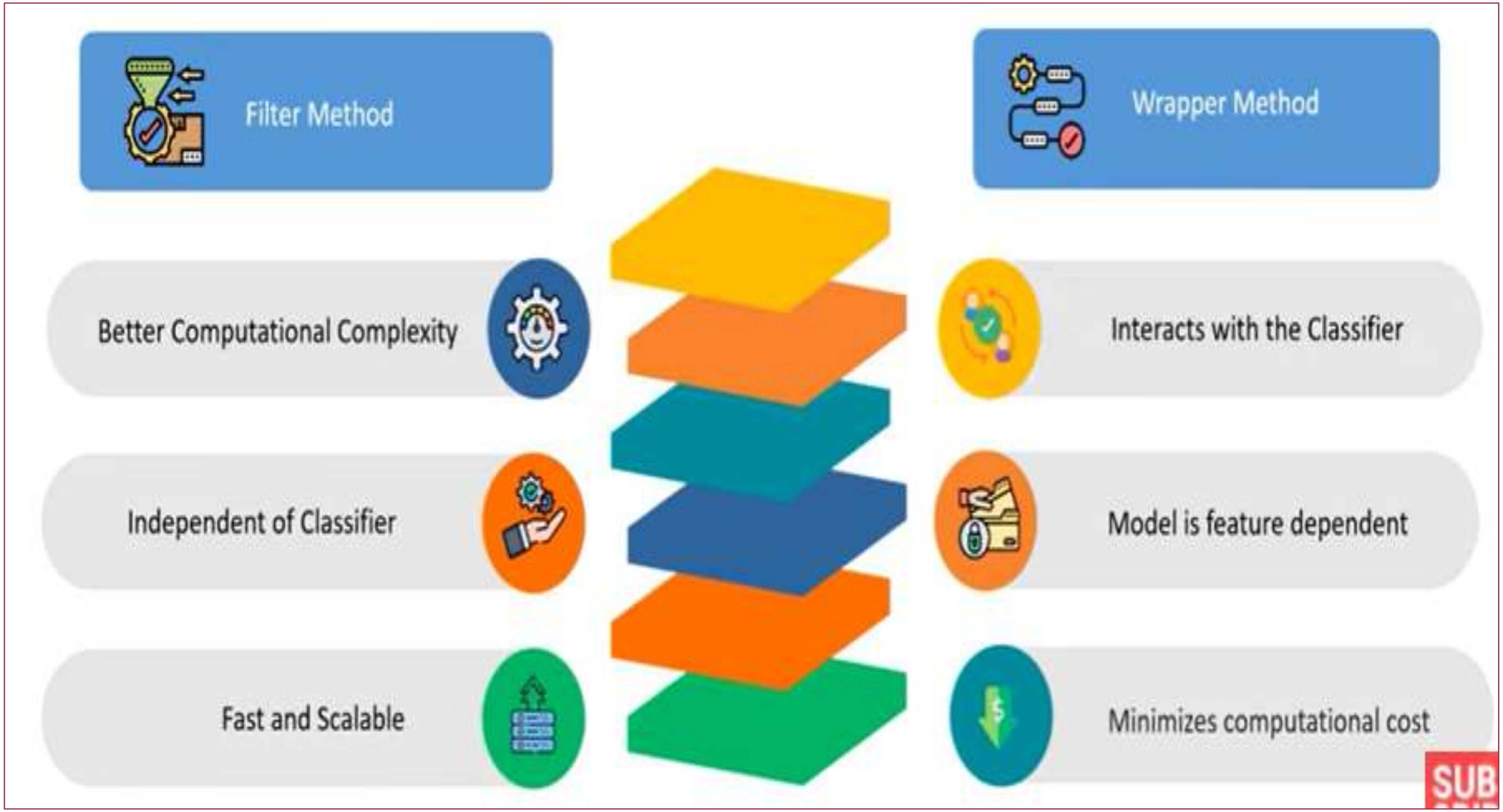
How to choose Feature Selection Method



How to choose Feature Selection Method

Input Variable	Output Variable	Feature Selection technique
Numerical	Numerical	•Pearson's correlation coefficient (For linear Correlation) •Spearman's rank coefficient (for non-linear correlation)
Numerical	Categorical	•ANOVA correlation coefficient (linear) •Kendall's rank coefficient (nonlinear)
Categorical	Numerical	•Kendall's rank coefficient (linear) •ANOVA correlation coefficient (nonlinear)
Categorical	Categorical	•Chi-Squared test (contingency tables) •Mutual Information

Advantages of Filter & Wrapper Methods



Filter Method

Better Computational Complexity

Independent of Classifier

Fast and Scalable

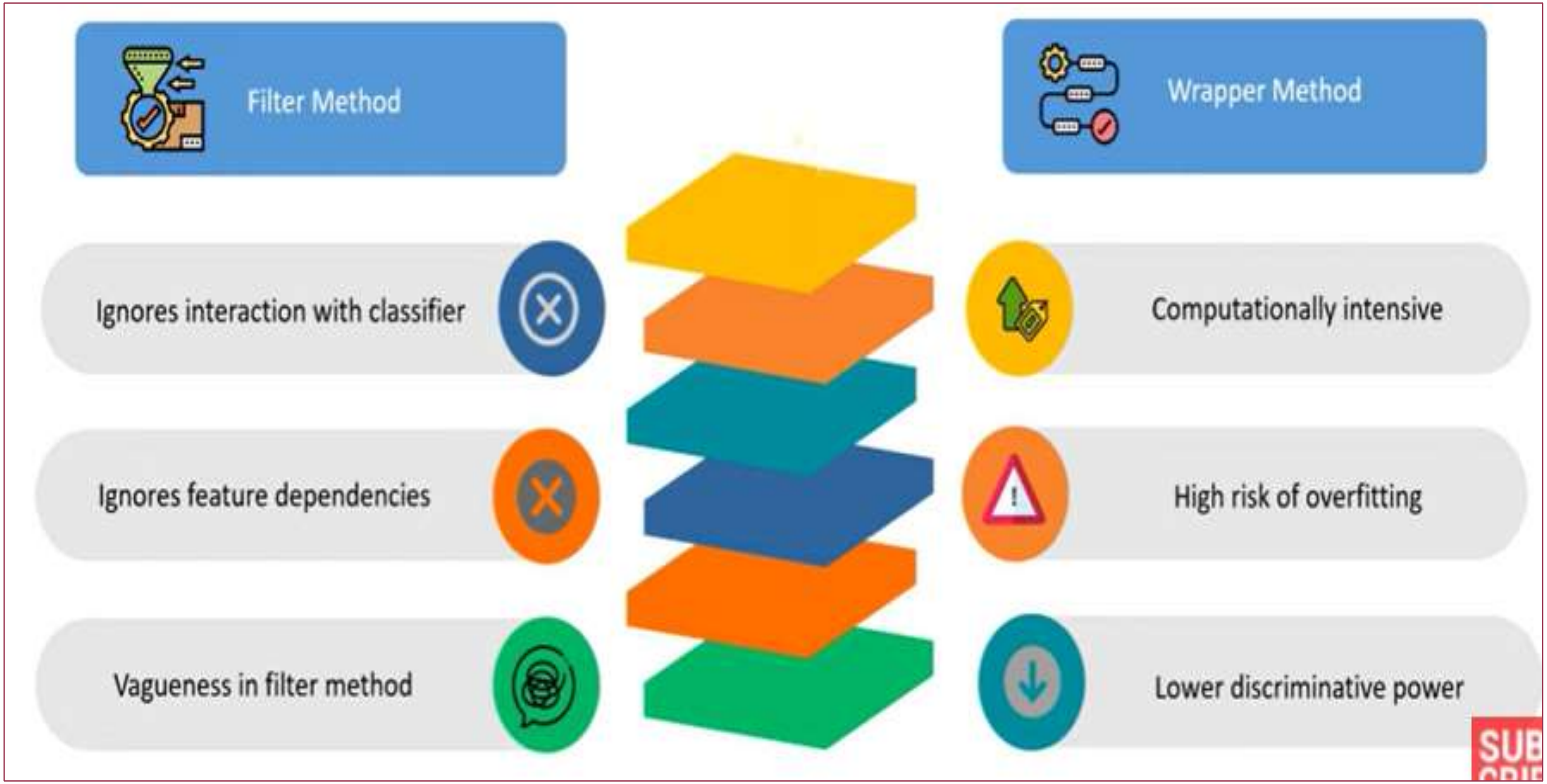
Wrapper Method

Interacts with the Classifier

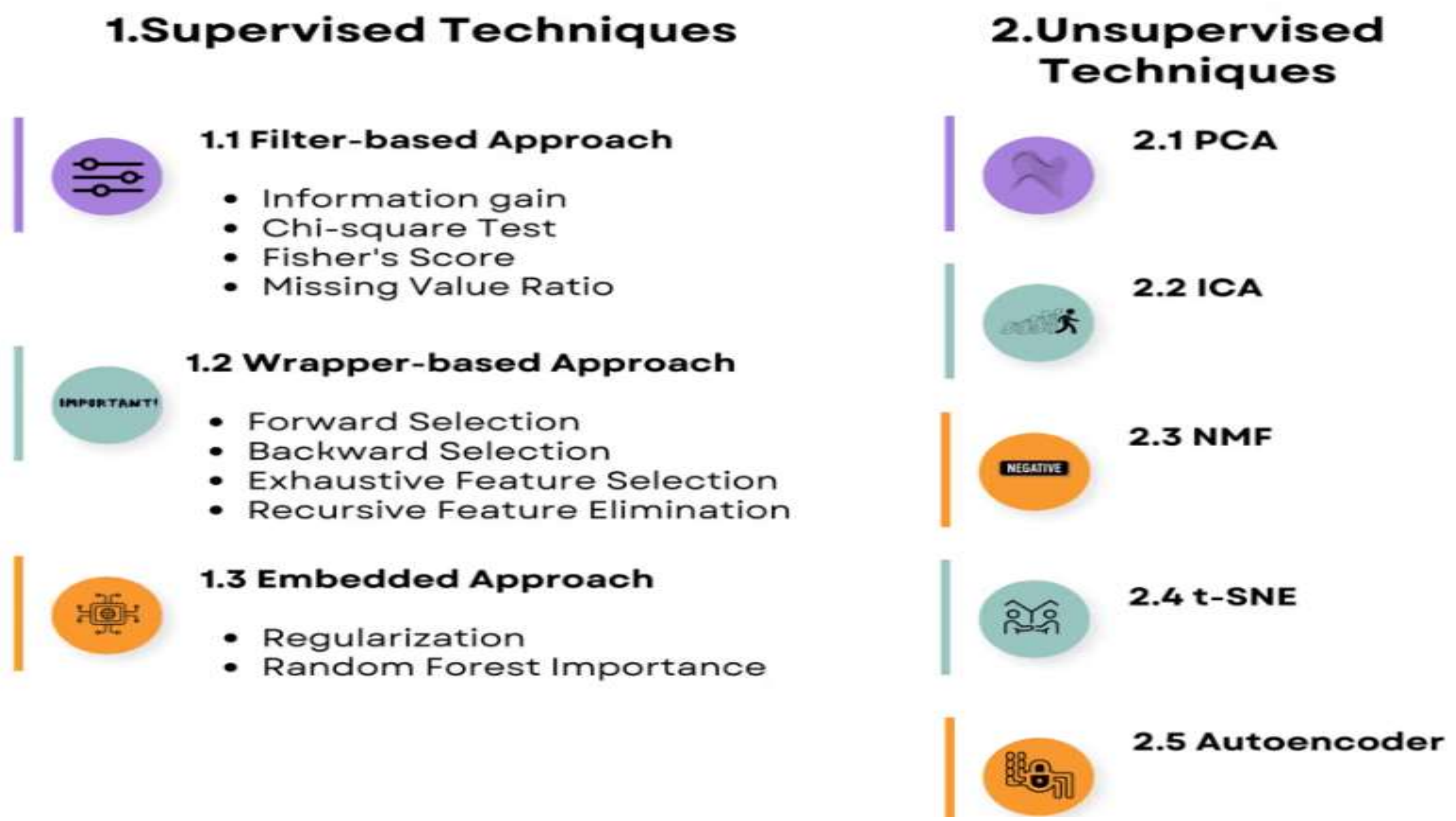
Model is feature dependent

Minimizes computational cost

Limitations of Filter & Wrapper Methods


Filter Method
Wrapper Method
Ignores interaction with classifier
Computationally intensive
Ignores feature dependencies
High risk of overfitting
Vagueness in filter method
Lower discriminative power

Advanced Feature Selection Techniques


IMPORTANT!
NEGATIVE

1.Supervised Techniques

1.1 Filter-based Approach

- Information gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio

1.2 Wrapper-based Approach

- Forward Selection
- Backward Selection
- Exhaustive Feature Selection
- Recursive Feature Elimination

1.3 Embedded Approach

- Regularization
- Random Forest Importance

2.Unsupervised Techniques

2.1 PCA

2.2 ICA

2.3 NMF

2.4 t-SNE

2.5 Autoencoder

Feature Scaling

Feature Scaling / Normalization

- One key aspect of Feature Engineering is scaling, normalization, and standardization, which involves transforming the data to make it more suitable for modeling.
- The process of transforming the values of numeric columns in a dataset to the same/standard scale is referred to as Normalization.
- Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale.
- Every dataset does not need to be normalized for machine learning. It is required only when features of machine learning models have different ranges.

Feature Scaling

- Normalization is also known as feature scaling, it is applied on numerical features.
- Feature scaling is a data preprocessing technique used to transform the values of features or variables in a dataset to a similar scale.
- Normalization and standardization are not the same processes.
- Standardization, interestingly, refers to setting the mean to zero and the standard deviation to one.
- Normalization in machine learning is the process of translating data into the range [0, 1] (or any other range).

Types of Feature Scaling

- Types of Feature Scaling:
  - Standardization:
    - Standard scaler / Z-score normalization
  - Normalization:
    - Min - max normalizaion
    - Mean normalization
    - Constant factor normalization
    - Robust scaling

Standardization

- This is also called as Z-score Normalization.
- Standardization is a scaling technique where the values are centered around the mean with a unit standard deviation.
- This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
- Standardization or z-score normalization does not get affected by outliers because there is no predefined range of transformed features.

$$Z = \frac{\overset{\text{Score}}{x} - \overset{\text{Mean}}{\mu}}{\underset{\text{SD}}{\sigma}}$$

Min-max normalization

- **Min-max normalization: This method performs a linear regression on the original data. Suppose min_A and max_A are minimum and maximum values of attribute A, min-max normalization maps a value v to v^1 in the range [new_min_A, new_max_A] by computing,**

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Ex:- Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0].

Let us take v = \$73,600

Then \$73,600 (v) is mapped to 0.716 (v^1)

Constant Factor Normalization

- The simplest normalization technique is constant factor normalization.
- Expressed as a math equation constant factor normalization is

$$x' = x / k$$

where x is a raw value, x' is the normalized value, and k is a numeric constant.

Feature Normalization

1. If you do not know which scaler to use, apply all and check the effect on the models.
2. If you do not understand the data, use standard scaler. It works most of the times.
3. If you know the max and min values of the feature, then use min max scaler like in CNN.
4. If most of the values in the feature column is 0 or sparse matrix, then use Max Absolute Scaling.
5. If the data has outliers, use Robust Scaling.

Normalization Vs Standardization

Normalization	Standardization
This technique uses minimum and max values for scaling of model.	This technique uses mean and standard deviation for scaling of model.
It is helpful when features are of different scales.	It is helpful when the mean of a variable is set to 0 and the standard deviation is set to 1.
Scales values ranges between [0, 1] or [-1, 1].	Scale values are not restricted to a specific range.
It got affected by outliers.	It is comparatively less affected by outliers.
Scikit-Learn provides a transformer called Min-max Scaler for Normalization.	Scikit-Learn provides a transformer called StandardScaler for Normalization.
It is also called Scaling normalization.	It is known as Z-score normalization.
It is useful when feature distribution is unknown.	It is useful when feature distribution is normal.

Introduction to Dimensionality Reduction

Dimensionality/Feature reduction

- Many modern data domains involve huge numbers of features / dimensions
  - Documents: thousands of words, millions of bigrams
  - Images: thousands to millions of pixels
  - Genomics: thousands of genes, millions of DNA polymorphisms

Why reduce dimensions/features?

- High dimensionality has many costs
  - Redundant and irrelevant features degrade performance of some ML algorithms
  - Difficulty in interpretation and visualization
  - Computation may become infeasible
    - what if your algorithm scales as $O(n^3)$?

Feature Reduction

- How to find the ‘best’ low dimension space that conveys maximum useful information?
- One answer: Find “Principal Components”

Dimensionality Reduction

- **Dimensionality reduction** is the task of reducing the number of features in a dataset.
- The higher the number of features, the more difficult it is to model them, this is known as the **curse of dimensionality.**
- The process of dimensionality reduction essentially transforms data from high- dimensional feature space to a low-dimensional feature space.
- When the data is sparse, observations or samples in the training dataset are difficult to cluster as high-dimensional data causes every observation in the dataset to appear equidistant from each other.
- If data is meaningful and non-redundant, then there will be regions where similar data points come together and cluster, furthermore they must be statistically significant.

Dimensionality Reduction - Tools & Libraries

- The most popular library for dimensionality reduction is scikit-learn (sklearn). The library consists of three main modules for dimensionality reduction algorithms:
- Decomposition algorithms
    - Principal Component Analysis
    - Kernel Principal Component Analysis
    - Non-Negative Matrix Factorization
    - Singular Value Decomposition
- Manifold learning algorithms
    - t-Distributed Stochastic Neighbor Embedding
    - Spectral Embedding
    - Locally Linear Embedding
- Discriminant Analysis
    - Linear Discriminant Analysis

Principal Component Analysis (PCA)

- Principal Component Analysis, or PCA, is a dimensionality-reduction method to find lower-dimensional space by preserving the variance as measured in the high dimensional input space. It is an unsupervised method for dimensionality reduction.
- The main idea behind PCA is to figure out patterns and correlations among various features in the data set. On finding a strong correlation between different variables, a final decision is made about reducing the dimensions of the data in such a way that the significant data is still retained.
- PCA transformations are linear transformations.
- It involves the process of finding the principal components, which is the decomposition of the feature matrix into eigenvectors. This means that PCA will not be effective when the distribution of the dataset is non- linear.

PCA - Implementation

- **Standardization:** The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution Zero centered.
- **Finding covariance:** Covariance helps to understand the relationship between the mean and original data.
  - If positive then: the two variables increase or decrease together (correlated)
  - If negative then: one increases when the other decreases (Inversely correlated)
- **Determining the principal components:** Principal components can be determined by calculating the eigen vectors and eigen values.
  - Eigenvectors of the Covariance matrix are actually the directions of the axes where there is the most variance (most information) and those are called Principal Components.
  - Eigenvalues are the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component.
- **Final output:** It is the dot product of the standardized matrix and the eigenvectors matrix .

PCA - Pros & Cons

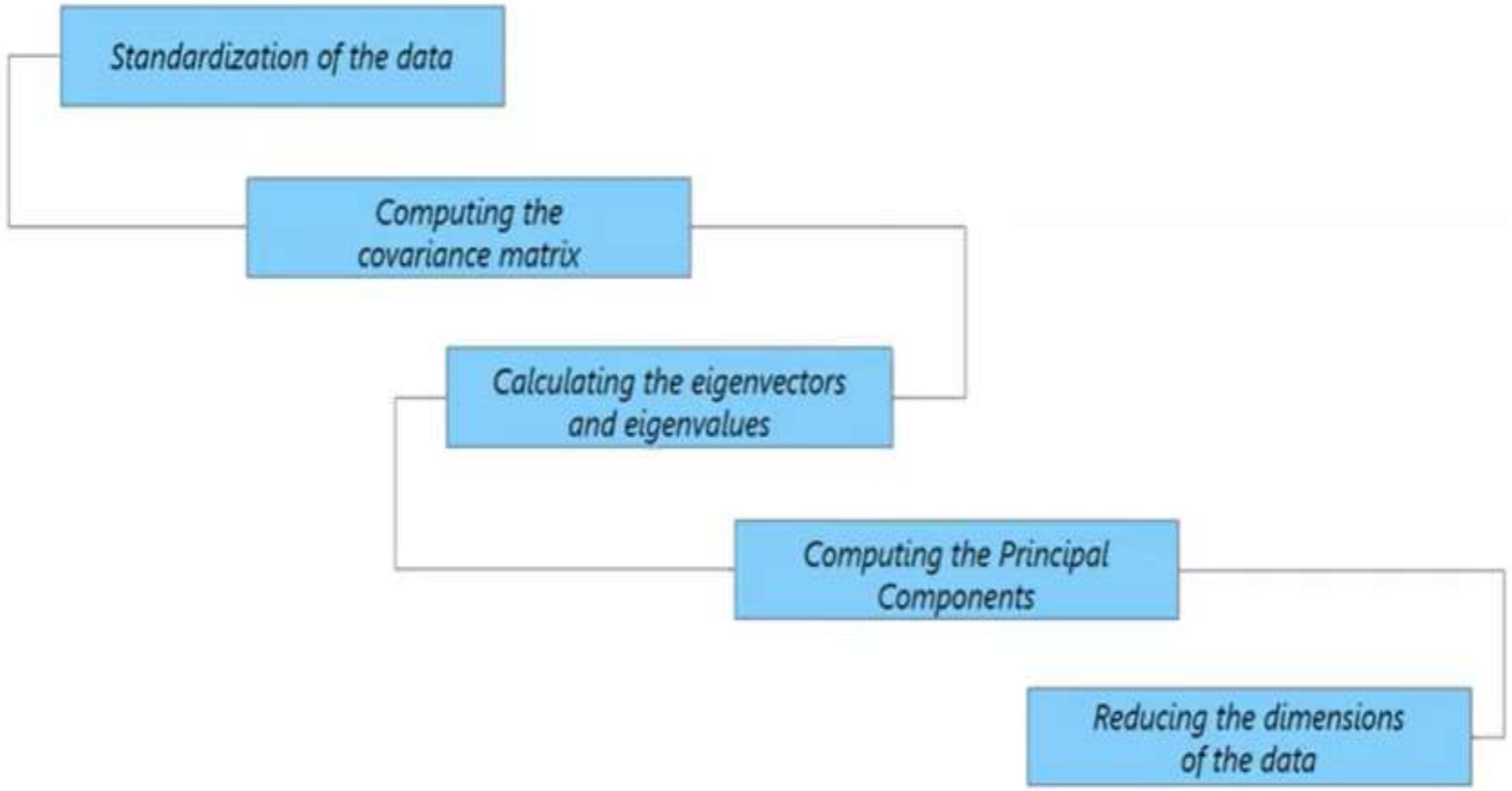
Advantages of Principal Component Analysis:

- Correlated features are removed
- Enhances the performance of the algorithm
- Enhanced Visualization

Disadvantages of Principal Component Analysis:

- The major components are difficult to comprehend
- Data normalization is required
- Loss of information

PCA – Step by Step Process


Standardization of the data
Computing the covariance matrix
Calculating the eigenvectors and eigenvalues
Computing the Principal Components
Reducing the dimensions of the data

PCA – Step by Step Process

Steps Involved in the PCA

Step 1: **Standardize the dataset.**

Step 2: **Calculate the covariance matrix for the features in the dataset.**

Step 3: **Calculate the eigenvalues and eigenvectors for the covariance matrix.**

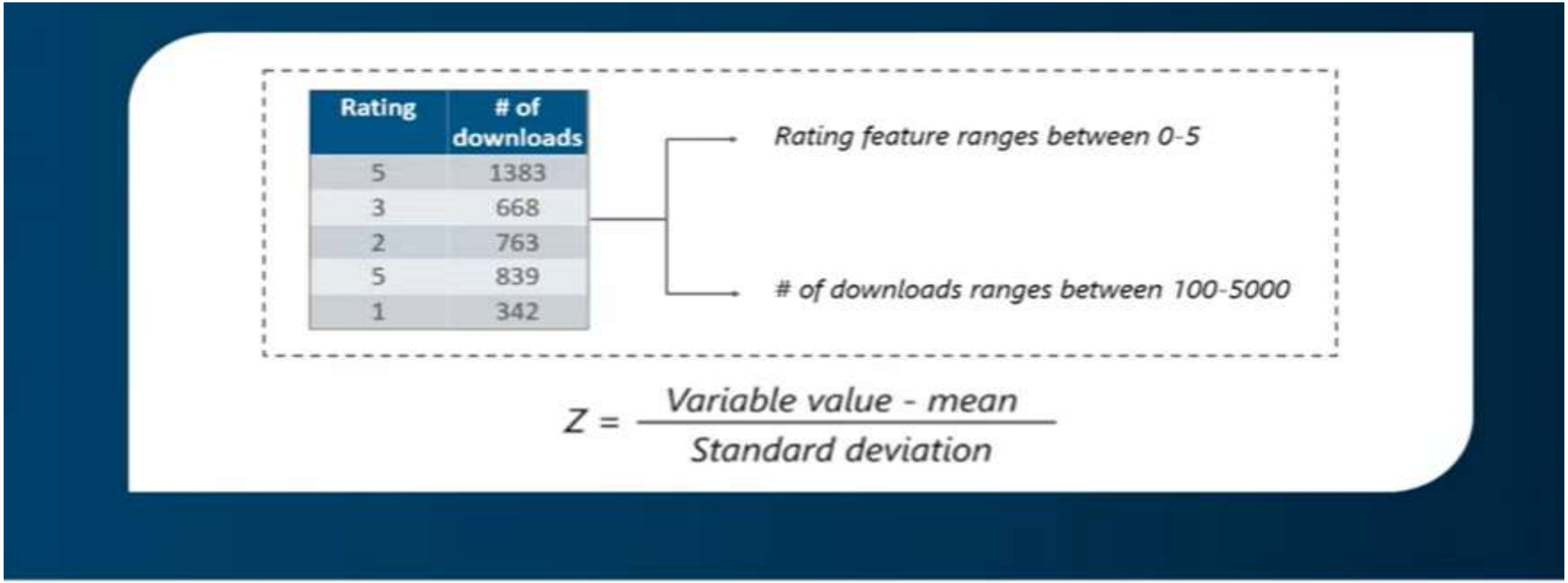
Step 4: **Sort eigenvalues in descending order and then consider the corresponding eigenvectors.**

Step 5: **Pick k eigenvalues and form a matrix of eigenvectors.**

Step 6: Transform the original matrix.

Transformed Data = Standardized matrix * top k eigenvectors

PCA – Step by Step Process



Rating	# of downloads
5	1383
3	668
2	763
5	839
1	342

Rating feature ranges between 0-5

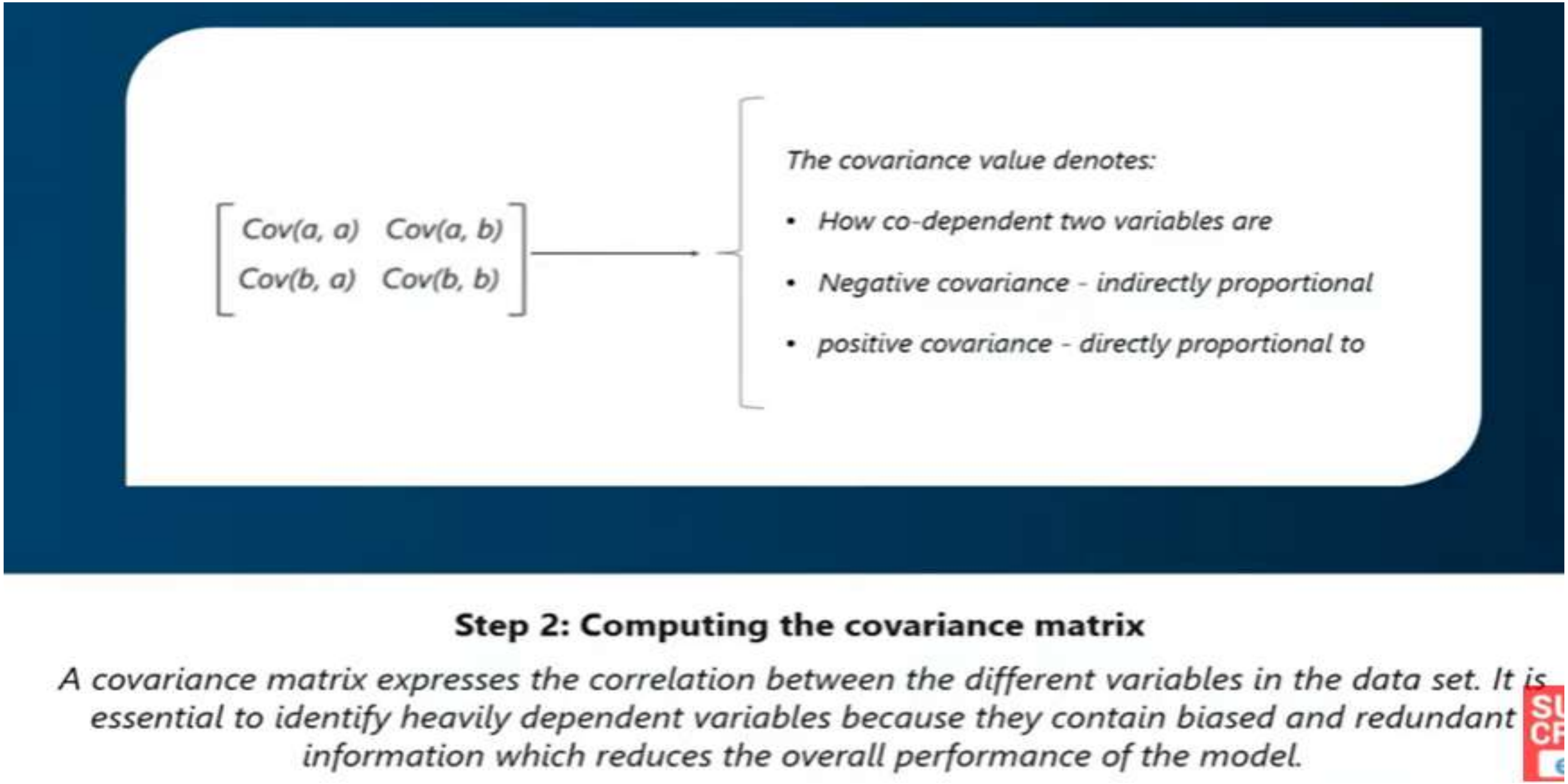
# of downloads ranges between 100-5000

$$Z = \frac{\textit{Variable value - mean}}{\textit{Standard deviation}}$$

Step 1: Standardization of the data

Standardization is all about scaling your data in such a way that all the variables and their values lie within a similar range.

PCA – Step by Step Process



$$\begin{bmatrix} Cov(a, a) & Cov(a, b) \\ Cov(b, a) & Cov(b, b) \end{bmatrix} \longrightarrow$$

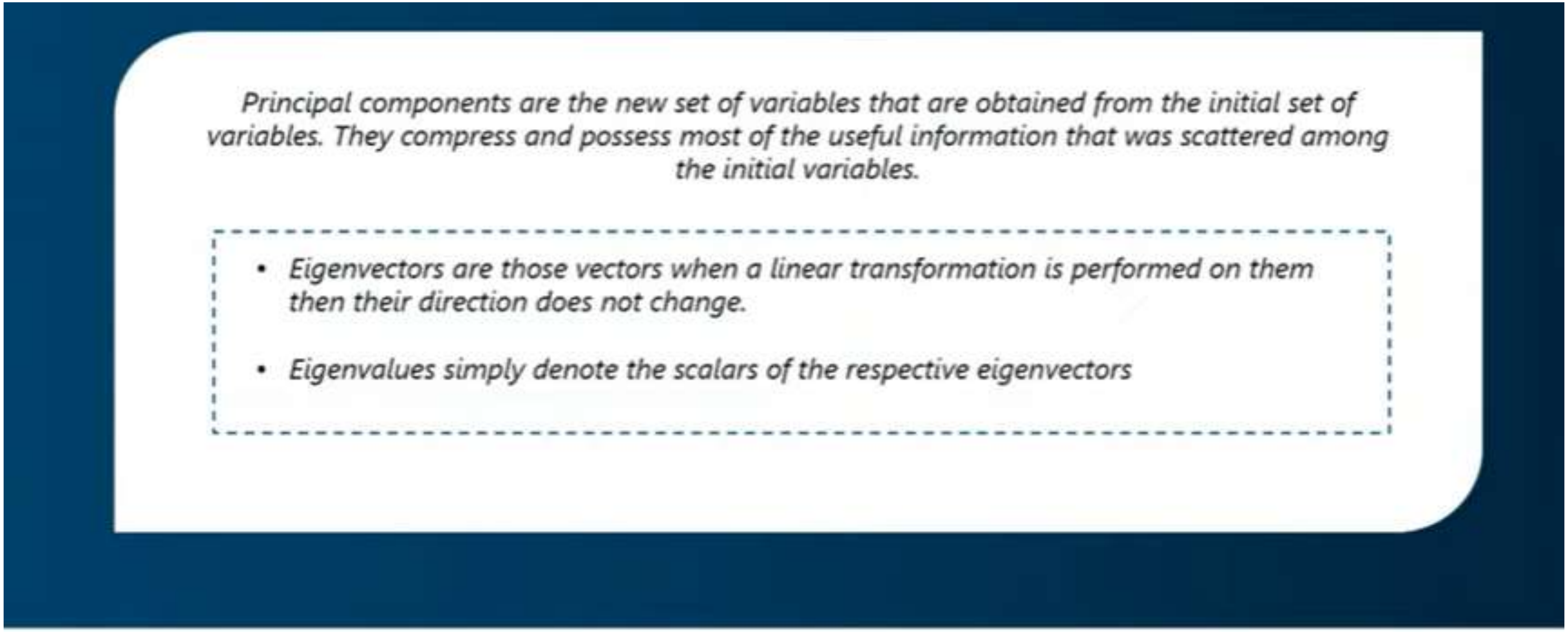
The covariance value denotes:

- *How co-dependent two variables are*
- *Negative covariance - indirectly proportional*
- *positive covariance - directly proportional to*

Step 2: Computing the covariance matrix

A covariance matrix expresses the correlation between the different variables in the data set. It is essential to identify heavily dependent variables because they contain biased and redundant information which reduces the overall performance of the model.

PCA – Step by Step Process



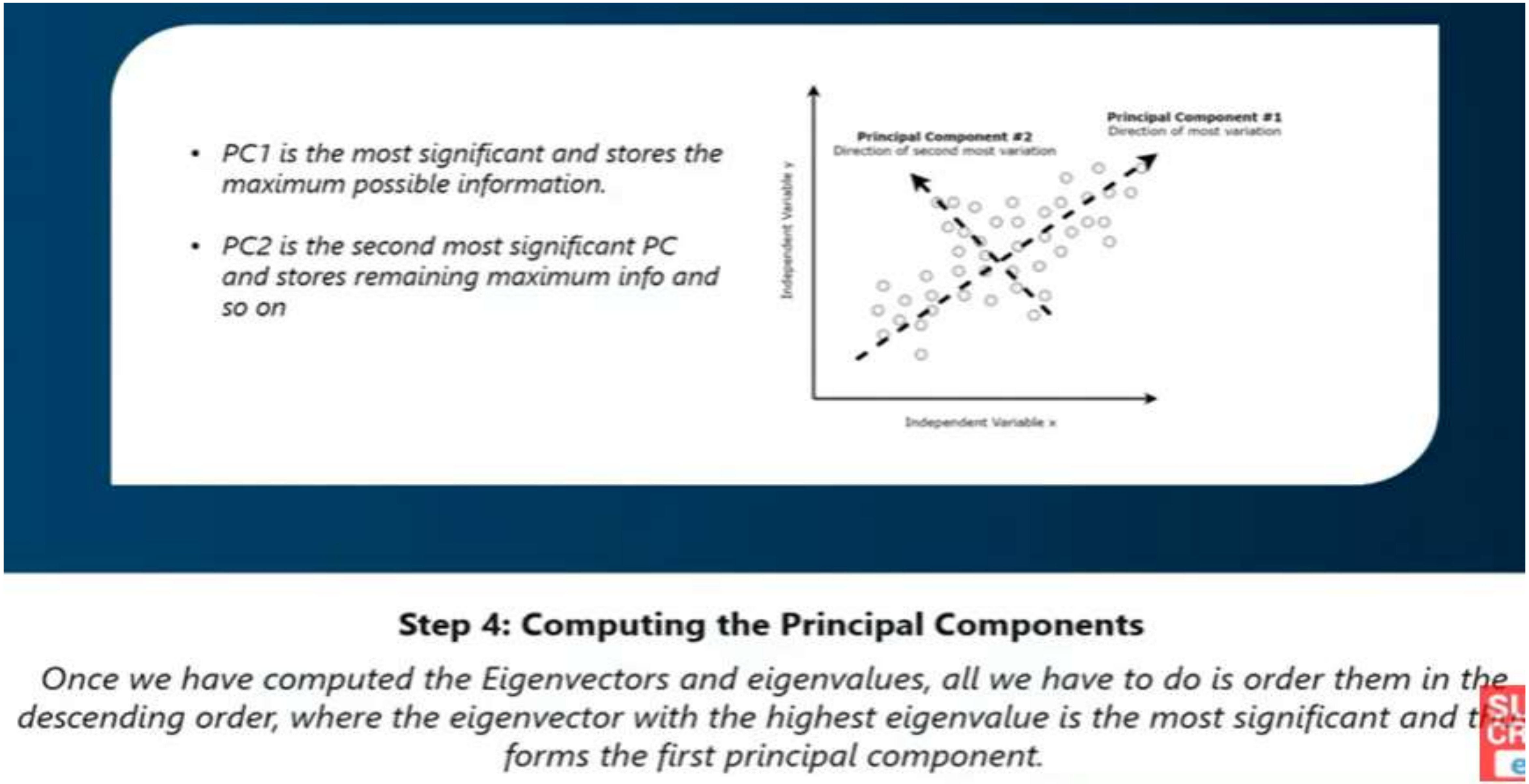
Principal components are the new set of variables that are obtained from the initial set of variables. They compress and possess most of the useful information that was scattered among the initial variables.

- *Eigenvectors are those vectors when a linear transformation is performed on them then their direction does not change.*
- *Eigenvalues simply denote the scalars of the respective eigenvectors*

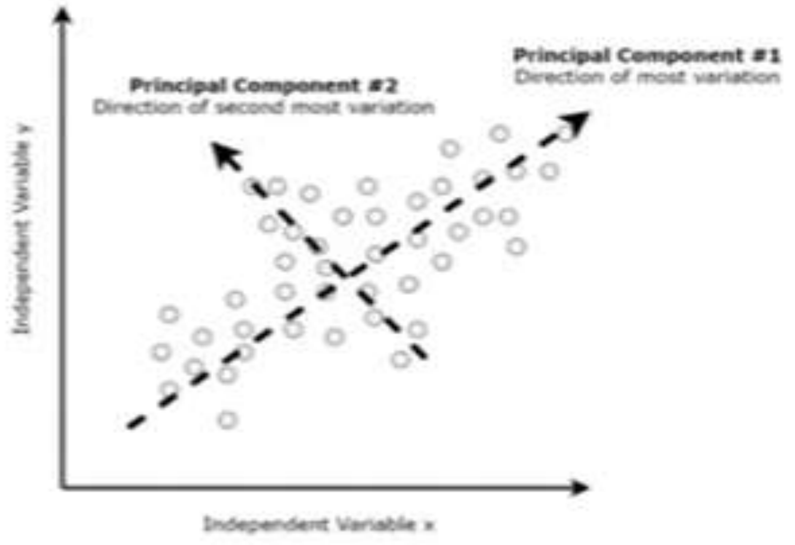
Step 3: Calculating the Eigenvectors and Eigenvalues

Eigenvectors and eigenvalues are the mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of the data set.

PCA – Step by Step Process



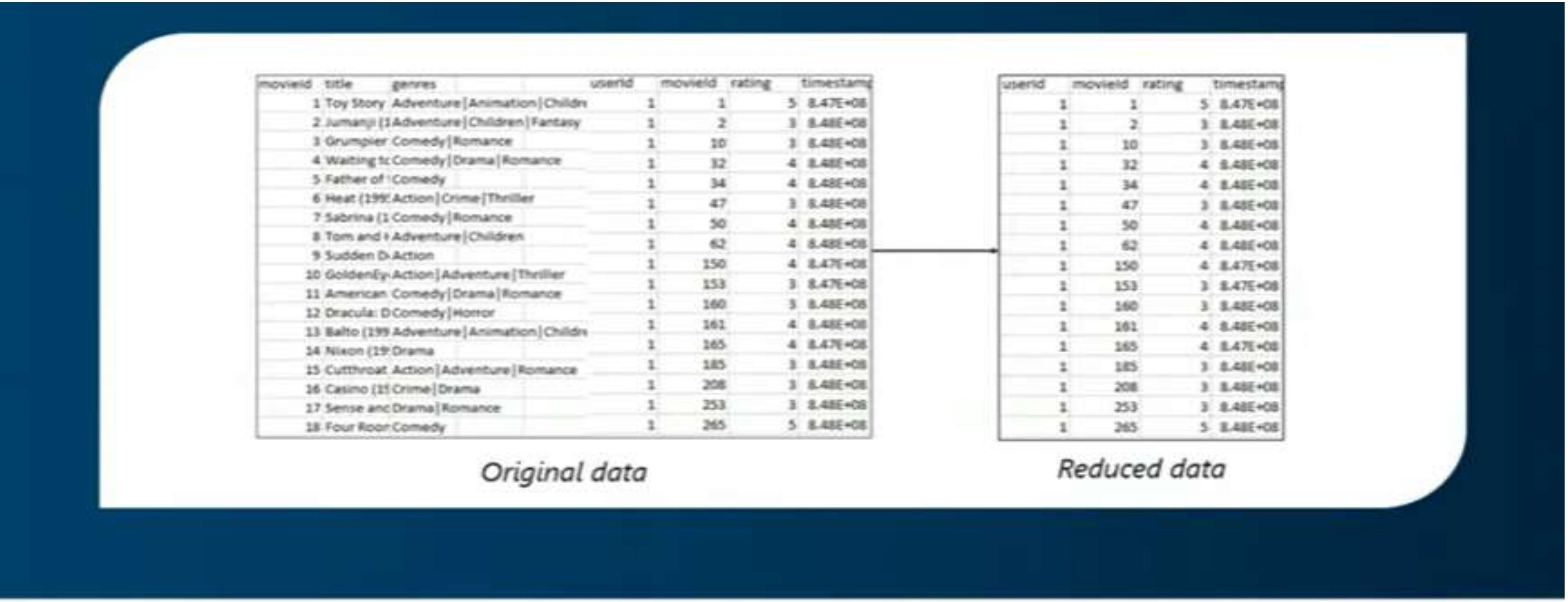
- *PC1 is the most significant and stores the maximum possible information.*
- *PC2 is the second most significant PC and stores remaining maximum info and so on*



Step 4: Computing the Principal Components

Once we have computed the Eigenvectors and eigenvalues, all we have to do is order them in the descending order, where the eigenvector with the highest eigenvalue is the most significant and t forms the first principal component.

PCA – Step by Step Process



movieId	title	genres	userId	movieId	rating	timestam
1	Toy Story	Adventure|Animation|Childn	1	1	5	8.47E+08
2	Jumanji (1	Adventure|Children|Fantasy	1	2	3	8.48E+08
3	Grumpier	Comedy|Romance	1	10	3	8.48E+08
4	Waiting tc	Comedy|Drama|Romance	1	32	4	8.48E+08
5	Father of	Comedy	1	34	4	8.48E+08
6	Heat (199	Action|Crime|Thriller	1	47	3	8.48E+08
7	Sabrina (1	Comedy|Romance	1	50	4	8.48E+08
8	Tom and I	Adventure|Children	1	62	4	8.48E+08
9	Sudden D	Action	1	150	4	8.47E+08
10	GoldenEy	Action|Adventure|Thriller	1	153	3	8.47E+08
11	American	Comedy|Drama|Romance	1	160	3	8.48E+08
12	Dracula: D	Comedy|Horror	1	161	4	8.48E+08
13	Balto (199	Adventure|Animation|Childn	1	165	4	8.47E+08
14	Nixon (19	Drama	1	185	3	8.48E+08
15	Cutthroat	Action|Adventure|Romance	1	208	3	8.48E+08
16	Casino (19	Crime|Drama	1	253	3	8.48E+08
17	Sense anc	Drama|Romance	1	265	5	8.48E+08
18	Four Roor	Comedy				

Original data

userId	movieId	rating	timestam
1	1	5	8.47E+08
1	2	3	8.48E+08
1	10	3	8.48E+08
1	32	4	8.48E+08
1	34	4	8.48E+08
1	47	3	8.48E+08
1	50	4	8.48E+08
1	62	4	8.48E+08
1	150	4	8.47E+08
1	153	3	8.47E+08
1	160	3	8.48E+08
1	161	4	8.48E+08
1	165	4	8.47E+08
1	185	3	8.48E+08
1	208	3	8.48E+08
1	253	3	8.48E+08
1	265	5	8.48E+08

Reduced data

Step 5: Reducing the dimensions of the data set

The last step in performing PCA is to re-arrange the original data with the final principal components which represent the maximum and the most significant information of the data se

Covariance Matrix - Example

Original Data

X **=**

X1	X2	X3
10	20	10
2	5	2
8	17	7
9	20	10
12	22	11

Centered Data

A **=**

1.8	3.2	2
-6.2	-11.8	-6
-0.2	0.2	-1
0.8	3.2	2
3.8	5.2	3

Covariance Matrix

$Cov(X) = (A^T A)/n =$

	X1	*X2*	*X3*
X1	14.2	25.3	13.5
X2	25.3	46.7	24.75
X3	13.5	24.75	13.5

Total Variation = 14.2 + 46.7 + 13.5 = 74.4

Eigenvalues and Eigenvectors

	X1	*X2*	*X3*
X1	14.2	25.3	13.5
X2	25.3	46.7	24.75
X3	13.5	24.75	13.5

Covariance Matrix

$\lambda_1 = 73.718$
$\lambda_2 = 0.384$
$\lambda_3 = 0.298$

Eigenvalues

Note: $\lambda_1 + \lambda_2 + \lambda_3 = 74.4$
(Equal to total variance in original dimensions)

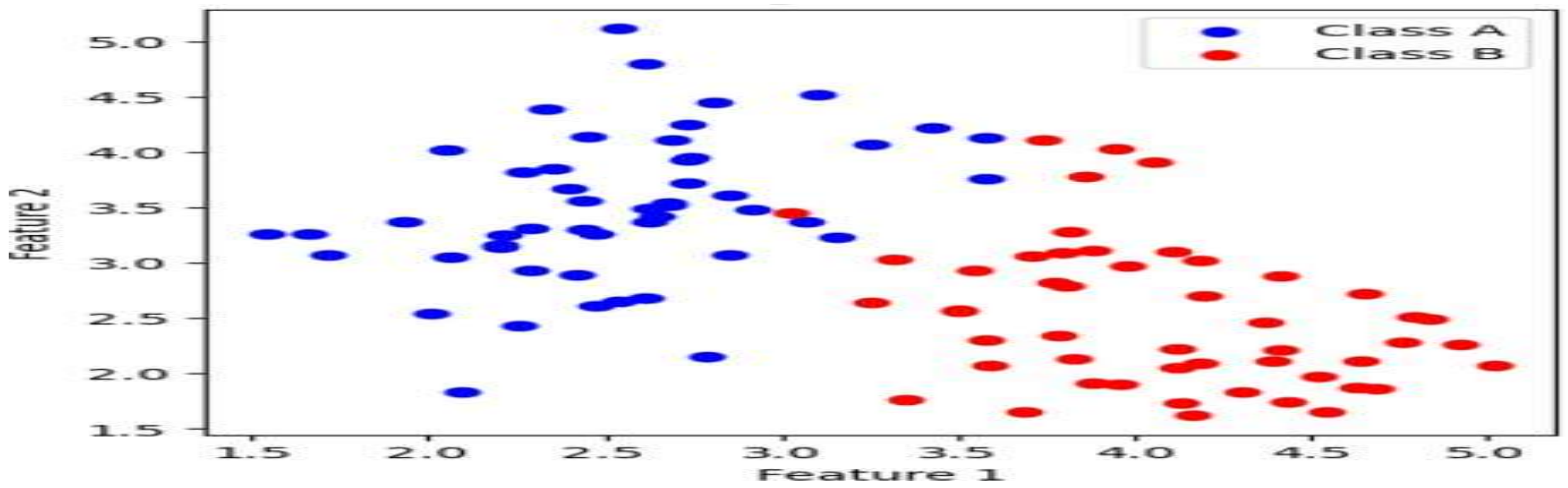
Eigenvectors

Z =

Z1	Z2	Z3
0.434	0.900	-0.044
0.795	-0.406	-0.451
0.424	-0.161	0.891

Linear Discriminant Analysis (LDA)

- LDA is a supervised **dimensionality reduction and classification technique**. Its primary purpose is to find a linear combination of features that best separates two or more classes in a dataset.
- The main goal of LDA is to find a lower-dimensional subspace that maximizes the separation between different classes while preserving the information that is relevant for discrimination.
- **LDA maximizes the ratio of the between-class variance to the within-class variance.**



LDA - Implementation

- **Compute Class Means:** Calculate the mean vectors for each class (eg. Class A and Class B) for both features.
- **Compute within Class Scatter Matrix:** SW represents the within-class scatter matrix for the dataset. It measures the spread of data within each class.

$$S_W = \sum_{i=1}^{c} S_i$$

where

$$S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

(scatter matrix for every class)

and $\boldsymbol{m}_i$ is the mean vector

$$\boldsymbol{m}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in D_i}^{n} \boldsymbol{x}_k$$

- **Compute Between Class Scatter** $S_B = \sum^{c} N_i(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$

where

$\boldsymbol{m}$ is the overall mean, and $\boldsymbol{m}_i$ and N_i are the sample mean and sizes of the respective classes.

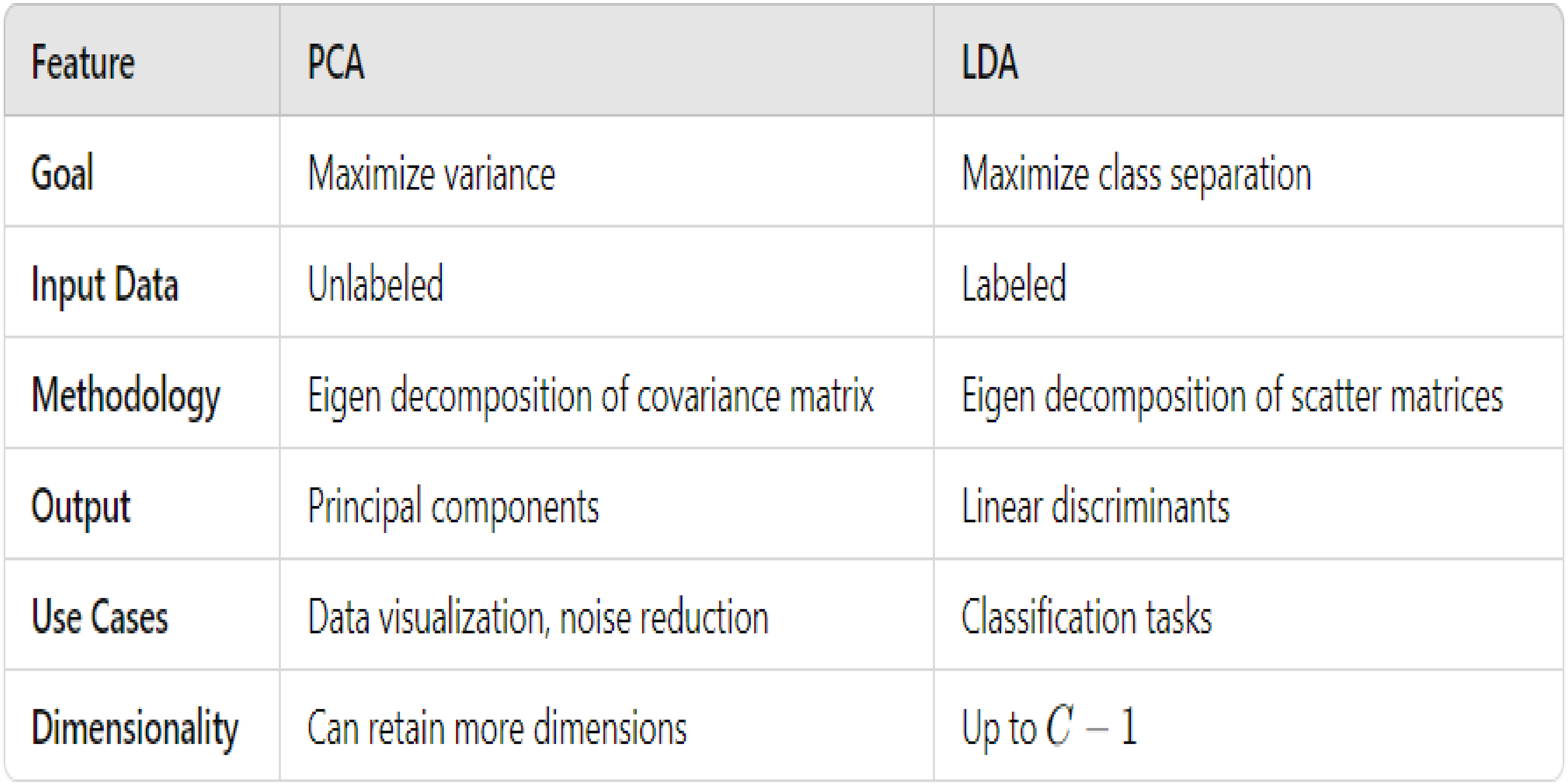
LDA - Implementation

- Solving the generalized eigenvalue problem for the matrix $S_W^{-1}S_B$

Selecting linear discriminants for the new feature subspace:

- Sorting the eigenvectors by decreasing eigenvalues
- Choosing k eigenvectors(E) with the largest eigenvalues.
- Transforming the samples onto the new subspace: Perform matrix multiplication on the input feature vector with the eigenvector with larger value. $x' = X'.E$

PCA vs LDA



Feature	PCA	LDA
Goal	Maximize variance	Maximize class separation
Input Data	Unlabeled	Labeled
Methodology	Eigen decomposition of covariance matrix	Eigen decomposition of scatter matrices
Output	Principal components	Linear discriminants
Use Cases	Data visualization, noise reduction	Classification tasks
Dimensionality	Can retain more dimensions	Up to $C - 1$

Difference between LDA & PCA

PCA

- Does feature classification
- In PCA, the shape and location of the original data changes when transformed to a different space
- PCA is unsupervised

LDA

- Does Data Classification
- Whereas LDA does not change the location but tries to provide more class separability & draw a decision region between the given classes
- LDA is supervised

Thank you

Unit II
Supervised Learning - I

SUPERVISED LEARNING-I UNIT2

Regression models:

- Simple linear regression
- Multiple linear regression
- Cost Function, Gradient Descent
- Performance Metrics: Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-Squared error, Adjusted R Square.

Classification models:

- Decision Trees - ID3, CART
- Naive Bayes
- K-Nearest Neighbours (KNN)
- Logistic regression, Multinomial logistic regression
- Support Vector Machines (SVM).

Supervised Learning

- **Classification and Regression algorithms are Supervised Learning algorithms.**
- **Both the algorithms can be used for forecasting in Machine learning and operate with the labeled datasets.**
- **Regression algorithms are used to determine continuous values such as price, income, age, etc.**
- **Classification algorithms are used to forecast or classify the distinct values such as Real or False, Male or Female, Spam or Not Spam, etc.**
- **Types of Regression Algorithms:**
  - **Simple Linear Regression**
  - **Multiple Linear Regression**
- **Types of Classification Algorithms:**
  - **Decision Trees**
  - **Naïve Bayes**
  - **K Nearest Neighbors**
  - **Logistic Regression**

Regression Models

Supervised Learning - Regression

Regression models:

- Simple linear regression
- Multiple linear regression

Cost Function, Gradient Descent

Performance Metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-Squared error
- Adjusted R Square

Types of Regression

- Univariate Vs Multivariate
  - **Univariate :** One dependent and one independent variable
  - **Multivariate :** Multiple independent and multiple dependent variables
- Linear Vs Non-Linear
  - **Linear :** Relationship is linear between dependent and independent variables
  - **Nonlinear :** Relationship is non-linear between dependent and independent variables – function is quadratic, polynomial--When the best fitting line is not a straight line but a curve, it is referred to as Non-Linear Regression.
- Simple vs Multiple
  - **Simple Linear :** one dependent and one independent variable
  - **Multiple Linear :** one dependent and many independent variable

Linear vs Non-Linear

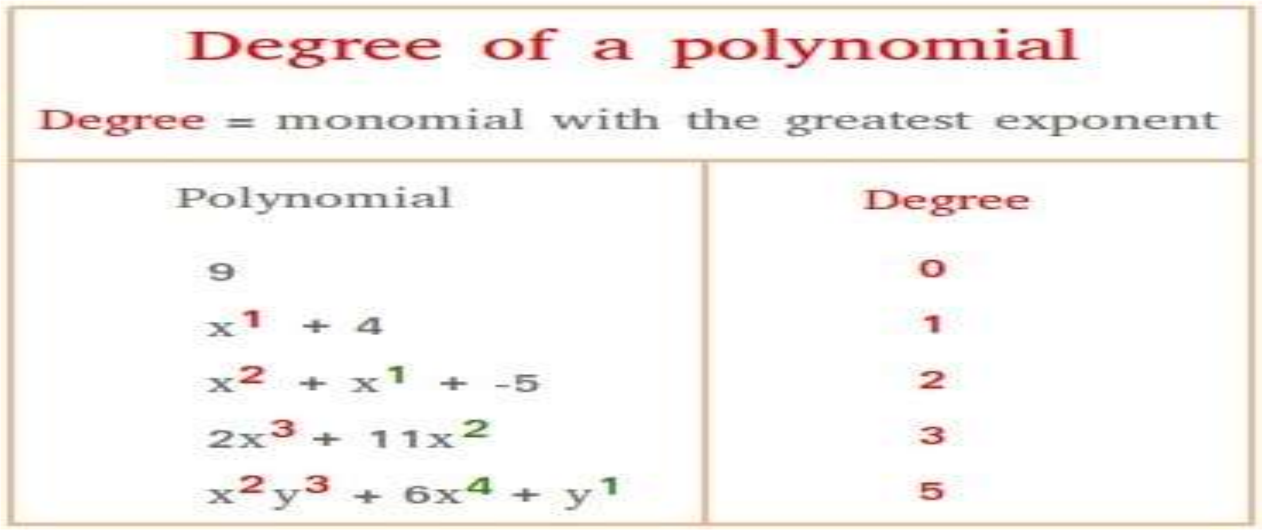
A polynomial's degree is the highest or the greatest power of a variable in a polynomial equation. The degree indicates the highest exponential power in the polynomial (ignoring the coefficients).

- Linear Regression - The degree of a polynomial is 1 in linear regression.

$$y = w_0 + w_1 x$$

- Non-linear Regression - The degree of a polynomial > 1

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



Degree of a polynomial

Degree = monomial with the greatest exponent

Polynomial	Degree
9	0
$x^1 + 4$	1
$x^2 + x^1 + -5$	2
$2x^3 + 11x^2$	3
$x^2y^3 + 6x^4 + y^1$	5

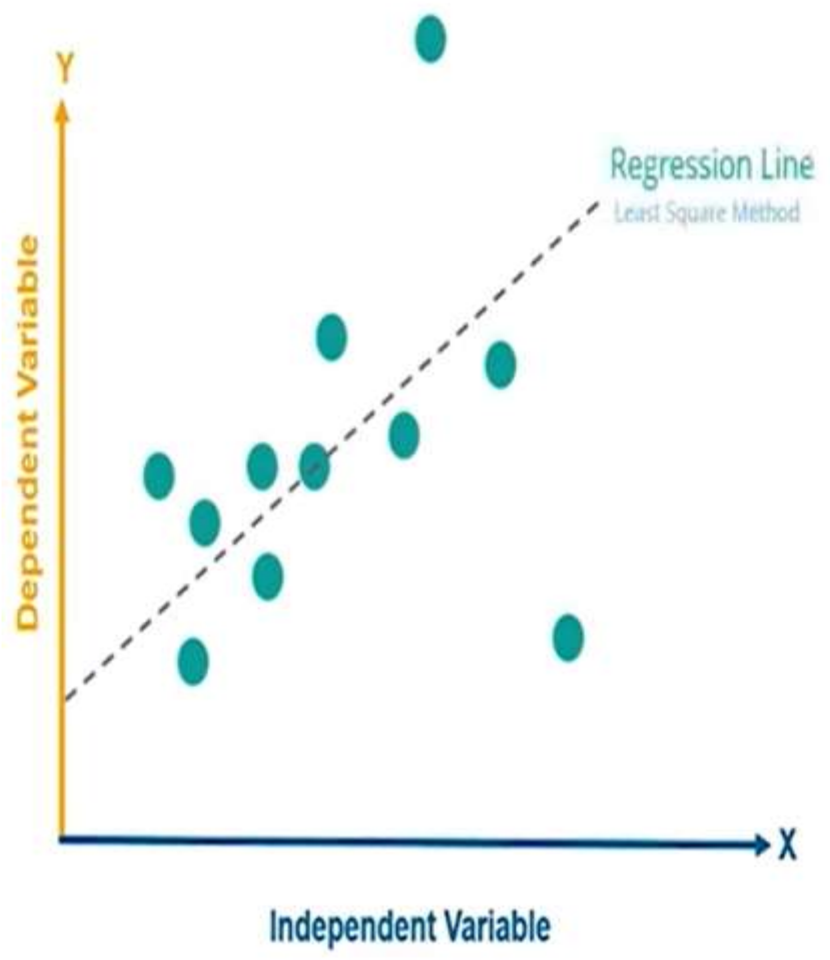
Regression

- Regression in machine learning consists of mathematical methods that allow data scientists to predict a continuous outcome (y) based on the value of one or more predictor variables (X).
- Regression analysis is a form of predictive modeling technique which investigates the relationship between a dependent and independent variable.
- The aim of Regression is to find a function of X to predict y. y = f(X)
    - X: *independent variable also known as predictor variable, Regressor variable, exploratory variable, input variable*
    - y: dependent variable also known as *Response variable, Regressand, predicted variable, output variable*

- Types of Regression Models:
    - Simple Linear Regression: one dependent and one independent variable
    - Multiple Linear Regression : one dependent and multiple independent variables

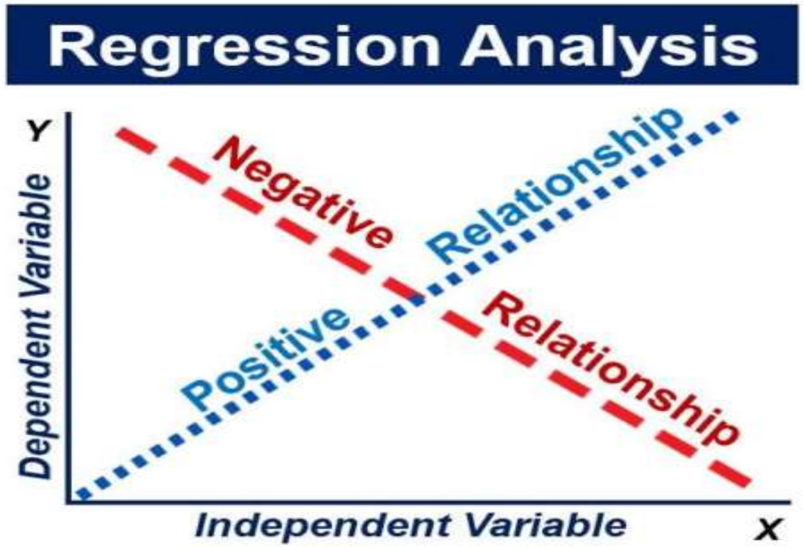
Linear Regression

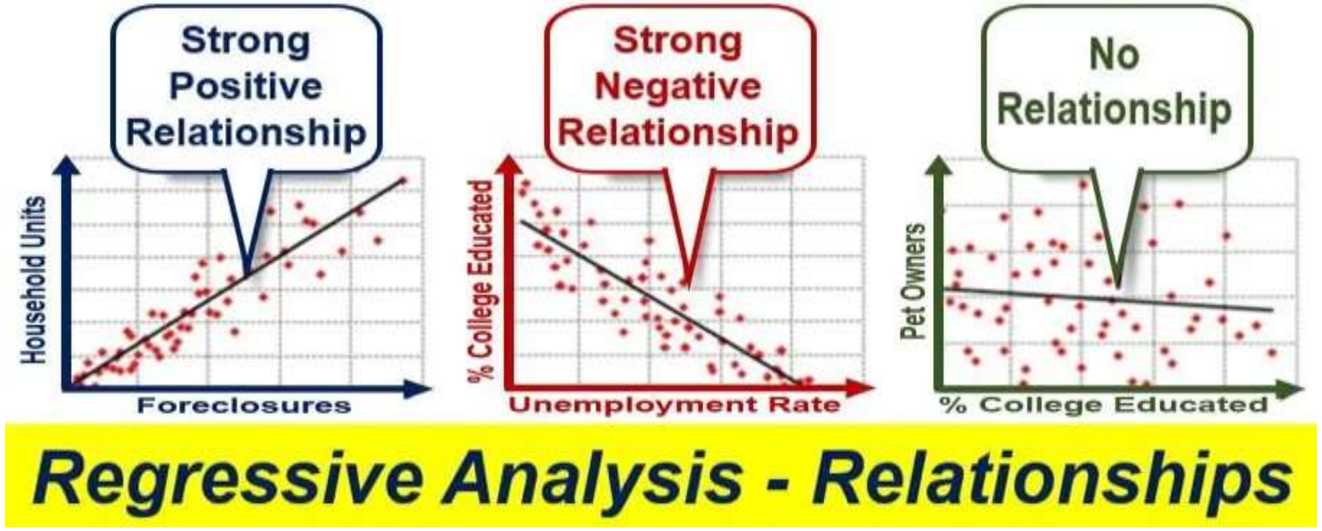
- Linear regression is used to predict the relationship between two variables by applying a linear equation to observed data. There are two types of variable, one variable is called an independent variable, and the other is a dependent variable. Linear regression is commonly used for predictive analysis.
- The measure of the relationship between two variables is shown by the correlation coefficient. The range of the coefficient lies between -1 to +1. This coefficient shows the strength of the association of the observed data between two variables.



Regression Analysis

- Is there a relationship between these variables?
- Is the relationship linear and how strong is the relationship?
- How accurately can we estimate the relationship?
- How good is the model for prediction purposes?
- The strongest linear relationship occurs when the slope is 1. This means that when one variable increases by one, the other variable also increases by the same amount.





Simple Linear Regression

- Least Squares method is used to determine the best-fitting line for the given data by reducing the sum of the squares of the vertical deviations from each data point to the line.
- If a point rests on the fitted line accurately, then the value of its perpendicular deviation is 0.
- Linear regression determines the straight line, known as the least-squares regression line or LSRL.
- Suppose y is a dependent variable and X is an independent variable, then the population regression line is given by the equation; y
- When a random sample of observations is given, then the regression line is expressed as:

$$Y = b_0 + b_1X$$

where b_0 = Mean of Y - b_1*Mean of X

$$b_1 = \frac{\Sigma(x-\bar{x}) * (y-\bar{y})}{\Sigma(x-\bar{x})^2}$$

Where $\mathbf{b_0}$ is a constant

- $\mathbf{b_1}$ is the regression coefficient
- X is the independent variable
- $\hat{y}$ is known as the predicted value of the dependent variable.

Regression - Performance Metrics

- The performance of a Regression model is reported as errors in the prediction.
- Following are the popular metrics that are used to evaluate the performance of Regression models.
- **Mean Absolute Error:** MAE is one of the simplest metrics, which measures the absolute difference between actual and predicted values, where absolute means taking a number as positive.

$$MAE = 1/N \sum |Y - Y'|$$

- Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.
- **Mean Squared Error:** MSE measures the average of the Squared difference between predicted values and the actual value given by the model.

$$MSE = 1/N \sum (Y - Y')^2$$

- **R^2 (R-Squared Error):** R^2error is also known as **Coefficient of Determination**, which is another popular metric used for Regression model evaluation.
- The R-squared metric enables us to compare the model with a constant baseline to determine the performance of the model.
- To select the constant baseline, we need to take the mean of the data and draw the line at the mean.

Regression - Performance Metrics

$$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$$

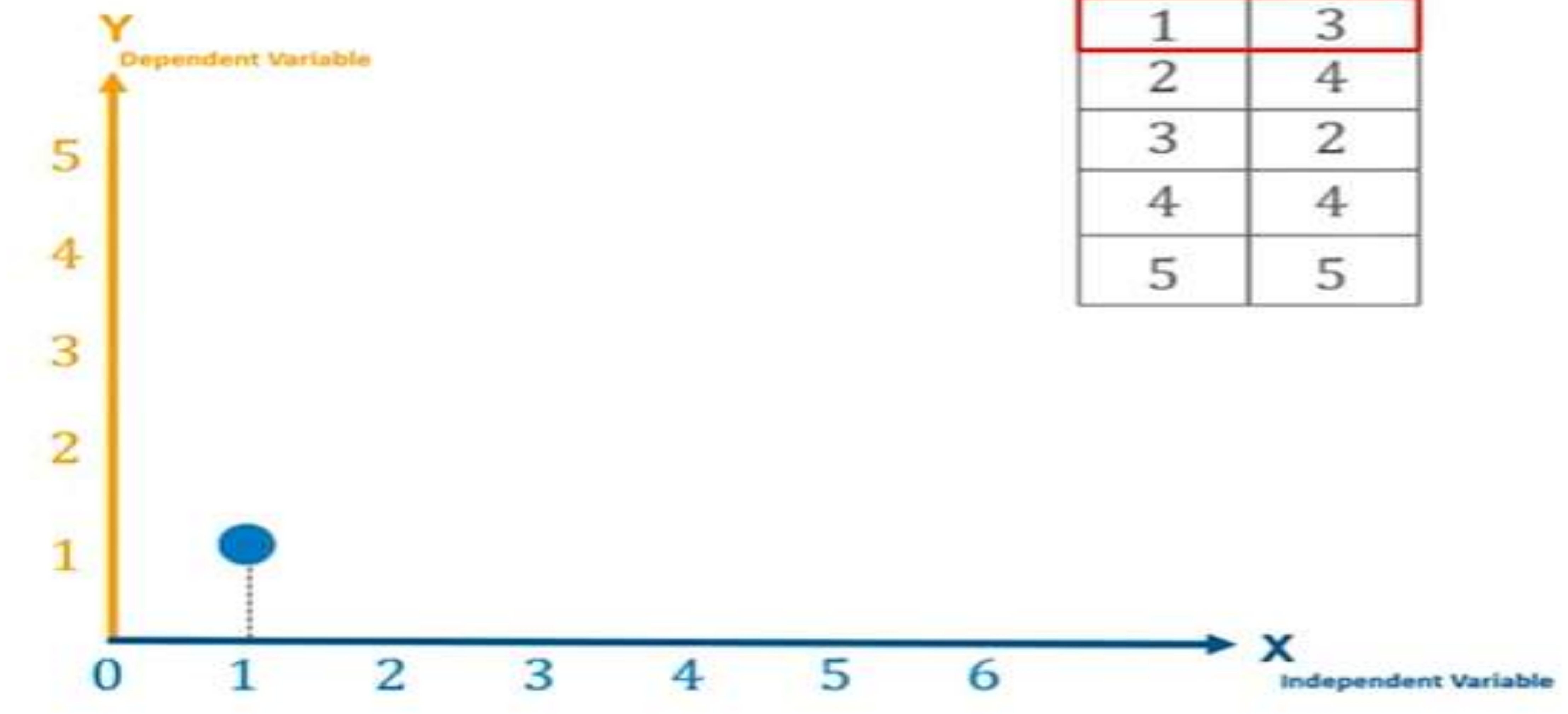
Adjusted $R^{2:}$ Adjusted R^2, as the name suggests, is the improved version of R^2 error.

- R^2 has a limitation of improvement of a score on increasing the terms, even though the model is not improving, and it may mislead the data scientists.
- To overcome the issue of R square, adjusted R squared is used, which will always show a lower value than R^2. It is because it adjusts the values of increasing predictors and only shows improvement if there is a real improvement.
- We can calculate the adjusted R squared as follows:

$$R_a^2 = 1-\left[\left(\frac{n-1}{n-k-1}\right) \times \left(1 - R^2\right)\right]$$

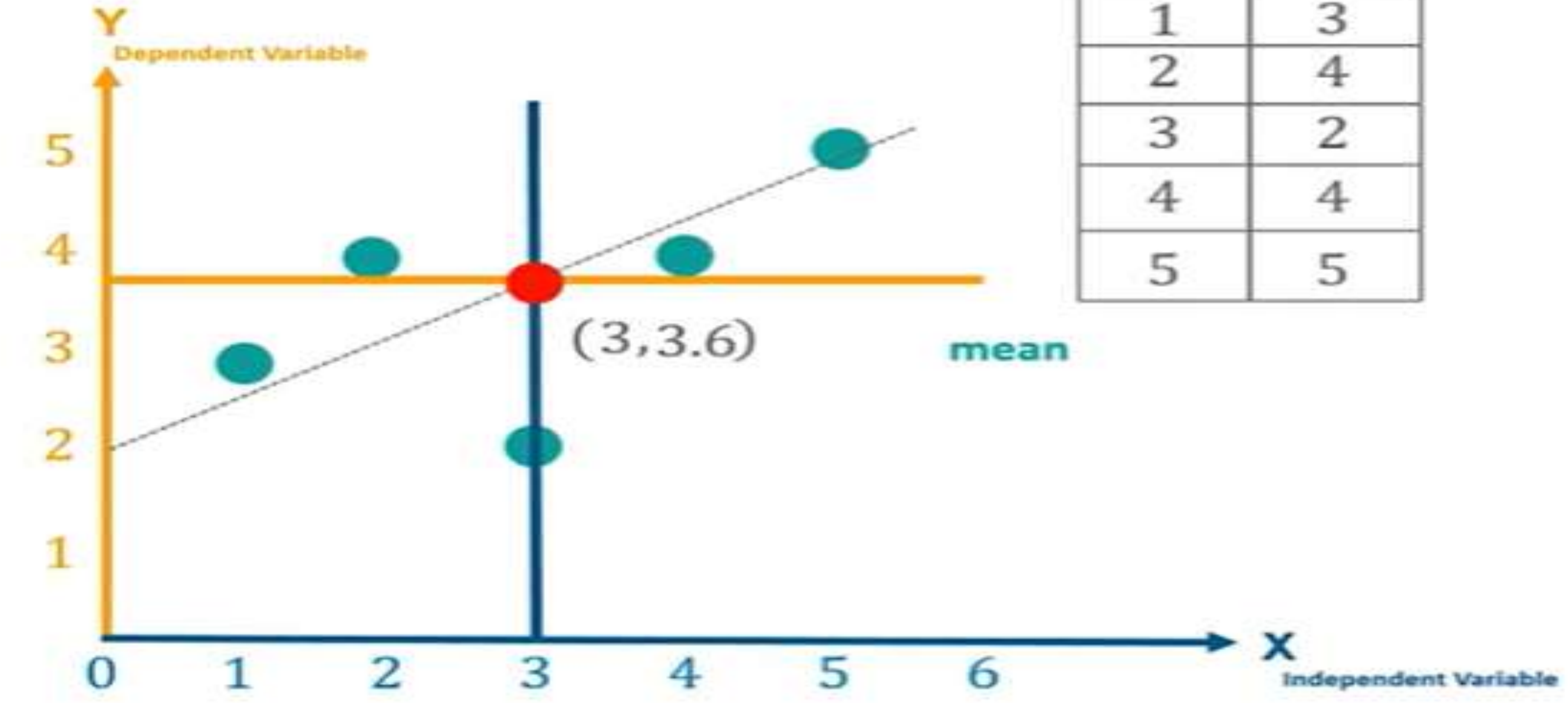
- n is the number of observations
- k denotes the number of independent variables
- R_{a_2} denotes the adjusted R_2

Simple Linear Regression – Problem 1



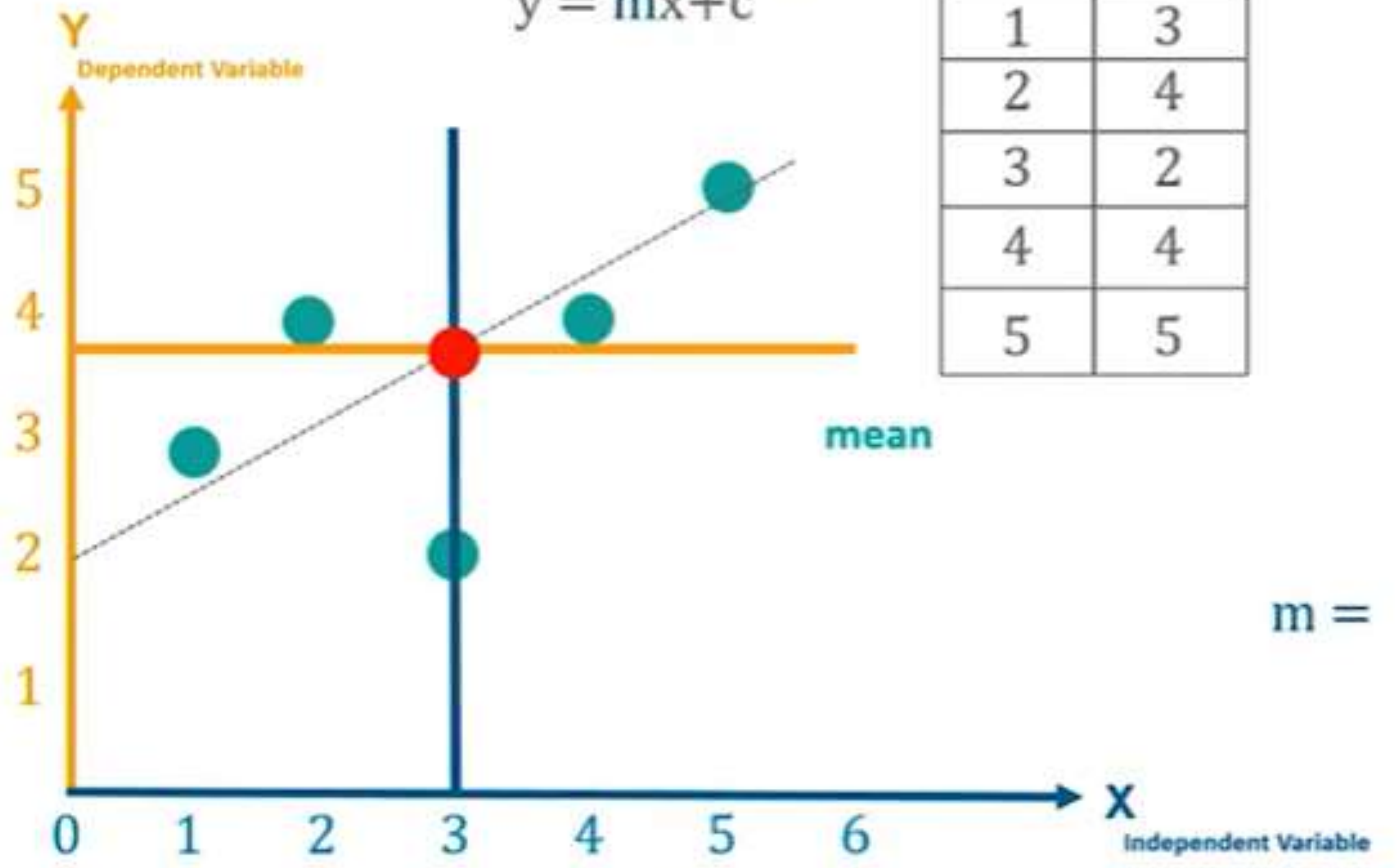
x	y
1	3
2	4
3	2
4	4
5	5

Simple Linear Regression – Problem 1



x	y
1	3
2	4
3	2
4	4
5	5

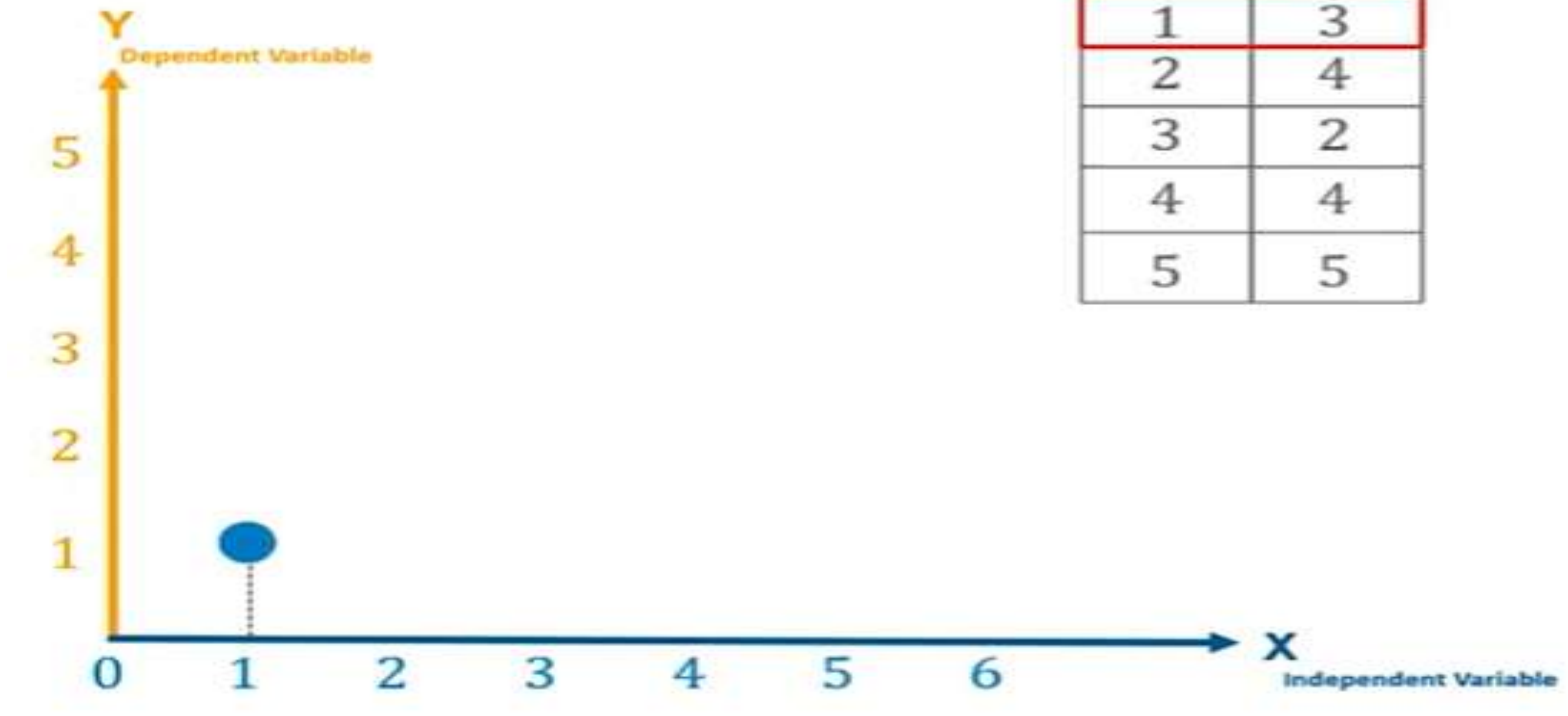
Simple Linear Regression – Problem 1



x	y
1	3
2	4
3	2
4	4
5	5

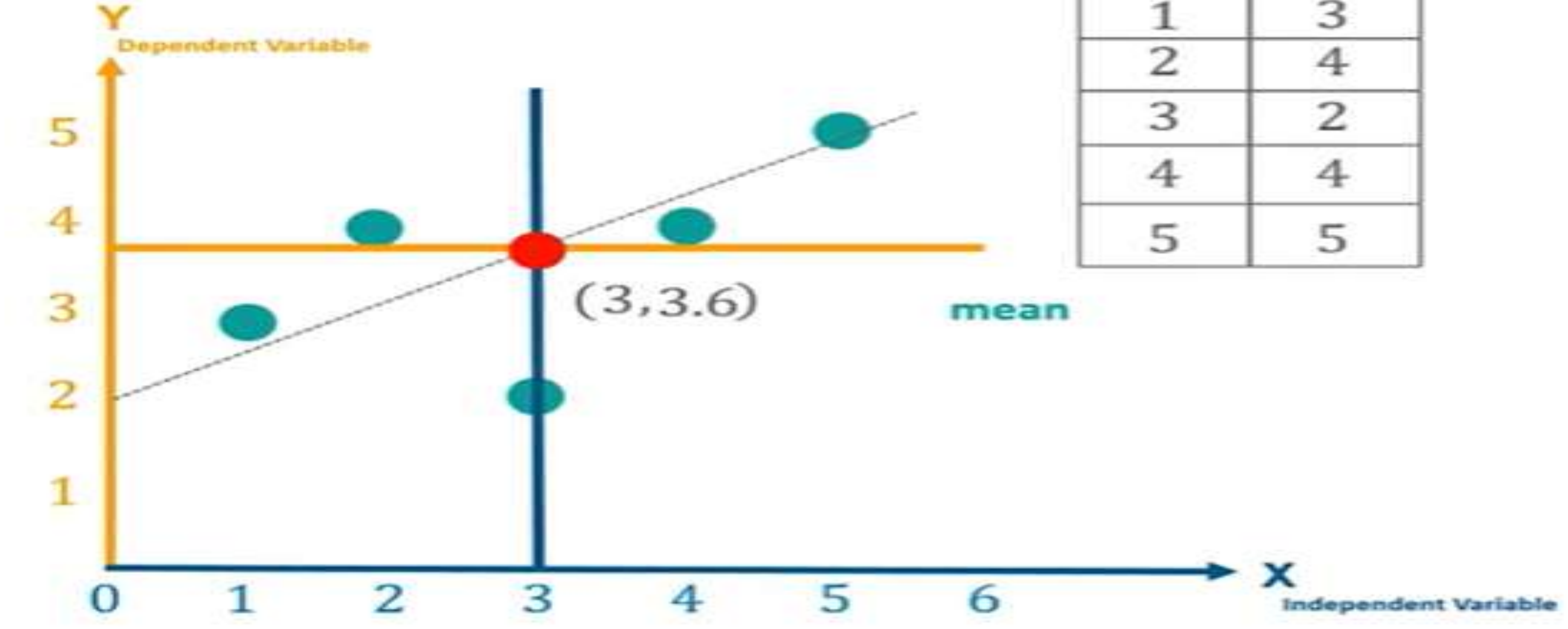
$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

Simple Linear Regression – Problem 1



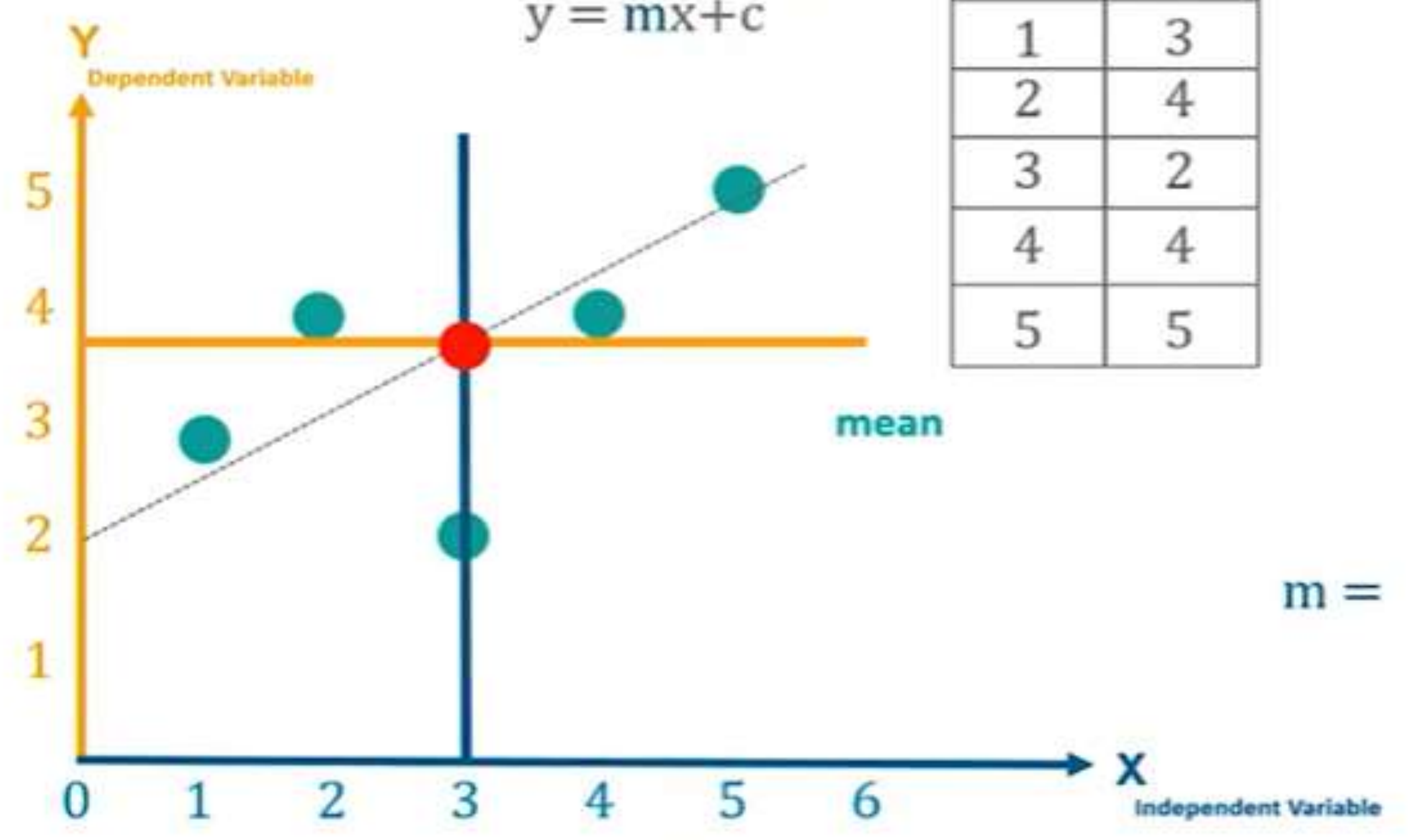
x	y
1	3
2	4
3	2
4	4
5	5

Simple Linear Regression – Problem 1



x	y
1	3
2	4
3	2
4	4
5	5

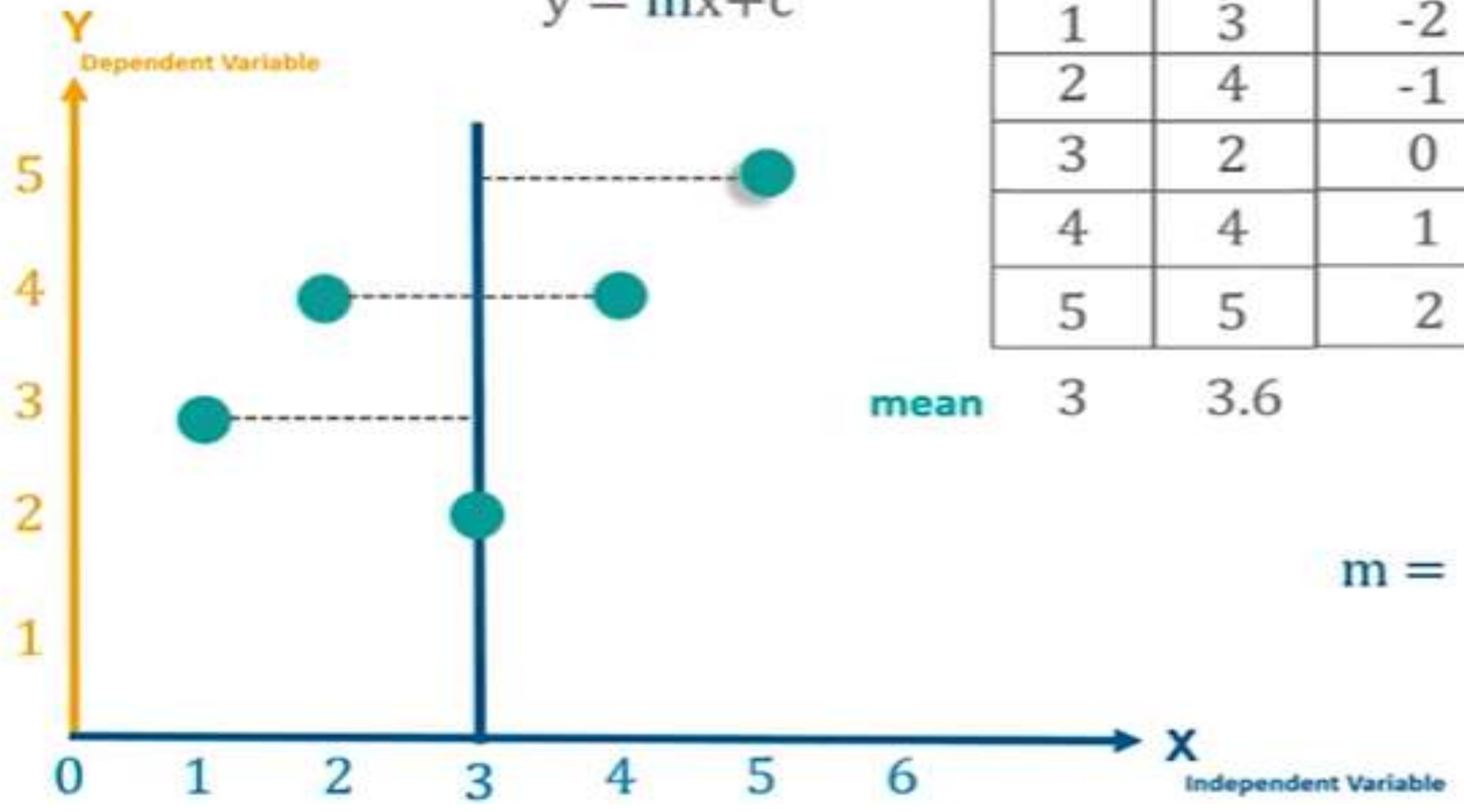
Simple Linear Regression – Problem 1



x	y
1	3
2	4
3	2
4	4
5	5

$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

Simple Linear Regression – Problem 1



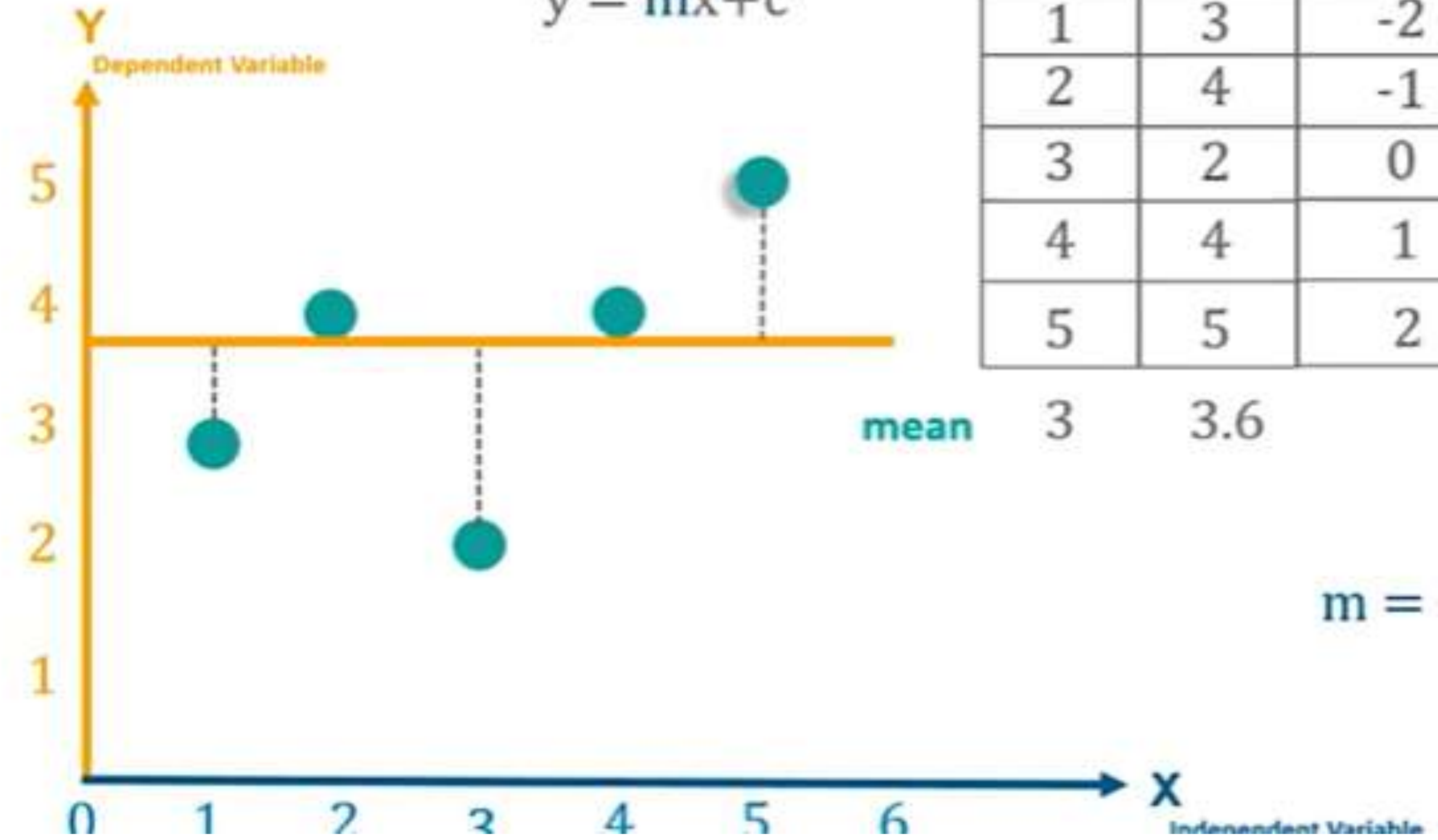
x	y	$x-\bar{x}$
1	3	-2
2	4	-1
3	2	0
4	4	1
5	5	2

mean 3 3.6

$$m = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2}$$

Simple Linear Regression – Problem 1

$y = mx + c$

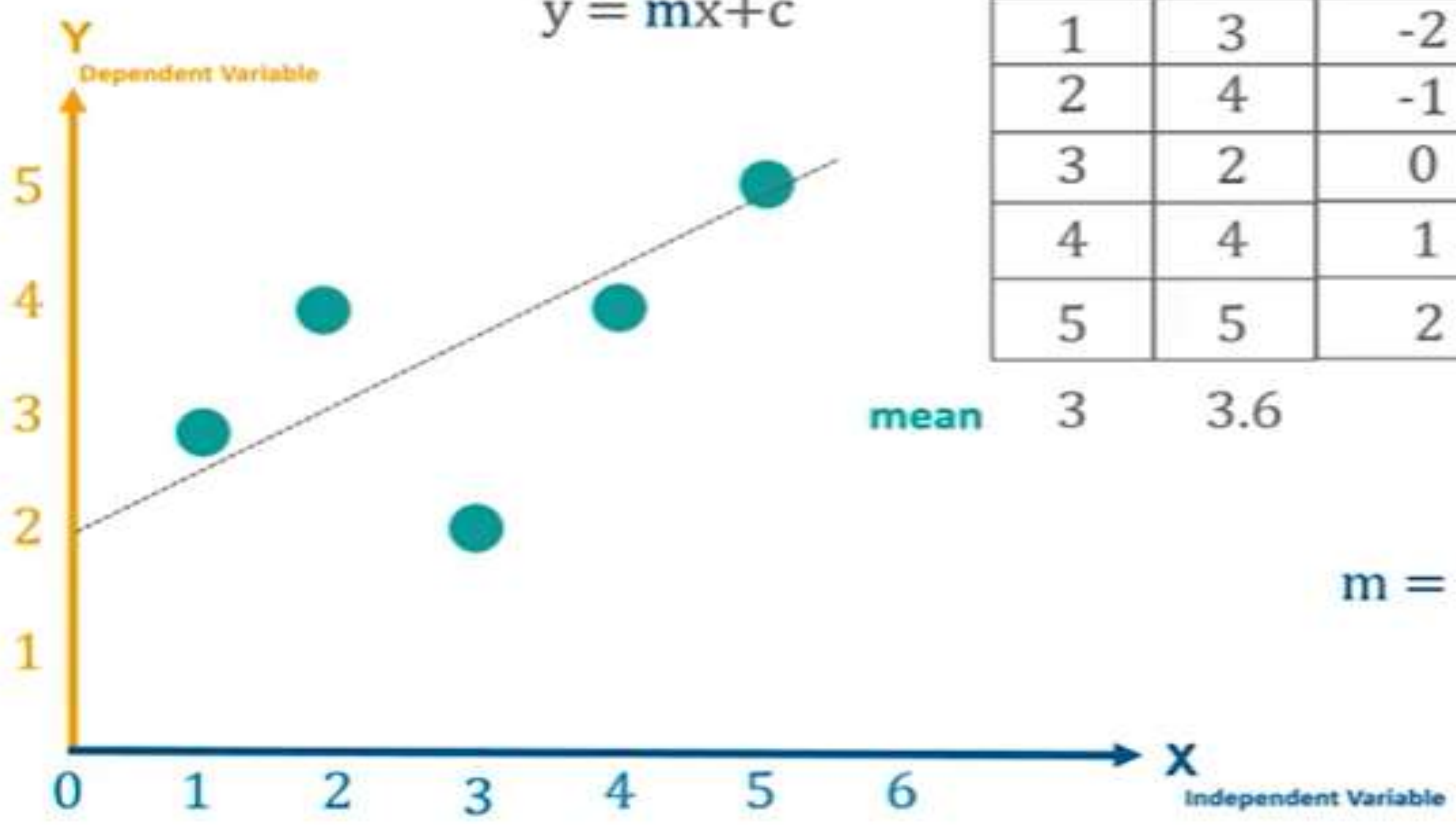


x	y	$x-\bar{x}$	$y-\bar{y}$
1	3	-2	-0.6
2	4	-1	0.4
3	2	0	-1.6
4	4	1	0.4
5	5	2	1.4

mean 3 3.6

$$m = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2}$$

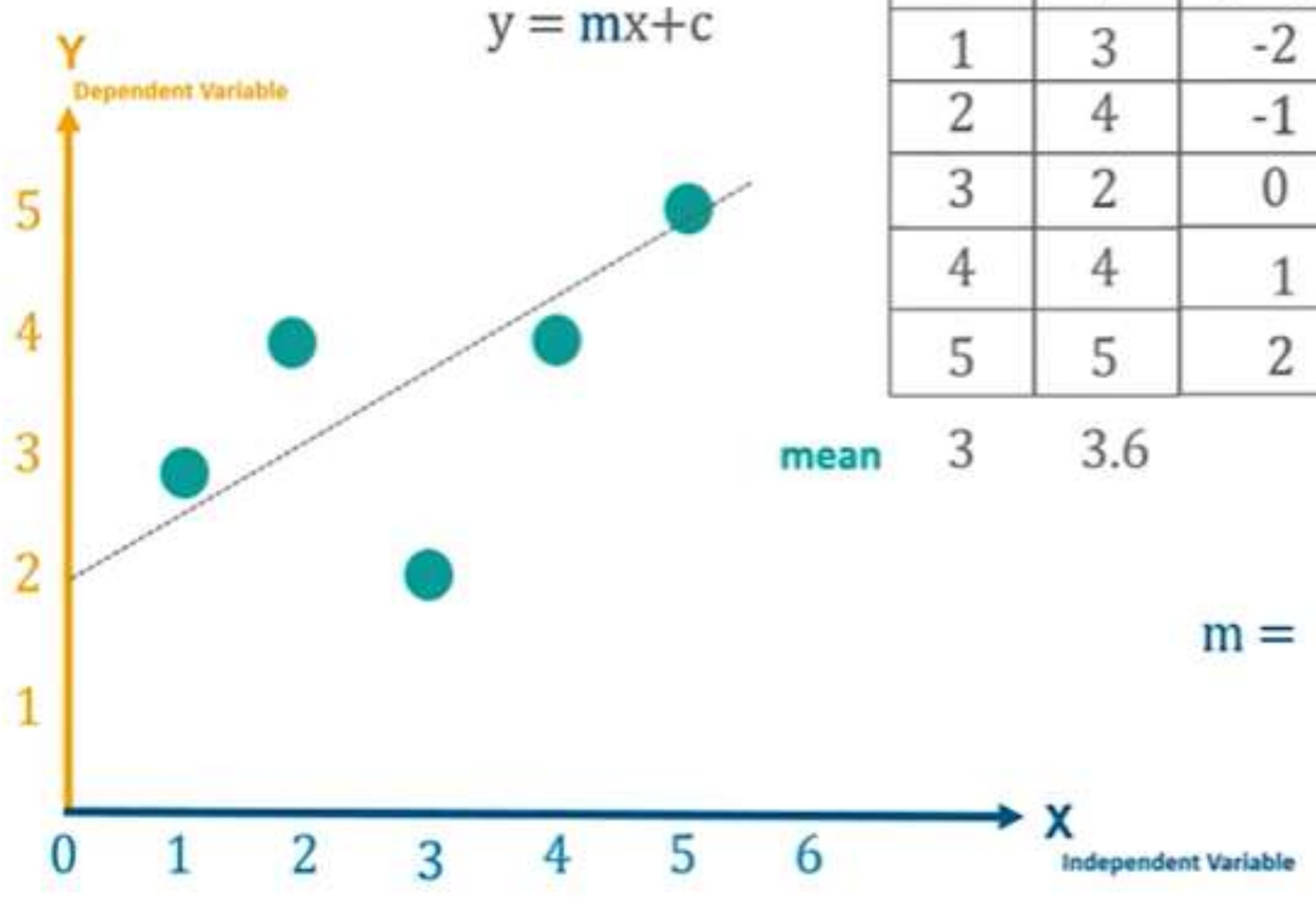
Simple Linear Regression – Problem 1



x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$
1	3	-2	-0.6	4
2	4	-1	0.4	1
3	2	0	-1.6	0
4	4	1	0.4	1
5	5	2	1.4	4
mean 3	3.6			

$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

Simple Linear Regression – Problem 1



x	y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(x-\bar{x})(y-\bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean 3	3.6			$\Sigma = 10$	$\Sigma = 4$

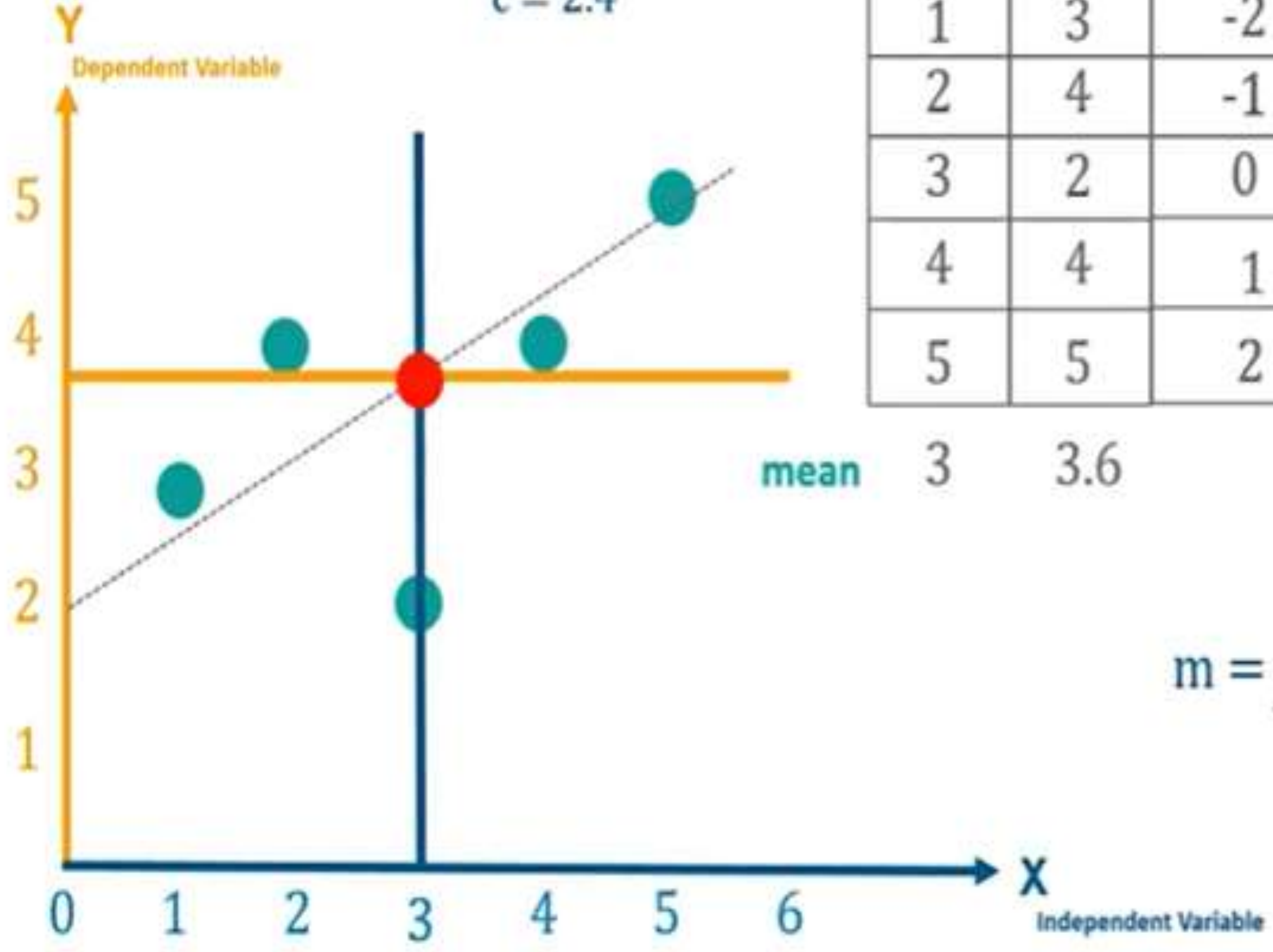
$$m = \frac{\Sigma (x-\bar{x})(y-\bar{y})}{\Sigma (x-\bar{x})^2}$$

SUB

Simple Linear Regression – Problem 1

$$y = mx + c$$

$$c = 2.4$$



	x	y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(x-\bar{x})(y-\bar{y})$
	1	3	-2	-0.6	4	1.2
	2	4	-1	0.4	1	-0.4
	3	2	0	-1.6	0	0
	4	4	1	0.4	1	0.4
	5	5	2	1.4	4	2.8
mean	3	3.6			$\Sigma = 10$	$\Sigma = 4$

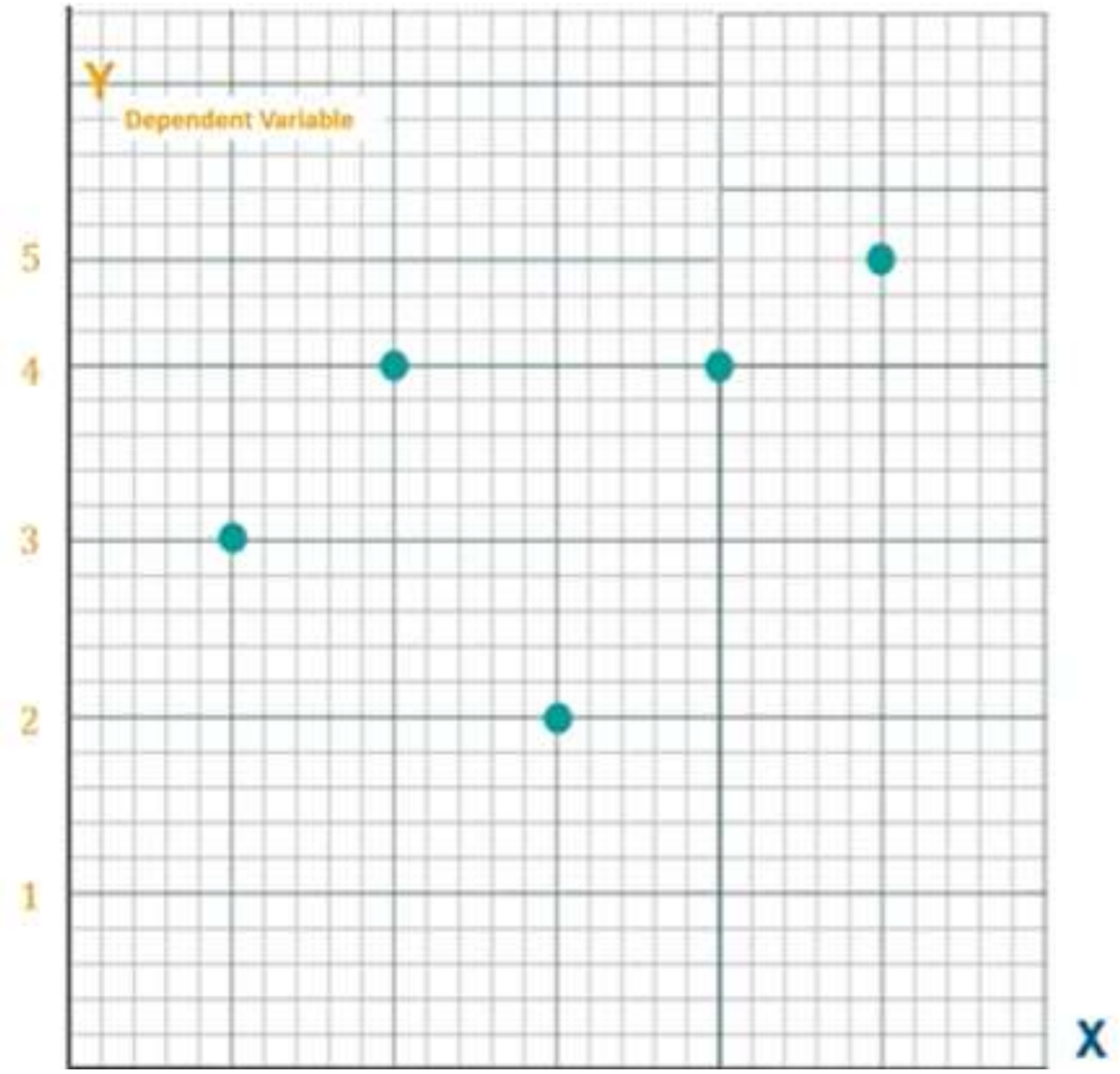
$$m = \sum \frac{(x-\bar{x})(y-\bar{y})}{(x-\bar{x})^2} = \frac{4}{10}$$

$$m = 0.4$$

$$c = 2.4$$

$$y = 0.4x + 2.4$$

Simple Linear Regression – Problem 1



$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

For given $m = 0.4$ & $c = 2.4$, lets predict values for y for $x = \{1,2,3,4,5\}$

$y = 0.4 \times 1 + 2.4 = 2.8$

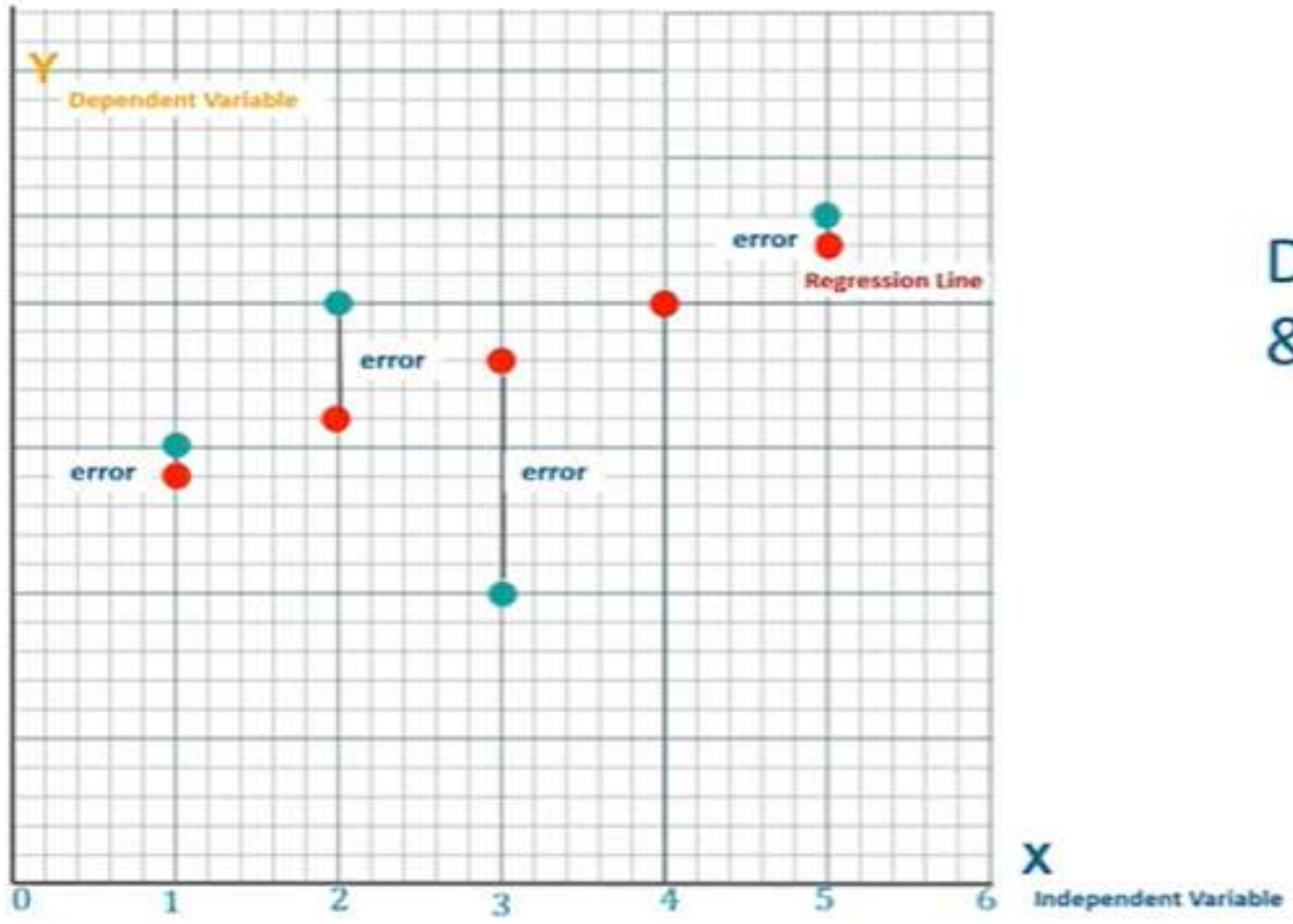
$y = 0.4 \times 2 + 2.4 = 3.2$

$y = 0.4 \times 3 + 2.4 = 3.6$

$y = 0.4 \times 4 + 2.4 = 4.0$

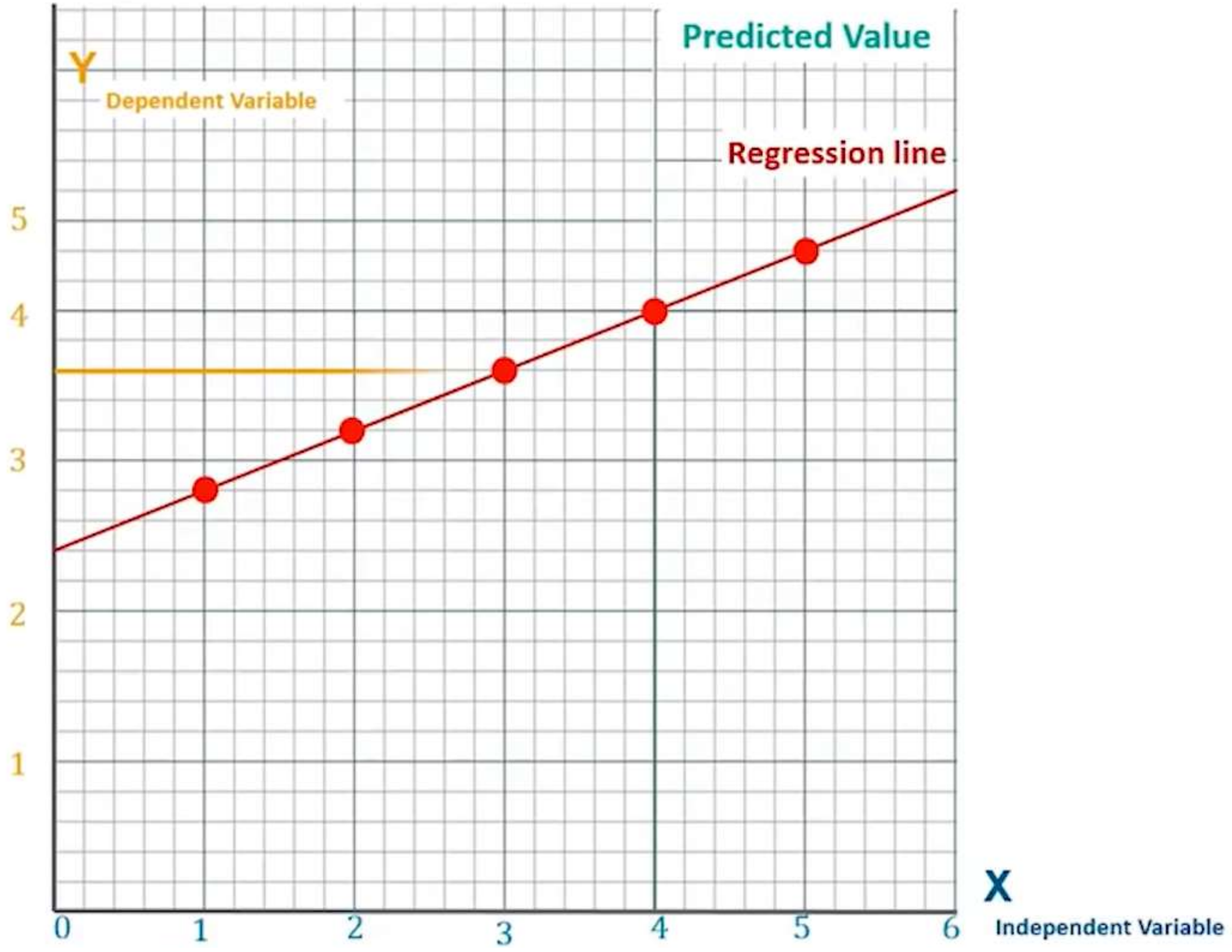
$y = 0.4 \times 5 + 2.4 = 4.4$

Simple Linear Regression – Problem 1



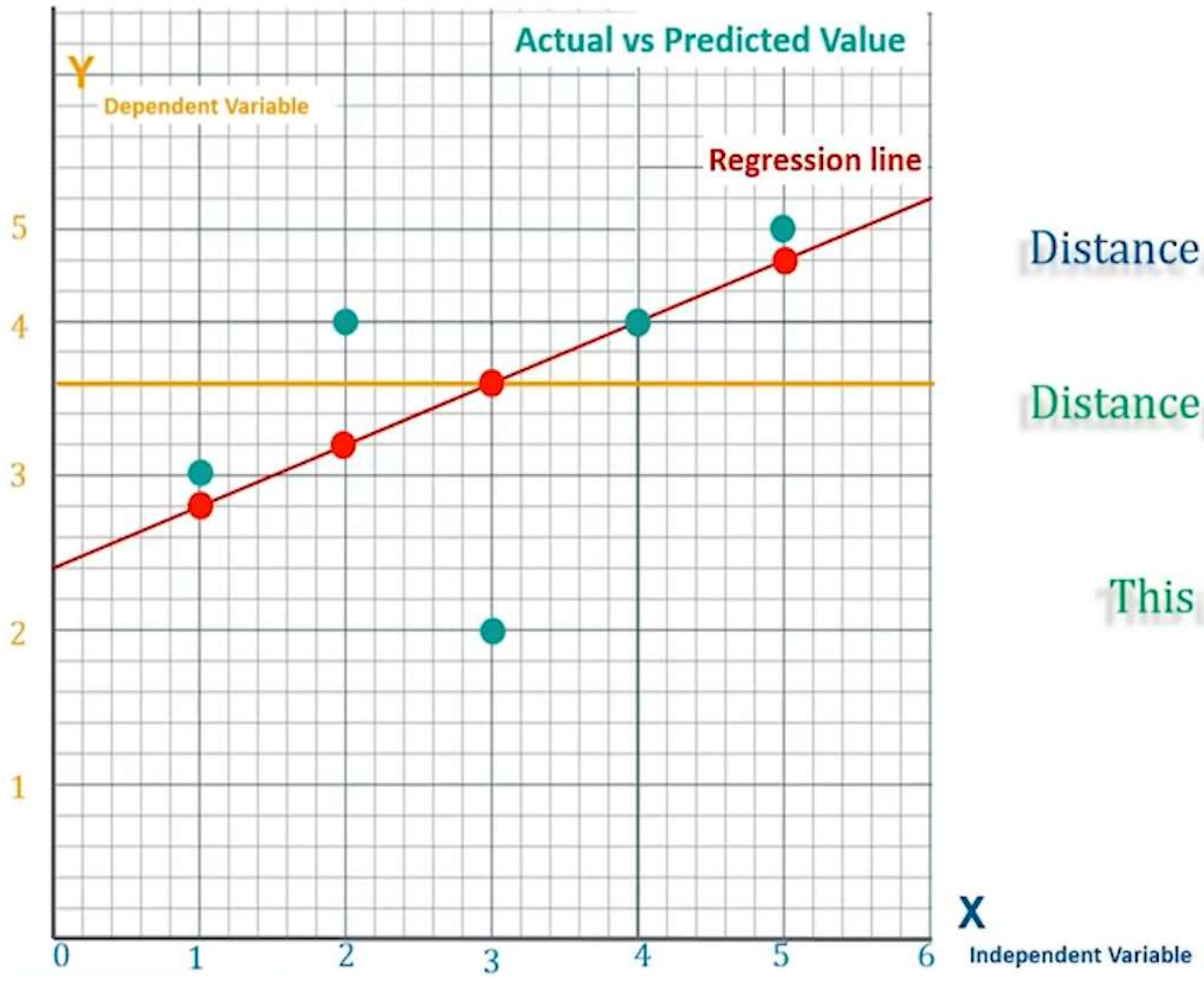
Distance between actual & predicted value

Calculation of R^2



x	y_p
1	2.8
2	3.2
3	3.6
4	4.0
5	4.4

Calculation of R^2



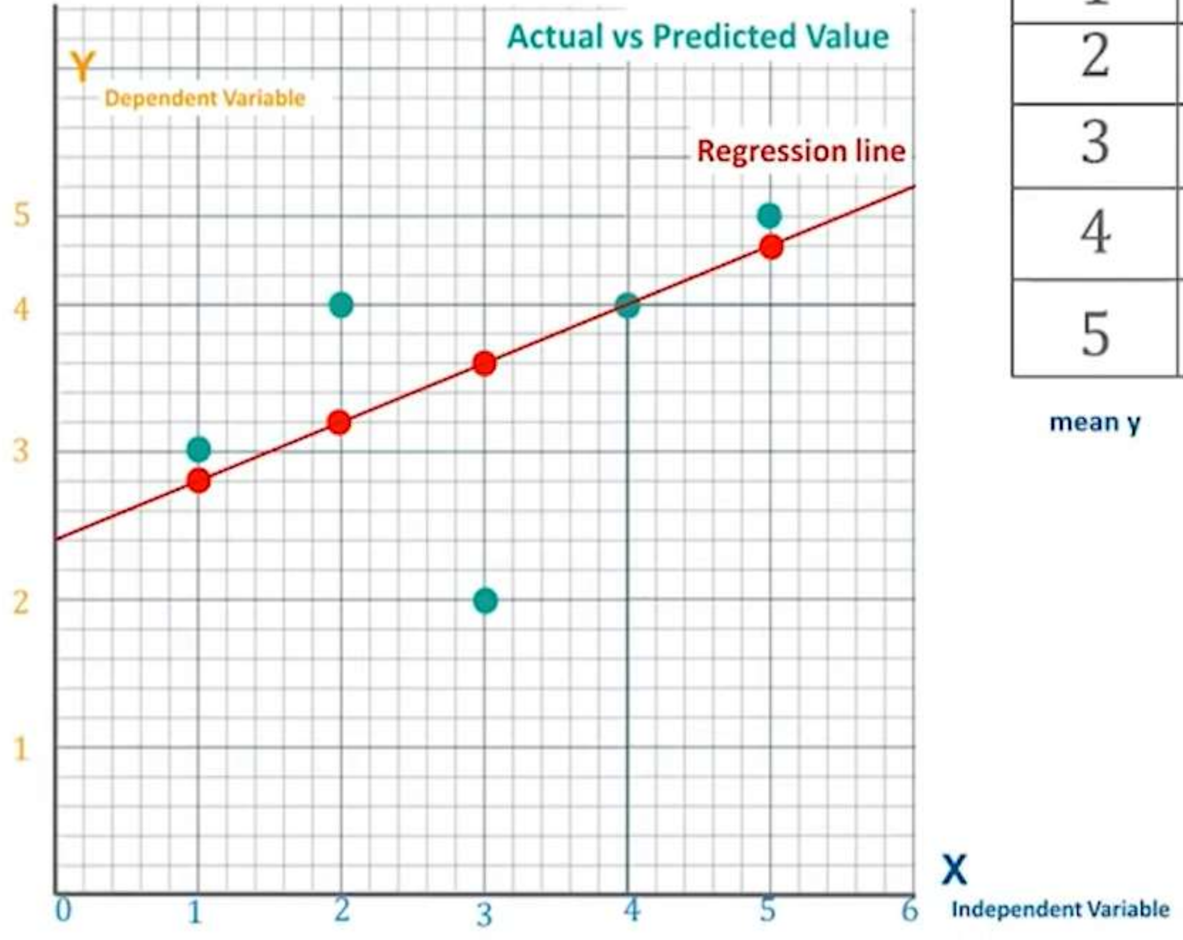
Distance actual - mean

vs

Distance predicted - mean

This is nothing but $R^2 = \dfrac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$

Calculation of R^2

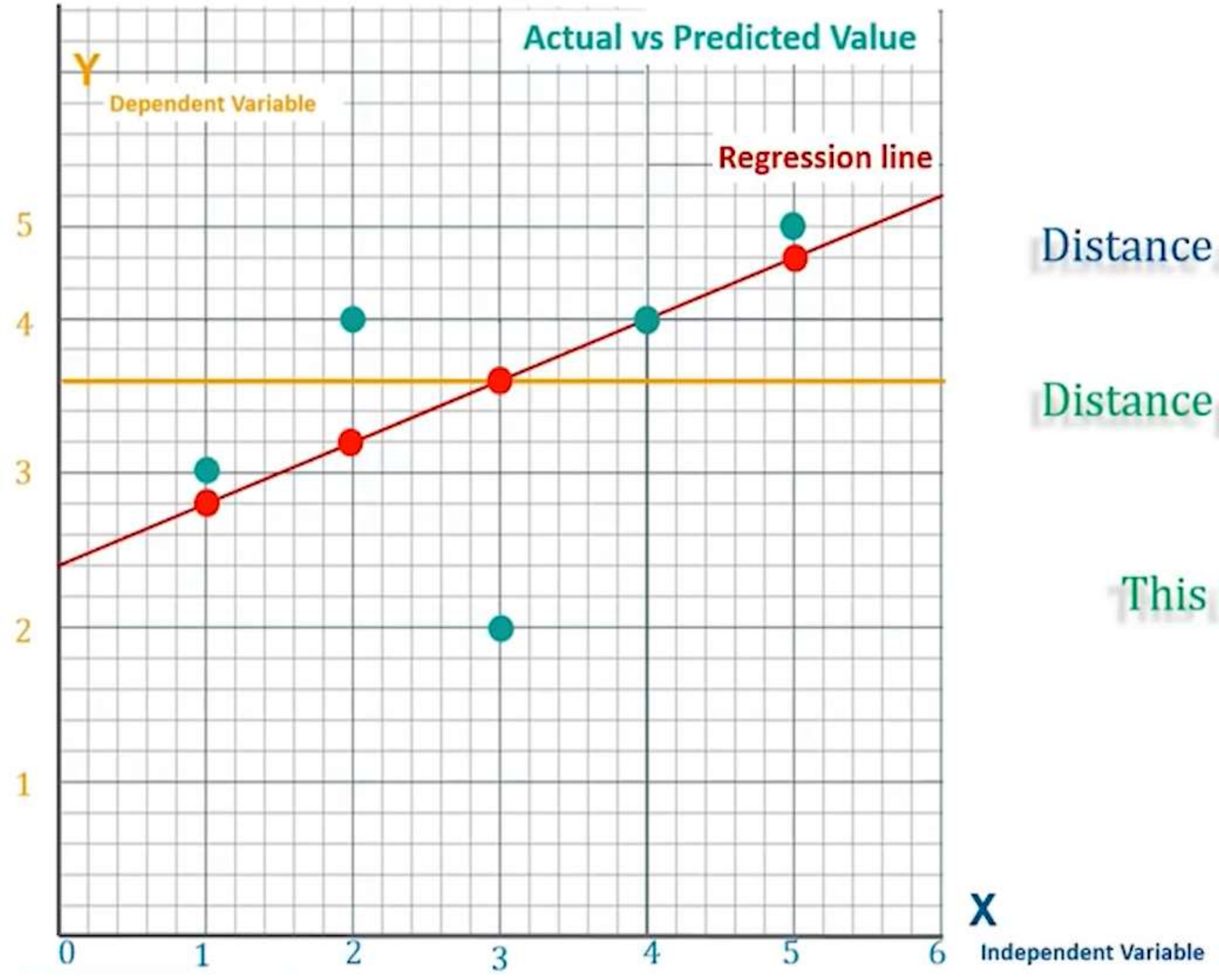


x	y	$y-\bar{y}$	$(y-\bar{y})^2$	y_p	$(y_p - \bar{y})$
1	3	– 0.6	0.36	2.8	-0.8
2	4	0.4	0.16	3.2	-0.4
3	2	–1.6	2.56	3.6	0
4	4	0.4	0.16	4.0	0.4
5	5	1.4	1.96	4.4	0.8

mean y 3.6

$$R^2 = \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$$

Calculation of R^2



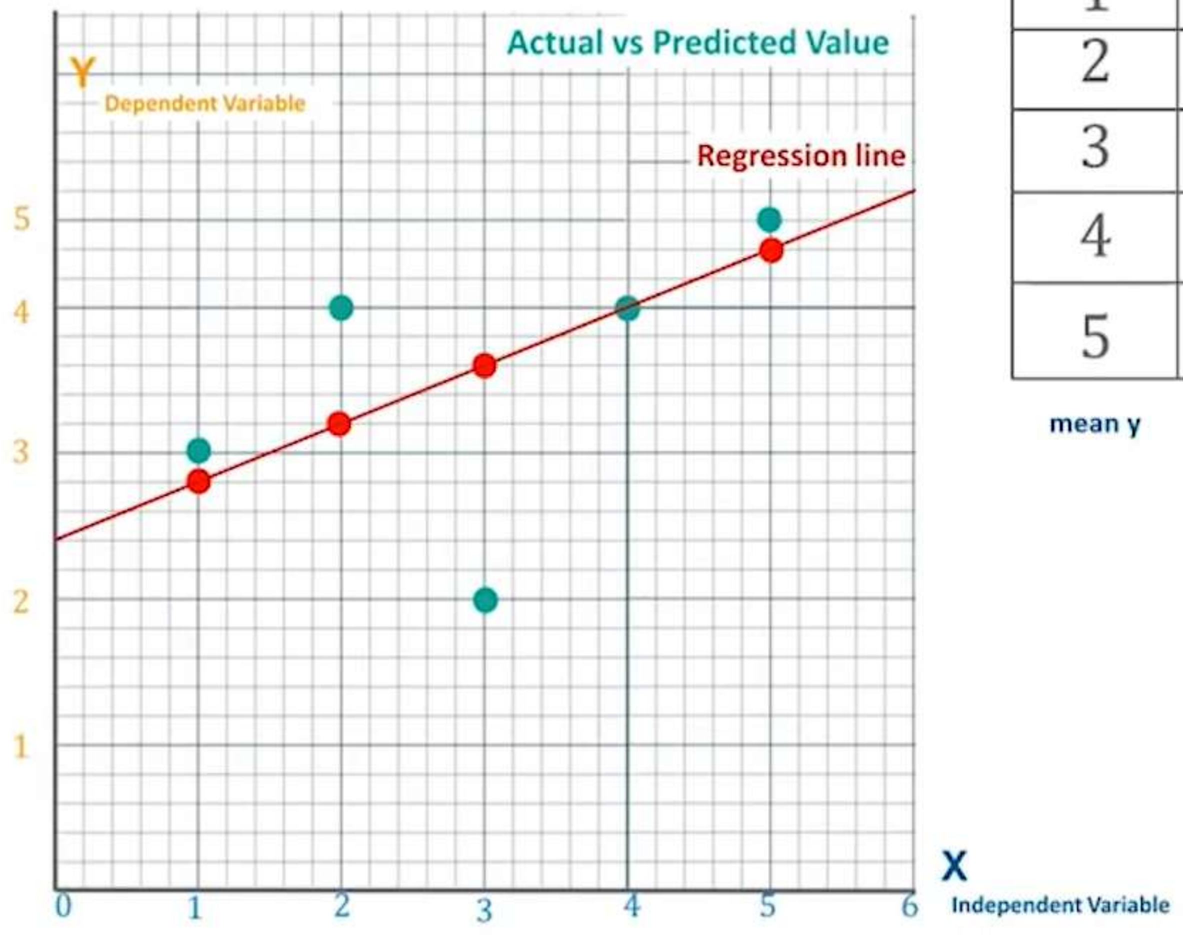
Distance actual - mean

vs

Distance predicted - mean

This is nothing but $R^2 = \dfrac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$

Calculation of R^2

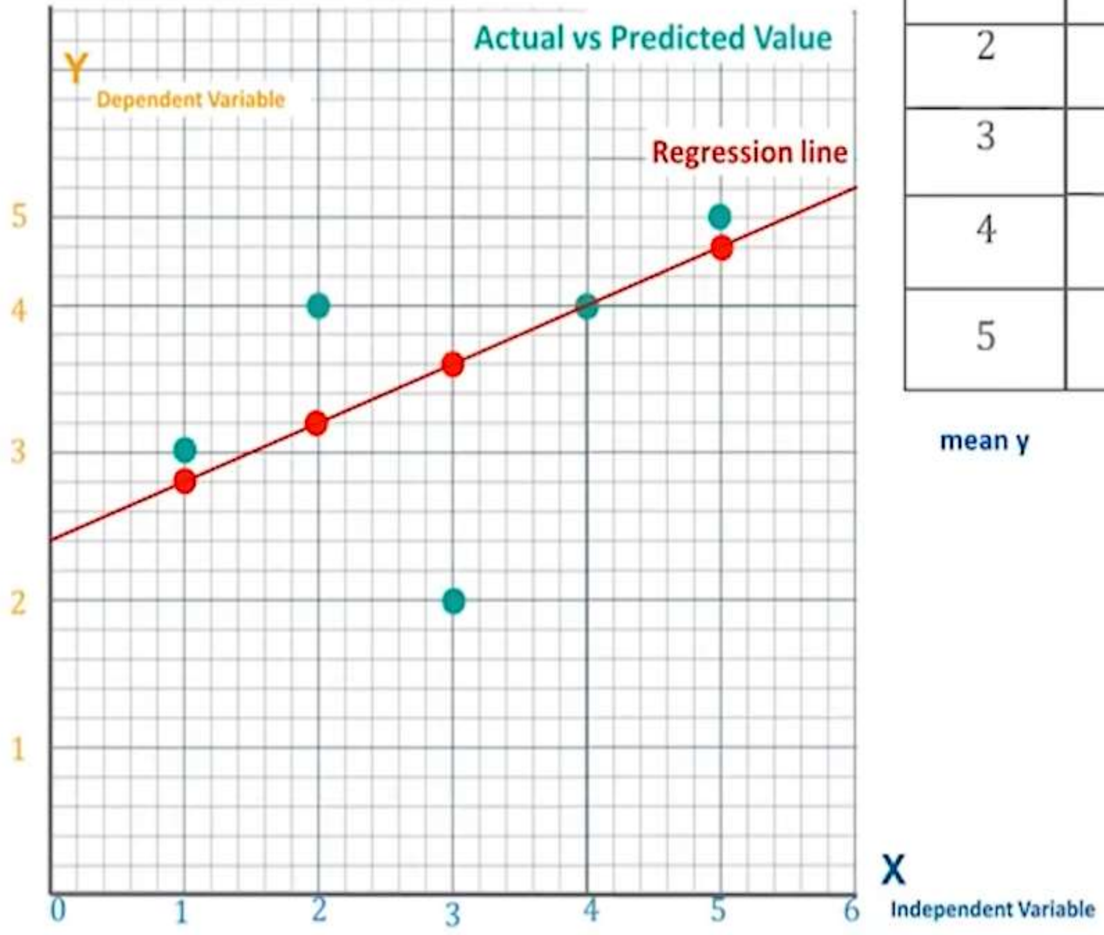


x	y	$y-\bar{y}$	$(y-\bar{y})^2$	y_p	$(y_p - \bar{y})$
1	3	– 0.6	0.36	2.8	-0.8
2	4	0.4	0.16	3.2	-0.4
3	2	–1.6	2.56	3.6	0
4	4	0.4	0.16	4.0	0.4
5	5	1.4	1.96	4.4	0.8

mean y 3.6

$$R^2 = \frac{\sum (y_p - \bar{y})^2}{\sum (y-\bar{y})^2}$$

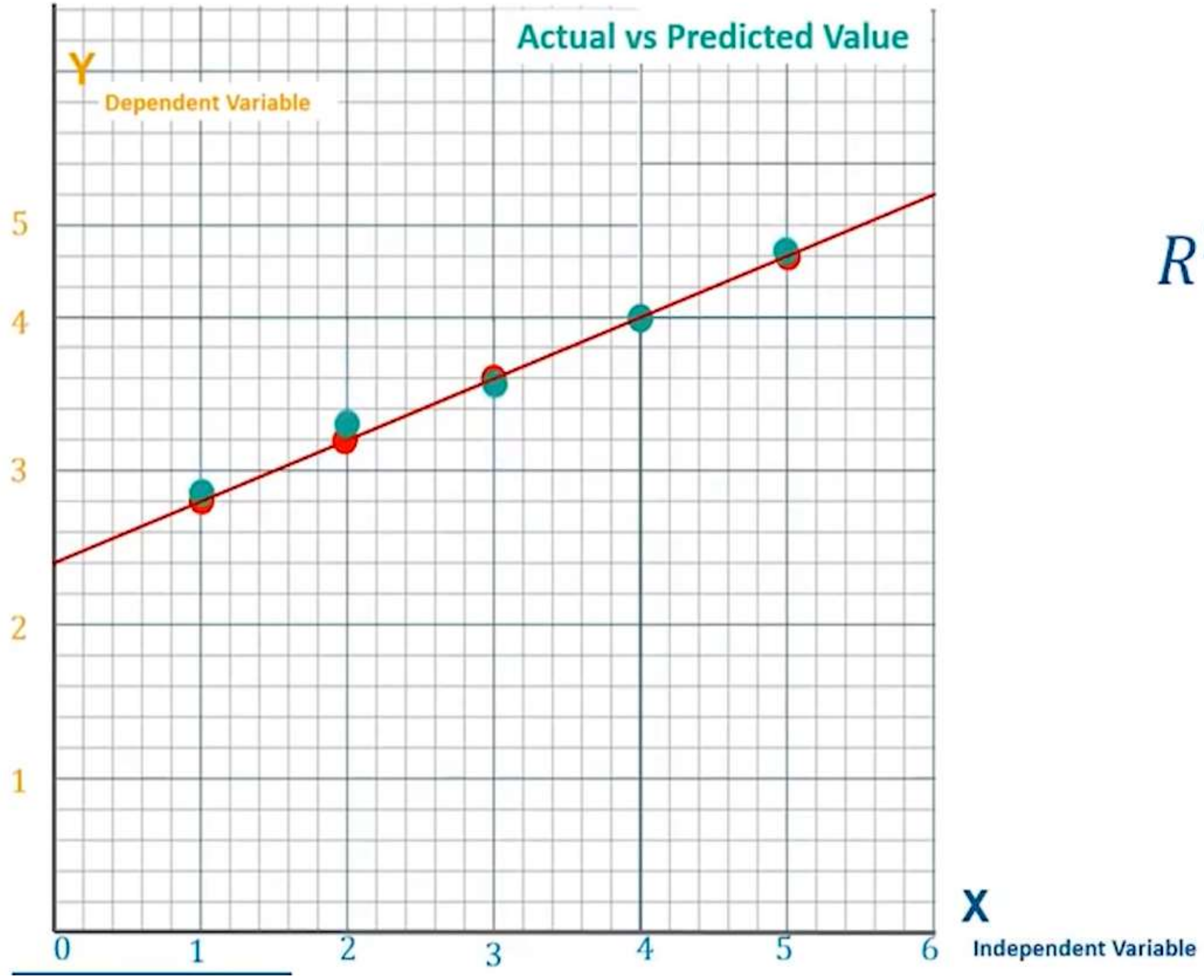
Calculation of R^2



x	y	$y - \bar{y}$	$(y - \bar{y})^2$	y_p	$(y_p - \bar{y})$	$(y_p - \bar{y})^2$
1	3	– 0.6	0.36	2.8	-0.8	0.64
2	4	0.4	0.16	3.2	-0.4	0.16
3	2	–1.6	2.56	3.6	0	0
4	4	0.4	0.16	4.0	0.4	0.16
5	5	1.4	1.96	4.4	0.8	0.64
mean y	3.6		Σ 5.2			Σ 1.6

$$R^2 = \frac{1.6}{5.2} = \frac{\Sigma\left((y_p - \bar{y}\right)^2}{\Sigma\,(y - \bar{y})^2}$$

Calculation of R^2



$$R^2 \approx 0.9$$

Simple Linear Regression – Problem 2

Example:- Straight-line regression using the method of least squares. Table 6.7 shows a set of paired data where *x* is the number of years of work experience of a college graduate and *y* is the corresponding salary of the graduate.

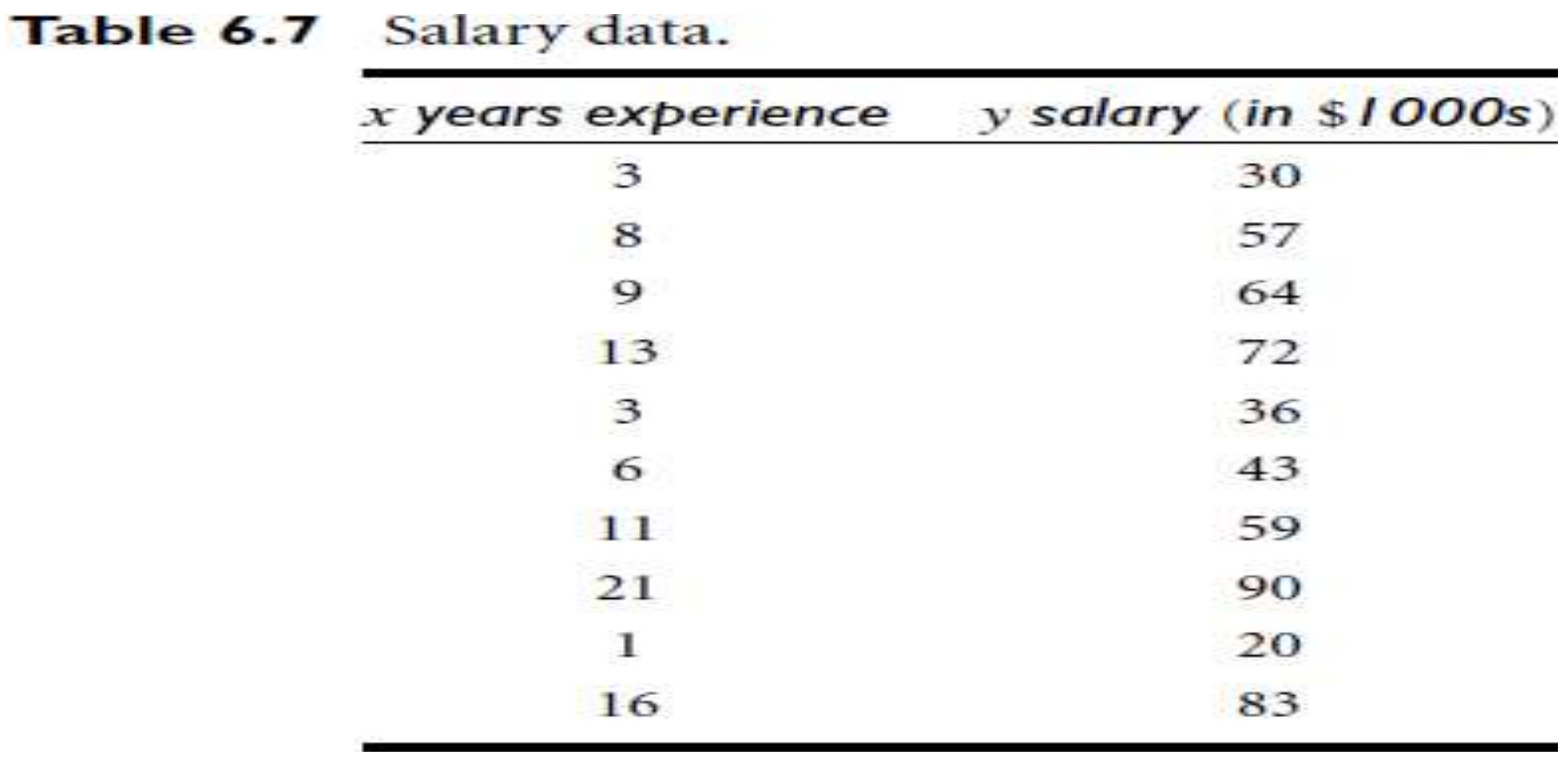


Table 6.7 Salary data.

x years experience	*y* salary (in $1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Linear Regression – Problem 2

- The 2-D data can be graphed on a *scatter plot*, as in Figure 6.26. The plot suggests a linear relationship between the two variables, *x* and *y*.
- We model the relationship that salary may be related to the number of years of work experience with the equation $y = w_0 + w_1 x$.

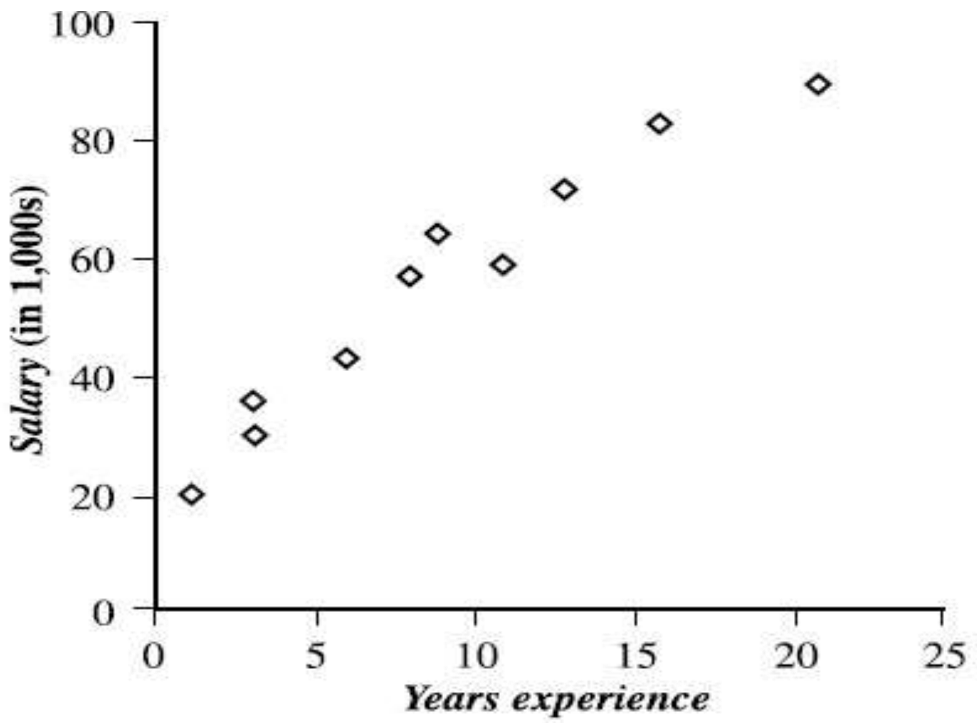


Figure 6.26 Plot of the data in Table 6.7 for Example 6.11. Although the points do not fall on a straight line, the overall pattern suggests a linear relationship between *x* (*years experience*) and *y* (*salary*).

Linear Regression – Problem 2

- Given the above data, we compute mean(x)= 9.1 and mean(y)= 55.4.

  Substituting these values into Equations (1) and (2), we get

$$w_1 = \frac{(3-9.1)(30-55.4)+(8-9.1)(57-55.4)+\cdots+(16-9.1)(83-55.4)}{(3-9.1)^2+(8-9.1)^2+\cdots+(16-9.1)^2} = 3.5$$

$$w_0 = 55.4-(3.5)(9.1) = 23.6$$

- Thus, the equation of the least squares line is estimated by y = 23.6+3.5x. Using this equation, we can predict that the salary of a college graduate with, say, 10 years of experience is $58,600.

Multiple Linear Regression

- Multiple regression, also known as multiple linear regression (MLR), is a statistical technique that uses two or more explanatory variables to predict the outcome of a response variable.
- In other words, it can explain the relationship between multiple independent variables against one dependent variable.
- The formula for multiple linear regression, which produces a more specific calculation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

y is the predicted or expected value of the dependent variable.

X1, X2, Xp are three independent or predictor variables

β_0 is the value of y when all the independent variables are equal to zero.

β_1 , $\beta_{2,}$ β_p are the estimated regression coefficients.

- Each regression coefficient represents the change in y relative to a one-unit change in the respective independent variable.

Multiple Linear Regression

- Obtain the input dataset and identify independent and dependent features.
- Fit the regression line of the form:

$$Y = B_0 + B_1 X_1 + B_2 X_2 + \dots + B_p X_p$$

- **Calculate** $X_1^2, X_2^2, X_1 y, X_2 y, X_1 X_2$
- **Calculate Regression sums using** $\sum X_1^2, \sum X_2^2, \sum X_1 Y, \sum X_2 Y, \sum X_1 X_2$

$$R1 = \sum x_1^2 = \sum X_1^2 - \frac{(\sum X_1)^2}{n}$$

$$R3 = \sum x_1 y = \sum X_1 Y - \frac{\sum X_1 \sum Y}{n}$$

$$R2 = \sum x_2^2 = \sum X_2^2 - \frac{(\sum X_2)^2}{n}$$

$$R4 = \sum x_2 y = \sum X_2 Y - \frac{\sum X_2 \sum Y}{n}$$

$$R5 = \sum x_1 x_2 = \sum X_1 X_2 - \frac{\sum X_1 \sum X_2}{n}$$

Multiple Linear Regression

- Compute B_0, B_1, B_2

$$B_0 = \overline{y} - B_1\overline{X}_1 - B_2\overline{X}_2$$

$$B_1 = \frac{R_2R_3 - R_5R_4}{R_1R_2 - R_5{}^2}$$

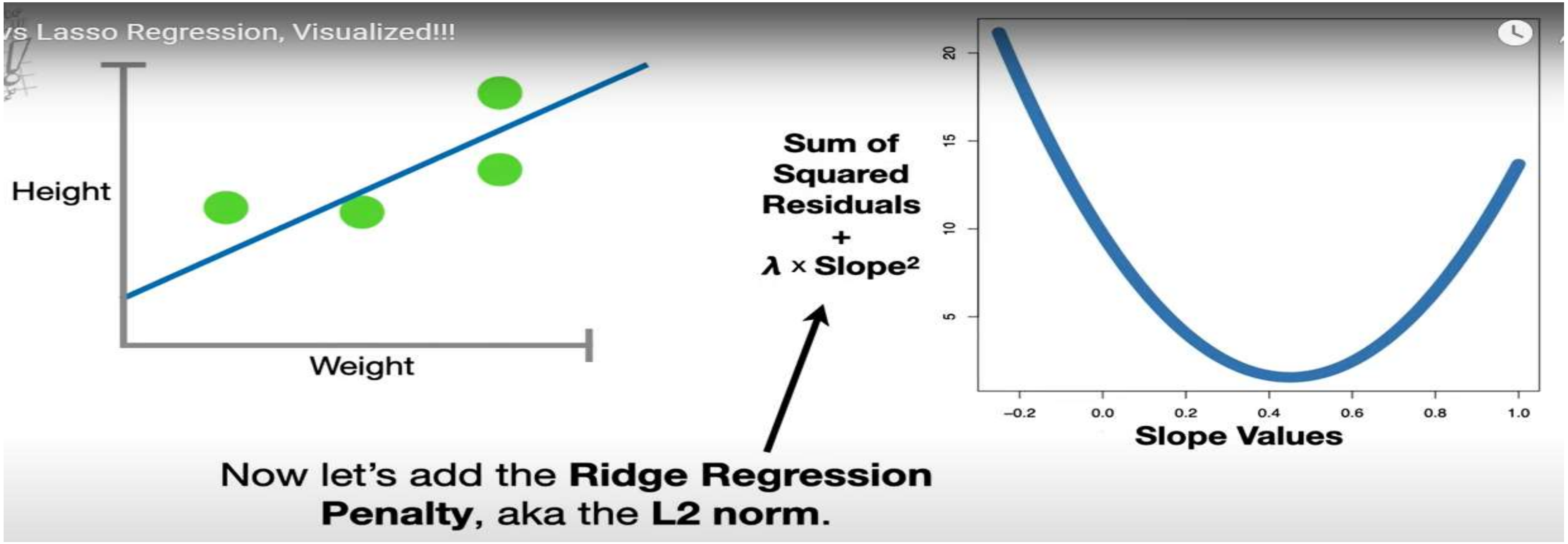
$$B_2 = \frac{R_1R_4 - R_5R_3}{R_1R_2 - R_5{}^2}$$

- Substitute B0, B1, B2 values and compute $\hat{y}$
- Calculate the adjusted R2 to find whether the line obtained is best fit or not

URL :https://www.statology.org/multiple-linear-regression-by-hand/

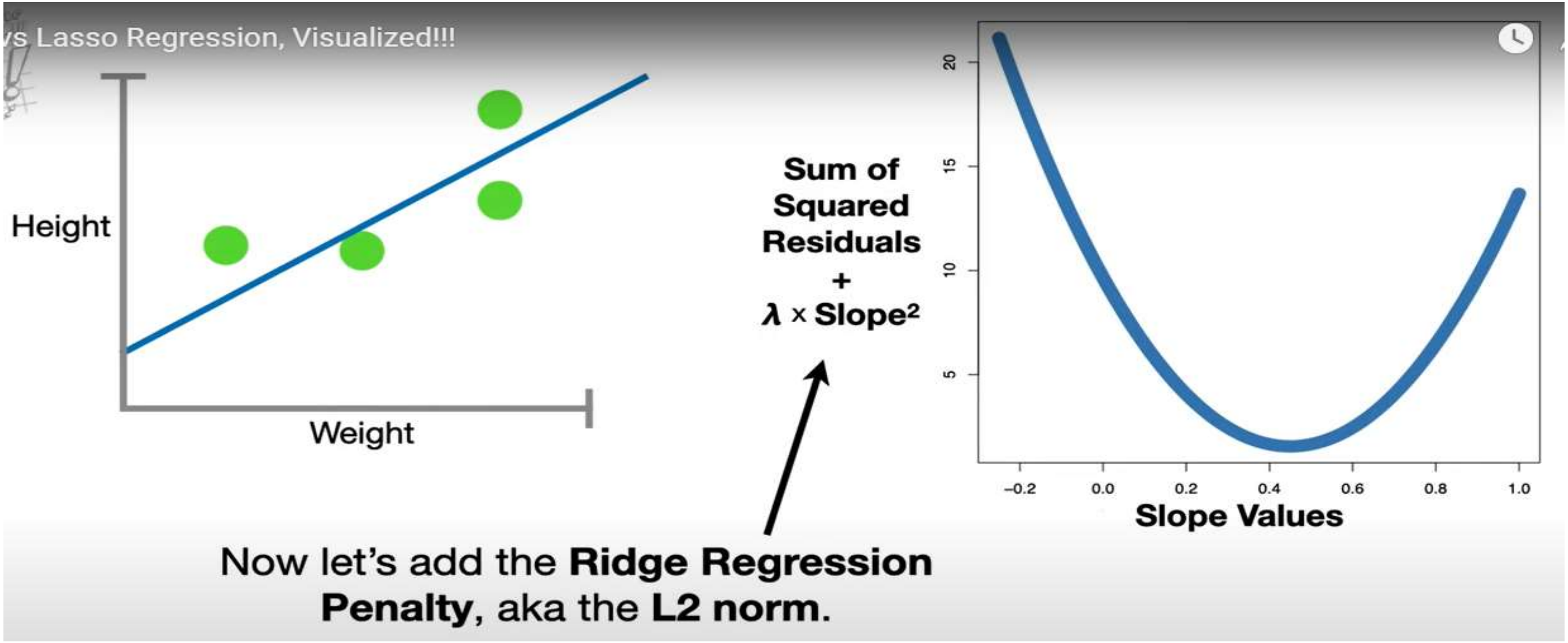
Regression – plotting Residuals

- To find the best fit line, change the slope and find the sum of squares residual error.
- Plot these residuals on a graph. At some point the residual error hits minimum i.e the slope is used to find the best fit line.



Regularization

Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.



s Lasso Regression, Visualized!!!
Height
Weight
Sum of Squared Residuals + λ × Slope²
20
15
10
5
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
Slope Values

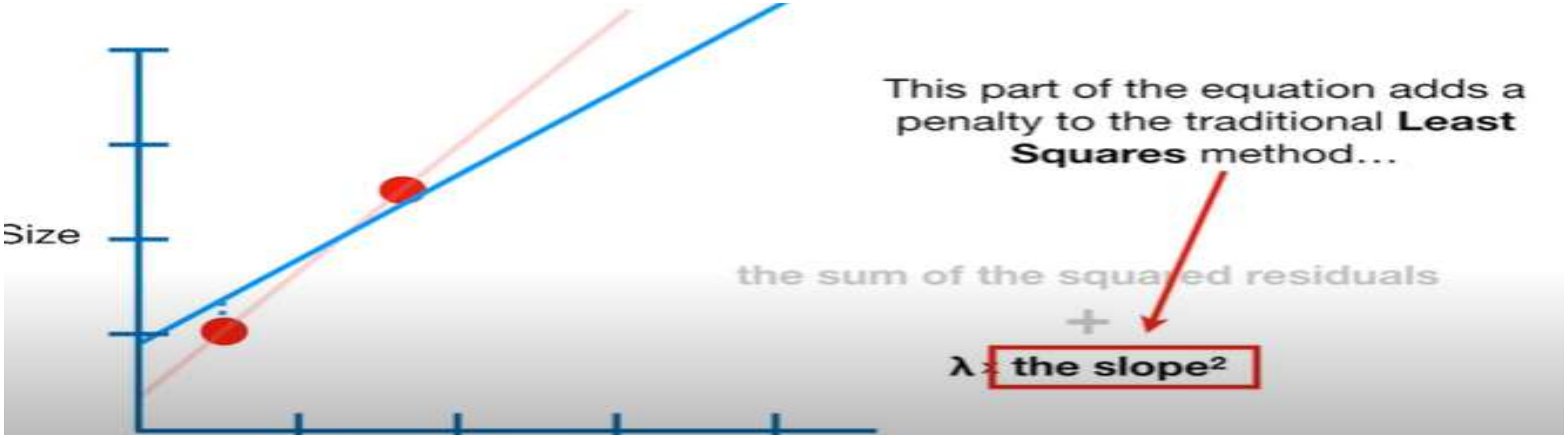
Now let's add the **Ridge Regression Penalty**, aka the **L2 norm.**

Regularization

- When it comes to training models, there are two major problems one can encounter: overfitting and underfitting.
- Overfitting happens when the model performs well on the training set but not so well on unseen (test) data.
- Underfitting happens when it neither performs well on the train set nor on the test set.
- Regularization is implemented to avoid overfitting of the data, especially when there is a large variance between train and test set performances.
- There are different ways of reducing model complexity and preventing overfitting in linear models. This includes ridge and lasso regression models.

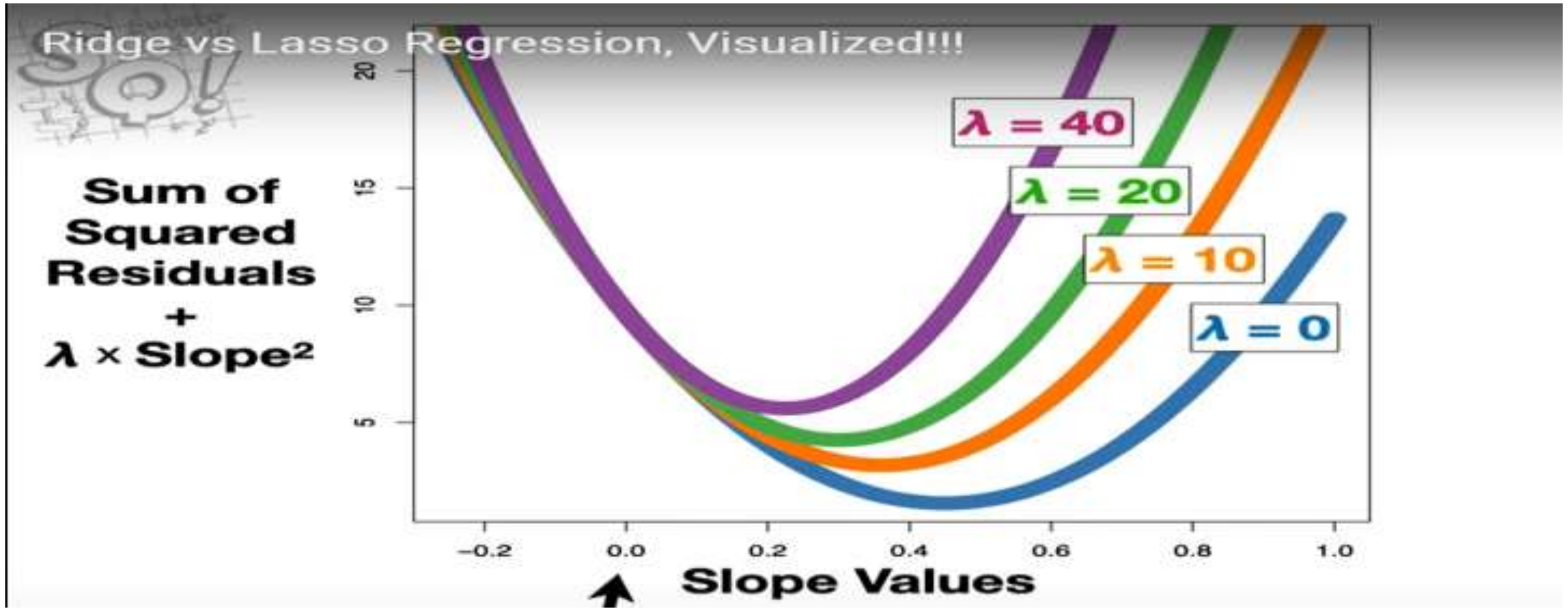
Ridge Regression (L2 norm)

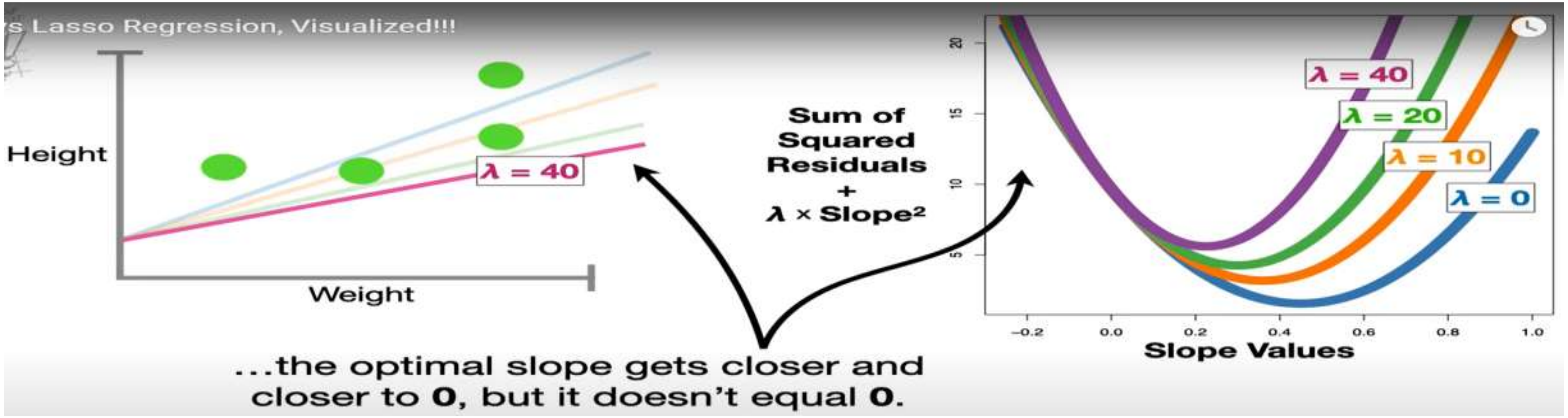
- Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated.
- Ridge Regression solves the problem of overfitting , as just regular squared error regression fails to recognize the less important features and uses all of them, leading to overfitting. Ridge regression adds a slight bias, to fit the model according to the true values of the data.
  - MSE= sum of the squared error + λ*slope^2



Size
This part of the equation adds a penalty to the traditional **Least Squares** method...
the sum of the squared residuals
+
λ × the slope²

Ridge Regression – L2 norm





...the optimal slope gets closer and closer to **0**, but it doesn't equal **0**.

Lasso Regression (L1 Norm)

- **This is a regularization technique used in feature selection using a Shrinkage method also referred to as the penalized regression method. Lasso is short for Least Absolute Shrinkage and Selection Operator, which is used both for regularization and model selection.**
- **If a model uses the L1 regularization technique, then it is called lasso regression.**

MSE = sum of the squared error + λ*slope

https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

Multicollinearity in Regression

Multicollinearity in regression occurs when independent variables are highly correlated with each other, making it difficult to isolate the individual impact of each predictor on the dependent variable. This can lead to unstable and unreliable regression results.

What it means:

- High Correlation:

Multicollinearity implies a strong linear relationship between two or more independent variables.

- Difficulty in Interpretation:

It becomes challenging to determine the unique effect of each correlated predictor on the outcome variable.

- Unstable Coefficients:

The estimated regression coefficients can be highly sensitive to small changes in the data or model specification.

Multicollinearity in Regression

Consequences:

Inflated Standard Errors: Multicollinearity leads to larger standard errors for the regression coefficients, making it harder to find statistically significant relationships.

Reduced Statistical Significance: Variables that are truly important might appear insignificant due to inflated standard errors.

Unreliable Coefficients: The estimated coefficients may be unstable and unreliable, making it difficult to draw meaningful conclusions about the relationships between predictors and the outcome.

Difficulty in Model Interpretation: It becomes harder to understand the individual contributions of each predictor to the model's predictions.

How to Detect:

Correlation Matrix: Calculate the correlation coefficients between all pairs of independent variables. High correlations (e.g., above 0.8 or 0.9) suggest potential multicollinearity.

Variance Inflation Factor (VIF): Calculate the VIF for each predictor. A VIF value above 5 or 10 indicates a multicollinearity problem.

Eigenvalues and Condition Indices: These can be used to assess the severity of multicollinearity, especially when dealing with multiple correlated variables.

Multicollinearity in Regression

How to Address:

Remove Highly Correlated Variables:

Identify and remove one or more of the highly correlated variables from the model.

Combine Variables:

Create a new variable by combining the highly correlated ones (e.g., by averaging or summing them).

Regularization Techniques:

Techniques like Ridge or Lasso regression can help to shrink the coefficients of less important variables and mitigate the effects of multicollinearity.

Partial Least Squares Regression:

This technique transforms the original variables into a new set of uncorrelated variables, which can then be used in the regression model.

Collect More Data:

In some cases, collecting more data can help to reduce the impact of multicollinearity.

Use Domain Knowledge:

Incorporate prior knowledge about the relationships between variables to guide model selection and variable removal.

Multicollinearity in Regression - VIF

VIF (Variance Inflation Factor) quantifies **how much the variance of a regression coefficient is inflated** due to multicollinearity with other predictors.

For each predictor x_j, the VIF is defined as:

$$\text{VIF}_j = \frac{1}{1 - R_j^2}$$

Where:

- R_j^2 is the coefficient of determination when x_j is regressed on **all other predictors**.

Assumptions of Linear Regression

Regression analysis, particularly linear regression, relies on several key assumptions for accurate and reliable results. These assumptions ensure that the model's coefficients are unbiased, efficient, and that inferences drawn from the model are valid. Mainly, these include linearity, independence of errors, homoscedasticity, normality of errors, and no multicollinearity.

Here's a breakdown of the key assumptions:

1. Linearity:

The relationship between the dependent variable and the independent variable(s) should be linear. This means that the change in the dependent variable is constant for each unit change in the independent variable.

Checking: Scatter plots can be used to visually examine the relationship between variables, and residual plots can help identify any non-linear patterns.

2. Independence of Errors:

The errors (residuals) should be independent of each other. This means that the error for one observation should not be related to the error for any other observation.

Checking: Autocorrelation plots can help detect patterns in the residuals that indicate dependence.

Assumptions of Linear Regression

3. Homoscedasticity:

The variance of the errors (residuals) should be constant across all levels of the independent variables. This means that the spread of the errors should be consistent regardless of the value of the independent variable.

Checking: Residual plots can be used to identify any non-constant variance in the residuals.

4. Normality of Errors:

The errors (residuals) should be normally distributed. This assumption is often used for hypothesis testing and constructing confidence intervals.

Checking: Histograms and Q-Q plots can be used to examine the distribution of the residuals.

5. No Multicollinearity:

The independent variables should not be highly correlated with each other. If there is high multicollinearity, it can make it difficult to interpret the individual effects of independent variables.

Checking: Variance Inflation Factor (VIF) can be used to assess the extent of multicollinearity.

Assumptions of Linear Regression

6. No Endogeneity:

The independent variables should not be correlated with the error term. If they are, the regression coefficients will be biased.

Checking: This is a complex assumption, and statistical tests and econometric techniques are often used to assess endogeneity.

7. Representative Sample:

The sample data should be representative of the population to which the regression model will be applied.

Checking: This is more of a general consideration for any statistical analysis, and it involves careful sample selection and ensuring that the sample accurately reflects the target population.

Violations of these assumptions can lead to biased and unreliable regression models, potentially resulting in inaccurate predictions and invalid inferences. It's crucial to assess these assumptions and consider appropriate methods for addressing any violations.

Linear Regression – Cost Function

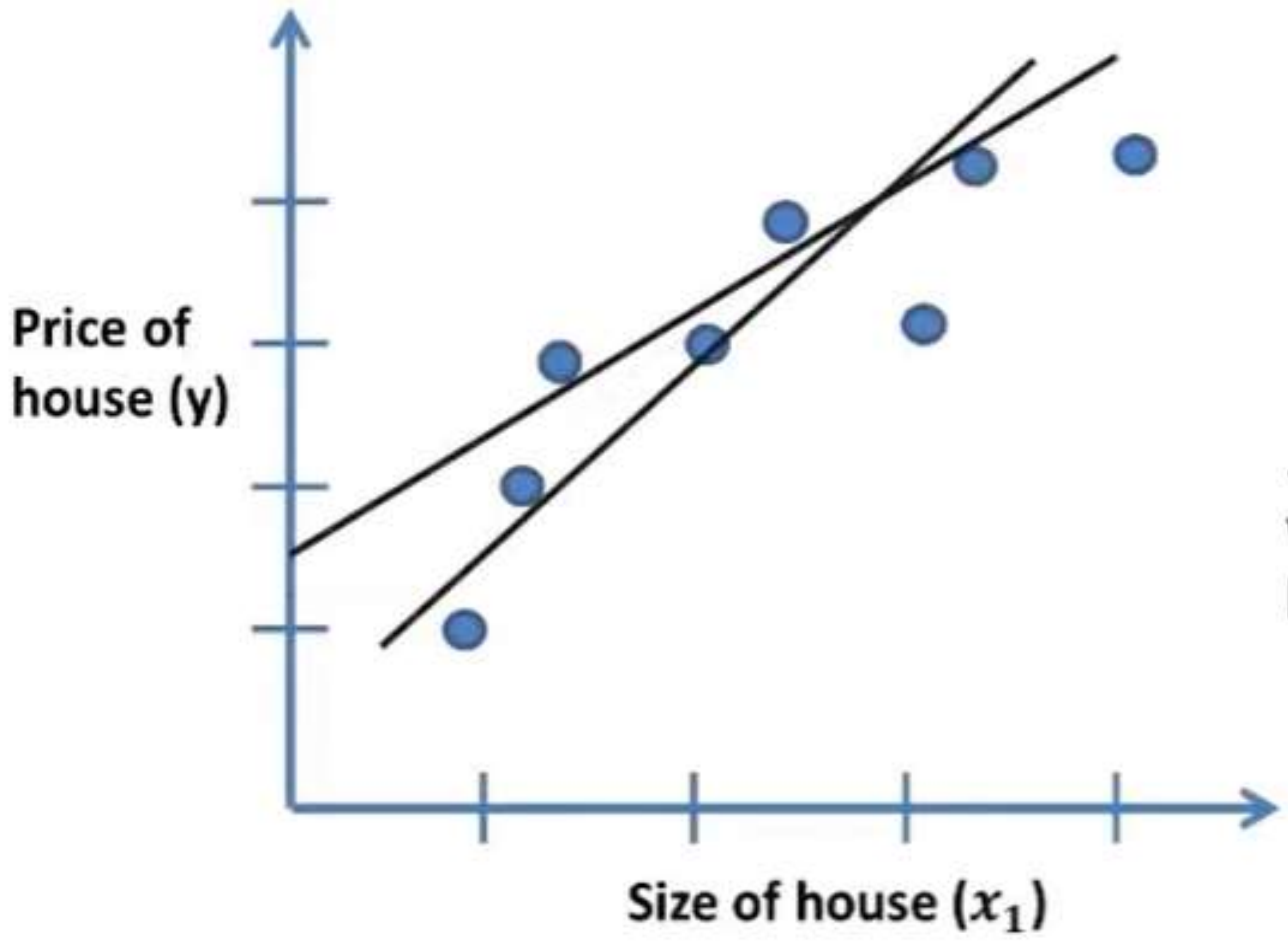
- The output which is obtained or predicted by an algorithm is referred to as yˆ (pronounced as y hat). The difference between the actual and predicted values is the error, i.e., y - y^. Different values of y- y^ (loss function) are obtained when the model repeatedly tries to find the best relation. The average summation of all loss function values is called the cost function. The machine learning algorithm tries to obtain the minimum value of the cost function. In other words, it tries to reach the global minimum.

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

where J = cost function, n= number of observations (i = 1 to n),
∑ = summation, predi = predicted output and yi = actual value.
As shown above, the error difference is squared for each value and then the average of the sum of squares of error gives us the cost function.
It is also referred to as Mean Square Error (MSE).

Cost Function

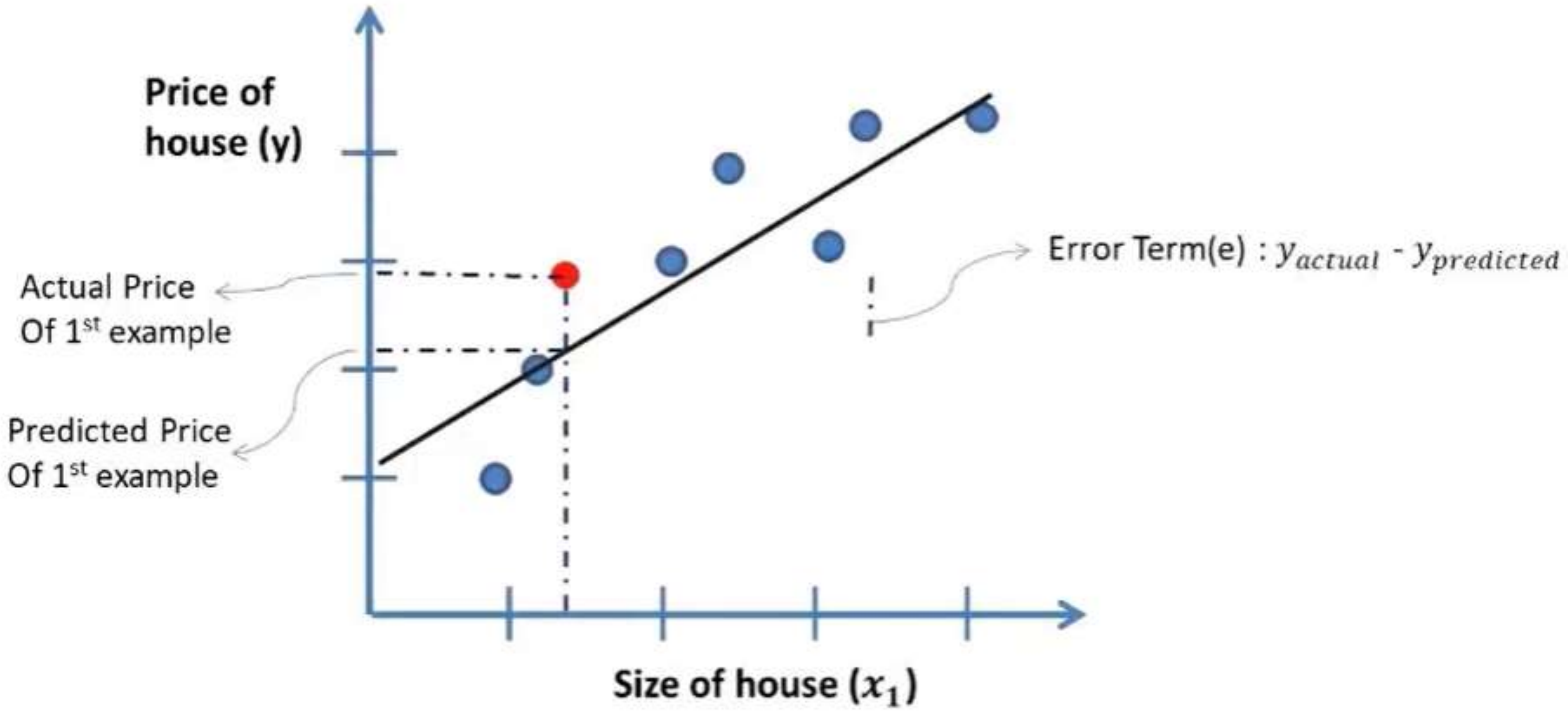


As per our linear regression model,
We need to fit a straight line to it...

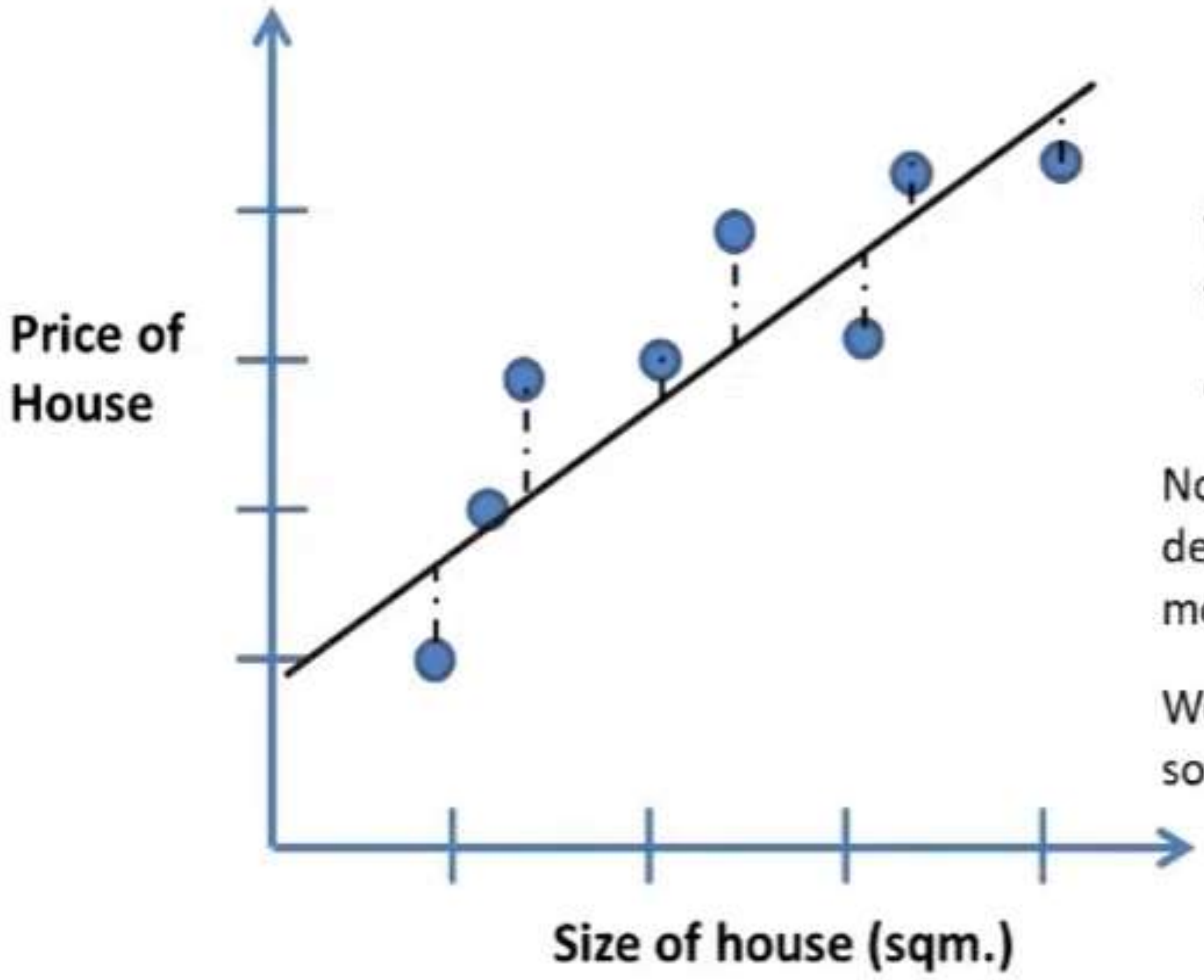
$$y = a_0 + a_1 x_1$$

...depending on values of $\boldsymbol{a_0}$ and $\boldsymbol{a_1}$, we can have many possibilities, which look promising.

Cost Function


Price of house (y)
Actual Price Of 1st example
Predicted Price Of 1st example
Error Term(e) : y_{actual} - $y_{predicted}$
Size of house (x_1)

Cost Function



for i^{th} example, we define error term as

$$e_i = Y_{predicted}^{\;i} \; - Y_{\;actual}^{\;i}$$

Now e_i can be **+ve or –ve**, depending on whether y_{actual} is more or $y_{predicted}$.

We will ***square*** e_i to make it positive so the order doesn't matter.

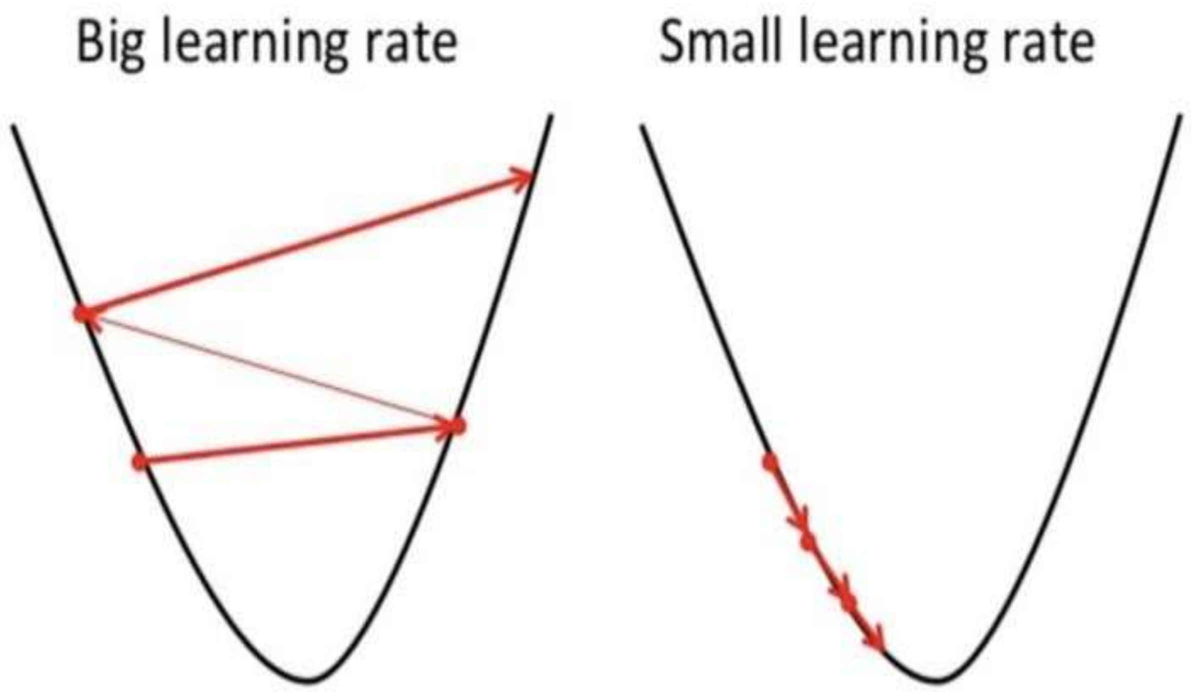
Gradient Descent

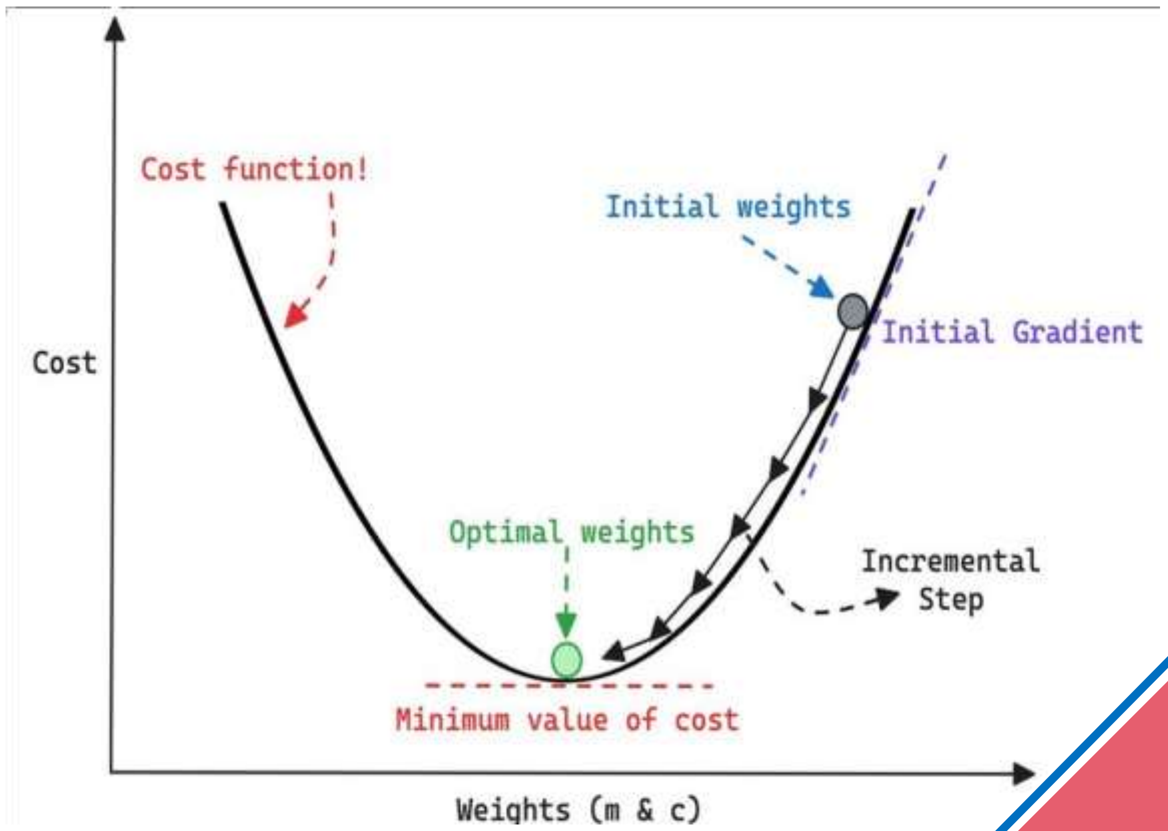
- Gradient descent is an iterative optimization algorithm to find the minimum of a function.
- Gradient Descent is an algorithm that finds the best-fit line for a given training dataset in a smaller number of iterations.
- Gradient descent finds the minimum value by taking steps from an initial guess until it reaches the best value.

n

Gradient Descent

- The basic principle of gradient descent is to choose the step size (also called as learning rate) appropriately so that we can get close to the exact solution.
- Gradient descent stops when step size is very close to zero along with max number of steps we want to perform.





Unit II

Supervised Learning - I

SUPERVISED LEARNING-I UNIT2

Regression models:

- Simple linear regression
- Multiple linear regression
- Cost Function, Gradient Descent
- Performance Metrics: Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-Squared error, Adjusted R Square.

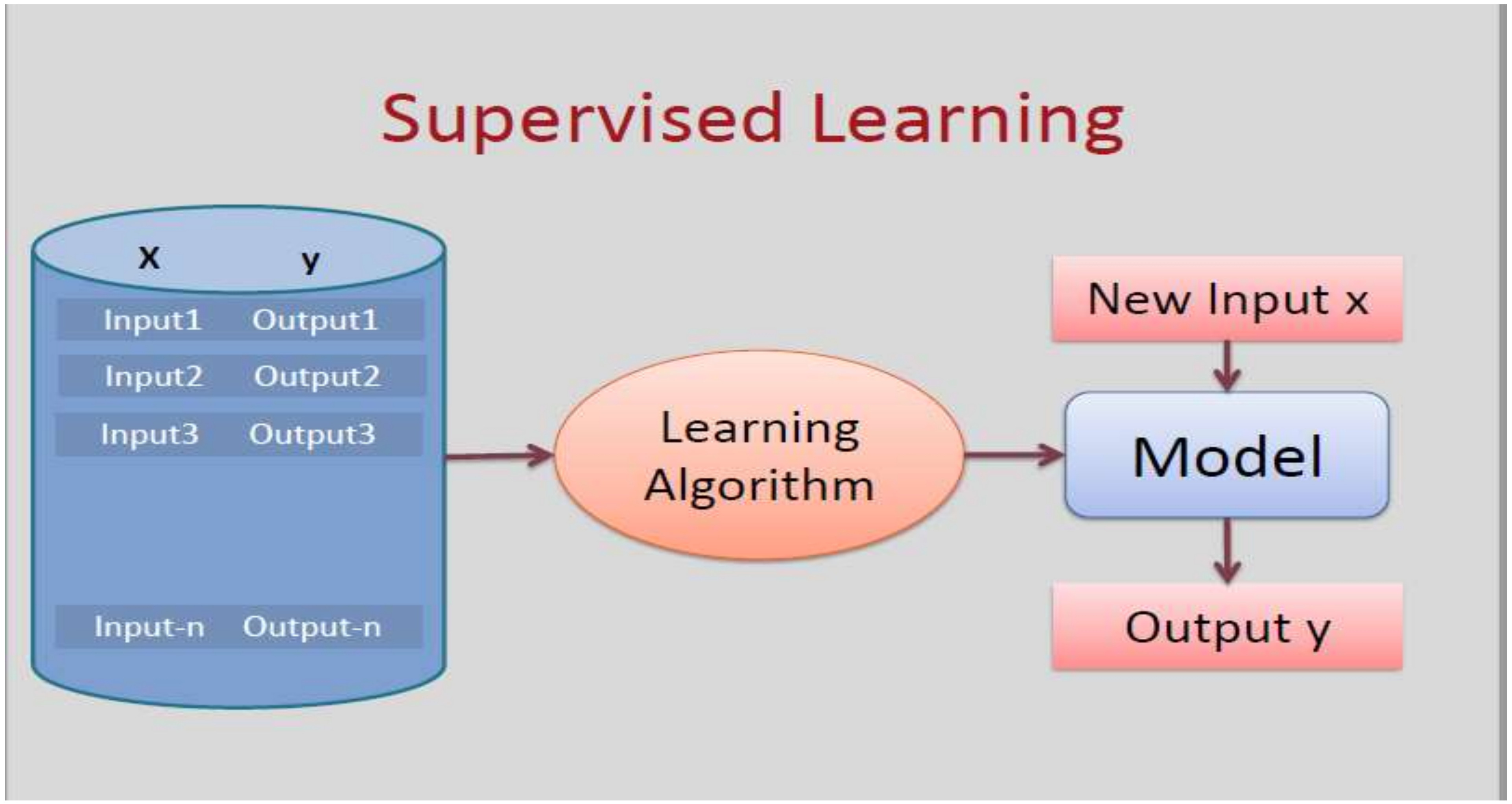
Classification models:

- Decision Trees - ID3, CART
- Naive Bayes
- K-Nearest Neighbours (KNN)
- Logistic regression, Multinomial logistic regression
- Support Vector Machines (SVM).

Supervised Learning

- **Classification and Regression algorithms are Supervised Learning algorithms.**
- **Both the algorithms can be used for forecasting in Machine learning and operate with the labeled datasets.**
- **Regression algorithms are used to forecast Continuous values such as price, income, age etc**
- **Classification algorithms are used to forecast or classify the distinct values such as Real or False, Male or Female, Spam or Not Spam, etc.**
- **Types of Regression Algorithms:**
  - **Simple Linear Regression**
  - **Multiple Linear Regression**
- **Types of Classification Algorithms:**
  - **Decision Trees**
  - **Naïve Bayes**
  - **K Nearest Neighbors**
  - **Logistic Regression , Multinominal Logistic Regression**
  - **SVM**

Supervised Learning Approach



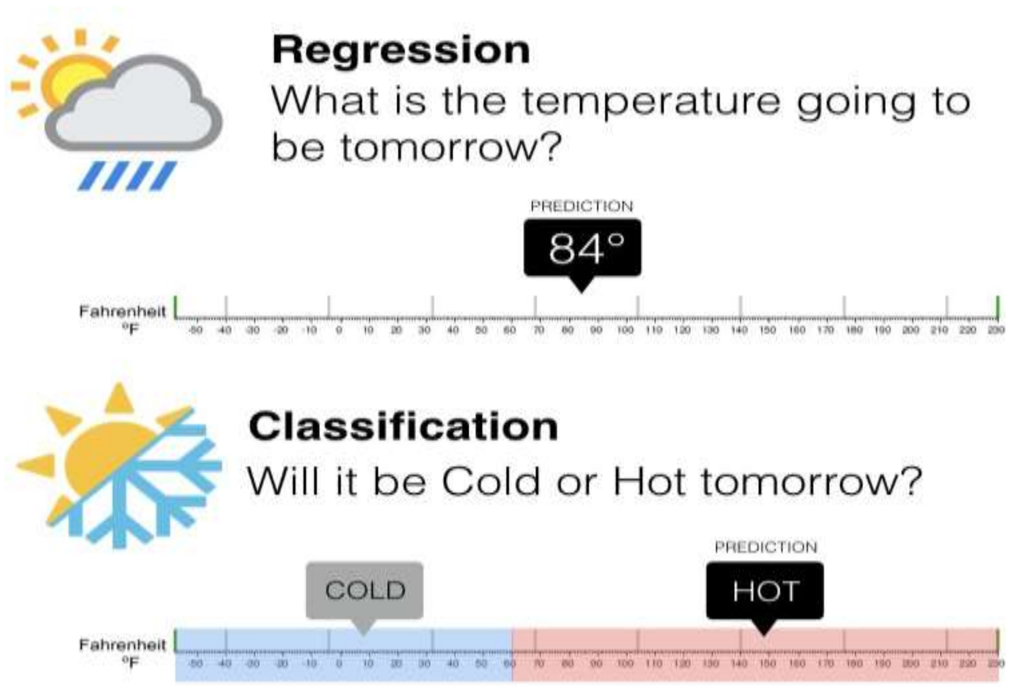
SUPERVISED LEARNING

There are two types of Supervised learning methods

Classification: It is a Supervised Learning task where output is having defined labels (discrete value/ categorical value). Learning function that maps (classifies) a data item into one of several predefined classes.

It can be either binary or multi class classification. In binary classification, model predicts either 0 or 1 ; yes or no but in case of multi class classification, model predicts more than one class.

Regression: It is a Supervised Learning task where output is having continuous value. Learning function which maps a data item to a real valued prediction variables.



Note: An easy way to distinguish between classification and regression tasks is to ask whether there is some kind of continuity in the output.

Classification Models

Supervised Learning - Classification

Classification models:

- Decision Trees - ID3, CART
- Naive Bayes
- K-Nearest Neighbours (KNN)
- Logistic regression, Multinomial logistic regression
- Support Vector Machines (SVM).

Classification

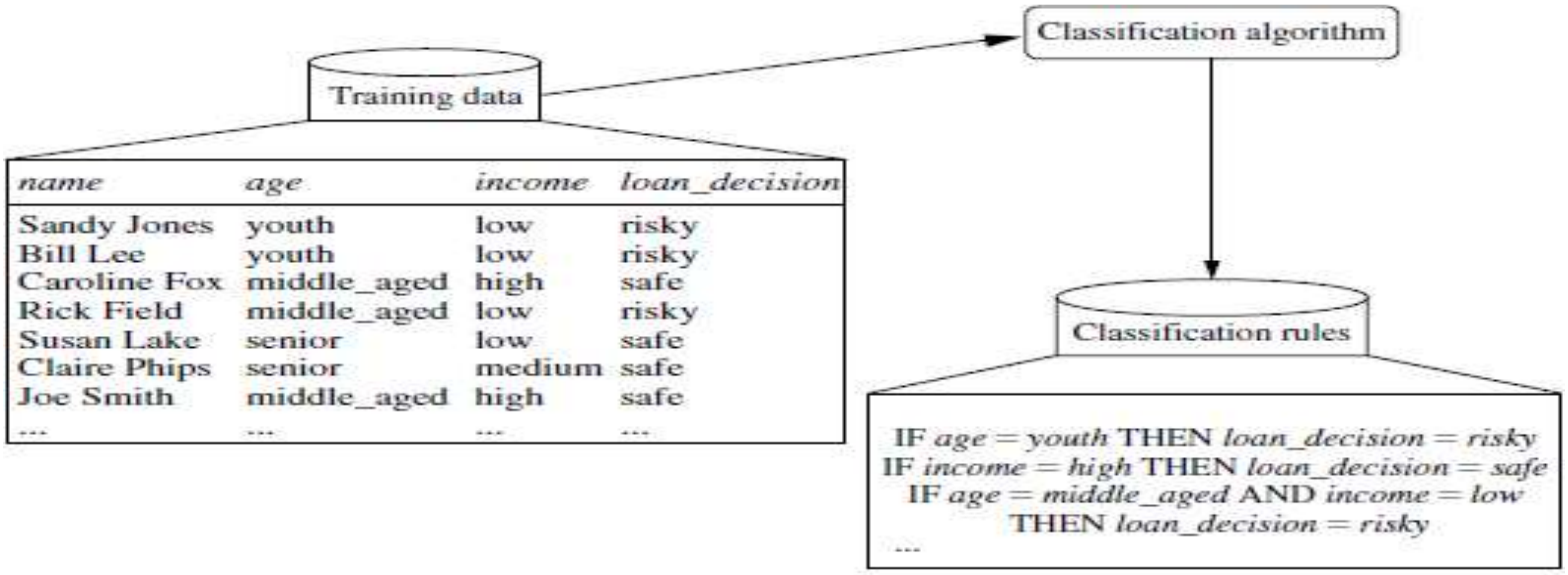
- Classification is the process of predicting the class of given data points. Classes are sometimes called targets, labels or categories.
- Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).
- **There are 2 types of learners in classification— lazy learners and eager learners.**
- **Lazy learners** store the training data and wait until testing data appears. When it does, classification is conducted based on the most related stored training data. Compared to eager learners, lazy learners spend less training time but more time in predicting.
- **Eager learners** construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Because of this, eager learners take a long time for training and less time for predicting.

Lazy Learners vs Eager Learners

Aspect	Lazy Learners	Eager Learners
Timing of Model Building	The model is built during prediction.	The model is built before prediction.
Data Dependency	Relies heavily on the training data during prediction.	Less dependent on training data during prediction.
Computational Efficiency	Faster during training, but slower during prediction due to real-time model building.	Slower during training, but faster during prediction due to pre-built model.
Examples	k-Nearest Neighbors (KNN)	Decision Trees, Support Vector Machines (SVM), Neural Networks
Memory Usage	Less memory usage during training, but more during prediction.	More memory usage during training, but less during prediction.

Classification

(a) *Learning*: Training data are analyzed by a classification algorithm. Here, the class label attribute is *loan decision*, and the learned model or classifier is represented in the form of classification rules.



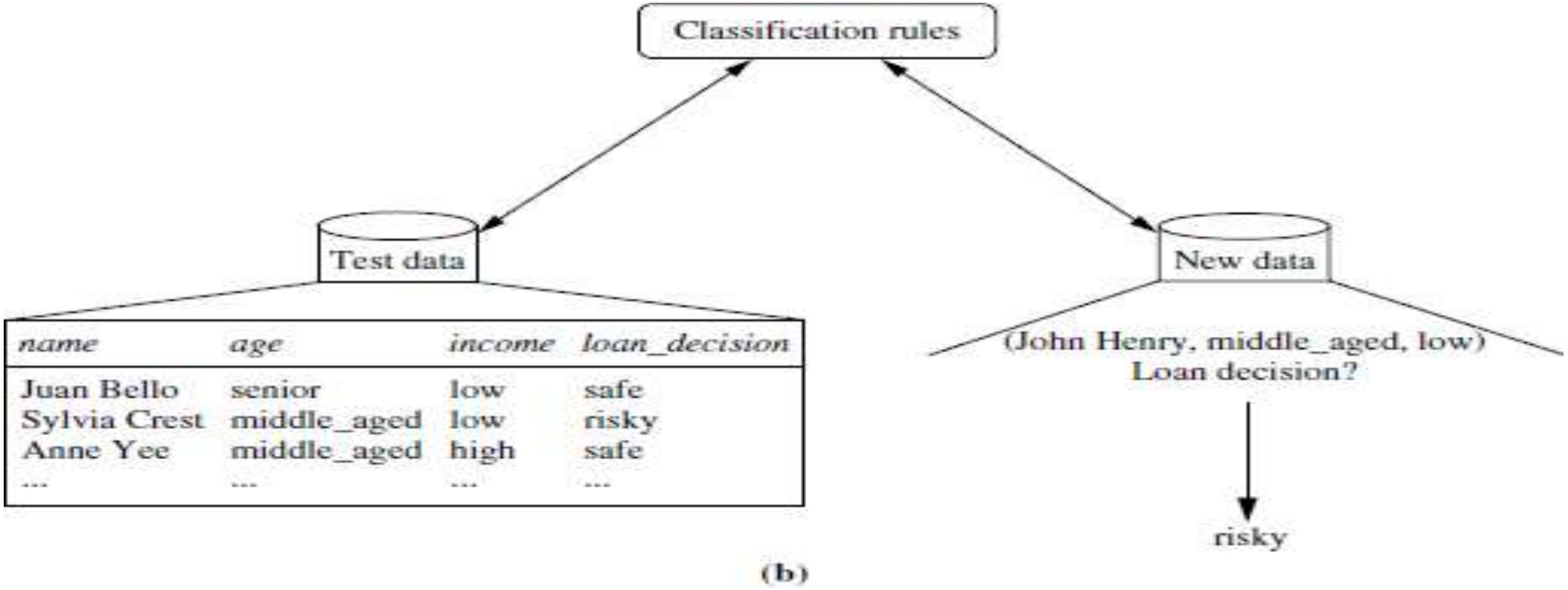
name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

IF *age* = *youth* THEN *loan_decision* = *risky*
IF *income* = *high* THEN *loan_decision* = *safe*
IF *age* = *middle_aged* AND *income* = *low*
THEN *loan_decision* = *risky*
...

(a)

Classification

(b) *Classification*: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.



name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

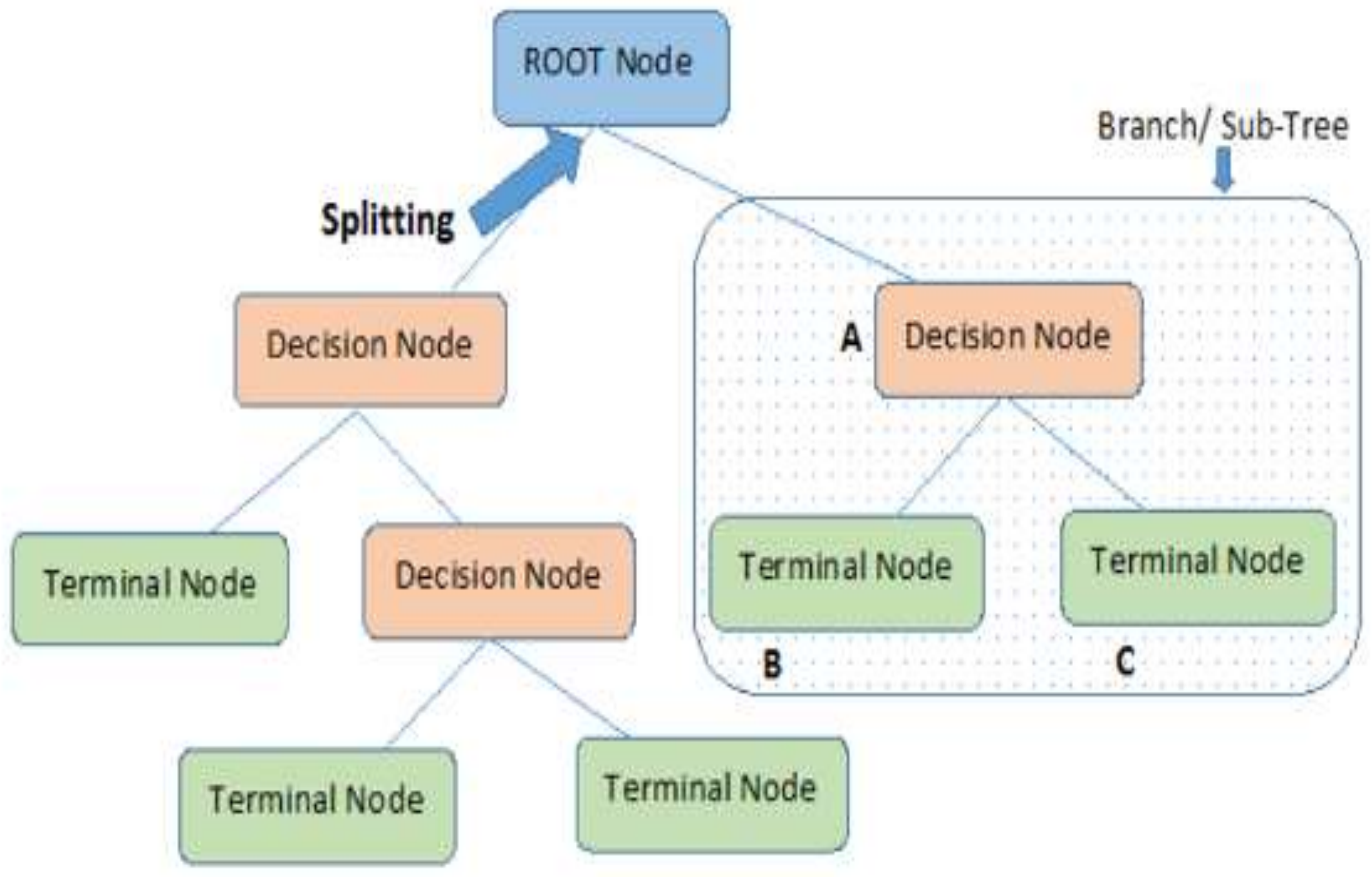
Decision Tree Induction

- A **Decision Tree** is a non-parametric supervised learning algorithm for classification and regression tasks.
- It breaks down a dataset into smaller and smaller subsets while at the same time, an associated decision tree is incrementally developed.
- It has a hierarchical tree structure consisting of a **root node, branches, internal nodes and leaf nodes**. **The final result is a tree with decision nodes and leaf nodes.**
- Decision Trees are the foundation for many classical machine learning algorithms like **Random Forests, Bagging**, and **Boosting**.
- **Types of Decision Trees:**
- **ID3 (Iterative Dichotomiser 3)** → uses Entropy function and Information gain as metrics.
- **CART (Classification and Regression Trees)** → uses Gini Index(Classification) as metric.

Decision Tree Induction

Decision Tree Terminology:

- Root Node
- Decision Nodes
- Leaf Nodes
- Sub-Tree
- Pruning
- Parent and Child node

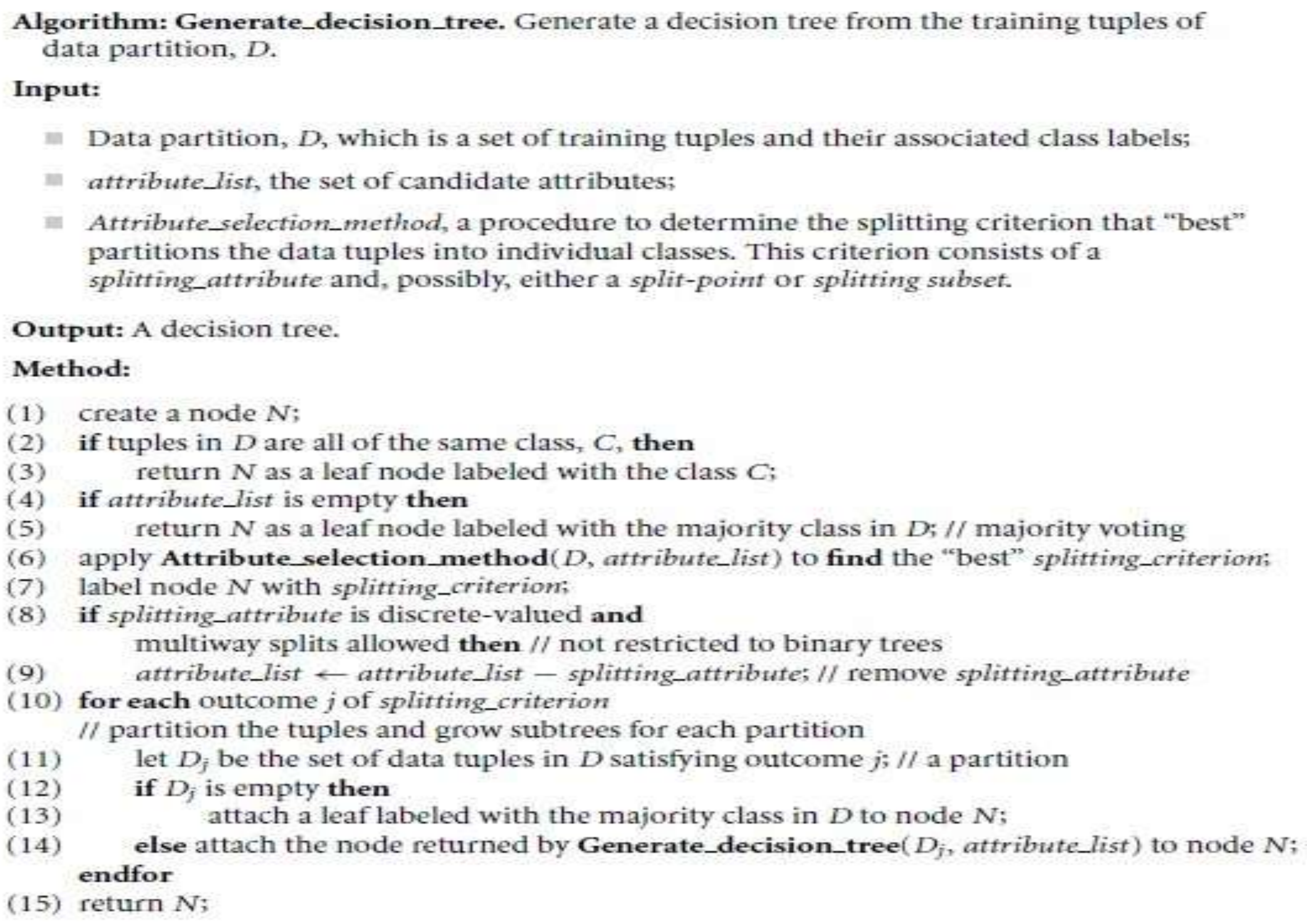


Decision Tree Induction – ID3

- ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.
- The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking.
- A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.
- Steps of ID3 Algorithm:

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset S into subsets using the feature for which the Information Gain is maximum.
3. Make a decision node using the feature with the maximum Information gain.
4. If all rows belong to same class, make current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Decision Tree Induction – ID3



Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

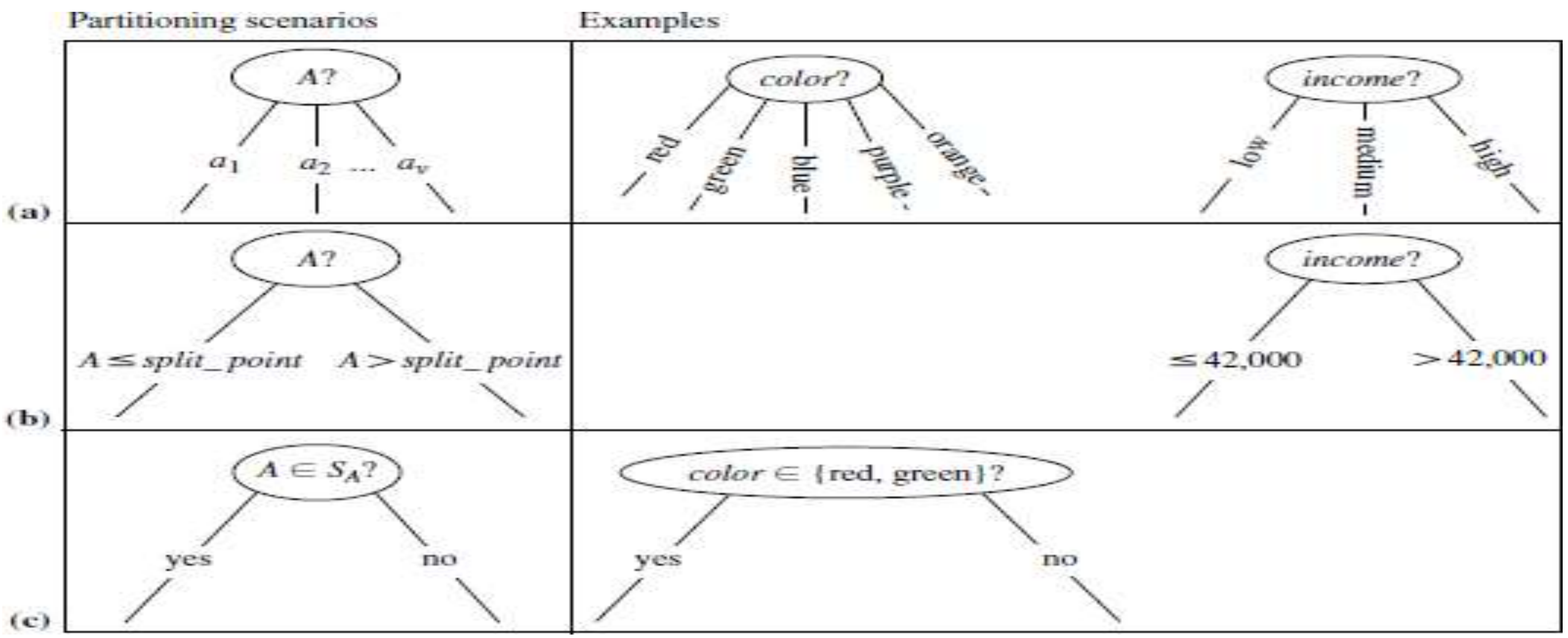
```
(1)   create a node N;
(2)   if tuples in D are all of the same class, C, then
(3)       return N as a leaf node labeled with the class C;
(4)   if attribute_list is empty then
(5)       return N as a leaf node labeled with the majority class in D; // majority voting
(6)   apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)   label node N with splitting_criterion;
(8)   if splitting_attribute is discrete-valued and
          multiway splits allowed then // not restricted to binary trees
(9)       attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10)  for each outcome j of splitting_criterion
      // partition the tuples and grow subtrees for each partition
(11)      let $D_j$ be the set of data tuples in D satisfying outcome j; // a partition
(12)      if $D_j$ is empty then
(13)          attach a leaf labeled with the majority class in D to node N;
(14)      else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node N;
      endfor
(15)  return N;
```

Decision Tree Induction – ID3

- The tree starts as a single node, N, representing the training tuples in D (step1).
- If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are terminating conditions. All terminating conditions are explained at the end of the algorithm.
- Otherwise, the algorithm calls Attribute selection method to determine the splitting criterion. The splitting criterion tells us which attribute to test at node N by determining the "best" way to separate or partition the tuples in D into individual classes (step 6).
- The node N is labeled with the splitting criterion, which serves as a test at the node (step 7). A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly (steps 10 to 11).2

Decision Tree Induction – ID3

- There are three possible scenarios- (a) If *A* is discrete-valued, then one branch is grown for each known value of *A*. (b) If *A* is continuous-valued, then two branches are grown, corresponding to *A* <=*split point* and *A* > *split point*. (c) If *A* is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for *A*.



Decision Tree Induction – ID3

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$

- **Expected information (entropy) needed to classify a tuple in D:**

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- **Information needed (after using A to split D into v partitions) to classify D:**

$$E(A) = Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- **Information gained by branching on attribute A**

$$Gain(A) = Info(D) - Info_A(D)$$

Decision Tree Induction – ID3 - Problem

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$\frac{5}{14} I(2,3)$ **means "age <=30" has 5 out of 14 samples, with 2 yes and 3 no's.**

Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

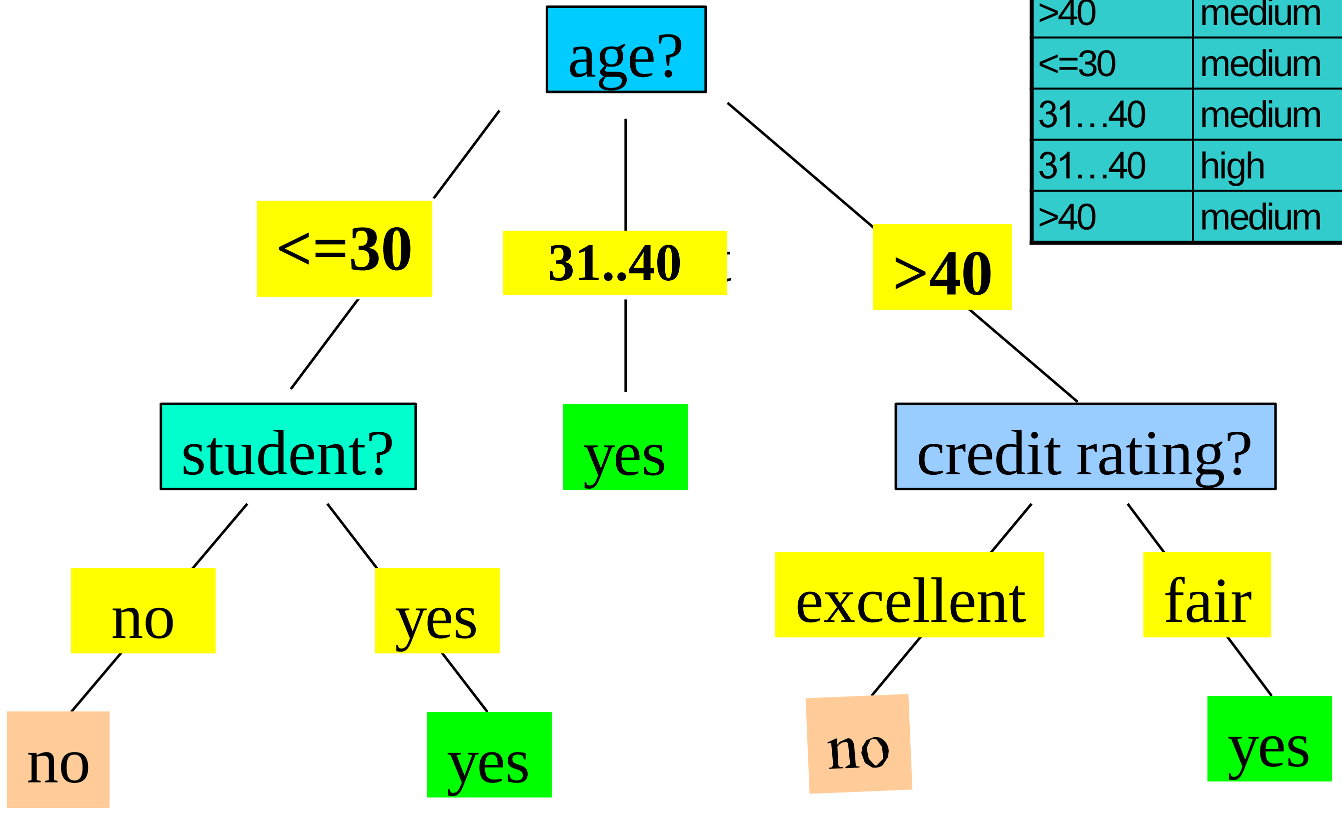
$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Decision Tree Induction – ID3 - Problem

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3.

- Resulting tree:



Decision Tree – ID3

Pros:

- **Builds fastest tree**
- **Builds short tree**
- **Searches the whole dataset to build the decision tree**
- **Generates Multi-branch tree**

Cons:

- **Handles only Categorical data**
- **Cannot perform pruning**
- **Prioritizes attributes with more values**
- **Do not handle imbalanced and missing data values**

Decision Tree – CART

- CART is an alternative decision tree building algorithm.
- It can handle both classification and regression tasks.
- It generates binary trees. This algorithm uses a new metric named gini index to create decision points for classification tasks.
- Gini index is a metric for classification tasks in CART.
- It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

$$\text{Gini Index(Class)} = 1 – \Sigma\,(P_i)^2 \text{ for } i=1 \text{ to number of subclasses}$$

- Gini Index of feature = Weighted sum of the Gini Index of classes in a feature.

$$\text{Gini Index(feature)} = \sum_{j=1}^{m} \frac{|D_j|}{|D|} Gini(D_j)$$

Decision Tree Induction - CART

Procedure:

1. **Start: Begin with the full dataset.**

2. **Split: Find the best feature and split point to divide the data into two subsets.**

3. **Create Node: Make a node in the tree based on the chosen split, with lower gini index.**

4. **Partition Data: Split the dataset into two subsets based on the chosen split.**

   2n - 2 possible ways to form two partitions of the data.

   n is the number of categories in an attribute

5. **Recursive Splitting: Repeat steps 2-4 for each subset until stopping criteria are met.**

6. **Stopping Criteria: Stop growing the tree when certain conditions are reached, all instances belong to the same class.**

7. **Pruning (optional): Trim the tree to avoid overfitting.**

8. **Output: Use the resulting tree for making predictions on new data.**

Decision Tree Induction - CART

- If a data set *D* contains examples from *n* classes, gini index, ***gini*(*D*) is defined as**

$$gini(D) = 1 - \sum_{j=1}^{n} (p_j)^2$$

where p_j is the relative frequency of class *j* in *D*

- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- **Reduction in Impurity:**

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- **The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)**

Decision Tree Induction – CART : Example

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- **Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2**

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- **All attributes are assumed continuous-valued**
- **May need other tools, e.g., clustering, to get the possible split values**
- **Can be modified for categorical attributes**

Decision Tree Induction - CART

Pros:

- Handles both categorical and numerical values
- Handles Outliers
- Generates only binary trees

Cons:

- Produces unstable decision trees
- Has a preference towards multivalued attributes

Naïve Bayes Algorithm

Naïve Bayesian Classification

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a Statistical Classifier – performs probabilistic prediction i.e. predicts class membership probabilities.(The probability the given tuple belongs to a particular class)
- Some popular applications of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.
- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

Bayes' Theorem: Basics

Goal : Determine the most probable hypothesis given the data plus any initial knowledge of the priori probabilities of various hypothesis in H. We have many hypothesis and we can find out the most probable hypothesis.

Bayes' Theorem:

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Let X be a data sample ("*evidence*"): class label is unknown

Let H be a *hypothesis* that X belongs to class Ci

Classification is to determine P(H|X), (i.e., *posteriori probability of hypothesis*): the probability that the hypothesis holds given the observed data sample X

P(H) (*prior probability*): the initial probability of Hypothesis H

E.g., X will buy computer, regardless of age, income, …

P(X): probability that sample data is observed

P(X|H) (likelihood): probability of observing the sample X, given that the hypothesis holds

E.g., Given that X will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

Given training data X, *posteriori probability of a hypothesis* H, P(H|X), follows the Bayes' theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Informally, this can be viewed as

posteriori = likelihood x prior/evidence

Predicts X belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *k* classes

Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification is to Derive the Maximum Posteriori

Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector X = (x_1, x_2, …, x_n)

Suppose there are *m* classes C_1, C_2, …, C_m.

Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|X)

This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

This greatly reduces the computation cost: Only counts the class distribution

If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

and $P(x_k|C_i)$ is
$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Bayes' Theorem Example

P(King) = 4/52 =1/13

$$P(King|Face) = \frac{P(Face|King).P(King)}{P(Face)}$$

P(Face|King) = 1

P(Face) =12/52 = 3/13

Bayes' Theorem Example

$$P(King) = 4/52 = 1/13$$

$$P(King|Face) = \frac{P(Face|King).P(King)}{P(Face)}$$

$$= \frac{1.(1/13)}{3/13} = 1/3$$

$$P(Face|King) = 1$$

$$P(Face) = 12/52 = 3/13$$

Classification Steps

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Frequency Table		Play	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	3	2

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	Strong	6	2
	Weak	3	3

Classification Steps

Likelihood Table		Play		
		Yes	No	
Outlook	Sunny	3/10	2/4	5/14
	Overcast	4/10	0/4	4/14
	Rainy	3/10	2/4	5/14
		10/14	4/14	

$P(x|c) = P(Sunny|Yes) = 3/10 = 0.3$

$P(x) = P(Sunny) = 5/14 = 0.36$

$P(c) = P(Yes) = 10/14 = 0.71$

Likelihood of 'Yes' given Sunny is

$$P(c/x) = P(Yes/Sunny) = P(Sunny|Yes)* P(Yes) / P(Sunny) = (0.3 \times 0.71) /0.36 = 0.591$$

Similarly Likelihood of 'No' given Sunny is

$$P(c/x) = P(No/Sunny) = P(Sunny|No)* P(No) / P(Sunny) = (0.4 \times 0.36) /0.36 = 0.40$$

Classification Steps

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	Weak	6/9	2/5	8/14
	Strong	3/9	3/5	6/14
		9/14	5/14	

P(Yes|High) = 0.33 x 0.6 / 0.5 = 0.42

P(No|High) = 0.8 x 0.36 / 0.5 = 0.58

P(Yes|Weak) = 0.67 x 0.64 / 0.57 = 0.75

P(No|Weak) = 0.4 x 0.36 / 0.57 = 0.25

Classification Steps

Suppose we have a day with the following values

Outlook	=	Rain
Humidity	=	High
Wind	=	Weak
Play	=	?

Likelihood of 'Yes' on that Day = P(Outlook = Rain|Yes)*P(Humidity= High|Yes)* P(Wind= Weak|Yes)*P(Yes)

= 2/9 * 3/9 * 6/9 * 9/14 = 0.0199

Likelihood of 'No' on that Day = P(Outlook = Rain|No)*P(Humidity= High|No)* P(Wind= Weak|No)*P(No)

= 2/5 * 4/5 * 2/5 * 5/14 = 0.0166

Naïve Bayes Classifier: Problem 1

Class:

C1:buys_computer = ‘yes’

C2:buys_computer = ‘no’

Data to be classified:

X = (age <=30,

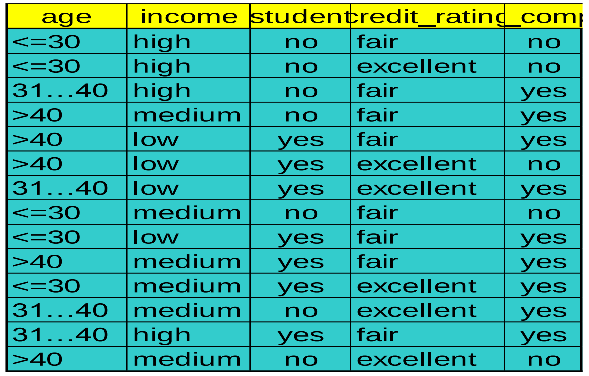
Income = medium,

Student = yes

Credit_rating = fair)

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier:



age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
P(buys_computer = "no") = 5/14= 0.357

Compute $P(X|C_i)$ for each class

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)

Use Laplacian correction (or Laplacian estimator)

Adding 1 to each case

Prob(income = low) = 1/1003

Prob(income = medium) = 991/1003

Prob(income = high) = 11/1003

The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

Advantages:

- Easy to implement
- Good results obtained in most of the cases

Disadvantages:

- Assumption: class conditional independence, therefore loss of accuracy
- Practically, dependencies exist among variables
  - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayes Classifier

How to deal with these dependencies? Bayesian Belief Networks

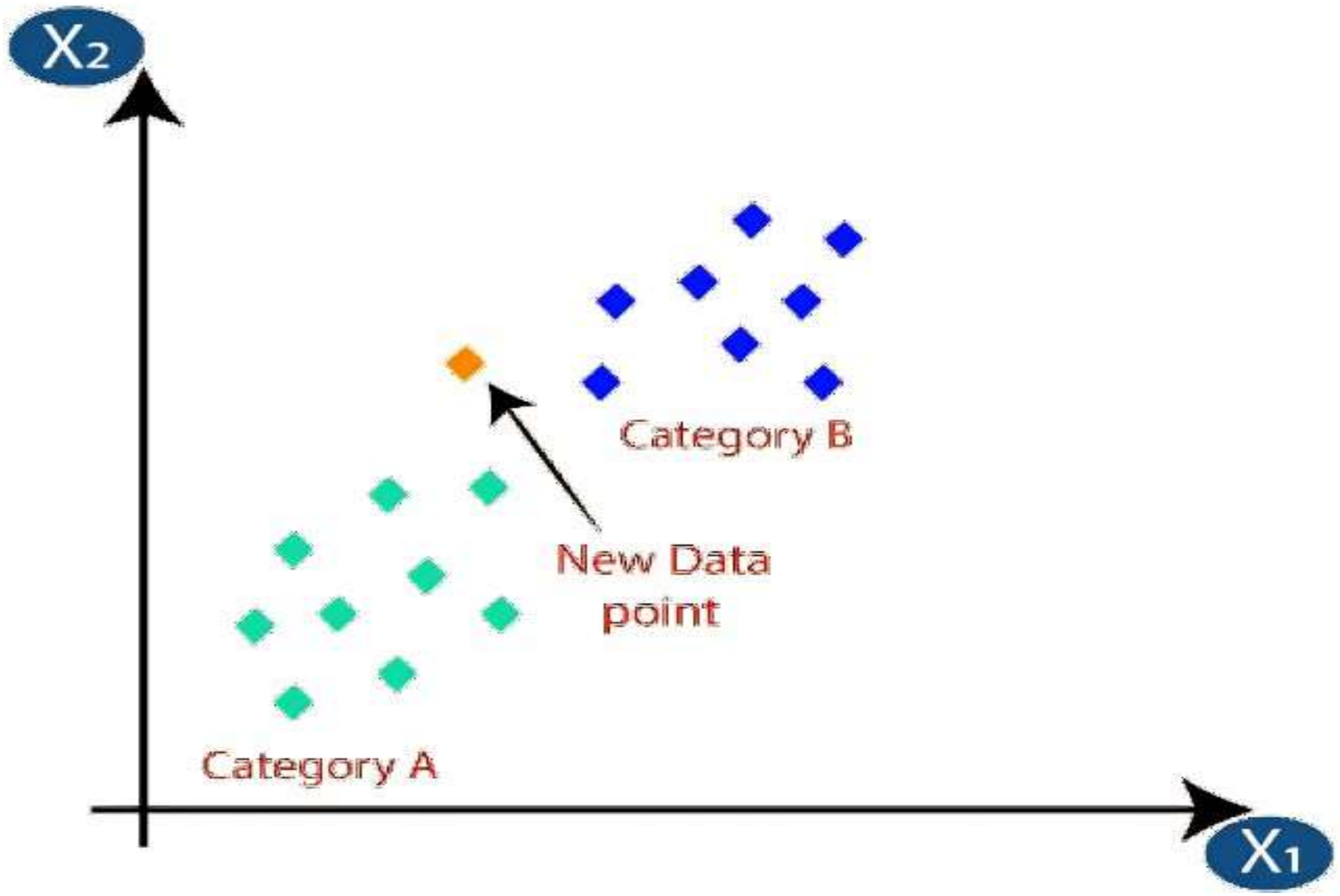
KNN Algorithm

K-Nearest-Neighbours (KNN)

- The abbreviation KNN stands for *"K*-Nearest Neighbor". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements.
- It is a voting system, where the majority class label determines the class label of a new data point among its nearest *'k'* (where k is an integer) neighbors in the feature space.
- 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.

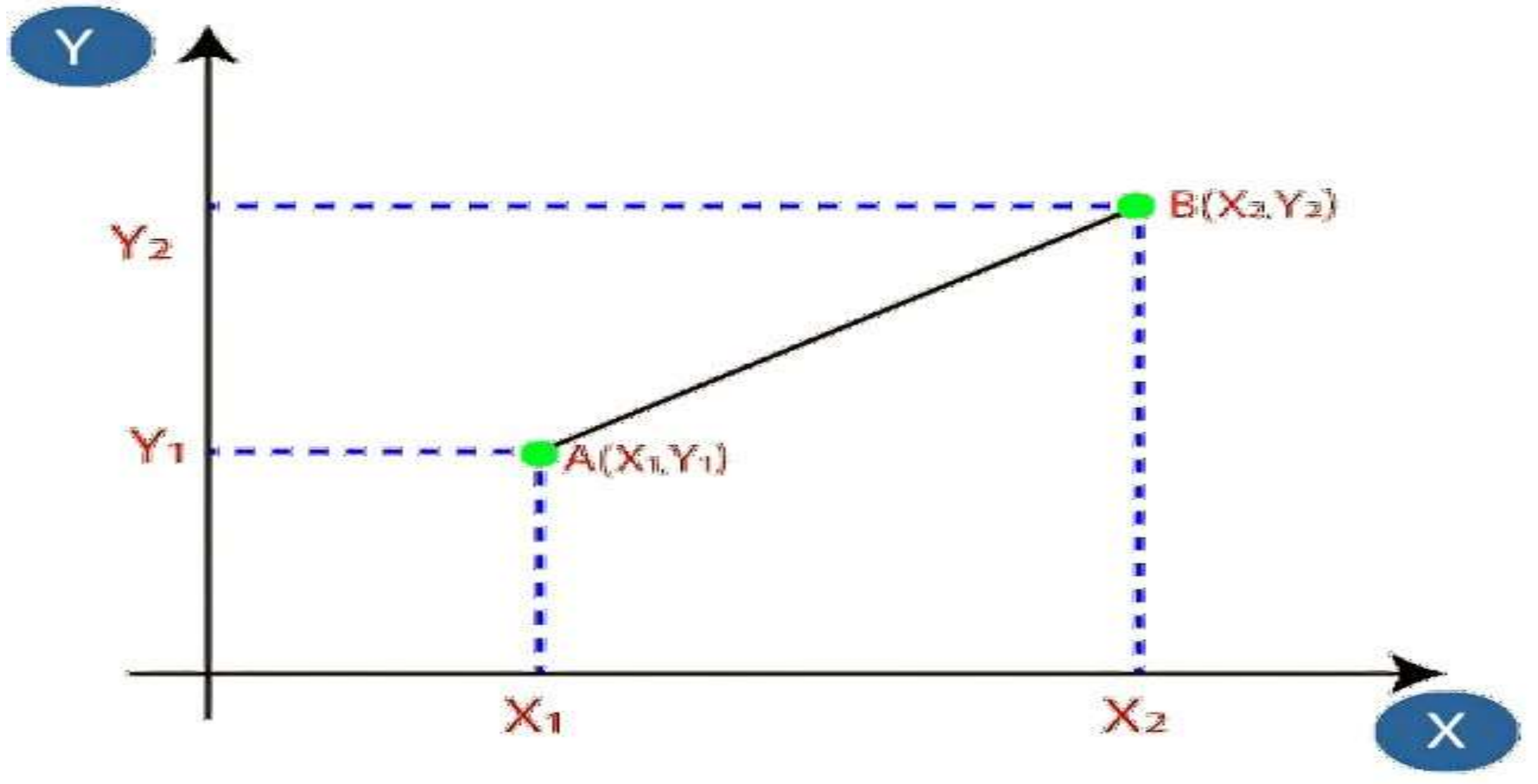
K-Nearest-Neighbours (KNN)

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



K-Nearest-Neighbours (KNN)

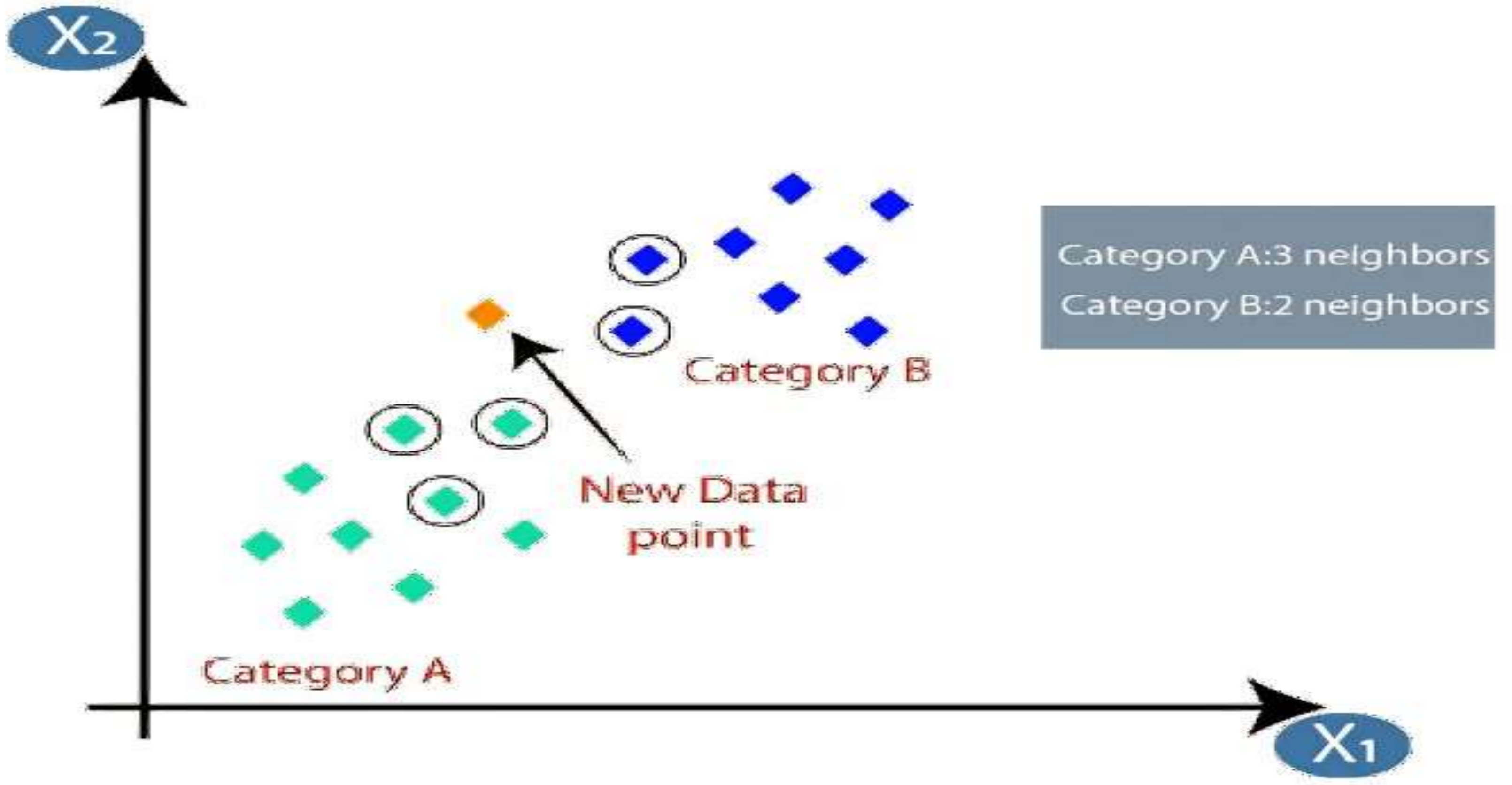
- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

K-Nearest-Neighbours (KNN)

- 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:



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

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 neighbor is maximum.

Step-6: Our model is ready

Types of Distance Metrics

a. Euclidean distance: It is the square root of the sum of squared distance between two points.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

b. Manhattan distance: It is the sum of the absolute values of the differences between two points.

$$\text{Manhattan Distance} = d(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i|\right)$$

Types of Distance Metrics

c. **Minkowski distance:** **It is used to find distance similarity between two points. Based on the below formula changes to either Manhattan distance (When p=1) and Euclidean distance (When p=2).**

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

d. **Hamming Distance:** **It is used for categorical variables. This metric will tell whether two categorical variables are the same or not.**

$$\text{Hamming Distance } = D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right)$$

$$\begin{array}{ll} x=y & D=0 \\ x\neq y & D\neq 1 \end{array}$$

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**

KNN Algorithm

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

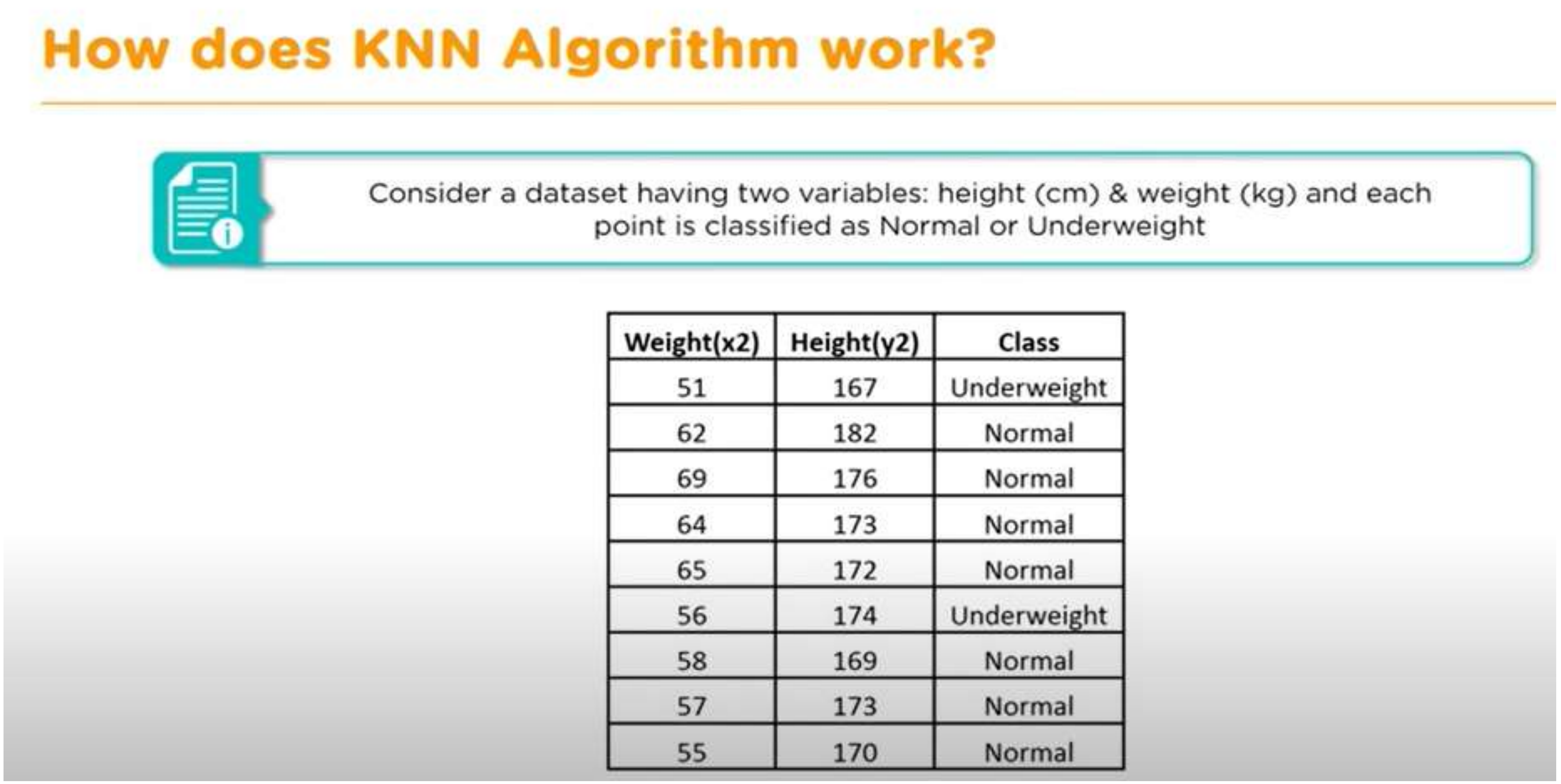
Disadvantages of KNN Algorithm:

- Always needs to determine the value of K,

  which may be complex some time.
- The computation cost is high because of calculating the distance

  between the data points for all the training samples.

K-Nearest-Neighbours (KNN) – Problem 1

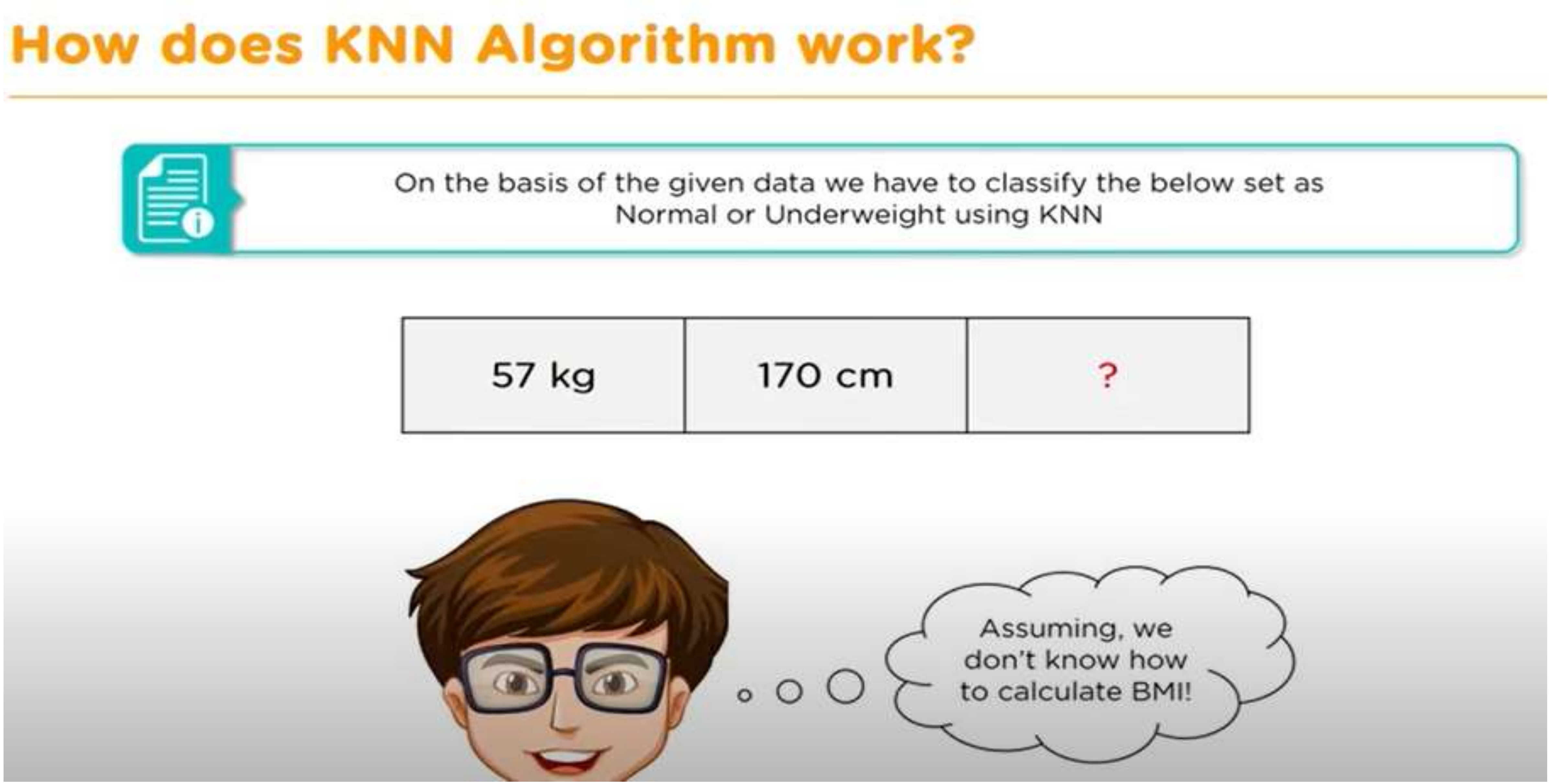


How does KNN Algorithm work?

Consider a dataset having two variables: height (cm) & weight (kg) and each point is classified as Normal or Underweight

Weight(x2)	Height(y2)	Class
51	167	Underweight
62	182	Normal
69	176	Normal
64	173	Normal
65	172	Normal
56	174	Underweight
58	169	Normal
57	173	Normal
55	170	Normal

K-Nearest-Neighbours (KNN)



How does KNN Algorithm work?

On the basis of the given data we have to classify the below set as Normal or Underweight using KNN

57 kg	170 cm	?

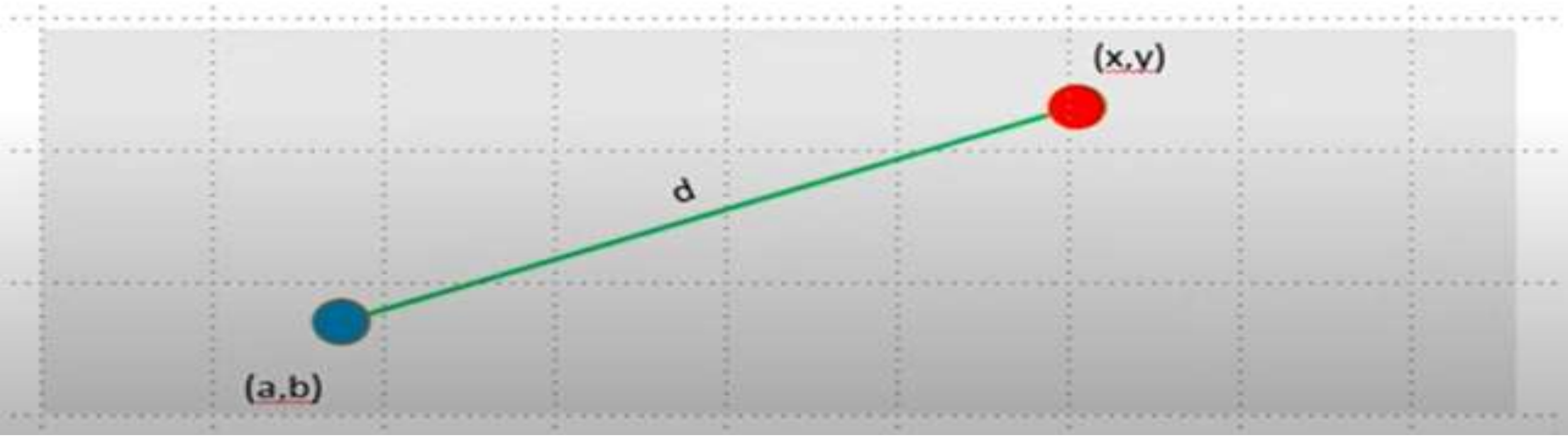
Assuming, we don't know how to calculate BMI!

K-Nearest-Neighbours (KNN)

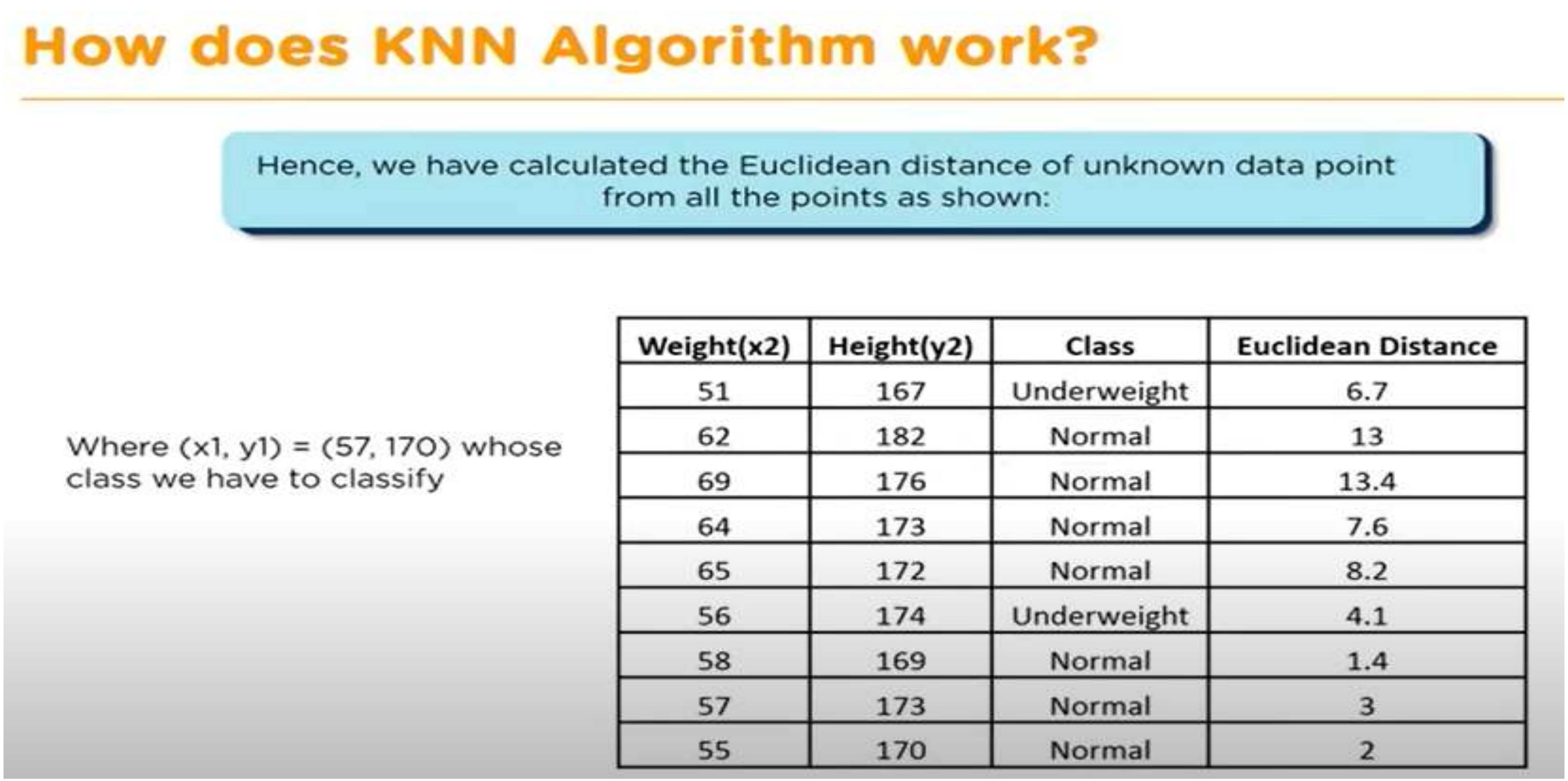
To find the nearest neighbors, we will calculate Euclidean distance

According to the **Euclidean distance** formula, the **distance** between two points in plane with coordinates (x, y) and (a, b) is given by:

$$\boxed{\text{dist}(d) = \sqrt{(x - a)^2 + (y - b)^2}}$$



K-Nearest-Neighbours (KNN)



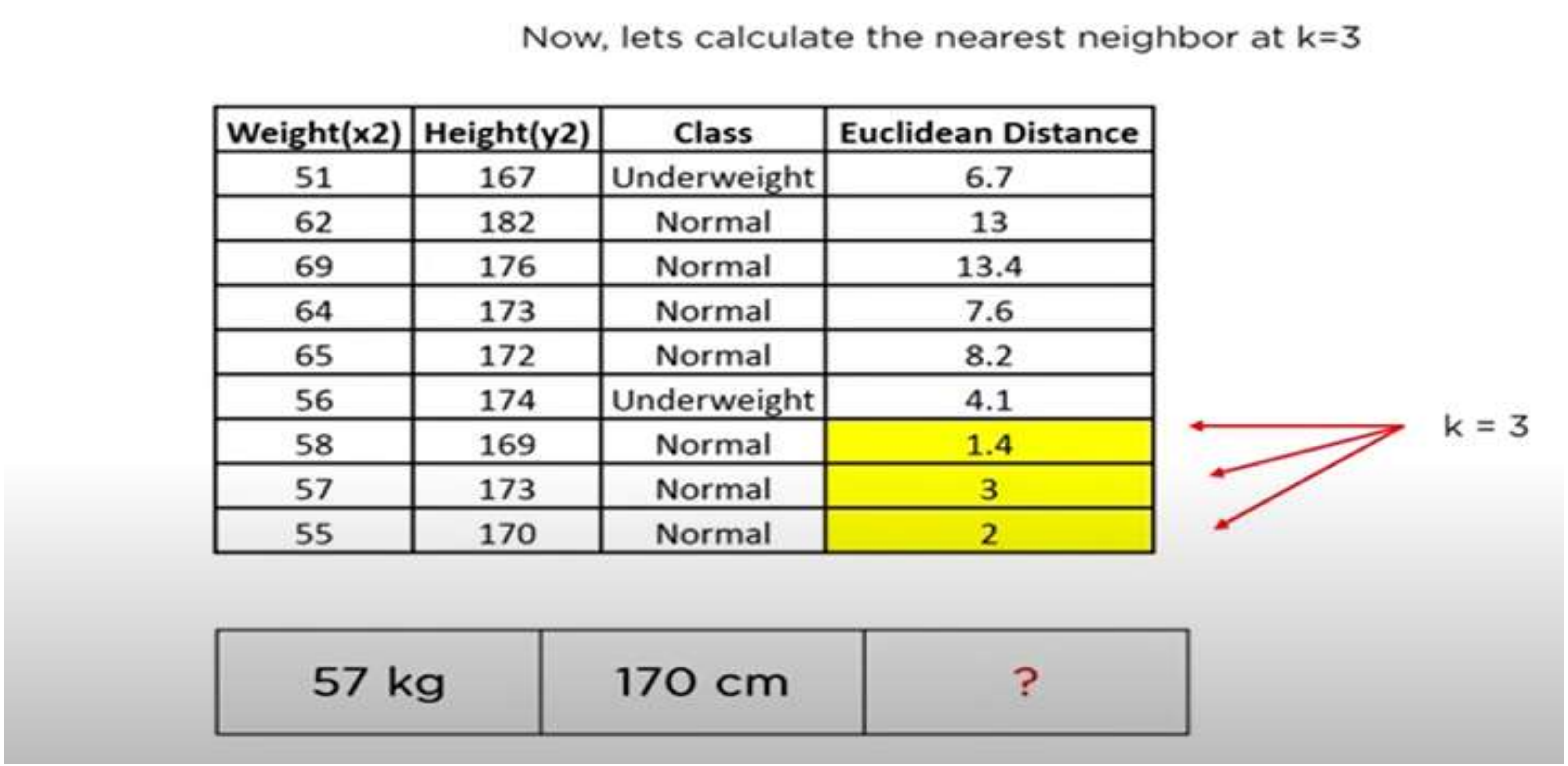
How does KNN Algorithm work?

Hence, we have calculated the Euclidean distance of unknown data point from all the points as shown:

Where (x1, y1) = (57, 170) whose class we have to classify

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

K-Nearest-Neighbours (KNN)



Now, lets calculate the nearest neighbor at k=3

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

k = 3

57 kg	170 cm	?

K-Nearest-Neighbours (KNN)

Class	Euclidean Distance
Underweight	6.7
Normal	13
Normal	13.4
Normal	7.6
Normal	8.2
Underweight	4.1
Normal	1.4
Normal	3
Normal	2

k = 3

So, majority neighbors are pointing towards '*Normal*'

Hence, as per KNN algorithm the class of (57, 170) should be 'Normal'

K-Nearest-Neighbours (KNN) – Problem 2

X1	X2	Classification label
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

If X1= 3 and X2= 7, Find the classification label.

1. Determine parameter K = number of nearest neighbors

Suppose use K = 3

2. Calculate the distance between the query-instance and all the training samples

Coordinate of query instance is (3, 7), instead of calculating the distance we compute square distance which is faster to calculate (without square root)

X1	X2	Square distance to (3,7)
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

3. *Sort the distance and determine nearest neighbors based on the K-th minimum distance*

X1	X2	Square distance to (3,7)	Rank	Is it included in 3-Nearest neighbours
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes
7	4	$(7-3)^2+(4-7)^2=25$	4	No
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes

4. Gather the category " *of the nearest neighbors.* Notice in the second row last column that the category of nearest neighbor (Y) is not included because the rank of this data is more than 3 (=K). $\longrightarrow$ Label

5. *Use simple majority of the category of nearest neighbors as the prediction value of the query instance*

We have 2 good and 1 bad, since 2>1 then we conclude that a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7 is included in **Good** category.

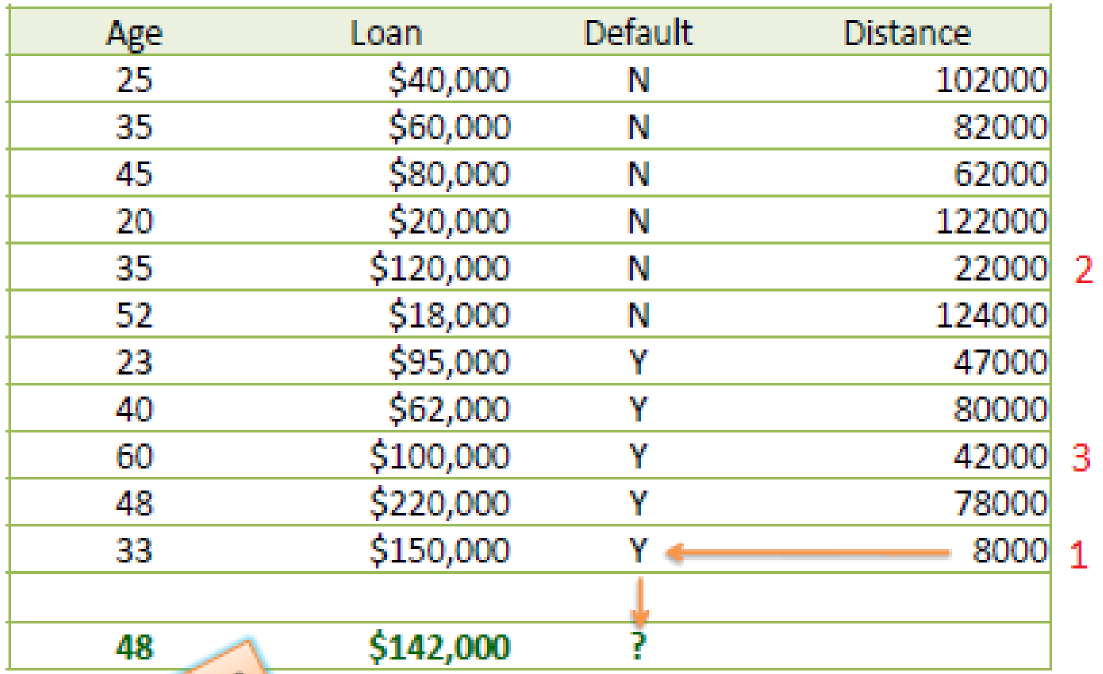
KNN – Problem 2

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.

(**Note: default-N means a borrower has failed to repay a loan.**
default-Y means a borrower have paid their loan)

- We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance.
- If **K=1** then the nearest neighbour is the last case in the training set with Default=Y.
- D = Sqrt[(48-33)^2 + (142000-150000)^2]

  = 8000.01 >> Default=Y
- With K=3, there are two Default=Y and one Default=N
- Out of three closest neighbour, the prediction for the unknown case is again Default=Y.



Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	$142,000	?		

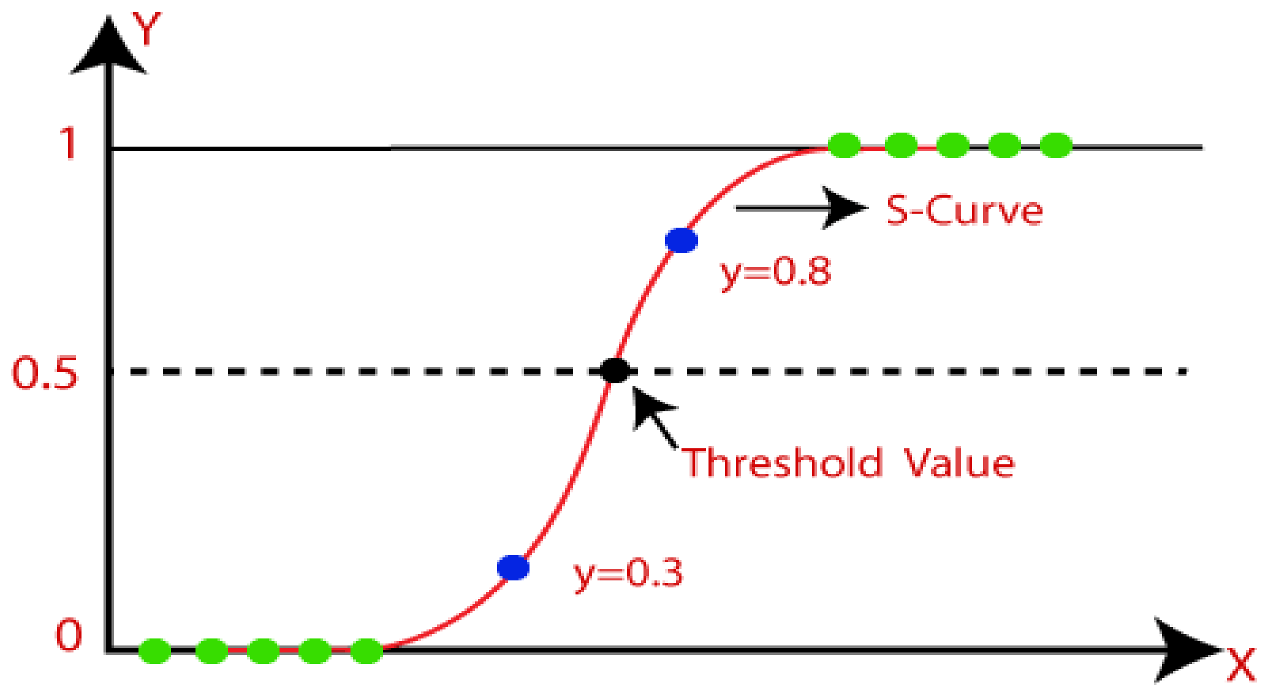
Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Logistic Regression

Logistic Regression

- It is used for predicting the categorical dependent variable using a given set of independent variables.
- Instead of giving a categorical or discrete value, it gives the probabilistic values that lie between 0 and 1.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).



$$p_i = \frac{1}{1+e^{-(WX+b)}}$$

Logistic Regression

- The sigmoid function is at the core of logistic regression, serving as the link function that maps the linear combination of input features to a probability.
- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.
- The S-form curve is called the Sigmoid function or the logistic function.

Assumptions:
- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Cost Function:
- Cost function for logistic regression called log loss which is also derived from the maximum likelihood estimation method.

Linear Vs Logistic Regression

	Linear Regression	Logistic Regression
1	Continuous variables	Categorical variables
2	Solves Regression Problems	Solves Classification Problems
3	Straight line	S-Curve

Logistic Regression

Process using Gradient Descent:

1. Initialize parameters - Intercept and coefficient(s)
2. Substitute the values in sigmoid function and obtain

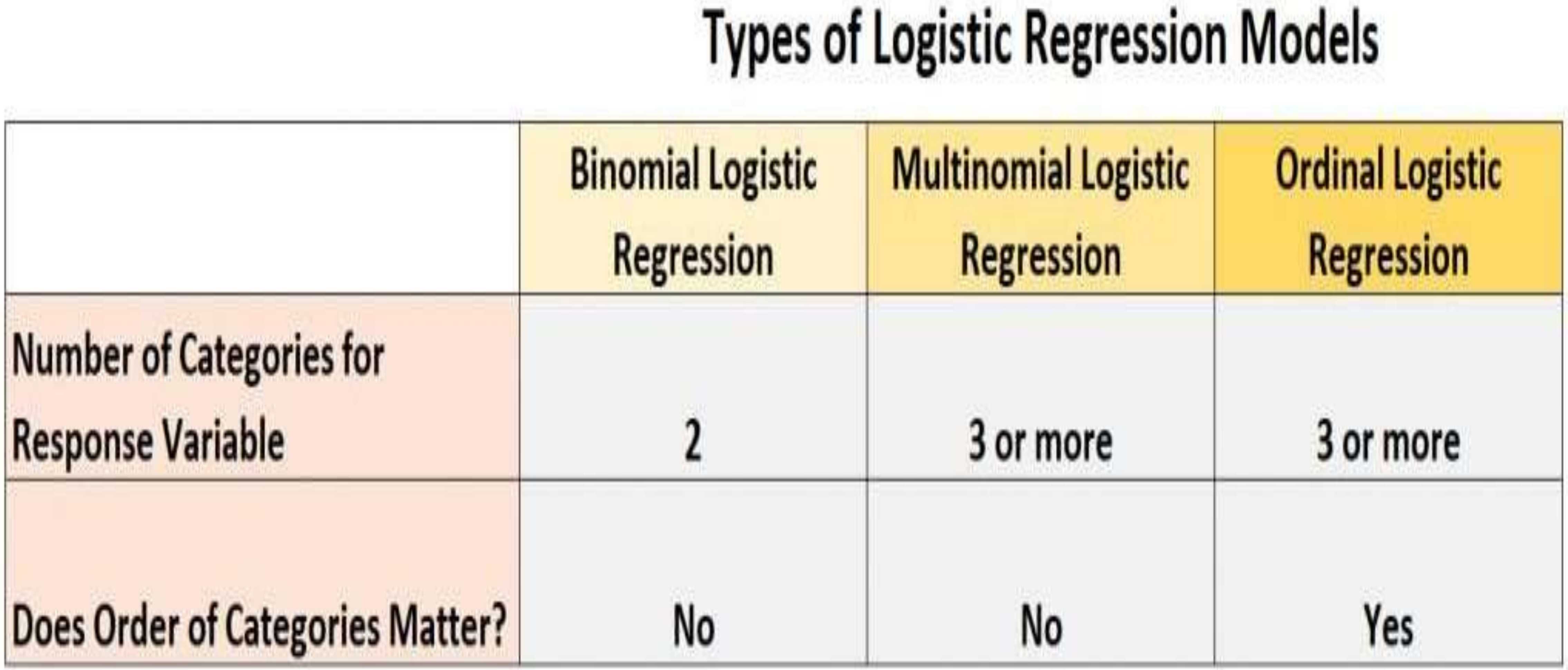
$$p_i = \frac{1}{1+e^{-(WX+b)}}$$

3. Compute the cost function $Logloss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \, ln \, p_i + (1 - y_i) \, ln(1 - p_i)]$

4. Compute Gradients $\frac{\partial J}{\partial w} = \frac{1}{N}\left[\sum_{i=1}^{N}(p_i - y_i)\, x_i\right]$ $\frac{\partial J}{\partial b} = \frac{1}{N}[\sum_{i=1}^{N}(p_i - y_i)]$

5. Update the intercept and coefficients using the partial derivatives
6. Reiterate the steps through steps 1 through 3 until optimal values of parameters are obtained.

Types of Logistic Regression



Types of Logistic Regression Models

	Binomial Logistic Regression	Multinomial Logistic Regression	Ordinal Logistic Regression
Number of Categories for Response Variable	2	3 or more	3 or more
Does Order of Categories Matter?	No	No	Yes

Multinominal Logistic Regression

Assumptions:

- **The Dependent variable should be either nominal or ordinal variable.**
- **Set of one or more Independent variables can be continuous, ordinal or nominal.**
- **The observations and dependent variables must be mutually exclusive and exhaustive.**
- **No Multicollinearity between Independent variables.**
- **There should be no Outliers in the data points.**

Multinominal Logistic Regression

Process using Gradient Descent:

1. Initialize parameters - Intercept and coefficient(s)
2. Compute Logits: Calculate the logits for all samples in the batch for each class using the current model parameters (weights and biases).

$$z_c = \mathbf{X} \cdot \mathbf{W}_c^T + b_c$$

3. Compute Softmax Probabilities: Apply the softmax function to the logits to obtain the predicted probabilities for each class for all samples in the batch.

$$\hat{y}_c = \frac{e^{z_c}}{\sum_{k=1}^{C} e^{z_k}}$$

4. Compute Loss: Calculate the cross-entropy loss between the predicted probabilities and the true class labels for all samples in the batch.

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \cdot \log(p_{i,k})$$

5. Compute Gradients: Compute the gradients of the loss function with respect to the model parameters (weights and biases) using the entire batch of samples. This involves taking the derivative of the loss function with respect to each parameter.

6. Update Parameters: Update the model parameters (weights and biases) using the gradients and the chosen optimization algorithm (e.g., gradient descent, SGD, Adam)..

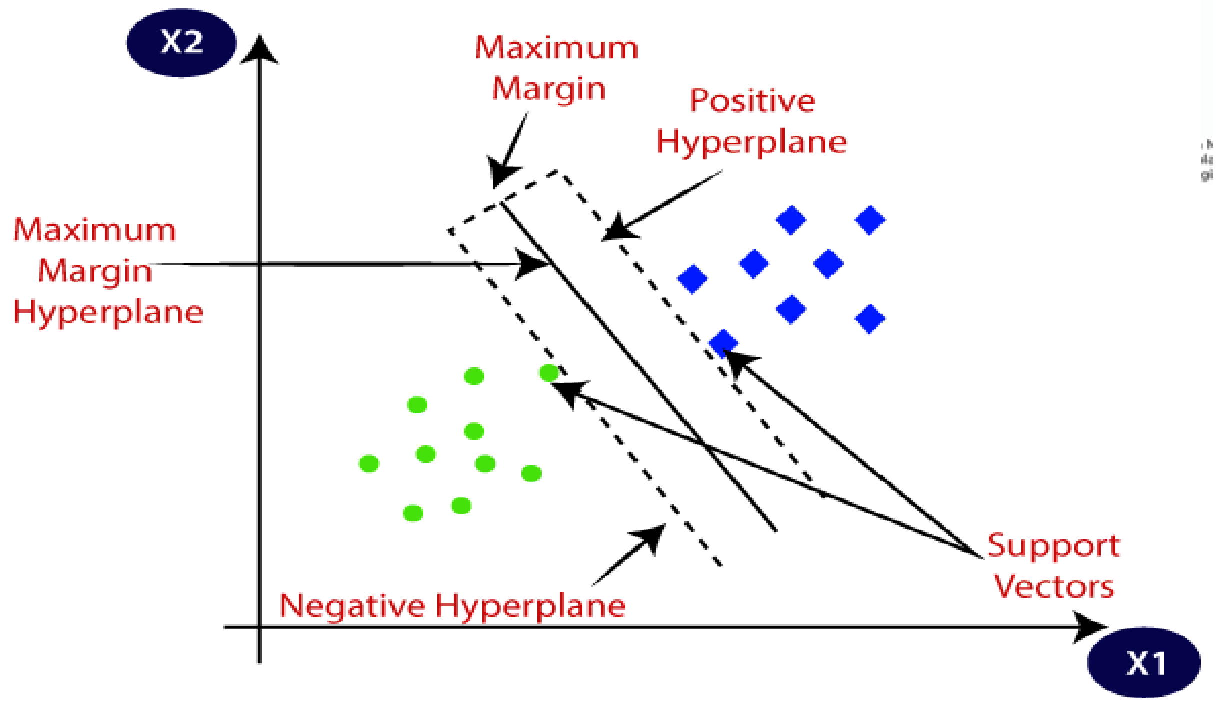
Support Vector Machine

- **Step 1: SVM algorithm predicts the classes. One of the classes is identified as 1 while the other is identified as -1.**
- **Step 2: SVM classifier uses a loss function known as the hinge loss function to find the maximum margin.**
- **Step 3: There is a trade-off between maximizing margin and the loss generated if the margin is maximized to a very large extent. A regularization parameter is used to handle this.**
- **Step 4: weights are optimized by calculating the gradients.**
- **Step 5: The gradients are updated only by using the regularization parameter when there is no error in the classification while the loss function is also used when misclassification happens.**

Support Vector Machine

Support Vector Machine

- Support Vector Machines uses the concept of Margins to come up with predictions.
- The goal of the SVM algorithm is to create a hyperplane in an N-dimensional space that divides the data points belonging to different classes.
- This hyperplane is chosen as the hyperplane providing the maximum margin between the two classes is considered.
- These margins are calculated using data points known as Support Vectors.
- Support Vectors are those data points that are near to the hyper-plane and help in orienting it.



Support Vector Machine

Types of Margins:

Hard Margin: Hard Margin refers to that kind of decision boundary that makes sure that all the data points are classified correctly.

While this leads to the SVM classifier not causing any error, it can also cause the margins to shrink thus making the whole purpose of running an SVM algorithm futile.

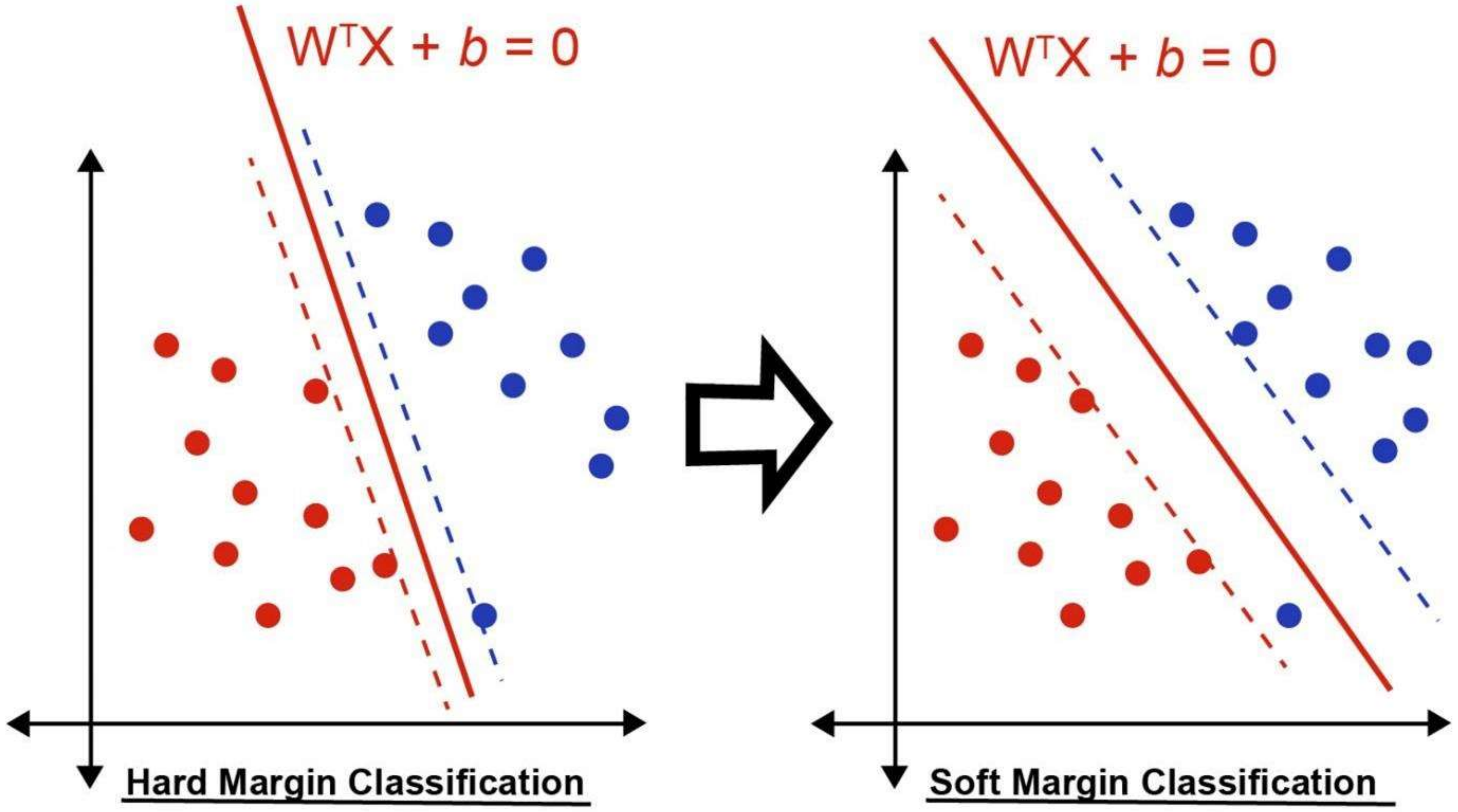
Soft Margin: A regularization parameter is also added to the loss function in the SVM classification algorithm.

This combination of the loss function with the regularization parameter allows the user to maximize the margins at the cost of misclassification.

However, this classification needs to be kept in check, which gives birth to another hyper-parameter that needs to be tuned.

Support Vector Machine

Types of Margins:



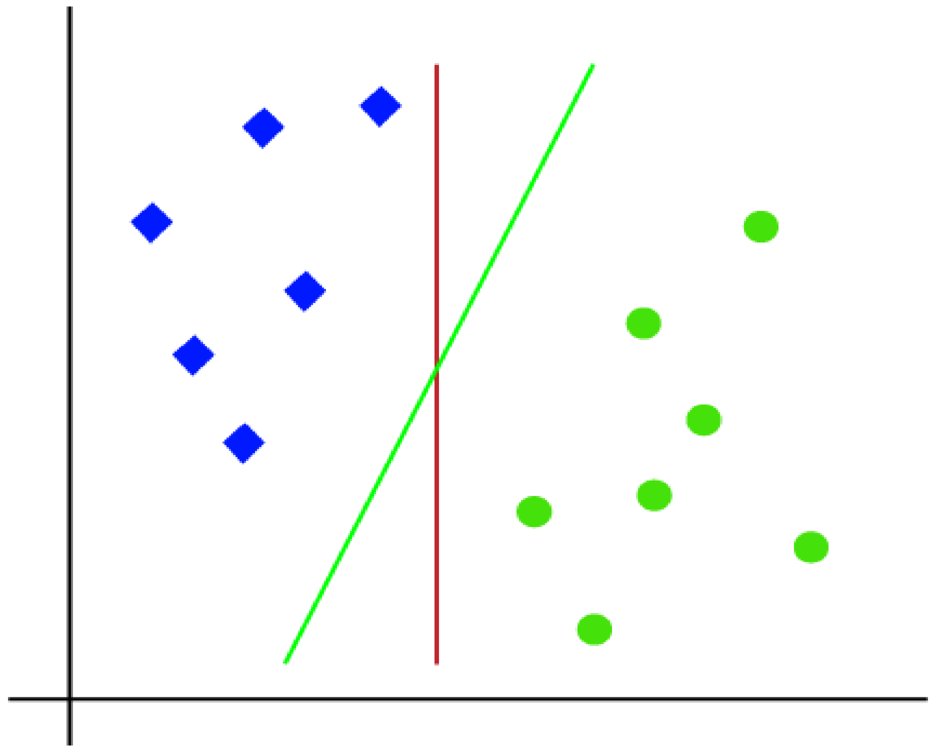
Support Vector Machine

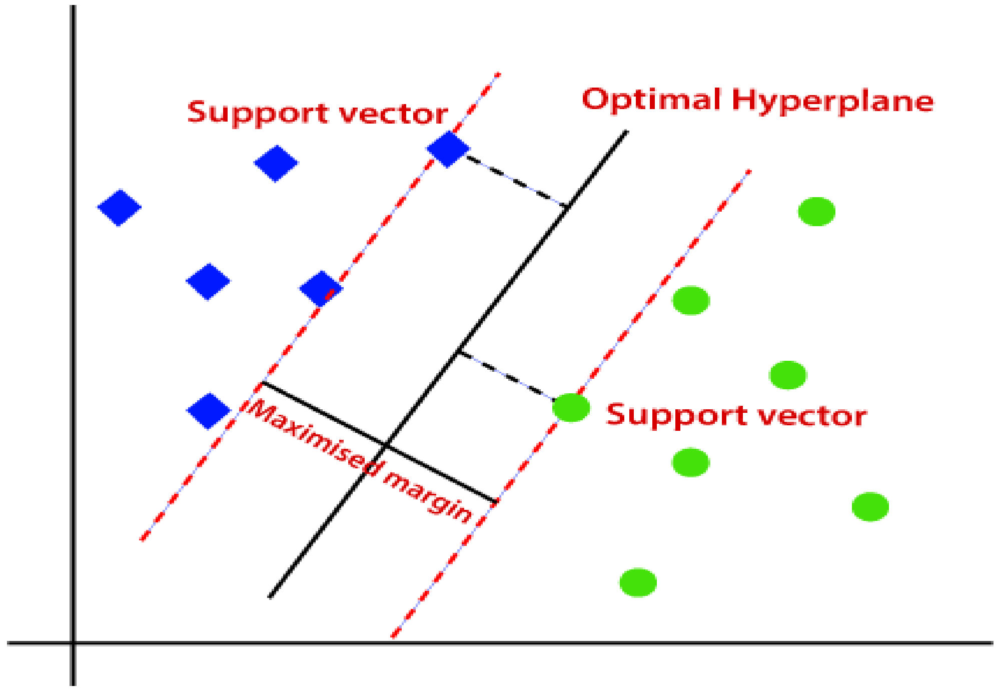
- The data points or vectors that are closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Types of SVM:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Support Vector Machine





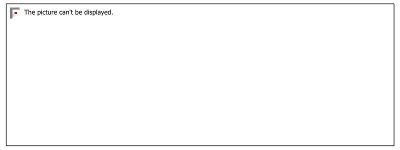
Linear SVM

- To compute the decision boundary, Lagrangian function is used to obtain w and b values.

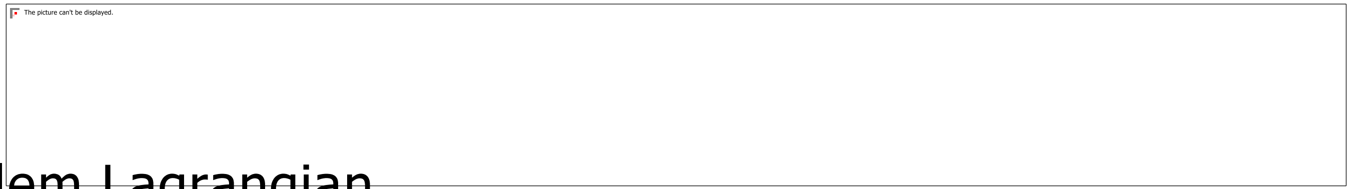
$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{m}\alpha_i[y_i(W.X_i+b)-1]$$

Steps:

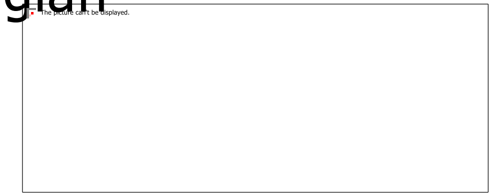
- Compute the partial derivatives of L with respect to w and b.


The picture can't be displayed.

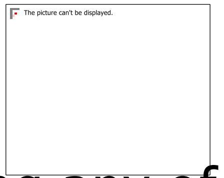
- Solve the dual maximization problem


The picture can't be displayed.

- Compute partial derivatives of dual problem Lagrangian


The picture can't be displayed.

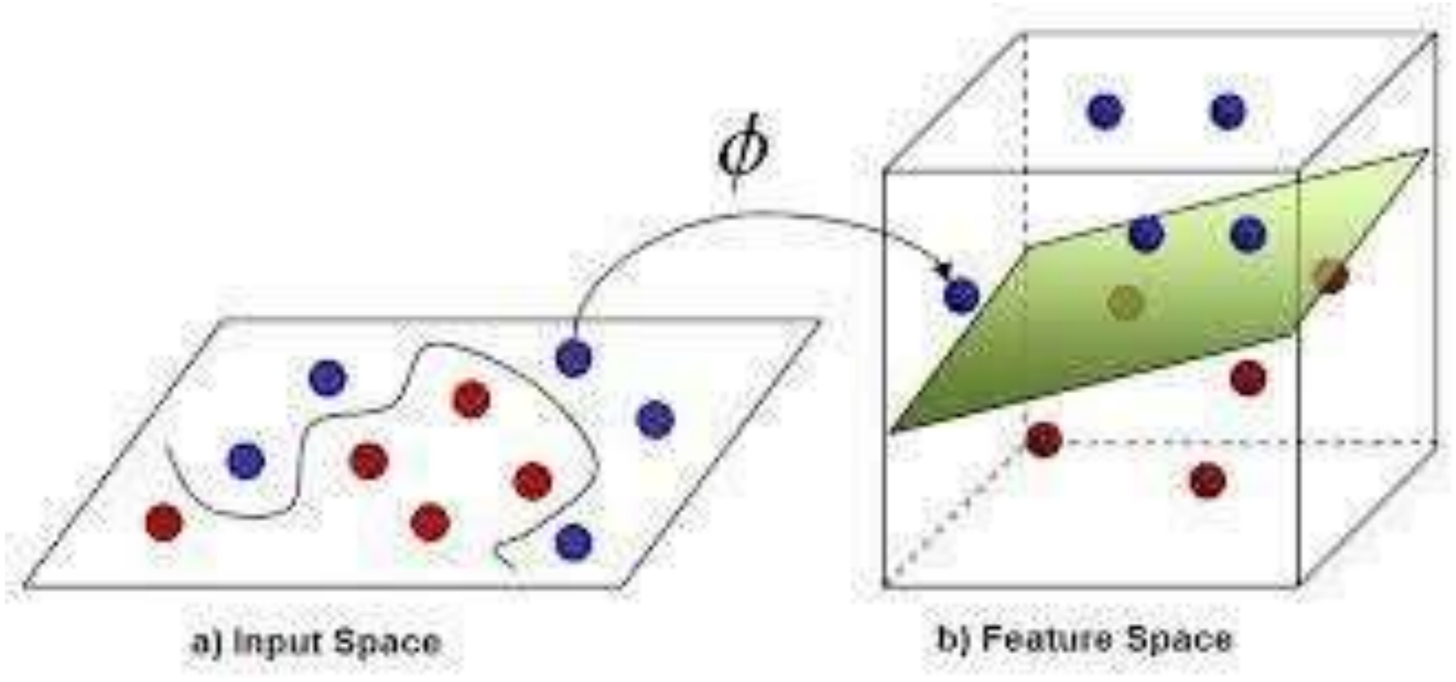
- Compute margin


The picture can't be displayed.

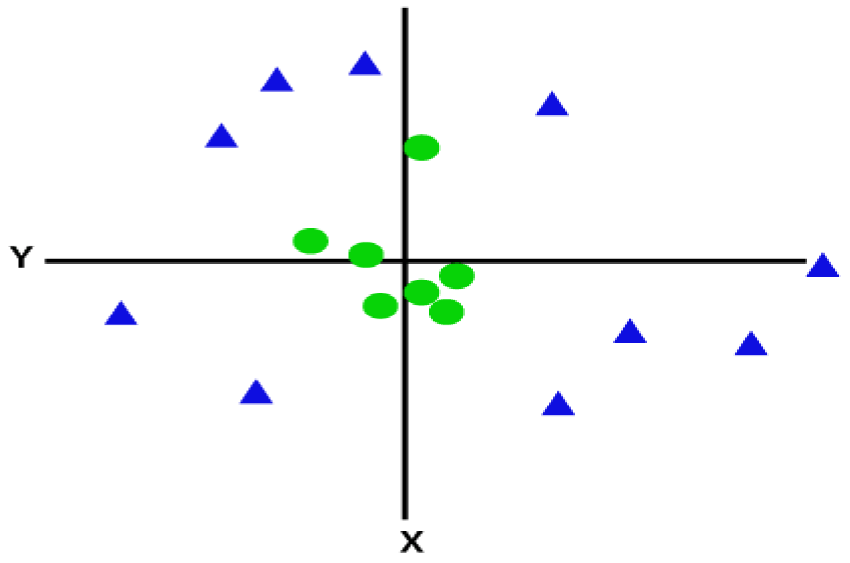
- Compute bias using any of the support vectors

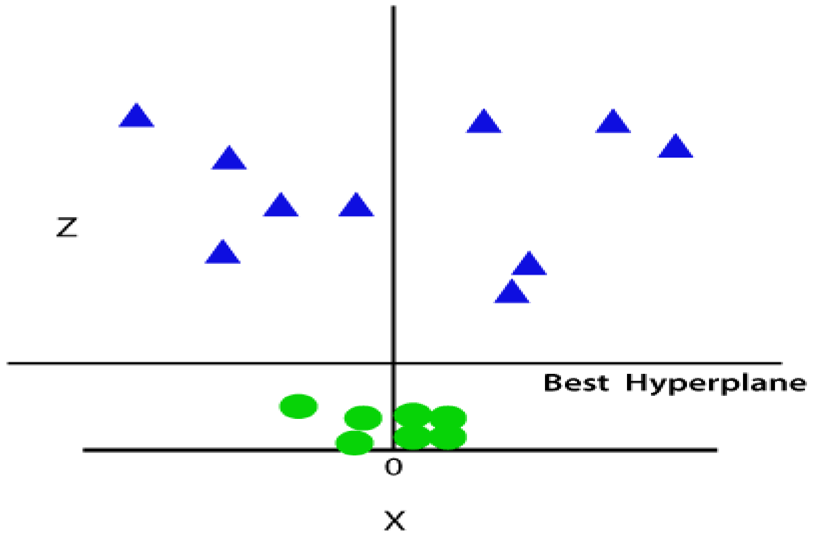
Non Linear SVM

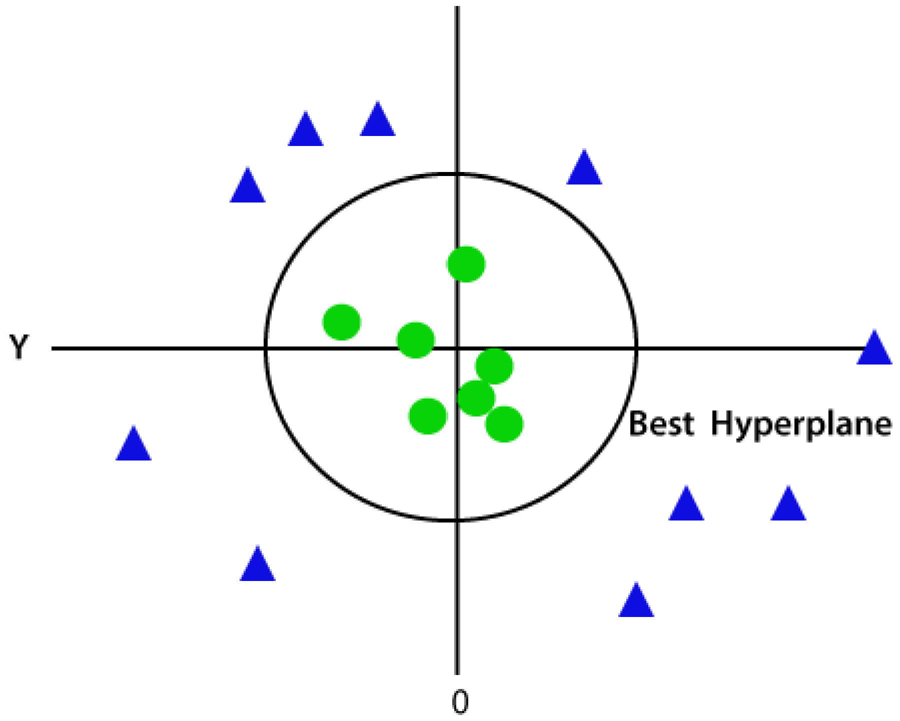
- Nonlinear SVM (Support Vector Machine) is necessary when the data cannot be effectively separated by a linear decision boundary in the original feature space.
- Nonlinear SVM addresses this limitation by utilizing **kernel functions** to map the data into a higher-dimensional space where linear separation becomes possible.
- Kernel Function transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces.



Support Vector Machine







Non Linear SVM

$$L(w^*,b^*,\alpha)=\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(\phi(X_i).\phi(X_j))$$

- $\phi(X_i)$ Is the transformed/ mapped feature space.
- **Issues in feature transformation:**
  - It is difficult to specify the explicit form of *$\phi(X)$*
  - If the original feature space is a quadratic or higher polynomial dimensions, then computing the dot product is time consuming and computationally expensive.
- To solve this, Kernel trick is used. Instead of computing ϕ, Kernel function gives the direct value of similarity.
- For example, if kernel is to be used as a quadratic polynomial of degree 2, then

$$\phi(X,Y)=(X^2,\sqrt{2}XY,Y^2)$$

- Then the feature transformation with kernel results in:

$$\phi(a).\phi(b)=(a.b)^2$$

Non Linear SVM – Types of Kernels

Linear Kernel	$K(x_1, x_2) = (x_1 * x_2)$
Polynomial Kernel	$K(x_1, x_2) = (x_1 + x_2 + 1$
Gaussian Kernel	$K(x_i, x_j) = exp(-\sigma \left\|\|x_i - x_j\|\right\|^2)$
Exponential Kernel	$K(x, y) = \exp(-\frac{\|\|x-y\|\|}{2\sigma^2})$
Laplacian Kernel	$K(x, y) = \exp(-\frac{\|\|x-y\|\|}{\sigma})$
Hyperbolic Kernel	$K(x, y) = tanh(\alpha x^T y + c)$

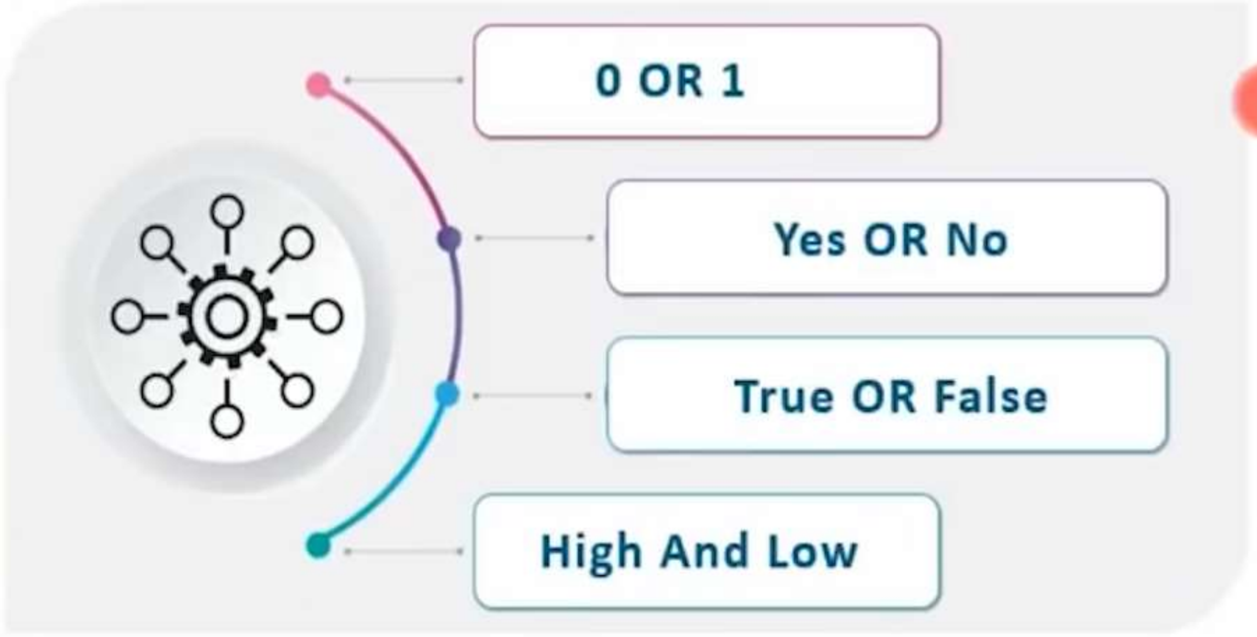
Support Vector Machine

Advantages of SVM:

- Effective in high dimensional cases
- It is memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

Logistic Regression: What And Why?

Logistic Regression produces results in a binary format which is used to predict the outcome of a categorical dependent variable. So the outcome should be discrete/ categorical such as:



Logistic Regression

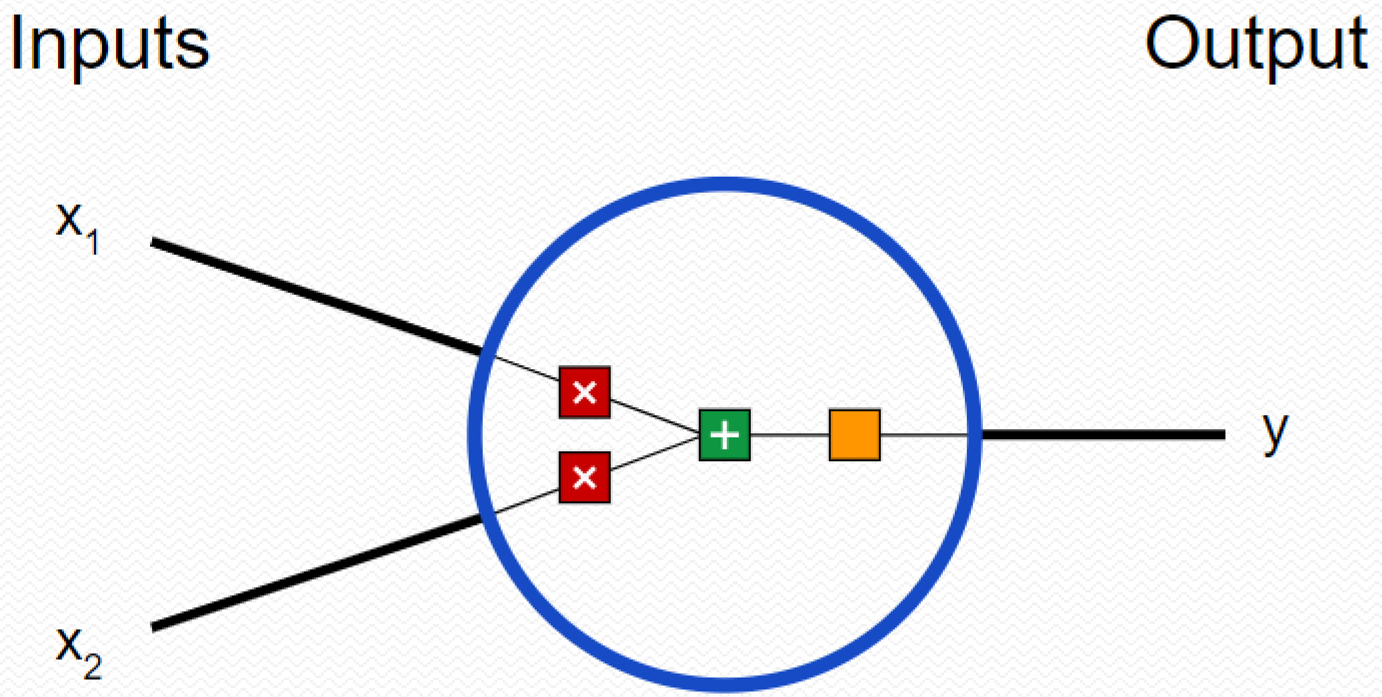
- There are many important research topics for which the dependent variable is "limited."
- For example:
  - Opinion poll responses (Agree, Neutral, Disagree)
  - Whether or not a person smokes (Smoker/Non-Smoker)
  - Success of a medical Treatment (Survives/dies)

  Data is not continuous or distributed normally.
- Logistic regression relies on an estimation procedure
  - $\text{Log}[(p/(1-p)] = C+B_1X_1$
  - Models the probability of an outcome.

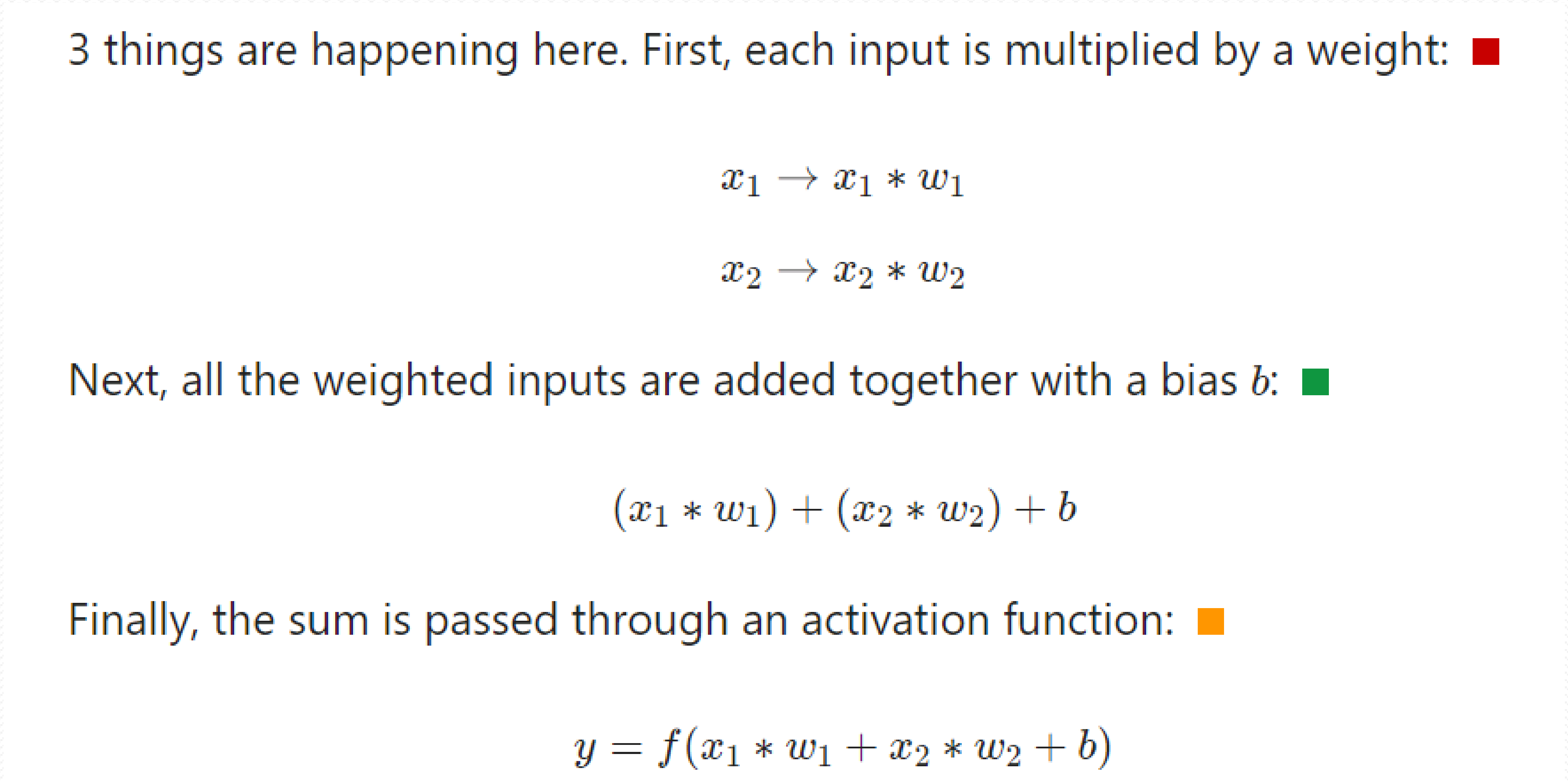
THANK YOU

Neural Network Representation - Building Blocks

- Basic Unit of a Neural Network is **Neuron.**
- A neuron takes inputs, does some math with them, and produces one output.



Neural Network Representation - Building Blocks



3 things are happening here. First, each input is multiplied by a weight: 🟥

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

Next, all the weighted inputs are added together with a bias b: 🟩

$$(x_1 * w_1) + (x_2 * w_2) + b$$

Finally, the sum is passed through an activation function: 🟧

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

This process of passing inputs forward to get an output is known as **feedforward**.

Neural Network Representation

- Neural Networks are identified by nodes, bias, weight, and activation function.
  - **Node**:

    These are the elementary units in a neural network.

    An artificial neuron is a computational unit.

    In neural network, a neuron receives an input, processes it and generates an output that is either sent to other neurons for further processing or is the final output.
  - **Weight:**

    Weights are numerical parameters which determine how strongly each of the neurons affects the other.

    Initialize the weights randomly and these weights are updated during the model training process.
  - **Bias:**

    It means how far off our predictions are from real values.

    The bias allows the model to emphasize specific features to make better generalizations for the larger dataset.
  - **Activation Function:**

    Activation functions define the output of the neuron in a range of values (0 to 1) or (-1 to 1)

Perceptron

Dr. Y. Krishna Bhargavi
Associate Professor
Department of CSE
GRIET

Perceptron

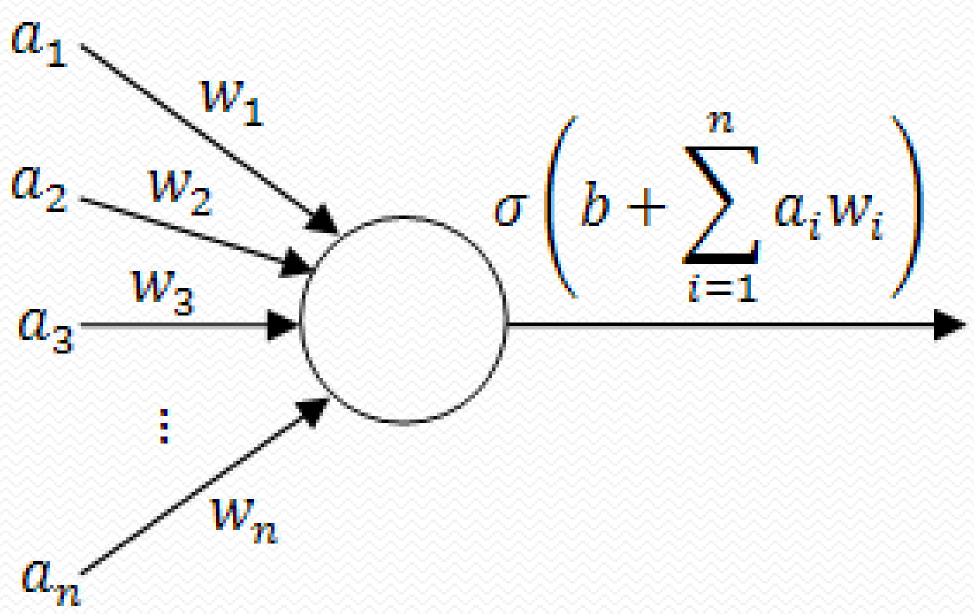
- Frank Rosenblatt first proposed in 1958 is a simple neuron which is used to classify its input into one or two categories.
- Also referred to as Artificial Neuron or neural network unit.
- It performs computations to detect features or patterns in the input data.
- It is this algorithm that allows artificial neurons to learn and process features in a data set.

Components of Perceptron:

- Input values or One input layer
- Weights and Bias
- Net sum
- Activation Function

Types of Perceptron

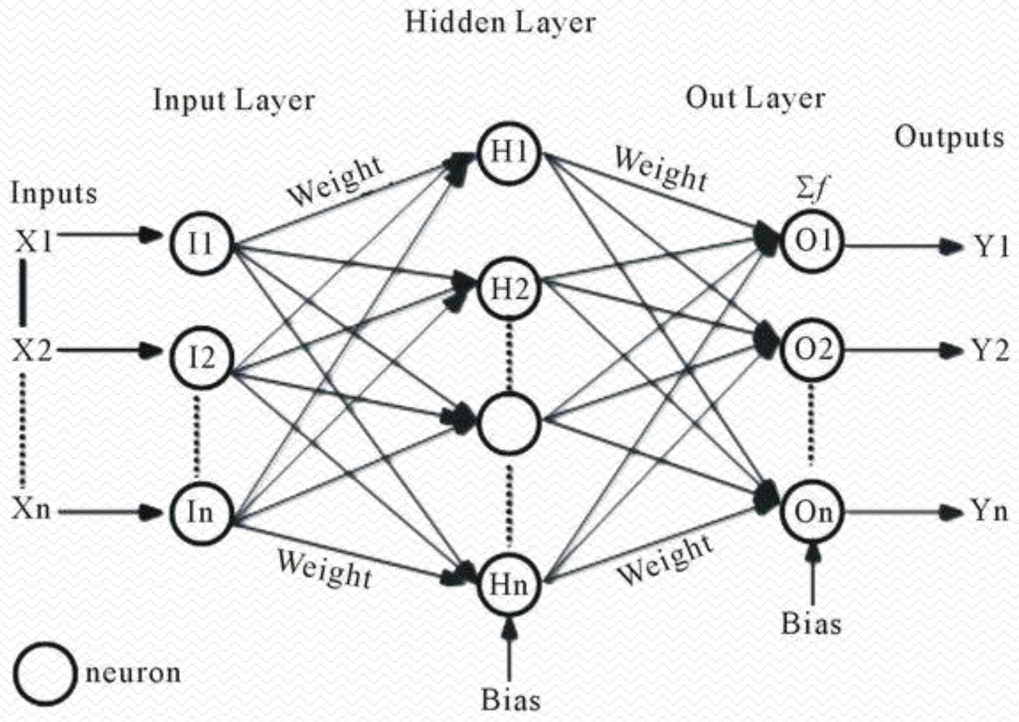
- Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks.
- A **single-layer perceptron model** is the simplest type of artificial neural network.
- A single-layer perceptron consists of only one layer of artificial neurons (perceptrons).
- It is limited to solving linearly separable problems, meaning it can only learn and represent linear decision boundaries.
- Single-layer perceptrons use a simple threshold activation function.
- They are not capable of approximating complex non-linear functions.



Types of Perceptron

Multilayer perceptron:

- An MLP consists of one input layer, one or more hidden layers, and one output layer.
- Each neuron in the hidden layers and output layer is typically a perceptron.
- MLPs can solve non-linear problems because they can approximate complex functions by combining multiple linear transformations with non-linear activation functions.
- They use activation functions like sigmoid, tanh, ReLU, etc., which introduce non-linearity into the network.



Neural Network Representation

- **Input Layer:**

  The input layer takes raw input from the domain.

  No computation is performed at this layer. Nodes here just pass on the information (features) to the hidden layer.
- **Hidden Layers:**

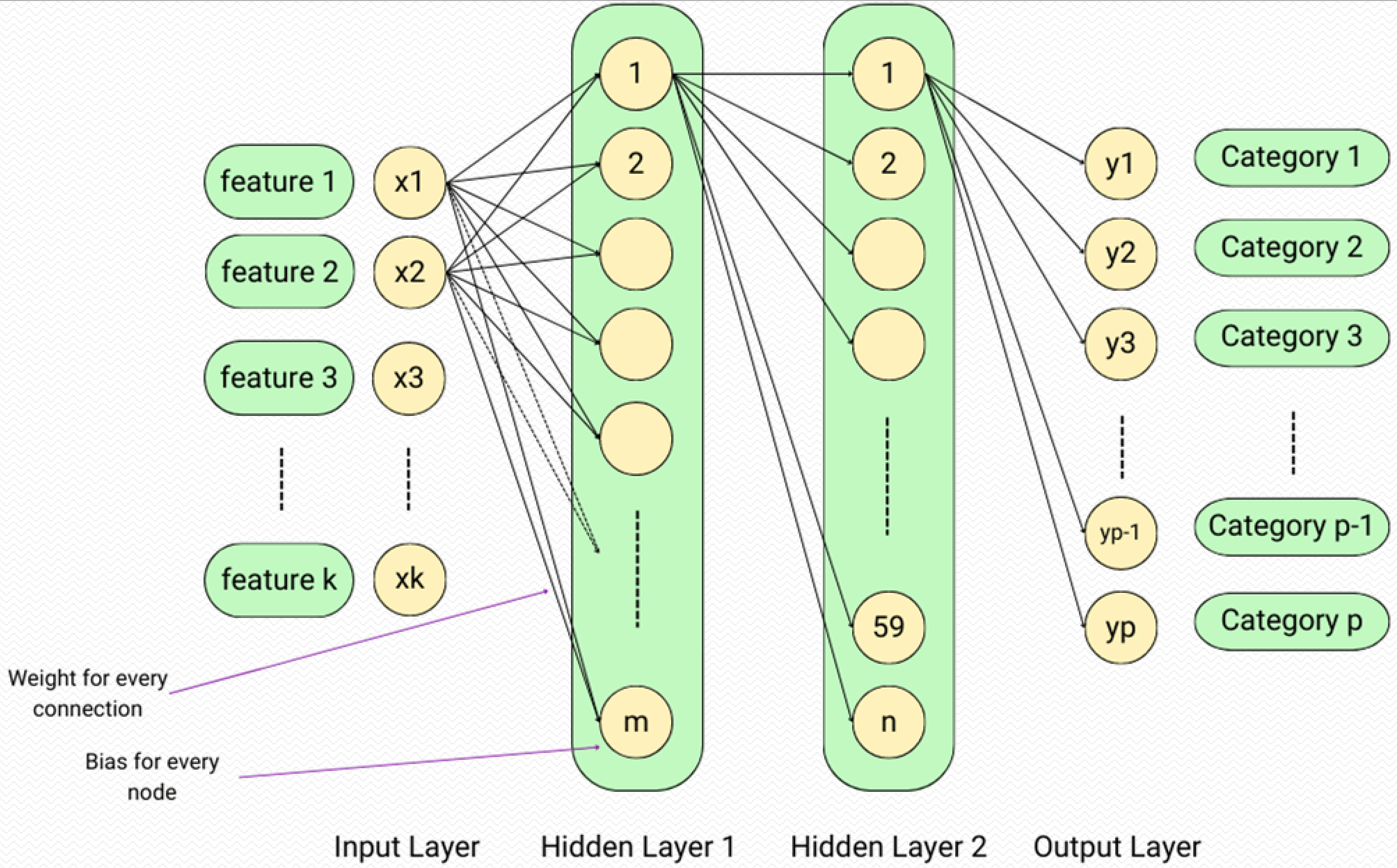
  They provide an abstraction to the neural network.

  The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer.

  There will always be a minimum of one hidden layer in a neural network.
- **Output Layer:**

  It is the final layer of the network that brings the information learned through the hidden layer and delivers the final value as a result.

Neural Network Representation



Neural Network Representation

- Neural network is the fusion of artificial intelligence and brain-inspired design that reshapes modern computing.
- Neural networks mimic the basic functioning of the human brain and are inspired by how the human brain interprets information.

Types of Neural Networks:

- **Artificial Neural Network**:

  ANN is also known as an artificial neural network.

  It is a feed-forward neural network because the inputs are sent in the forward direction.

  It can also contain hidden layers which can make the model even denser.

- **Convolution Neural Network:**

  CNNs is mainly used for Image Data. It is used for Computer Vision.

  It contains a combination of convolutional layers and neurons.

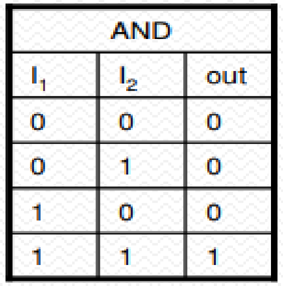
- **Recurrent Neural Network:**

  RNN is used to process and interpret time series data.

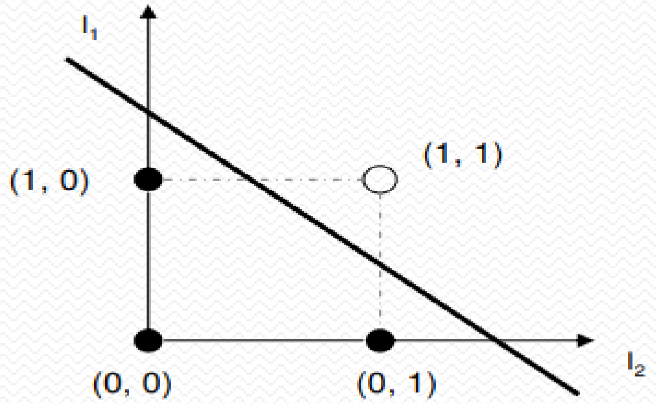
  In this type of model, the output from a processing node is fed back into nodes in the same or previous layers.

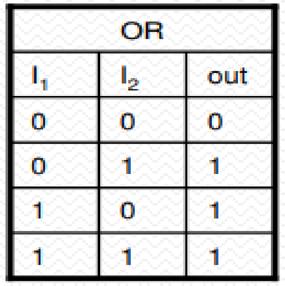
Perceptron - Problems

- Basic perceptron can generalize any kind of linear problem. The both AND and OR Gate problems are linearly separable problems.
- On the other hand, this form cannot generalize non-linear problems such as XOR Gate. Perceptron evolved to multilayer perceptron to solve non-linear problems and deep neural networks were born.

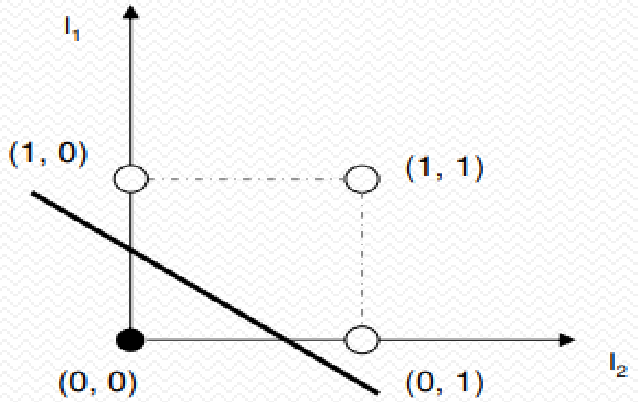


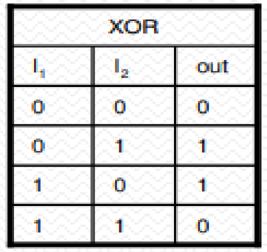
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



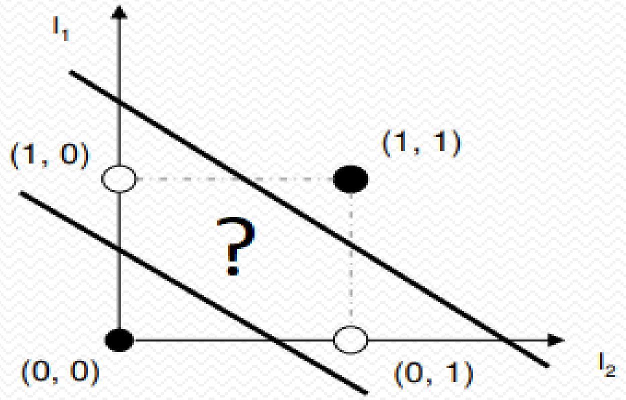


OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1





XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



Activation Functions

Dr. Y. Krishna Bhargavi
Associate Professor
Department of CSE
GRIET

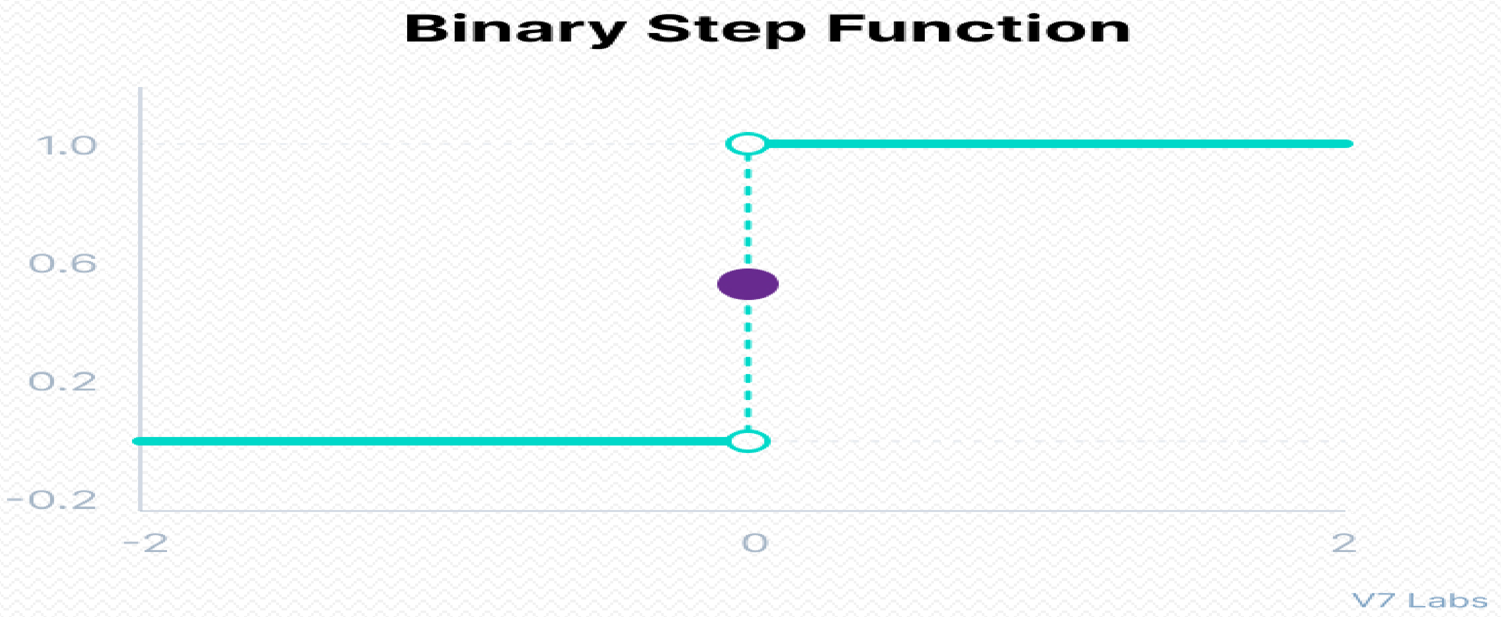
Activation Function

- It is also called as transfer function that performs mathematical computation and maps the resulting values between (0,1) or (-1,1) depending on the activation function.
- Activation function defines the output of input or set of inputs.
- The Activation Functions can be basically divided into 3 types-
  - Binary step function
  - Linear Activation Function
  - Non-linear Activation Functions
    - **Derivative or Differential:** Change in y-axis w.r.t. change in x-axis. It is also known as slope.
    - **Monotonic function:** A function which is either entirely non-increasing or non-decreasing.
    - The Nonlinear Activation Functions are mainly divided on the basis of their range or curves-

Activation Function

Binary step function

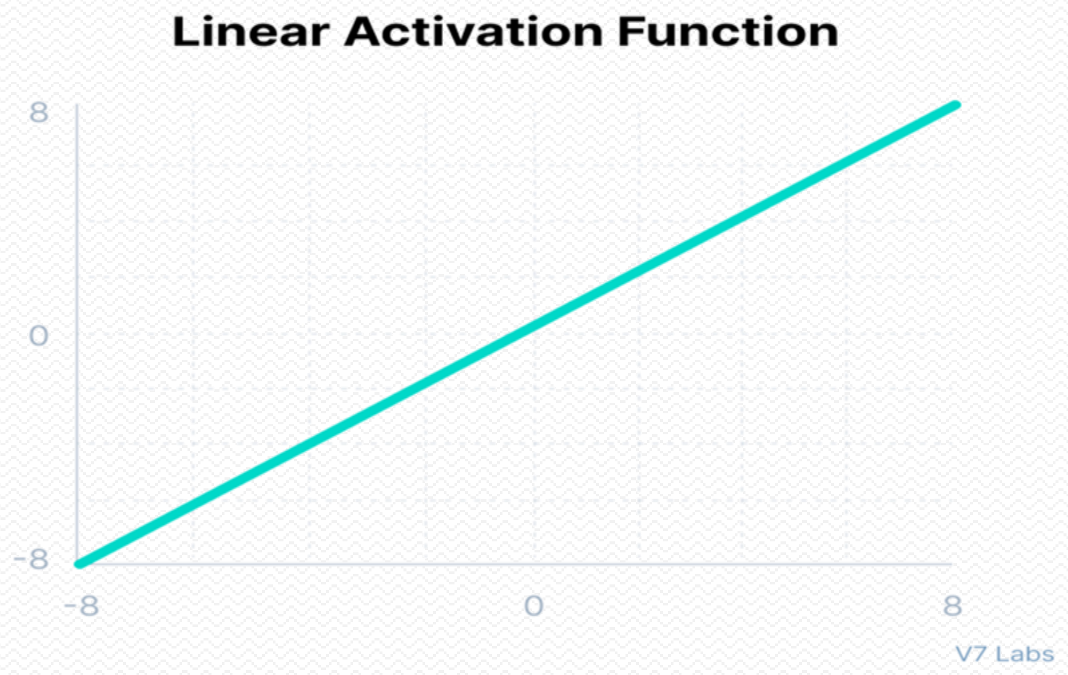
- The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.
- It is very simple and useful to classify binary problems or classifier.
- Range: [-∞,∞]
- Mathematical representation: $f(x) = 1$ if $x >$ threshold
  $= 0$ otherwise



Activation Function

Linear Activation function

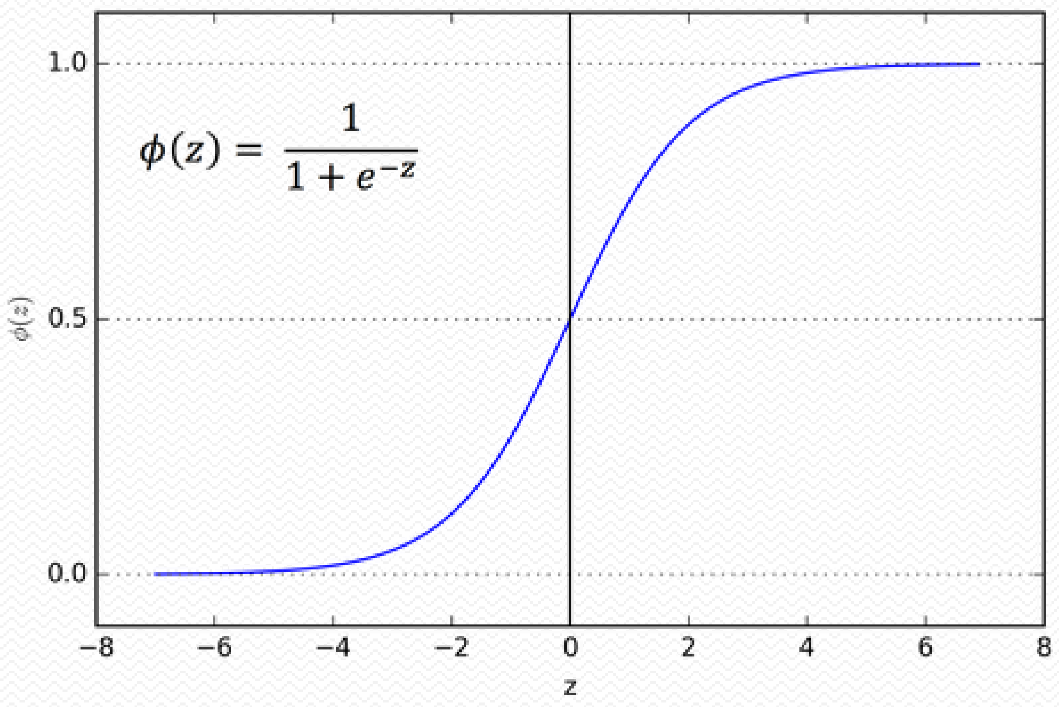
- This function is also known as Identity Function. Here the activation is proportional to the input.
- The linear activation function, also known as "no activation," or "identity function"
- Range: [-∞,∞]
- Mathematical representation: f (x) = x



Activation Functions

Sigmoid Function:

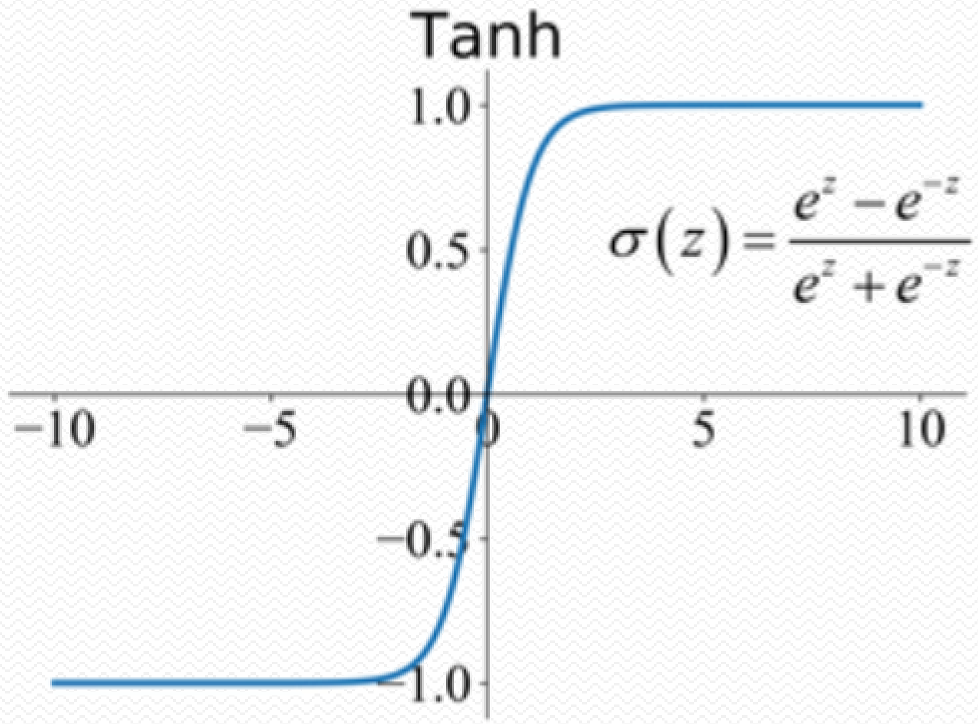
- The Sigmoid Function curve looks like a S-shape. The range of the sigmoid function is from (0 to 1)
- The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points.
- The function is monotonic but function's derivative is not.
- The logistic sigmoid function can cause a neural network to get stuck at the training time.



Activation Functions

tanh or Hyperbolic Tangent Function:

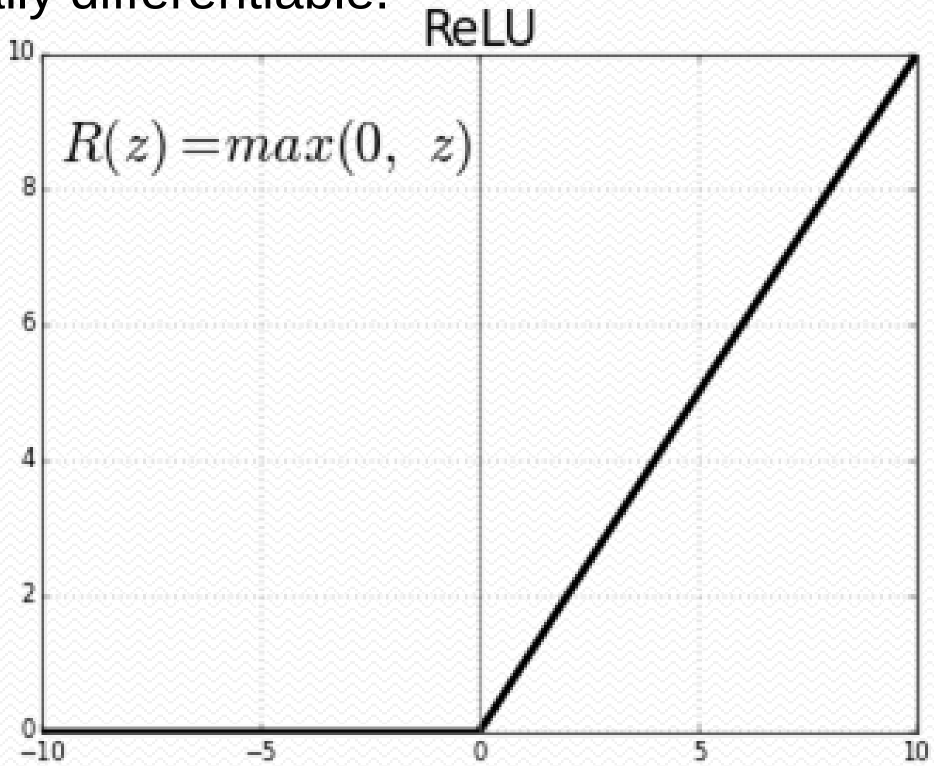
- tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).
- The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.
- The function is differentiable.
- The function is monotonic while its derivative is not monotonic.
- The tanh function is mainly used classification between two classes.
- Both tanh and logistic sigmoid activation functions are used in feed-forward nets.



Activation Functions

ReLU (Rectified Linear Unit):

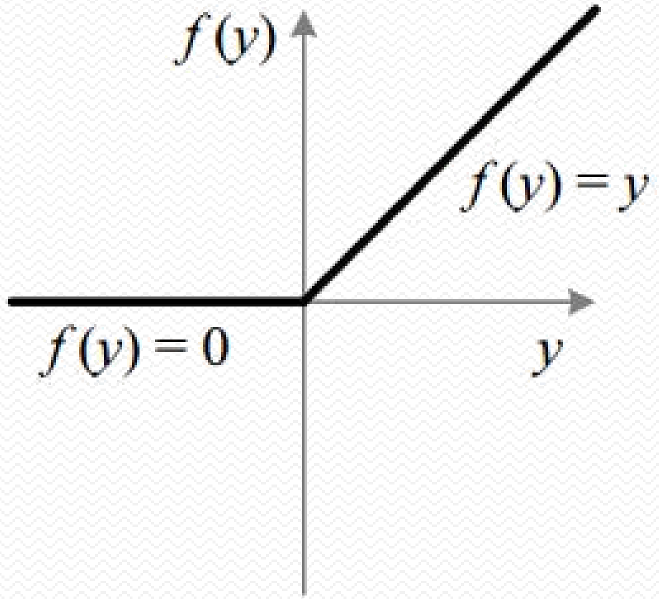
- ReLU is half rectified (from bottom).
- f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
- Range: [0 to infinity)
- The function and its derivative both are monotonic.
- The ReLU (Rectified Linear Unit) function is partially differentiable.

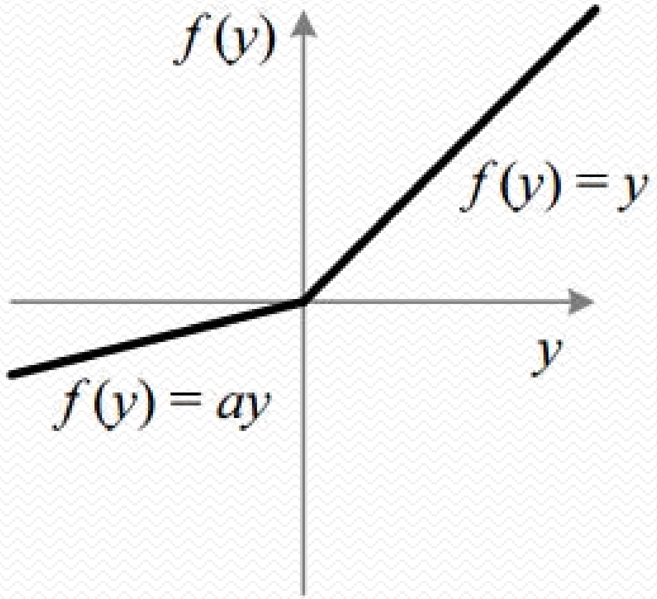


Activation Functions

Leaky ReLU (Rectified Linear Unit):

- The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so.
- When a is not 0.01 then it is called Randomized or parametric ReLU.
- Therefore the range of the Leaky ReLU is (-infinity to infinity).
- Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.





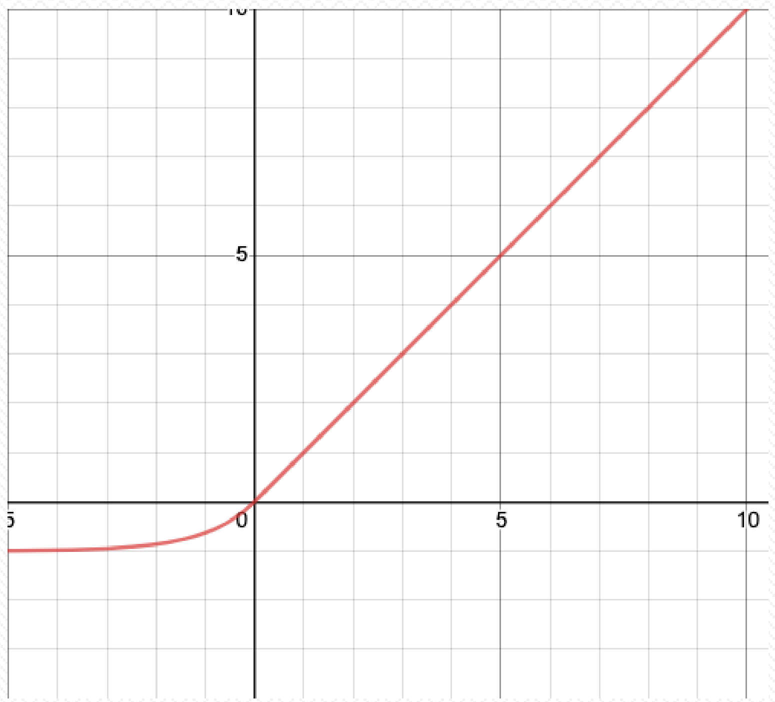
Activation Functions

- **ReLU:** Default choice, efficient, and works well unless facing dead neurons.
- **Leaky ReLU:** If you observe dead neurons or want to ensure some gradient flow for negative inputs, use this variant.
- **Randomized ReLU:** If you're dealing with overfitting or need regularization and want to introduce randomness to your network.

Activation Functions

Exponential Linear Units (ELUs) Function :

- Exponential Linear Unit, or ELU for short, is also a variant of ReLU that modifies the slope of the negative part of the function.
- ELU uses a log curve to define the negative values unlike the leaky ReLU and Parametric ReLU functions with a straight line.



Activation Functions

Softmax Function :

- Softmax is an activation function that scales numbers/logits into probabilities.
- The output of a Softmax is a vector with probabilities of each possible outcome.
- Sigmoid is typically used for binary classification tasks, while softmax is used for multi-class classification tasks.

$$softmax(Z_i) = \frac{\exp\,(Z_i)}{\sum \exp\,(Z_i)}$$

Back propagation Algorithm

Dr. Y. Krishna Bhargavi
Associate Professor
Department of CSE
GRIET

Back propagation Algorithm

- Back propagation is the neural network training process of feeding error rates back through a neural network to make it more accurate.
- Back propagation is a process involved in training a neural network.
- It takes the error rate of a forward propagation and feeds this loss backward through the neural network layers to fine-tune the weights.

Algorithm:

1. Take the input values and initialize the weights and biases.
2. Propagate the inputs forward.

Compute the NetInput and output of each unit in the hidden and output layers.

$NetInput(I_j) = \sum_i W_{ij} O_i + \theta_j$

W_{ij} : Weight of the connection from unit i in the previous layer to unit j

O_i : Output of unit i from previous layer

θ_j : Bias of the unit j

Apply activation function to the $NetInput(I_j)$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

Back propagation Algorithm

Algorithm:

3. Back propagate the error

Error at Output Layer

$$Err_j = O_j\left(1 - O_j\right)(T_j\text{-}O_j)$$

Err_j : Error for unit j in the output layer

O_j : Actual Output

T_j : Target Output

$O_j\left(1 - O_j\right)$: Derivative of the logistic function

Error at Hidden Layer

$$Err_j = O_j\left(1 - O_j\right)\sum_k Err_k W_{jk}$$

Err_k : Error of Unit k

W_{jk} : Weight of connection from unit j to a unit k in the next hidden layer

Back propagation Algorithm

Algorithm:

3. Back propagate the error

Update the weights and biases to reflect the propagated errors

$$W_{ij}' = l * Err_j O_i$$

$$W_{ij} = W_{ij} + W_{ij}'$$

l : Learning rate that generally ranges between 0 to 1.

$$l = \frac{1}{t}$$

t : Number of iterations through the training set

$$\theta_j' = l * Err_j$$

$$\theta_j = \theta_j + \theta_j'$$

4. Terminating Condition

- All W_{ij}' in previous epoch are so small as to be below the threshold
- Pre-specified number of epochs has expired
- Percentage of misclassified data in previous epoch is below threshold value

Back propagation Algorithm

Pros:

- No previous knowledge of a neural network is needed, making it easy to implement.
- It’s straightforward to program since there are no other parameters besides the inputs.
- It doesn’t need to learn the features of a function, speeding up the process.
- The model is flexible because of its simplicity and applicable to many scenarios.

Cons:

- Training data can impact the performance of the model, so high-quality data is essential.
- Noisy data can also effect back propagation, potentially tainting its results.
- It can take a while to train back propagation models and get them up to speed.
- Back propagation requires a matrix-based approach, which can lead to other issues.

THANK YOU

Convolutional Neural Networks

- A Convolutional Neural Network (CNN), also known as ConvNet, is a specialized type of deep learning algorithm mainly designed for tasks that necessitate object recognition, including image classification, detection, and segmentation.

Importance of CNN:

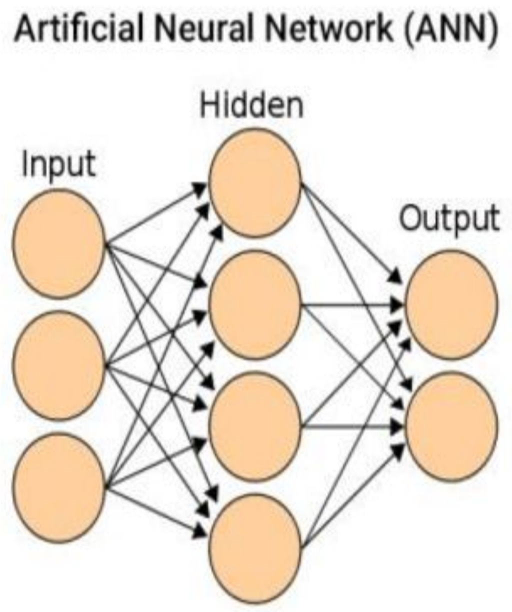
- Autonomous Feature Extraction
- Translation Invariant Characteristics
- Pre-trained CNN variants
- Wide range of applications

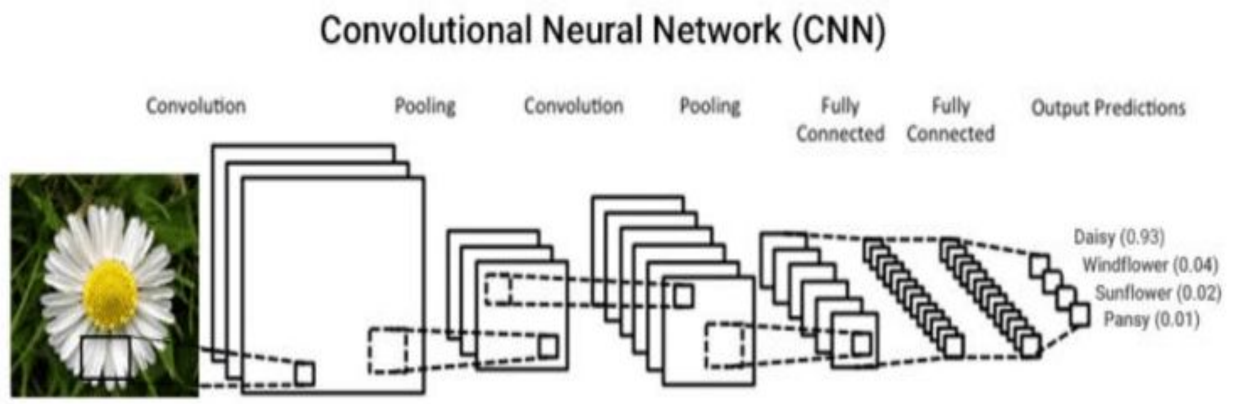
ANN vs CNN

ANN:

CNN: In CNN, the number of parameters/weights is independent of the size of the image. It depends on the filter size

1. **Sparsity of connections:**
2. **Parameter sharing:**





Convolutional Neural Network(CNN)

Drawbacks of ANN:

- In ANN, all the layers are fully connected layers
- In ANN, millions of parameters/weights
- Increases computational complexity

Advantages of CNN:

- Sparsity of connections:

  The weights of most of the connections are zeros
- Parameter sharing:

  Parameters/weights are shared between input and output nodes

Applications of CNN

Digit Recognition

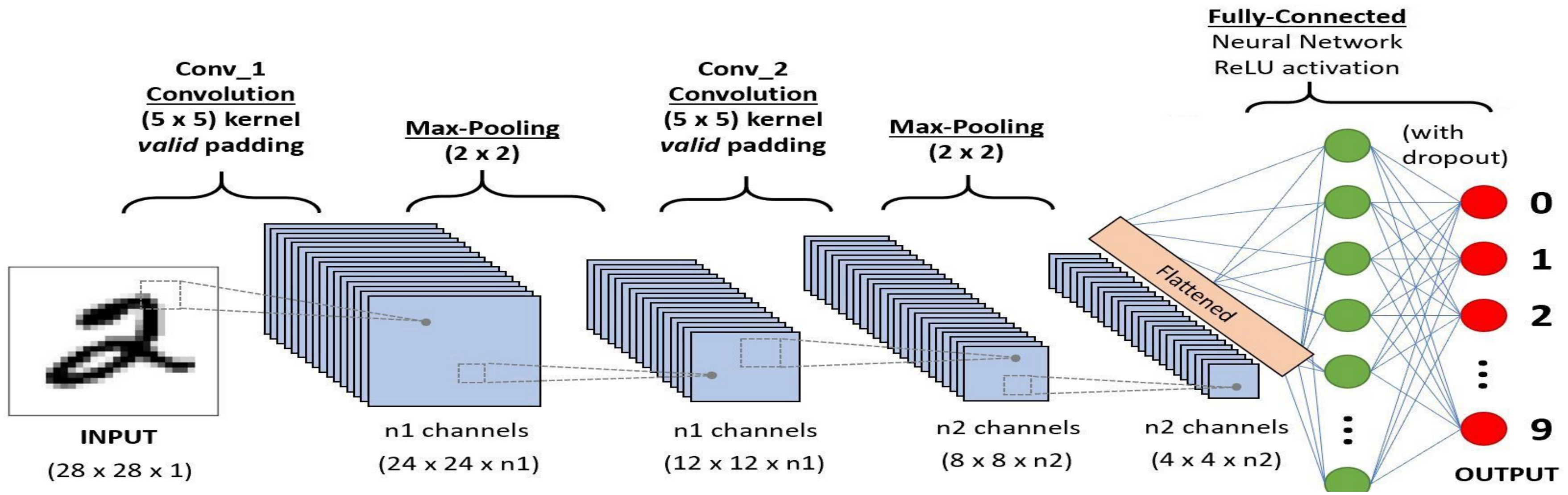
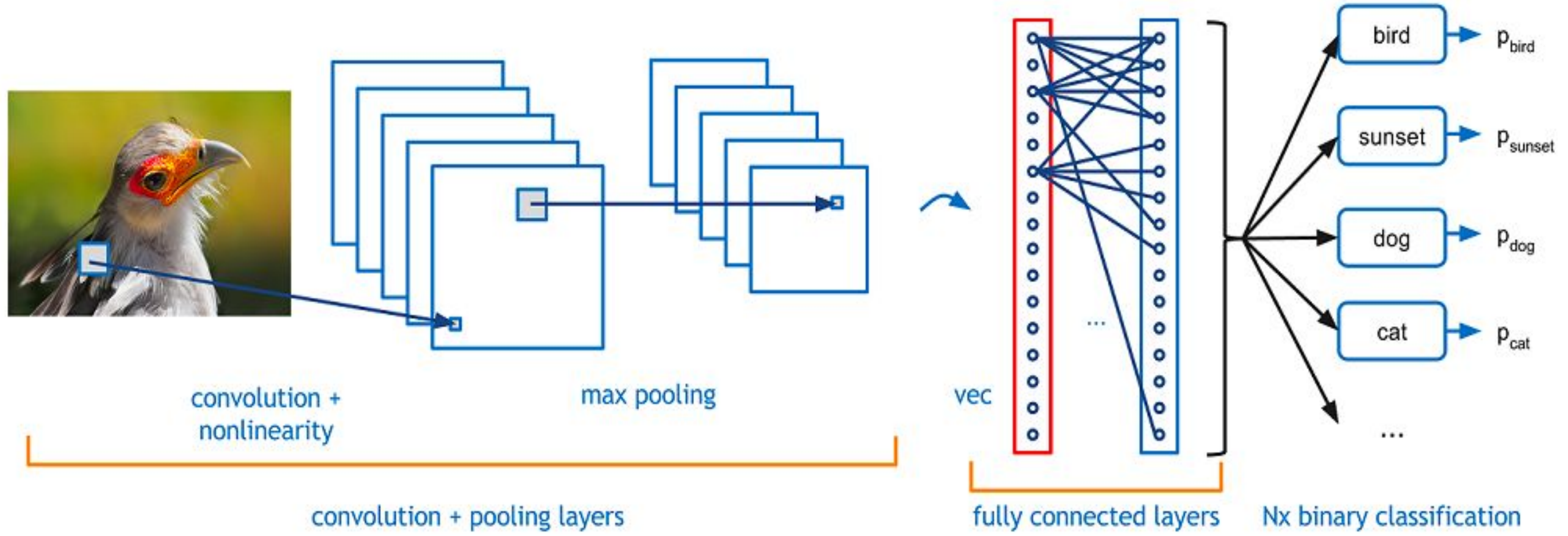


Image Classification



Convolutional Neural Networks

Types of Layers :

- Convolution layer
- Pooling layer
- Flattening layer
- Fully connected layer

1. Convolution layer (Convolution operation)

Types of Convolution:

- l-Dimensional Convolution

  input: signal

- 2-Dimensional Convolution

  input: gray-scale Image

- 3-Dimensional Convolution

  input: color Image

Convolution Layer

Convolution operation: hyper parameters

1. **Filter/Kernel :** it's a weights matrix

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

2. **Stride:** It is the **number of pixels shifts over the input matrix**

   when **stride is 1,** then we **move the filter to 1 pixel** at a time

   when **stride is 2,** then we **move the filter to 2 pixels** at a time etc

3. **Padding:** It is the process of **adding layers of zeros** to input matrix

4. **No.of filters:** for 3D input

Convolution Layer

Convolution Operation: Hyper Parameters

I) **Filter/Kernel matrix (Weights matrix)**

II) **Stride – 1,2,3,4,…**
(default stride is 1)

III) **Padding – 0,1,2,3,…**
(default padding is 0)

IV) **No. of filters / No. of Kernels (for 3D input)**

For 1-Dimensional (signal): Parameters I,II are used
For 2-Dimensional (gray image) : Parameters I, II, III are used
For 2-Dimensional (color image) : Parameters I,II, III, IV are used

1 - Dimensional Convolution

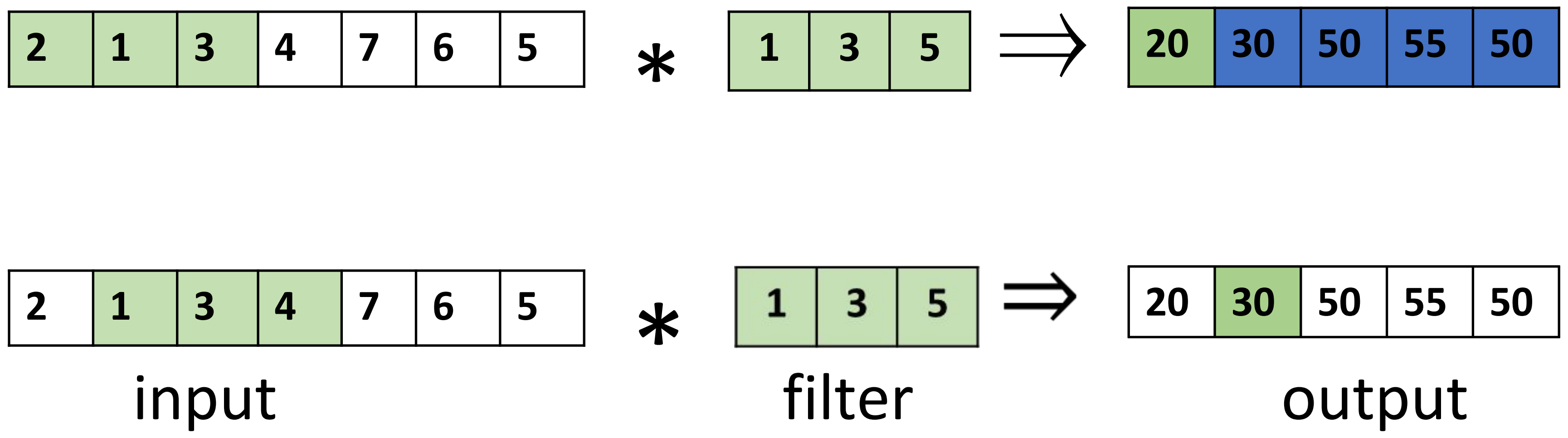
Convolution Operation : Signal (1 D)

Convolutional Operation	Gray Scale image	Example
Input size (1D)	n	7
filter size	f	3
stride	s = 1, 2, 3, …	s = 1 (default)
Output size (1D)	((n - f)/ s) + 1	o/p : (7 - 3) + 1 = 5

1-Dimensional convolution

Example

- Input - 1D
- Filter size -1D
- Stride – 1 (default)
- Output- 1D



2-Dimensional Convolution

Gray Scale Image (1 Channel)

3	0	1	2	7	4
1	5	8	9	3	1
2	7	23	50	250	3
5	55	34	3	1	89
67	45	4	56	34	23
17	13	17	20	23	16

6X6 Matrix

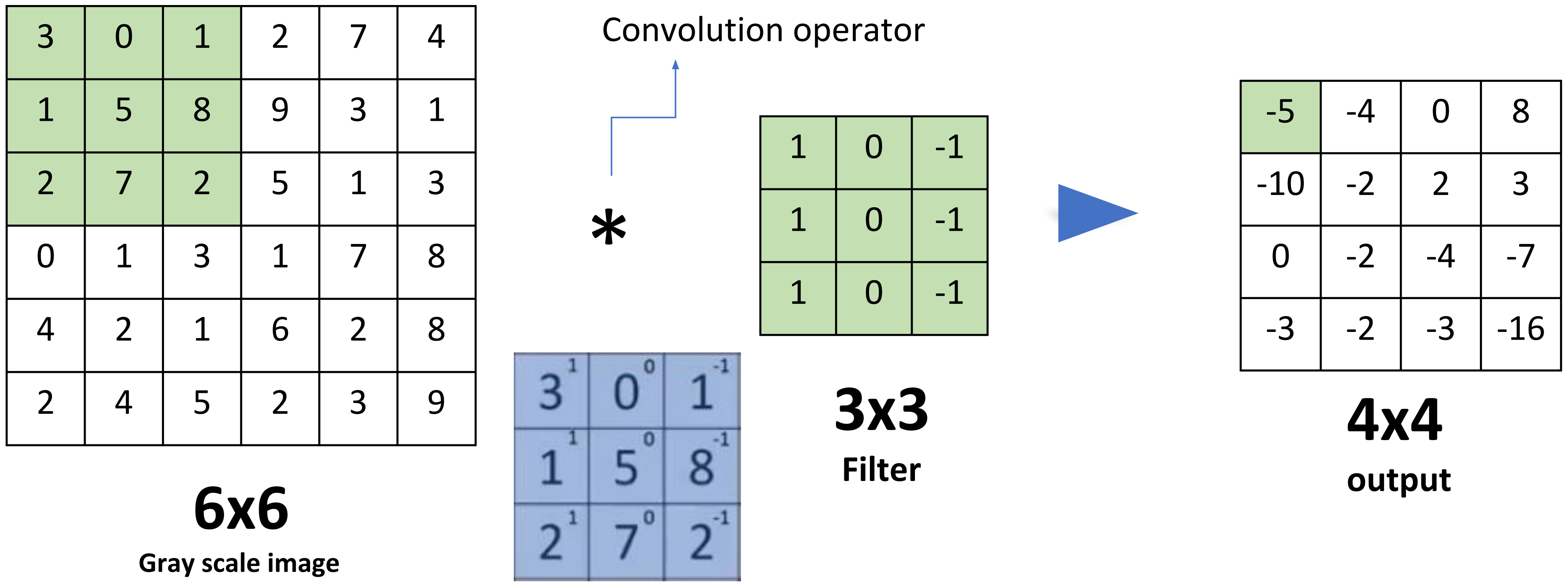
Each Pixel value is in the range of 0-255

2-Dimensional Convolution

Convolution Operation : Gray Scale image (2D)

Convolutional operation	Gray Scale image	Example
input size (2D)	n_x x n_y	6 x 6
filter size	f x f	3 x 3
stride	s = 1,2,3, …	s = 1 (default)
padding	p = 0, 1, 2,…	p = 0 (default)
output size (2D)	$(n_x + 2p - f)/s + 1$ x $(n_y + 2p - f)/s + 1$	((6+2*0-3)/1+1) x (6+2*0-3)/1+1)) o/p matrix - 4 x 4

Convolution Operation - Gray Scale Image



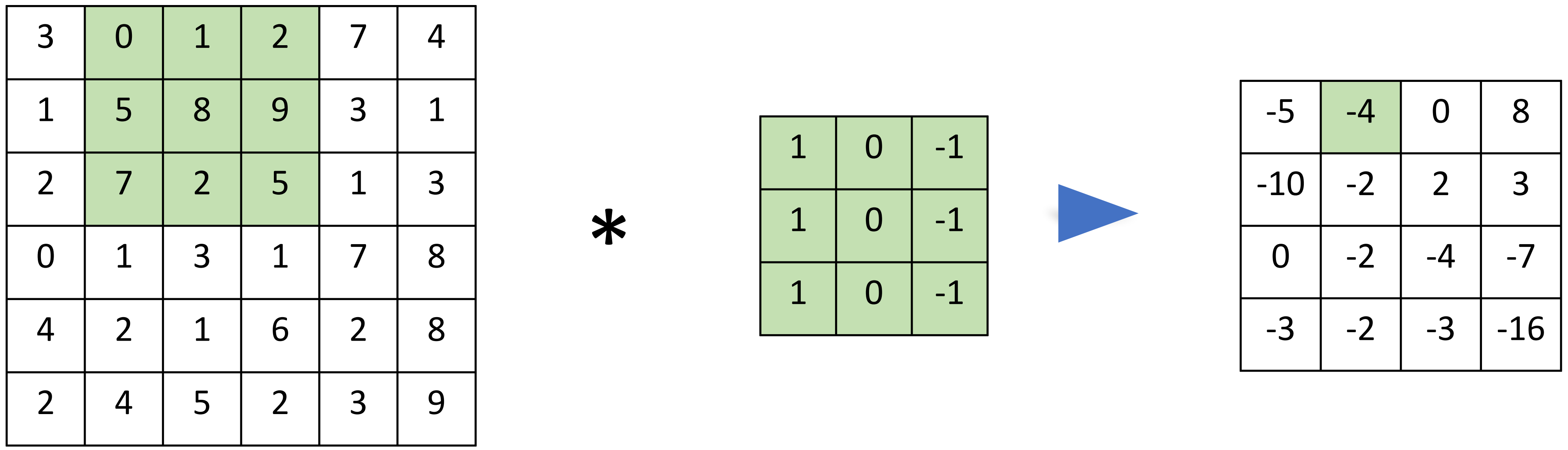
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

3^{1}	0^{0}	1^{-1}
1^{1}	5^{0}	8^{-1}
2^{1}	7^{0}	2^{-1}

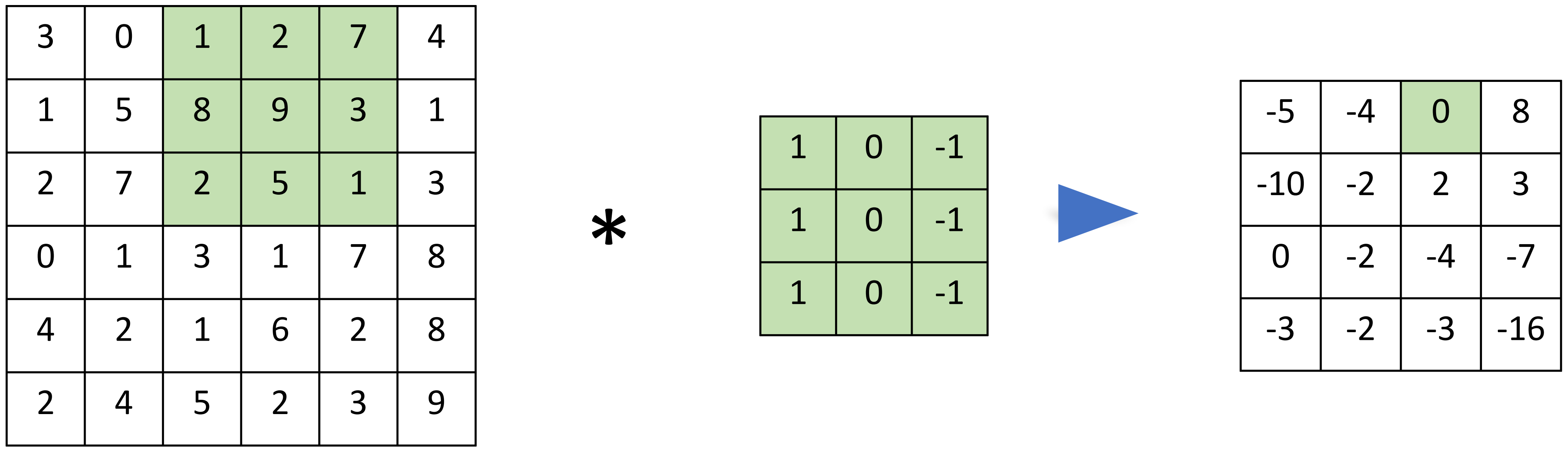
1	0	-1
1	0	-1
1	0	-1

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

So, we take the first 3 X 3 matrix from the 6 X 6 image and multiply it with the filter. Now, the first element of the 4 X 4 output will be the **sum of the element-wise product of these values**, i.e. 3*1 + 0 + 1*-1 + 1*1 + 5*0 + 8*-1 + 2*1 + 7*0 + 2*-1 = -5. To calculate the second element, we will **shift our filter one step towards the right** and again get the **sum of the element-wise product**



$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$



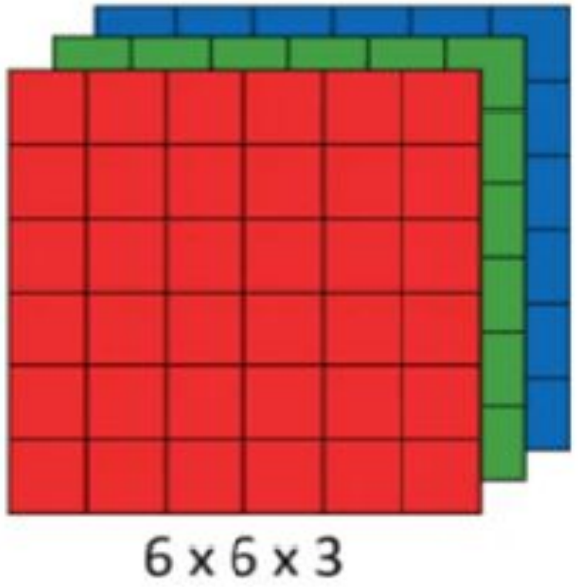
$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 \\ \hline
1 & 5 & 8 & 9 & 3 & 1 \\ \hline
2 & 7 & 2 & 5 & 1 & 3 \\ \hline
0 & 1 & 3 & 1 & 7 & 8 \\ \hline
4 & 2 & 1 & 6 & 2 & 8 \\ \hline
2 & 4 & 5 & 2 & 3 & 9 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
\end{array}
\;\blacktriangleright\;
\begin{array}{|c|c|c|c|}
\hline
-5 & -4 & 0 & 8 \\ \hline
-10 & -2 & 2 & 3 \\ \hline
0 & -2 & -4 & -7 \\ \hline
-3 & -2 & -3 & -16 \\ \hline
\end{array}
$$

3-Dimensional convolution

Colour Image (3 Channels)



Each Pixel is in the range of 0-255

3- Dimensional Convolution

Convolution Operation : Color image

Convolution operation	Color image	Example
input size (3D)	n_x x n_y x n_{c}, n_c is the number of channels	6 x 6 x 3
filter size (3D)	f x f x n_c	3 x 3 x 3
stride	s = 1, 2, 3,…	s = 1 (default)
padding	p = 0, 1, 2,…	p = 0 (default)
no. of filters	n_c' = 2,3,4,5,…	--
output size (3D)	((n_x + 2p - f)/s + 1) x ((n_y + 2p - f)/s + 1) x n_c'	((6+2*0-3)/1+1) x (6+2*0-3)/1+1)) x 3 o/p matrix - 4 x 4 x 3

Convolution operation - Color image

input : 6 X 6 X 3

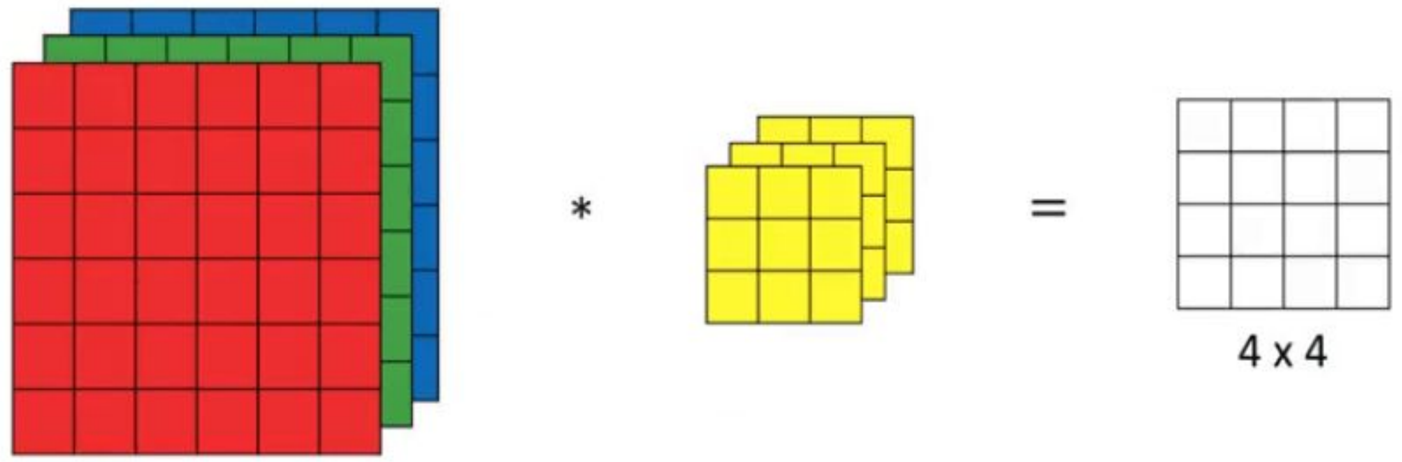
filter : 3 X 3 X 3

output : 4 X 4

input : 3D

filter : 3D

output : 2D



Keep in mind that the number of channels in the input and filter should be same.

Convolution operation - Color image

No.of channels in the input =

No.of channels in the filter = 3

After convolution, the output shape is a 4 X 4 matrix

Instead of using just a single filter, we can use multiple filters as well, hence output dimension will change.

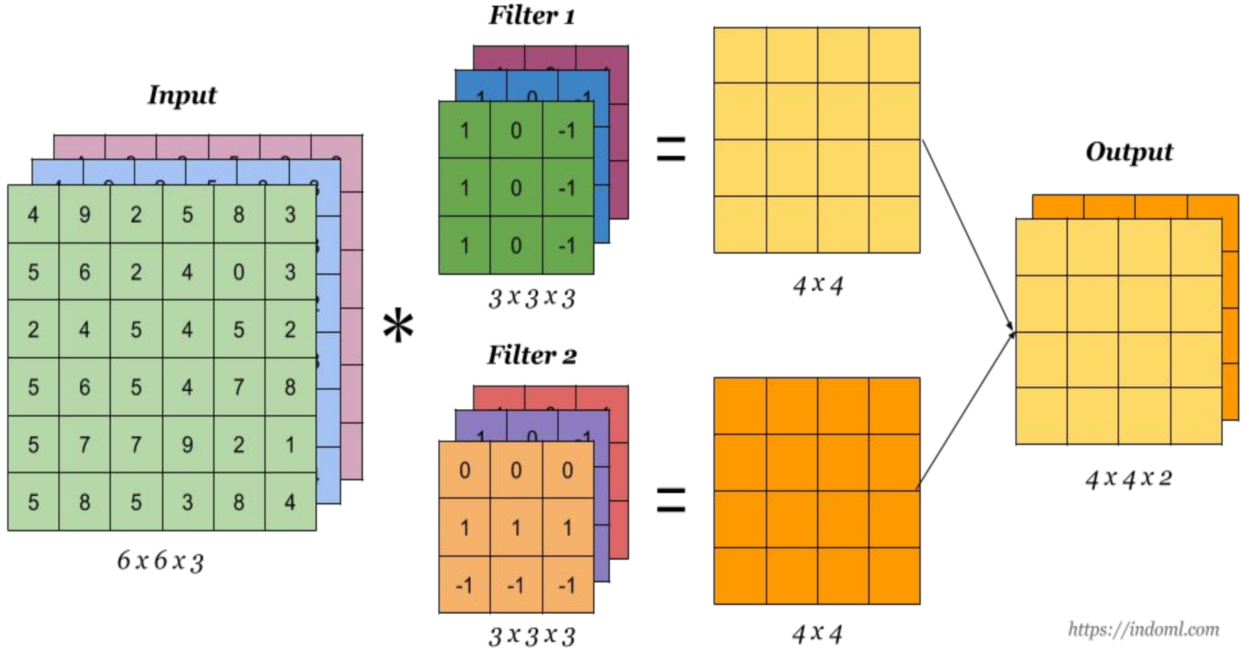
input size: 6x6x3 (3D)

stride: 1

padding: 0

no. of filters: 2

output size: 4x4x~~3~~ (3D)



Padding

- when we apply convolutional operation for an n x n input with a filter size of f x f the out put is (n-f+1) x (n-f+1), the image is shrinked.
- Also, the **pixels in corner are ignored compared to central pixels-information loss.**
- To overcome these issues, we can use **padding**,

  i.e., **we add one pixel all around the edges**. This means that the input will be an 8 X 8 matrix (instead of a 6 X 6 matrix). Applying convolution of 3 x 3 on it will result in a 6 x 6 matrix which is the original shape of the image

Input: n X n

Padding: p = 1

Filter size: f X f

Output: (n+2p-f+1) X (n+2p-f+1)

0	**0**	**0**	**0**	**0**
0	0	1	2	**0**
0	3	4	5	**0**
0	6	7	8	**0**
0	**0**	**0**	**0**	**0**

Padding

There are two common choices for padding:

Valid: It means **no padding**. If we use **valid padding**, the **output** will be **(n-f+1) X (n-f+1)**

Same: we apply **padding** so that the **output size is same as the input size**,

i.e., **n+2p-f+1 = n, So p = (f-1)/2**

here **p= (3-1)/2 = 1**

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	3	0	1	2	7	4	**0**
0	1	5	8	9	3	1	**0**
0	2	7	2	5	1	3	**0**
0	0	1	3	1	7	8	**0**
0	4	2	1	6	2	8	**0**
0	2	4	5	2	3	9	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

*

1	0	-1
1	0	-1
1	0	-1

→

-5	-5	-6	-1	6	10
-12	-5	-4	0	8	11
-13	-10	-2	2	3	11
-10	0	-2	-4	-7	10
-7	-3	-2	-3	-16	12
-6	0	-2	1	-9	5

Why Convolutions?

- **Convolutions reduce the number of parameters and speed up the training of the model significantly**
- For example **14 million parameters in a fully connected layer** can be reduced to just **156 parameters in case of convolutional layer**

Advantages of convolutional layers over fully connected layers:

1. **Parameter sharing:** In convolutions, a single filter is convolved over the entire input. Due to this, the parameters are shared between input and output nodes
2. **Sparsity of connections:** For each layer, each output value depends on a small number of inputs, instead taking account all the inputs ie the weights of most of the connections are zeros

2. Pooling layer (Pooling operation)

Pooling operation: Reduce the **size of the input**

advantage: **speed up the computation**

Types of Pooling:

1. **Max pooling:** Maximum element in the pooling window is selected
2. **Min pooling:** Minimum element in the pooling window is selected
3. **Avg pooling:** Average of all the elements of the pooling window is selected

Generally we use max pooling or avg pooling

Hyperparameters for pooling operation:

1. Filter size (elements are not required)
2. Stride
3. Max pooling or average pooling

2. Pooling layer

- The **pooling layer** will always **reduce** the size of each feature map by a **factor of 2.**

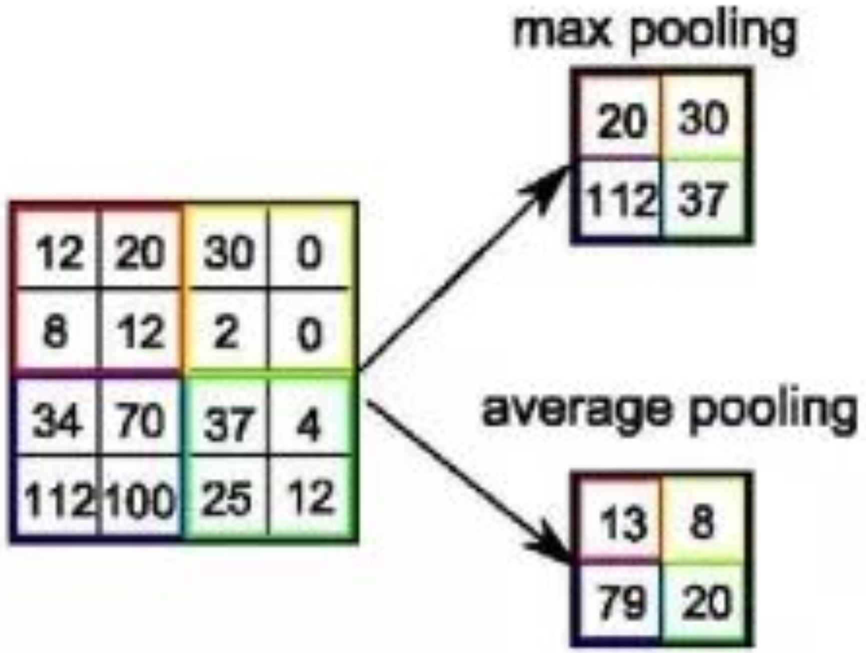
  Ex 1: If size of input is **4x4** then after applying pooling operation, size of output is **2x2**

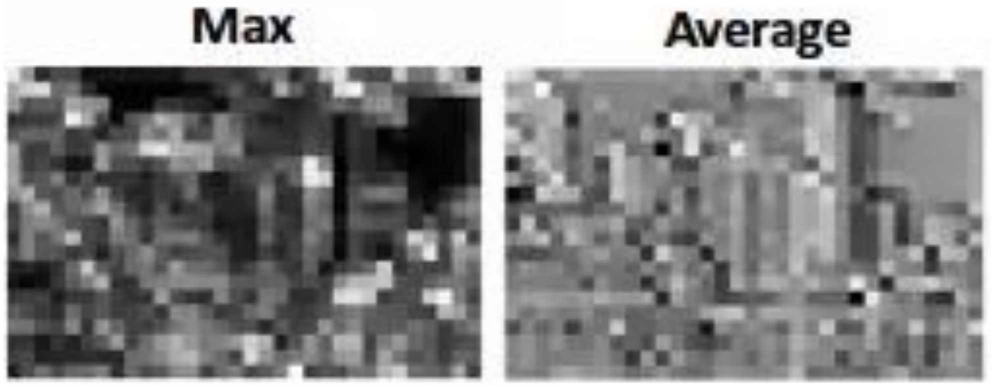
  Ex 2: If size of input is **6x6** then after applying pooling operation, size of output is **3x3**

Hyperparameters:

1. Filter size -2x2
2. Stride - 2
3. Max pooling or average pooling

Note: Max pooling preserves the important features



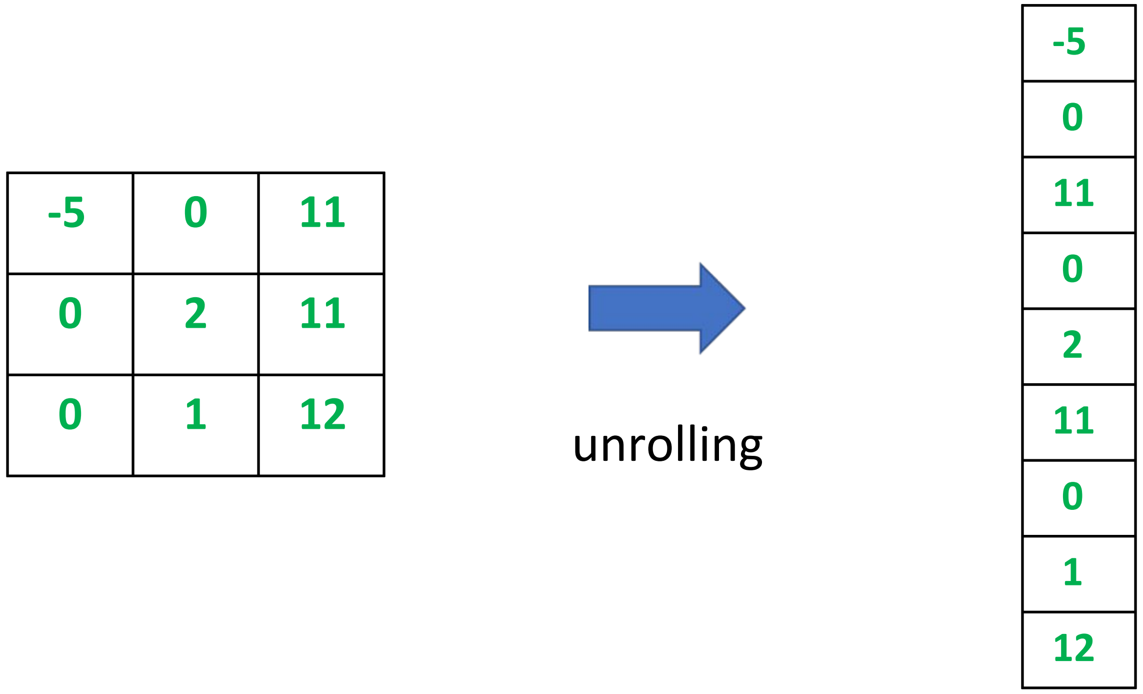


3. Flattening Layer

Fully connected input layer (flatten): Takes the output of the previous layer, ”flattens” it and turns it into a single vector that can be an input for the next stage.

Flattening(Unrolling): Conversion of matrix in to vector

Ex:

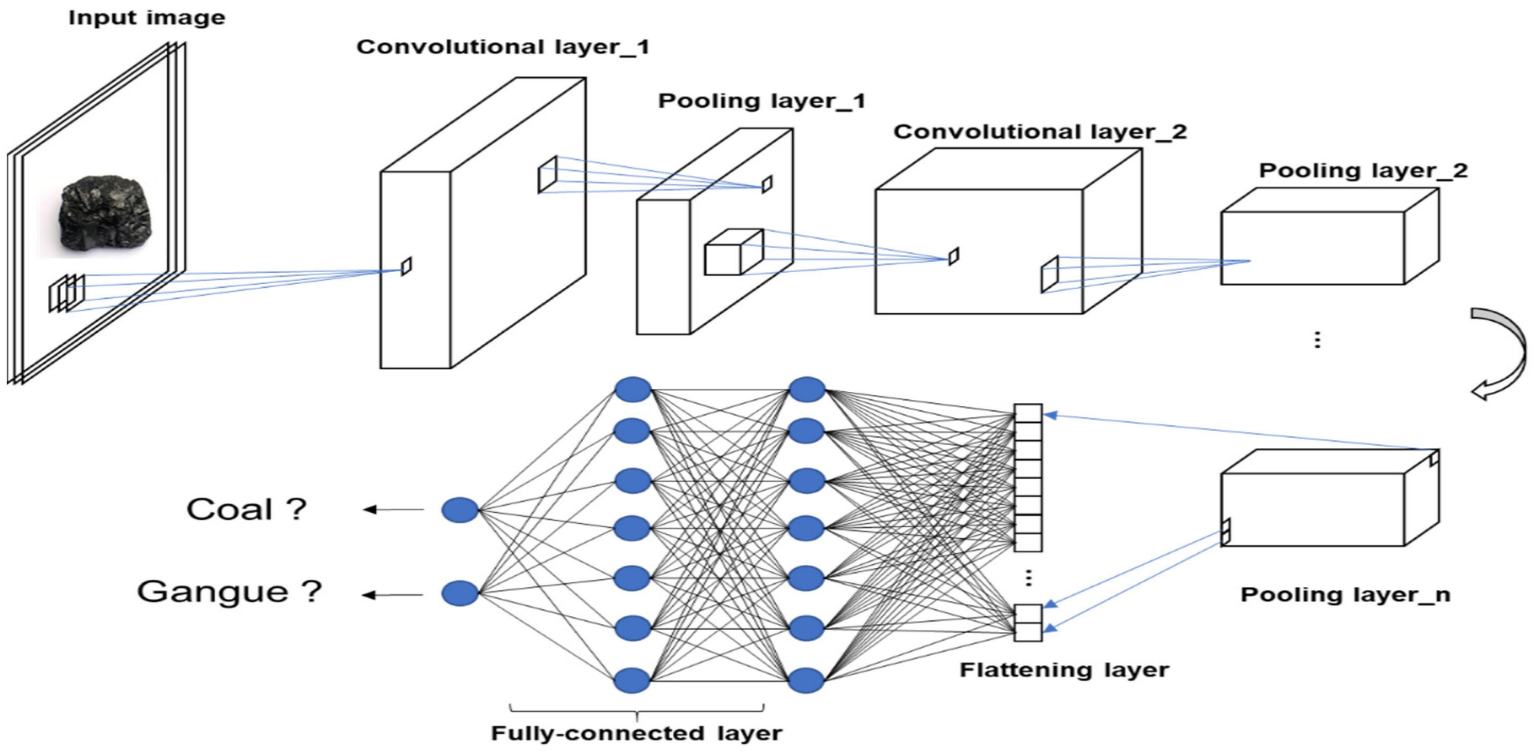


-5	0	11
0	2	11
0	1	12

-5
0
11
0
2
11
0
1
12

Convolutional Neural Network

- **The first fully connected layer** takes the inputs from the feature analysis and applies weights to predict the correct label.
- **Fully connected output layer** gives the final probabilities for each label.



Convolutional Neural Network

Order to be followed

1.Convolutional Layer

2.Pooling Layer (optional)

3.Flattening(unrolling)

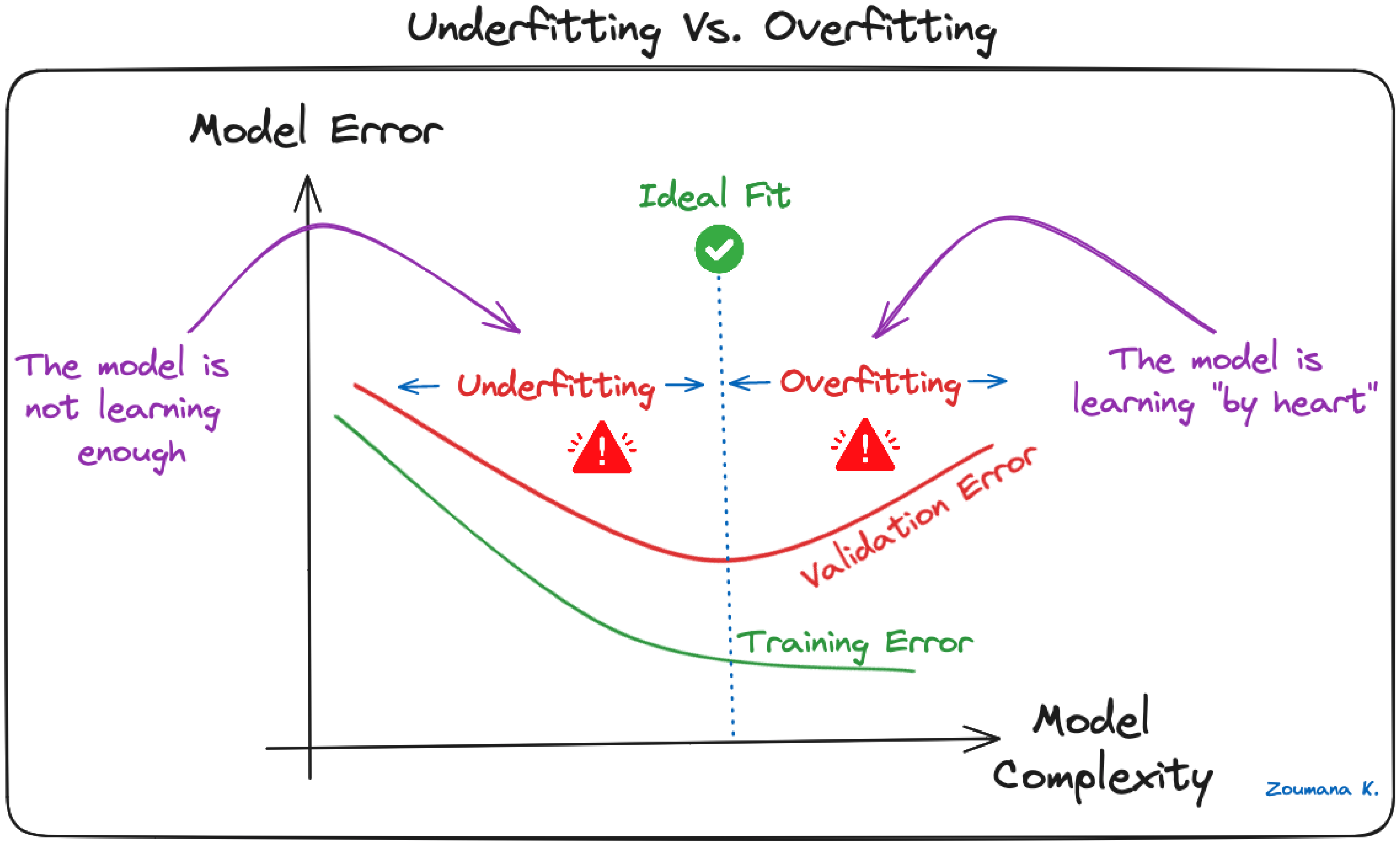
4.Fully connected layers

Note 1: The first layer must be convolutional layer

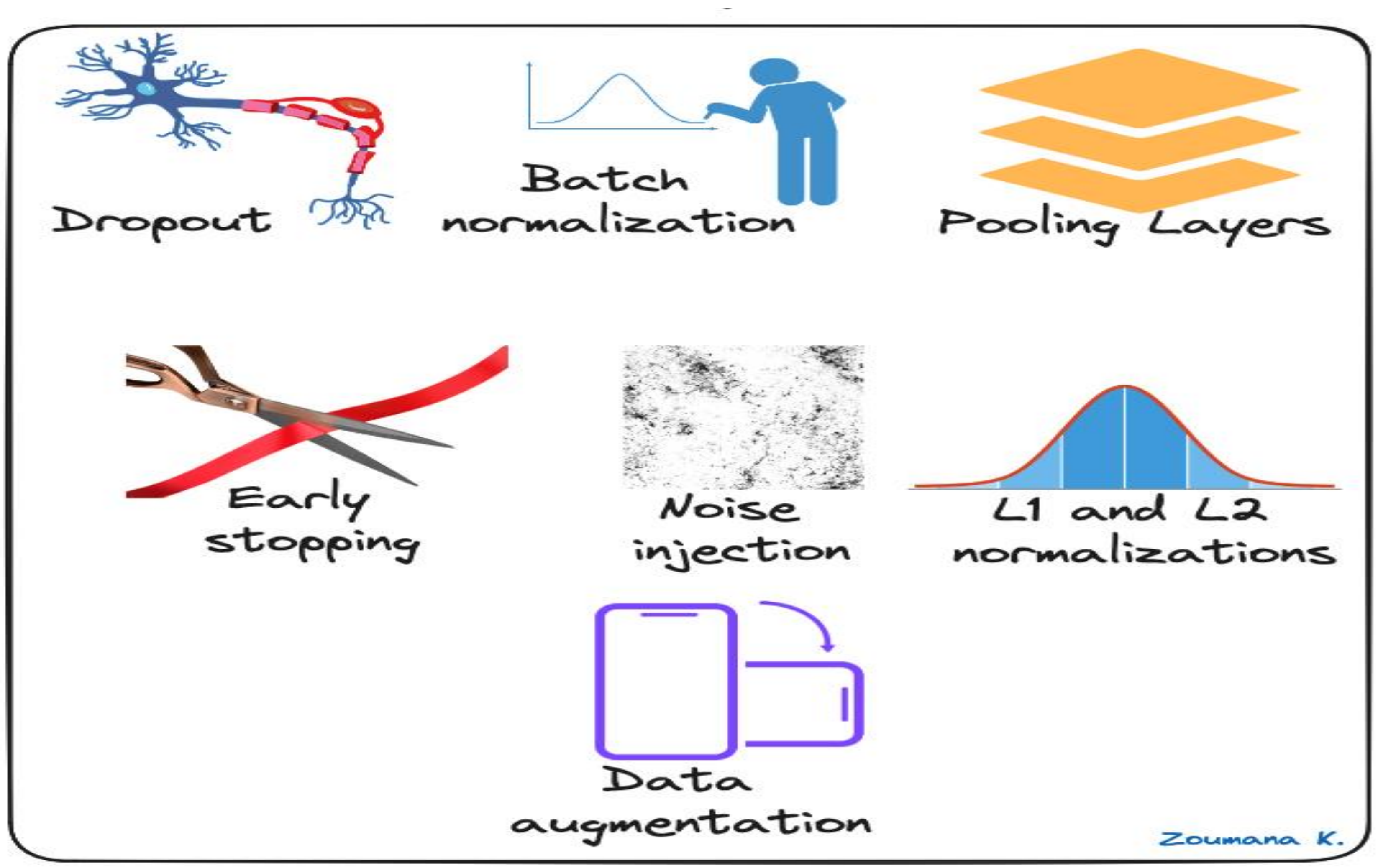
Note 2: At the end we can take any no.of fully connected layers

Overfitting and Regularization in CNN

- Overfitting is a common challenge in machine learning models and CNN.



Strategies to mitigate overfitting in CNN



Strategies to mitigate overfitting in CNN

- **Dropout:** This consists of randomly dropping some neurons during the training process, which forces the remaining neurons to learn new features from the input data.
- **Batch normalization:** The overfitting is reduced at some extent by normalizing the input layer by adjusting and scaling the activations. This approach is also used to speed up and stabilize the training process.
- **Pooling Layers:** This can be used to reduce the spatial dimensions of the input image to provide the model with an abstracted form of representation, hence reducing the chance of overfitting.
- **Early stopping:** This consists of consistently monitoring the model's performance on validation data during the training process and stopping the training whenever the validation error does not improve anymore.

Strategies to mitigate overfitting in CNN

- **Noise injection:** This process consists of adding noise to the inputs or the outputs of hidden layers during the training to make the model more robust and prevent it from a weak generalization.
- **L1 and L2 normalizations:** Both L1 and L2 are used to add a penalty to the loss function based on the size of weights. More specifically, L1 encourages the weights to be spare, leading to better feature selection. On the other hand, L2 (also called weight decay) encourages the weights to be small, preventing them from having too much influence on the predictions.
- **Data augmentation:** This is the process of artificially increasing the size and diversity of the training dataset by applying random transformations like rotation, scaling, flipping, or cropping to the input images.

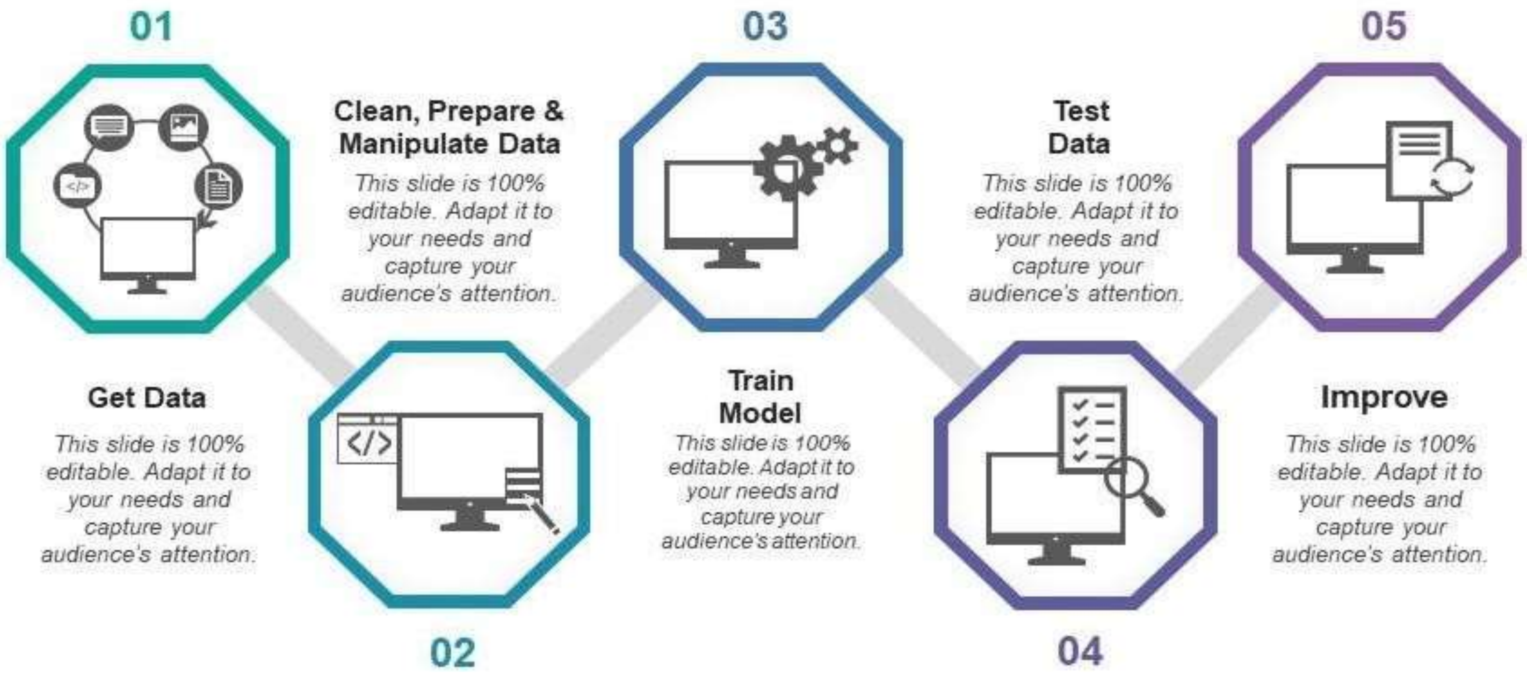
THANK YOU

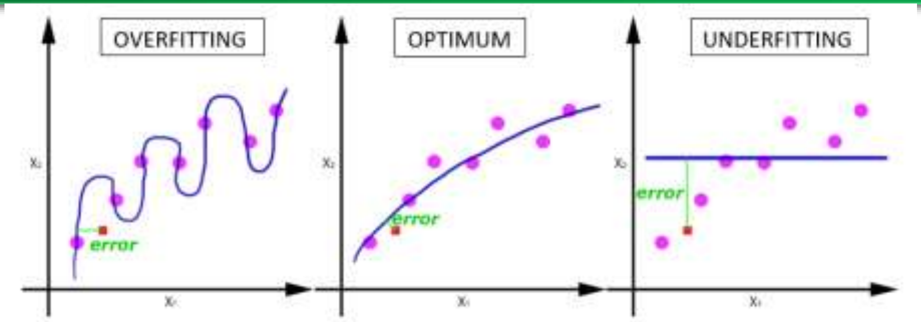
Unit-IV
CROSS VALIDATION

5. Cross Validation

- Model accuracy can be improved using cross-validation

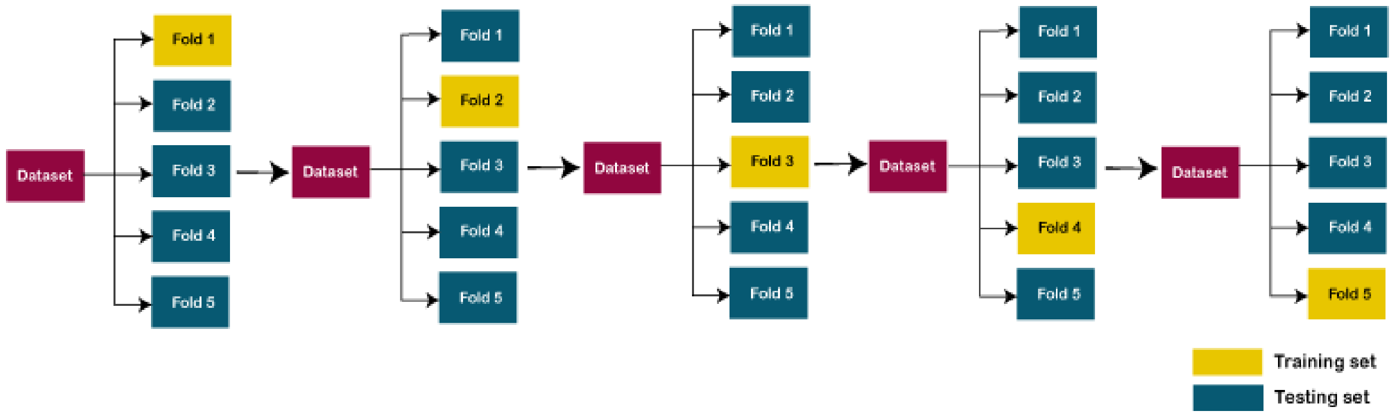
Cross Validation


01
Get Data
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.
02
Clean, Prepare & Manipulate Data
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.
03
Train Model
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.
04
Test Data
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.
05
Improve
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.



- These terms describe two opposing extremes which both result in poor performance.
- **Overfitting**
  - A model that was trained too much on the particulars of the training data (when the model learns the noise in the dataset).
  - A model that is overfitting will not perform well on new, unseen data.
  - Overfitting is arguably the most common problem in applied machine learning and is especially troublesome because a model that appears to be highly accurate will actually perform poorly in the wild.
- **Underfitting**
  - A model that has not been trained sufficiently.
  - This could be due to insufficient training time or a model that was simply not trained properly.
  - A model that is underfitting will perform poorly on the training data as well as new, unseen data alike.
- **Both underfitting and overfitting will yield poor performance**
  - the sweet spot is in between these two extremes.
  - As the number of training iterations increases (Cross Validation), the parameters of the model are updated and the curve goes from underfitting to optimal to overfitting.
- **Optimal fitting**
  - **The optimal state** is referred to as **generalization**.
  - This is where the model performs well on both training data and new data not seen during the training process.

- **What is Cross-Validation?**
  - **Cross-Validation** aims to test the model’s ability to make a prediction of new data not used in estimation so that problems like **overfitting or selection bias are flagged.**
  - Insight into the generalization of the database is given.



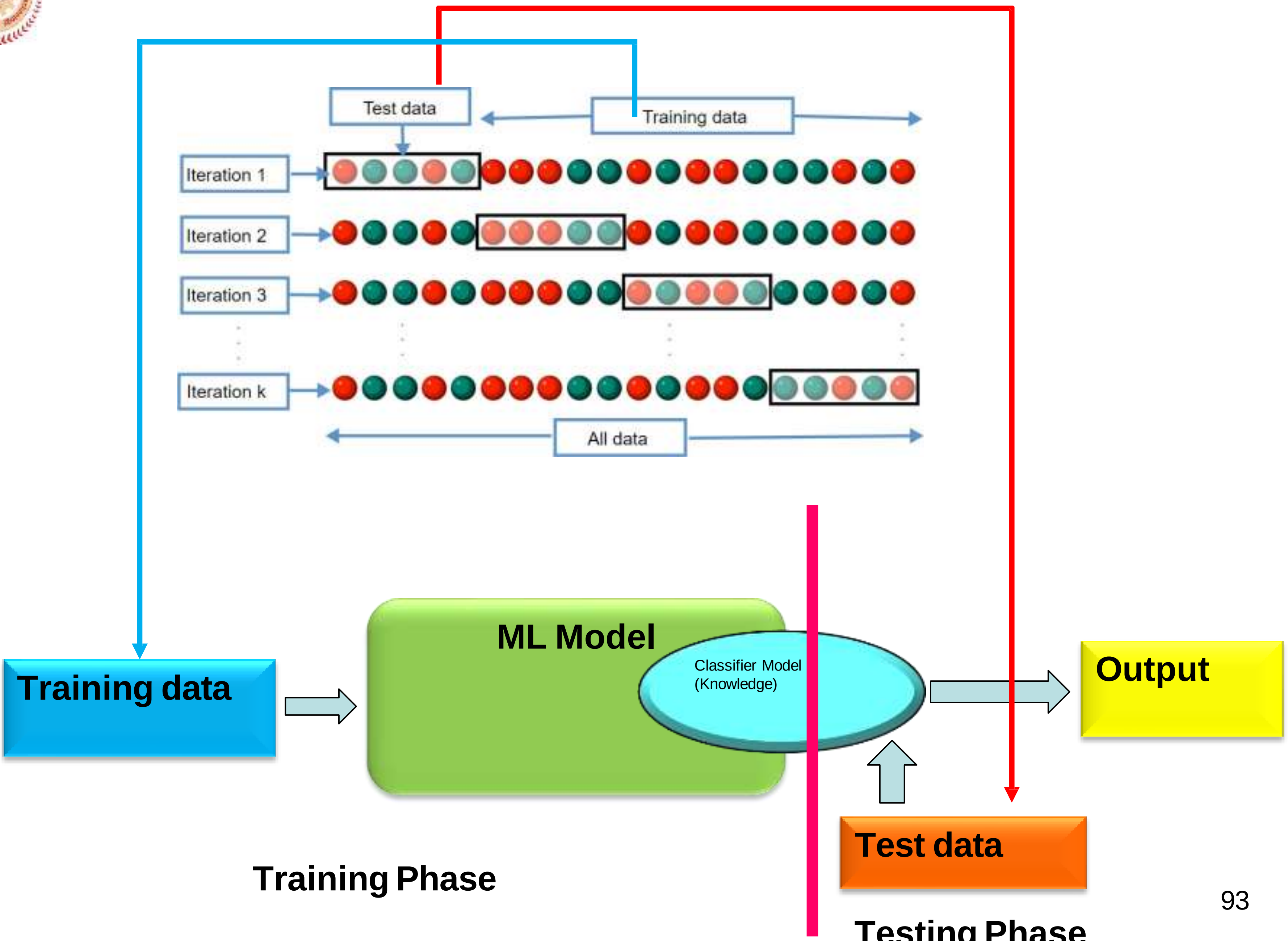
Cross-validation algorithm

Briefly, cross-validation algorithms can be summarized as follow:

1. Reserve a small sample of the data set
2. Build (or train) the model using the remaining part of the data set
3. Test the effectiveness of the model on the reserved sample of the data set.

   If the model works well on the test data set, then it’s good.

- **What is the use of cross-validation?**
  - **Cross-Validation** is primarily used in applied machine learning for estimation of the skill of the model on future data.
    - We use a given sample to estimate how the model is generally expected to perform while making predictions on unused data during the model training.
- **Does Cross-Validation reduce Overfitting?**
  - **Cross-Validation** is a strong protective action against overfitting.



Test data
Training data
Iteration 1
Iteration 2
Iteration 3
Iteration k
All data
ML Model
Classifier Model (Knowledge)
Training data
Output
Test data
Training Phase
Testing Phase

- **Cross-validation methods**
  1. Validation Set Approach.
  2. Leave-one-out cross-validation (LOOCV)
  3. k-Fold Cross-Validation
  4. Stratified k-fold Cross-Validation

1. The Validation set Approach

- **The validation set approach consists of randomly splitting the data into two sets:**
  - one set is used to **train the model**
  - another set is used to **test the model**

- **The process works as follows:**

  1. Build (train) the model on the training data set
  2. Apply the model to the test data set to predict the outcome of new unseen observations
  3. Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values.

- When comparing two models, the one that produces the lowest test sample RMSE is the preferred model.

- the RMSE and the MAE are measured on the same scale as the outcome variable. Dividing the RMSE by the average value of the outcome variable will give you the prediction error rate, which should be as small as possible:
- `RMSE(predictions,`

```
test.data$Fertility)/mean(test.data$Fertility)
```

- ## [1] 0.128

2. Leave one out cross validation - LOOCV

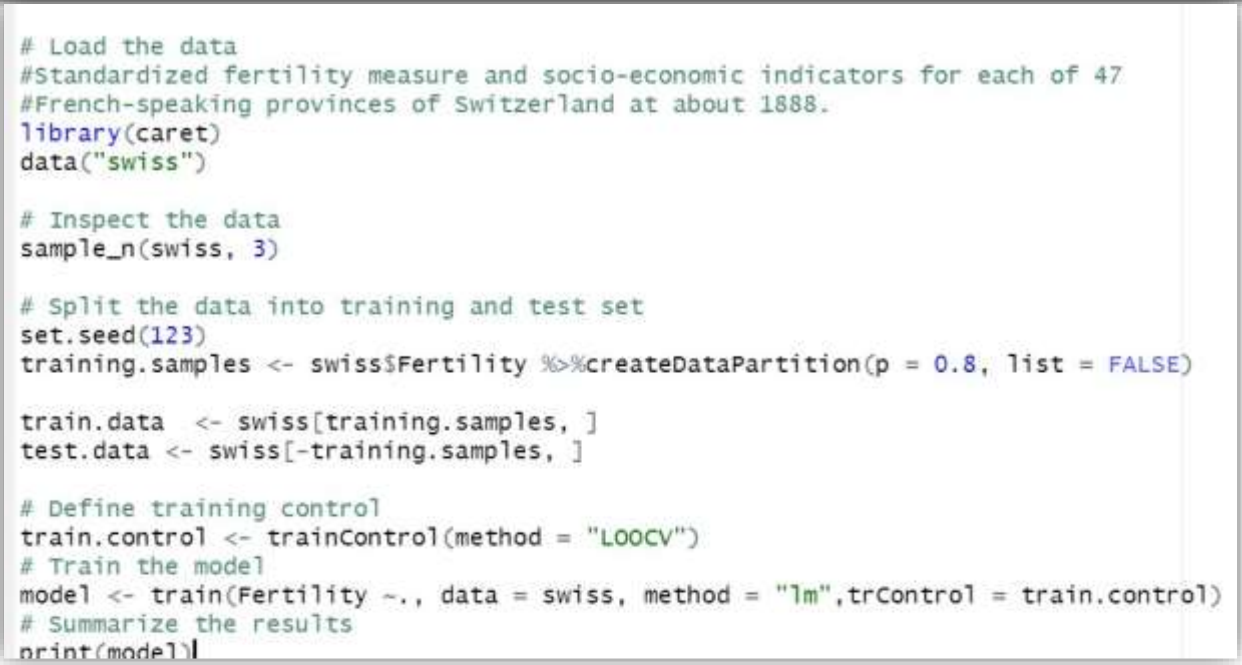
- **This method works as follows:**

1. Leave out one data point and build the model on the rest of the data set

2. Test the model against the data point that is left out in step 1 and record the test error associated with the prediction

3. Repeat the process for all data points

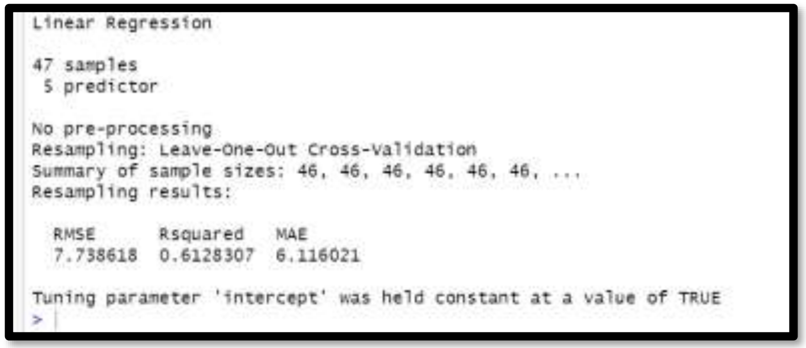
4. Compute the overall prediction error by taking the average of all these test error estimates recorded at step 2.



```
# Load the data
#Standardized fertility measure and socio-economic indicators for each of 47
#French-speaking provinces of Switzerland at about 1888.
library(caret)
data("swiss")

# Inspect the data
sample_n(swiss, 3)

# Split the data into training and test set
set.seed(123)
training.samples <- swiss$Fertility %>%createDataPartition(p = 0.8, list = FALSE)

train.data  <- swiss[training.samples, ]
test.data <- swiss[-training.samples, ]

# Define training control
train.control <- trainControl(method = "LOOCV")
# Train the model
model <- train(Fertility ~., data = swiss, method = "lm",trControl = train.control)
# Summarize the results
print(model)
```

Program

Output



```
Linear Regression

47 samples
 5 predictor

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 46, 46, 46, 46, 46, 46, ...
Resampling results:

  RMSE      Rsquared   MAE
  7.738618  0.6128307  6.116021

Tuning parameter 'intercept' was held constant at a value of TRUE
>
```

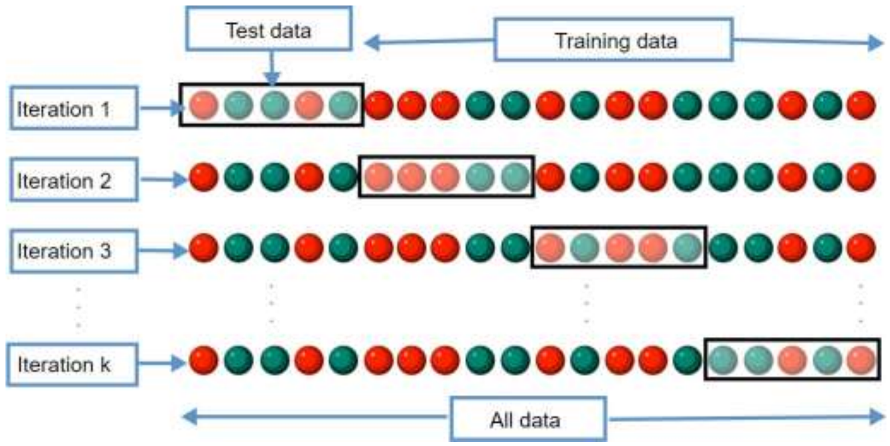
- The advantage of the LOOCV method is that we make use all data points reducing potential bias.
- The process is repeated as many times as there are data points, resulting to a higheinexecution time when n is extremely large.
- We test the model performance against one data point at each iteration.
- This might result in higher variation in the prediction error if some data points are outliers.
    - So, we need a good ratio of testing data points, a solution provided by the **k-fold cross-validation method**.

3. K-fold cross-validation

- The k-fold cross-validation method evaluates the model performance on a different subset of the training data and then calculates the average prediction error rate.

- **The algorithm is as follows:**
  1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
  2. Reserve one subset and train the model on all other subsets
  3. Test the model on the reserved subset and record the prediction error
  4. Repeat this process until each of the k subsets has served as the test set.
  5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

- K-fold cross-validation (CV) is a robust method for estimating the accuracy of a model.



4. Repeated (stratified) K-fold cross-validation

- The process of splitting the data into k-folds can be repeated a number of times, this is called repeated k-fold cross validation.
- The final model error is taken as the mean error from the number of repeats.

Setting repetitions is also a big challenge

•

- We generally recommend the **(repeated) k-fold cross-validation** to estimate the prediction error rate.
  - It can be used in regression and classification settings.

- Another alternative to cross-validation is the **bootstrap** resampling method
  - which consists of repeatedly and randomly selecting a sample of n observations from the original data set and evaluating the model performance on each copy.

Unsupervised Learning: Clustering

Unsupervised Learning: Clustering

Unsupervised Learning: Clustering - K-means, K-Modes, K-Prototypes, Gaussian Mixture Models, Expectation-Maximization.

Reinforcement Learning: Exploration and exploitation trade-offs, non-associative learning, Markov decision processes, Q-learning.

Unsupervised Learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets.

These algorithms discover hidden patterns or data groupings without the need for human intervention.

Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

Unsupervised learning models are utilized for three main tasks—

- Clustering
- Association
- Dimensionality reduction

Unsupervised Learning

Clustering:

Clustering or Cluster analysis is the method of grouping the entities based on similarities. Defined as an unsupervised learning problem that aims to make training data with a given set of inputs but without any target values.
It is the process of finding similar structures in a set of unlabeled data to make it more understandable and manipulative.

Types of Clustering Methods

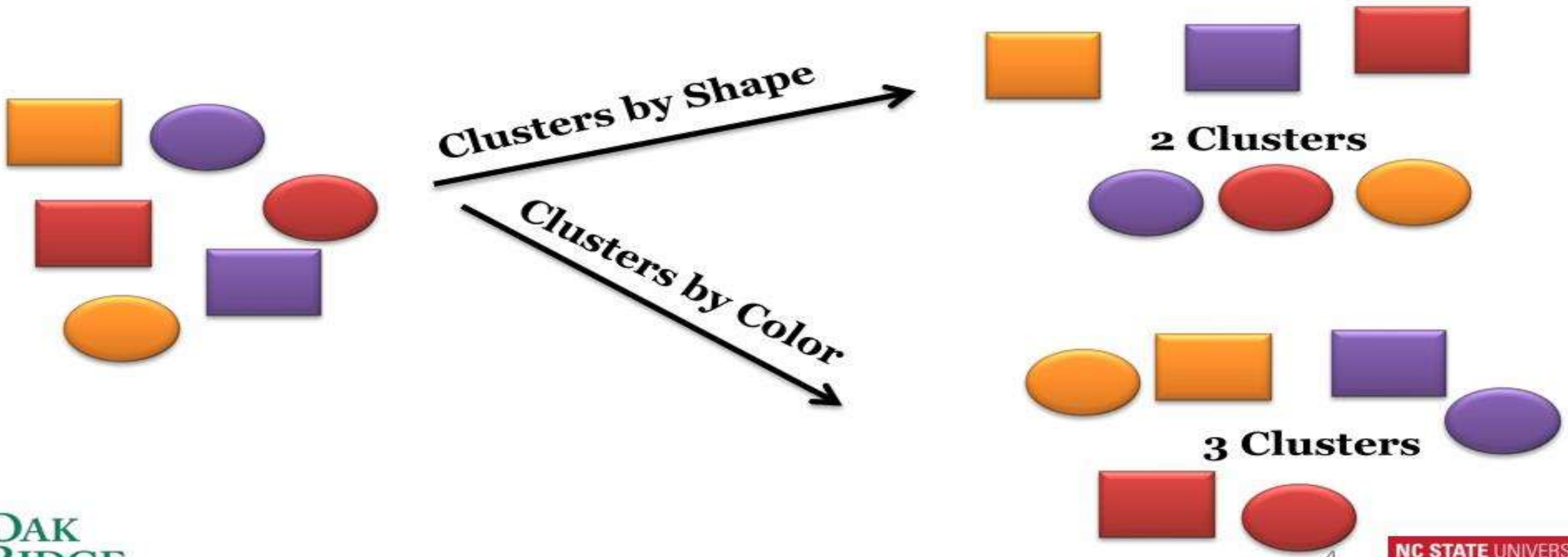
The cluster formation depends upon different parameters like shortest distance, graphs, and density of the data points.
Grouping into clusters is conducted by finding the measure of similarity between the objects based on some metric called the similarity measure. It is easier to find similarity measures in a lesser number of features.
Creating similarity measures becomes a complex process as the number of features increases. Different types of clustering approaches in data mining use different methods to group the data from the datasets.

Types of clustering approaches.

1. **Connectivity-based Clustering (Hierarchical clustering)**
2. **Centroids-based Clustering (Partitioning methods)**

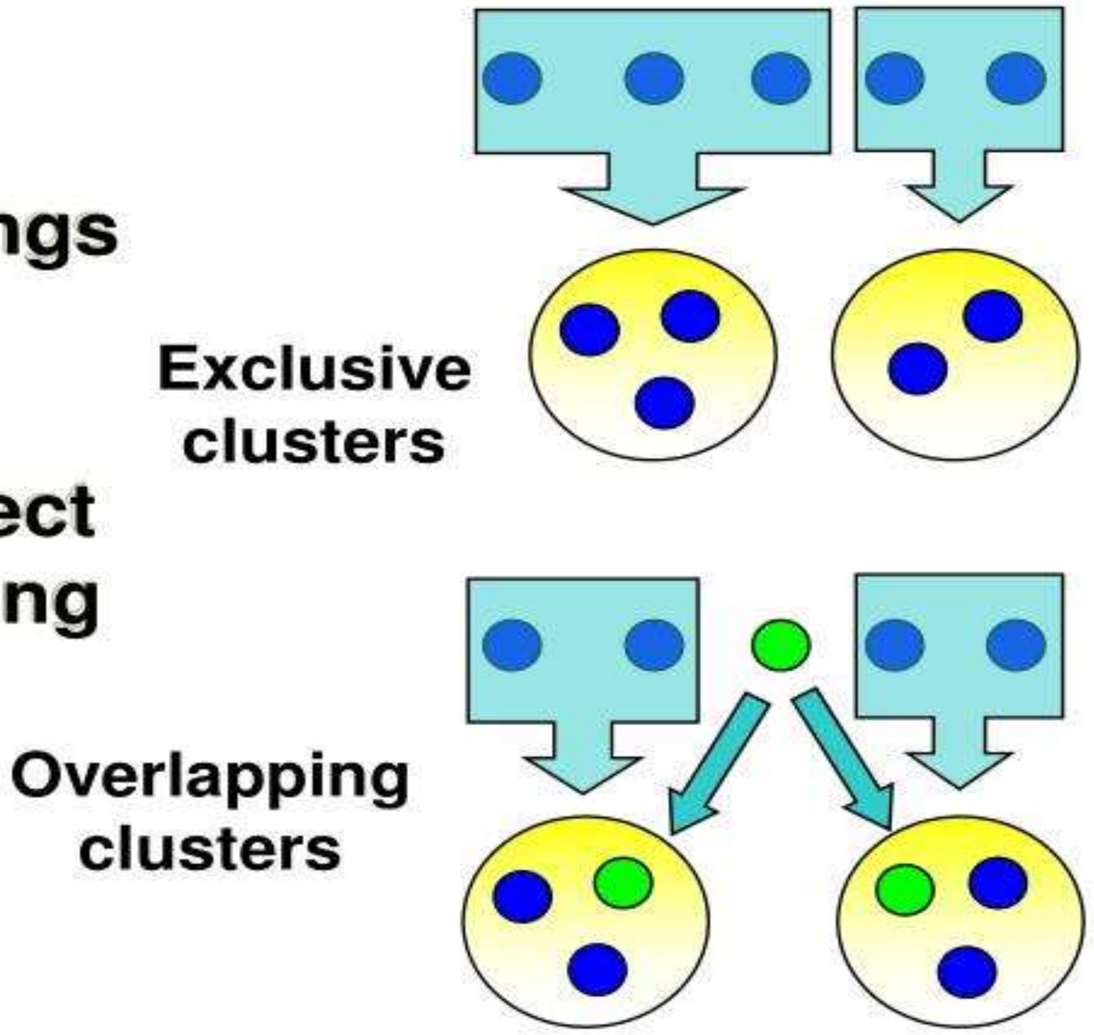
What is Cluster Analysis?

The process of dividing a set of input data into possibly overlapping, subsets, where elements in each subset are considered related by some similarity measure

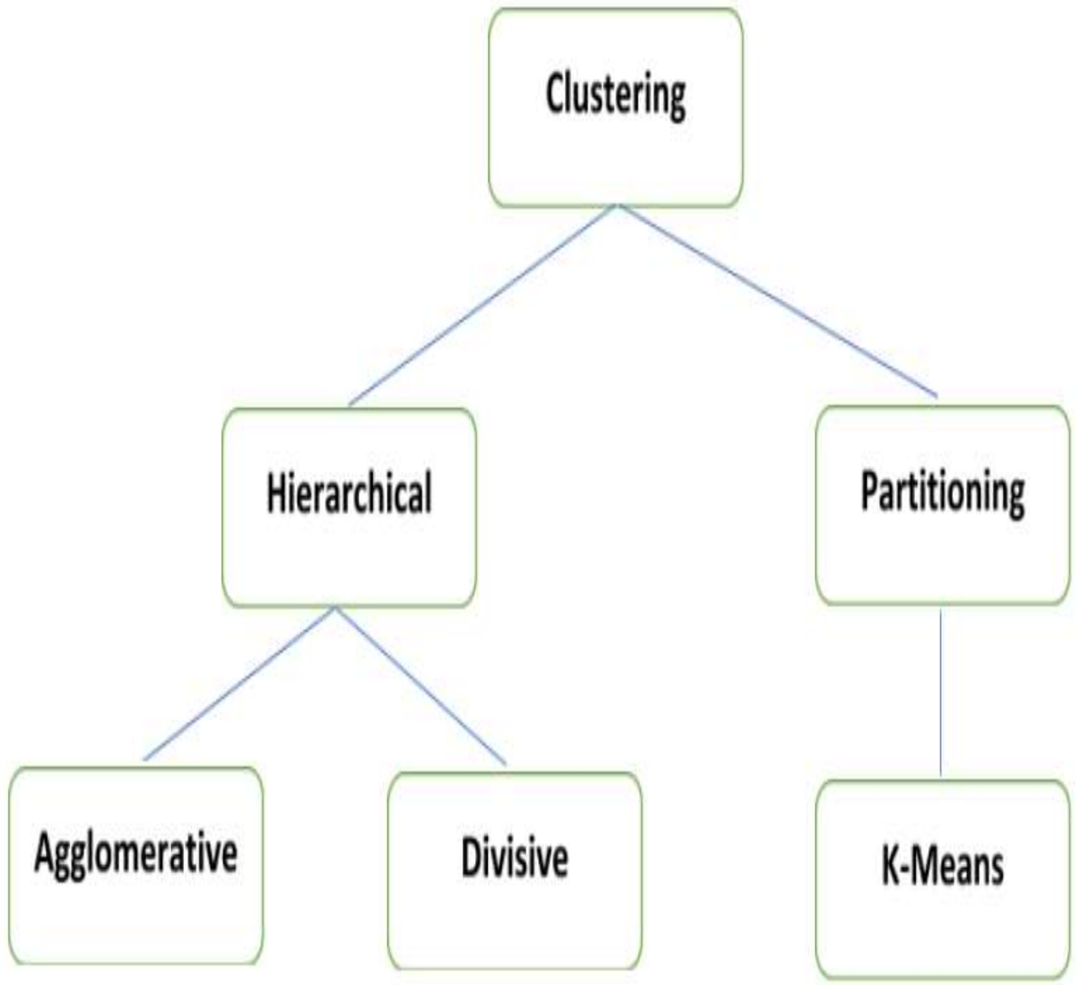


Types of clusters

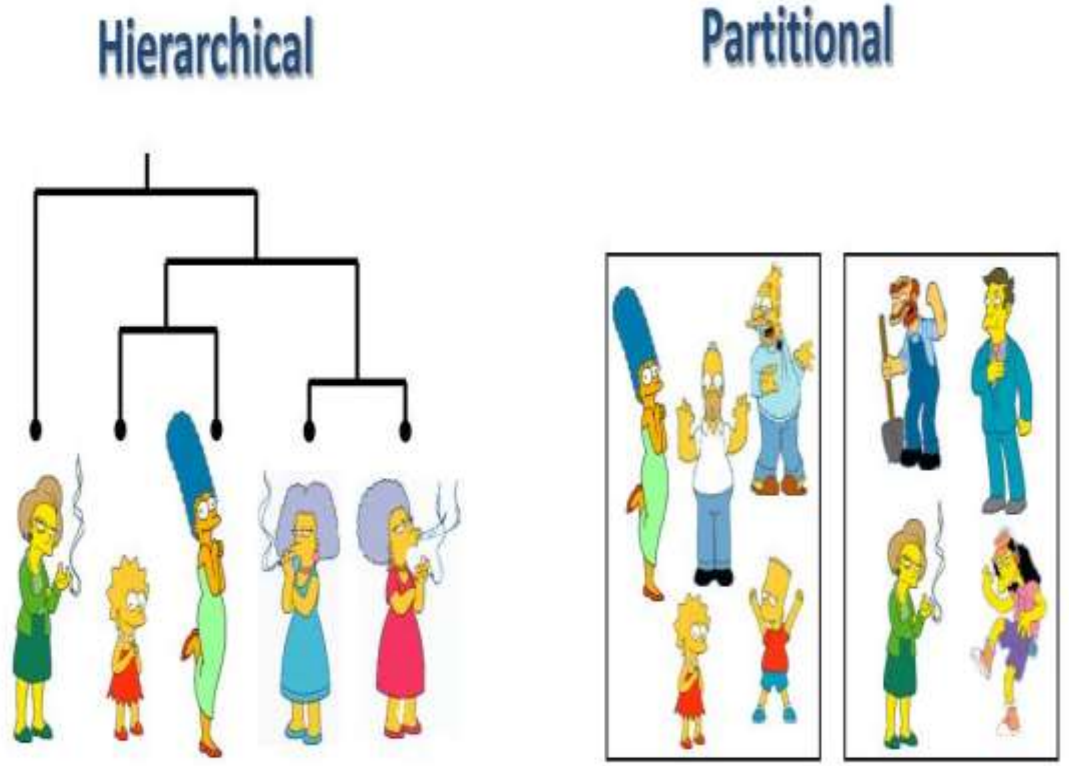
- *Exclusive*: each object/variable belongs to one and only one cluster.
- *Overlapping*: an object or variable may belong to more than one cluster.



Unsupervised Learning



- Partitional Clustering (unnested)
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering (nested)
  - A set of nested clusters organized as a hierarchical tree



Partitioning Clustering:

1. K-means
2. K-Modes
3. K-Prototypes

1. K-means clustering:

•K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

•K-means clustering is commonly used in market segmentation, document clustering, image segmentation, and image compression.

•K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

•It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm

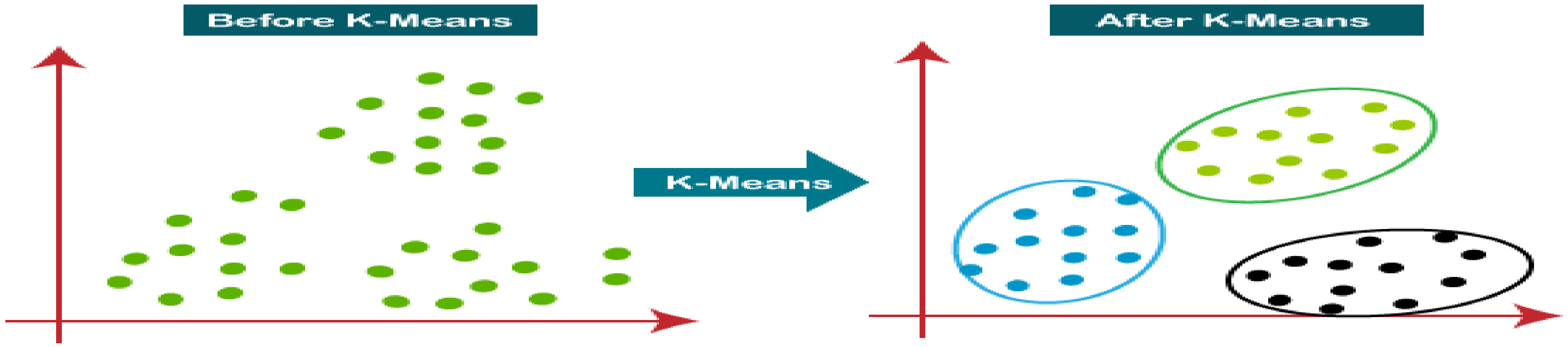
Unsupervised Learning

The k-means clustering algorithm mainly performs two tasks:

- **Determines the best value for K center points or centroids by an iterative process.**
- **Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.**

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



Unsupervised Learning

How does the K-Means Algorithm Work?

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

2. K-Modes clustering:

•**KModes clustering** is one of the unsupervised Machine Learning algorithms that is used to cluster **categorical variables.**

•Unlike traditional clustering algorithms that use distance metrics, KModes works by identifying the modes or most frequent values within each cluster to determine its centroid.

•KModes is ideal for clustering categorical data such as customer demographics, market segments, or survey responses

K-Modes vs KMeans

KMeans uses mathematical measures (distance) to cluster continuous data.

The lesser the distance, the more similar our data points are. Centroids are updated by Means.

But for categorical data points, we cannot calculate the distance. So we go for K - Modes algorithm.

It uses the dissimilarities(total mismatches) between the data points. The lesser the dissimilarities the more similar our data points are. It uses Modes instead of means.

Unsupervised Learning: K-Modes

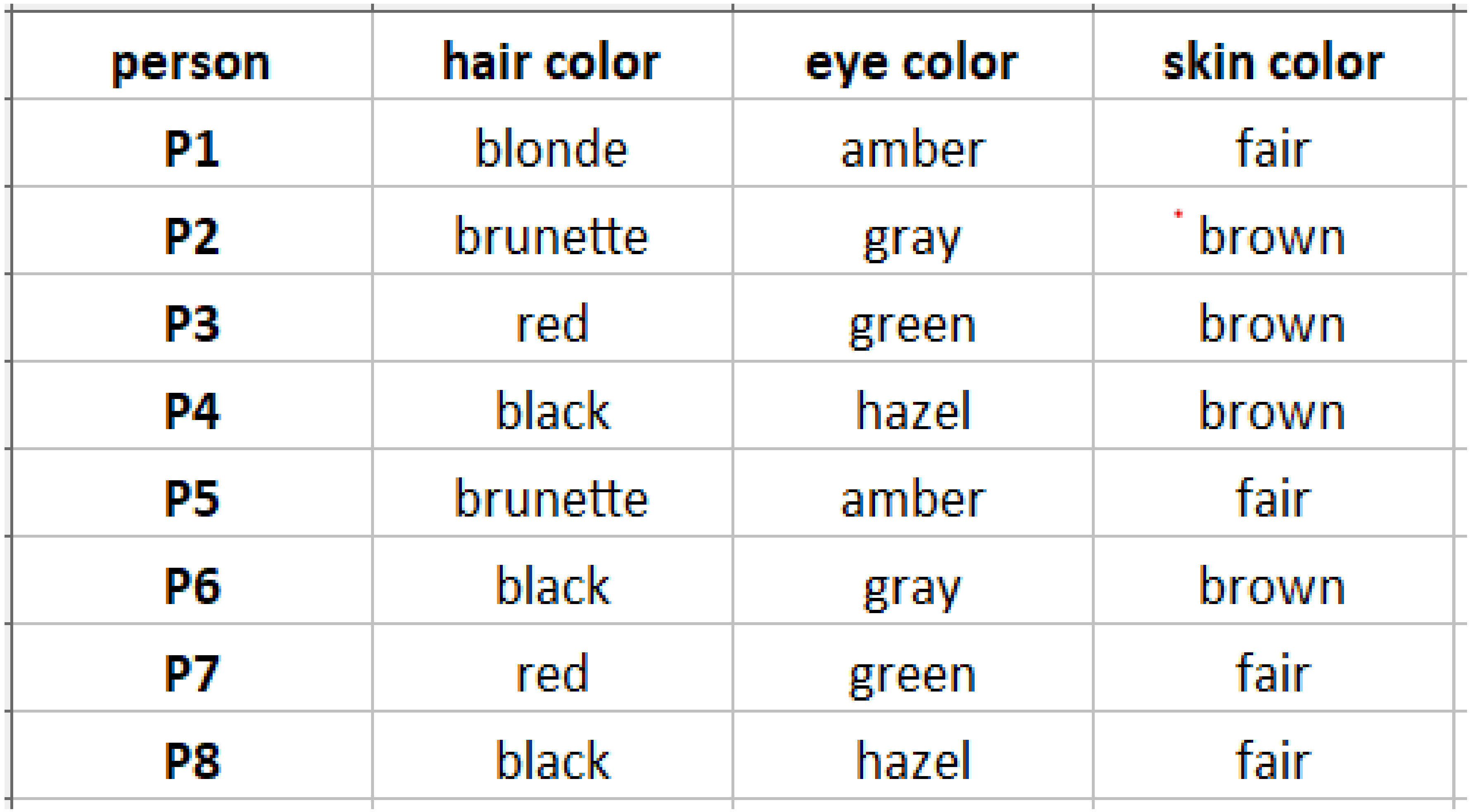
How Does the K - Modes Algorithm Work?

1. Pick K observations at random and use them as leaders/clusters
2. Calculate the dissimilarities and assign each observation to its closest cluster
3. Define new modes for the clusters
4. Repeat 2–3 steps until there are is no re-assignment required

Example: Imagine we have a dataset that has the information about hair color, eye color, and skin color of persons. We aim to group them based on the available information

Unsupervised Learning: K-Modes

Hair color, eye color, and skin color are all categorical variables. Below is how our dataset looks like.

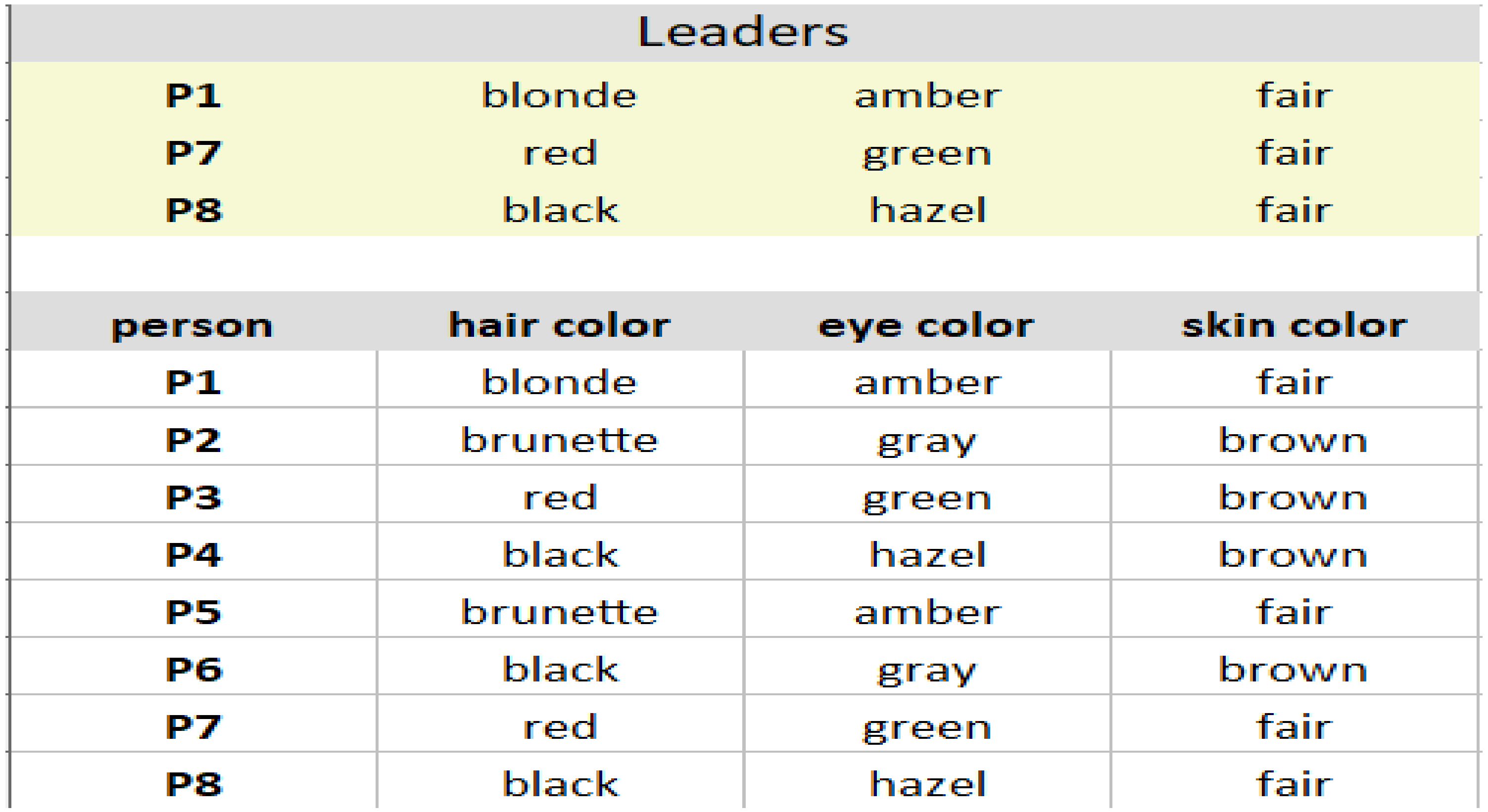


person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Let us proceed by defining the number of clusters(K)=3

Step 1: Pick K observations at random and use them as leaders/clusters

I am choosing P1, P7, P8 as leaders/clusters

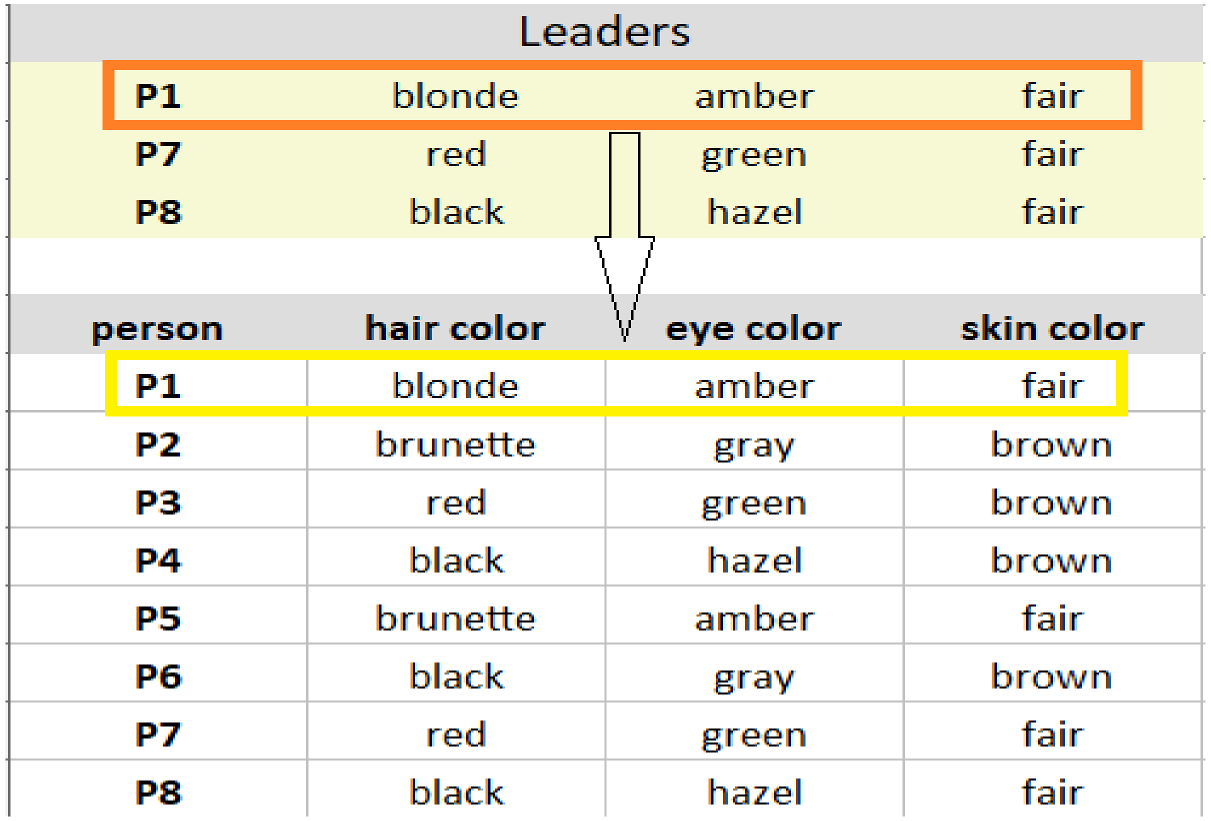


Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Step 2: Calculate the dissimilarities(no. of mismatches) and assign each observation to its closest cluster

Iteratively compare the cluster data points to each of the observations. Similar data points give 0, dissimilar data points give 1.



Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/Cluster P1 to the observation P1 gives 0 dissimilarities.

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/cluster

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Likewise, calculate all the dissimilarities and put them in a matrix as shown below and assign the observations to their closest cluster(cluster that has the least dissimilarity)

	Cluster 1 (P1)	Cluster 2 (P7)	Cluster 3 (P8)	Cluster
P1	0 ✓	2	2	Cluster 1
P2	3 ✓	3	3	Cluster 1
P3	3	1 ✓	3	Cluster 2
P4	3	3	1 ✓	Cluster 3
P5	1 ✓	2	2	Cluster 1
P6	3	3	2 ✓	Cluster 3
P7	2	0 ✓	2	Cluster 2
P8	2	2	0 ✓	Cluster 3

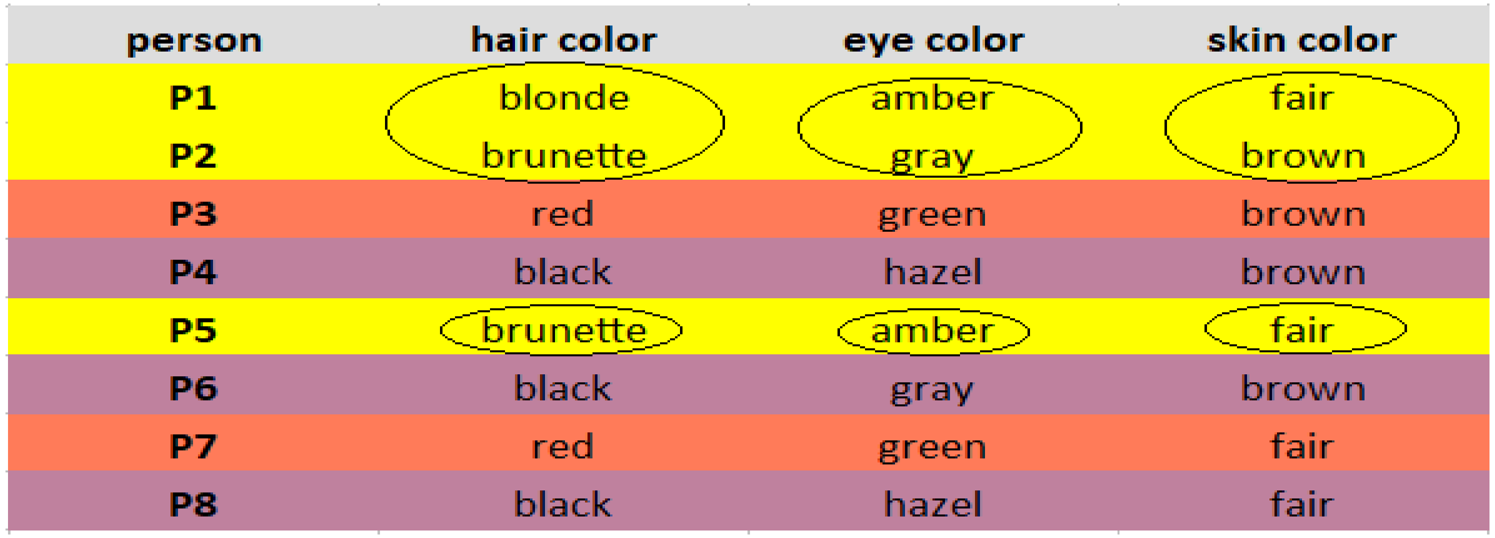
After step 2, the observations P1, P2, P5 are assigned to cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to cluster 3.

Note: If all the clusters have the same dissimilarity with an observation, assign to any cluster randomly. In our case, the observation P2 has 3 dissimilarities with all the leaders. I randomly assigned it to Cluster 1.

Step 3: Define new modes for the clusters

Mode is simply the **most observed value**.

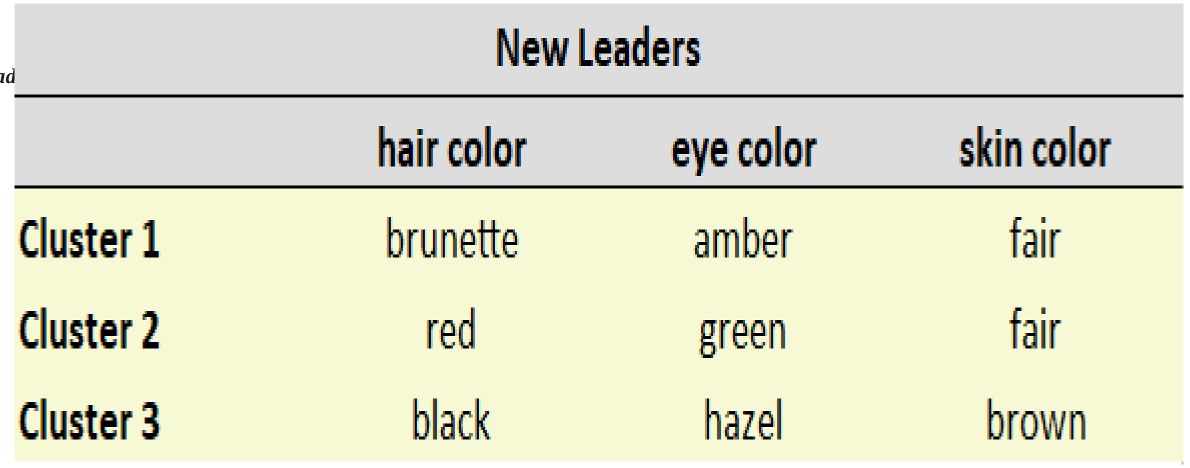
Mark the observations according to the cluster they belong to. Observations of Cluster 1 are marked in Yellow, Cluster 2 are marked in Brick red, and Cluster 3 are marked in Purple.



person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Considering one cluster at a time, for each feature, look for the Mode and update the new leaders.
Explanation: Cluster 1 observations(P1, P2, P5) has brunette as the most observed hair color, amber as the most observed eye color, and fair as the most observed skin color.
Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 3(P3, P7) have one occurrence of brown, fair skin color. I randomly chose brown as the mode.
Below are our new leaders after the update.

Obtained new lead



	New Leaders		
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

Repeat steps 2–4
After obtaining the new leaders, again calculate the dissimilarities between the observations and the newly obtained leaders.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing Cluster 1 to the observation P1 gives 1 dissimilarity.

New Leaders

	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing Cluster 1 to the observation P2 gives 2 dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix. Assign each observation to its closest cluster.

	Cluster 1	Cluster 2	Cluster 3	Cluster
P1	1 ✓	2	3	Cluster 1
P2	2 ✓	3	2	Cluster 1
P3	3	1 ✓	2	Cluster 2
P4	3	3	0 ✓	Cluster 3
P5	0 ✓	2	3	Cluster 1
P6	3	3	1 ✓	Cluster 3
P7	2	0 ✓	3	Cluster 2
P8	2	2	1 ✓	Cluster 3

The observations P1, P2, P5 are assigned to Cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to Cluster 3.
We stop here as we see there is no change in the assignment of observations.

3. K-Prototypes Clustering :

•Clustering algorithms are unsupervised machine learning algorithms used to segment data into various clusters.
•K-Prototypes clustering is a partitioning clustering algorithm. We use k-prototypes clustering to cluster datasets that have categorical as well as numerical attributes. The K-Prototypes clustering algorithm is an ensemble of k-means clustering and k-modes clustering algorithm. Hence, it can handle both numerical and categorical data.
In k-prototypes clustering, we select k-prototypes randomly at the start. After that, we calculate the distance between each data point and the prototypes. Accordingly, all the data points are assigned to clustering associated with different prototypes. After assigning data points to the clusters, we calculate the new prototype for the current cluster using the method discussed in the next sections. After that, we recalculate the distance of prototypes from the data points and reassign the clusters. This process is continued until the clusters converge.
Before getting into the numerical example for k-prototypes clustering, Let us first discuss the distance measures and calculation of prototypes in the k-prototypes clustering algorithm.

EM Algorithm-Steps

1. A set of initial values are considered
   - A set of incomplete data is given to the system
2. E-Step (Expectation Step)
   - Use observed data to estimate or guess the values of incomplete or missing data
3. M-Step (Maximization Step)
   - Use complete data generated in preceding E-Step to update the values
4. Convergence State Checking
   - Check the values are converging or not
   - If values are converging, stop the process
   - Else repeat step-2 and step-3 until the convergence occurs

- Usage:
  - Used to fill the missing data
  - Used for unsupervised clustering
  - Used to discover the values of latent variables
- Applications:
  - Clustering
  - Artificial Vision
  - NLP
  - Biological areas etc.
- Advantages:
  - With each iteration, likelihood increases
  - E-Step and M-Steps are easy to implement
- Disadvantages:
  - Slow Convergence
  - Makes Convergence to local optima only

Example

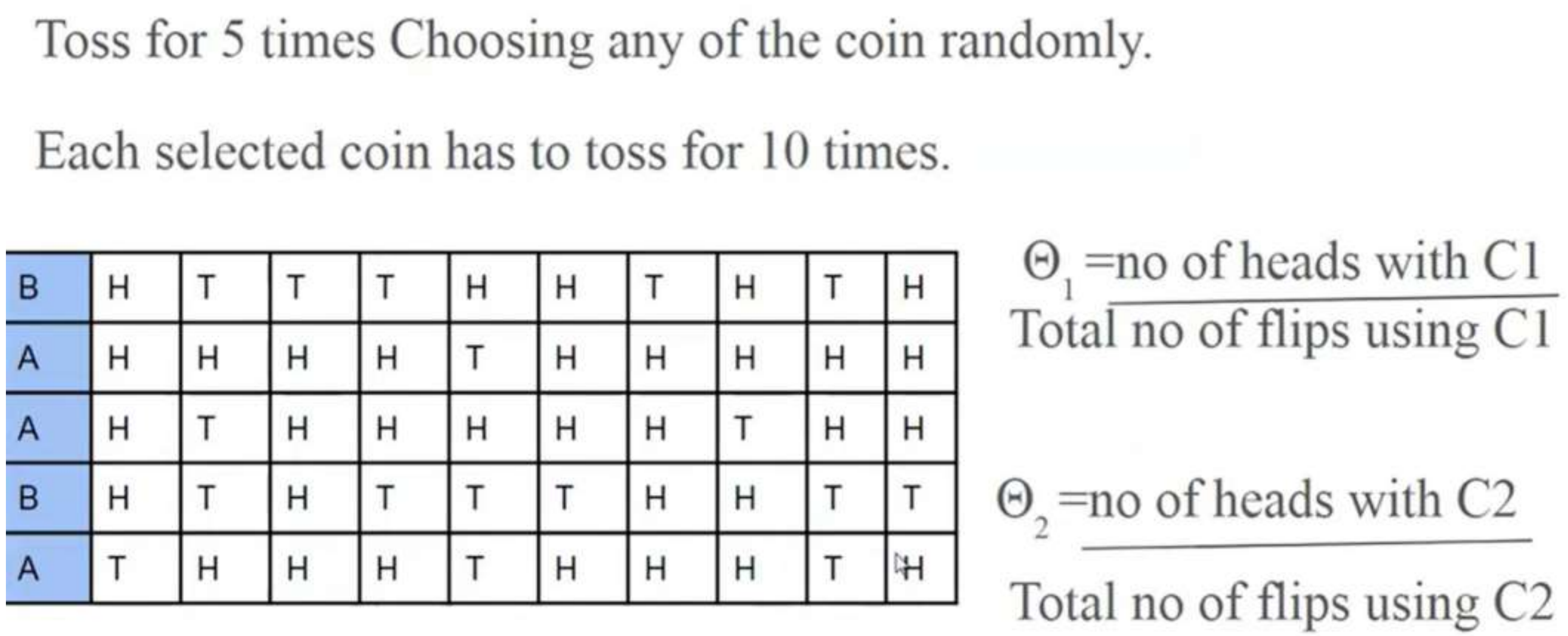
Let C1 and C2 be two coins.

Θ_1 be probability of getting head with C1

Θ_2 be probability of getting head with C2

Find value of Θ_1 and Θ_2 by tossing C1 and C2 for multiple times..

Case-1: If we know the Labels of Coins, then



Toss for 5 times Choosing any of the coin randomly.

Each selected coin has to toss for 10 times.

B	H	T	T	T	H	H	T	H	T	H
A	H	H	H	H	T	H	H	H	H	H
A	H	T	H	H	H	H	H	T	H	H
B	H	T	H	T	T	T	H	H	T	T
A	T	H	H	H	T	H	H	H	T	H

$$\Theta_1 = \frac{\text{no of heads with C1}}{\text{Total no of flips using C1}}$$

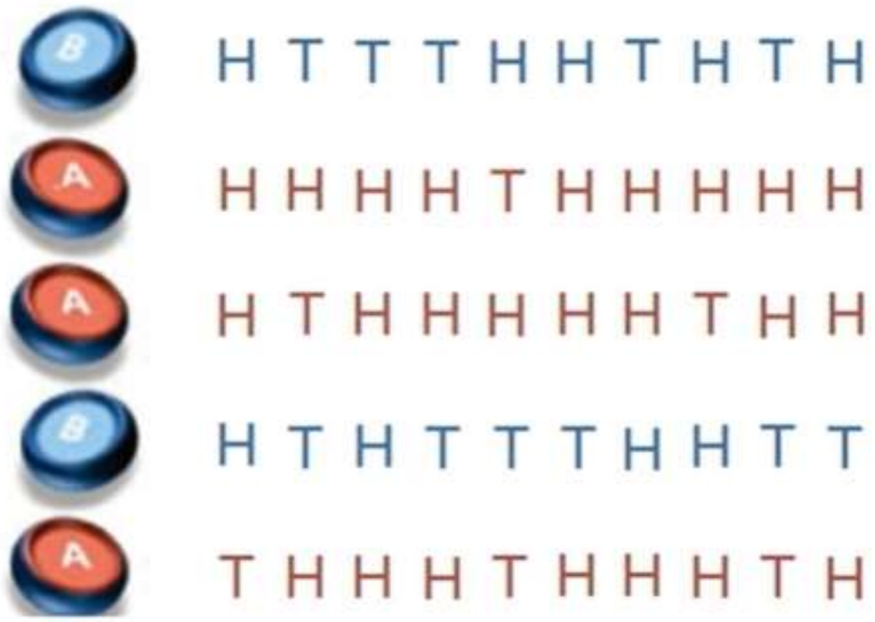
$$\Theta_2 = \frac{\text{no of heads with C2}}{\text{Total no of flips using C2}}$$

If we know the coin labels the probability will be as follows:

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	

$$\Theta_1 = 24/(24+6) = 0.8$$

$$\Theta_2 = 9/(9+11) = 0.45$$

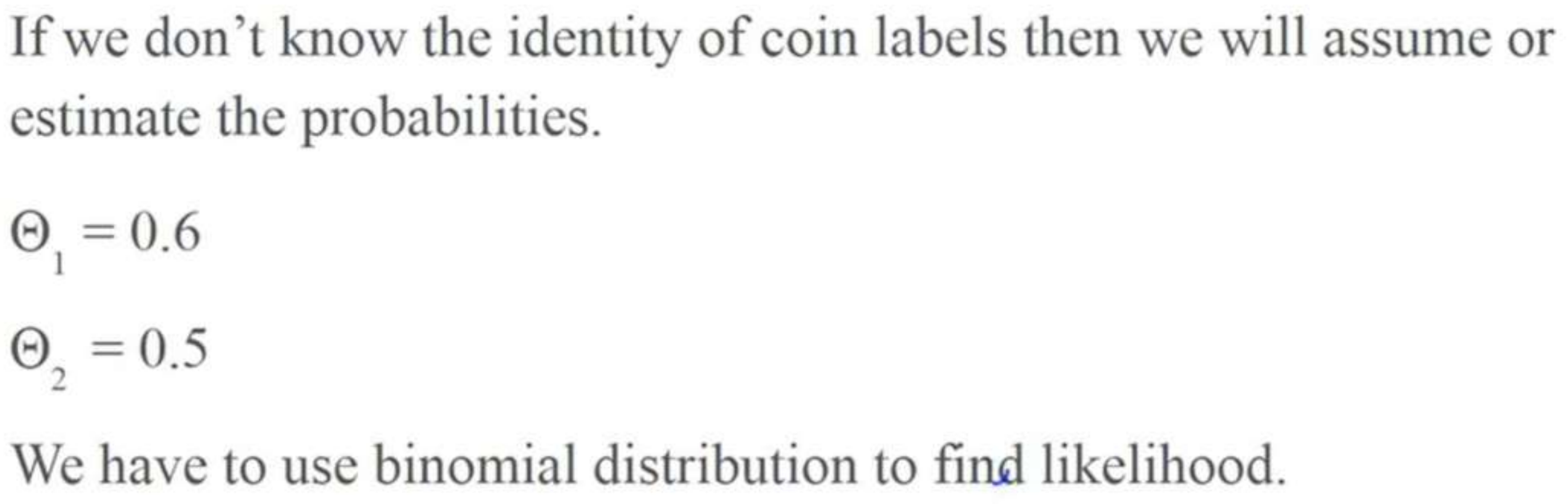


Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\theta_1 = \frac{24}{24+6} = 0.8$$

$$\theta_2 = \frac{9}{9+11} = 0.45$$

Case-2: If we don’t know the labels of coins, then



If we don’t know the identity of coin labels then we will assume or estimate the probabilities.

$\Theta_1 = 0.6$

$\Theta_2 = 0.5$

We have to use binomial distribution to find likelihood.

$$\boxed{L(C)=\Theta^k(1-\Theta)^{n-k}}$$

$$L(C)=\Theta^{k}(1-\Theta)^{n-k}$$

Likelihood For first coin Flips

$$L(A)=0.6^{5}(1-0.6)^{10-5}=0.0007963$$

$$L(B)=0.5^{5}(1-0.5)^{10-5}=0.0009766$$

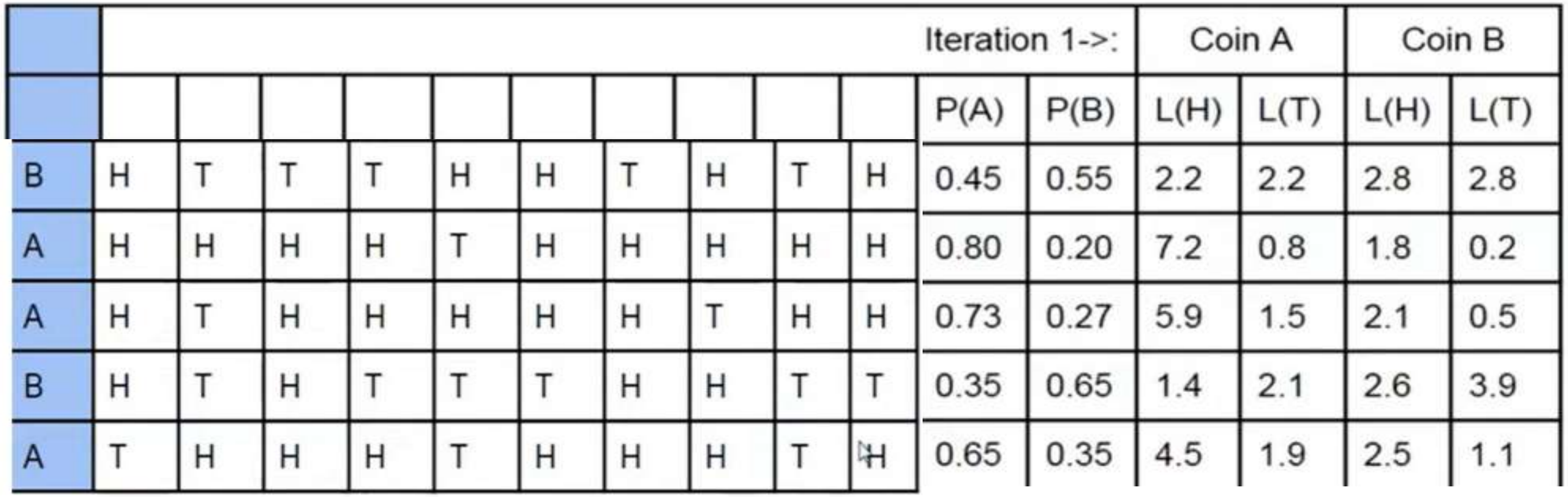
$$P(A)=L(A)/L(A)+L(B)=0.0007963/(0.0007963+0.0009766)=0.45$$

$$P(B)=L(B)/L(A)+L(B)=0.0009766/(0.0007963+0.0009766)=0.55$$

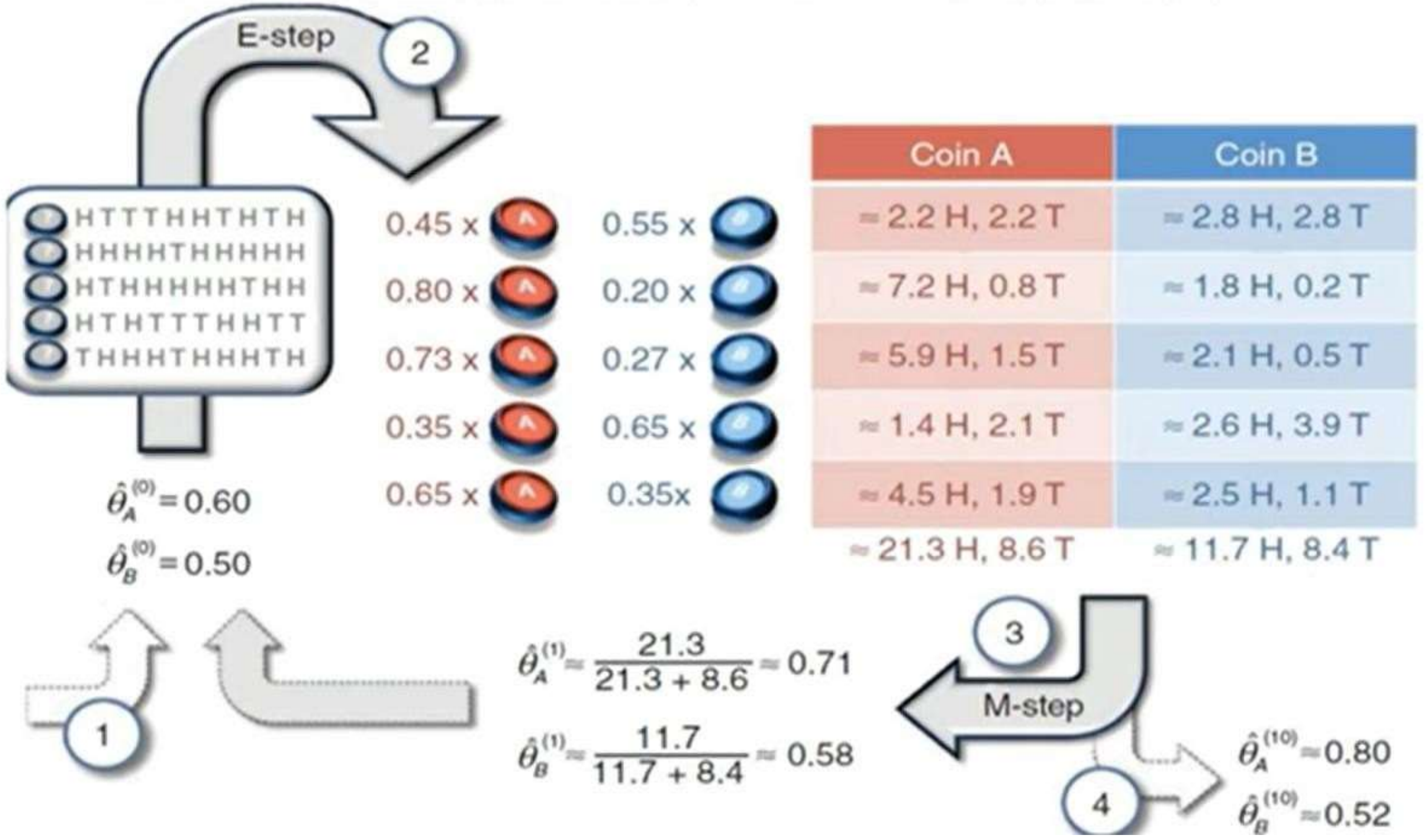
Similarly continue this procedure for next coins' sequences

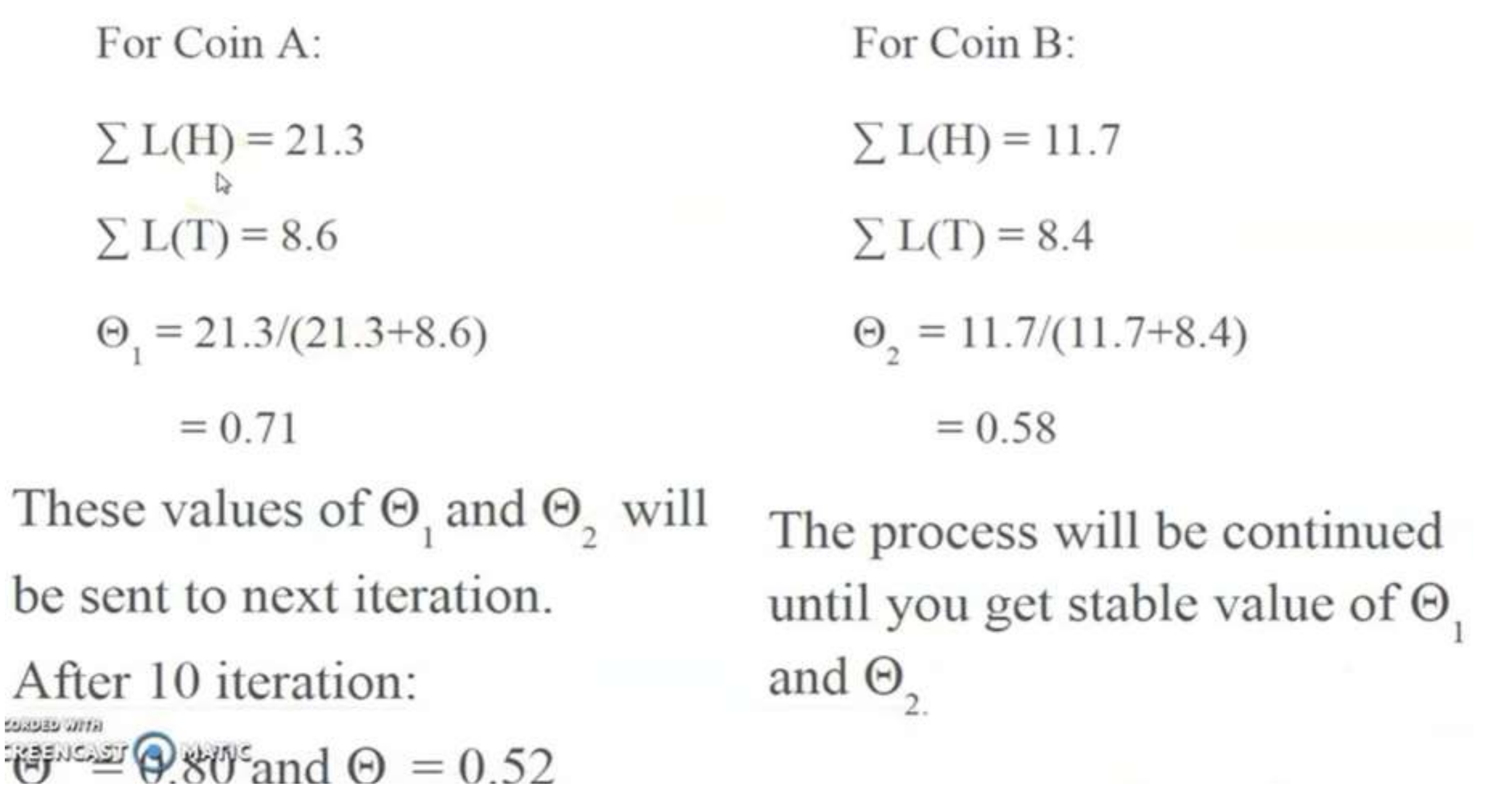
In similar fashion find probability of all coins with all flips. It will be as follows:

L(H): Likely no of heads L(T): Likely no of tails



										Iteration 1->:		Coin A		Coin B	
										P(A)	P(B)	L(H)	L(T)	L(H)	L(T)
B	H	T	T	T	H	H	T	H	T	H	0.45	0.55	2.2	2.2	2.8	2.8
A	H	H	H	H	T	H	H	H	H	H	0.80	0.20	7.2	0.8	1.8	0.2
A	H	T	H	H	H	H	H	T	H	H	0.73	0.27	5.9	1.5	2.1	0.5
B	H	T	H	T	T	T	H	H	T	T	0.35	0.65	1.4	2.1	2.6	3.9
A	T	H	H	H	T	H	H	H	T	H	0.65	0.35	4.5	1.9	2.5	1.1


E-step
2
H T T T H H T H T H
H H H H T H H H H H
H T H H H H H T H H
H T H T T T H H T T
T H H H T H H H T H
0.45 x
0.55 x
0.80 x
0.20 x
0.73 x
0.27 x
0.35 x
0.65 x
0.65 x
0.35x
Coin A
Coin B
≈ 2.2 H, 2.2 T
≈ 2.8 H, 2.8 T
≈ 7.2 H, 0.8 T
≈ 1.8 H, 0.2 T
≈ 5.9 H, 1.5 T
≈ 2.1 H, 0.5 T
≈ 1.4 H, 2.1 T
≈ 2.6 H, 3.9 T
≈ 4.5 H, 1.9 T
≈ 2.5 H, 1.1 T
≈ 21.3 H, 8.6 T
≈ 11.7 H, 8.4 T
$\hat{\theta}_A^{(0)} = 0.60$
$\hat{\theta}_B^{(0)} = 0.50$
1
3
M-step
$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71$
$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$
4
$\hat{\theta}_A^{(10)} \approx 0.80$
$\hat{\theta}_B^{(10)} \approx 0.52$



For Coin A:

$\sum L(H) = 21.3$

$\sum L(T) = 8.6$

$\Theta_1 = 21.3/(21.3+8.6)$

$= 0.71$

For Coin B:

$\sum L(H) = 11.7$

$\sum L(T) = 8.4$

$\Theta_2 = 11.7/(11.7+8.4)$

$= 0.58$

These values of Θ_1 and Θ_2 will be sent to next iteration.

After 10 iteration:

$\Theta = 0.80$ and $\Theta = 0.52$

The process will be continued until you get stable value of Θ_1 and $\Theta_{2.}$

Recent Hierarchical Clustering Methods

- Major weakness of agglomerative clustering methods
  - Do not scale well:
    - time complexity of at least $O(n^2)$, where n is the number of total objects
  - Can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996):
    - uses CF-tree and incrementally adjusts the quality of sub-clusters
  - ROCK (1999):
    - clustering categorical data by neighbor and link analysis
  - CHAMELEON (1999):
    - hierarchical clustering using dynamic modeling

Introduction to BIRCH

- Designed for very large data sets
  - Time and memory are limited
  - Incremental and dynamic clustering of incoming objects
  - Only one scan of data is necessary
  - Does not need the whole data set in advance
- Two key phases:
  - Scans the database to build an in-memory tree
  - Applies clustering algorithm to cluster the leaf nodes

Similarity Metric(1)

Given a cluster of instances $\{\vec{X}_i\}$, we define:

Centroid: $\vec{X}0 = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$

Radius: average distance from member points to centroid

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}0)^2}{N}\right)^{\frac{1}{2}}$$

Diameter: average pair-wise distance within a cluster

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Similarity Metric(2)

centroid Euclidean distance: $D0 = ((\vec{X}0_1 - \vec{X}0_2)^2)^{\frac{1}{2}}$

centroid Manhattan distance: $D1 = |\vec{X}0_1 - \vec{X}0_2| = \sum_{i=1}^{d} |\vec{X}0_1^{(i)} - \vec{X}0_1^{(i)}|$

average inter-cluster: $D2 = (\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{N_1 N_2})^{\frac{1}{2}}$

average intra-cluster: $D3 = (\frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{(N_1 + N_2)(N_1 + N_2 - 1)})^{\frac{1}{2}}$

variance increase:

$$(\sum_{k=1}^{N_1+N_2} (\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2} \vec{X}_l}{N_1+N_2})^2$$

$$- \sum_{i=1}^{N_1} (\vec{X}_i - \frac{\sum_{l=1}^{N_1} \vec{X}_l}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2} \vec{X}_l}{N_2})^2)^{\frac{1}{2}}$$

Properties of Clustering Feature

- **CF entry is more compact**
  - **Stores significantly less than all of the data points in the sub-cluster**
- **A CF entry has sufficient information to calculate D0-D4**
- **Additivity theorem allows us to merge sub-clusters incrementally & consistently**

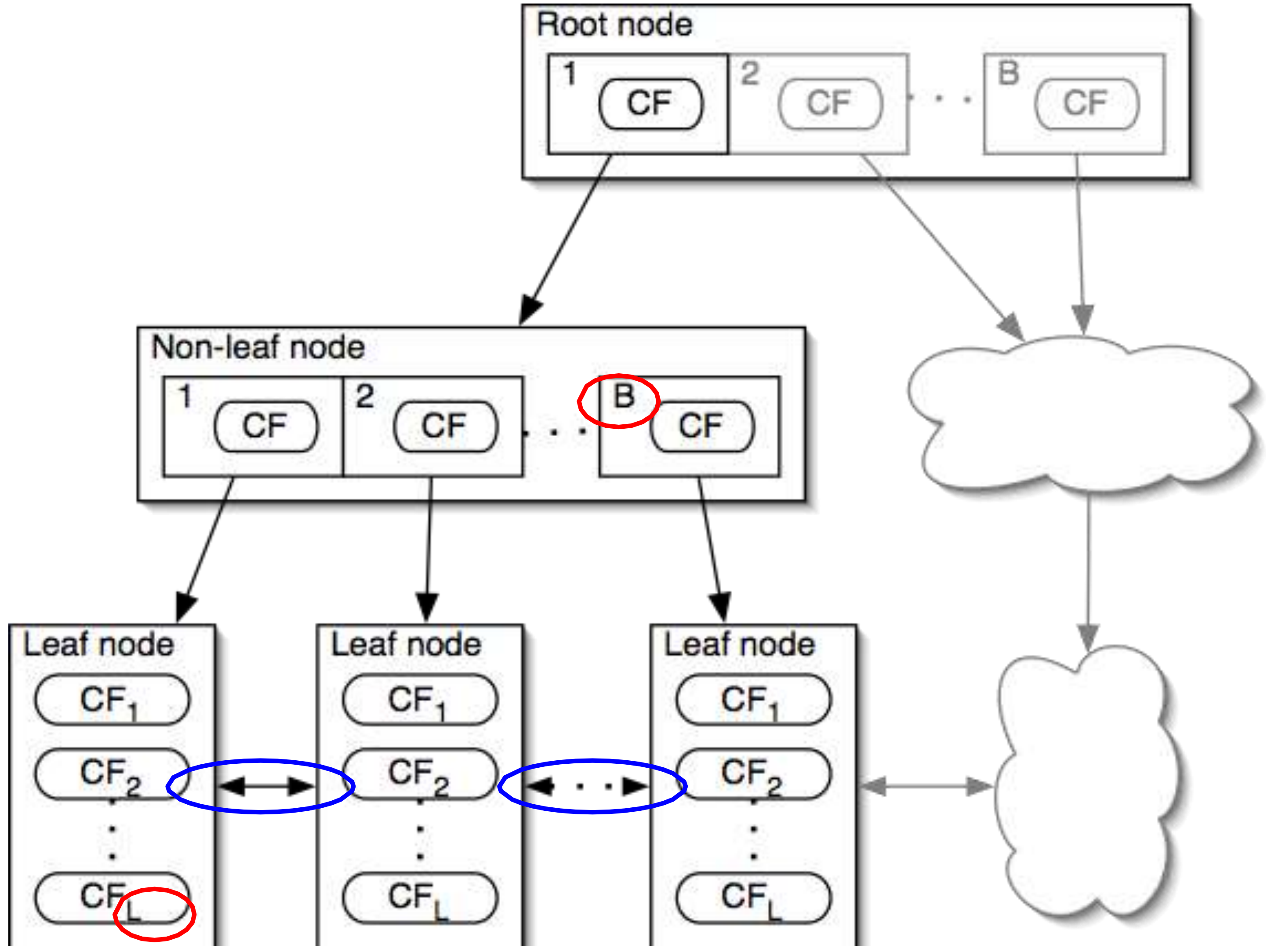
$$\mathbf{CF_1} + \mathbf{CF_2} = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

CF-Tree

Each non-leaf node has at most B entries

Each leaf node has at most L CF entries, each of which satisfies threshold T

Node size is determined by dimensionality of data space and input parameter P (page size)



Clustering Feature

The Birch algorithm builds a dendrogram called clustering feature tree (CF tree) while scanning the data set.

Each entry in the CF tree represents a cluster of objects and is characterized by a 3-tuple: (N, LS, SS), where N is the number of objects in the cluster and LS, SS are defined in the following.

$$LS = \sum P_i$$

$$P_i \in N$$

$$SS = \sum_{P_i \in N} \left| \overrightarrow{P_i} \right|^2$$

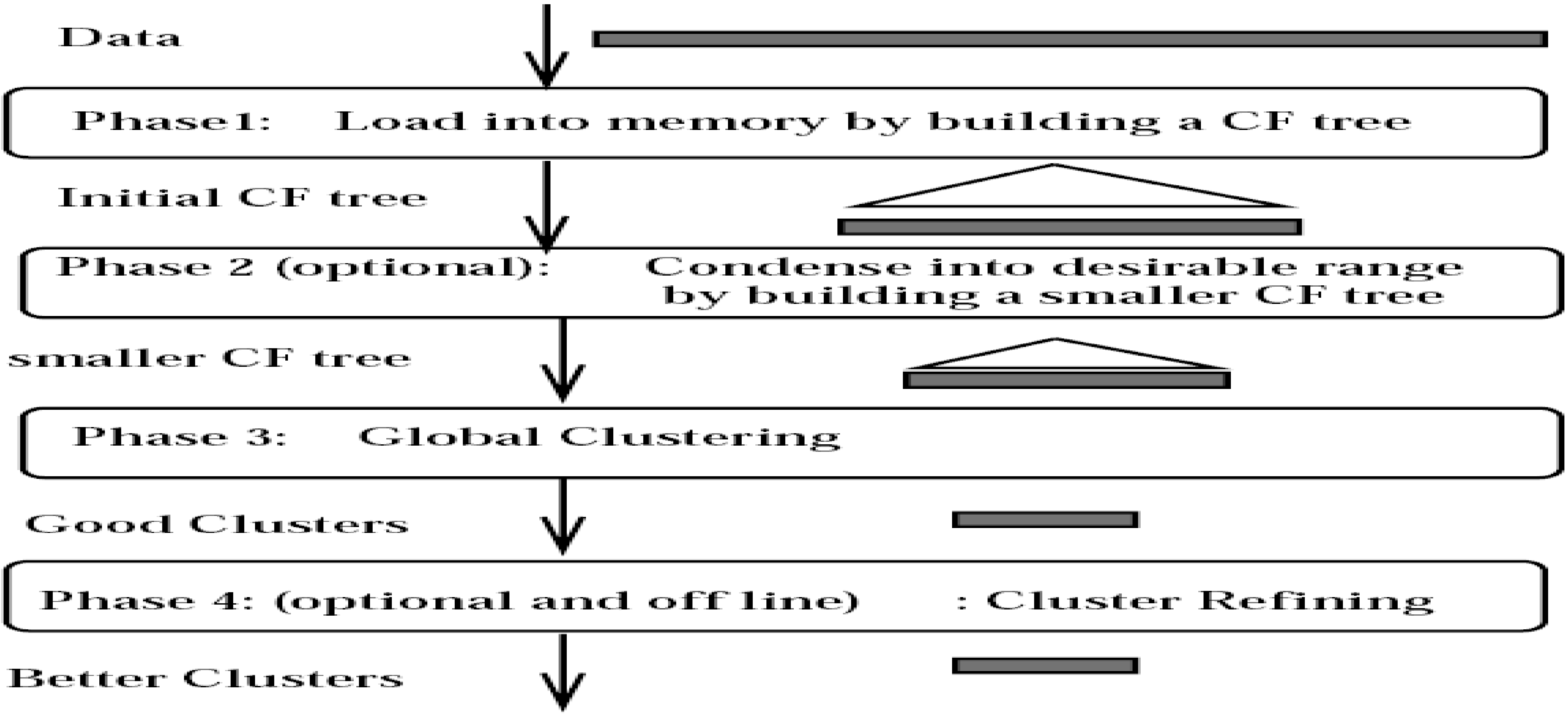
CF-Tree Insertion

- **Recurse down from root, find the appropriate leaf**
  - **Follow the "closest"-CF path, w.r.t. D0 / … / D4**
- **Modify the leaf**
  - **If the closest-CF leaf cannot absorb, make a new CF entry. If there is no room for new leaf, split the parent node**
- **Traverse back**
  - **Updating CFs on the path or splitting nodes**

CF-Tree Rebuilding

- If we run out of space, increase threshold T
  - By increasing the threshold, CFs absorb more data
- Rebuilding "pushes" CFs over
  - The larger T allows different CFs to group together
- Reducibility theorem
  - Increasing T will result in a CF-tree smaller than the original
  - Rebuilding needs at most h extra pages of memory

BIRCH Overview


Data
Phase1: Load into memory by building a CF tree
Initial CF tree
Phase 2 (optional): Condense into desirable range by building a smaller CF tree
smaller CF tree
Phase 3: Global Clustering
Good Clusters
Phase 4: (optional and off line) : Cluster Refining
Better Clusters

BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

- Begins by partitioning objects hierarchically using tree structures, and then applies other clustering algorithms to refine the clusters.

  Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering

1. Phase 1: scan DB to build an initial in-memory CF tree.

   (A multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
2. Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans.

- *Weakness:* Handles only numeric data, and sensitive to the order of the data record.

Clustering Feature Vector in BIRCH

Clustering Feature (CF): ***CF = (N, LS, SS)***

N: **Number of data points**

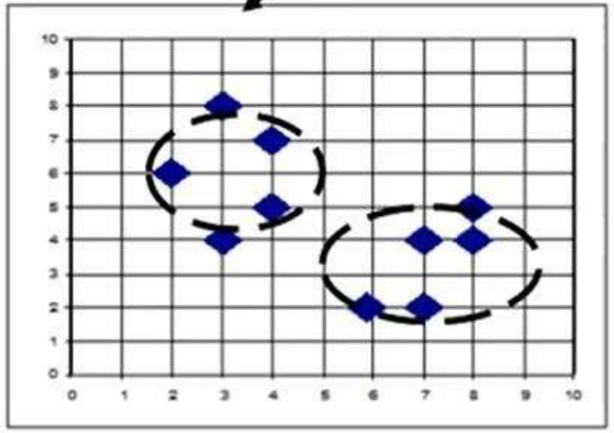
LS: linear sum of N points:

$$\sum_{i=1}^{N} X_i$$

SS: square sum of N points

$$\sum_{i=1}^{N} X_i^2$$

CF = (5, (16,30),(54,190))



(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF-Tree in BIRCH

Clustering feature:

- Summary of the statistics for a given subcluster: the 0-th, 1st, and 2nd moments of the subcluster from the statistical point of view
- Registers crucial measurements for computing cluster and utilizes storage efficiently

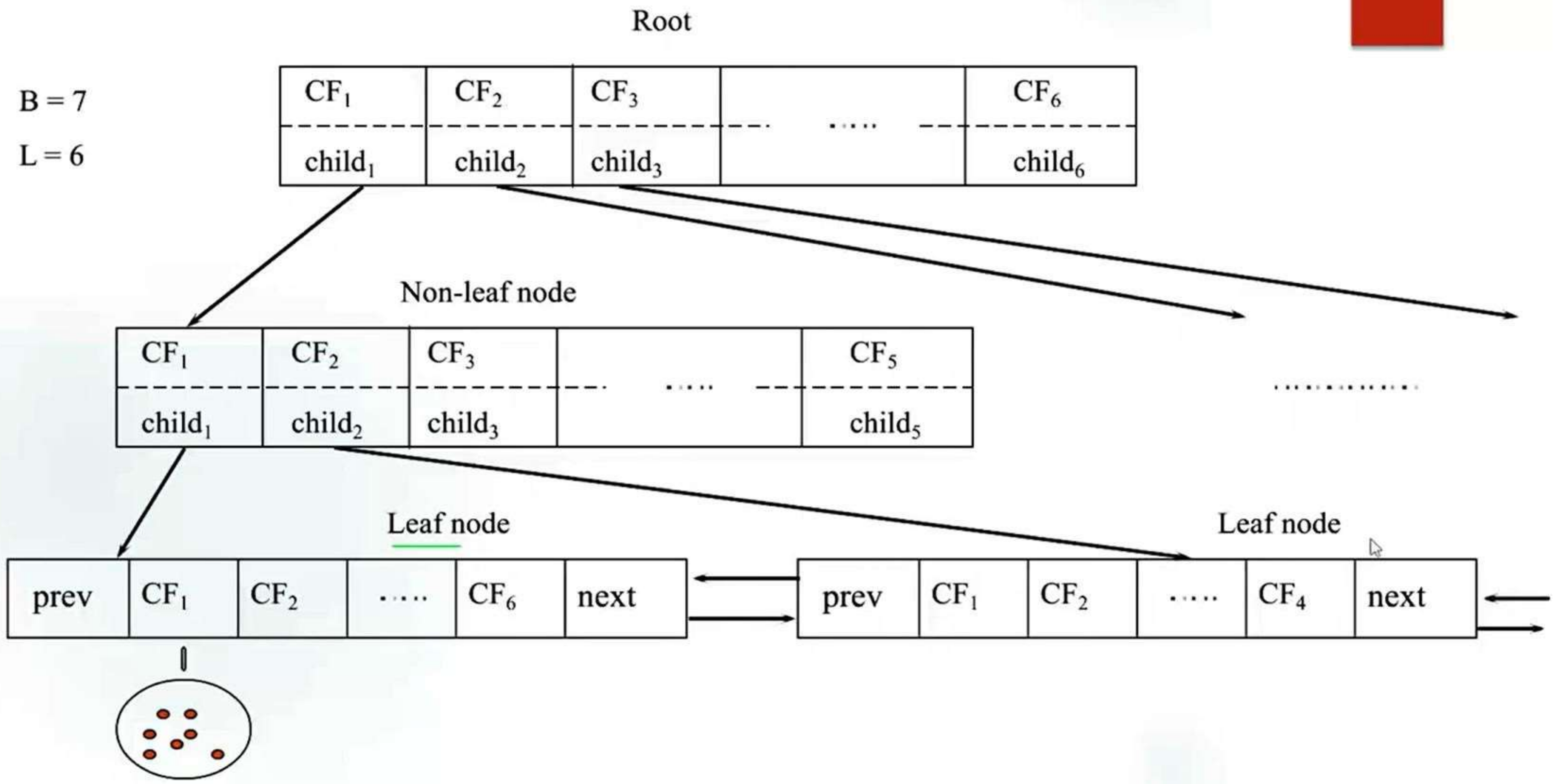
A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering

- A nonleaf node in a tree has descendants or "children"
- The nonleaf nodes store sums of the CFs of their children

A CF tree has two parameters

- Branching factor: max # of children
- Threshold: max diameter of sub-clusters stored at the leaf nodes

The CF Tree Structure



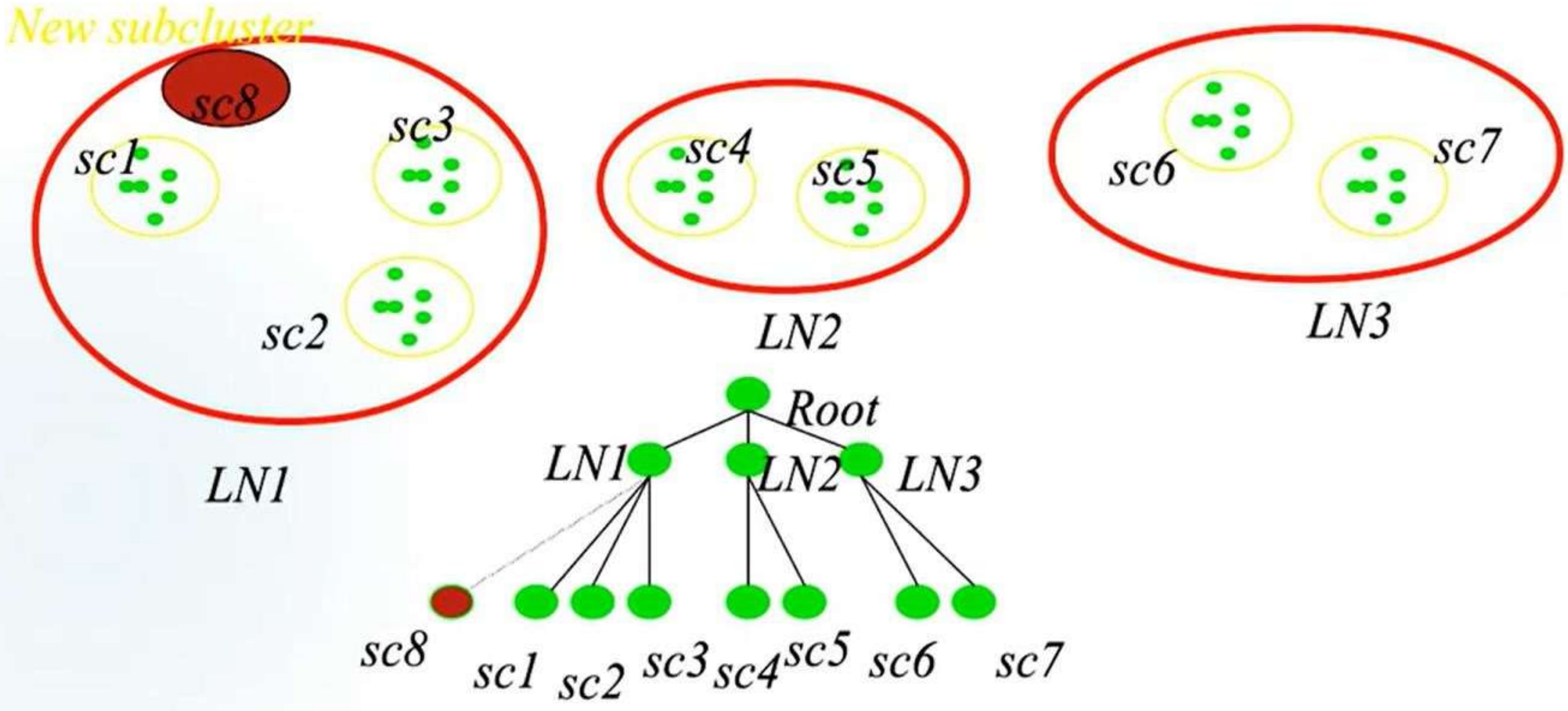
The Birch Algorithm

- Cluster Diameter

$$\sqrt{\frac{1}{n(n-1)} \sum (x_i - x_j)^2}$$

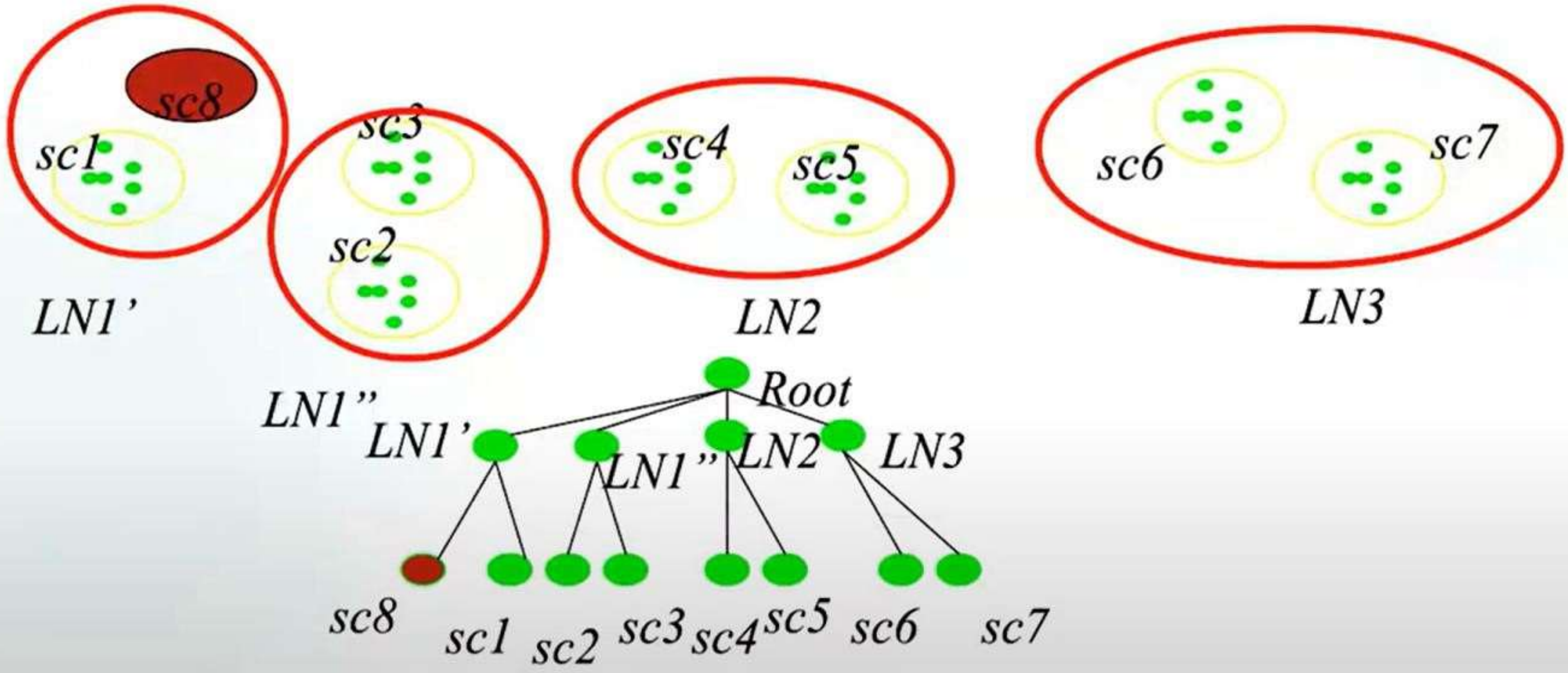
- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter, then split leaf, and possibly parents
- Algorithm is O(n)
- Concerns
  - Sensitive to insertion order of data points
  - Since we fix the size of leaf nodes, so clusters may not be so natural
  - Clusters tend to be spherical given the radius and diameter measures

Example of BIRCH

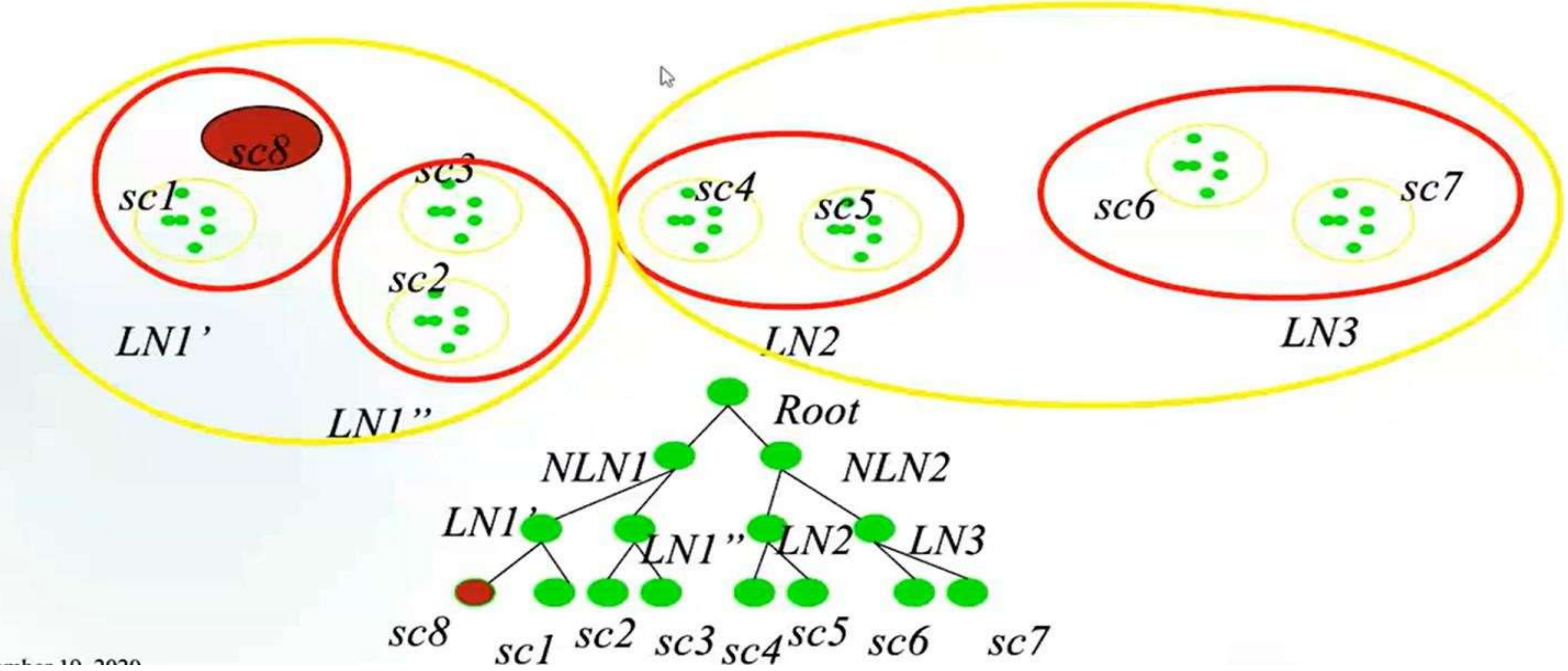


Insertion Operation in BIRCH

If the branching factor of a leaf node can not exceed 3, then LN1 is split.



If the branching factor of a non-leaf node can not exceed 3, then the root is split and the height of the CF Tree increases by one.



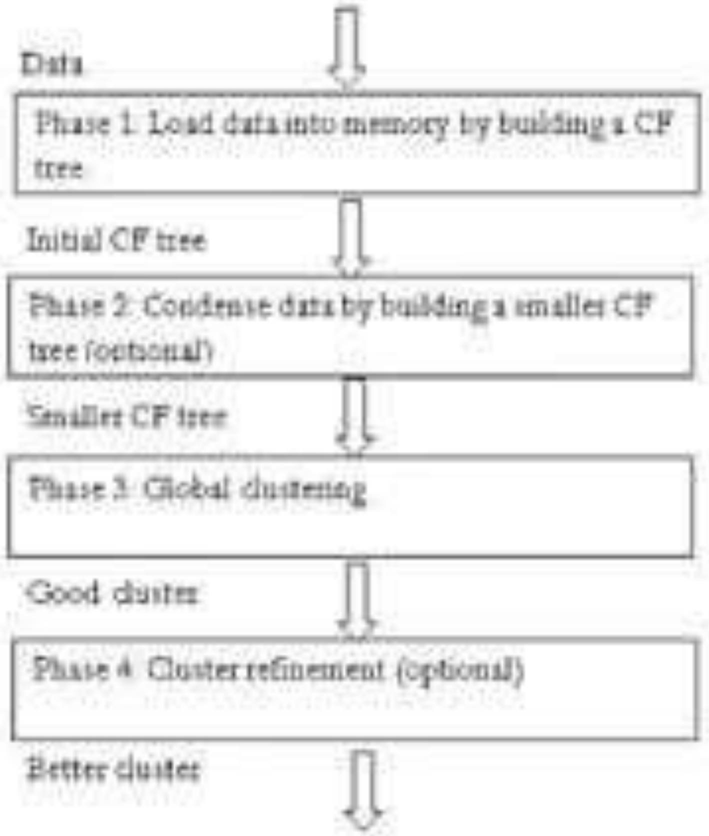
Mathematical Example

Introduction

BIRCH (*balanced iterative reducing and clustering using hierarchies*) is an unsupervised data-mining algorithm used to perform hierarchical-clustering over particularly large data-sets.

- The BIRCH algorithm takes **as input a set of N data points, represented as real-valued vectors, and a desired number of clusters K**. It operates in four phases, the second of which is optional. tree, while removing outliers and grouping crowded subclusters into larger ones.

- Phase 1: Load data into memory
  Scan DB and load data into memory by building a CF tree. If memory is exhausted rebuild the tree from the leaf node.
- Phase 2: Condense data
  Resize the data set by building a smaller CF tree
  Remove more outliers
  Condensing is optional
- Phase 3: Global clustering
  Use existing clustering algorithm (e.g. KMEANS, HC) on CF entries
- Phase 4: Cluster refining
  Refining is optional
  Fixes the problem with CF trees where same valued data points may be assigned to different leaf entries.


Data
Phase 1: Load data into memory by building a CF tree
Initial CF tree
Phase 2: Condense data by building a smaller CF tree (optional)
Smaller CF tree
Phase 3: Global clustering
Good cluster
Phase 4: Cluster refinement (optional)
Better cluster

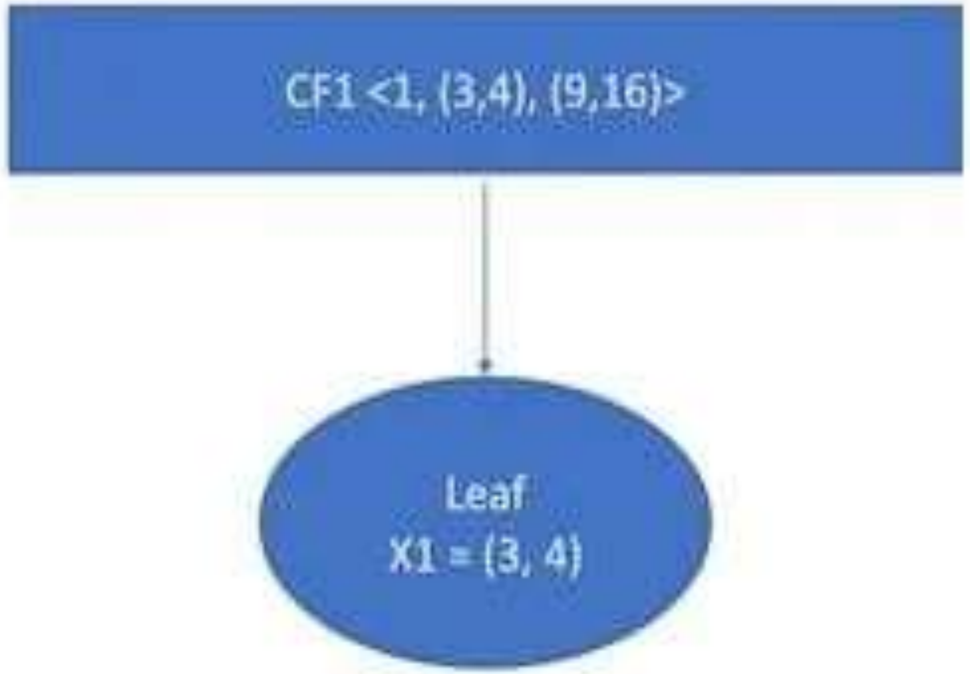
Example

Let Have Following Data
X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)
Cluster the Above Data Using BIRCH Algorithm, , considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and
Cluster Feature:

->Consider Data Pint (3,4):

As it is alone in the Feature map, Hence
1. Radius = 0
2. Cluster Feature CF1 <N, LS, SS>
   N = 1 as there is now one data point under
consideration.
   LS = Sum of Data Point under consideration = (3,4)
   SS = Square Sum of Data Point Under Consideration
   $= (3^2, 4^2)=(9,16)$
3. Now construct the Leaf with Data Point X1 and Branch
as CF1.



Example

Let Have Following Data
X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)
Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

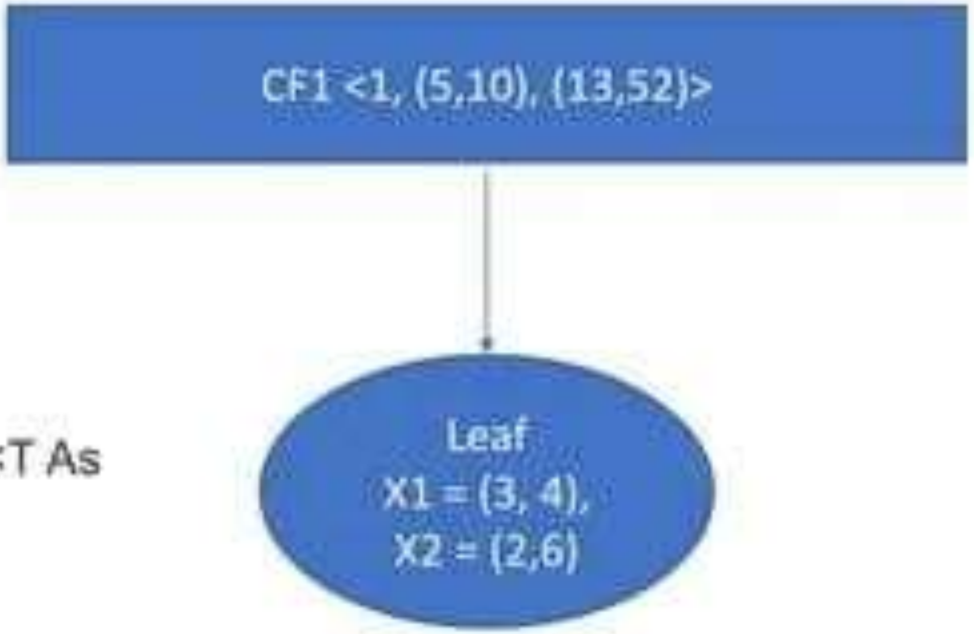
->Consider Data Pint x2 = (2,6):

1. Linear Sum LS = (3,4) + (2,6) = (5,10)
2. Square Sum SS = (3^2+2^2 , 4^2+6^2) =(13, 52)

Now Evaluate Radius considering N=2

$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(13,52)-(5,10)^2/2}{2}} = \sqrt{\frac{(13,52)-(25,100)/2}{2}} =$
$\sqrt{\frac{(13,52)-(12.5,50)}{2}} = \sqrt{(6.5,26)-(6.25,25)} = \sqrt{(0.25,1)}$ =(0.5, 1)<T As
(0.25,1) < (T, T), hence X2 will cluster with Leaf X1.

2. Cluster Feature CF1 <N, LS, SS> = <2,(5,10),(13,52)>
   N = 2 as there is now two data point under CF1.
   LS = (3,4) + (2,6) = (5,10)
   SS = (3^2+2^2 , 4^2+6^2) =(13, 52)



Example

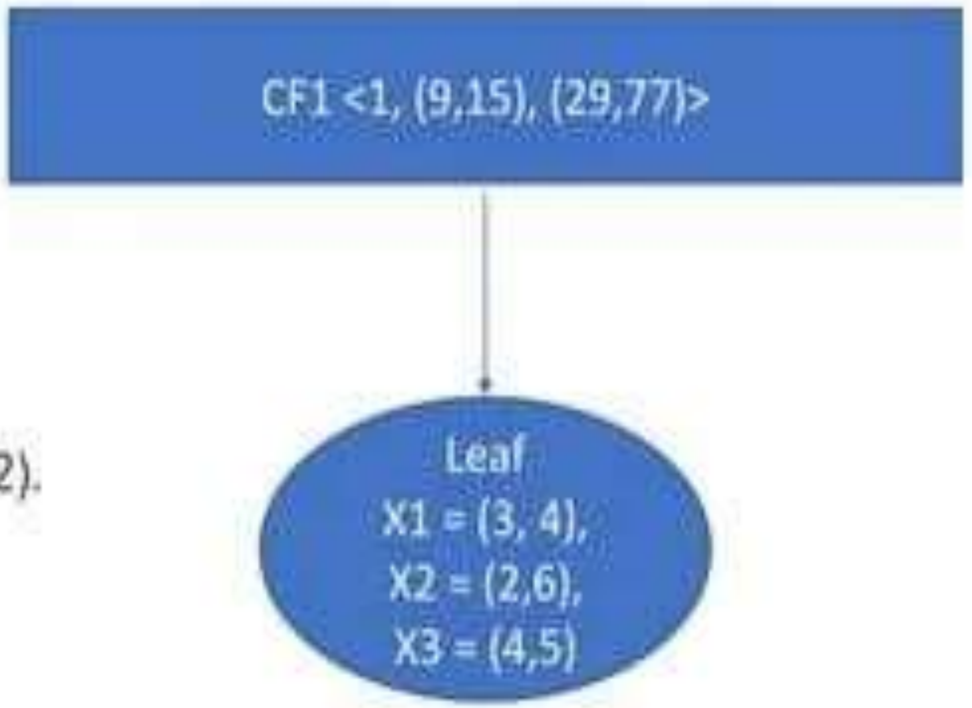
Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

->Consider Data Pint x3 = (4,5) on CF1:



1. Linear Sum LS = (4,5) + (5,10) = (9,15)
2. Square Sum SS = $(4^2+13\ ,\ 5^2+52)$ =(29, 77)

Now Evaluate Radius considering N=3

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(29,77)-(9,15)^2/3}{3}} = (0.47,\ 0.4714) < T$$

As (0.47, 0.471) < (T, T), hence X3 will cluster with Leaf (X1, x2).

2. Cluster Feature CF1 <N, LS, SS> = <3,(9,15),(29,77)>

   N = 3 as there is now Three data point under CF1.

   LS = (4,5) + (5,10) = (9,15)

   SS = $(4^2+13\ ,\ 5^2+52)$ =(29, 77)

Example

Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

->Consider Data Pint x4 = (4,7) on CF1:

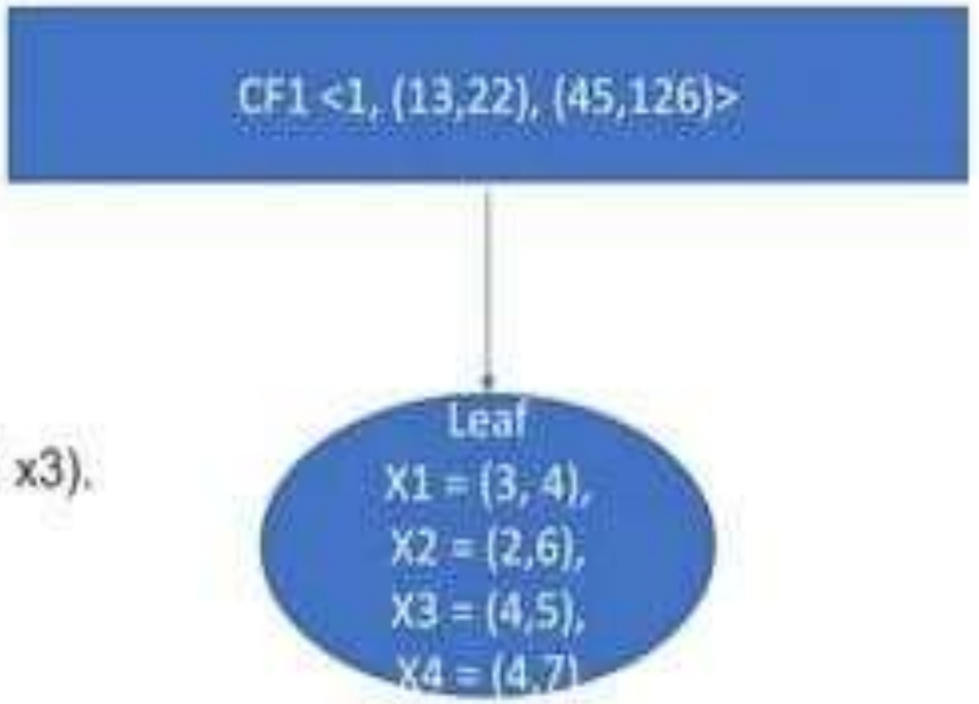
1. Linear Sum LS = (4,7) + (9,15) = (13,22)
2. Square Sum SS = (4^2+29 , 7^2 + 77) =(45, 126)

Now Evaluate Radius considering N=4

$$R = \sqrt{\frac{SS - LS^2/N}{N}} = \sqrt{\frac{(45,126)-(13,22)^2/4}{4}} = (0.41, 0.55)$$

As (0.41, 0.55) < (T, T), hence X4 will cluster with Leaf (X1, x2, x3).

2. Cluster Feature CF1 <N, LS, SS> = <4,(13,22),(45,126)>
   N = 4 as there is now four data point under CF1.
   LS = (4,7) + (9,15) = (13,22)
   SS = (4^2+29 , 7^2 + 77) =(45, 126)



Example

Let Have Following Data
X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)
Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

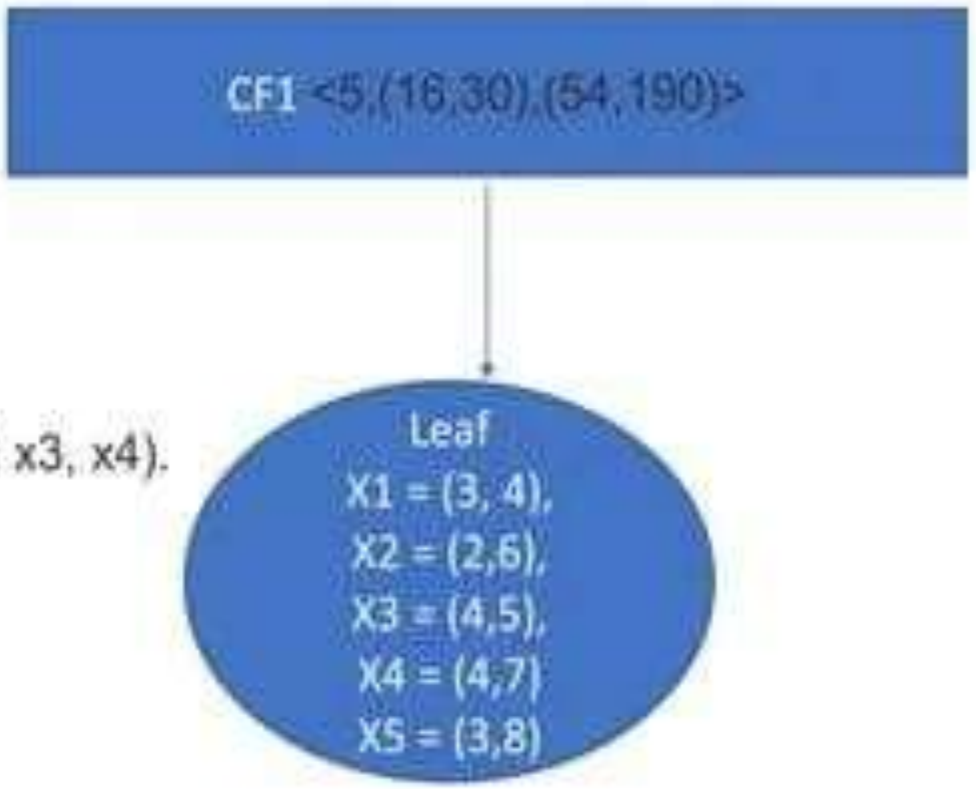
->Consider Data Pint x5 = (3,8) on CF1:

1. Linear Sum LS = (3,8) + (13,22) = (16,30)
2. Square Sum SS = (3^2+45 , 8^2 + 126) =(54, 190)

Now Evaluate Radius considering N=5

$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(54,190)-(16,30)^2/5}{5}}$ =(0.33, 0.63)

As (0.33, 0.63) < (T, T), hence X5 will cluster with Leaf (X1, x2, x3, x4).
2. Cluster Feature CF1 <N, LS, SS> = <5,(16,30),(54,190)>
N = 5 as there is now four data point under CF1.



Example

Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

->Consider Data Pint x6 = (6,2) on CF1:

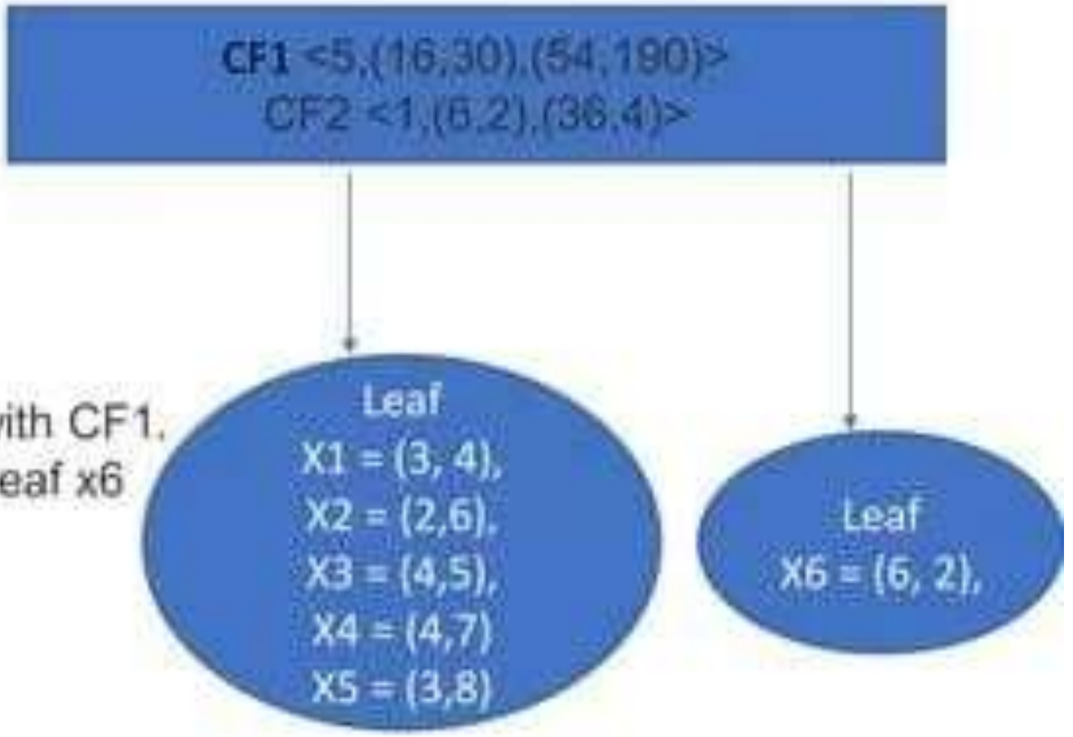
1. Linear Sum LS = (6,2) + (16,30) = (22,32)
2. Square Sum SS = (6^2+54 , 2^2 + 190) =(90, 194)

Now Evaluate Radius considering N=6

$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(90,194)-(22,32)^2/6}{6}}$ =(1.24, 1.97)

As (1.24, 1.97) < (T, T), False. hence X6 will Not form cluster with CF1. CF1 will remain as it was in previous step. And New CF2 with leaf x6 will be created.

2. Cluster Feature CF2 <N, LS, SS> = <1,(6,2),(36,4)>
   N = 1 as there is now one data point under CF2.
   LS = (6,2)
   SS = (6^2, 2^2)= (36,4)



Example

Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:
->Consider Data Pint x7 = (7,2). As There are Two Branch CF1 and CF2 hence we need to find with which branch X7 is nearer, then with that leaf radius will be evaluated.
With CF1 = LS/N= (16,30)/5=(8,6) As there are N=5 Data Point
With CF2 = LS/N= (6,2)/1=(6,2) As there is N=1 Data Point
Now x7 is closer to (6,2) then (8,6). Hence X7 will calculate radius with CF2.

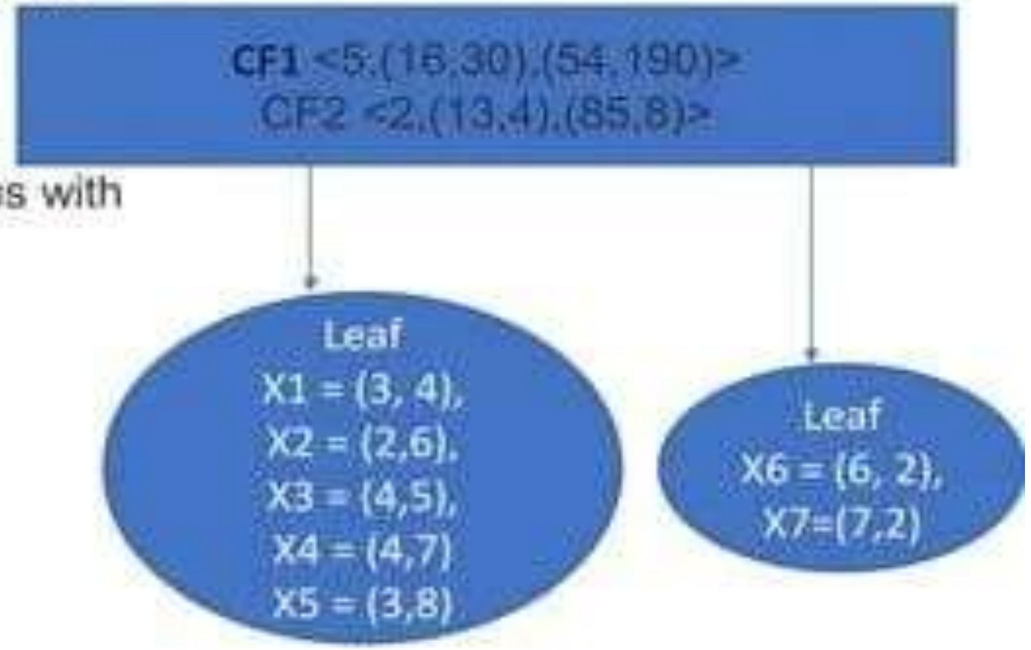
1. Linear Sum LS = (7,2) + (6,2) = (13,4)
2. Square Sum SS = (7^2+36 , 2^2 + 4) =(85, 8)

Now Evaluate Radius considering N=2

$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(85,8)-(13,4)^2/2}{2}}$ =(0.5, 0)

As (0.5, 0) < (T, T), True. hence X7 will form cluster with CF2

2. Cluster Feature CF2 <N, LS, SS> = <2,(13,4),(85,8)>
   N = 2 as there is now two data point under CF2.
   LS = (7,2) + (6,2) = (13,4)
   SS = (7^2+36 , 2^2 + 4) =(85, 8)



Example

Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4), x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

->Consider Data Pint x8 = (7,4). As There are Two Branch CF1 and CF2 hence we need to find with which branch X8 is nearer, then with that leaf, radius will be evaluated.

With CF1 = LS/N= (16,30)/5=(8,6) As there are N=5 Data Point

With CF2 = LS/N= (13,4)/2=(6.5,2) As there is N=2 Data Point

Now x8 is closer to (6.5,2) then (8,6). Hence X8 will calculate radius with CF2.

1. Linear Sum LS = (7,4) + (13,4) = (20,8)
2. Square Sum SS = (7^2+85 , 4^2 + 8) =(134, 24)

Now Evaluate Radius considering N=3

$$R = \sqrt{\frac{SS - LS^2/N}{N}} = \sqrt{\frac{(134,24)-(20,8)^2/3}{3}} = (0.47, 0.94)$$

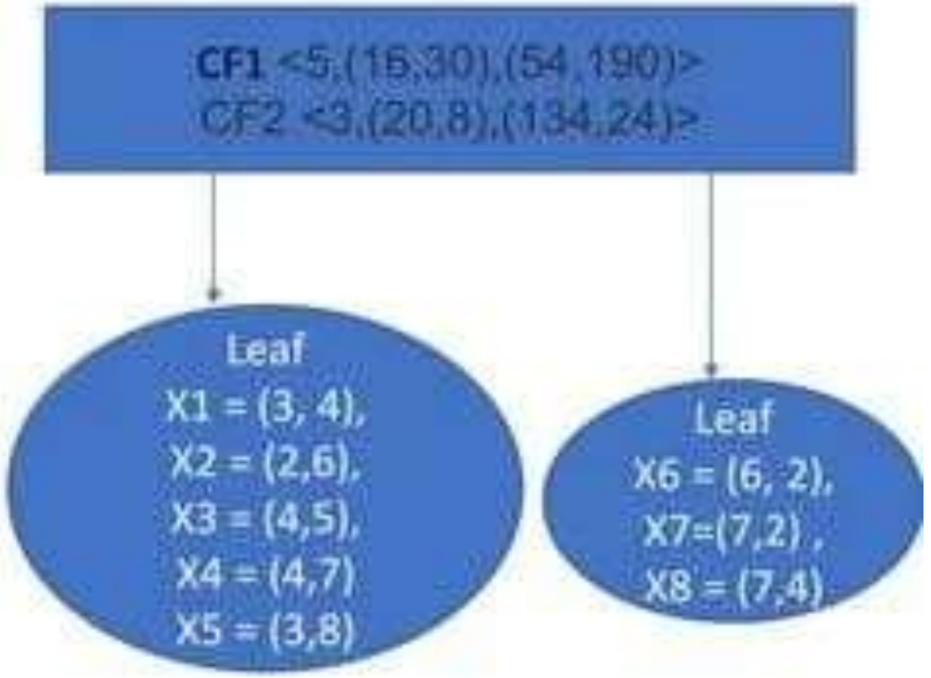
As (0.47, 94) < (T, T), True. hence X8 will form cluster with CF2

2. Cluster Feature CF2 <N, LS, SS> = <3,(20,8),(134,24)>

   N = 3 as there is now two data point under CF2.

   LS (7,4) + (13,4) = (20,8)

   SS = (134,24)



Example

Let Have Following Data

X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4) , x10=(7,9)

Cluster the Above Data Using BIRCH Algorithm, considering $T<1.5$, and Max Branch = 2

->Consider Data Pint x9 = (8,4). As There are Two Branch CF1 and CF2 hence we need to find with which branch X9 is nearer, then with that leaf, radius will be evaluated.

With CF1 = LS/N= (16,30)/5=(8,6) As there are N=5 Data Point

With CF2 = LS/N= (20,8)/3=(6.6,2.6) As there is N=3 Data Point

Now x9 is closer to (6.6,2.6) then (8,6). Hence X8 will calculate radius with CF2.

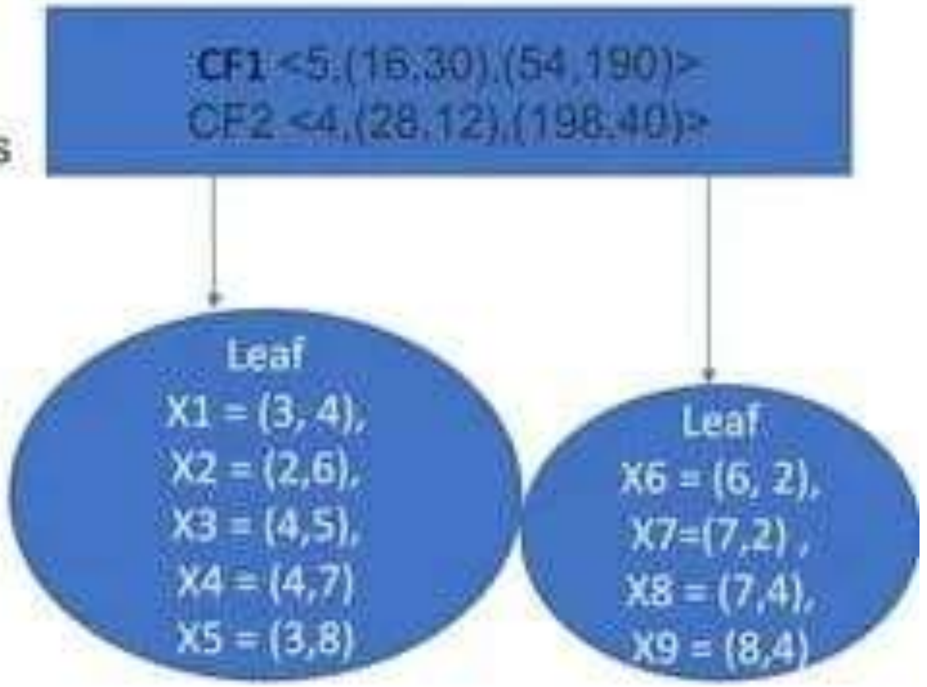
1. Linear Sum LS = (8,4) + (20,8) = (28,12)
2. Square Sum SS = $(8^2+134\ ,\ 4^2 + 24)$ =(198, 40)

Now Evaluate Radius considering N=4

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(198,40)-(28,12)^2/4}{4}} = (0.70,\ 1)$$

As (0.7, 1) < (T, T), True. hence X9 will form cluster with CF2

2. Cluster Feature CF2 <N, LS, SS> = <4,(28,12),(198,40)>
   N = 4 as there is now four data point under CF2.
   LS = (28,12)
   SS = (198,40)



Example

Let Have Following Data
X1=(3,4), x2= (2,6), x3=(4,5), x4=(4,7), x5=(3,8), x6=(6,2), x7=(7,2), x8=(7,4), x9=(8,4) , x10=(7,9)
Cluster the Above Data Using BIRCH Algorithm, considering T<1.5, and Max Branch = 2

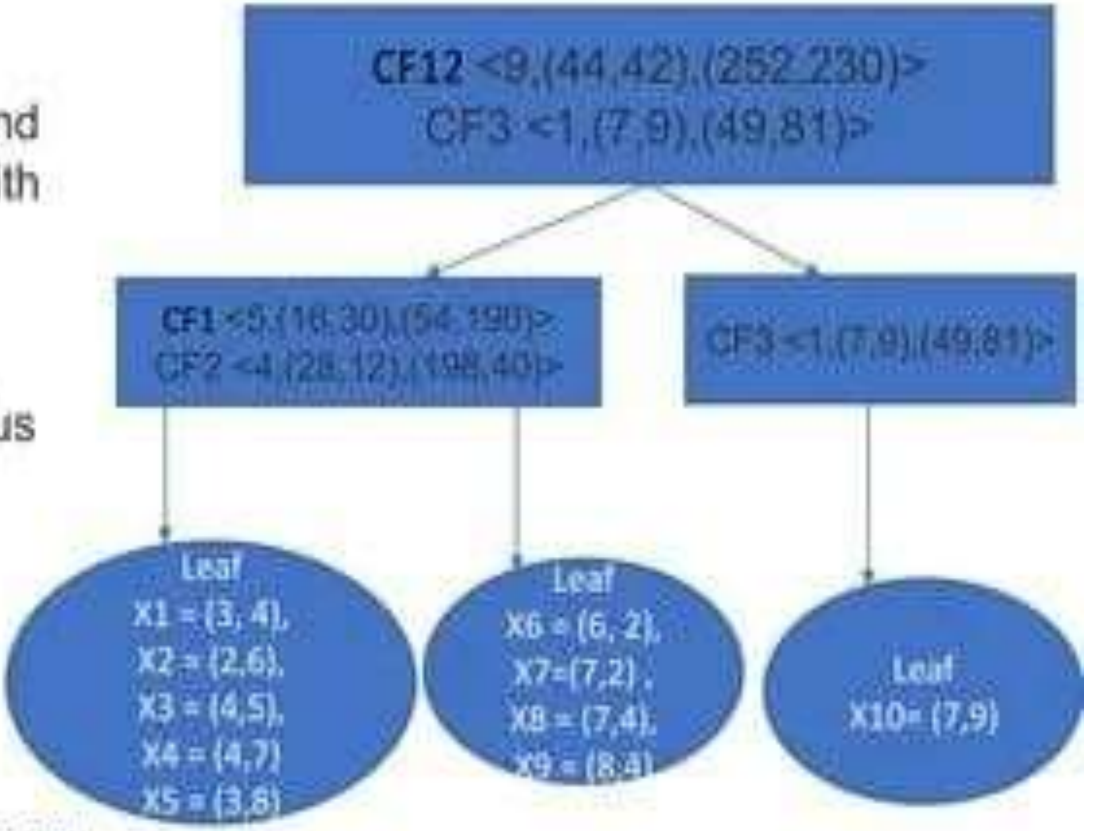
->Consider Data Pint x10 = (7,9). As There are Two Branch CF1 and CF2 hence we need to find with which branch X9 is nearer, then with that leaf, radius will be evaluated.
With CF1 = LS/N= (16,30)/5=(8,6) As there are N=5 Data Point
With CF2 = LS/N= (28,12)/4=(7,3) As there is N=4 Data Point
Now x10 is closer to (8,6) then (7,3). Hence X10 will calculate radius with CF1.

1. Linear Sum LS = (7,9) + (16,30) = (23,39)
2. Square Sum SS = (7^2+54 , 9^2 + 190) =(103, 271)

Now Evaluate Radius considering N=6

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(103,271)-(23,39)^2/6}{6}} = (1.57, 1.70)$$

As (1.57, 1.70) < (T, T). False. hence X10 will become new leaf and Create new cluster feature CF3. But in a Branch only two CF is allowed hence Branch will Split.
2. Cluster Feature CF3 <N, LS, SS> = <1,(7,9),(49,81)>


CF12 <9,(44,42),(252,230)>
CF3 <1,(7,9),(49,81)>
CF1 <5,(16,30),(54,190)>
CF2 <4,(28,12),(198,40)>
CF3 <1,(7,9),(49,81)>
Leaf
X1 = (3, 4),
X2 = (2,6),
X3 = (4,5),
X4 = (4,7)
X5 = (3,8)
Leaf
X6 = (6, 2),
X7=(7,2) ,
X8 = (7,4),
X9 = (8,4)
Leaf
X10= (7,9)