Machine Learning

&

Neural Networks

Introduction to Machine Learning

UNIT-1

Human learning

- Learning is typically referred to as the process of **Gaining Information** through **Observation.**

- For example, with **more knowledge**, the ability to do homework with **less number of mistakes** increases.
- In the same way, information from **past rocket launches** helps in taking the **right precautions** and **makes more successful rocket launch.**

- Thus, with **more learning**, tasks can be performed **more efficiently.**

Types ofHuman Learning

1. Learning under **expert** guidance

   -Either somebody who is an **expert** in the subject directly teaches us

2. Learning guided by **knowledge** gained from experts

   -We build our own notion indirectly based on what we have **learnt** from the **expert** in the **past**

   -There is **no direct learning**. It is some past information shared on some different context, which is used as a learning **to make decisions.**

3. Learning by **self**

   -We do it **ourselves**, may be after **multiple attempts**, some being **unsuccessful.**

   -Humans are left to learn on **their own**

WHAT IS MACHINE LEARNING?

- **According to Tom M.Mitchell…**

-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.**

- A machine can be considered to learn if it is able to **learn from experience (past data)** by doing a **certain task** and improve its **performance** in doing the **similar tasks in the future.**

Example: *Prediction of a person belongs to obesity category or not*

- **E** represents the past data having labels or assigned classes (for example whether the person of **a class obesity** or **not**), **T** is the task of assigning class or label to new data and **P** is the performance measure indicated by the percentage of **obesity people correctly** classified.

WHAT IS MACHINE LEARNING?

- A subset of **Artificial intelligence** known as machine learning focuses primarily on the creation of **model** that enable a computer to independently **learn from data** and **previous experiences** **to** **make predictions**

or

- Machine Learning is a branch of **Artificial intelligence** that develops **models** by learning the **hidden patterns** of the datasets used it to make **predictions on new similar type data**, without being explicitly programmed for each task.

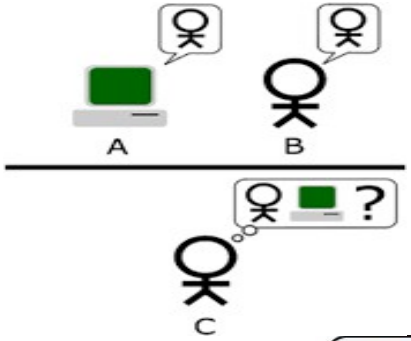
or

- Machine learning is a subfield of **Artificial intelligence** that uses **algorithms** **trained on data sets** to **create models** that enable machines to perform **tasks** that would otherwise only be possible for humans, such as categorizing images, analyzing data, or predicting price fluctuations.

or

- A "Field of study thatgives computers the capability **to learn** without being explicitly programmed".

Evolution or History of machine learnin



E JOURNEY OF MACHINE LEARNING

1950 Alan Turing proposes "learning machine"

1952 Arthur Samuel developed **first machine learning program** that could **play Checkers**

1957 Frank Rosenblatt designed the **first neural network program** simulating human brain

1967 **Nearest neighbour algorithm created** – start of basic pattern recognition

1979 Stanford University students **develop first self – driving cart** that can navigate and avoid obstacles in a room

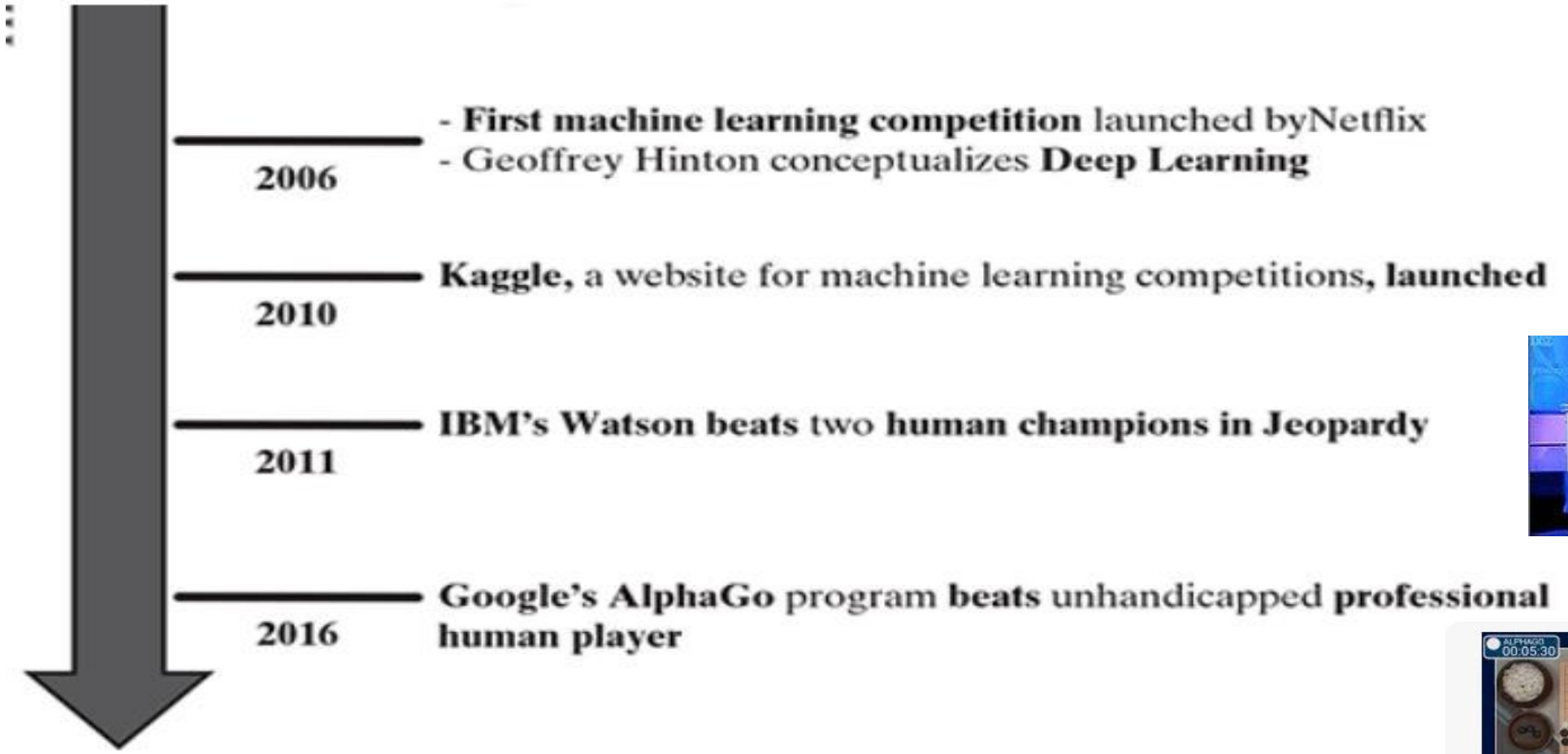
1982 **Recurrent Neural Network** developed

1989 - **Reinforcement Learning** conceptualized
- **Beginning of commercialization** of Machine Learning

1995 **Random Forest** and **Support Vector machine** algorithms developed

1997 **IBM's Deep Blue beats** the world chess champion Gary Kasparov

Evolution of machine learning (Continue..)



Popular Mechanics
AI AlphaGo Beats World Champion ...

FIG. 1.1 Evolution of machine learning

Types of machine Learning

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

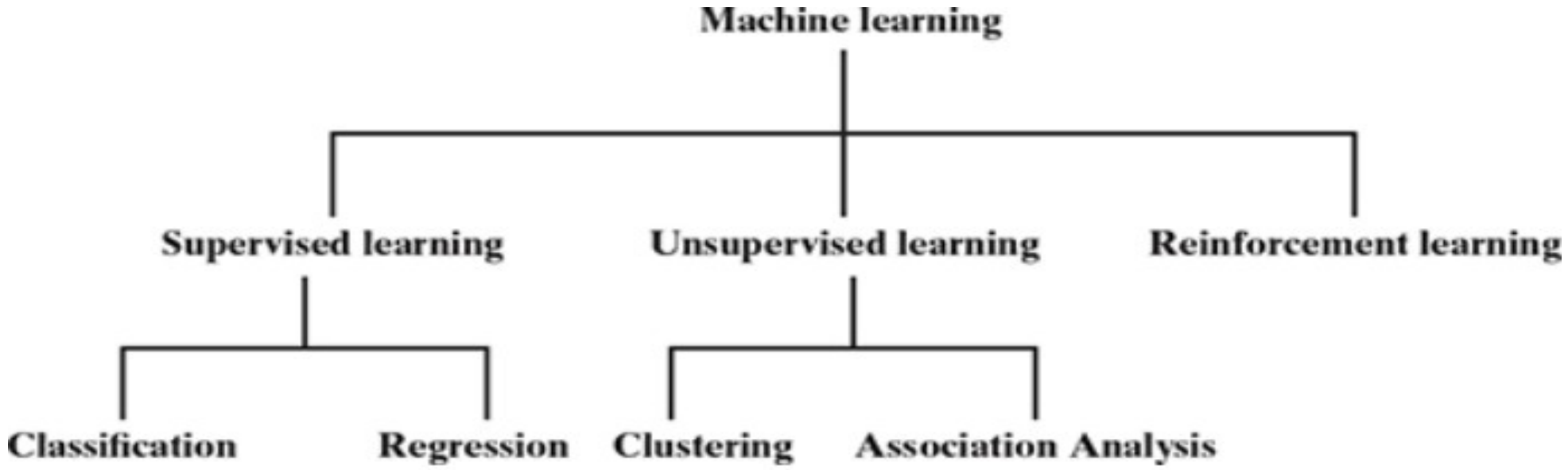
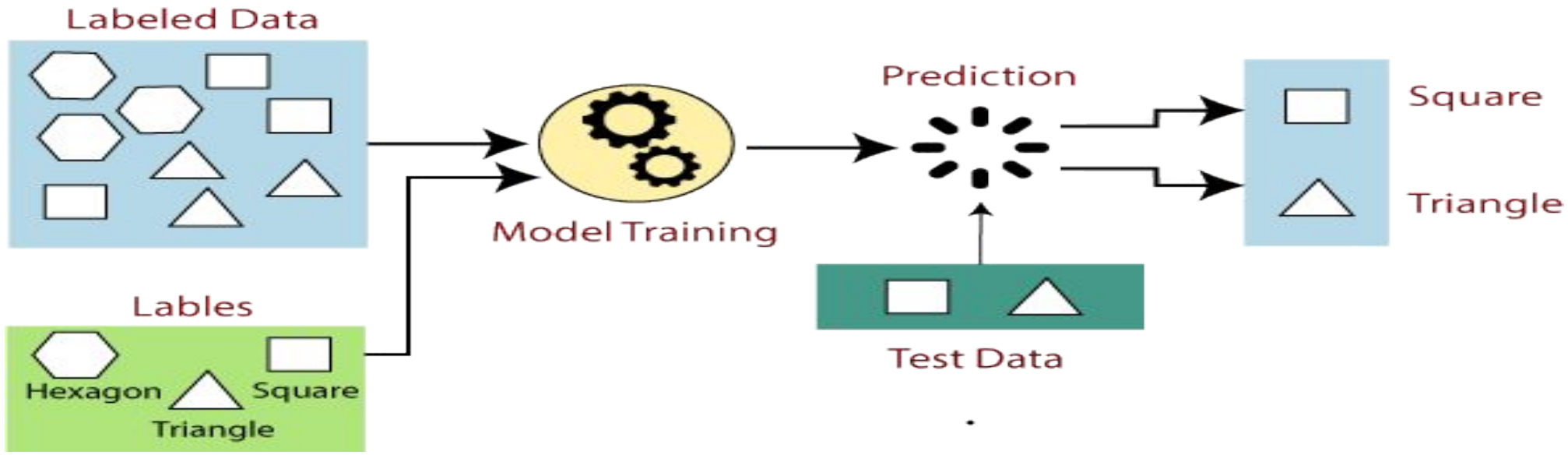


FIG. 1.3 Types of machine learning

Supervised Learning

- Supervised learning is a machine learning method in which **models** are **trained** using **labeled data.**
- In supervised learning technique, we train the machines using the **"labelled" dataset, and based on the training,** the machine **predicts** the **output.**
- Here, the **labelled data specifies** that some of the **inputs** are already mapped to **the output.** More preciously, we can say; first, we **train** the machine with **the input** and **corresponding output (train dataset)** , and then we ask the machine to **predict** the **output** using the **test dataset.**



Supervised Learning

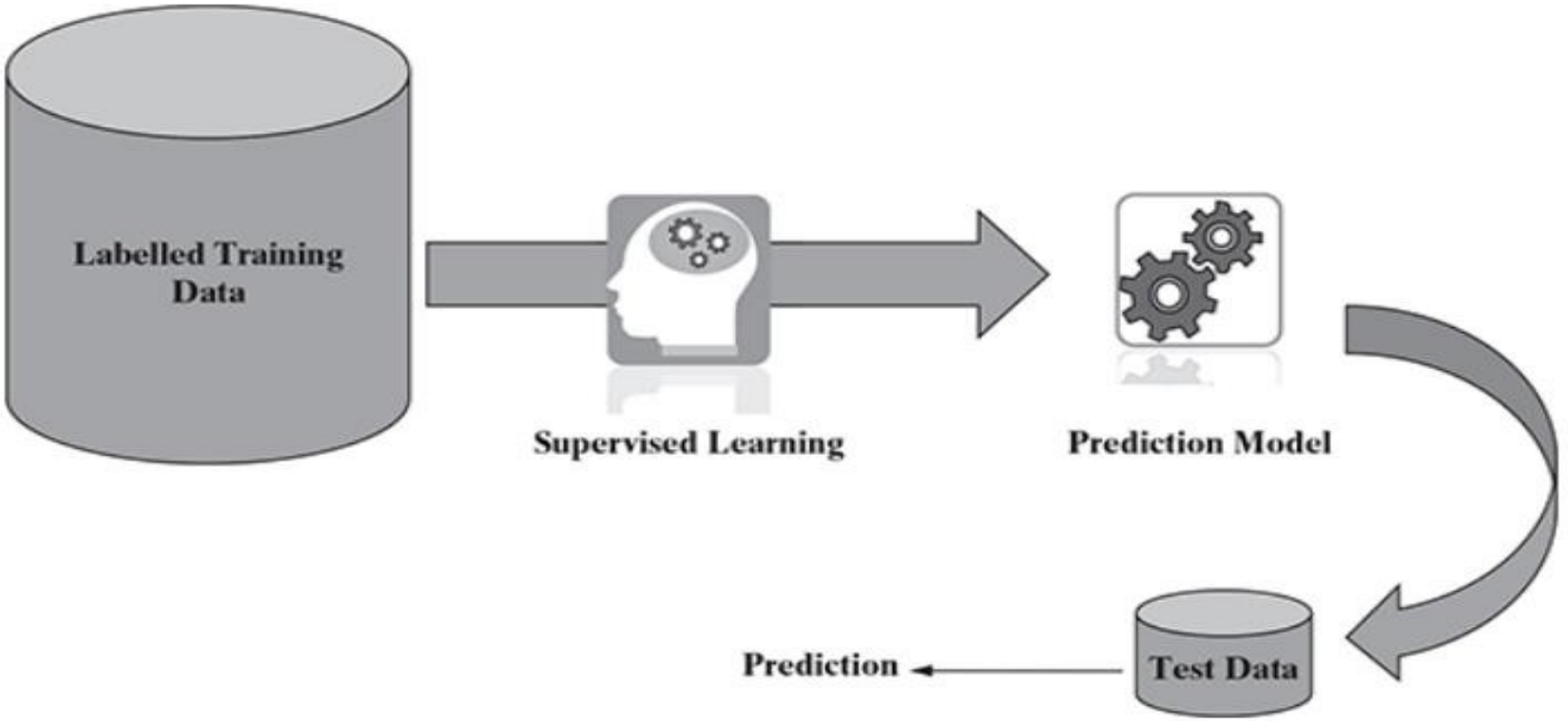


FIG. 1.4 Supervised learning

Some examples of supervised learning are :

- Predicting the results of a game
- Predicting whether a tumour is malignant or benign
- Predicting the price of domains like real estate, stocks, etc.
- Classifying texts such as classifying emails as spam or non-spam

Dr. Durga Prasad K, Department of

Supervised Learning Algorithms

1. Classification

Classification is a **supervised machine learning** method where the **model** tries to **predict** the **correct label** of a given input data. In classification, the model is fully trained using the **training data**, and then it is evaluated on **test data** before being used to perform **prediction** on **new unseen data**.

Applications:

Disease Diagnosis

Fraud Detection

Sentiment Analysis

Supervised Learning Algorithms

2. Regression

Regression is a method for understanding the **relationship** between **independent variables** or **features** and a **dependent variable** or **outcome.**

- **Linear Regression**

In Linear regression, the objective is to **predict numerical features** like real estate or stock price, temperature, marks in an examination, sales revenue, etc. The underlying **predictor variable** and the **target variable** are **continuous** in nature.

- **Logistic Regression**

Logistic regression predicts the output of a **categorical dependent variable**. Therefore the outcome must be a **categorical** or **discrete value.** It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as **0 and 1**, it gives the **probabilistic values** which lie between **0 and 1.**

Supervised Learning Algorithms

Typical applications of regression can be seen in..

- Sales prediction for managers
- Price prediction in real estate
- Weather forecast
- Skill demand forecast in job market
- Demand forecasting in retails

Unsupervised Learning

- Unlike supervised learning, in unsupervised learning, there is **no labelled training data** to **learn from** and **no prediction to be made.**
- In unsupervised learning, the models are **trained with the data** that is neither **classified nor labelled**, and the model acts on that data without any supervision.
- The main aim of the **unsupervised learning algorithm** is **to group or categories the unsorted dataset** according to **the similarities, patterns, and differences.** Machines are instructed to find the

hidden patterns from the input dataset.

1. **Clustering**
2. **Association Analysis**

Unsupervised Learning Algorithms

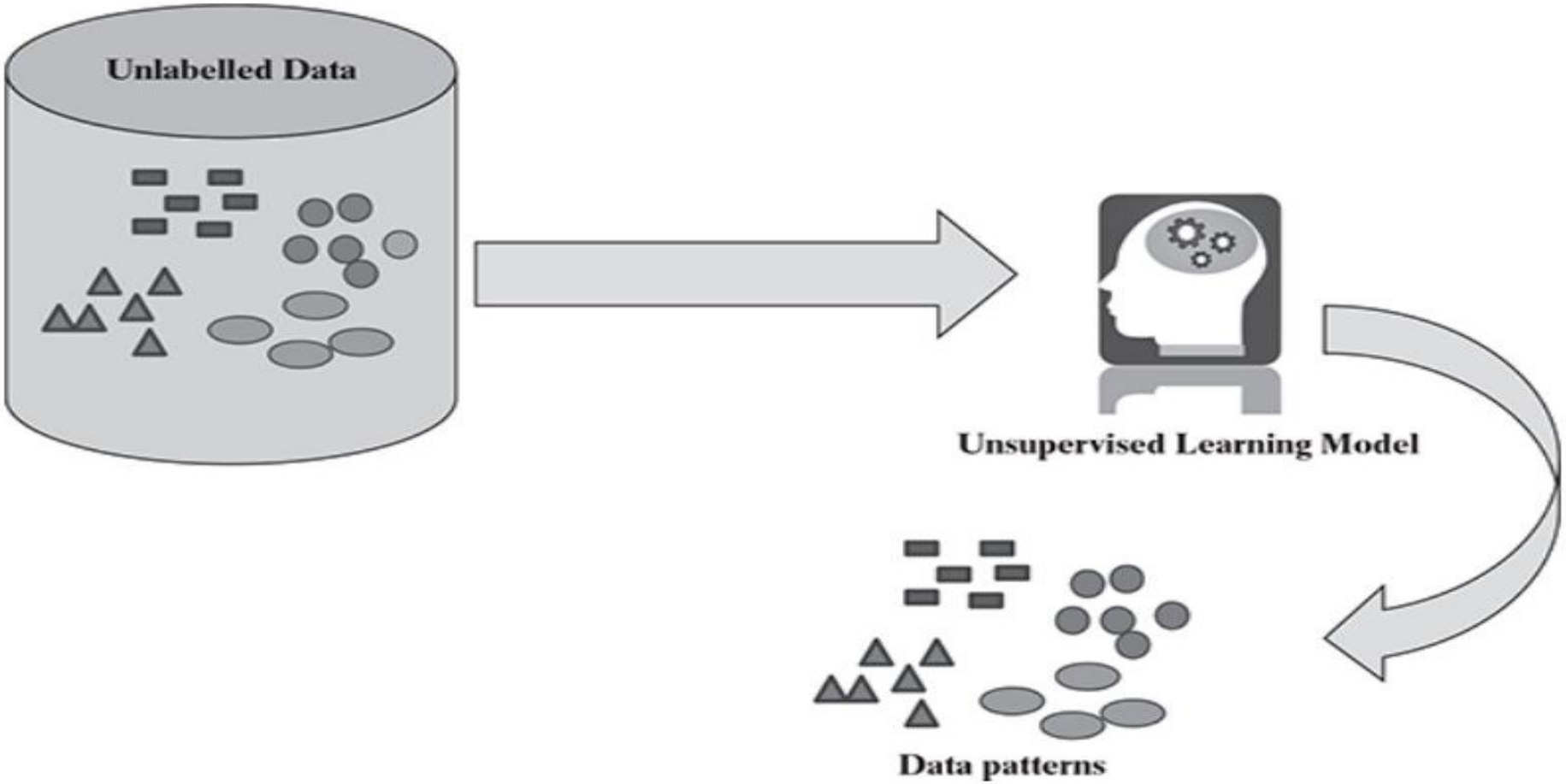
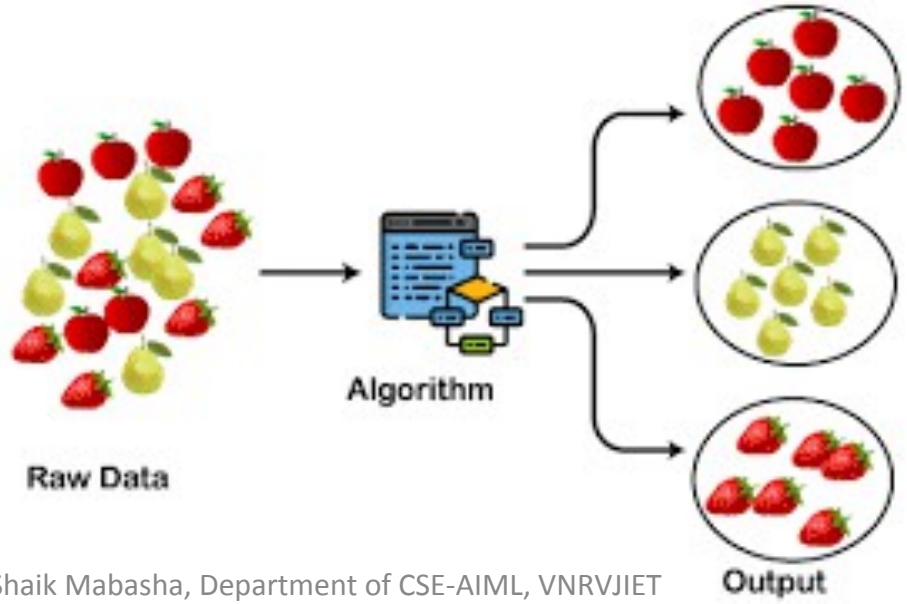


FIG. 1.8 Unsupervised learning

Unsupervised Learning Algorithms

1. Clustering

- Clustering is the main type of unsupervised learning.
- It intends to group or organize **similar objects** or **data points** together.
- For that reason, objects belonging to the same cluster are quite similar to each other w rent clusters are quite dissimilar.



Unsupervised Learning Algorithms

2. Association Analysis

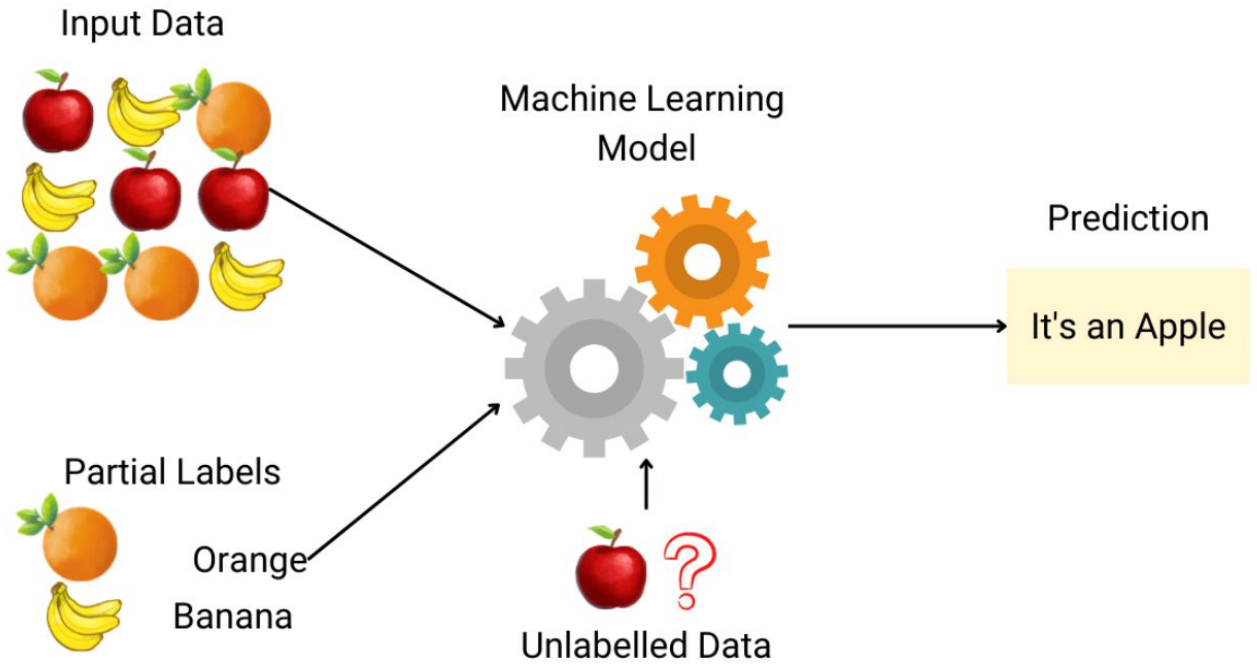
- An Association Analysis or Association rule is an unsupervised learning method used to find relationships between different items in a dataset.
- It determines the set of items that occurs together in the dataset. *
- Association rule makes marketing strategy more effective.
- Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item.
- A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning Algorithms

- Hierarchical clustering
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- K-medoids
- K-means

Semi supervised Learning

- In semi-supervised learning, the model learns from a **small amount of labeled data** along with a **large amount of unlabeled data**, combining supervision with self-discovery to **improve prediction accuracy**.



Reinforcement Learning

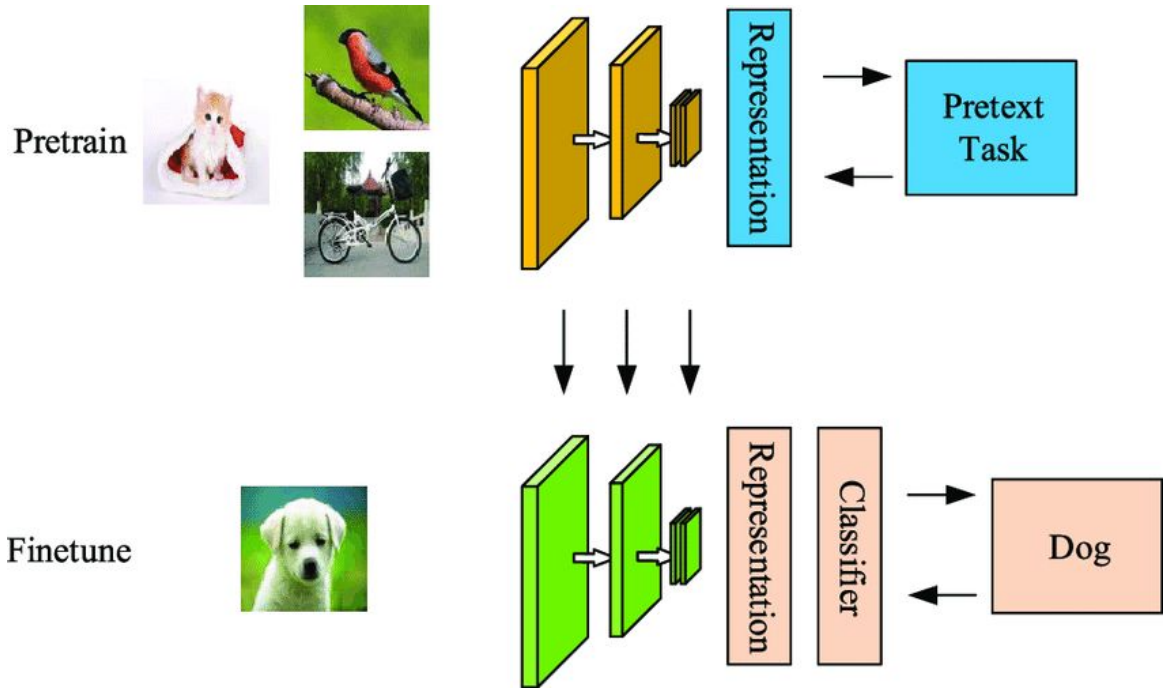
- **Reinforcement Learning** is a **feedback-based Machine learning technique** in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike **supervised learning.**

Or

- **Reinforcement learning (RL)** is defined as a **sub-field** of **machine learning** that enables AI-based systems to take actions in a dynamic environment through **trial and error methods** to **maximize** the collective rewards based on the **feedback generated** for respective actions.

Self supervised Learning

- In self-supervised learning, the model **generates its own labels from the input data** by creating pretext tasks, enabling it to learn useful representations without requiring human-annotated labels.



Comparison – supervised, unsupervised, and reinforcement learning

SUPERVISED	UNSUPERVISED	REINFORCEMENT
This type of learning is used when you know how to classify a given data, or in other words classes or labels are available.	This type of learning is used when there is no idea about the class or label of a particular data. The model has to find pattern in the data.	This type of learning is used when there is no idea about the class or label of a particular data. The model has to do the classification – it will get rewarded if the classification is correct, else get punished.
Labelled training data is needed. Model is built based on training data.	Any unknown and unlabelled data set is given to the model as input and records are grouped.	The model learns and updates itself through reward/ punishment.
The model performance can be evaluated based on how many misclassifications have been done based on a comparison between predicted and actual values.	Difficult to measure whether the model did something useful or interesting. Homogeneity of records grouped together is the only measure.	Model is evaluated by means of the reward function after it had some time to learn.
There are two types of supervised learning problems – classification and regression.	There are two types of unsupervised learning problems – clustering and association.	No such types.
Simplest one to understand.	More difficult to understand and implement than supervised learning.	Most complex to understand and apply.

Standard algorithms include

- Naïve Bayes
- *k*-nearest neighbour (kNN)
- Decision tree
- Linear regression
- Logistic regression
- Support Vector Machine SVM), etc.

Standard algorithms are

- *k*-means
- Principal Component Analysis (PCA)
- Self-organizing map (SOM)
- Apriori algorithm
- DBSCAN etc.

Standard algorithms are

- Q-learning
- Sarsa

Practical applications include

- Handwriting recognition
- Stock market prediction
- Disease prediction
- Fraud detection, etc.

Practical applications include

- Market basket analysis
- Recommender systems
- Customer segmentation, etc.

Practical applications include

- Self-driving cars
- Intelligent robots
- AlphaGo Zero (the latest version of DeepMind's AI system playing Go)

Problems Not To Be Solved Using Machine Learning

- Machine learning should not be applied to tasks in which humans are very effective or frequent human intervention is needed.
- For example, Air traffic control is a very complex task needing intense human involvement.
- At the same time, for very simple tasks which can be implemented using traditional programming paradigms, there is no sense of using machine learning.
- For example, simple rule-driven or formula-based applications like price calculator engine, dispute tracking application, etc. do not need machine learning techniques.

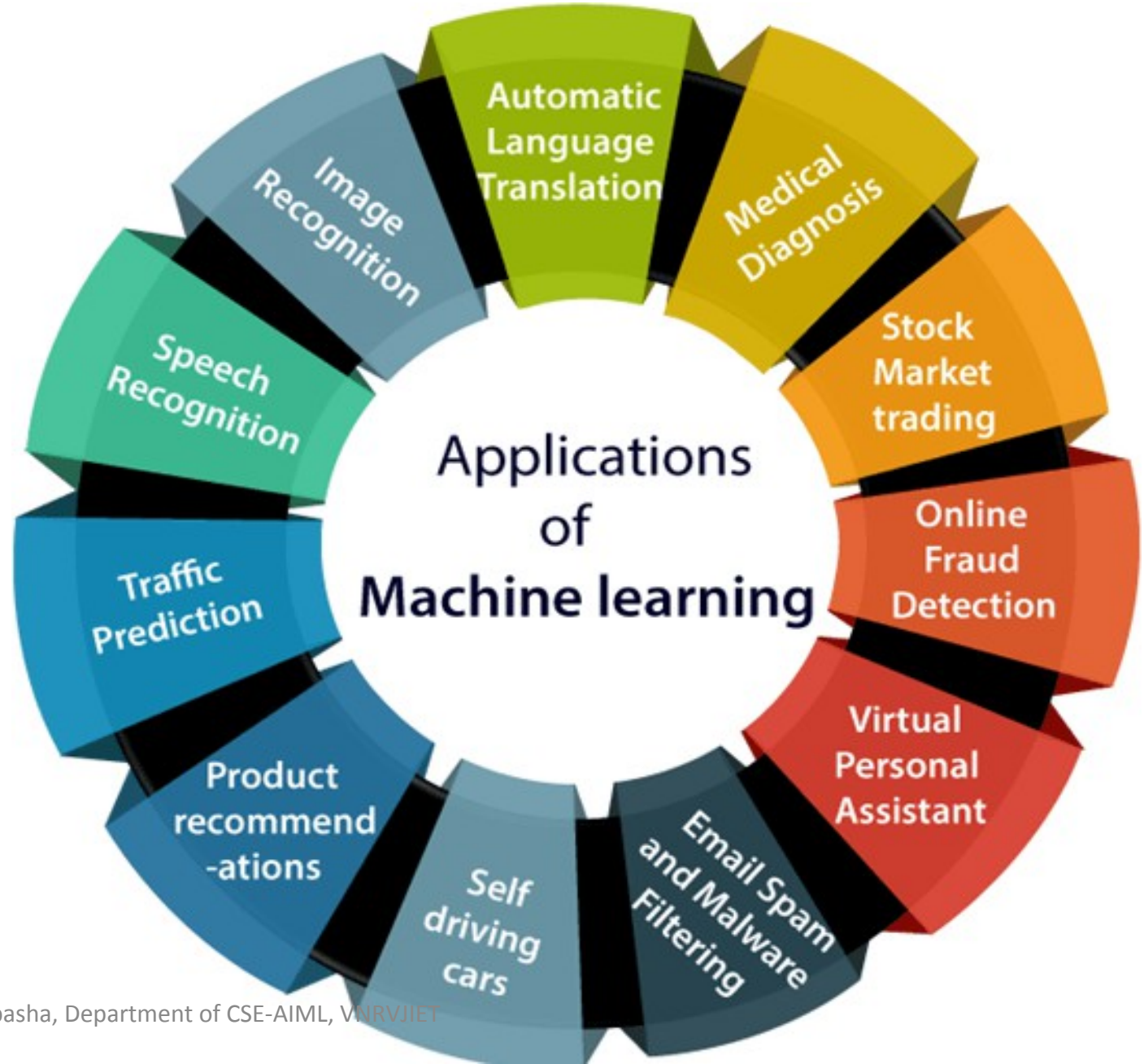
Problems Not To Be Solved Using Machine Learning

- For situations where training data is not sufficient, machine learning cannot be used effectively.
- This is because, with small training data sets, the impact of bad data is exponentially worse.
- For the quality of prediction or recommendation to be good, the training data should be sizeable.

Applications of Machine Learning

Major Domains:

- Banking and finance
- Healthcare
- Agriculture
- Robotics
- Social Media Analysis
- Gaming
- Insurance
- Vision systems



Tools In Machine Learning

- **Python-** Python is a high-level programming language , used to create a range of applications, including data science, software and web development, automation, and improving the ease of everyday tasks.
  - **NumPy- for mathematical functionalities**

  -For fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
  - **SciPy- Algorithms and advanced mathematical tools**

  - Provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems.
  - **matplotlib - Numerical plotting**

  -Matplotlib is an amazing visualization library in Python for **2D plots** of arrays. Matplotlib is a multi-platform **data visualization library** built on NumPy. Matplotlib consists of several **plots like line, bar, scatter, histogram, etc.**
  - **Scikit-learn or sklearn** - A machine learning library, **classification, regression, and clustering algorithms** embedded in it.

- **R**

-R is a language for **statistical computing** and **data analysis.** It is an open source language, extremely popular

Tools In Machine Learning

- **Matlab**

MATLAB (matrix laboratory) is a licensed commercial software with a robust support for a wide range of **numerical computing.** MATLAB also provides extensive support of **statistical functions** and has a huge number of machine learning algorithms in-built. It also has the ability to scale up for **large datasets by parallel processing** on clusters and cloud.

- **SAS (earlier known as 'Statistical Analysis System')**

is another licensed commercial software which provides strong support for machine learning functionalities.

- **IBM, SPSS (originally named as Statistical Package for the Social Sciences)**

is a popular package supporting specialized data mining and statistical analysis. Originally popular for statistical analysis in social science (as the name reflects), SPSS is now popular in other fields as well.

- **Julia**

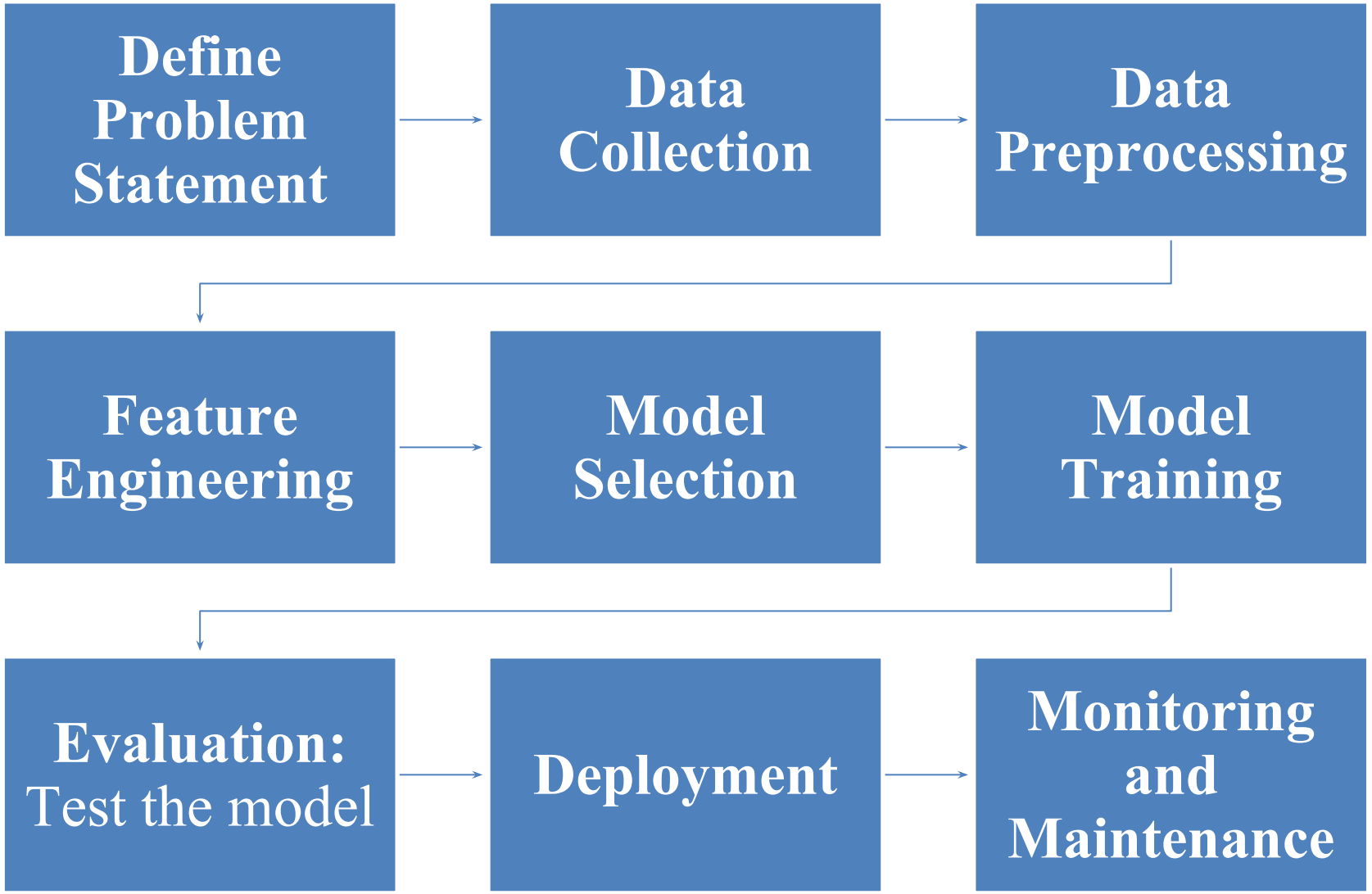
is an open source, liberal license programming language for **numerical analysis** and **computational science.**

It has baked in all good things of **MATLAB, Python, R**, and other programming languages used for machine learning for which it is gaining steady attention from machine learning development community. Another big point in favor of Julia is its ability to implement **high performance** machine learning algorithms.

Issues In Machine Learning

- Machine learning is a field which is relatively new and still evolving.
- Also, the level of research and kind of use of machine learning tools and technologies varies drastically from country to country.
- The biggest fear and issue arising out of machine learning is related to privacy and the breach of it.
- The primary focus of learning is on analyzing data, both past and current, and coming up with insight from the data. This insight may be related to people and the facts revealed might be private enough to be kept confidential.
- Even if there is no breach of privacy, there may be situations where actions were taken based on machine learning may create an adverse reaction.

Steps in a Machine Learning Workflow



Supervised Learning: Classification

Examples Of Supervised Learning

- In supervised learning, the **labelled training data** provide the basis for **learning.**
- According to the definition of machine learning, this **labelled training data** is **the experience** or **prior knowledge** or **belief.**
- **Training data** is the **past information** with known value of **class field** or '**label'.** Hence, we say that the **'training data is labelled'** in the case of **supervised learning.**

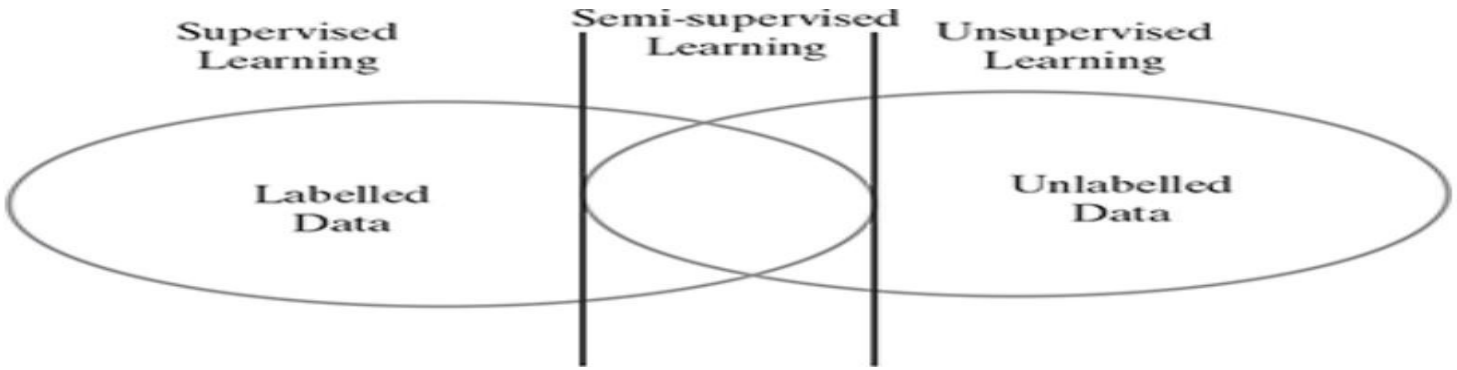


FIG. 7.1 Supervised learning vs. unsupervised learning

Classification Model

Classification

- Let us consider two examples, say '**predicting whether a tumour is malignant or benign**' and '**price prediction in the domain of real estate**'.

Are these two problems same in nature?

- When we are trying to predict a categorical or nominal variable, the problem is known as a **classification problem.**
- Whereas when we are trying to predict a numerical variable such as 'price', 'weight', etc. the problem falls under the **category of regression.**
- In classification, the whole problem centres around assigning a label or category or class to a test data on the basis of the label or category or class information that is imparted by the training data.
- Because the target objective is to assign **a class label**

Classification Model

Classification

Classification is a **supervised machine learning** method. In Classification, a program **learns** from the **given dataset** or **observations** and then **classifies new observation** into a number of **classes or groups**. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc.** Classes can be called as **targets/labels** or **categories.**

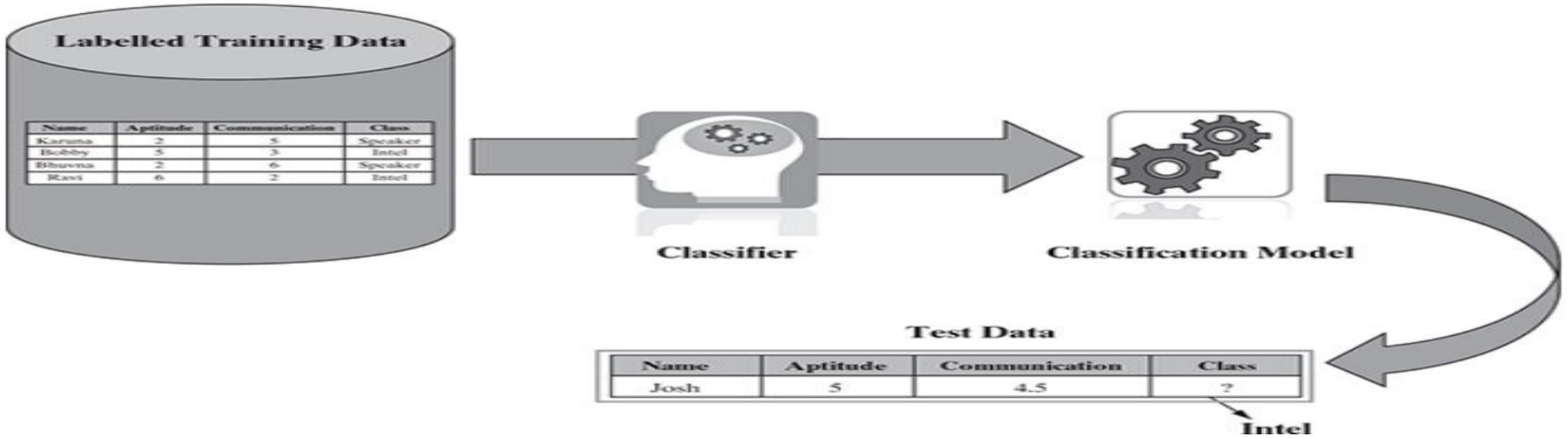


FIG. 1.5 Classification

Types and Examples

Type	Classes	Examples
Binary	2 classes	Spam/Not Spam, Malignant/Benign
Multi-Class	3+ classes	Handwritten Digits, Iris Flower Types
Multi-Label	Multiple classes	Image Tagging, Movie Genres
Imbalanced	Unequal class sizes	Fraud Detection, Rare Disease Diagnosis
Ordinal	Ordered classes	Star Ratings, Severity Levels
Hierarchical	Class hierarchy	Animal Taxonomy, Product Categorization
Streaming	Real-time	Real-Time Spam Filtering
One-Class	Single-class focus	Anomaly Detection, Fraud Detection
Time-Series	Sequential	Activity Recognition, Trend Classification

Supervised Learning Algorithms

Some typical classification problems include:

- **Image** classification
- Prediction of **disease**
- Win–loss **prediction of games**
- Predictionof natural calamity like **earthquake, flood,** etc.
- Recognition of **handwriting**

Supervised Learning- Classification Algorithms

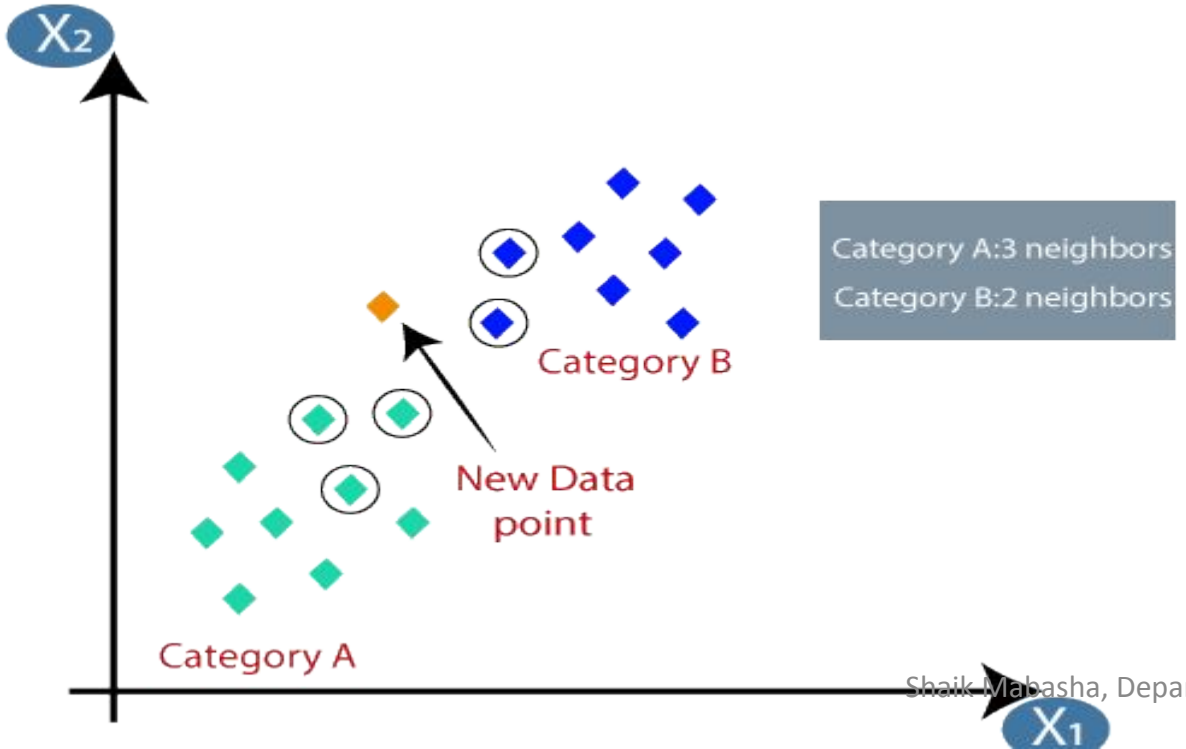
- **K-Nearest Neighbour**
- **Decision Tree**
- **RandomForest**
- **Support Vector Machine**

K-Nearest Neighbour Classifier

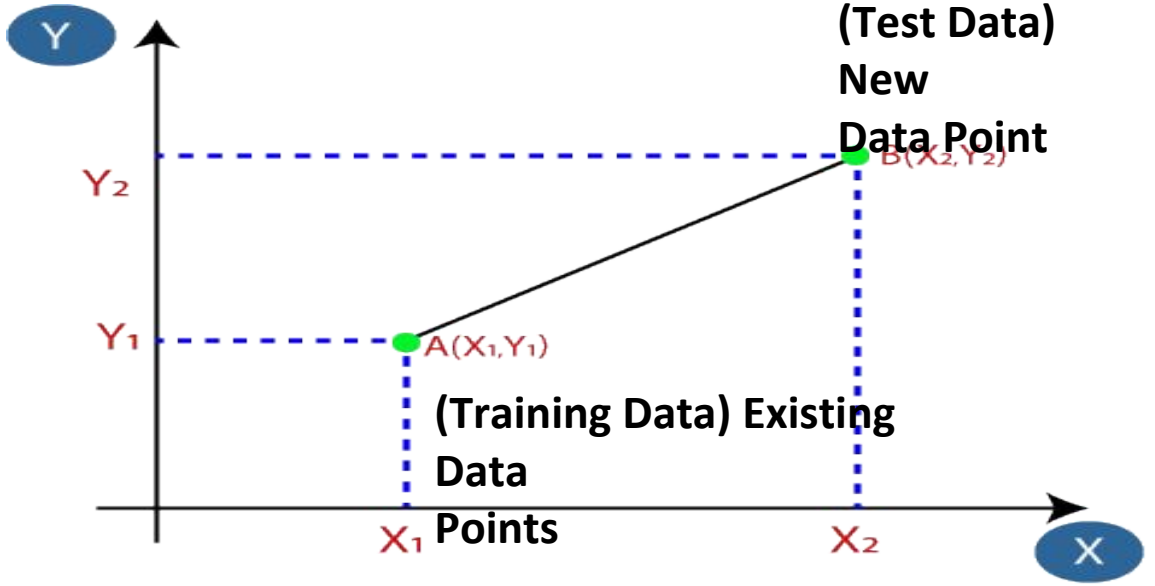
- The **K-Nearest Neighbors (KNN)** algorithm is a popular machine learning technique used for classification tasks. It relies on the idea that **similar data points** tend to have **similar labels** or **values.**
- The kNN algorithm is a simple but extremely powerful **classification algorithm**. The name of the algorithm originates from the underlying philosophy of kNN – i.e. people having **similar background or mindset** tend to stay **close to each other.**
- During thetraining phase,theKNN algorithm **stores** theentire **training dataset** as a **reference**.
- When making predictions, it **calculates** the **distance** between the **input data point** and all the **training examples,** using a chosen distance metric such as **Euclidean distance.**

K-Nearest Neighbour Classifier

- Next, the algorithm identifies the **K nearest neighbors** to the **input data point** based on their **distances.**
- In the case of classification, the algorithm assigns the most common class label among the K neighbors as the predicted label for the input data point.



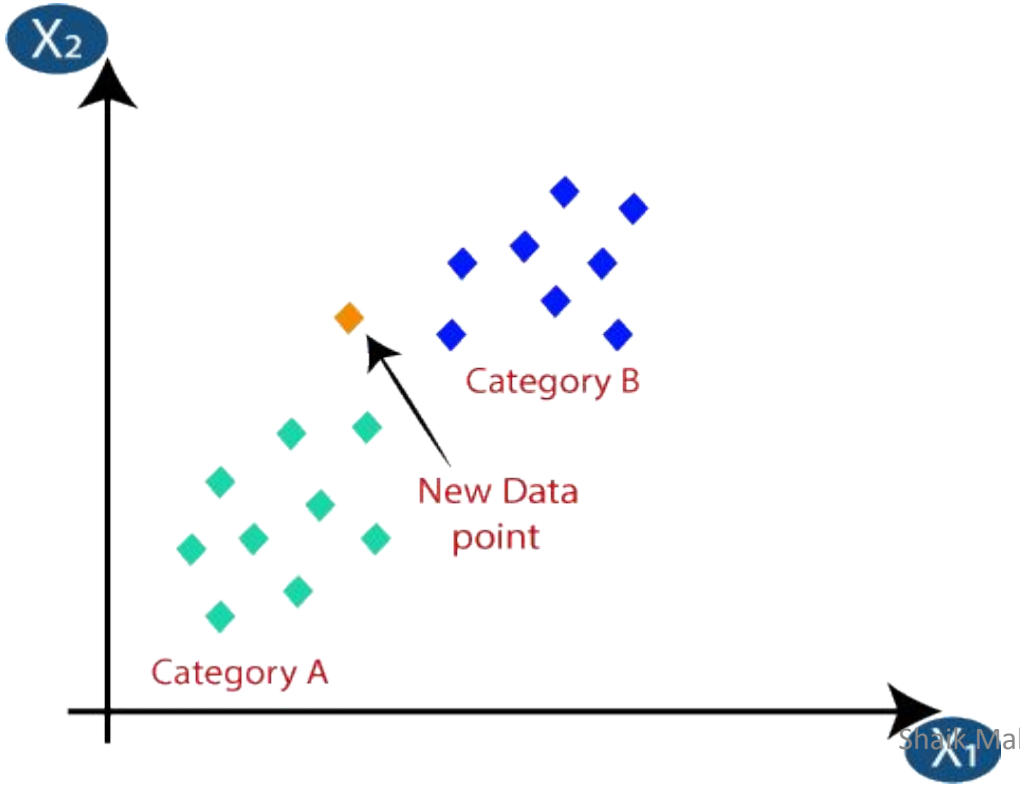
Category A:3 neighbors
Category B:2 neighbors

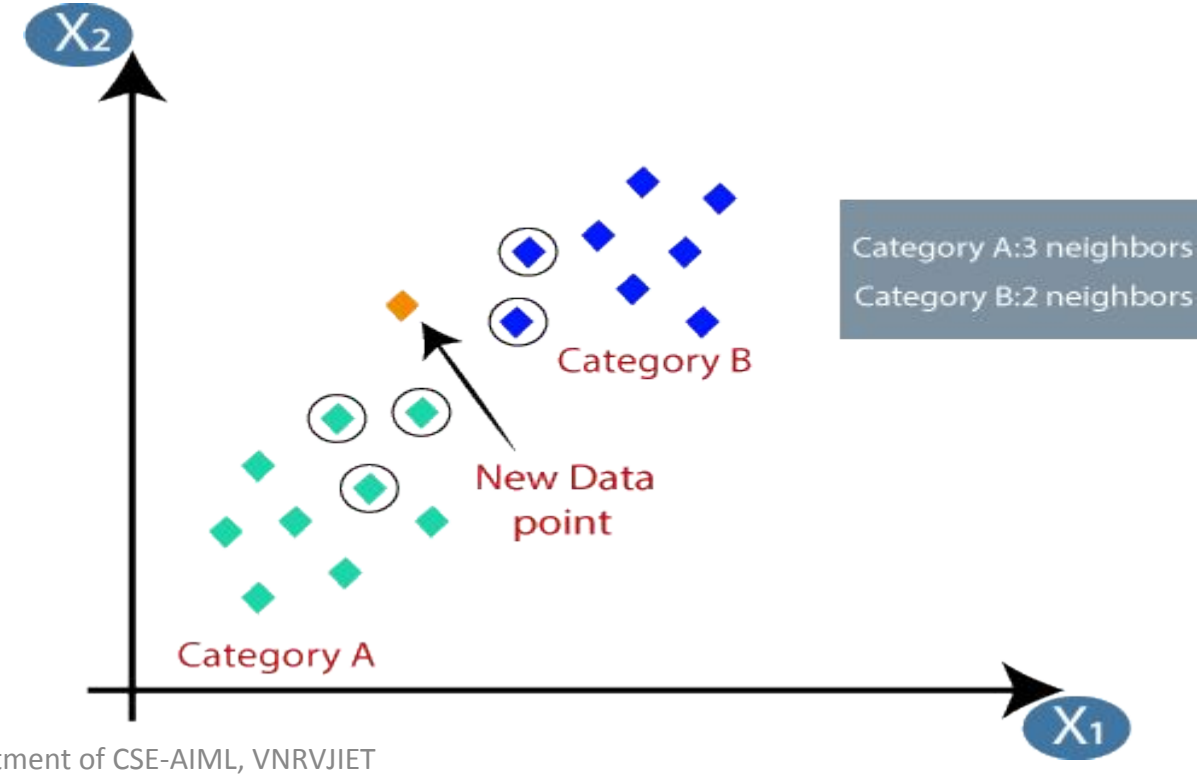


Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

K-Nearest Neighbour Classifier

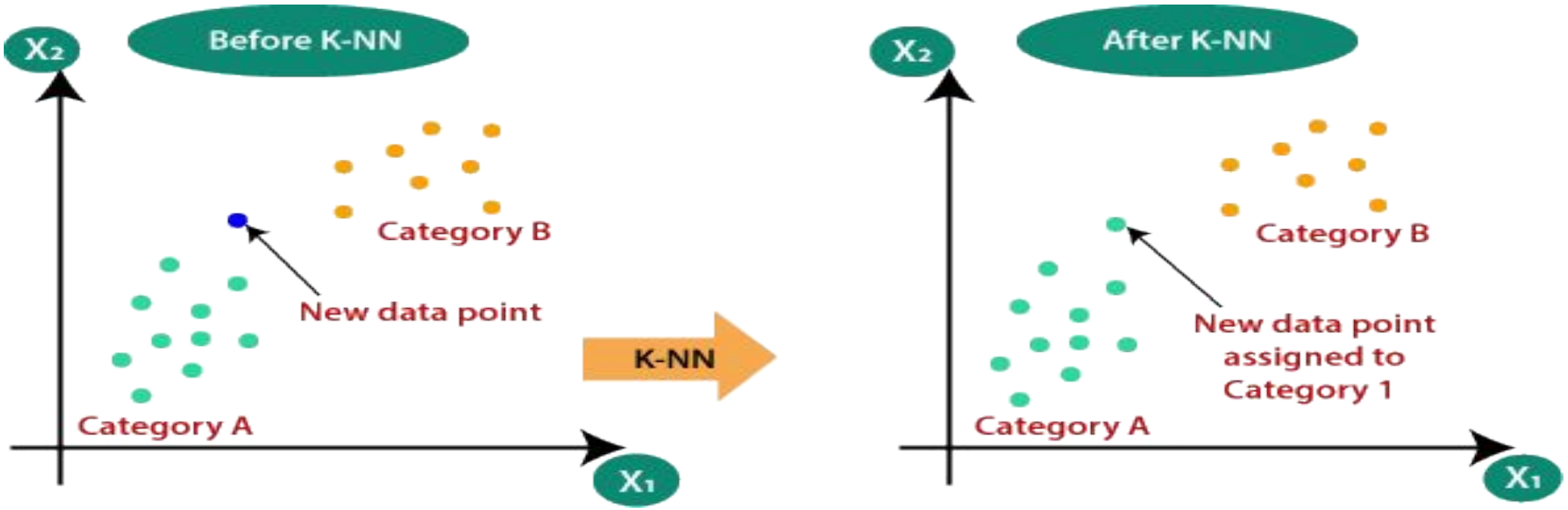
1. Finding the similarities as a measure of distance (Euclidean distance)
2. Identifying K nearest neighbors to the input data point based on their distances.





K-Nearest Neighbour Classifier

3. Assigns the most common class label among the K neighbors as the predicted label for the input data point.



K-Nearest Neighbour Classifier Algorithm

Input: Training data set, test data set (or data points), value of '*k*' (i.e. number of nearest neighbours to be considered)

Steps:

Do for all test data points

Calculate the distance (usually Euclidean distance) of the test data point from the different training data points.
Find the closest '*k*' training data points, i.e. training data points whose distances are least from the test data point.

If $k = 1$

Then assign class label of the training data point to the test data point

Else

Whichever class label is predominantly present in the training data points, assign that class label to the test data point

End do

K-Nearest Neighbour Classifier Algorithm

Why the kNN algorithm is called a lazy learner?

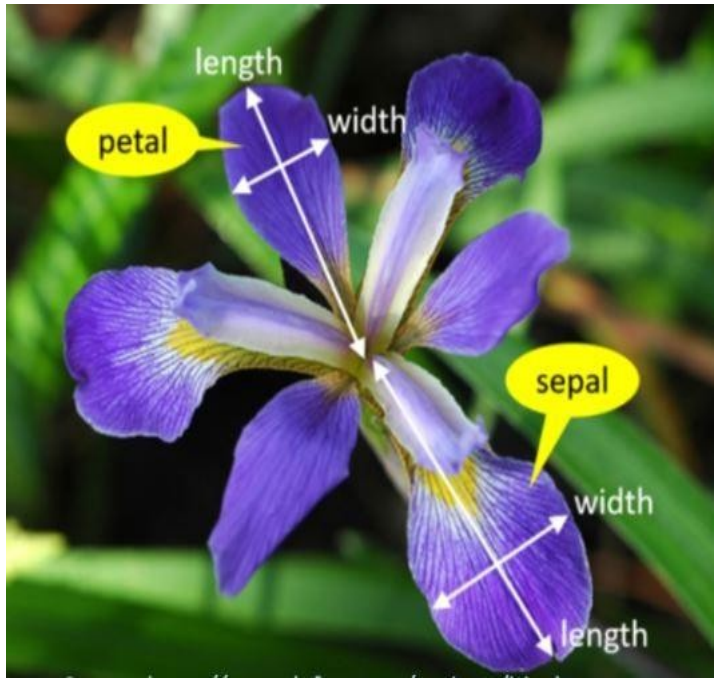
- It **stores** the training data and directly applies the philosophy of nearest neighborhood finding to arrive at the classification.
- So, for kNN, there is no learning happening in the real sense.
- Therefore, kNN falls under the category of lazy learner.

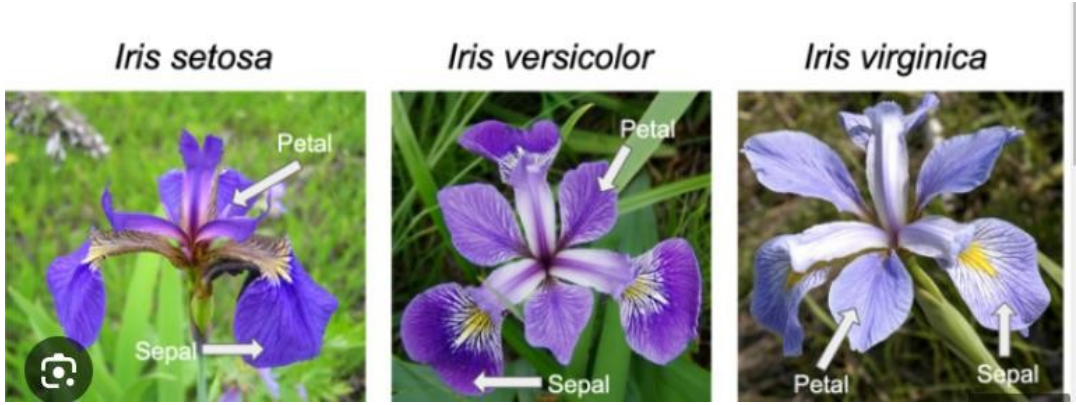
K-Nearest Neighbour Classifier Algorithm

Dataset: Iris

Training Data

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor





K-Nearest Neighbour Classifier

Training Data

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Test Data →

Sepal Length	Sepal Width	Species
5.2	3.1	?

Step 1: Find Distance

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(x-a)^2 + (y-b)^2}$$

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(5.2-5.3)^2 + (3.1-3.7)^2}$$

$$\text{Distance (Sepal Length, Sepal Width)} = 0.608$$

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

K-Nearest Neighbour Classifier

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Step 2: Find Rank

K-Nearest Neighbour Classifier

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Step 3: Find the Nearest Neighbor

If k = 1 – Setosa

If k = 2 – Setosa

If k = 5 – Setosa

Test Data →

Sepal Length	Sepal Width	Species
5.2	3.1	?

→ **Setosa**

K-Nearest Neighbour Classifier Algorithm

Few strategies, highlighted below, are adopted by machine learning practitioners to arrive at a value for k.

- One common practice is to set **k** equal to the **square root of the number of training records.**
- An alternative approach is to test several k values on a variety of test data sets and choose the one that delivers the best performance.
- Another interesting approach is to choose a larger value of k, but apply a weighted voting process in which the vote of close neighbours is considered more influential than the vote of distant neighbours.

Class room Assignment

1 Apply KNN Classifier to classify New Instance Height and Weight Example of following

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

Class room Assignment

2

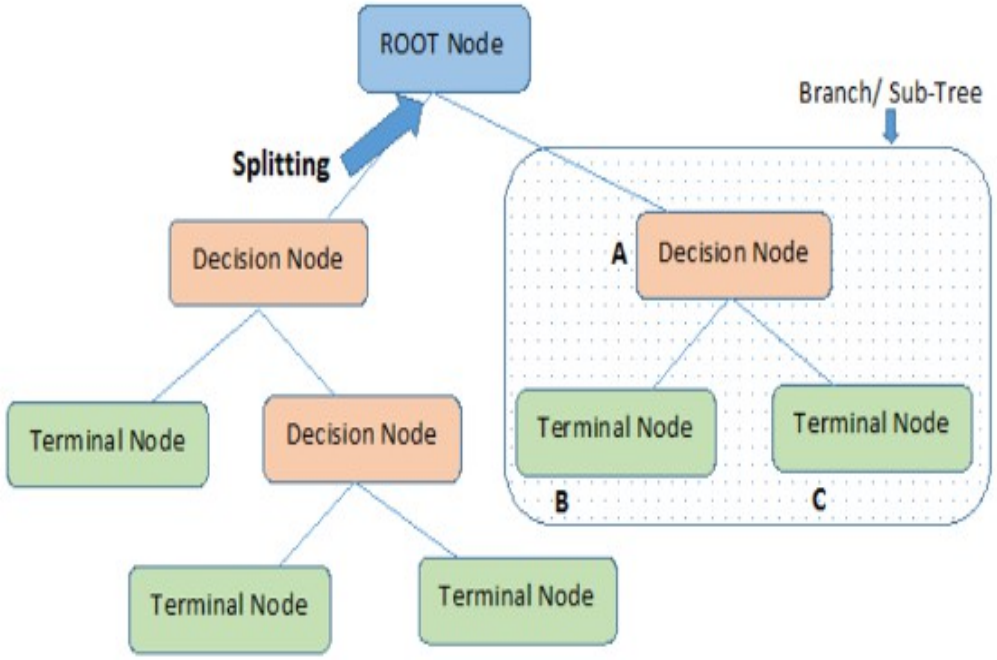
1. Students having good communication skills as well as a good level of aptitude have been classified as '**Leader**'
2. Students having good communication skills but not so good level of aptitude have been classified as '**Speaker**'
3. Students having not so good communication skill but a good level of aptitude have been classified as '**Intel**'

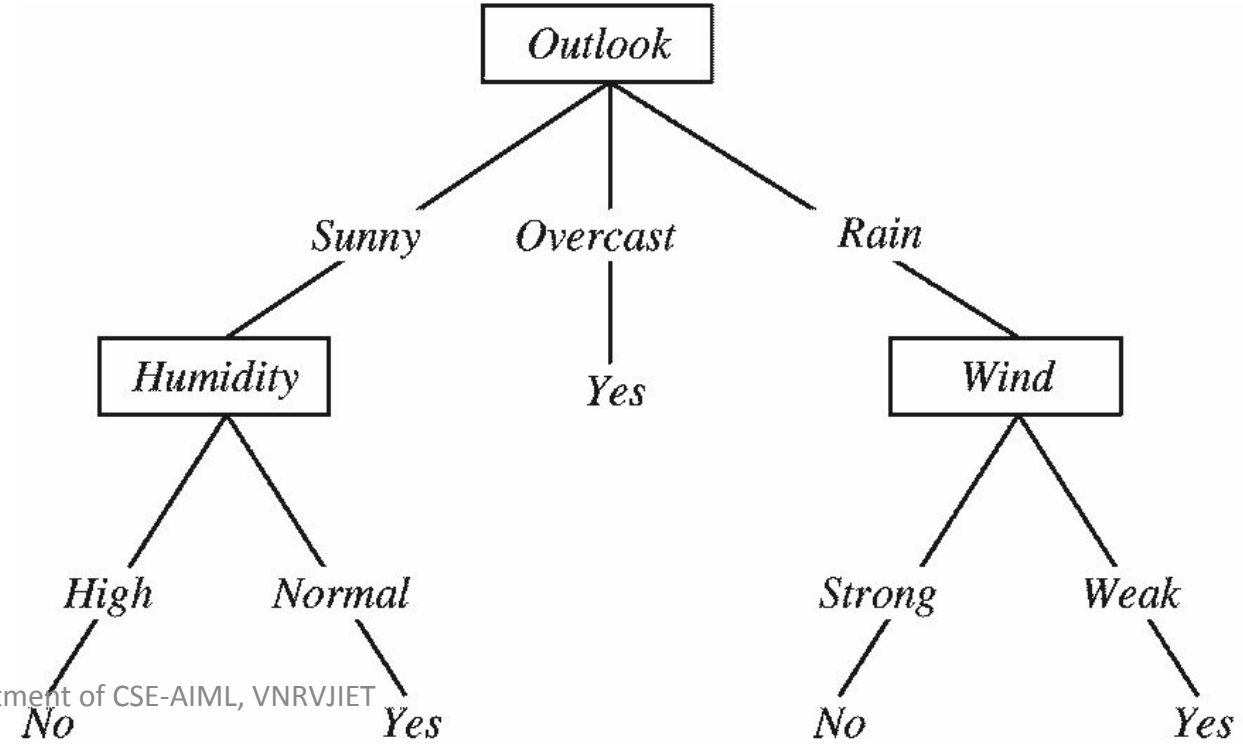
Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

FIG. 7.4 Student data set

Decision Tree Classification Algorithm

- A Decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data.
- Decision Tree is a Supervised learning technique that can be used for both **classification** and **Regression problems**, but mostly it is preferred for solving **Classification problems.**
- It is a tree-structured classifier, where **internal nodes** represent the **features of a dataset**, **branches** represent the **decision rules** and **each leaf node** represents **final result** or **the outcome.**
-





The Basic Decision Tree Learning Algorithm

- Most algorithms that have been developed for **learning decision trees** are variations on a core algorithm that employs a **top-down**, greedy search through the space of possible decision trees.
- This approach is exemplified by the **ID3 algorithm** (Quinlan 1986) and its **successor C4.5** (Quinlan 1993.
- **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 basic algorithm for **decision tree learning**, corresponding approximately to the **ID3 algorithm.**

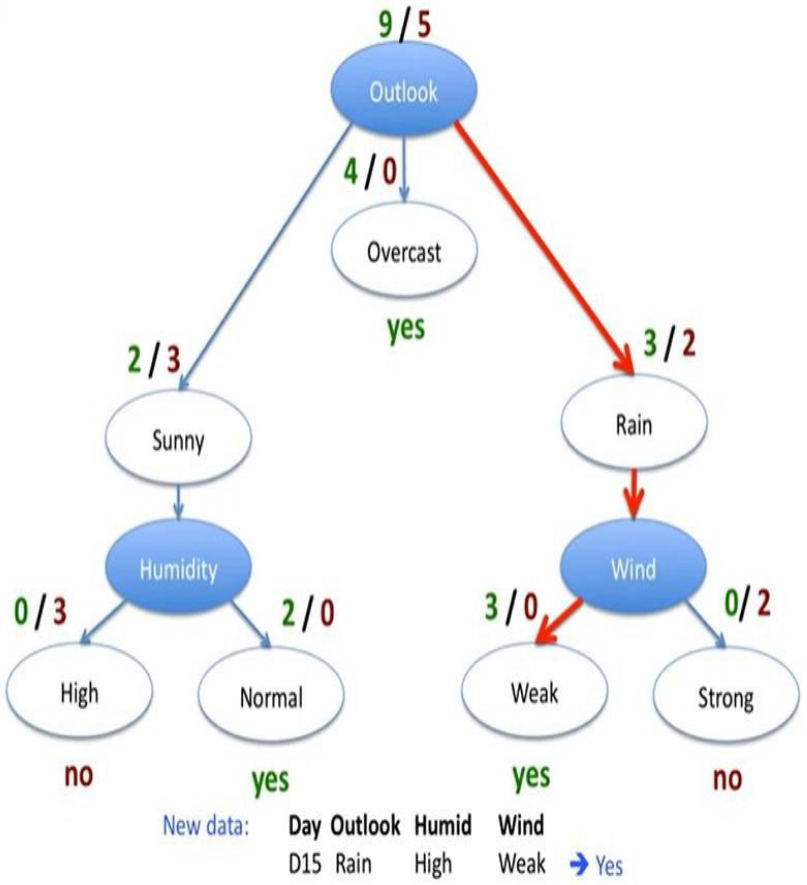
Decision Tree Classification Algorithm

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Tree classification example

Sample Dataset



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

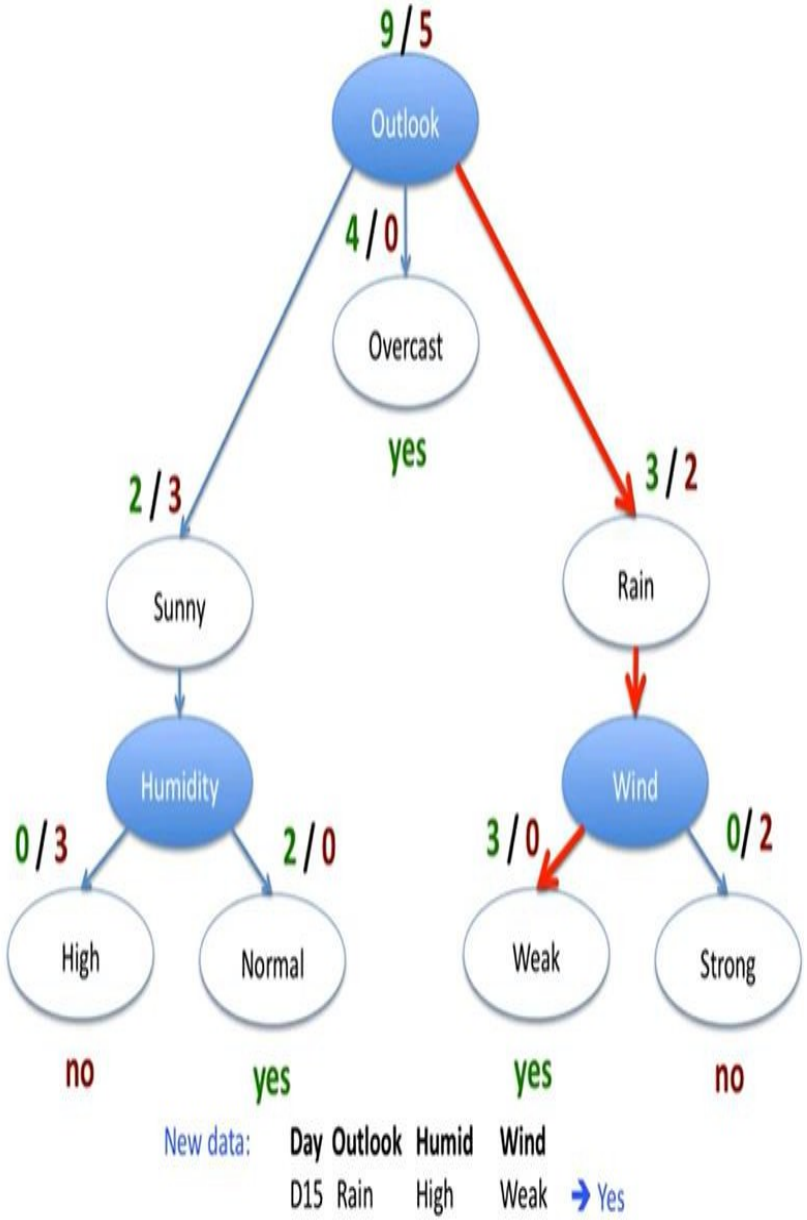
how to select the best attribute for the **Root node** and for **sub-nodes** ?

Decision Tree Classification Algorithm

Attribute Selection Measures:

- While implementing a Decision tree, the main issue arises that **how to select the best attribute** for the **root node** and for **sub-nodes**. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. **Information Gain**
2. **Gini Index**



Decision Tree Classification Algorithm

1. Information Gain

- Information gain will give **maximum information** out of **available attributes of** a dataset
- **Maximum information** of a **attribute** is considered as **ROOT NODE or Internal node (Splitting node)**
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute **having the highest information gain** is **split first.**
- Information gain is the measurement of **changes in entropy** after the segmentation or splitting of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- It can be calculated using the below formula:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} Entropy(S_v) \frac{|S_v|}{|S|}$$

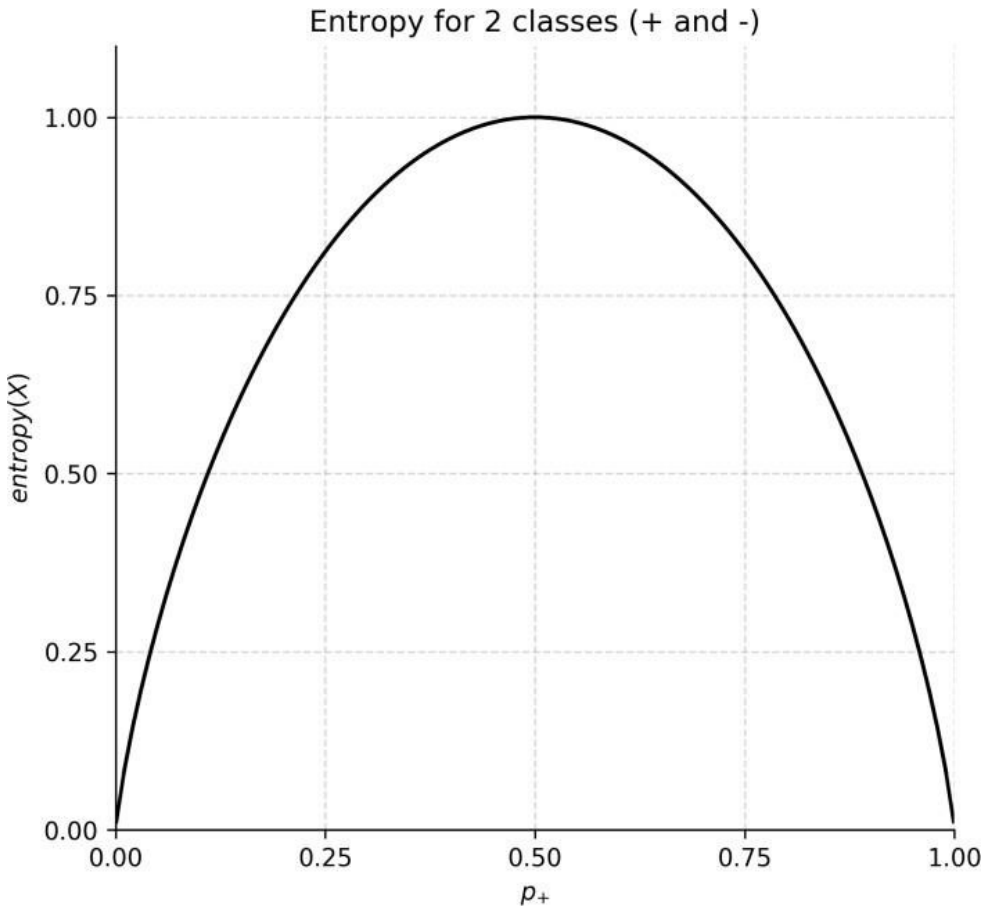
Entropy

- Entropy measures the degree of randomness in data

- For a set of samples X with k classes:

$$entropy_X(\ ) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

where p_i is the proportion of elements of class i



- Lower entropy implies greater predictability!

Decision Tree Classification Algorithm

Entropy

Entropy is a metric to measure the **impurity** in a **given attribute**. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= - P(+) log_2P(+) - P(-) log_2 P(-)

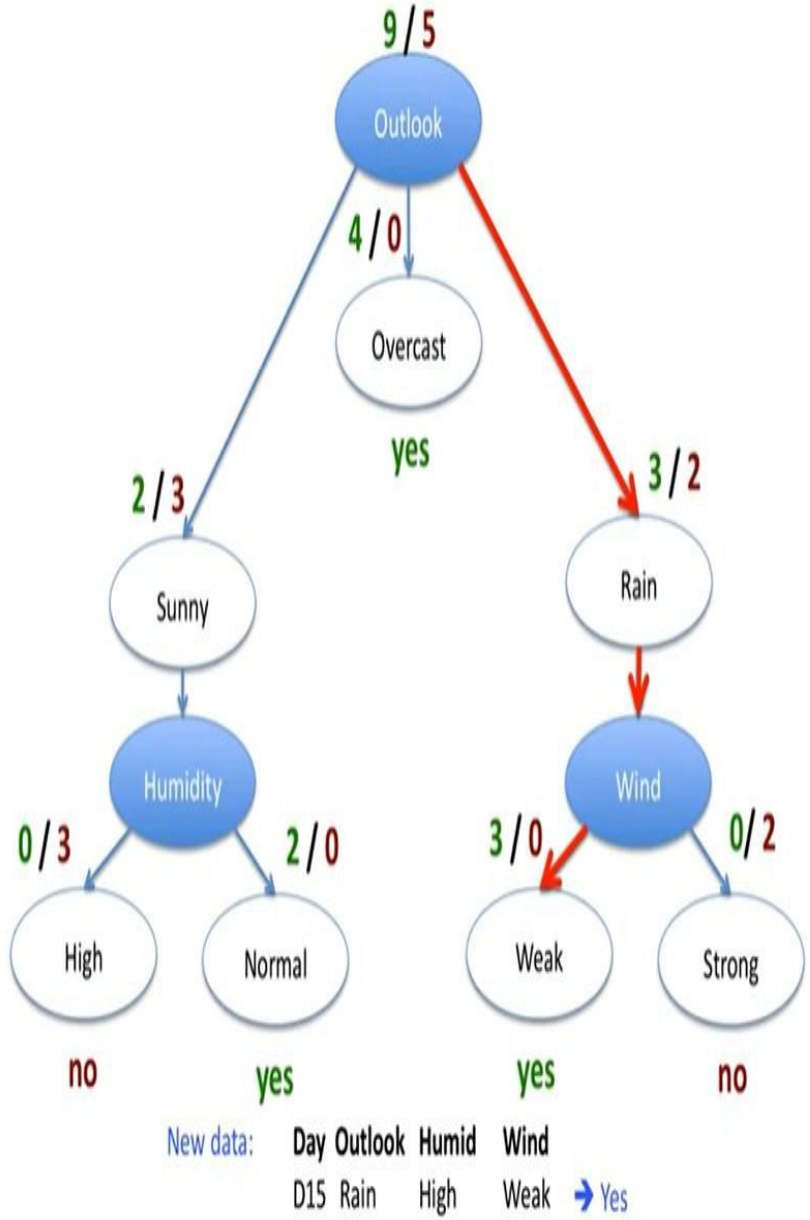
Entropy(s)= - P(yes) log_2P(yes) - P(no) log_2 P(no)

Here,

p_+ or P(yes) is the probability of positive class

p_- or P(no) is the probability of negative class

S is the subset of the training example



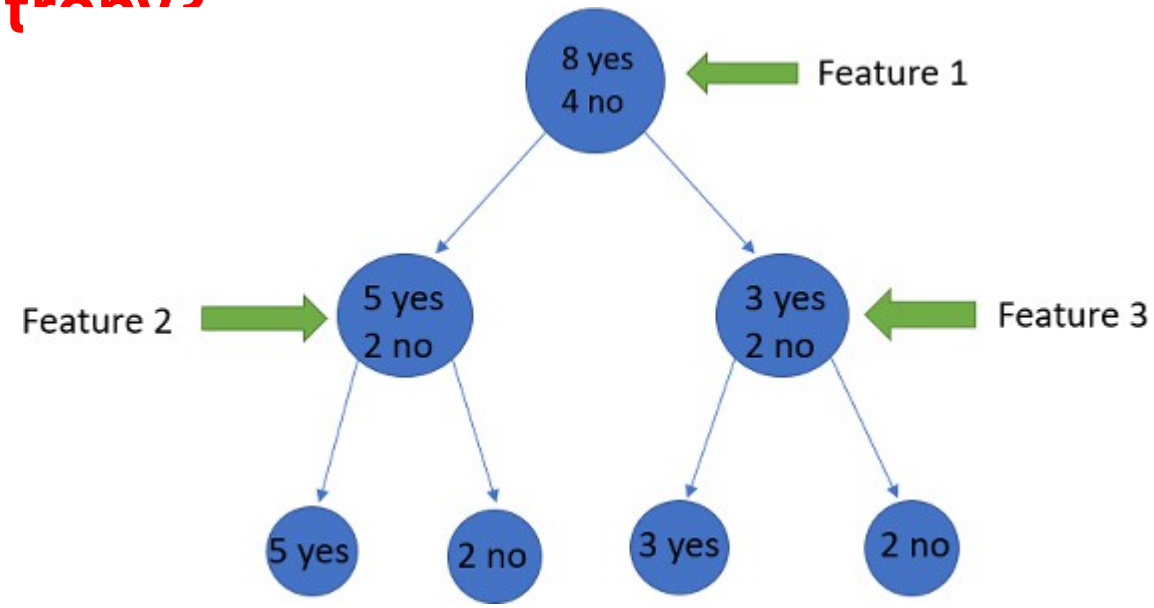
Examples

Entropy ([14+, 0–]) = – 14/14 log_2 (14/14) – 0 log_2 (0) = 0

Entropy ([9+, 5–]) = – 9/14 log_2 (9/14) – 5/14 log_2 (5/14) = 0.94

Entropy ([7+, 7–]) = – 7/14 log_2 (7/14) – 7/14 log_2 (7/14) = 1/2 + 1/2 = 1

How do Decision Trees use Entropy?



Entropy(s)= - P(yes) log_2P(yes) - P(no) log_2 P(no)

Entropy For feature 2,

$$\Rightarrow -\left(\frac{5}{7}\right)log_2\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)log_2\left(\frac{2}{7}\right)$$

$$\Rightarrow -(0.71 * -0.49) - (0.28 * -1.83)$$

$$\Rightarrow -(-0.34) - (-0.51)$$

$$\Rightarrow 0.34 + 0.51$$

$$\Rightarrow 0.85$$

Entropy For feature 3,

$$\Rightarrow -\left(\frac{3}{5}\right)log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)log_2\left(\frac{2}{5}\right)$$

$$\Rightarrow -(0.6 * -0.73) - (0.4 * -1.32)$$

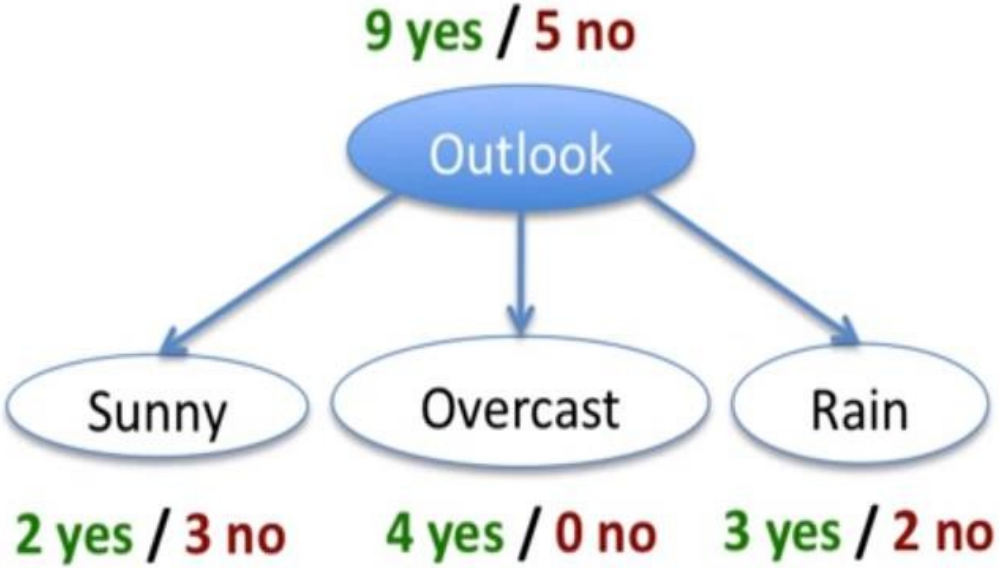
$$\Rightarrow -(-0.438) - (-0.528)$$

$$\Rightarrow 0.438 + 0.528$$

$$\Rightarrow 0.966$$

We can clearly see from the tree its **left node has low entropy** or **more purity** than right node

Which attribute to split on?



- Want to measure "purity" of the split
  - more certain about Yes/No after the split
    - pure set (4 yes / 0 no) => completely certain (100%)
    - impure (3 yes / 3 no) => completely uncertain (50%)

Decision Tree Algorithm

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

Decision Tree classification Using Entropy & Gain- Example-1

Decision Tree classification Using Entropy & Gain- Example-1

Calculation of Entropy of Outlook attribute values

and Calculation of Gain of Outlook attribute

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

Attribute: **Outlook**

$Values(Outlook) = Sunny, Overcast, Rain$

$S = [9+, 5-]$

$Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Sunny} \leftarrow [2+, 3-]$

$Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$

$S_{Overcast} \leftarrow [4+, 0-]$

$Entropy(S_{Overcast}) = -\frac{4}{4}log_2\frac{4}{4} - \frac{0}{4}log_2\frac{0}{4} = 0$

$S_{Rain} \leftarrow [3+, 2-]$

$Entropy(S_{Rain}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$

$$Gain\ (S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Gain(S, Outlook)$

$$= Entropy(S) - \frac{5}{14}Entropy(S_{Sunny}) - \frac{4}{14}Entropy(S_{Overcast}) - \frac{5}{14}Entropy(S_{Rain})$$

$$Gain(S, Outlook) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.2464$$

- Calculation of Entropy of Temperature attribute values and
- Calculation Gain of Temperature attribute

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

Attribute Temp

$Values\ (Temp) = Hot, Mild, Cool$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.94$

$S_{Hot} \leftarrow [2+, 2-]$ $\quad Entropy(S_{Hot}) = -\frac{2}{4} log_2 \frac{2}{4} - \frac{2}{4} log_2 \frac{2}{4} = 1.0$

$S_{Mild} \leftarrow [4+, 2-]$ $\quad Entropy(S_{Mild}) = -\frac{4}{6} log_2 \frac{4}{6} - \frac{2}{6} log_2 \frac{2}{6} = 0.9183$

$S_{Cool} \leftarrow [3+, 1-]$ $\quad Entropy(S_{Cool}) = -\frac{3}{4} log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4} = 0.8113$

$$Gain\ (S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Gain(S, Temp)$

$$= Entropy(S) - \frac{4}{14} Entropy(S_{Hot}) - \frac{6}{14} Entropy(S_{Mild})$$

$$- \frac{4}{14} Entropy(S_{Cool})$$

$$Gain(S, Temp) = 0.94 - \frac{4}{14} 1.0 - \frac{6}{14} 0.9183 - \frac{4}{14} .8113 = 0.028$$

Decision Tree classification Using Entropy & Gain- Example-1

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

Attribute: Humidity

$Values\ (Humidity) = High, Normal$

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.94$

$S_{High} \leftarrow [3+, 4-]$ $\quad Entropy(S_{High}) = -\frac{3}{7} log_2 \frac{3}{7} - \frac{4}{7} log_2 \frac{4}{7} = 0.9852$

$S_{Normal} \leftarrow [6+, 1-]$ $\quad Entropy(S_{Normal}) = -\frac{6}{7} log_2 \frac{6}{7} - \frac{1}{7} log_2 \frac{1}{7} = 0.5916$

$$Gain\ (S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Humidity)$$

$$= Entropy(S) - \frac{7}{14} Entropy(S_{High}) - \frac{7}{14} Entropy(S_{Normal})$$

$$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14} 0.5916 = 0.1516$$

Decision Tree classification Using Entropy & Gain- Example-1

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

Attribute: Wind

$Values\ (Wind) = Strong, Weak$

$S = [9+, 5-]$ $\qquad Entropy(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.94$

$S_{Strong} \leftarrow [3+, 3-]$ $\qquad Entropy(S_{Strong}) = 1.0$

$S_{Weak} \leftarrow [6+, 2-]$ $\qquad Entropy(S_{Weak}) = -\frac{6}{8} log_2 \frac{6}{8} - \frac{2}{8} log_2 \frac{2}{8} = 0.8113$

$$Gain\ (S, Wind) = Entropy(S) - \sum_{v \in (Strong, Weak)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Wind) = Entropy(S) - \frac{6}{14} Entropy(S_{Strong}) - \frac{8}{14} Entropy(S_{Weak})$$

$$Gain(S, Wind) = 0.94 - \frac{6}{14}\ 1.0 - \frac{8}{14}\ 0.8113 = 0.0478$$

Comparing Gain of all attributes for Root node selection or Node splitting

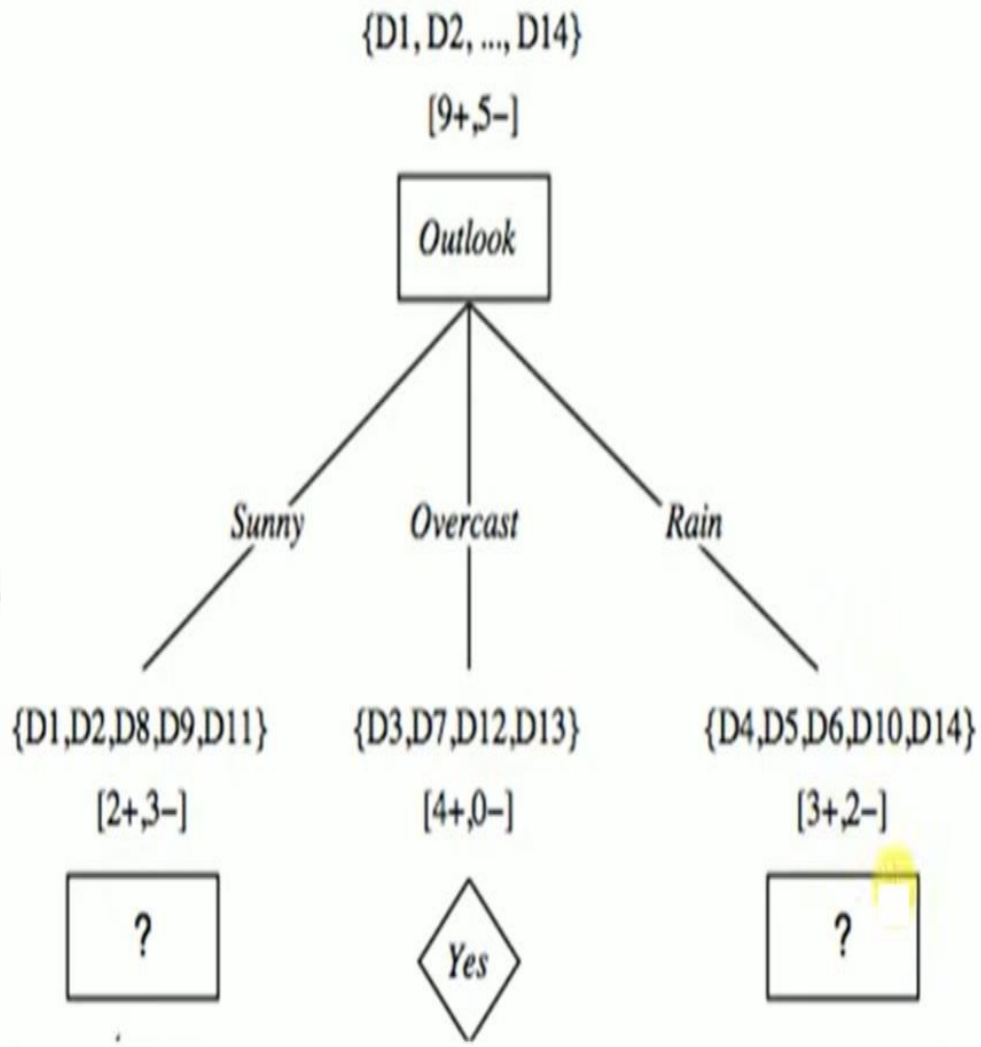
Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak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

$$Gain(S, Outlook) = 0.2464$$

$$Gain(S, Temp) = 0.0289$$

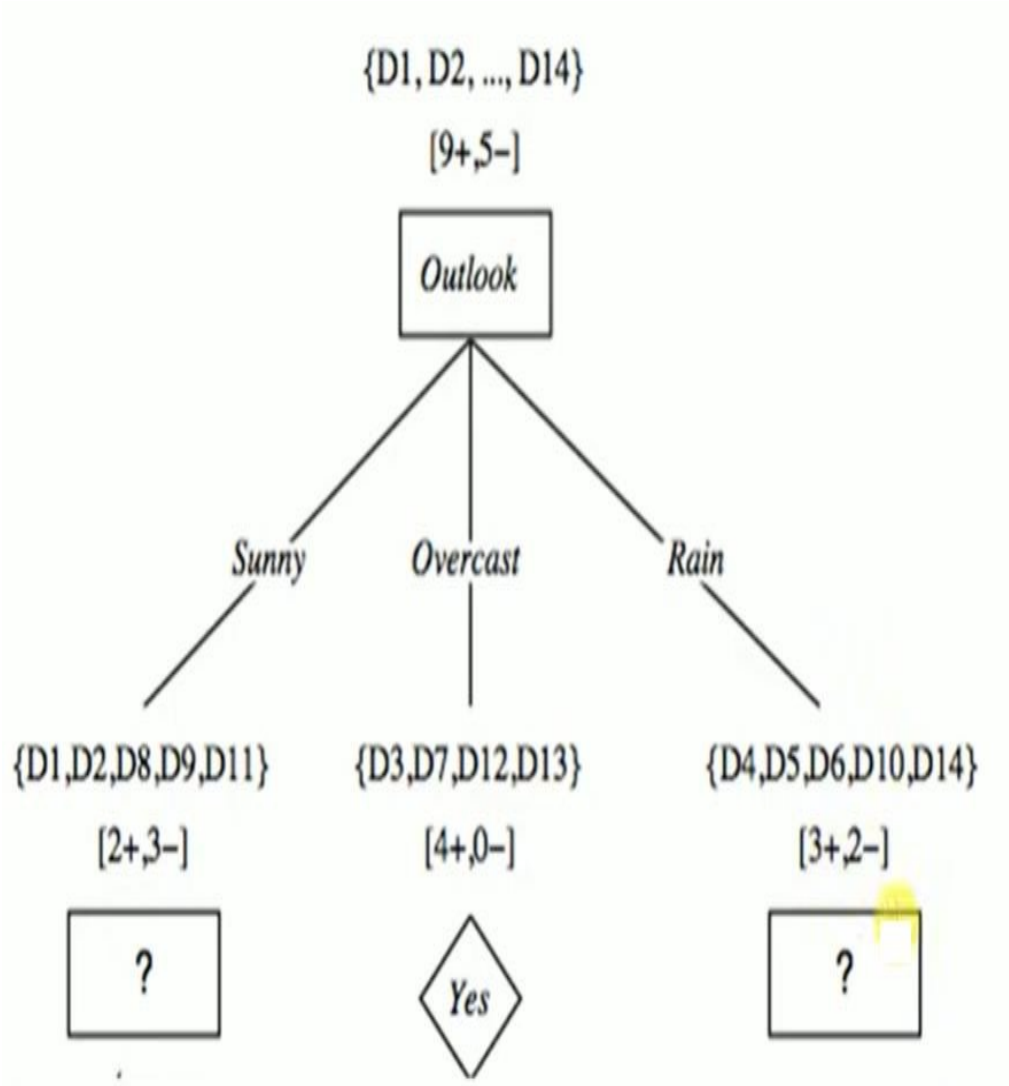
$$Gain(S, Humidity) = 0.1516$$

$$Gain(S, Wind) = 0.0478$$



Calculation of Gain of Left Internal node for Splitting

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes



Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Attribute: Temp

$$Values\,(Temp) = Hot, Mild, Cool$$

$$S_{Sunny} = [2+, 3-] \qquad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$$

$$S_{Hot} \leftarrow [0+, 2-] \qquad Entropy(S_{Hot}) = 0.0$$

$$S_{Mild} \leftarrow [1+, 1-] \qquad Entropy(S_{Mild}) = 1.0$$

$$S_{Cool} \leftarrow [1+, 0-] \qquad Entropy(S_{Cool}) = 0.0$$

$$Gain\,(S_{Sunny}, Temp) = Entropy(S) - \sum_{v \in (Hot, Mild, Cool)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Temp)$$

$$= Entropy(S) - \frac{2}{5}Entropy(S_{Hot}) - \frac{2}{5}Entropy(S_{Mild})$$

$$- \frac{1}{5}Entropy(S_{Cool})$$

$$Gain(S_{sunny}, Temp) = 0.97 - \frac{2}{5}0.0 - \frac{2}{5}\,1 - \frac{1}{5}0.0 = 0.570$$

Day	Temp	Humidity	Wind	Play Tennis
Dl	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
Dl1	Mild	Normal	Strong	Yes

Attribute: Humidity

$$Values\,(Humidity) = High, Normal$$

$$S_{Sunny} = [2+, 3-] \qquad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$$

$$S_{high} \leftarrow [0+, 3-] \qquad Entropy(S_{High}) = 0.0$$

$$S_{Normal} \leftarrow [2+, 0-] \qquad Entropy(S_{Normal}) = 0.0$$

$$Gain\,(S_{Sunny}, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Humidity) = Entropy(S) - \frac{3}{5}Entropy(S_{High}) - \frac{2}{5}Entropy(S_{Normal})$$

$$Gain(S_{sunny}, Humidity) = 0.97 - \frac{3}{5}0.0 - \frac{2}{5}0.0 = 0.97$$

Day	Temp	Humidity	Wind	Play Tennis
Dl	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
Dl1	Mild	Normal	Strong	Yes

Attribute: Wind

$Values\,(Wind) = Strong, Weak$

$S_{Sunny} = [2+, 3-]$

$S_{Strong} \leftarrow [1+, 1-]$

$S_{Weak} \leftarrow [1+, 2-]$

$$Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$$

$$Entropy(S_{Strong}) = 1.0$$

$$Entropy(S_{Weak}) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$$

$$Gain\,(S_{Sunny}, Wind) = Entropy(S) - \sum_{v \in (Strong, Weak)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

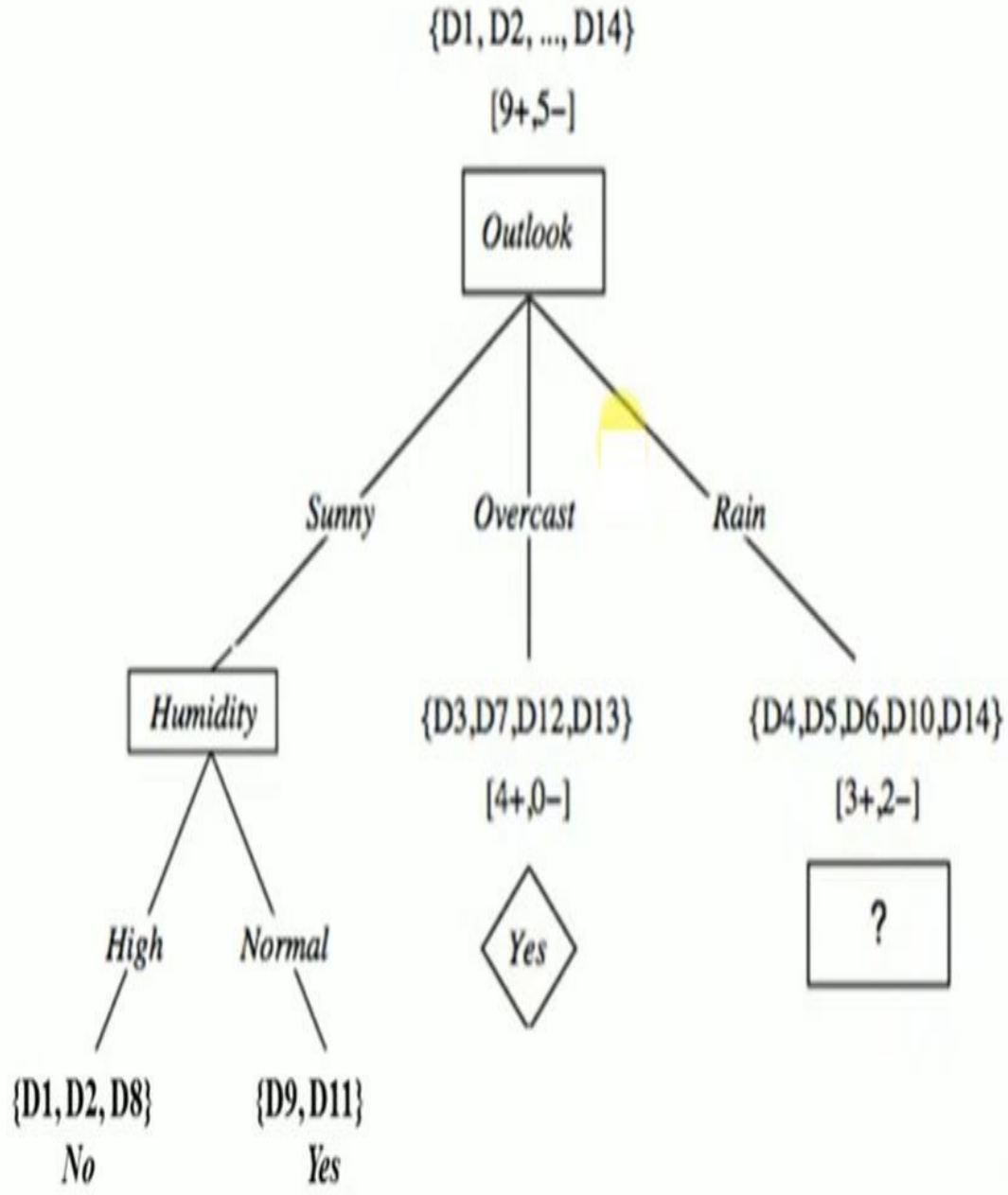
$$Gain\,(S_{sunny}, Wind) = 0.97 - \frac{2}{5}1.0 - \frac{3}{5}\,0.918 = 0.0192$$

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$$Gain(S_{sunny}, Temp) = 0.570$$

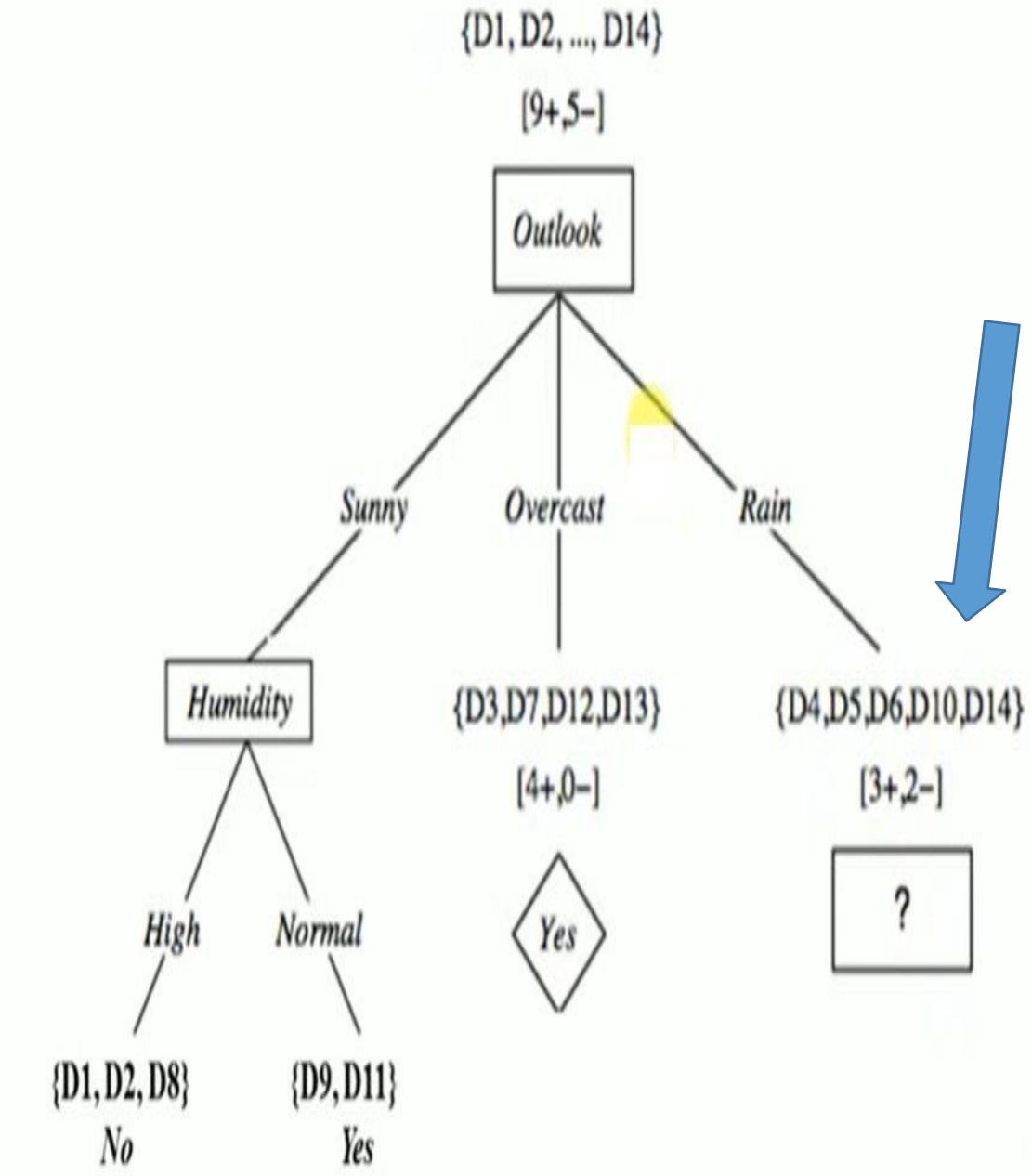
$$Gain(S_{sunny}, Humidity) = 0.97$$

$$Gain(S_{sunny}, Wind) = 0.0192$$



Calculation of Gain of Right Internal node for Splitting

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No



Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Attribute: Temp

$$Values\,(Temp) = Hot, Mild, Cool$$

$S_{Rain} = [3+, 2-]$ $\quad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 0-]$ $\quad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [2+, 1-]$ $\quad Entropy(S_{Mild}) = -\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.9183$

$S_{Cool} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Cool}) = 1.0$

$$Gain\,(S_{Rain}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Temp)$$

$$= Entropy(S) - \frac{0}{5}Entropy(S_{Hot}) - \frac{3}{5}Entropy(S_{Mild})$$

$$- \frac{2}{5}Entropy(S_{Cool})$$

$$Gain(S_{Rain}, Temp) = 0.97 - \frac{0}{5}0.0 - \frac{3}{5}0.918 - \frac{2}{5}1.0 = 0.0192$$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

Attribute: Humidity

$$Values\,(Humidity) = High, Normal$$

$$S_{Rain} = [3+, 2-] \qquad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$$

$$S_{High} \leftarrow [1+, 1-] \qquad Entropy(S_{High}) = 1.0$$

$$S_{Normal} \leftarrow [2+, 1-] \qquad Entropy(S_{Normal}) = -\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.9183$$

$$Gain\,(S_{Rain}, Humidity) = Entropy(S) - \sum_{v \in (High, Normal)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Humidity) = Entropy(S) - \frac{2}{5}Entropy(S_{High}) - \frac{3}{5}Entropy(S_{Normal})$$

$$Gain(S_{Rain}, Humidity) = 0.97 - \frac{2}{5}\,1.0 - \frac{3}{5}0.918 = 0.0192$$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

Attribute: Wind

$Values\ (wind) = Strong, Weak$

$S_{Rain} = [3+, 2-]$

$S_{Strong} \leftarrow [0+, 2-]$

$S_{Weak} \leftarrow [3+, 0-]$

$$Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$$

$$Entropy(S_{Strong}) = 0.0$$

$$Entropy(S_{weak}) = 0.0$$

$$Gain\ (S_{Rain}, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

$$Gain(S_{Rain}, Wind) = 0.97 - \frac{2}{5}\ 0.0 - \frac{3}{5}0.0 = 0.97$$

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

$$Gain(S_{Rain}, Temp) = 0.0192$$

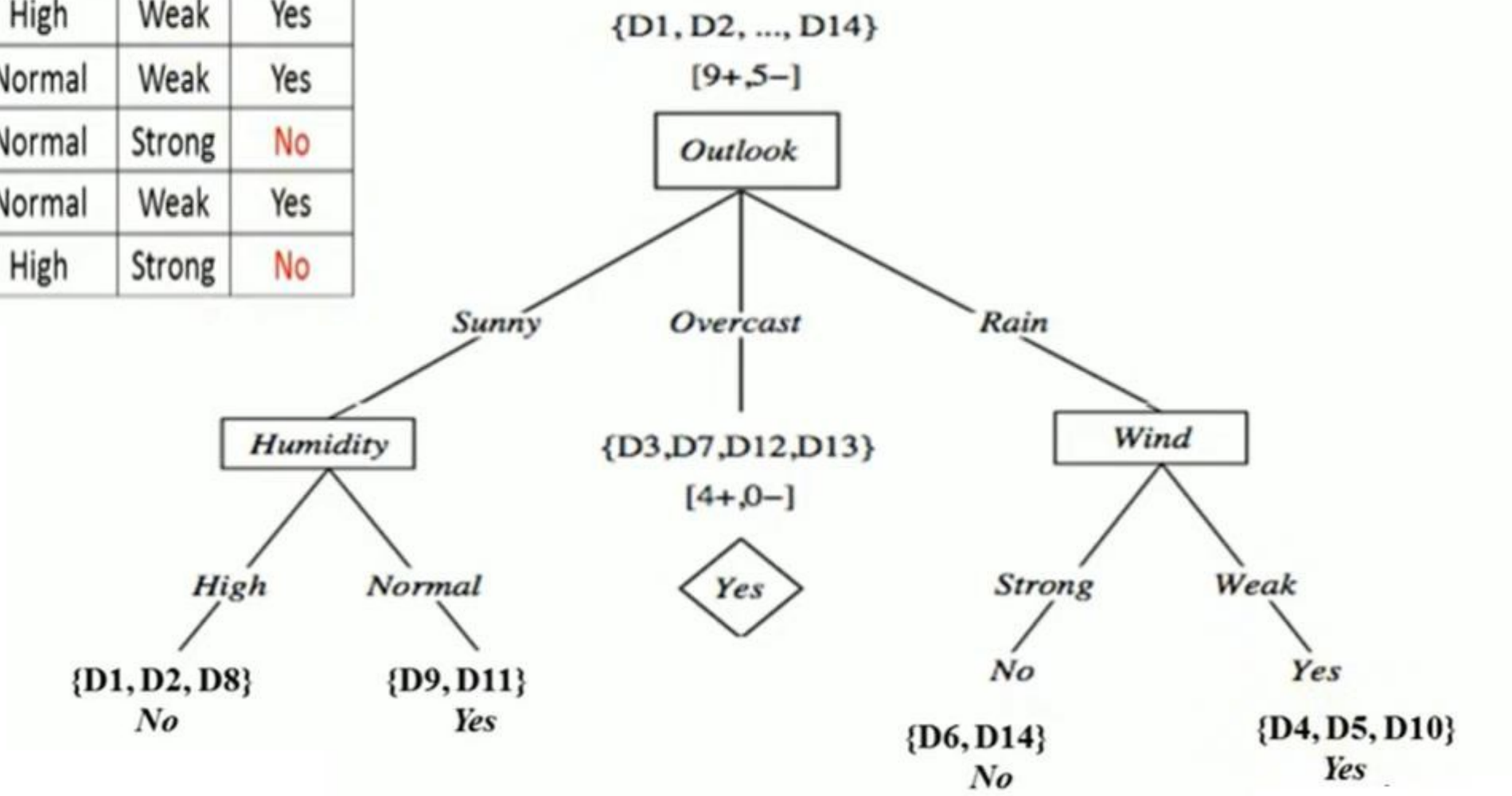
$$Gain(S_{Rain}, Humidity) = 0.0192$$

$$Gain(S_{Rain}, Wind) = 0.97$$

Decision Tree classification Using Entropy & Gain- Example-1

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
DlO	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

FINAL DECISION TREE



Decision Tree classification Using Entropy &Gain-Example-2

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

The entropy of the training examples is

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$Entropy(S) = -\frac{4}{9}\log_2\left(\frac{4}{9}\right) - \frac{5}{9}\log_2\left(\frac{5}{9}\right)$$

$$= 0.9911$$

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

What is the information gain of the a1 with respect to the training examples.

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$Entropy(S_T) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)$$

$$= 0.311 + 0.5 = 0.811$$

$$Entropy(S_F) = -\frac{1}{5}\log_2\left(\frac{1}{5}\right) - \frac{4}{5}\log_2\left(\frac{4}{5}\right)$$

$$= 0.4644 + 0.2576 = 0.722$$

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

What is the information gain of the a1 with respect to the training examples.

$$Gain\,(a_1) = Entropy(S) - \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain\,(a_1) = Entropy(S) - \frac{4}{9} Entropy(S_T)$$

$$- \frac{5}{9} Entropy(S_F)$$

$$Gain\,(a_1) = 0.9911 - \frac{4}{9} * 0.811 - \frac{5}{9} * 0.722 = 0.2295$$

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

What is the information gain of the a2 with respect to the training examples.

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$Entropy(S_T) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$

$$= 0.5288 + 0.4421 = 0.9709$$

$$Entropy(S_F) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)$$

$$= 0.5 + 0.5 = 1.0$$

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

What is the information gain of the a2 with respect to the training examples.

$$Gain\,(a_2) = Entropy(S) - \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain\,(a_2) = Entropy(S) - \frac{5}{9} Entropy(S_T)$$

$$- \frac{4}{9} Entropy(S_F)$$

$$Gain\,(a_2) = 0.9911 - \frac{5}{9} * 0.9709 - \frac{4}{9} * 1.0 = 0.0072$$

Decision Tree classification Using Entropy & Gain- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Which is the best splitting attribute between a1 and a2.

$$Gain\,(a_1) = 0.2295$$

$$Gain\,(a_2) = 0.0072$$

Higher Information Gain Produces Better Split

Hence, attribute a1 is the best split attribute

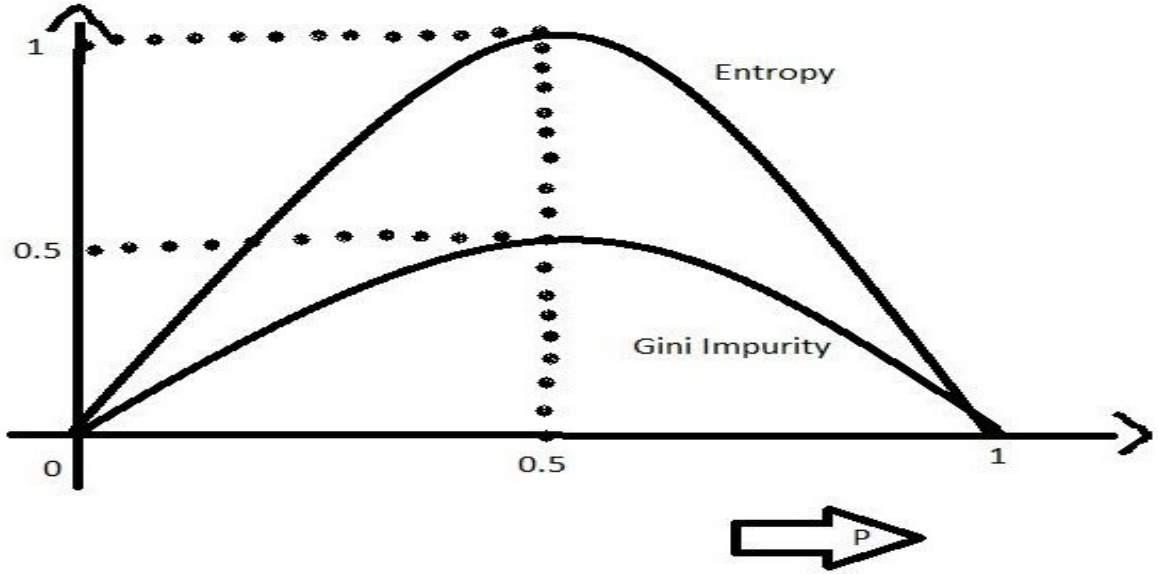
2. Gini Index

- The other way of splitting a decision tree is via the Gini Index. The Entropy and Information Gain method focuses on purity and impurity in a node.
- It is the probability of misclassifying a randomly chosen element in a set.
- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- The Gini Index is a measure of theinequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithms.
- Gini Index or Gini impurity measures the probability for a random instance being misclassified when chosen randomly.The lower theGini Index, the better the lower the likelihood of misclassification.
- An attribute with a lower Gini index should be preferred.
- Alower Gini Index indicates a more homogeneous or pure distribution,
  while a higher Gini Index indicates a more heterogeneous or impure distribution.
- In decision trees, theGini Index is used to evaluate thequality of a

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

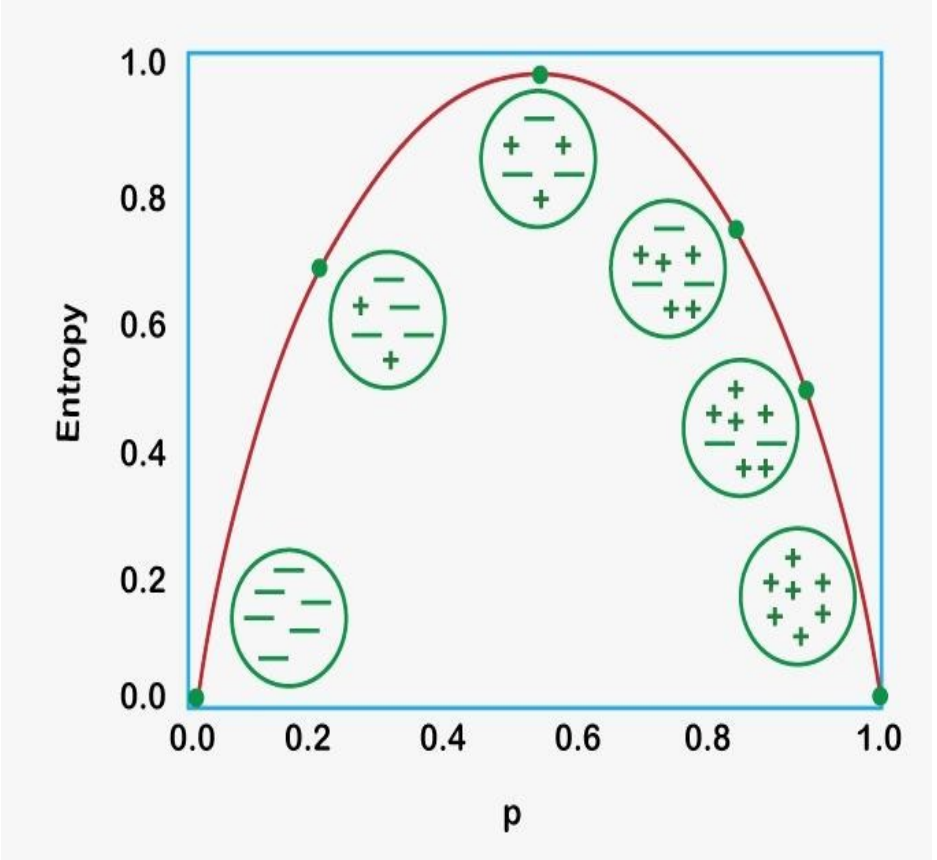
- Where **n** represents the no. of classes in the target variable
- **P(i)** represents the ratio of **Class occurrence/Total no. of observations** in node.
- It is calculated by summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.
- Compared to other impurity measures like entropy, the Gini Index is faster to compute and more sensitive to changes in class probabilities.
- In practice, the choice between using the Gini Index or other impurity measures depends on the specific problem and dataset, and often requires experimentation and tuning.

- Entropy has a maximum impurity of 1 and maximum purity is 0.
- Gini index has a maximum impurity is 0.5 and maximum purity is 0.
- Different Decision Tree algorithms. For example, **CART (Classification and Regression Trees)** uses Gini; **ID3 (Iterative Dichotomiser 3)** and **C4.5** use Entropy.



$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

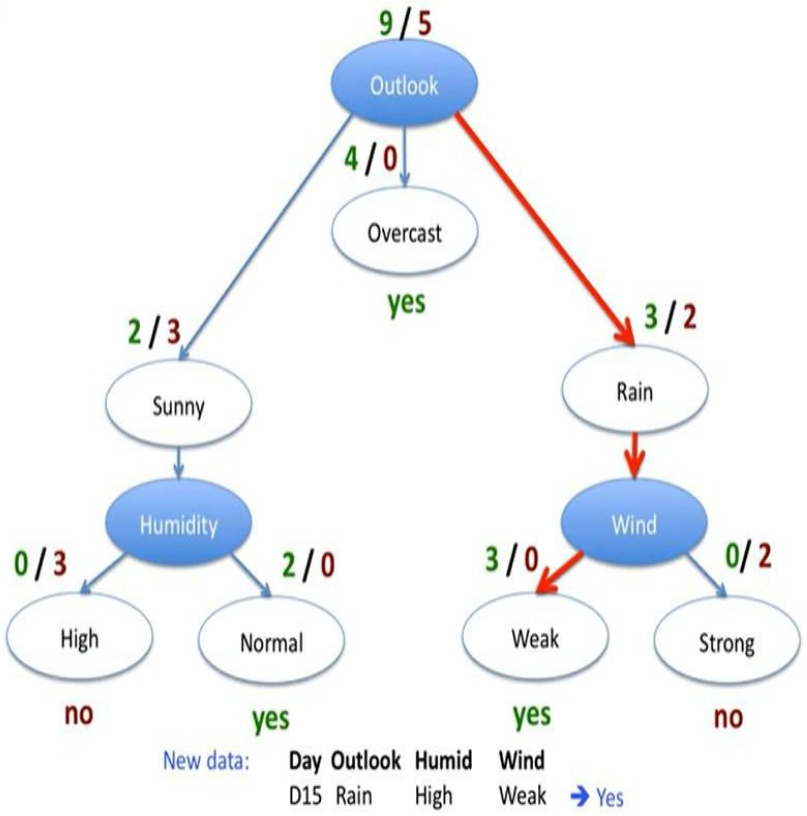
$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$



Decision Tree classification Using GINI INDEX- Example-1

Decision Tree classification Using GINI INDEX- Example-1

Sample Dataset



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

how to select the best attribute for the **Root node** and for **sub-nodes** ?

Decision Tree classification Using GINI INDEX- Example-1

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

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Gini (S) = 1 - ((9/14)^2 + (5/14)^2) = 0.45

Gini (Sunny) = 1 - ((2/5)^2 + (3/5)^2) = 0.48

Gini (Overcast) = 1 - ((4/4)^2 + (0/4)^2) = 0

Gini (Rain) = 1 - ((3/5)^2 + (2/5)^2) = 0.48

Weighted Average (Outlook) =
w1* Gini (Sunny)+w2*Gini (Overcast) +w3*Gini (Rain)

Weighted Average (Outlook) =
5/14*0.48 + 4/14*0 + 5/14*0.48 = 0.029 (Gini

Decision Tree classification Using GINI INDEX- Example-1

Note:

- Continue the procedure of Gini Calculation for the other values of Outlook Attribute and calculate GINI INDEX for Outlook Attribute
- Similarly Calculate GINI INDEX for other Attributes (Temp,Humidity and wind) after calculation of Gini for their individual values.
- Consider the Attribute as ROOT NODE/Splitting node which is having the least GINI INDEX.
- Continue the procedure until leaf nodes classifying the result

Decision Tree classification Using GINI INDEX- Example-2

Decision Tree classification Using GINI INDEX- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Compute the Gini Index of the attributes a1.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

$$Gini(T) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$Gini(F) = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.32$$

Decision Tree classification Using GINI INDEX- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Compute the Gini Index of the attributes a1.

$$GiniIndex(a_1) = \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Gini(S_v)$$

$$GiniIndex(a_1) = \left(\frac{4}{9}\right) * Gini(T) + \left(\frac{5}{9}\right) * Gini(F)$$

$$GiniIndex(a_1) = \left(\frac{4}{9}\right) * 0.375 + \left(\frac{5}{9}\right) * 0.32$$

$$GiniIndex(a_1) = 0.3444$$

Decision Tree classification Using GINI INDEX- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Compute the Gini Index of the attributes a2.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

$$Gini(T) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$Gini(F) = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

Decision Tree classification Using GINI INDEX- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Compute the Gini Index of the attributes a2.

$$GiniIndex(a_2) = \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Gini(S_v)$$

$$GiniIndex(a_2) = \left(\frac{5}{9}\right) * Gini(T) + \left(\frac{4}{9}\right) * Gini(F)$$

$$GiniIndex(a_2) = \left(\frac{5}{9}\right) * 0.48 + \left(\frac{4}{9}\right) * 0.5$$

$$GiniIndex(a_2) = 0.4889$$

Decision Tree classification Using GINI INDEX- Example-2

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Which is the best splitting attribute between a1 and a2.

$$GiniIndex(a_1) = 0.3444$$

$$GiniIndex(a_2) = 0.4889$$

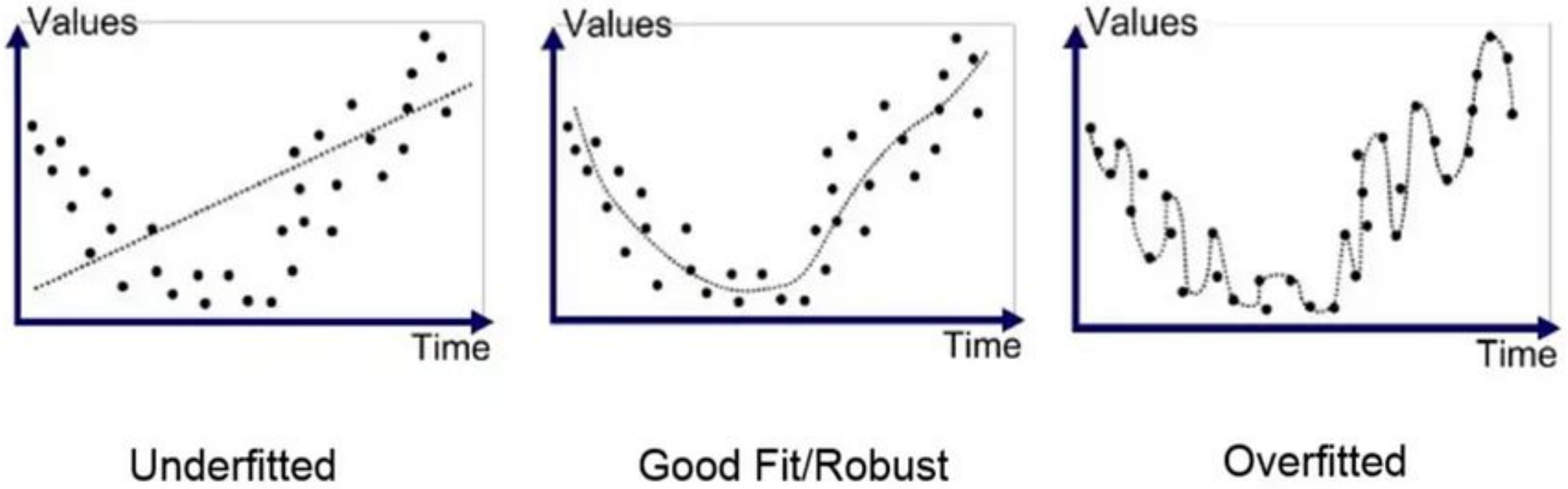
Smaller GiniIndex Produces Better Split

Hence, attribute a1 is the best split attribute

Decision Tree-Over-fitting

Issue

- **Overfitting**: When a model learns **too much from training data**, memorizing noise and performing **poorly on new data**. This can lead to high variance and low bias.
- **Underfitting**: When a model is **too basic**, **missing patterns in training and new data**. This can lead to low variance and high bias.



Decision Tree-Over-fitting Issue

- **Over-fitting:**
  - Too many branches, somemay reflect anomalies due to noise or outliers
  - Too much complexity
  - Poor accuracy for unseen samples or Testing data

- **Two approaches to avoid overfitting**
  - **Pruning**
    - **Pre-pruning**
    - **Post-pruning**

Pruning

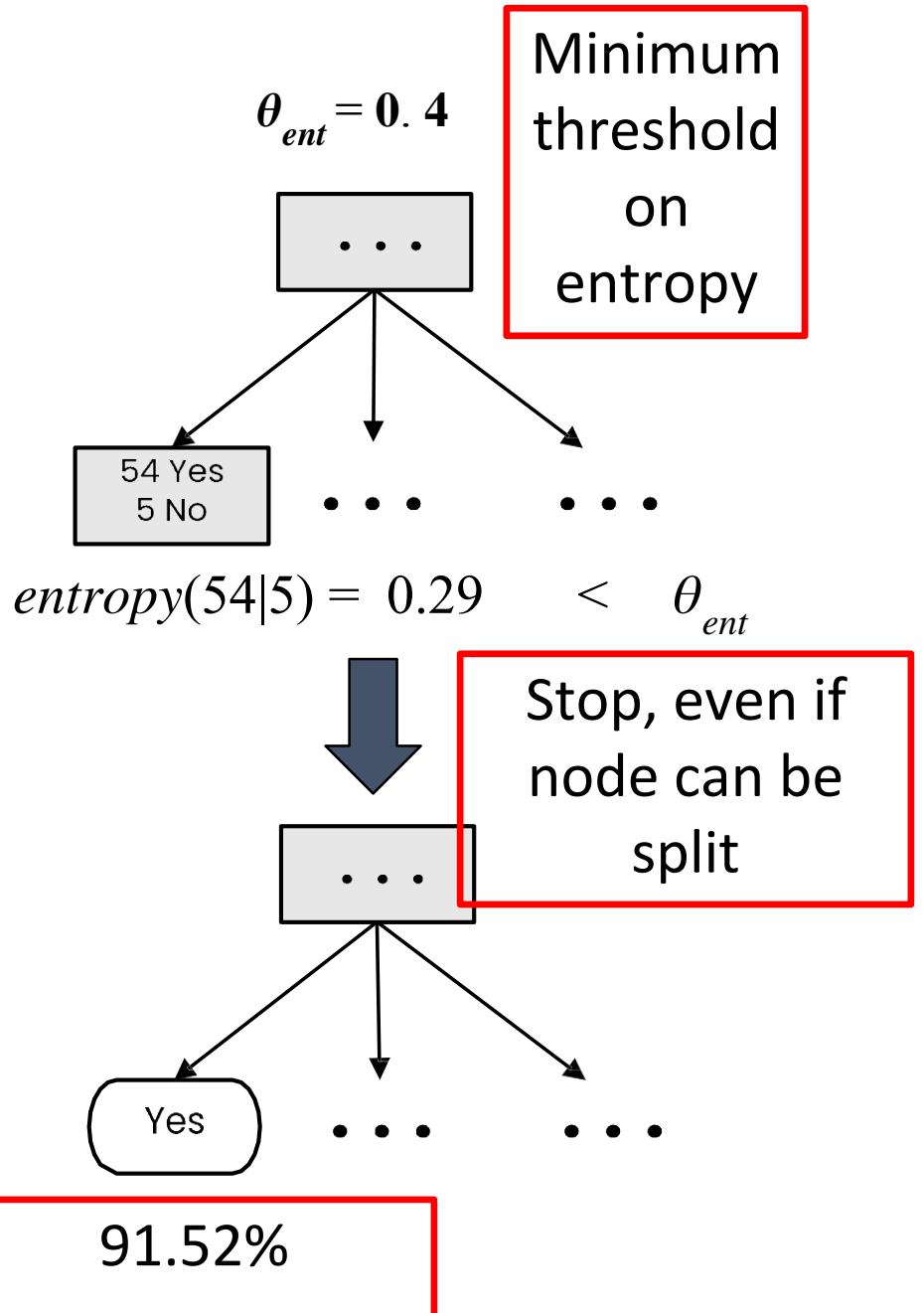
- Pruning is a technique that **reduces** the size of a decision tree by
  removing branches of the tree which provide little predictive power
- It is a **regularization** method that **reduces the complexity** of the final model, thus **reducing overfitting**
  - Decision trees are prone to overfitting!

- Pruning methods:
  - Pre-pruning: **Stop** the tree building algorithm before it **fully classifies the data**
  - Post-pruning: Build the **complete tree**, then replace some non-leaf nodes with leaf nodes if this improves **Testing error**

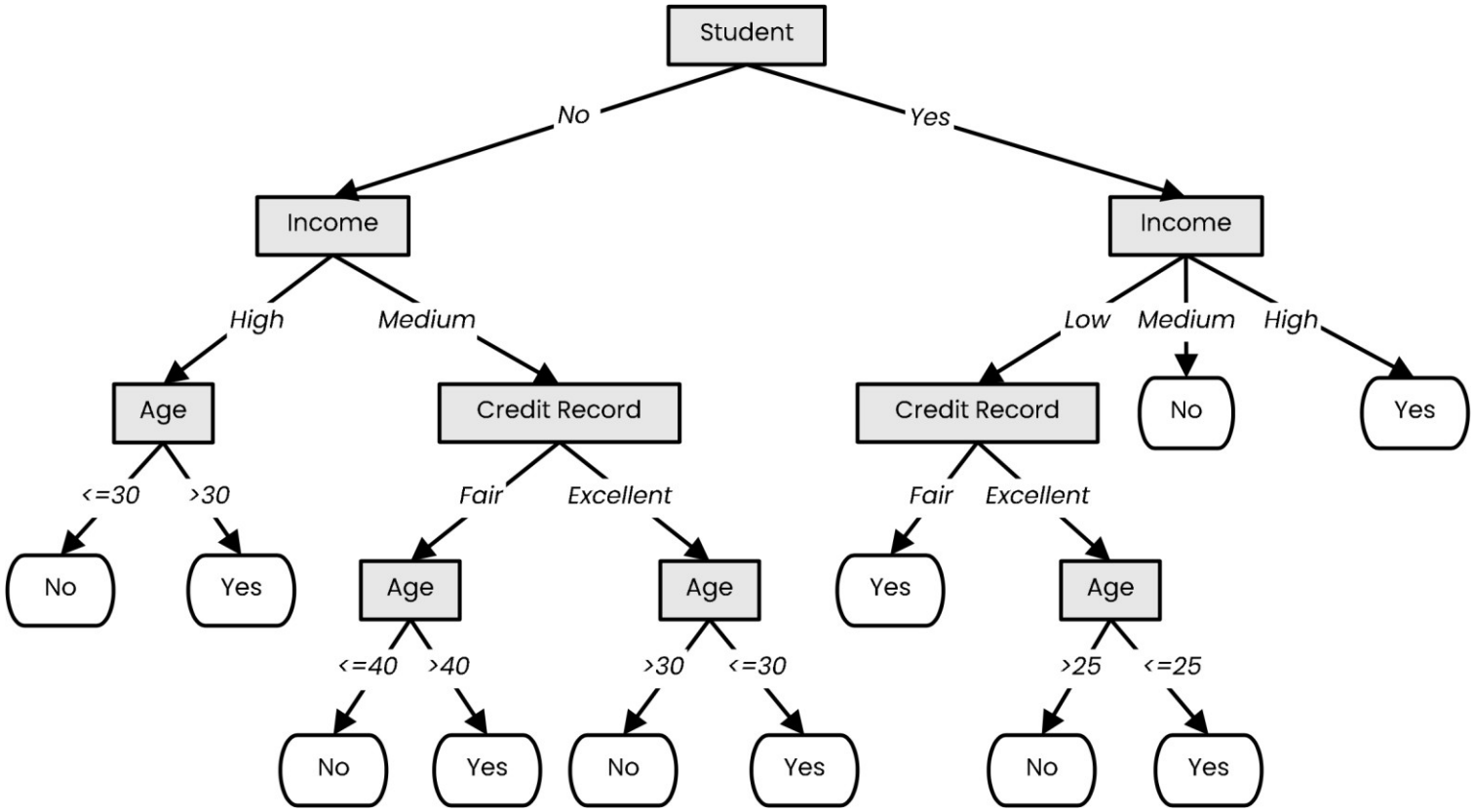
Pre-pruning

- Pre-pruning implies **early stopping:**
  - If some condition is met, the current node will **not be split**, **even if it is not 100% pure**
  - It will **become a leaf node** with **the label** of the **majority class** in the current set

- Common **stopping criteria** include setting a threshold on:
  - Entropy (or Gini Impurity) of the current set
  - Gain of the best-splitting attribute
  - **Depth** of the tree
  - Number of samples in the current set

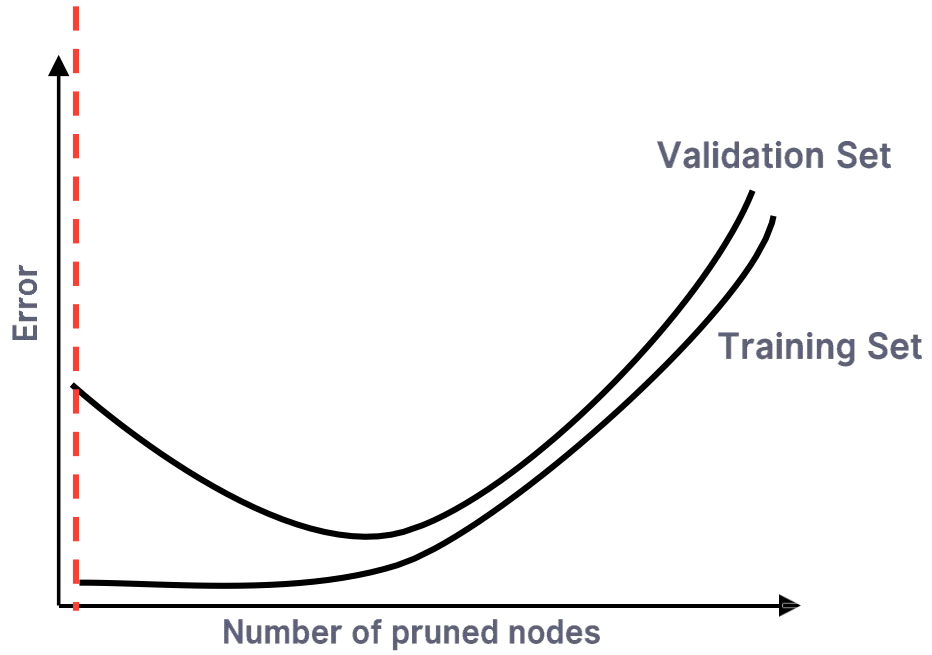


Post-pruning



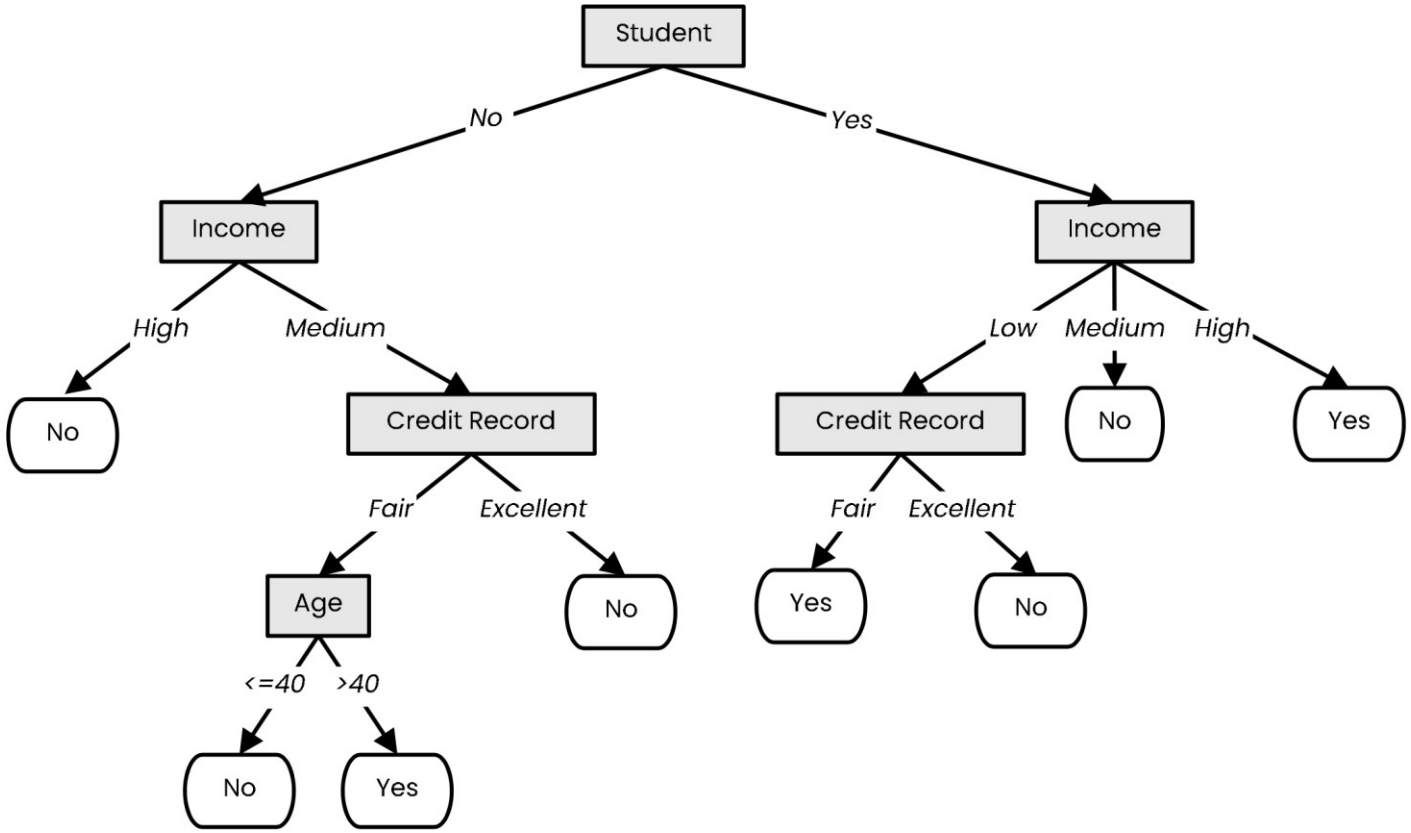
Grows a complete tree first and then removes branches that do not improve accuracy.

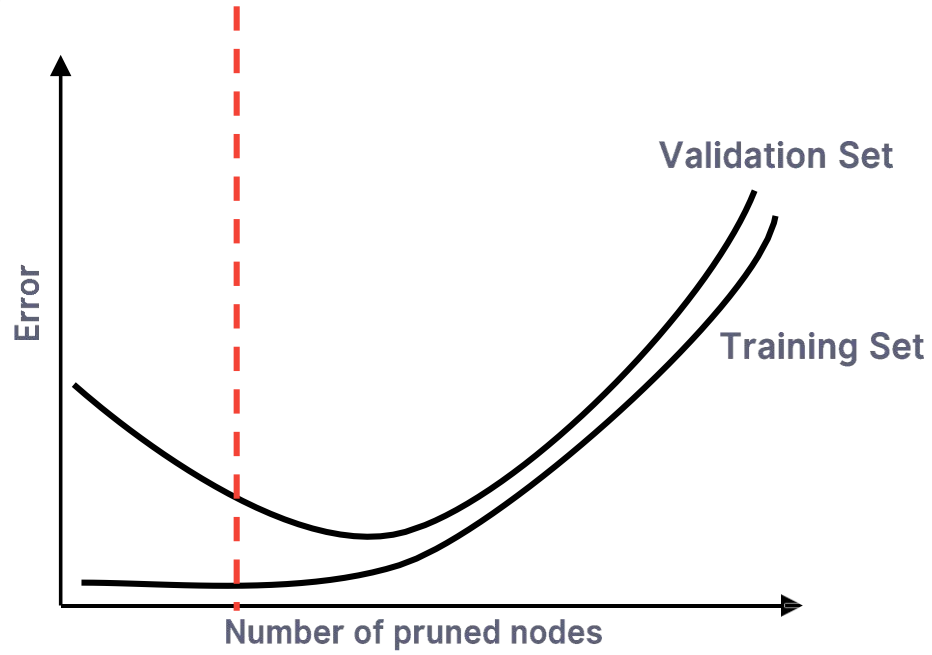
- Methods include:
- **Cost Complexity Pruning (CCP):** Removes nodes based on a penalty for complexity.
- **Reduced Error Pruning:** Evaluates each branch on a validation set and removes those that do not improve performance.



- **Prune** nodes in a **bottom-up manner**, if it decreases validation error

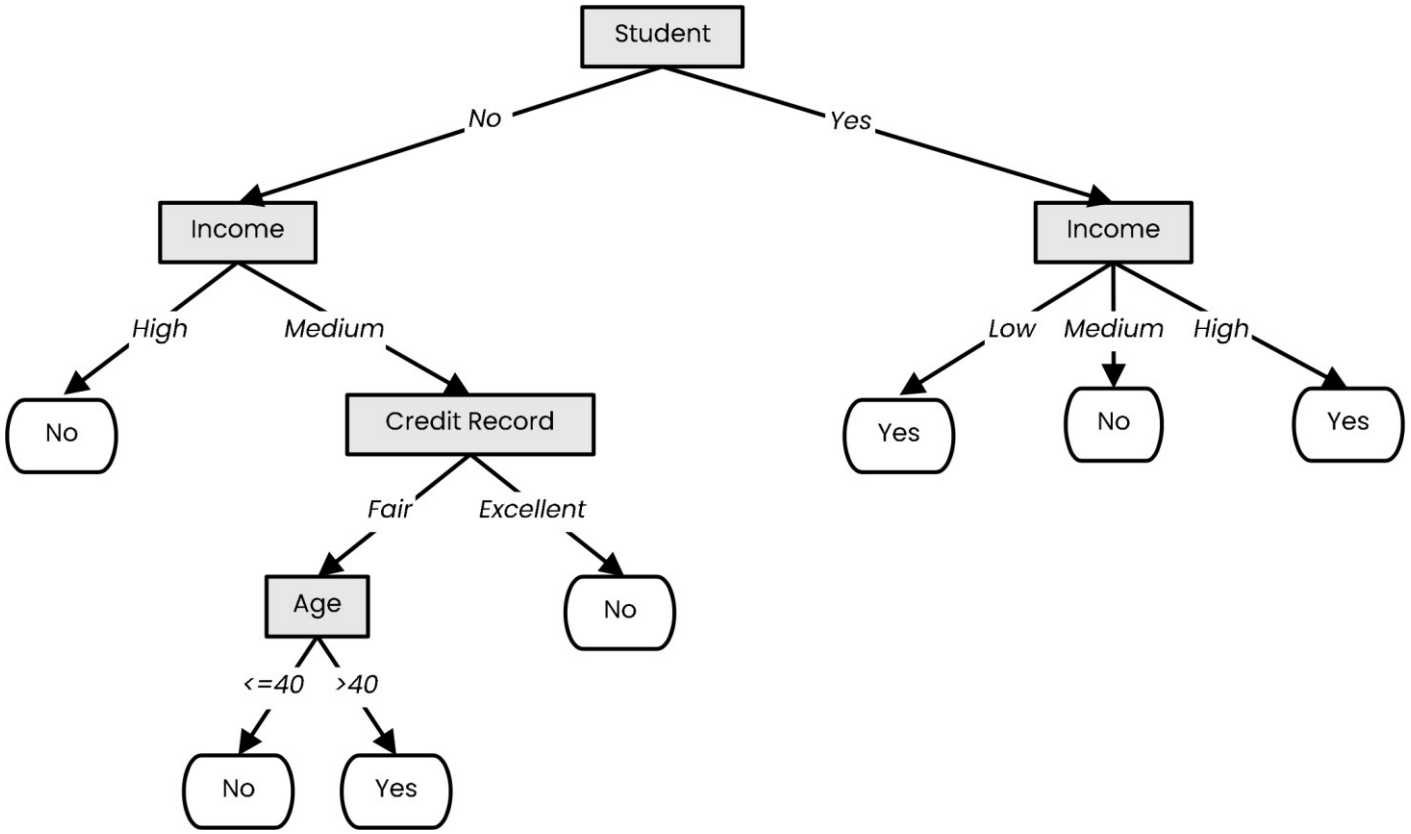
Post-pruning

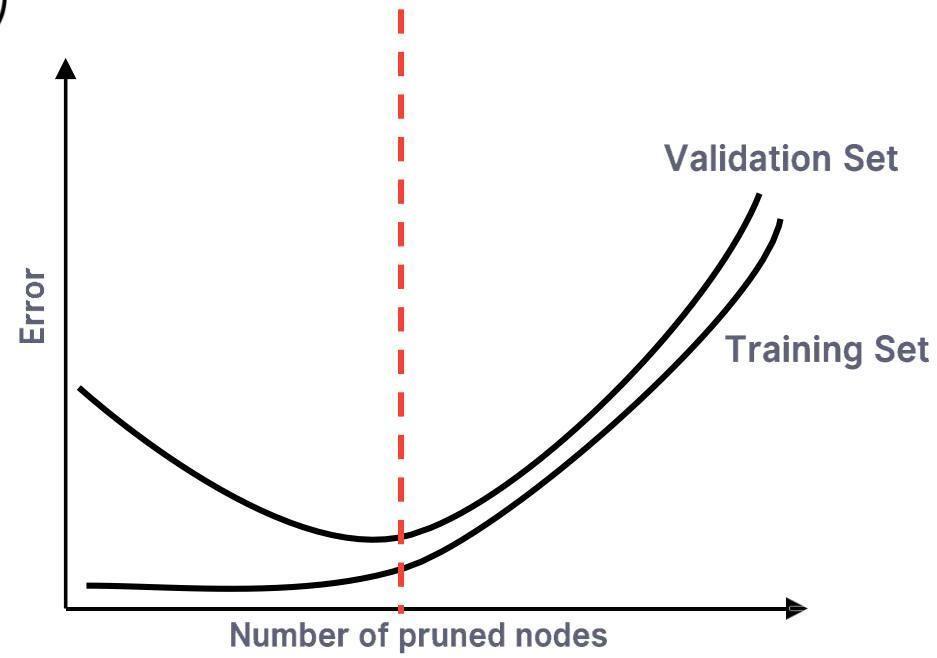




- **Prune** nodes in a **bottom-up manner**, if it decreases validation error

Post-pruning





- **Prune** nodes in a **bottom-up manner**, if it decreases validation error

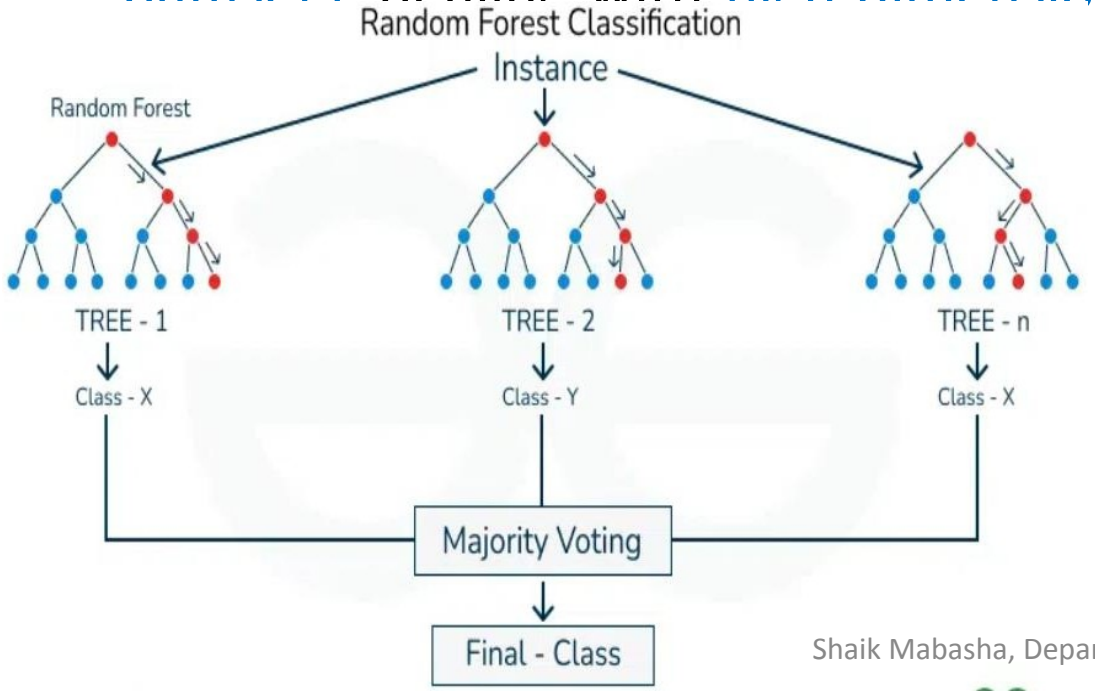
Enhancements to Basic Decision Tree Induction

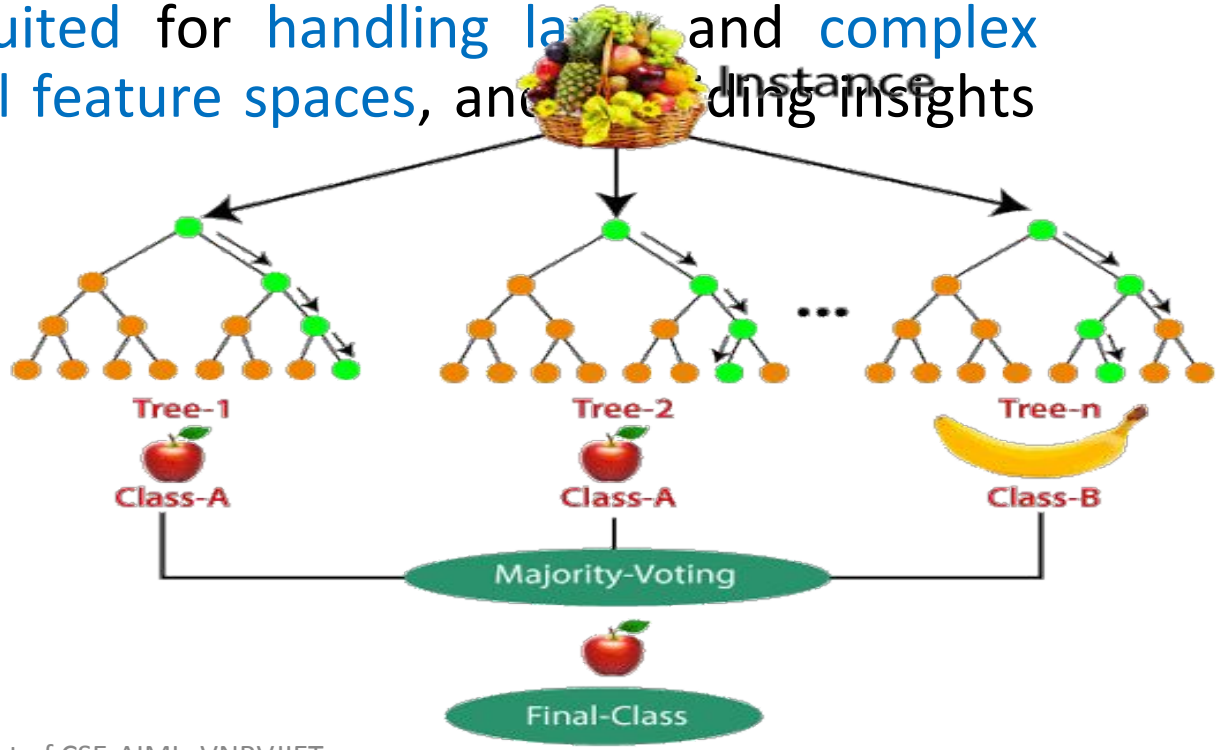
- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create **new attributes** based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Random Forest Model

Random Forest Model

- The Random forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees.
- We can reduce the amount of **overfitting issue** of **Decision Tree** by **majority voting and averaging their results of all decision trees** in **Random Forest**
- Random Forests are particularly well-suited for handling la and complex datasets, dealing with high-dimensional feature spaces, and ding insights





Random Forests Procedure

- Random Forests:
  - Instead of building a single decision tree and use it to make predictions, build many slightly different trees and combine their predictions
- We have **a single data set, so how do we obtain slightly different trees?**

1. **Bagging** (Bootstrap **Aggregating**):
   - Take **random** subsets of data points from the training set to create N smaller data sets
   - Fit a decision tree on each subset
2. **Random Subspace Method** (also known as **Feature selection/Feature Bagging**):
   - Fit N different decision trees by constraining each one to operate on a random subset of features

Bootstrap sample

- To build a tree first we need to take a **bootstrap sample**
  - **How?**
    - From our **n_samples** data points, we **repeatedly** draw **an sample** randomly with replacement **n_samples** times
    - **Replacement** meaning the **same sample can be picked multiple times**
  - **Example on Boot Strap Sample -**
    - Creating a bootstrap sample of the list **['a', 'b', 'c', 'd'].**
    - A possible bootstrap sample would be **['b', 'd', 'd', 'c'].**
    - Another possible sample would be **['d', 'a', 'd', 'a']**

- This will create a dataset that is as big as the original dataset, but **some data points will be missing from it** , and **some will be repeated**

How to do Bootstrap

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

id
2
1
3
1
4
4

id
4
1
3
0
0
2

id
3
3
2
5
1
2

From our **n_samples** data points (whole datset) , we **repeatedly** draw an sample randomly with replacement **n_samples** times

sampling data in ensemble methods

1. **Bagging (Bootstrap Aggregating):** Randomly selects subsets of data **with replacement** (some data points may appear multiple times in a sample, while others may be missing).

2. Pasting: Similar to bagging, but **without replacement** (each sample is drawn uniquely).

3. Random Subspace Method: Instead of sampling data points, it randomly selects **subsets of features**.

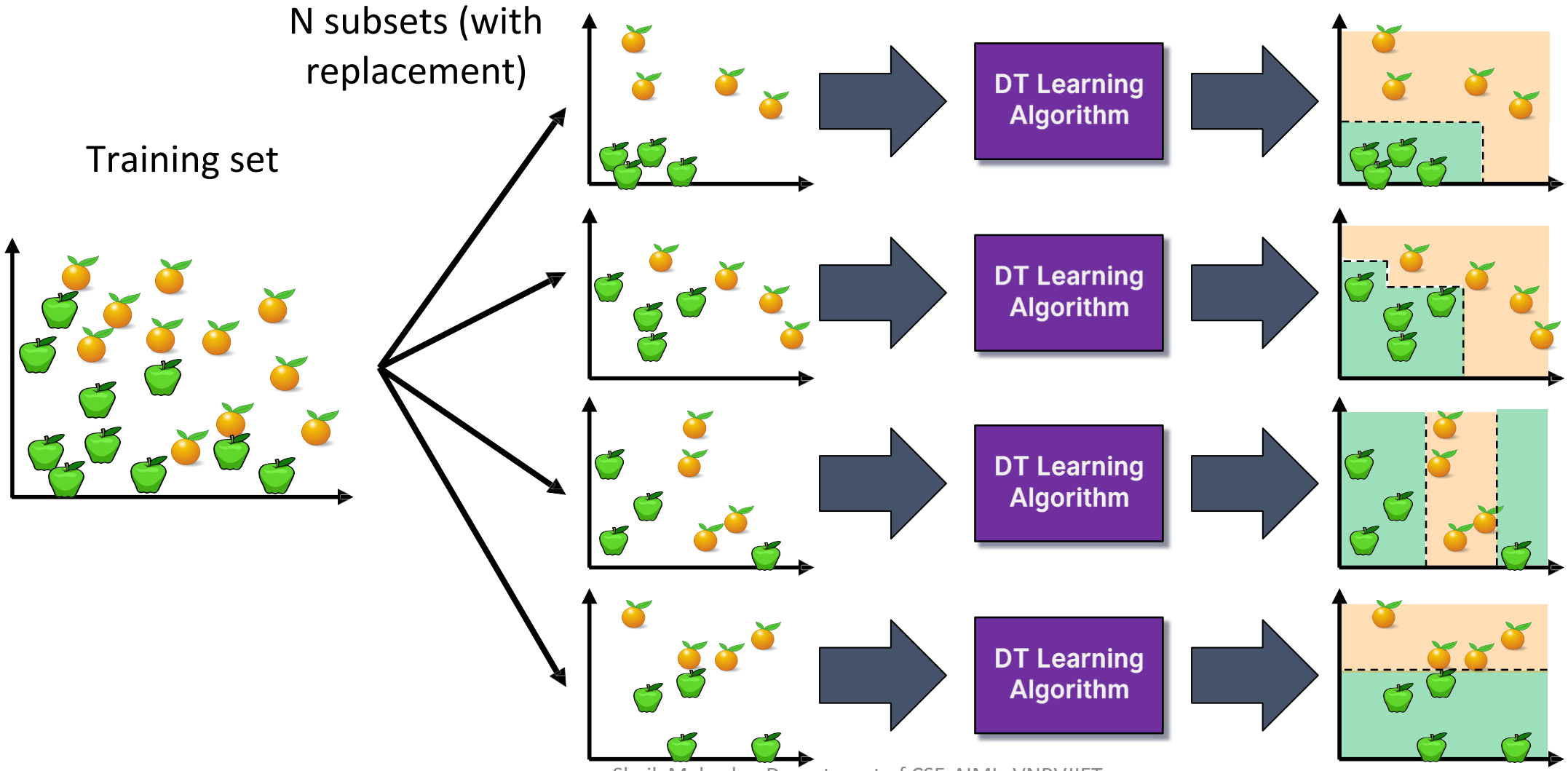
sampling data in ensemble methods

4. Random Patches Method: Combines **random sampling of data points (like bagging)** and **random selection of features (like random subspace)**.

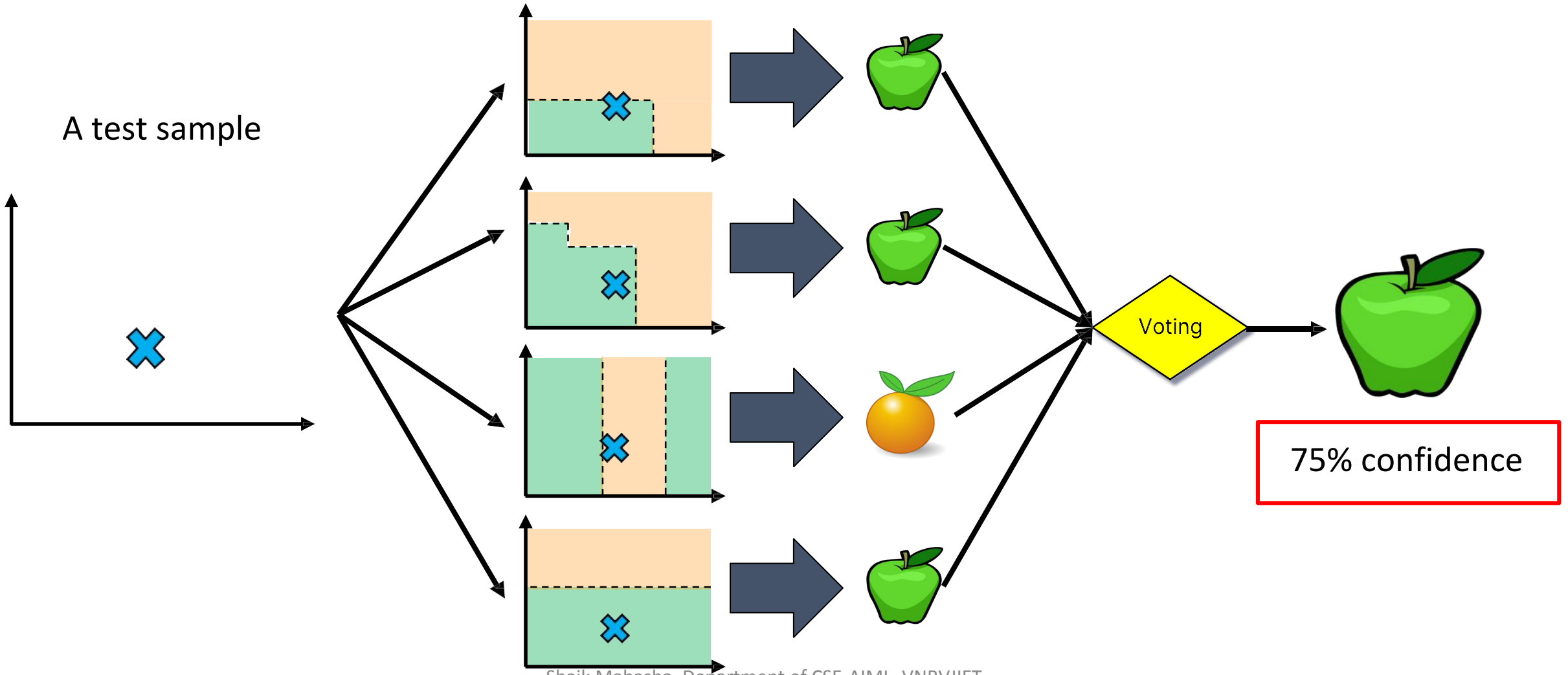
5. Boosting (Sequential Sampling): Unlike bagging, boosting assigns **higher weights** to misclassified samples in each iteration.

6. Cross-Validation Based Sampling: Uses k-fold cross-validation to create different training subsets.

Bagging at training time



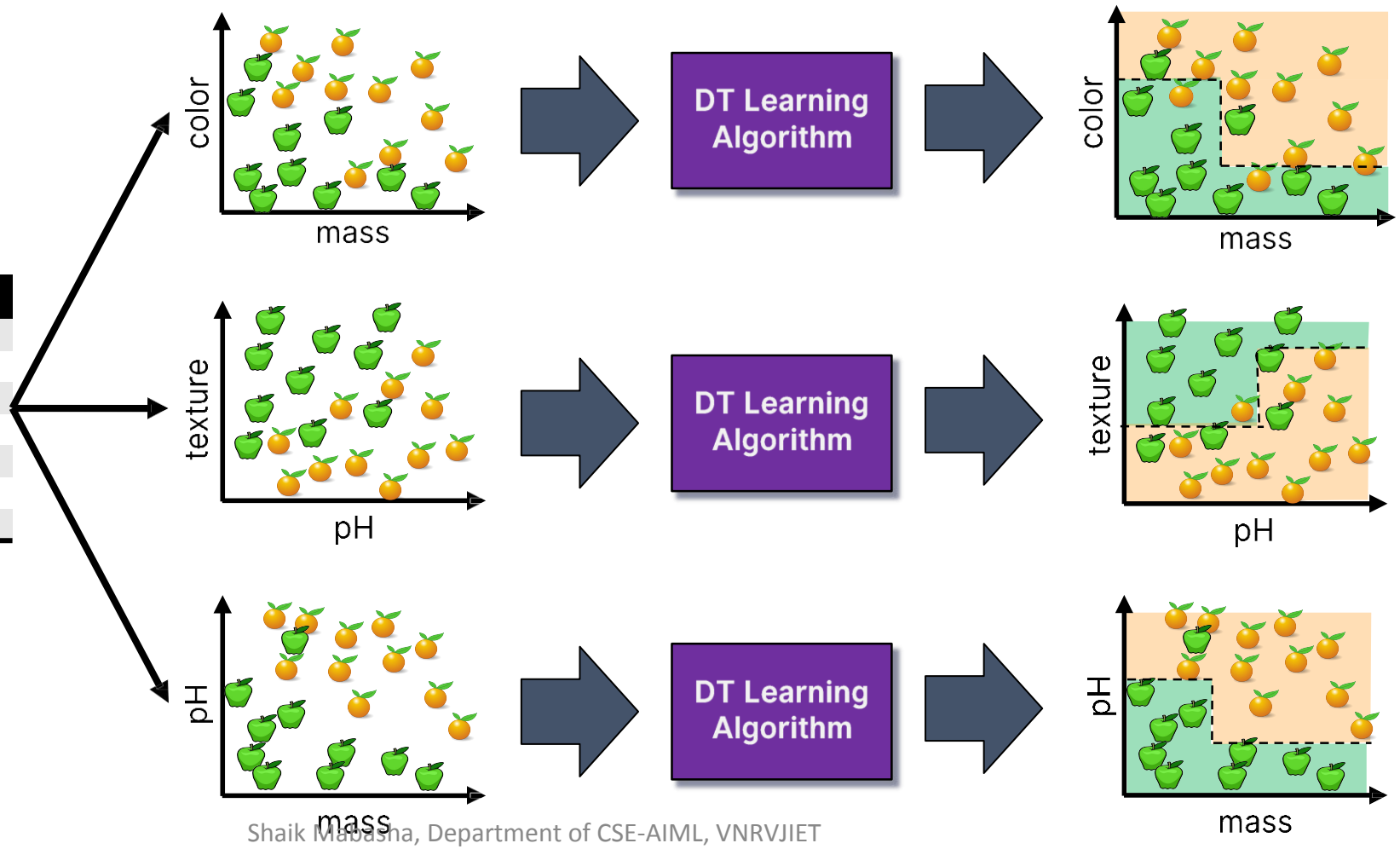
Bagging (**Bootstrap** **Aggregating**) at **Testing/inference time**



Random **Subspace** (Feature Bagging/Feature selection) Method at training time

Training data

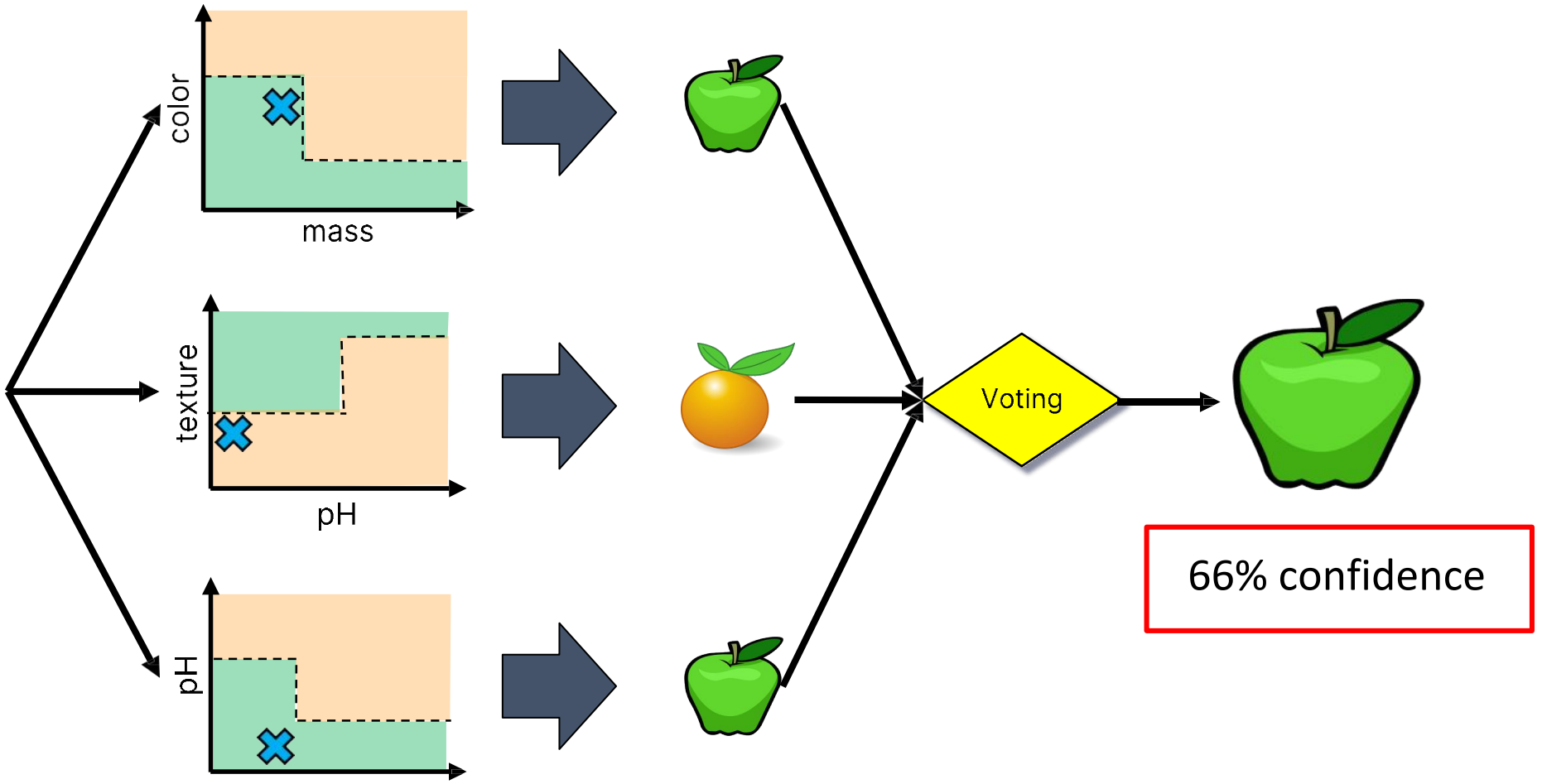
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random **Subspace (Feature Bagging/Feature selection)** Method at testing time

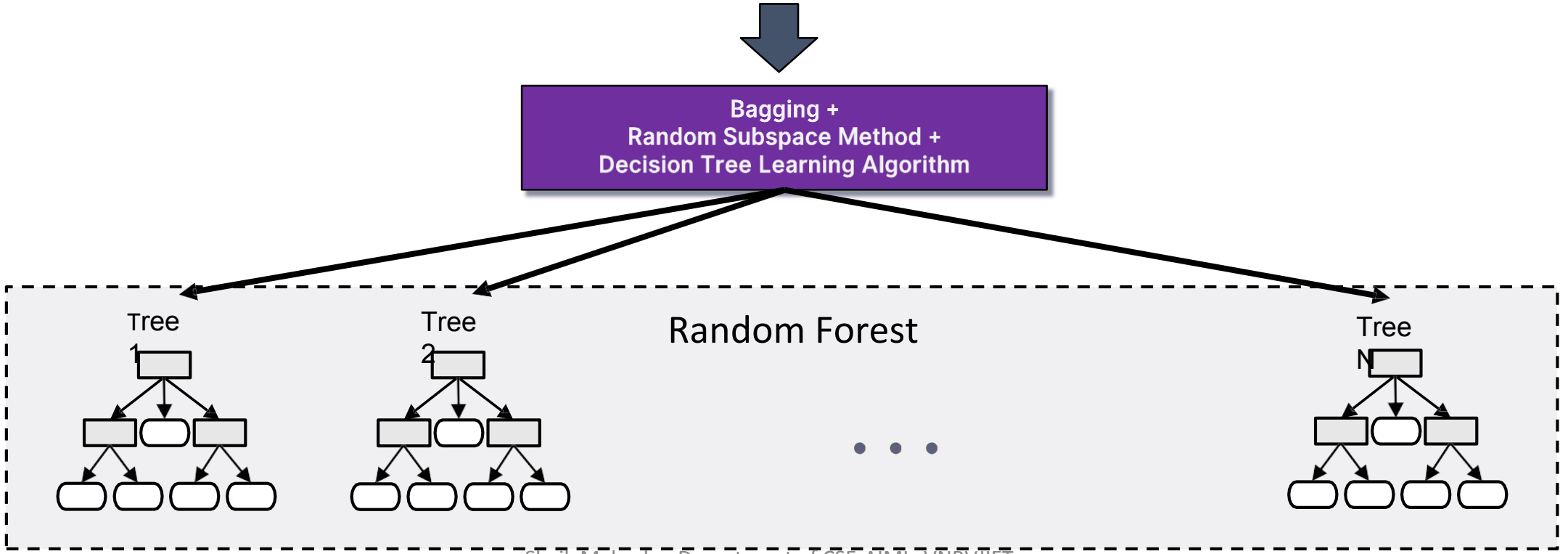
A test sample

87	Red	Smooth	3.1



Random Forests

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random Forest Model

Random Forest Algorithm:

Step 1: Select random samples from a given data or training set.

Step 2: This algorithm will construct a decision tree for every training

data. Step 3: **Voting** will take place by **averaging** the decision tree.

Step 4: Finally, select the **most voted prediction** result as the **final prediction result.**

Random **Subspace (Feature Bagging/Feature selection)–Example 2**

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

x_0, x_1

id
2
1
3
1
4
4

x_2, x_3

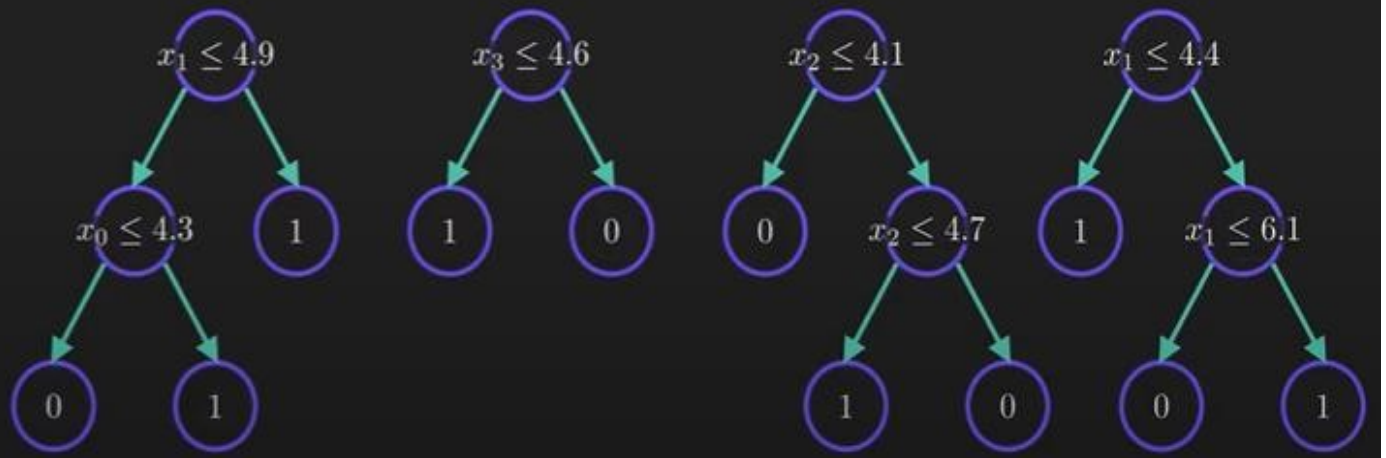
id
4
1
3
0
0
2

x_2, x_4

id
3
3
2
5
1
2

x_1, x_3

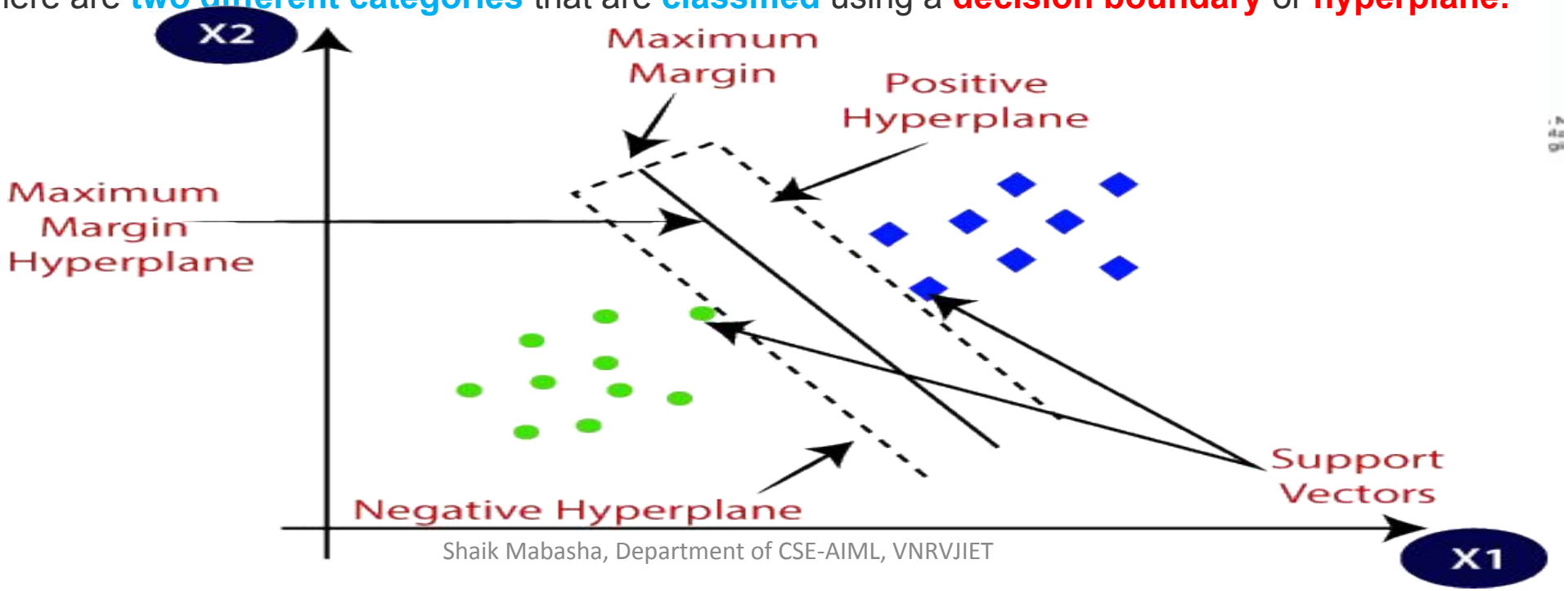
2.8	6.2	4.3	5.3	5.5



Support Vector Machine (SVM)

Support Vector Machine (SVM)

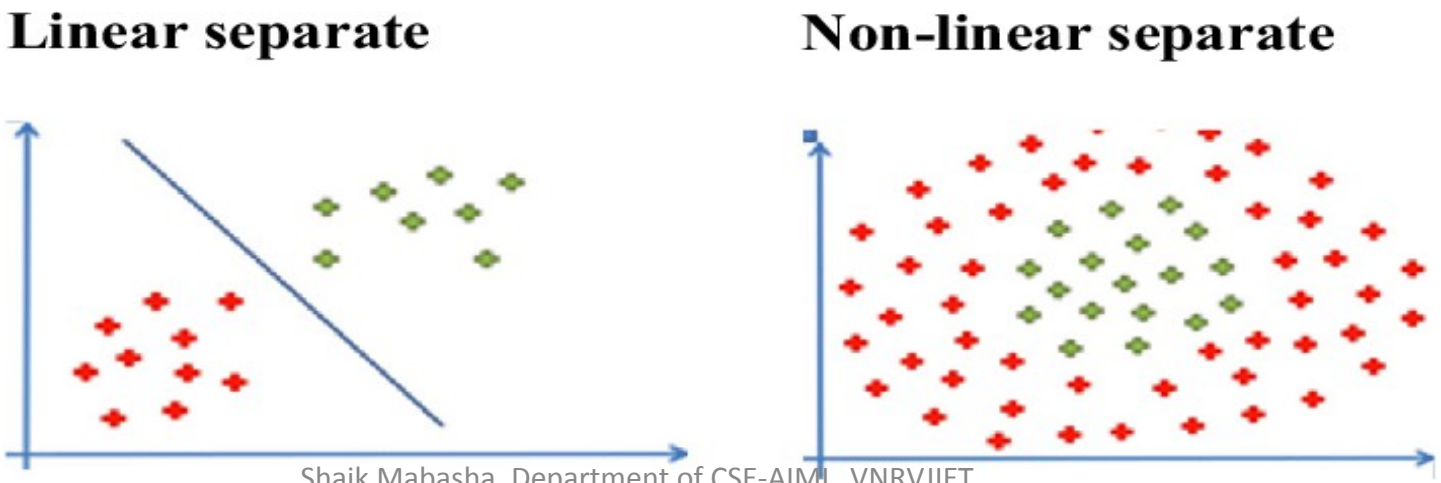
- Support Vector Machine or SVM is one of the most popular **Supervised Learning algorithms**, which is used for **Classification as well as Regression problems**. However, primarily, it is used for **Classification problems** in Machine Learning.
- **The goal** of the SVM algorithm is to create the **best line** or **decision boundary** that can **segregate n-dimensional space** into **classes** so that we can easily **put the new data point** in the **correct category in the future**.
- This **best decision boundary** is called a **hyperplane.**
- SVM chooses the **extreme points/vectors** that **help in creating** the **hyperplane.** These **extreme cases are** called as **support vectors**, and hence algorithm is termed as **Support Vector Machine**. Consider the below diagram in which there are **two different categories** that are **classified** using a **decision boundary** or **hyperplane:**



Support Vector Machine

Types of SVM (SVM)

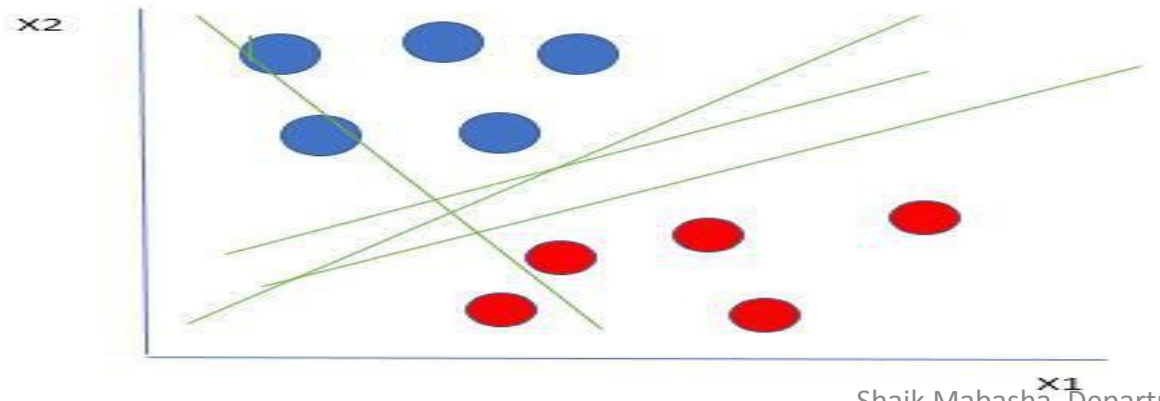
- **Linear SVM:** Linear SVM is used for **linearly separable data**, which means if a dataset can be **classified** into **two classes** by using **a single straight line**, then such data is termed as **linearly separable data,** and classifier is used called as **Linear SVM classifier**.
- **Non-linear 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.**

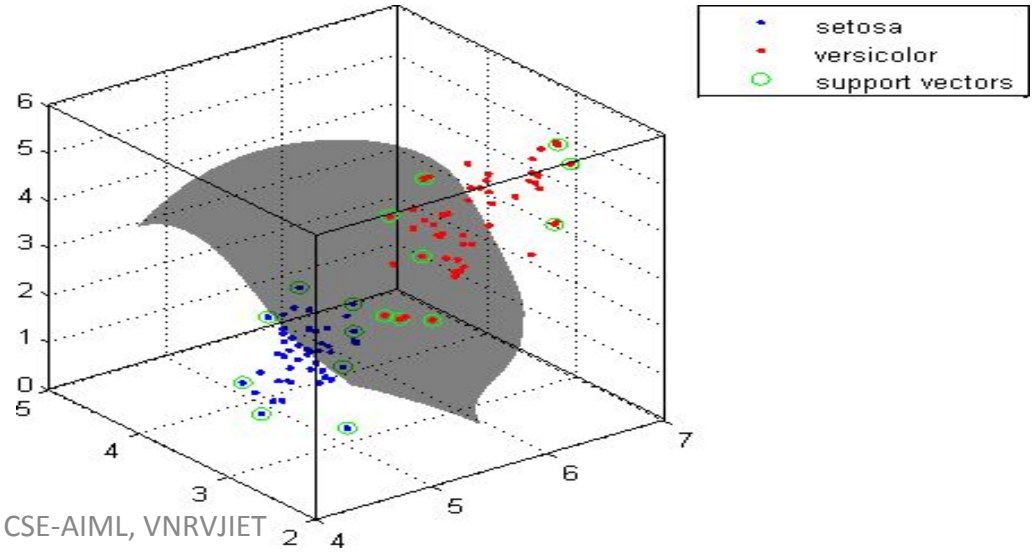


Support Vector Machines (SVM)

Hyper-plane in the SVM algorithm:

- **Hyperplane:** There can be **multiple lines/decision boundaries** to segregate the **classes** in **n-dimensional space**, but we need to **find out** the **best decision boundary** that helps to classify the data points. This **best boundary** is known as the **hyperplane of SVM**.
- The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features , then **hyperplane** will be **a straight line**. And if there are 3 or more features, then **hyperplane** will be **a N-dimension plane etc.**.





Support Vector Machines (SVM)

Hyper-plane in the SVM algorithm:

- Mathematically, a **hyper-plane** in an **n-dimensional Euclidean space** can be defined by a linear equation:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$

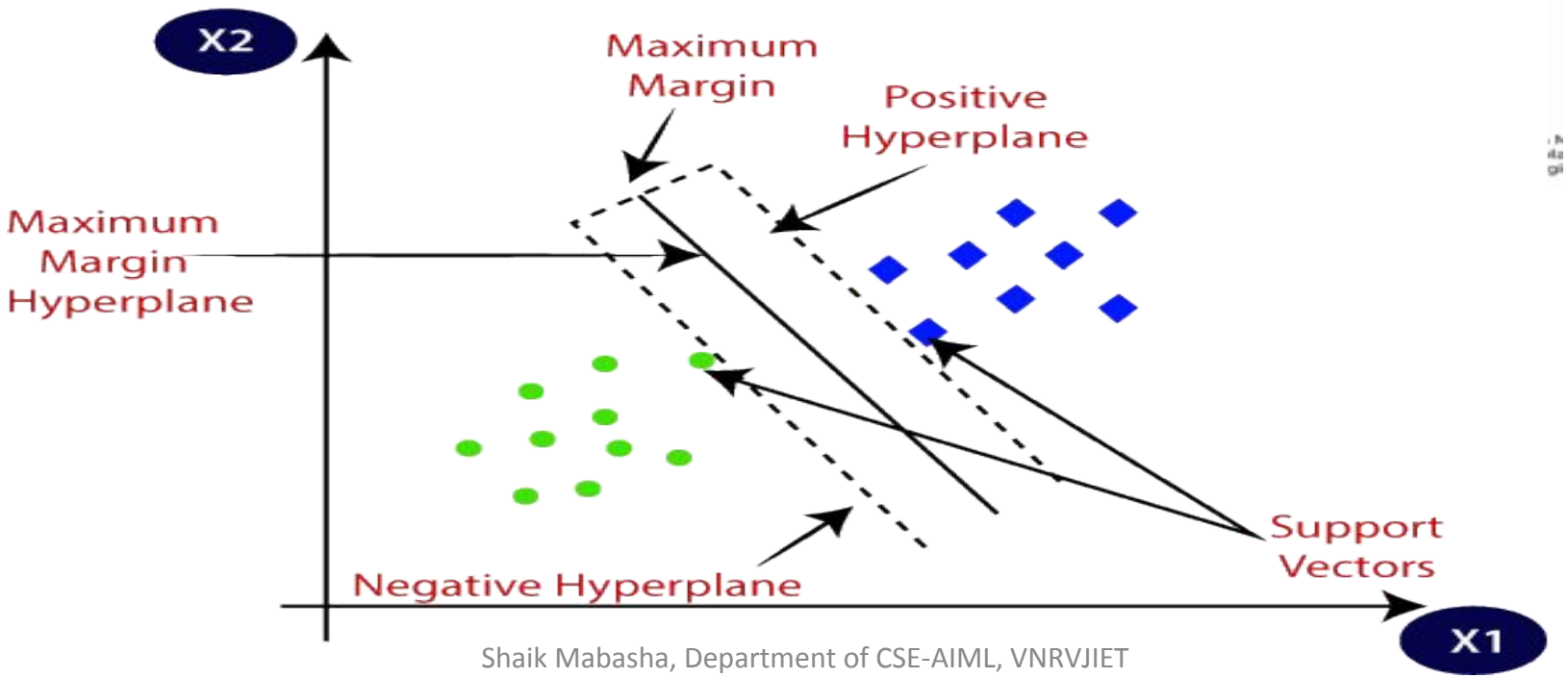
- Here, $x_1, x_2, \ldots, x_n$ are the **coordinates** in the **n-dimensional space**,

$a_1, a_2, \ldots, a_n$ are the **coefficients** that **determine** the **orientation of the hyperplane**, and b is a **constant term** that determines the **offset** of the hyperplane from the origin.

Support Vector Machines (SVM)

Support Vectors in the SVM algorithm:

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

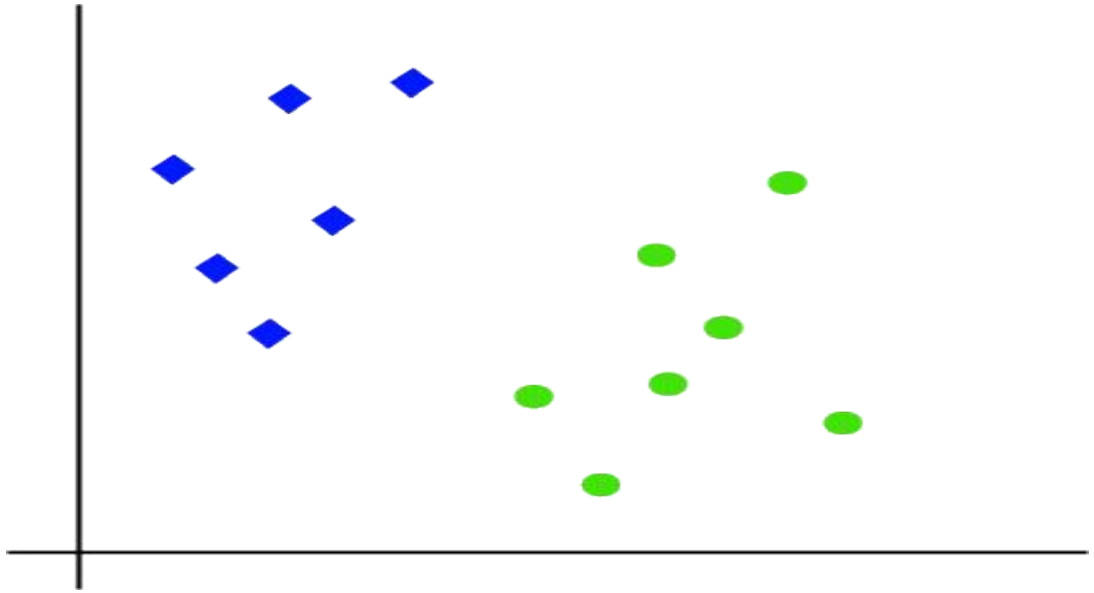


- **Support Vector Machines (SVM) :**

How does SVM works?

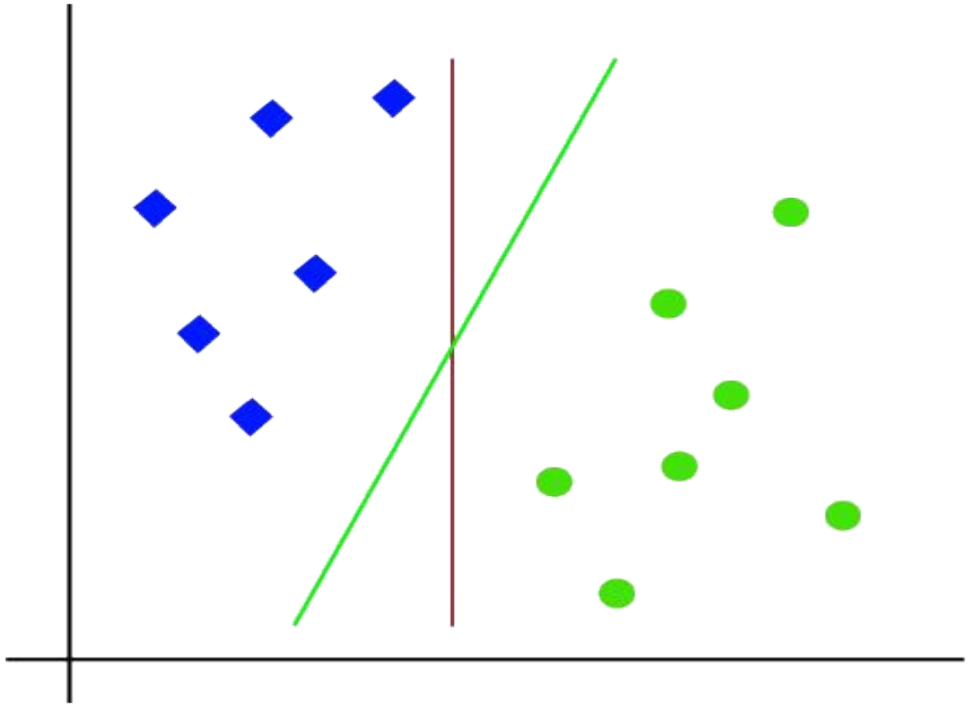
Linear SVM:

- The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has **two tags (green and blue),** and the dataset has two features **x1 and x2.** We want a classifier that can **classify** the pair(x1, x2) of coordinates in either **green or blue.**



Support Vector Machines (SVM)

- So as it is **2-d space** so by just using **a straight line**, we can easily **separate** these two classes. But there can be **multiple lines** that can separate these classes.

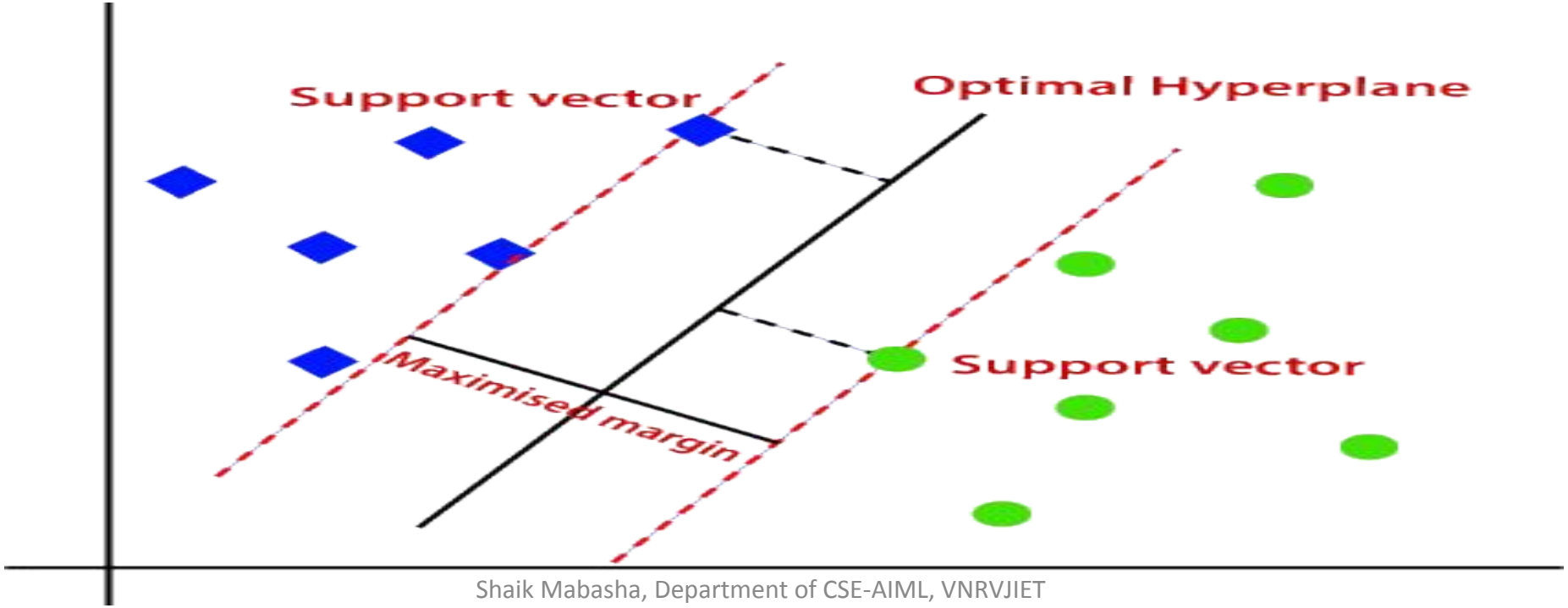


- SVM algorithm finds the **closest point of the lines** from **both the classes.** These points are **called support vectors.**
- Hence, the SVM algorithm helps to **find the best line or decision boundary**; this best boundary or region is called as a **hyperplane.**

Support Vector Machines (SVM)

- The distance **between** the **vectors** and the **hyperplane** is called as **margin**.
- We always create **a hyperplane** that has a **maximum margin**, which means the **maximum distance between the data points.**
- The goalof SVM is to **maximize this margin**. The **hyperplane** with maximum margin is called

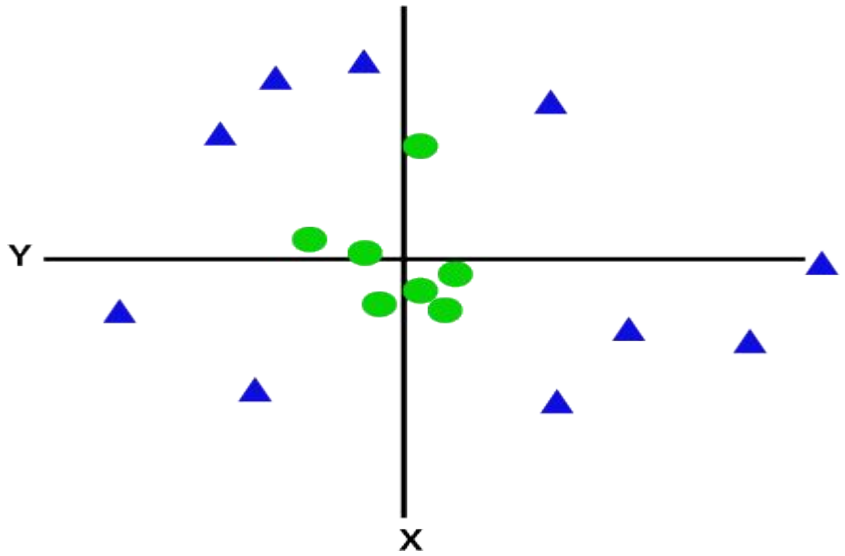
  the **optimal hyperplane**.

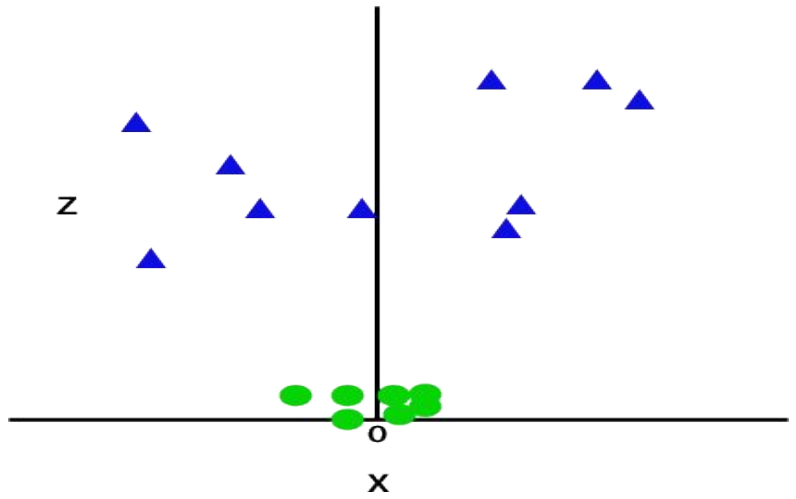


Support Vector Machine (SVM)

Non-Linear SVM:

- If data is linearly arranged, then we can separate it by using a straight line, but for **non-linear data,** we **cannot draw a single straight line.**

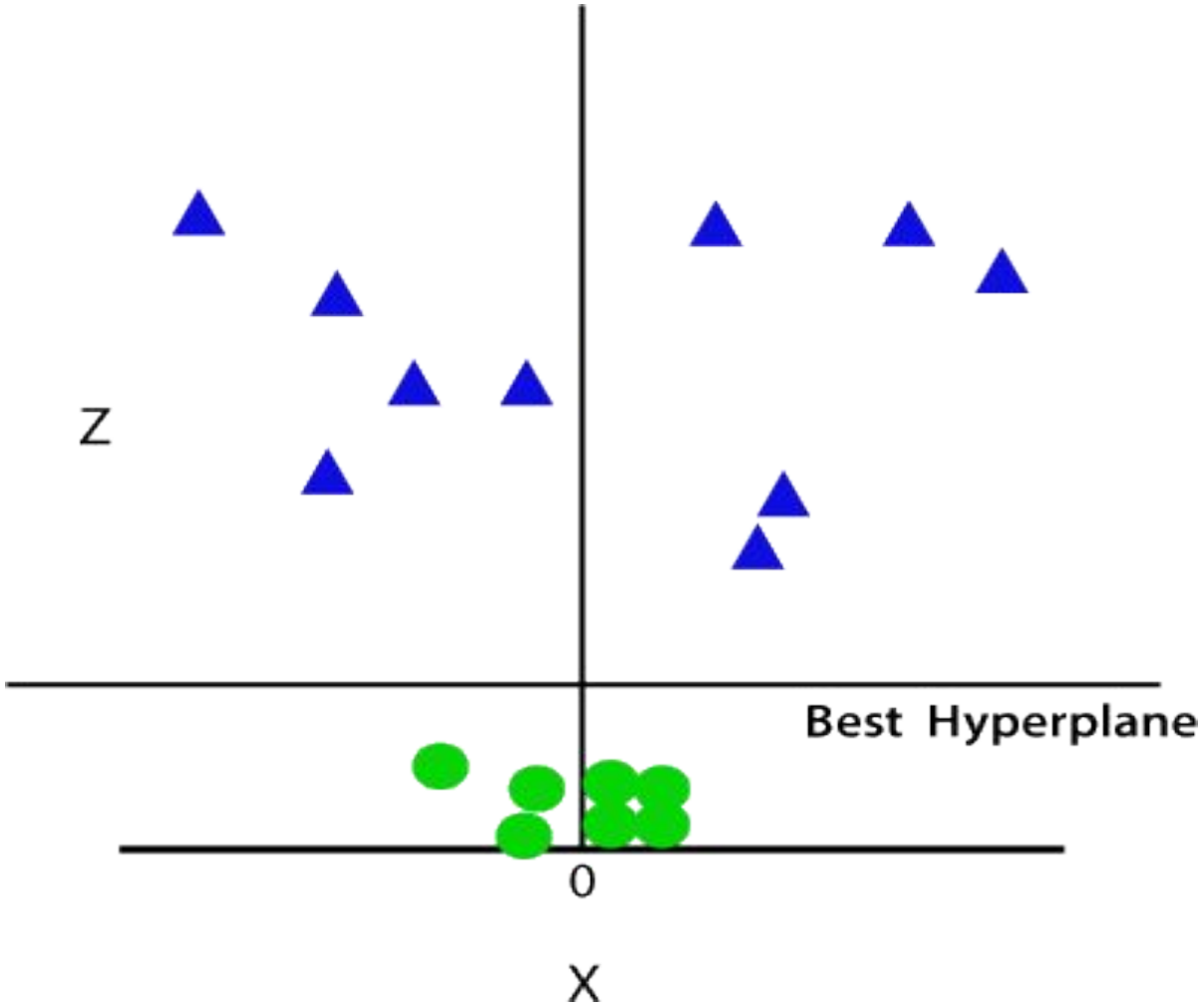




- So to separate these data points, we need to **add one more dimension**.
- For linear data, we have used two dimensions **x and y,** so for non-linear data, we will add a **third dimension Z.**
- It can be calculated as: $z=x^2+y^2$

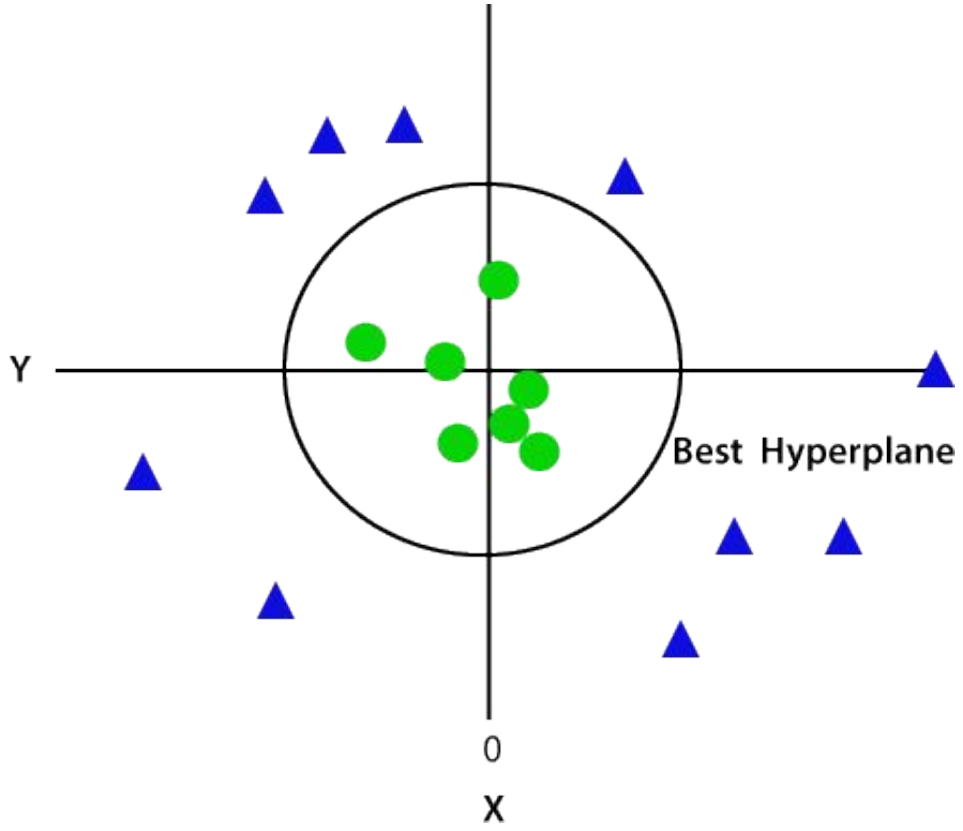
Support Vector Machines (SVM) :

- So now, SVM will divide the datasets into classes in the following way.



Support Vector Machine (SVM)

- Since we are in **3-d Space**, hence it is looking like a plane parallel to the x-axis.
- If we **convert** it in **2d space** with z=1, then it will become as:

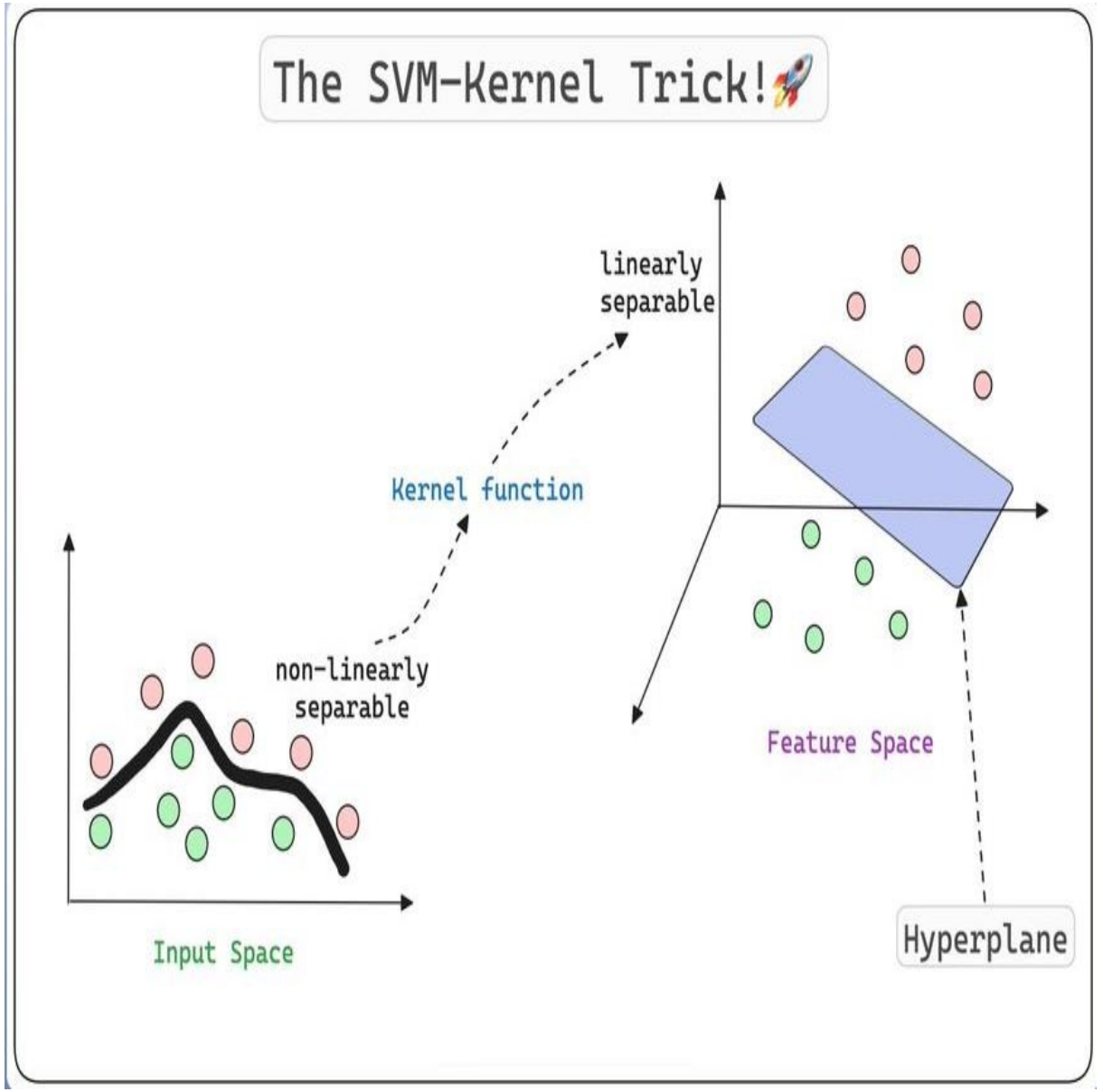


- Hence we get a circumference of radius 1 in case of non-linear data.

Support Vector Machine (SVM)

SVM Kernel:

- SVM algorithms use a set of **mathematical functions** that are defined as **the kernel**.
- **The function of kernel** is to take data as **input** and **transform it into the required form.**
- For example: The SVM kernel takes **low-dimensional input space** and **transforms** it into **higher-dimensional space**, ie it converts non-separable problems to separable problems.
- Different SVM algorithms use **different types of kernel functions**. These functions can be different types. For example linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid etc.



Support Vector Machine (SVM)

Advantages of SVM:

- Effective in **high dimensional cases**
- Its **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**

Applications of SVM

Handwriting recognition – We use SVMs to recognize handwritten characters used widely.

Face detection – SVMc classify parts of the image as a face and non-face and create a square boundary around the face.

Text and hypertext categorization – SVMs allow Text and hypertext categorization for both inductive and transductive models. They use training data to classify documents into different categories. It categorizes on the basis of the score generated and then compares with the threshold value.

Classification of images – Use of SVMs provides better search accuracy for image classification. It provides better accuracy in comparison to the traditional query-based searching techniques.

Bioinformatics – It includes protein classification and cancer classification. We use SVM for identifying the classification of genes, patients on the basis of genes and other biological problems.

Protein fold and remote homology detection – Apply SVM algorithms for protein remote homology detection.

Content beyond the Syllabus

Bias and Variance in Machine Learning

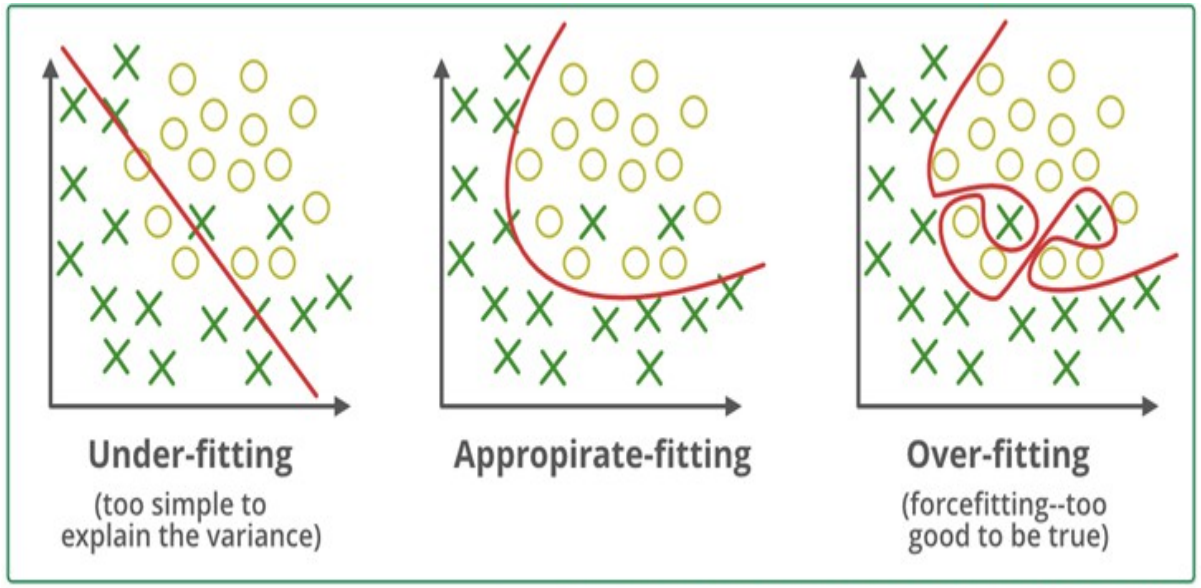
1.Bias: It is the error due to the model's inability to represent the true relationship between input and output accurately.

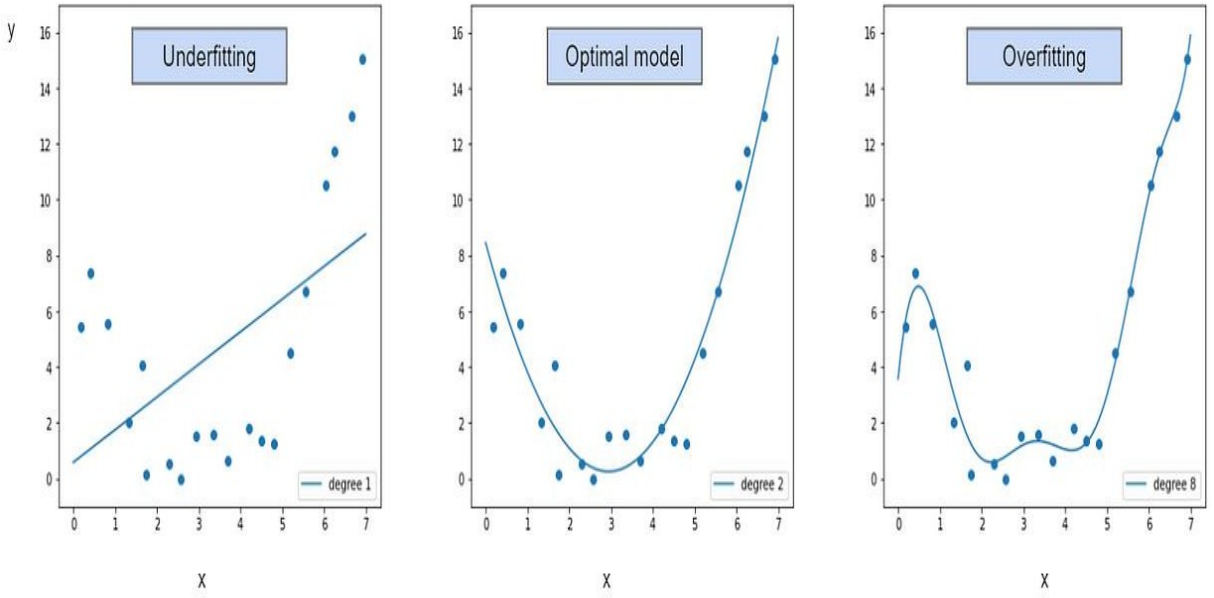
- When a model has poor performance both on the training and testing data means **high bias** because of the **simple model**, **indicating under-fitting.**
- Bias refers to the error due to overly simplistic assumptions in the learning algorithm. These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data.

2. Variance: is **the error** that occurs due to **sensitivity to small changes** in the training set.

- It's the **variability** of the model's predictions for **different instances** of training data.
- **High variance** occurs when a model learns the **training data's noise** and **random fluctuations** rather than the underlying pattern. As a result, the model performs well on the training data but poorly on the testing data, **indicating overfitting.**

Under-fitting vs Over-fitting





High bias & Low variance

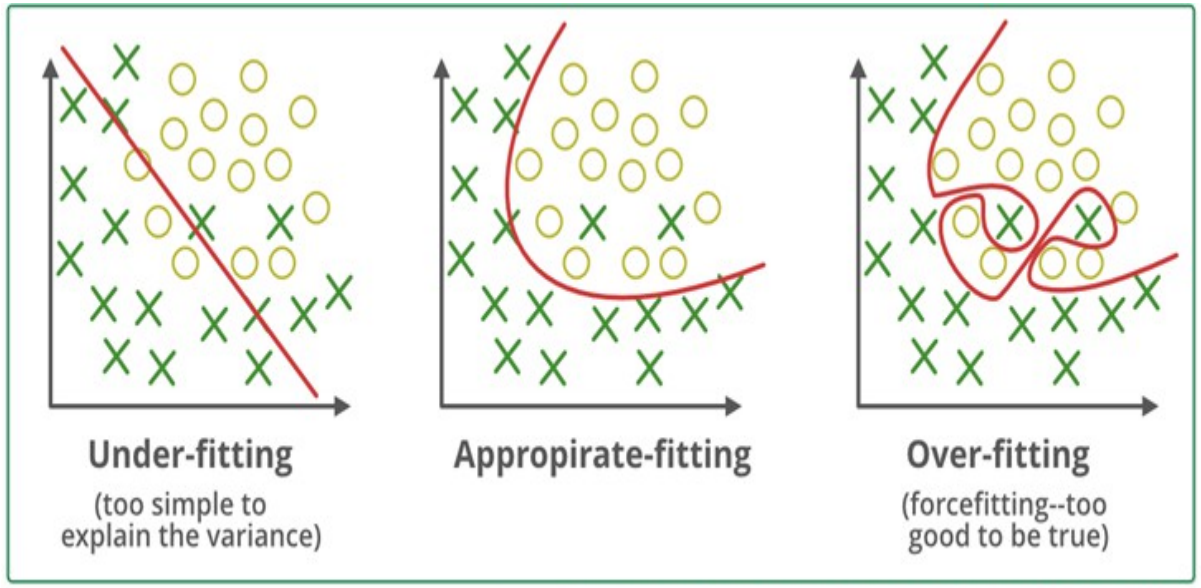
Low Bias & Low variance

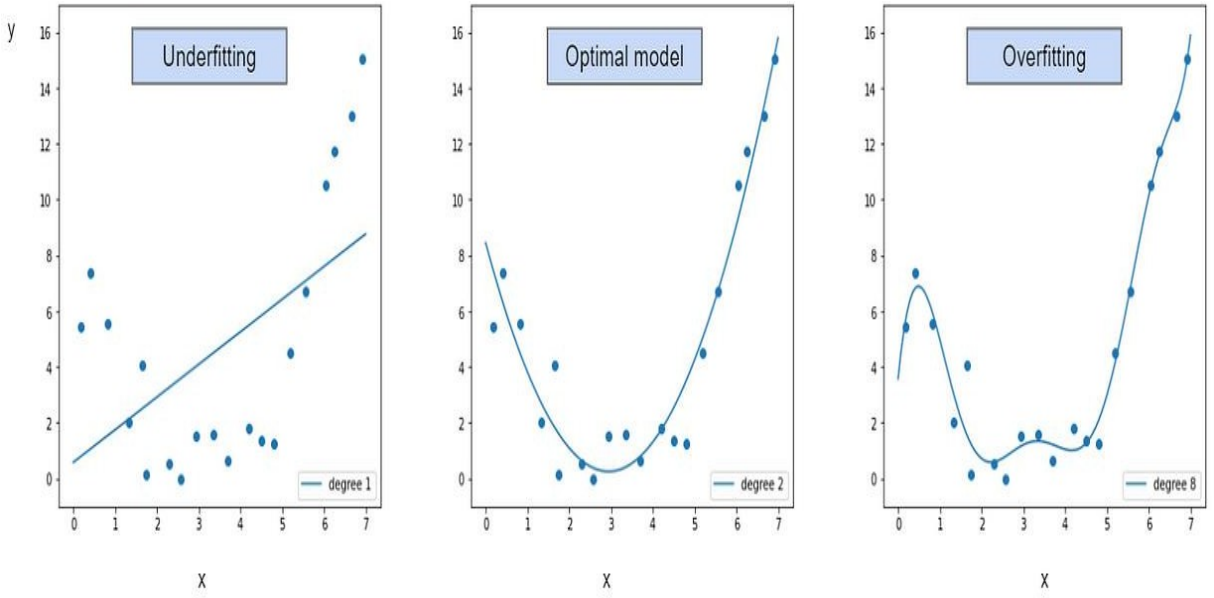
Low Bias High variance

Reasons for Under-fitting:

- The size of the training dataset used is not enough.
- The **model is too simple**, So it may be not capable to represent the complexities in the data.
- The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
- Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.

Under-fitting vs Over-fitting





High bias & Low variance

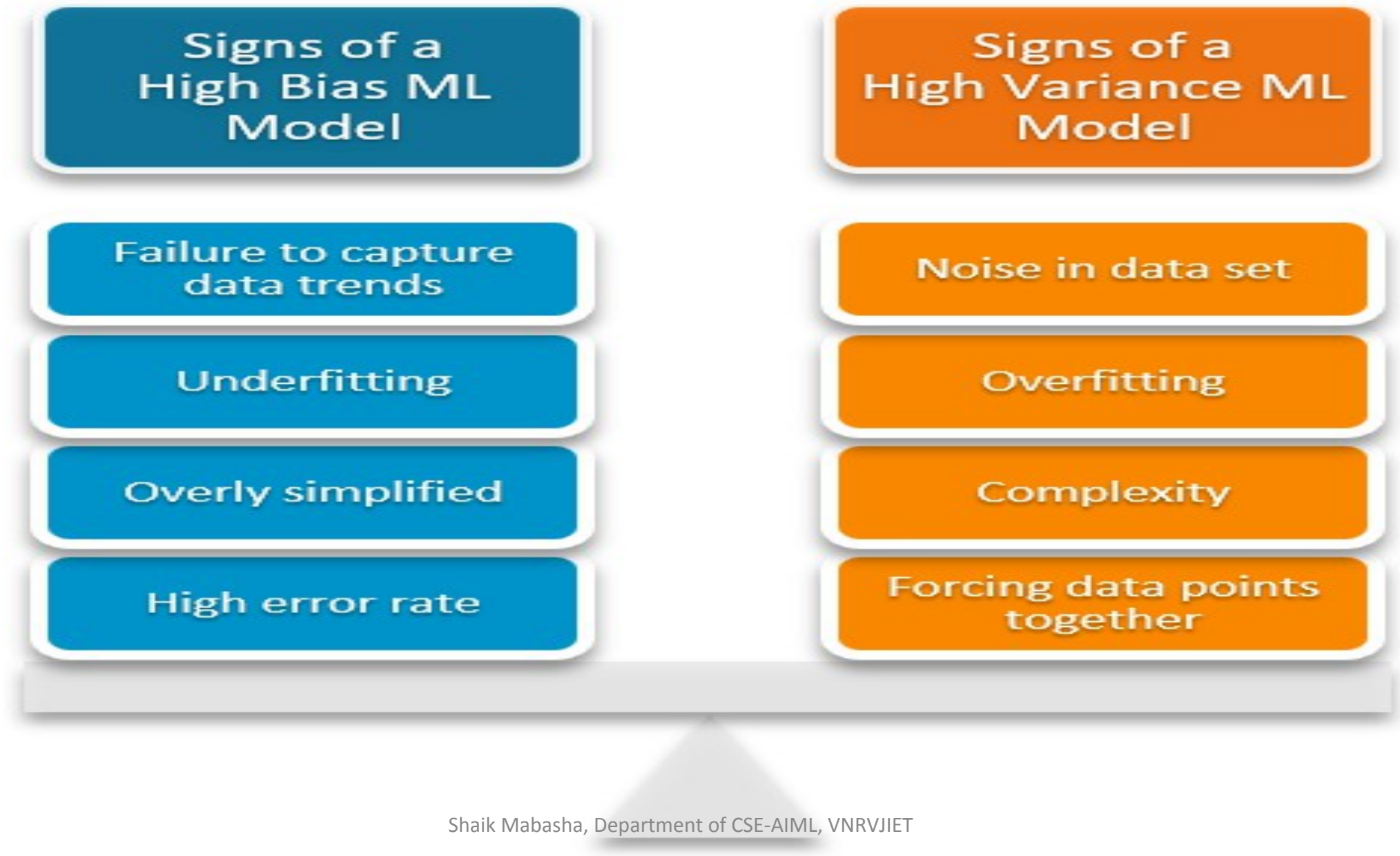
Low Bias & Low variance

Low Bias High variance

Reasons for Over-fitting:

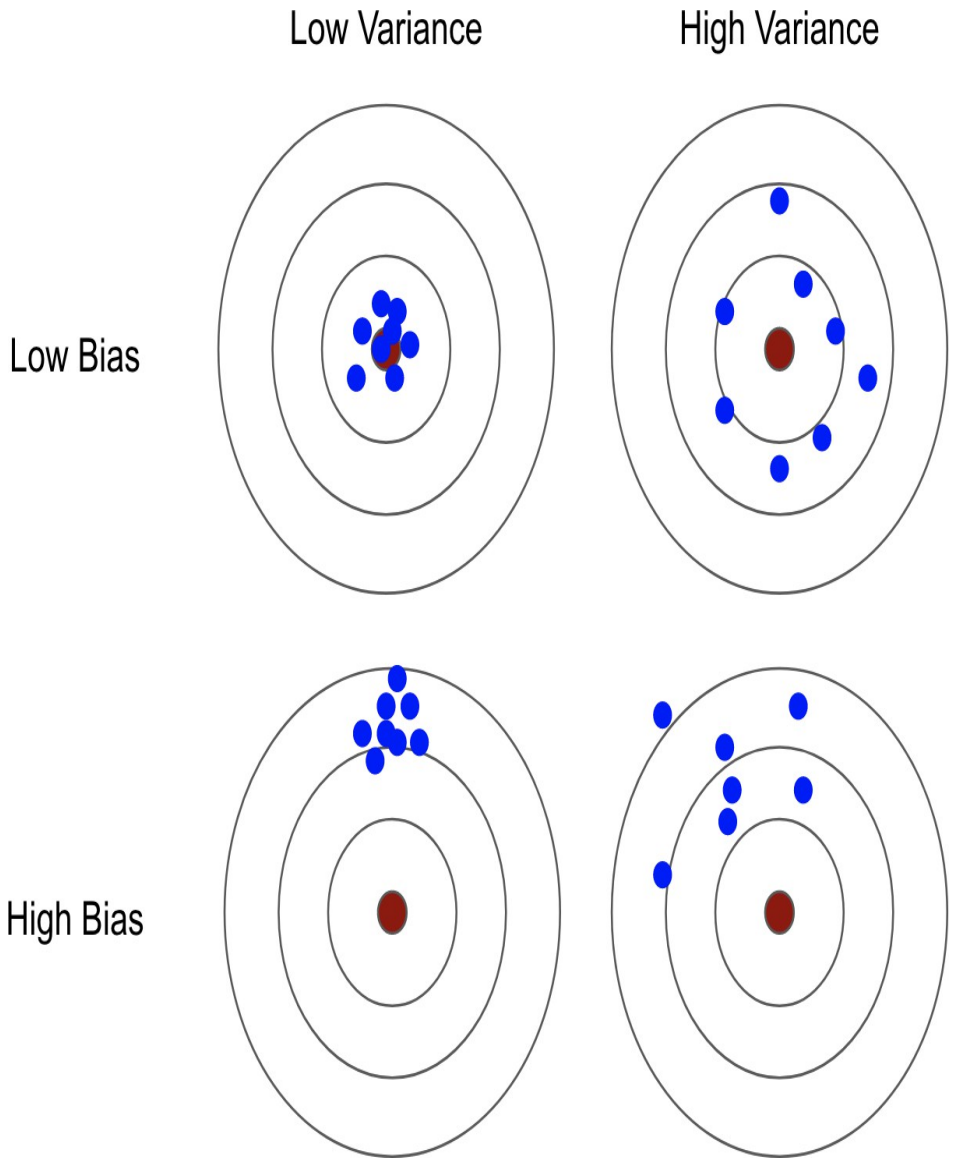
- The model is too complex.
- Too many Parameters and Rules
- The fluctuations or changes in the training data
- High variance and low bias

High Bias vs High Variance



Compariso
n

- **Low-Bias, Low-Variance:**
  The combination of low bias and low variance shows an ideal machine learning model. Best fit model
- **Low-Bias, High-Variance:** With low bias and high variance, **model predictions are inconsistent and accurate on average**. This case occurs when the model learns with a large number of parameters and hence leads to an **overfitting**
- **High-Bias, Low-Variance:** With High bias and low variance, **predictions are consistent** but **inaccurate on average.** This case occurs when a model does not learn well with the training dataset or uses few numbers of the parameter. It leads to **underfitting** problems in the model.
- **High-Bias, High-Variance:**
  With high bias and high variance, **predictions** are



CONFUSION MATRIX

CONFUSION MATRIX

- **Metric** is a technique to evaluate the performance of the model.
- List of various metric we will be covering in this blog.

1. *Accuracy*
2. *Null Accuracy*
3. *Precision*
4. *Recall*
5. *f1 score*
6. *ROC — AUC*
7. ***Accuracy*** : *This is the most naive and commonly used metric in the context of classification. It is just the mean of correct predictions.*

CONFUSION MATRIX

- A Confusion matrix or error matrix is used for summarizing the performance of a classification algorithm
- confusion matrix is used to find the accuracy of the classification model or a classifier.
- It will be applied on classification problems rather than regression.
- The given data set is divided into two ways
    1. Training data (80%) ---- Model can be build on training data
    2. Test Data (20%) ------ Accuracy of a model will be calculated using test data.
- To find the accuracy of a given model confusion matrix will be used.
- Calculating a confusion matrix can give you an idea of where the classification model is right and what types of errors it is making.
- A confusion matrix is used to check the performance of a classification model on a set of test data for which the true values are known.
- Performance measures such as precision, recall are calculated from the confusion matrix.

- Confusion matrix is always a square matrix.
- For example

S.NO	PATEINT NAME	AGE	ACTUAL VALUE (DIABITIES)	PREDICTED VALUE
1	A	45	YES	YES
2	B	50	NO	YES
3	C	57	YES	NO
4	D	68	YES	YES
5	E	30	NO	NO
6	F	76	NO	YES

- If the number of categories of target variable is 2 then we get confusion matrix order is 2 i.e 2 x2 .
- In general, if the order confusion matrix is n x n where n is the number of categories in target variable.

CONFUSION MATRIX

- A confusion matrix provides a summary of the predictive results in a classification problem.
- Correct and incorrect predictions are summarized in a table with their values and broken down by each class.

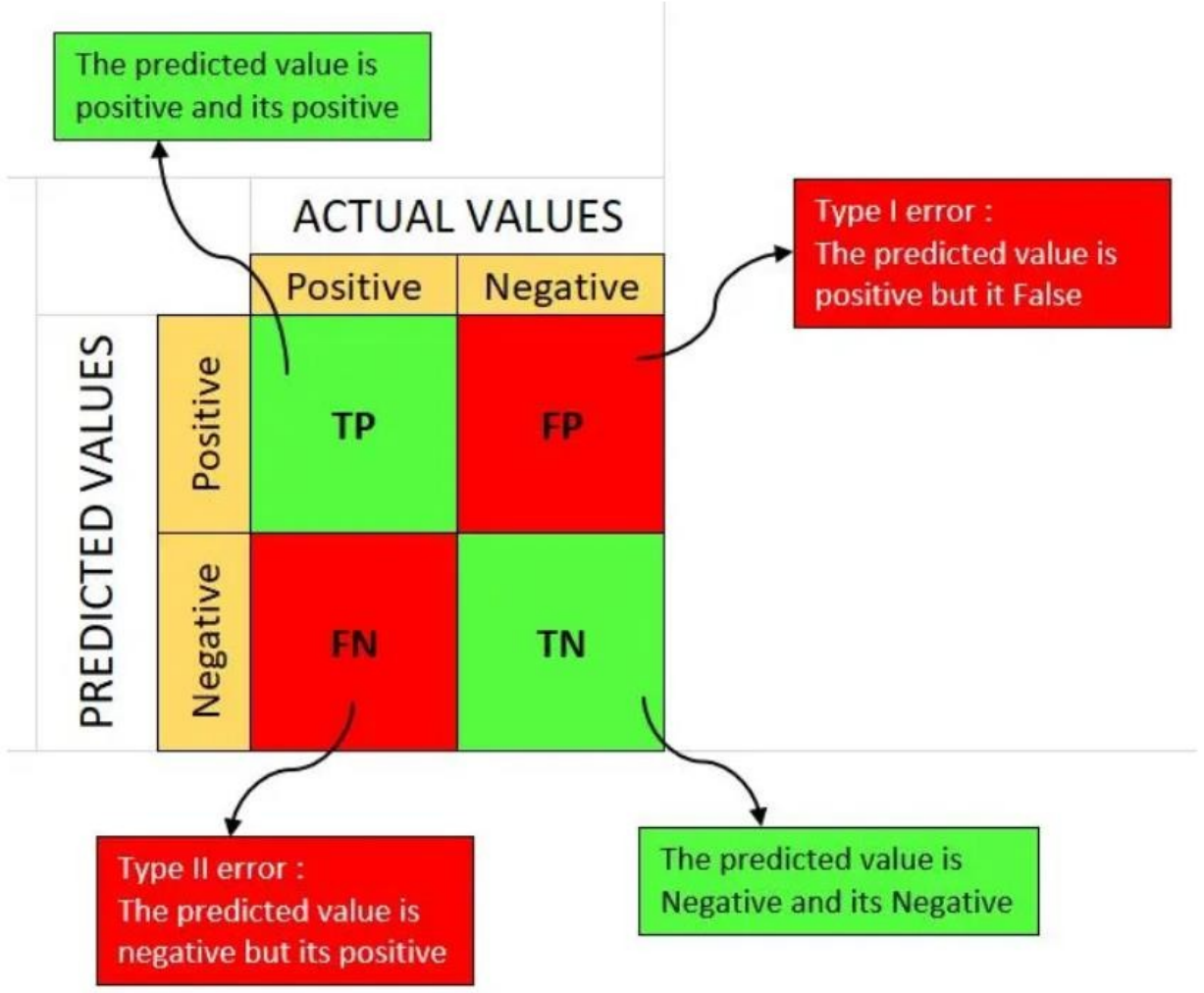


Fig: Confusion Matrix for the Binary Classification

*Remember, we describe **predicted values** as **Positive/Negative** and **Actual values** as **True/False**.*

CONFUSION MATRIX

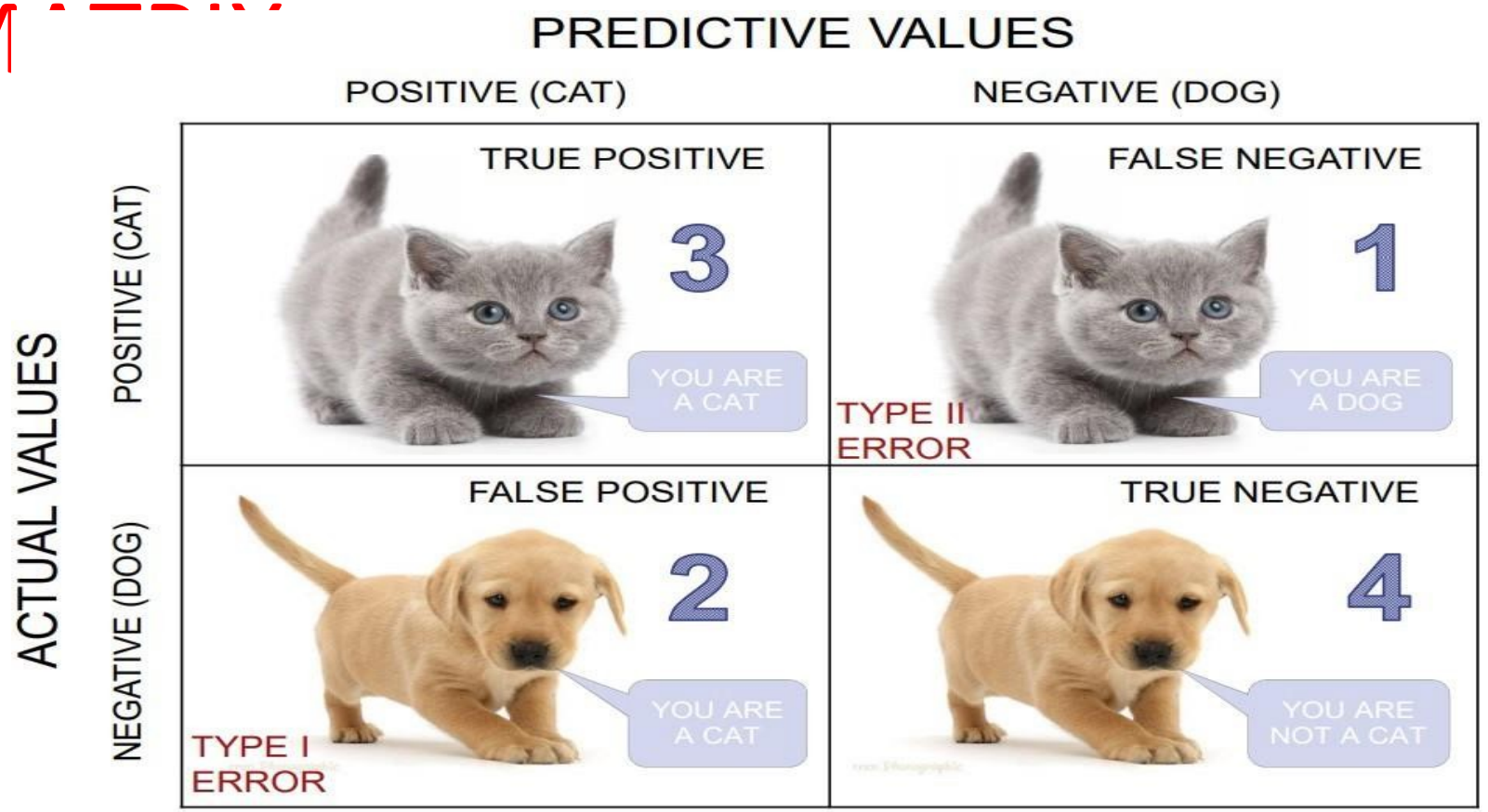
- We can not rely on a single value of accuracy in classification when the classes are imbalanced.
- For example, we have a dataset of 100 patients in which 5 have diabetes and 95 are healthy.
- However, if our model only predicts the majority class i.e. all 100 people are healthy even though we have a classification accuracy of 95%. Therefore, we need a confusion matrix.

- **Calculate A Confusion Matrix for 2x2 :**

- We have a total of 10 cats and dogs and our model predicts whether it is a cat or not.

- Actual values = [‘dog’, ‘cat’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘cat’, ‘dog’]
  Predicted values = [‘dog’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘cat’, ‘cat’, ‘cat’]

CONFUSION MATRIX



*Remember, we describe **predicted values** as **Positive/Negative** and*

***Actual values** **as** **True/False**.*

CONFUSION MATRIX

- **Definition of the Terms:**

- True Positive: You predicted positive and it's true. You predicted that an animal is a cat and it actually is.
- True Negative: You predicted negative and it's true. You predicted that animal is not a cat and it actually is not (it's a dog).
- False Positive (Type 1 Error): You predicted positive and it's false. You predicted that animal is a cat but it actually is not (it's a dog).
- False Negative (Type 2 Error): You predicted negative and it's false. You predicted that animal is not a cat but it actually is.
- A confusion matrix is used to check the performance of a classification model on a set of test data for which the true values are known.
- Most performance measures such as **precision, recall** are calculated from the confusion matrix.

CONFUSION MATRIX

Recall (aka Sensitivity)

- **Recall** is defined as the ratio of the total number of **correctly classified positive classes** divide by the **total number of positive classes**. Or, out of all the positive classes, how much we have predicted correctly. **Recall should be high.**

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}}$$

- **Precision** is defined as the ratio of the total number of **correctly classified positive classes** divided by **the total number of predicted positive classes**. Or, out of all the predictive positive classes, how much we predicted correctly. **Precision should be high.**

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}}$$

CONFUSION MATRIX

Classification Accuracy:

- Classification Accuracy is given by the relation·

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

F-score or F1-score:

- It is **difficult to compare two models** with **different Precision and Recall**. So to make them comparable, we use F-Score. **It is the Harmonic Mean of Precision and Recall**. As compared to Arithmetic Mean, Harmonic Mean punishes the extreme values more. F-score should be high.

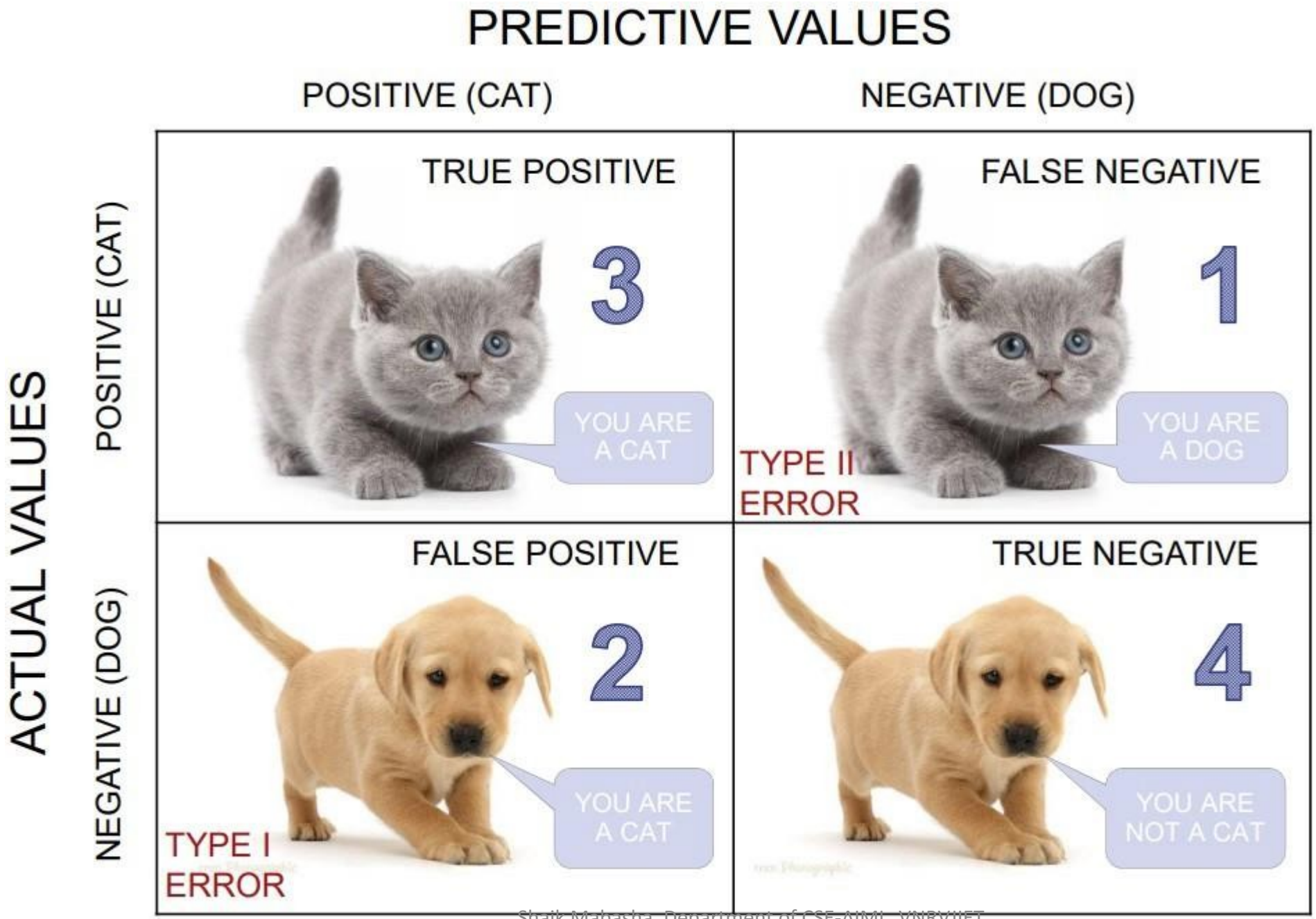
$$F\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

CONFUSION MATRIX

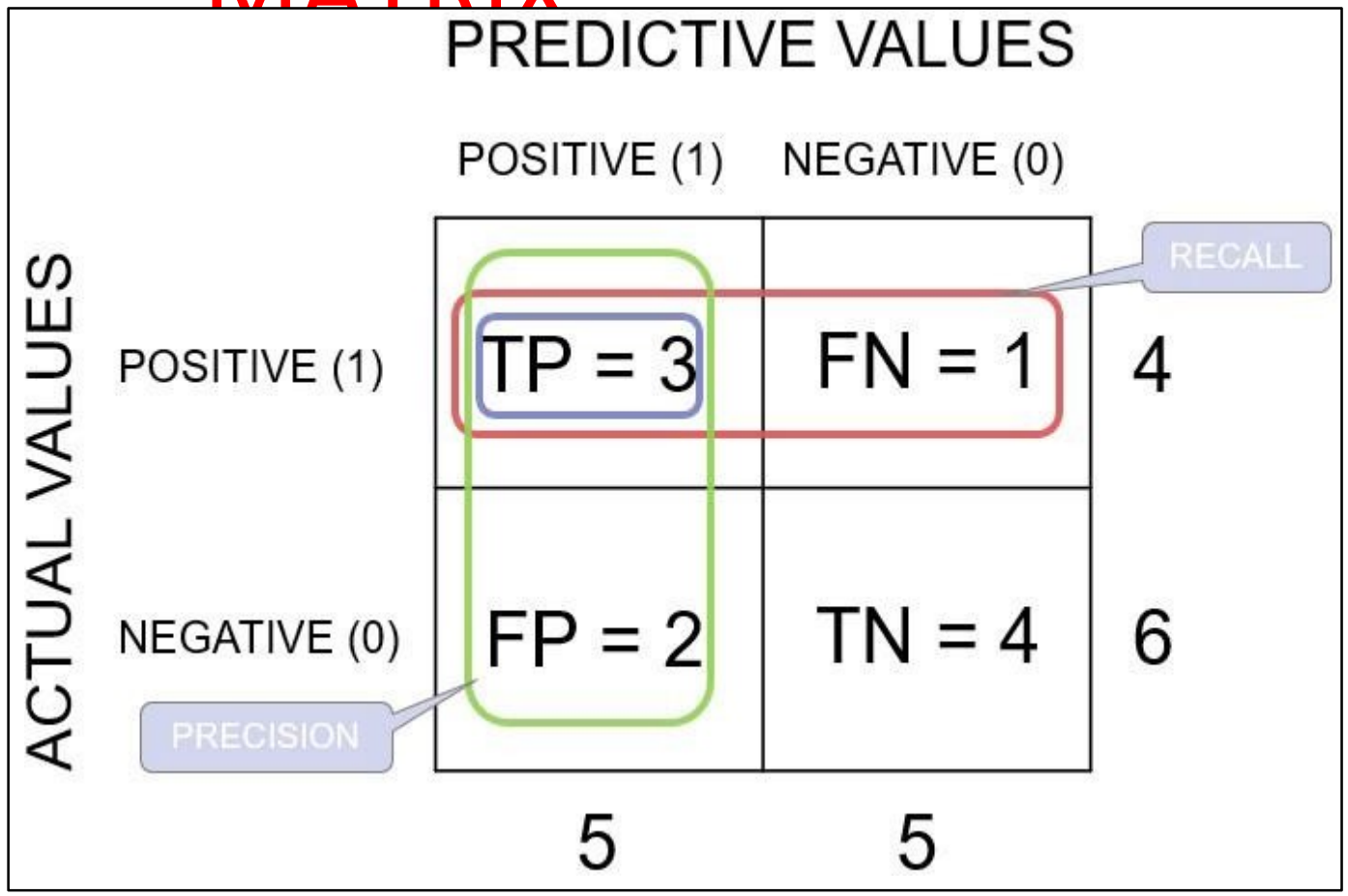
- **Specificity:**
  Specificity determines the proportion of actual negatives that are correctly identified.

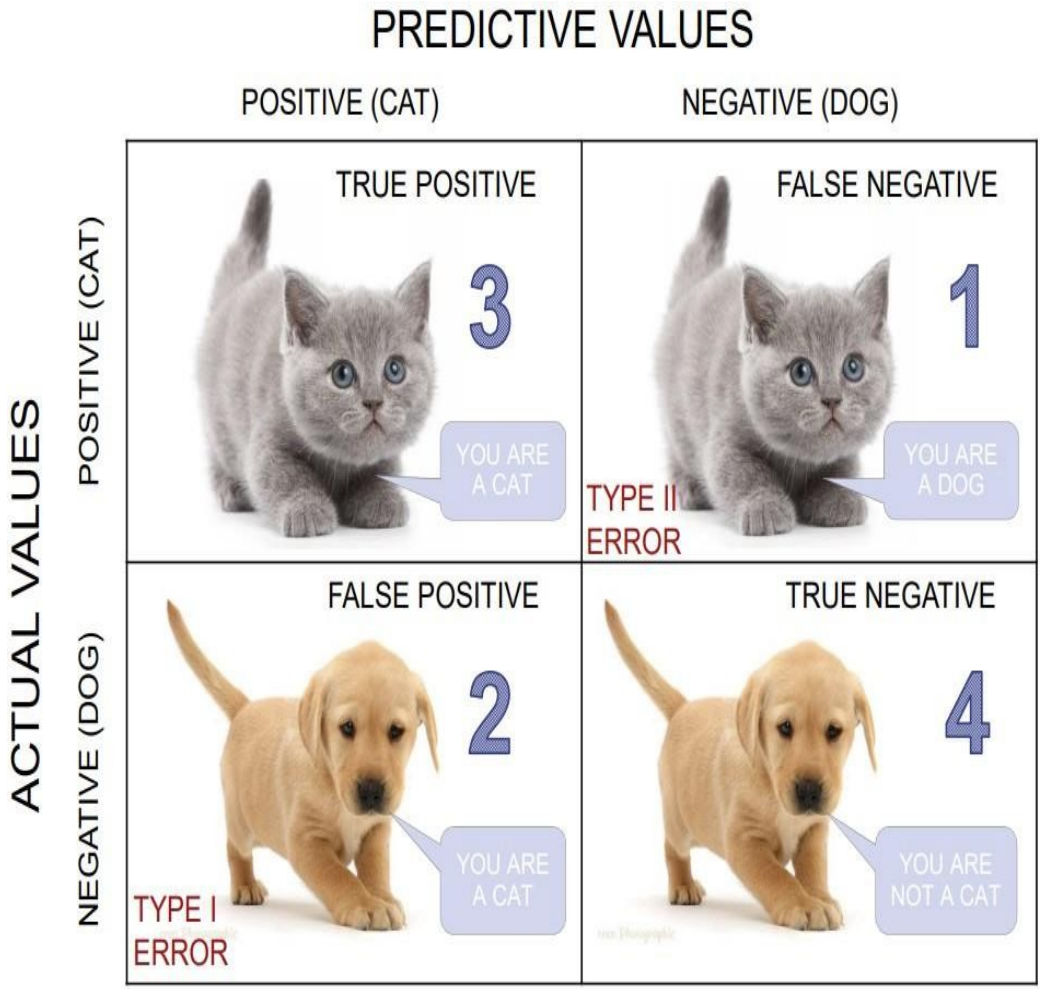
$$Specificity = \frac{TN}{TN + FP}$$

- **Example to interpret confusion matrix:**
  Let's calculate confusion matrix using above cat and dog example:


PREDICTIVE VALUES
POSITIVE (CAT)
NEGATIVE (DOG)
ACTUAL VALUES
POSITIVE (CAT)
NEGATIVE (DOG)
TRUE POSITIVE
3
YOU ARE A CAT
FALSE NEGATIVE
1
YOU ARE A DOG
TYPE II ERROR
FALSE POSITIVE
2
YOU ARE A CAT
TYPE I ERROR
TRUE NEGATIVE
4
YOU ARE NOT A CAT

CONFUSION MATRIX





CONFUSION MATRIX

- **Classification Accuracy:**
  Accuracy = (TP + TN) / (TP + TN + FP + FN) = (3+4)/(3+4+2+1) = 0.70

- **Recall:** Recall gives us an idea about when it's actually yes, how often does it predict yes.
  Recall = TP / (TP + FN) = 3/(3+1) = 0.75

- **Precision:** Precision tells us about when it predicts yes, how often is it correct.
  Precision = TP / (TP + FP) = 3/(3+2) = 0.60

- **F-score:**
  F-score = (2*Recall*Precision)/(Recall + Precision) = (2*0.75*0.60)/(0.75+0.60) = 0.67

- **Specificity:**

CONFUSION MATRIX

Summary:

- **Precision** is how certain you are of **your true positives**. **Recall** is how certain you are that **you are not missing any positives.**
- Choose **Recall** if the occurrence of **false negatives is unaccepted/intolerable.** For example, in the case of diabetes that you would rather have some extra false positives (false alarms) over saving some false negatives.
- Choose **Precision** if you want to be more **confident of your true positives**. For example, in case of spam emails, you would rather have some spam emails in your inbox rather than some regular emails in your spam box. You would like to be extra sure that email X is spam before we put it in the spam box.
- Choose **Specificity** if you want to **cover all true negatives,** i.e. meaning we do not want any false alarms or false positives. For example, in case of a drug test in which all people who test positive will immediately go to jail, you would not want anyone drug-free going to jail.

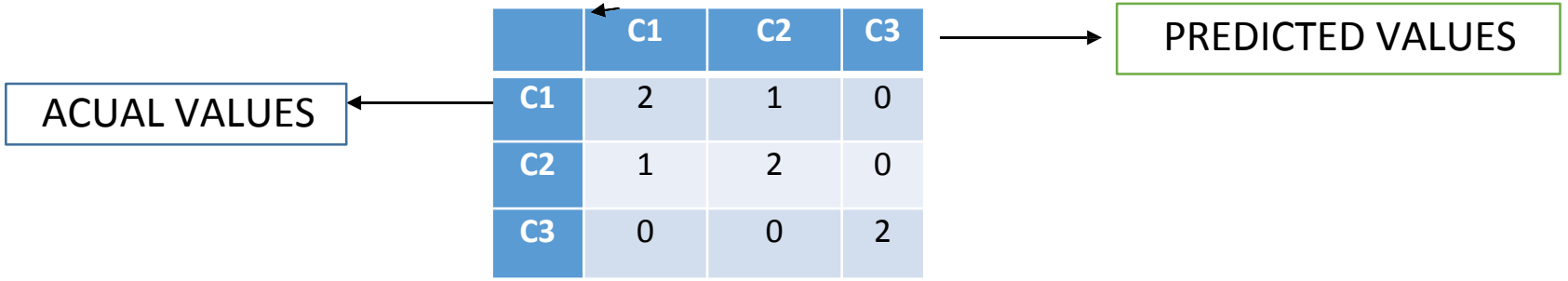
CONFUSION MATRIX

We can conclude that:

- Accuracy value of 70% means that identification of 3 of every 10 cats is incorrect, and 7 is correct.
- Precision value of 60% means that label of 4 of every 10 cats is a not a cat (i.e. a dog), and 6 are cats.
- Recall value is 70% means that 3 of every 10 cats, in reality, are missed by our model and 7 are correctly identified as cats.
- Specificity value is 60% means that 4 of every 10 dogs (i.e. not cat) in reality are miss-labeled as cats and 6 are correctly labeled as dogs.

CONFUSION MATRIX FOR 3 X 3

- Consider the following clusters:



	C1	C2	C3
C1	2	1	0
C2	1	2	0
C3	0	0	2

- 1. **TP = True positives** = Actual C1 is predicted as C1+

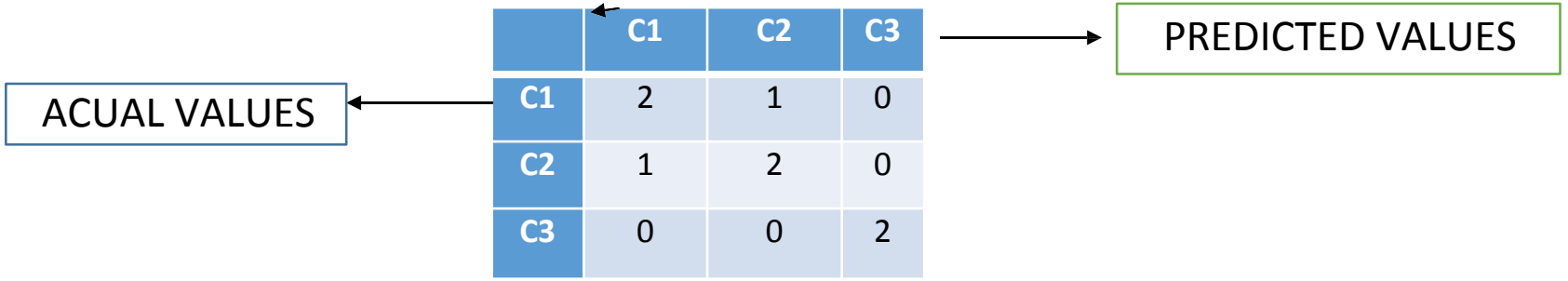
Actual C2 is predicted as C2+

Actual C3 is predicted as C3

= 2+2+2 = 6

CONFUSION MATRIX FOR 3 X 3

- Consider the following clusters:



	C1	C2	C3
C1	2	1	0
C2	1	2	0
C3	0	0	2

- **2. TN = True negatives** = actual non C1 is predicted as non C1 +

actual non C2 is predicted as non C2 +

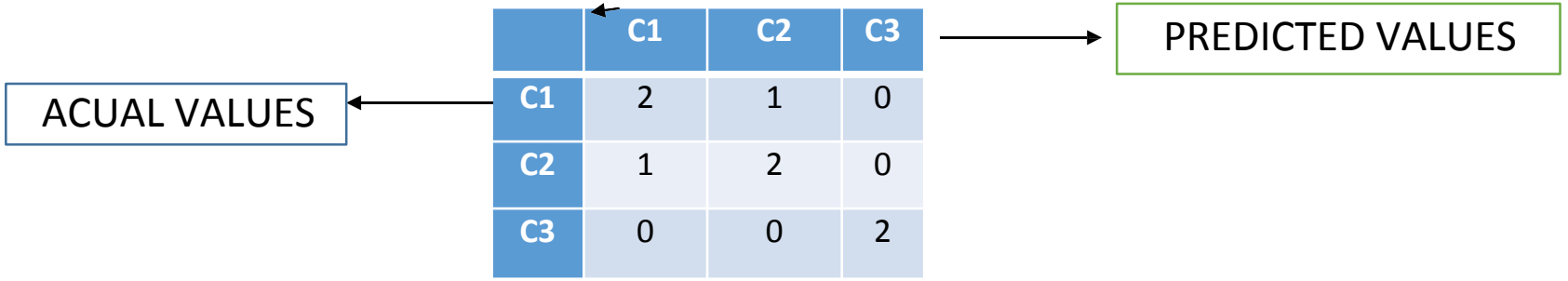
actual non C3 is predicted as non C3

= 4+4+6

= 14

CONFUSION MATRIX FOR 3 X 3

- Consider the following clusters:



- **3. FP = False positives** = Actual non C1 is predicted as C1 +

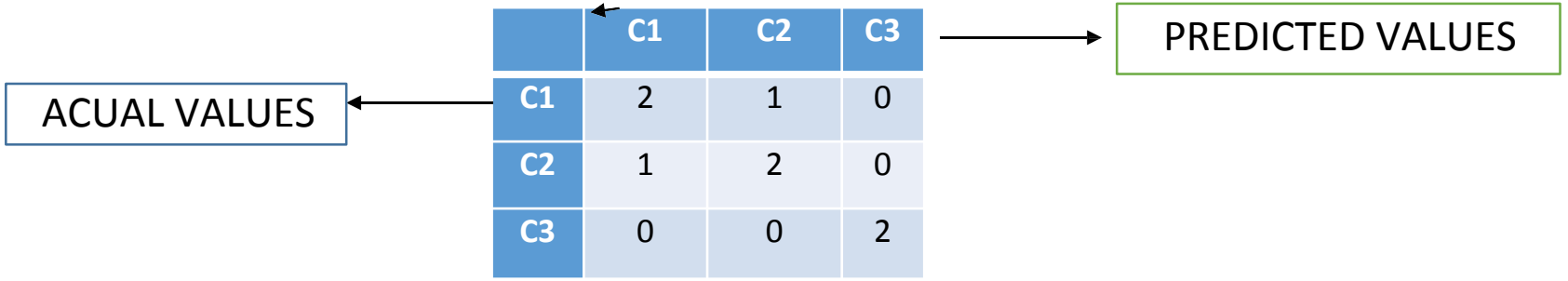
  Actual non C2 is predicted as C2 +

  Actual non C3 is predicted as C3 +

  = 1+1+0 = 2

CONFUSION MATRIX FOR 3 X 3

- Consider the following clusters:



	C1	C2	C3
C1	2	1	0
C2	1	2	0
C3	0	0	2

- 4. FN = False Negative = Actual C1 is predicted as non C1 + Actual C2 is predicted as non C2 + Actual C3 is predicted as non C3 +

$$=1+1+0=2$$

CONFUSION MATRIX

We can conclude that:

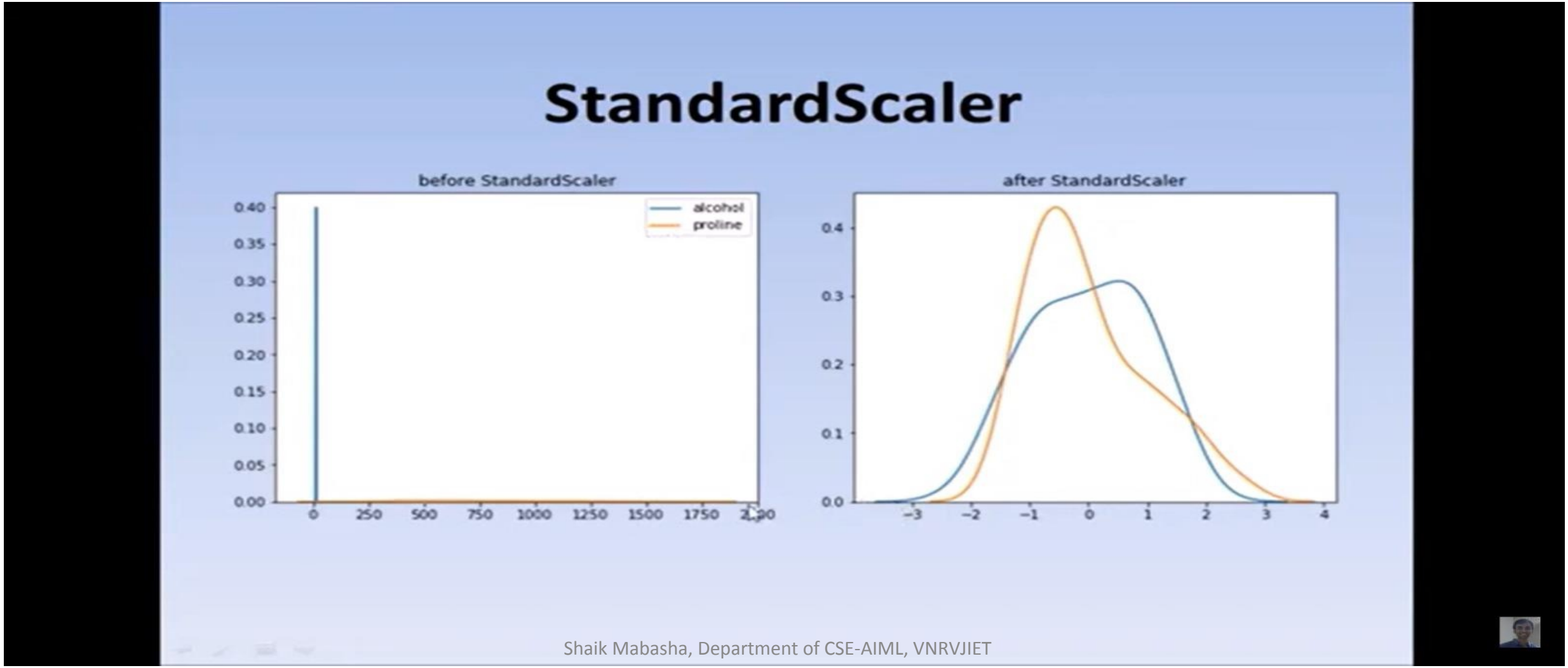
- **Accuracy** = (TP+TN) / (TP+FP+FN+TN) = (6+14) / (6+14+2+2) = 0.833
- Precision = TP/ (TP+FP) = 6/ (6+2) = 0.75.
- Recall = TP / (TP+FN) = 6/ (6+2) = 0.75
- **F-measure** = (2*P*R)/(P+R) =(2* 0.75 * 0.75)/(0.75 + 0.75) = 0.75.
- **Purity:** Sum of correctly assigned objects and dividing by N(total number of objects).

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

where $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.

StandardScaler

One of the most commonly used feature scaling techniques is StandardScaler. It scales the data such that the **mean is 0 and the standard deviation is 1.** This means that the scaled data will have a normal distribution with a mean of 0 and a standard deviation of 1



How to transform/scaling a new data feature using standard scaler

StandardScaler

- For each feature X, we calculate the mean (X_m) and standard deviation (X_s)
- For each value in that feature X (X_i), calculate:
- New $X_i = \frac{Xi - Xm}{Xs}$
- Xm = 18.71
- Xs = 13.46

original	scaled
5	-1.01857
10	-0.6471
12	-0.49851
14	-0.34993
18	-0.05275
23	0.318722
49	2.250371