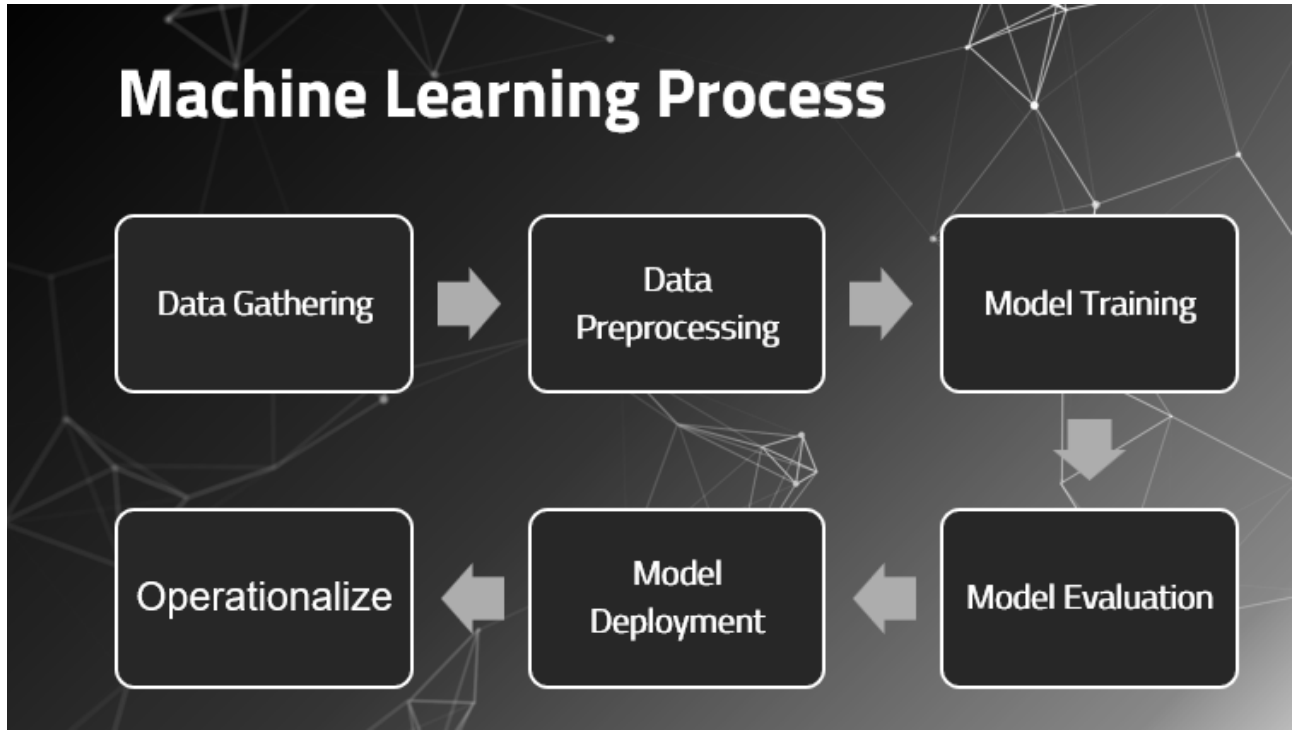
SUPERVISED MACHINE LEARNING (CLASSIFICATION)

Machine Learning – Yousef Elbaroudy

GUIDELINES

- Try to focus on the important information mentioned through the session
- Apply what you take on the practical section
- Do not try to memorize everything you got, just learn
- Don’t mind to ask about anything you want to know

Enjoy the Session ☺


Machine Learning Process
Data Gathering
Data Preprocessing
Model Training
Model Evaluation
Model Deployment
Operationalize

REWIND

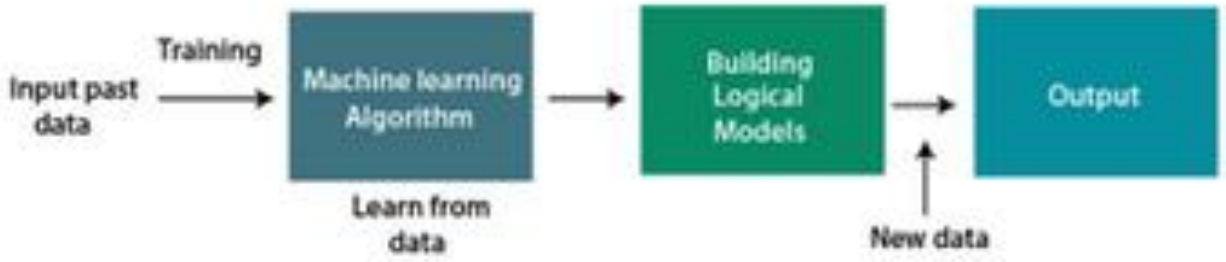
WHAT IS MACHINE LEARNING ?

- **Machine Learning** is a branch of artificial intelligence that is concerned with the development of algorithms which allow a computer to learn automatically from **the past data and past experiences**.
- Machine learning builds **mathematical or statistical models** to make predictions using **historical data or information**.

Historical data: Known as *Training data*

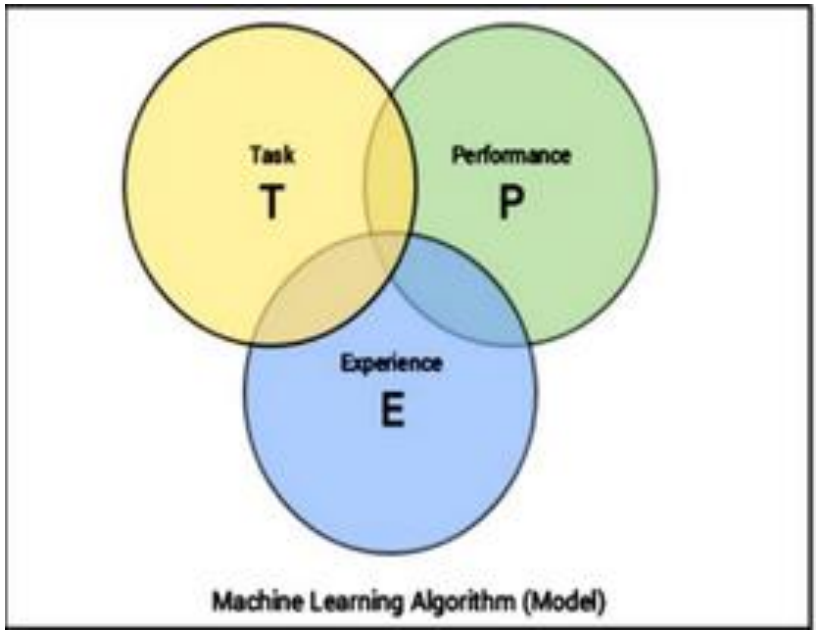
ALWAYS IN MACHINE LEARNING

- The accuracy of predictions depends on the **amount** of data: **More** data → **better** predictions.



TOM MITCHELL'S DEFINITION FOR MACHINE LEARNING

"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."



DEFINING TASK T

Defining task T

Classification: classify animal images into dogs and cats.
Regression: predicting house price.
Anomaly detection: indication of fraud.
Structured annotation: text annotation like grammar, sentiment, named entities, image annotation like annotate specific areas of images.
Translation: natural language translate.
Clustering or grouping: group similar products, events, entities
Transcription: These tasks usually entail various representations of data that are usually continuous and unstructured and converting them into more structured and discrete data elements. Examples include speech to text, optical character recognition, images to text,

DEFINING EXPERIENCE E

Defining experience E

The process of consuming a dataset that consists of data samples or data points such that a learning algorithm or model learns inherent patterns is defined as the experience, *E* which is gained by the learning algorithm.

DEFINING PERFORMANCE P

Defining performance P

performance measures include **accuracy, precision, recall, F1 score, sensitivity, specificity,**
error rate, misclassification rate.
Performance measures are usually evaluated on training data samples
(used by the algorithm to gain experience, E) as well as data samples which it has not seen or learned from before, which are usually known as validation and test data samples.
The idea behind this is to ***generalize*** the algorithm so that it doesn't become too biased only on the training data points and performs well in the future on newer data points.

HOW MACHINE LEARNS ?

Data storage utilizes observation, memory, and recall to provide a factual basis for further reasoning.

Abstraction involves the translation of stored data into broader representations and concepts.

Generalization uses abstracted data to create knowledge and inferences that drive action in new contexts.

Evaluation provides a feedback mechanism to measure the utility of learned knowledge and inform potential improvements.

ABSTRACTION

- This work **Abstraction** of assigning meaning to stored data occurs during the abstraction process, in which raw data comes to have a more abstract meaning.
- This type of connection, say between an object and its representation
- During a machine's process of knowledge representation, the computer summarizes stored raw data using a model, an explicit description of the patterns within the data.

DIFFERENT TYPES OF MODELS

Mathematical equations

Statistical Models

Relational diagrams such as trees and graphs

Logical if/else rules

Groupings of data known as clusters

ABSTRACTION (CONT.)

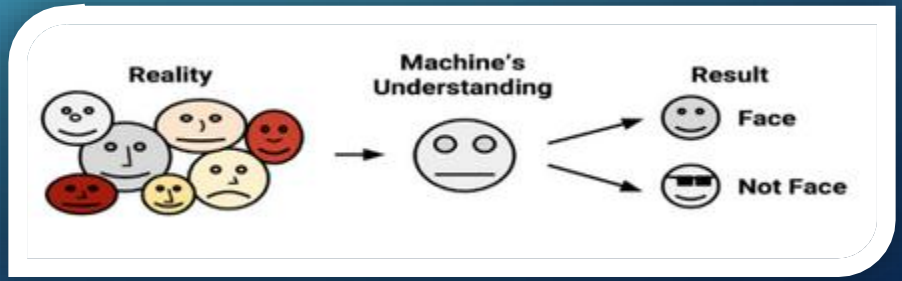
- The process of fitting a model to a dataset is known as **training**. When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

Note ! You might wonder why this step is called training rather than learning.

- **First**, note that the process of learning does not end with data abstraction; the learner must still generalize and evaluate its training.
- **Second**, the word training better connotes the fact that the human teacher trains the machine student to understand the data in a specific way.

GENERALIZATION

- The term **generalization** describes the process of turning abstracted knowledge into a form that can be utilized for future action, on tasks that are similar, but not identical, to those it has seen before.
- In **generalization**, the learner is tasked with limiting the patterns it discovers to only those that will be most relevant to its future tasks.
- The algorithm is said to have a bias if the conclusions are systematically erroneous, or wrong in a predictable manner.



EVALUATION

- Bias is a necessary evil associated with the abstraction and generalization processes inherent in any learning task. In order to drive action in the face of limitless possibility, each learner must be biased in a particular way.

- Therefore, the final step in the generalization process is to **evaluate or measure** the learner's success in spite of its biases and use this information to inform additional training if needed.

- Generally, evaluation occurs after a model has been trained on an initial training dataset. Then, the model is **evaluated on a new test dataset** in order to judge how well its characterization of the training data generalizes to new, **unseen data**.

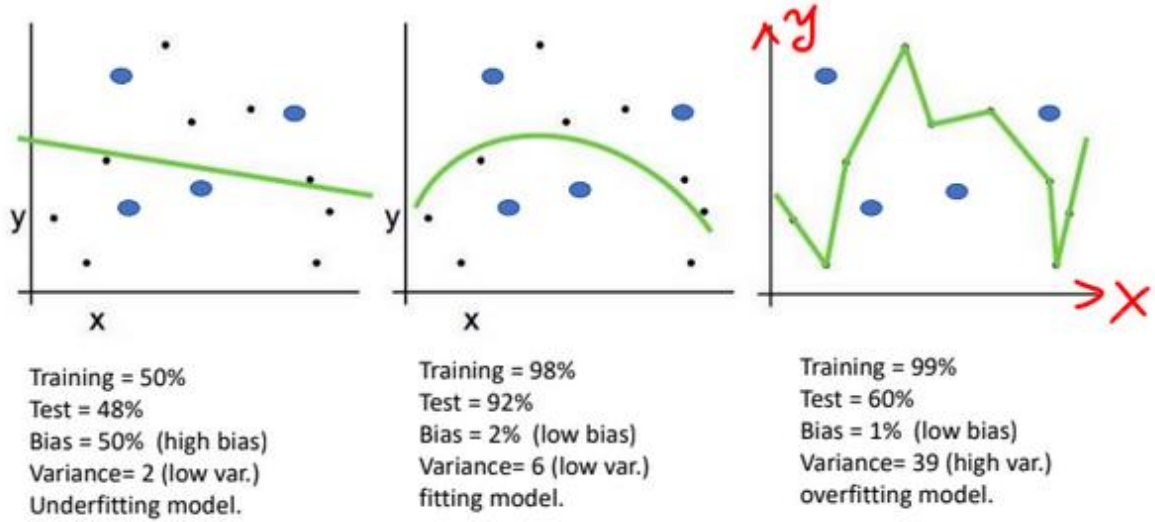
EVALUATION (CONT.)

- **In parts, models fail to perfectly generalize due to the problem of noise, Noisy data is caused by seemingly random events, such as:**

- Measurement error due to imprecise sensors that sometimes add or subtract a bit from the readings.
- Issues with human subjects, such as survey respondents reporting random answers to survey questions, in order to finish more quickly.
- Data quality problems, including missing, null, truncated, incorrectly coded, or corrupted values.
- Phenomena that are so complex or so little understood that they impact the data in ways that appear to be unsystematic.

Trying to model noise is the basis of a problem called OVERFITTING !

UNDERFITTING VS. OVERFITTING VS. GOOD FITTING



TRAINING DATASET

- The **training set** is the portion of the dataset that is used to train the machine learning model. It contains labeled examples (input data and corresponding target labels) that the model uses to learn the patterns and relationships in the data.

- The goal is for the model to generalize from the training data and capture the underlying patterns so that it can make accurate predictions on new, unseen data.

VALIDATION DATASET

- The **validation set** is used during the training process to fine-tune the model's hyperparameters and monitor its performance. It is separate from the training set and contains examples that the model has not seen during training.

- The validation set allows you to assess the model's performance on unseen data and make decisions about parameter adjustments, such as adjusting the learning rate or choosing the number of hidden units in a neural network

- By evaluating the model's performance on the validation set, you can make informed decisions to improve the model's generalization capabilities.

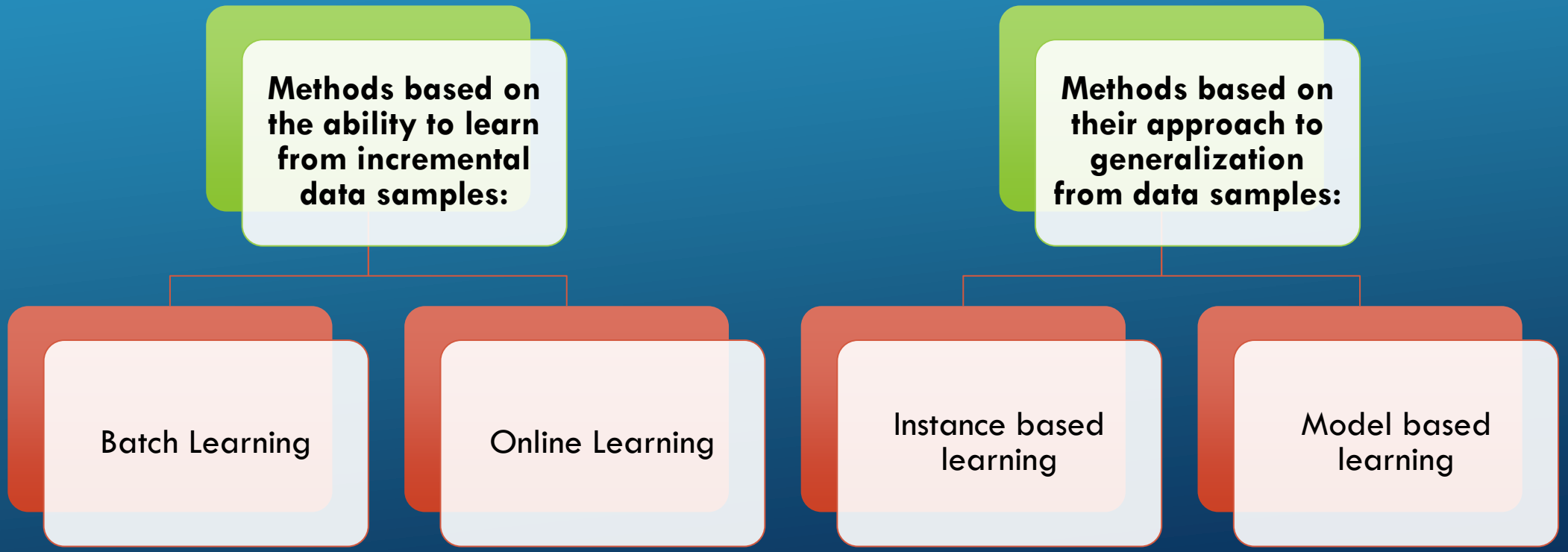
TEST DATASET

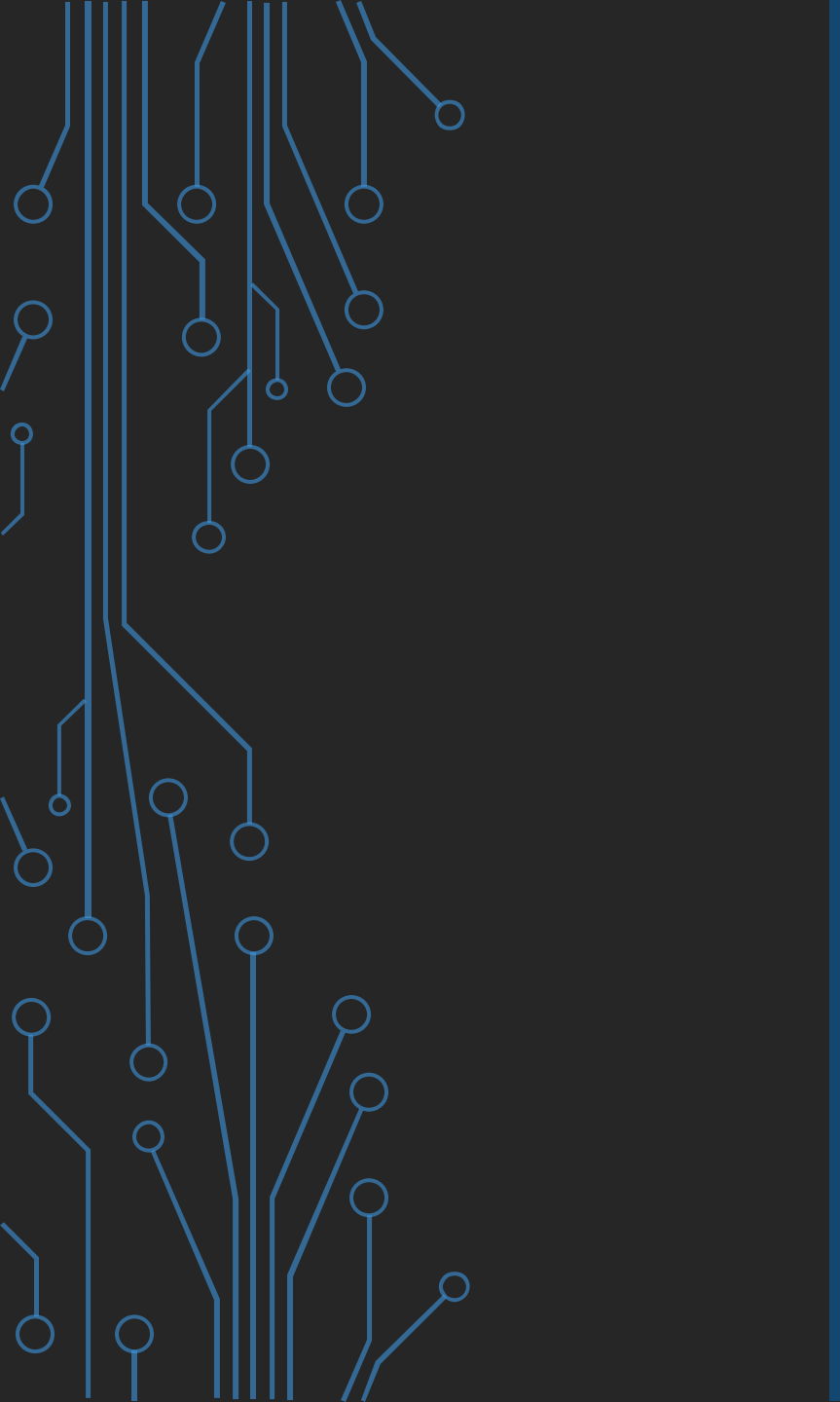
- The **test set** is used to evaluate the final performance of the trained machine learning model. It is a separate, independent dataset that the model has never seen before during training or validation.
- The purpose of the test set is to provide an unbiased assessment of the model's performance on completely new data.
- By evaluating the model on a test set, you can estimate how well the model is likely to perform in real-world scenarios.

TRAINING VS. VALIDATION VS. TEST

- Training Set: Used for training the model by adjusting its parameters based on labeled examples.
- Validation Set: Used to fine-tune hyperparameters and monitor performance during training.
- Test Set: Used to assess the final performance and generalization capabilities of the trained model.

ADDITIONAL MACHINE LEARNING METHODS





SUPERVISED LEARNING (CLASSIFICATION)

WHAT IS CLASSIFICATION ?

- **Classification** is a fundamental task in machine learning and data analysis. It involves categorizing data instances into predefined classes or categories based on their features or attributes.

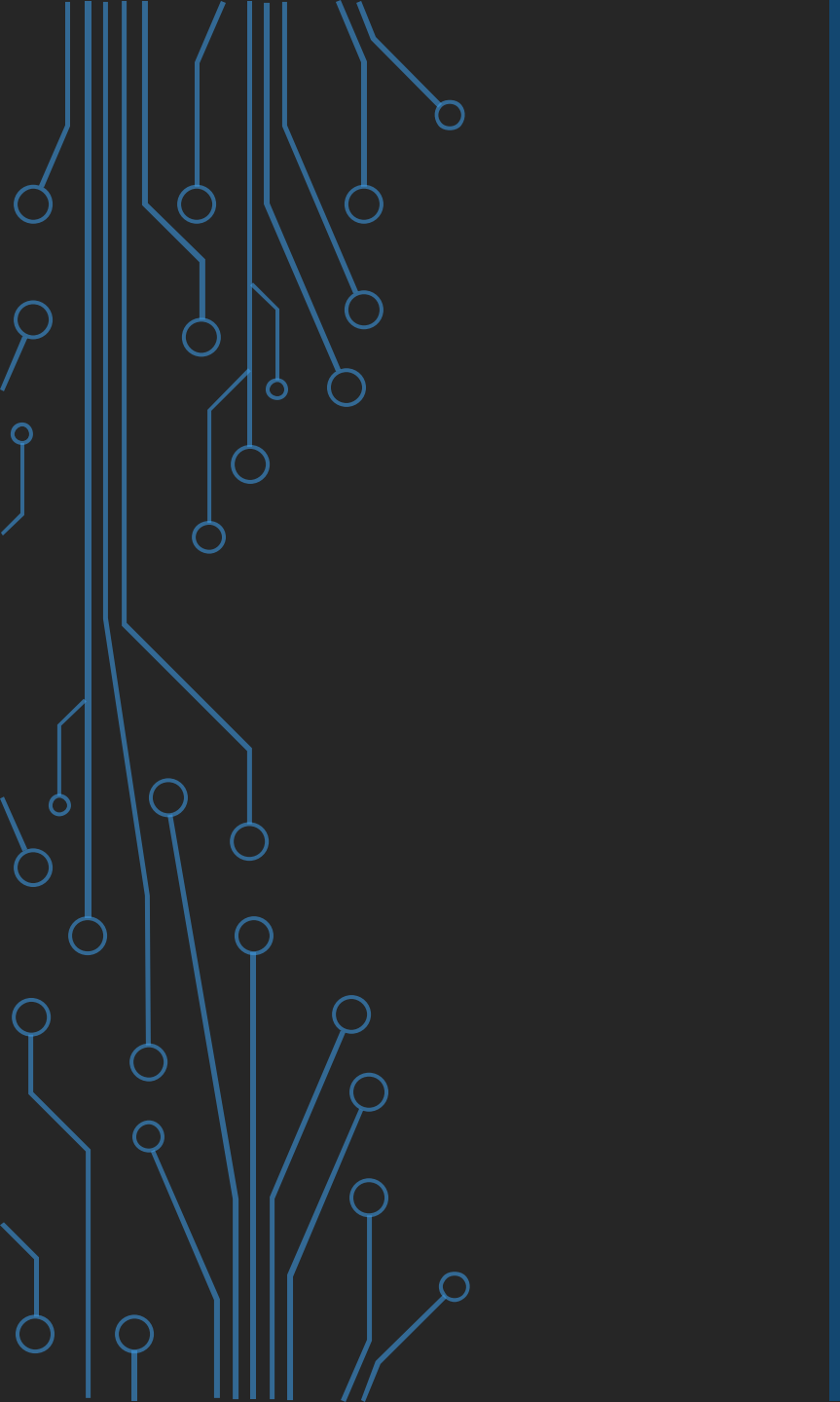
- The goal of classification is to build a predictive model that can assign the correct class label to new, unseen data based on patterns and relationships learned from a labeled dataset.

In a classification problem, you have a dataset containing examples with their corresponding class labels.

TYPES OF CLASSIFICATION

Binary Classification

Multi Classification



NEAREST NEIGHBOR

NEAREST NEIGHBOR

- **Neighbors-based classification** is a type of ***instance-based learning*** or ***non-generalizing learning***: it does not attempt to construct a general internal model, but simply stores instances of the training data.

- Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

TYPES OF NEAREST NEIGHBOR

K-Nearest Neighbor

Radius Neighbor

WHAT IS K-NEAREST NEIGHBORS (KNN) ?

- **K-Nearest Neighbors (KNN)** is a simple yet powerful machine learning algorithm used for both classification and regression tasks.

- It is a non-parametric and instance-based learning algorithm, meaning it makes predictions based on the similarity between input data points and the data points in its training set.

Sure! Imagine you have a pet dog, and you want to figure out if your dog is more like a "friendly" dog or a "shy" dog. You have a bunch of pictures of other dogs that are labeled as either "friendly" or "shy." You want to use these pictures to decide if your dog is more like the "friendly" dogs or the "shy" dogs.

STORYTELLING OF GPT

K-Nearest Neighbors, or KNN for short, is a bit like asking your friends who have dogs for their opinions. Here's how it works:

1. You show your friends the picture of your dog, and they compare it to pictures of their own dogs. They look at the dogs that are most similar to yours.
2. Let's say you ask three friends (k = 3), and two of them say their dogs are "friendly" while one says their dog is "shy."
3. Since more of your friends have "friendly" dogs, you might guess that your dog is more likely to be "friendly" as well.

STORYTELLING OF GPT (CONT.)

In KNN, the "k" represents the number of friends you ask. The algorithm looks at the features of the dogs (like size, color, and fur type) and compares them to decide if your dog is more like the "friendly" dogs or the "shy" dogs. The decision is based on what the majority of your friends' dogs are like.

So, KNN helps you make a guess about whether your dog is more like "friendly" dogs or "shy" dogs by asking its closest neighbors (other dogs) for their opinions. It's like asking your friends for advice about your dog's behavior based on what their dogs are like.

STORYTELLING OF GPT (CONT.)

HOW IT WORKS ?

Training Phase:

- The algorithm is provided with a labeled training dataset, where each data point consists of features (attributes) and a corresponding class label (for classification) or a target value (for regression).
- KNN does not explicitly "train" a model like many other algorithms. Instead, it memorizes the entire training dataset.

HOW IT WORKS ? (CONT.)

Prediction Phase:

- When a new, unlabeled data point needs to be classified or predicted, KNN identifies the "k" closest data points (neighbors) from the training dataset based on a **distance metric**, often using Euclidean distance.
- The value of "k" is a user-defined parameter that determines how many neighbors influence the prediction. For example, if k = 3, the algorithm considers the labels or values of the three closest neighbors.

It has no loss function !

HOW IT WORKS ? (CONT.)

Classification:

- For classification tasks, the algorithm assigns the class label that is most common among the k nearest neighbors. This is often done using majority voting. For instance, if out of the k neighbors, 2 belong to class A and 1 belongs to class B, the algorithm would predict class A for the new data point.

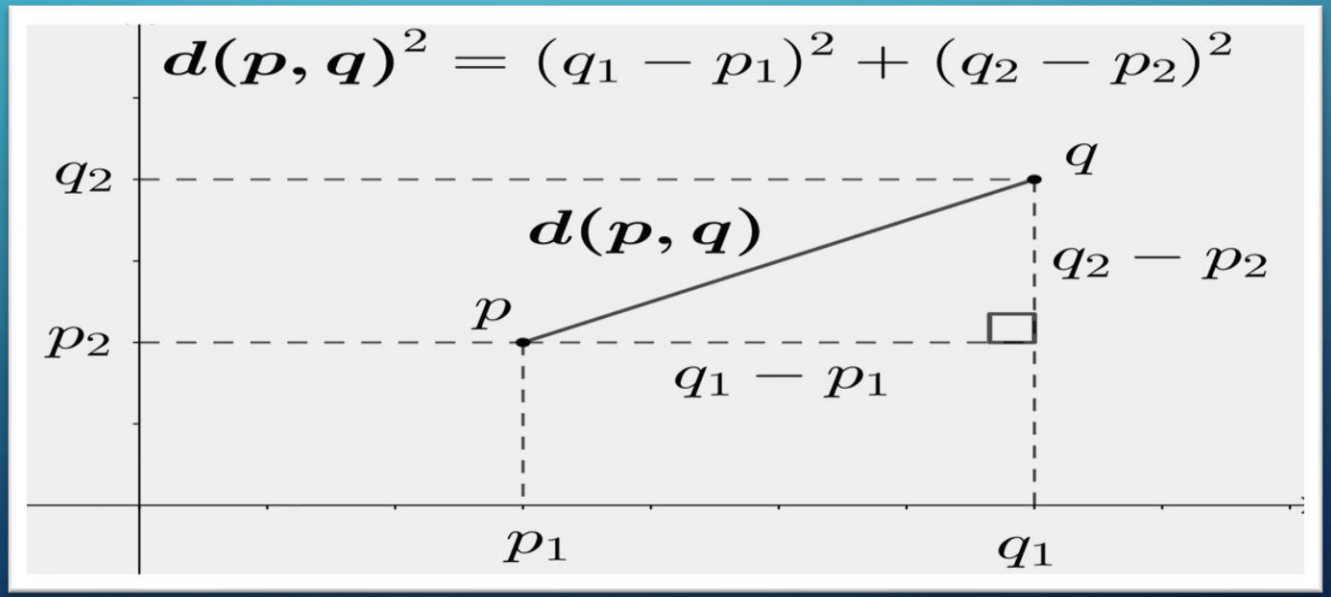
In other words, it is the using of MODE

SIMILARITY METRICS FOR KNN

Euclidean Distance	**Manhattan Distance**	**Chebyshev Distance**	**Minkowski Distance**
Hamming Distance	**L1 Norm**	**L2 Norm**	**Jaccard Similarity**

EUCLIDEAN DISTANCE (L2 NORM)

- This is the most common distance metric and is used for continuous numerical data.
- It calculates the straight-line distance between two points in the feature space. If you have two points with coordinates (x1, y1) and (x2, y2), the Euclidean distance is:



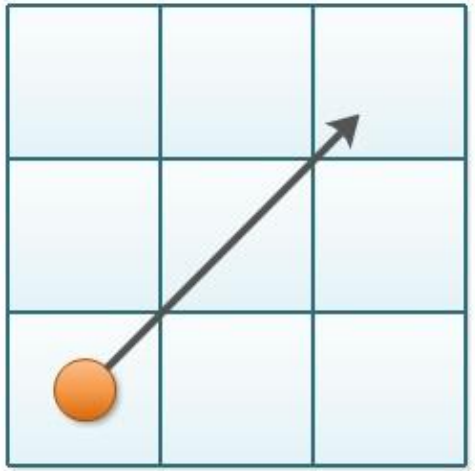
MANHATTAN DISTANCE (L1 NORM)

- Also known as the "city block" or "taxicab" distance, this metric calculates the distance by summing the absolute differences between corresponding coordinates.
- It's often used when movement can only occur along grid lines (like streets in a city).

CHEBYSHEV DISTANCE

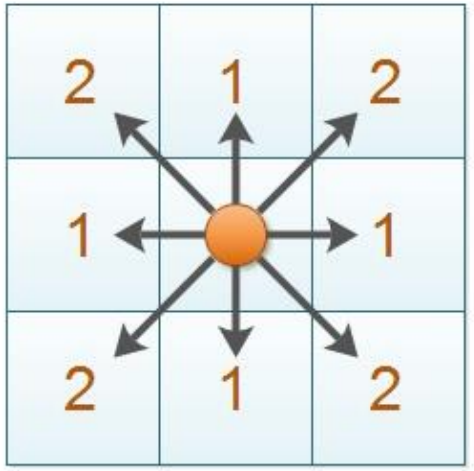
- This metric calculates the maximum absolute difference between coordinates.
- It's useful when you want to emphasize the largest difference between dimensions.

Euclidean Distance



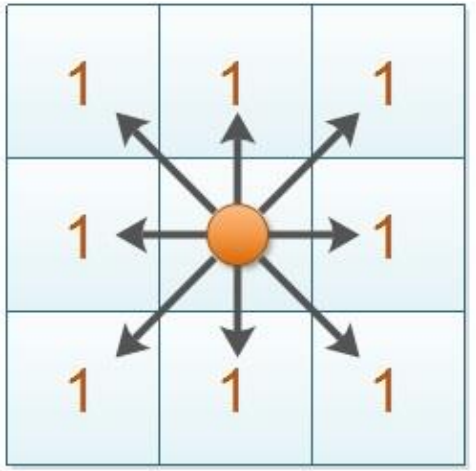
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Manhattan Distance



$$|x_1 - x_2| + |y_1 - y_2|$$

Chebyshev Distance



$$\max(|x_1 - x_2|, |y_1 - y_2|)$$

MINKOWSKY DISTANCE

- This is a generalization of both **Euclidean and Manhattan distances**. It includes a parameter "p" that allows you to control the distance calculation.

- When p = 1, it's equivalent to the Manhattan distance, and when p = 2, it's the Euclidean distance.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

IS IT ACTUALLY “DISTANCE” ?

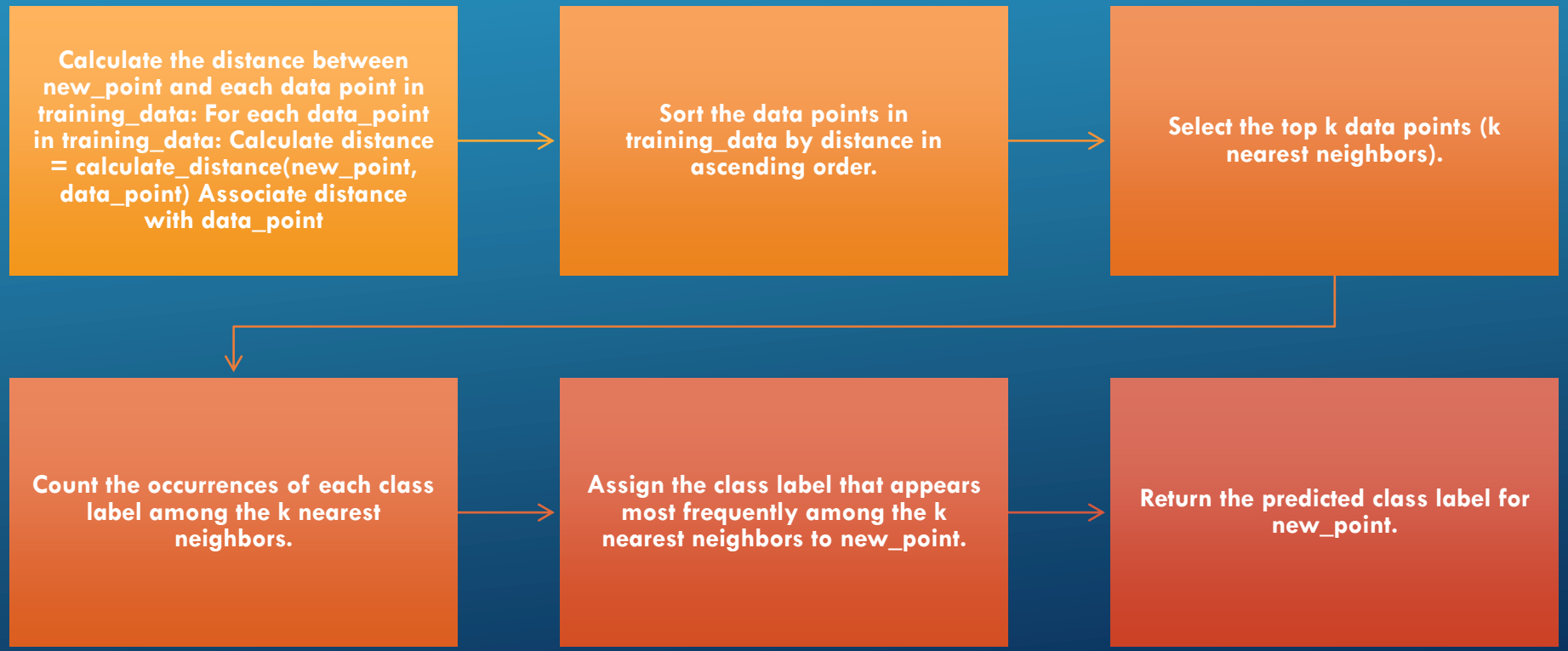
- The calculations using the previously mentioned metrics can be divided into two types of **“Weights”:**
  - Uniform Weights
  - Inversely Proportional (Actual Distance)

HOW TO FIND NEAREST POINTS ? (SEARCHING ALGORITHM)

- Brute Force
- KDTree
- BallTree

STEPS

Inputs:
- Training dataset: training_data (features and corresponding class labels)
- New data point: new_point (features for which we want to predict the class)
- Number of neighbors: k



PROS & CONS.

- KNN is simple to understand and implement.
- It is a lazy learner, meaning it doesn't perform explicit training and retains the entire training dataset.
- It can handle multi-class classification and can be adapted for multi-label classification as well.
- KNN's performance can be sensitive to the choice of distance metric, data scaling, and the value of "k."
- It works well when the decision boundaries are irregular and when the dataset is relatively small.
- KNN can be computationally expensive, especially for large datasets, as it requires calculating distances for all data points.

TIME FOR PRACTICALITY (15 MINUTES)

BREAK (10 MINUTES)

DECISION TREE

A SEQUENCE OF DECISIONS

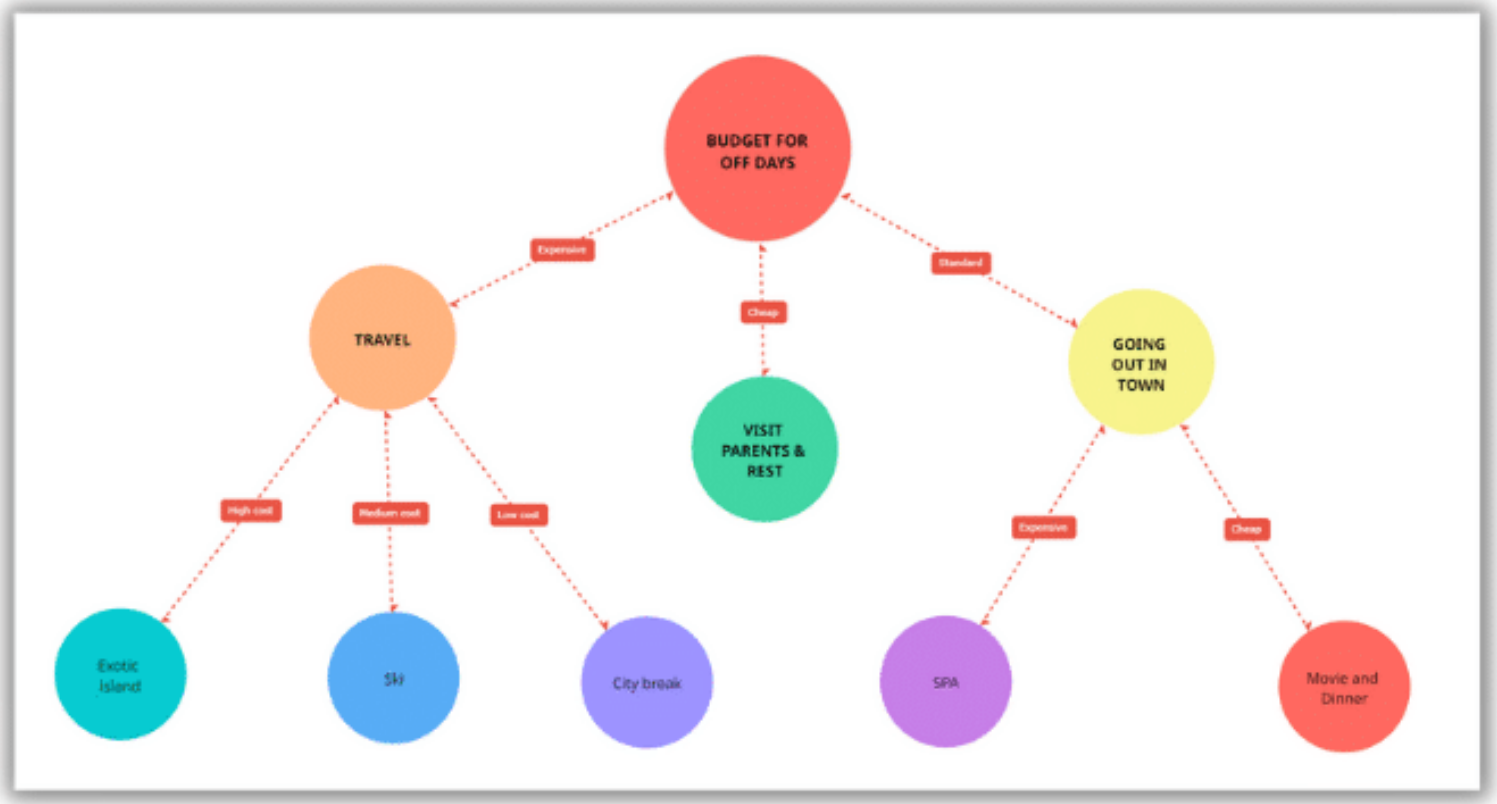
In the summer days, do you have any criteria to get up from your bed and get out into the hot weather for important mission ?

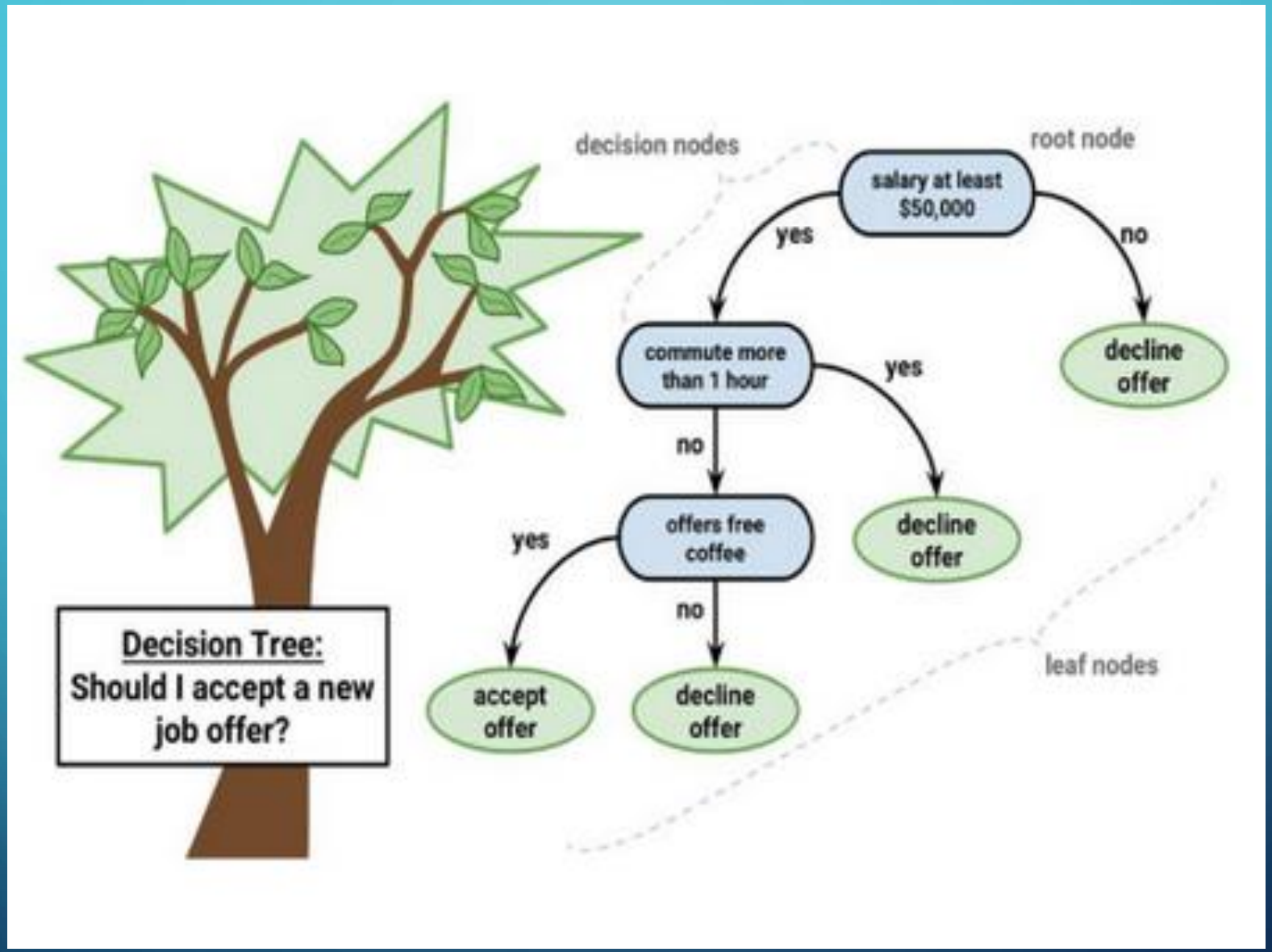
Is there a truly something important to make you go out in that hot weather ?

You should re-consider you Decisions based on your priorities. You can use such a thing that makes logic rules that help you make a decision such as:

Decision Tree

Decision tree examples




Decision Tree:
Should I accept a new job offer?
decision nodes
root node
salary at least $50,000
yes
no
commute more than 1 hour
yes
decline offer
no
offers free coffee
yes
decline offer
yes
no
accept offer
decline offer
leaf nodes

```
public static string DetermineGender(int input) {
  string gender = string.Empty;

  if (input == 0) {
    gender = "male";
  } else if (input == 1) {
    gender = "woman";
  } else {
    gender = "unknown";
  }

  return gender;
}
```

SOMETHING YOU MAY BE FAMILIAR WITH (GROUP OF IF-ELSEIF STATEMENTS)

WHAT IS “DECISION TREE” ?

- **Decision tree** is one of the most powerful tools of **supervised learning** algorithms used for both classification and regression tasks in machine learning.
- It builds a **flowchart-like tree structure** where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

- It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

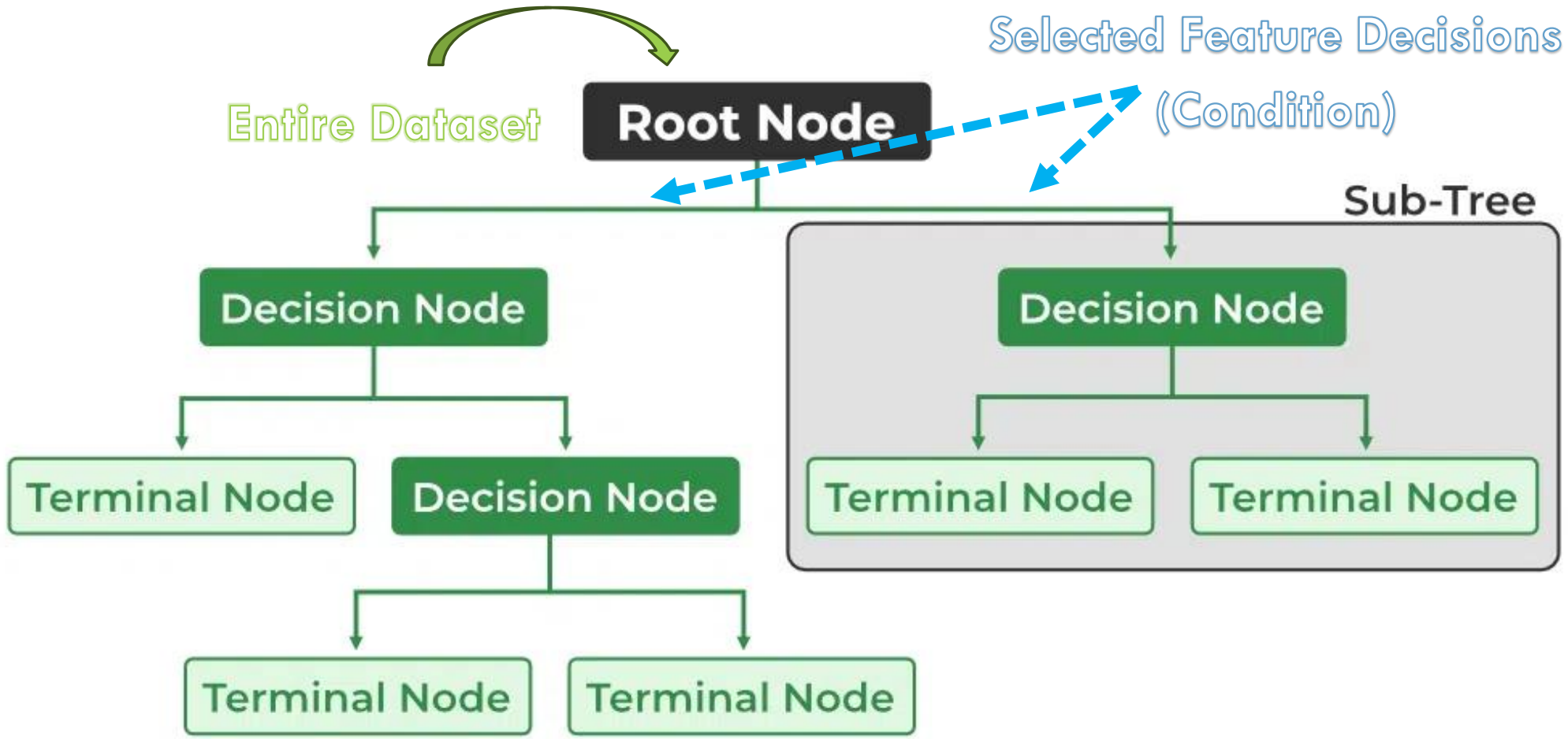
WHY DECISION TREE ?

- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into sub-trees.

Before going to Decision Tree You should know some TERMINOLOGIES

DECISION TREE TERMINOLOGIES

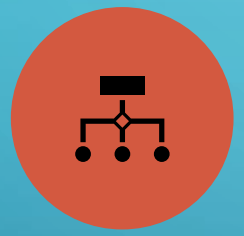
Terminology	Description
Root Node	Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
Decision/Internal Node	A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
Leaf/Terminal Node	A node without any child nodes that indicates a class label or a numerical value.
Splitting	The process of splitting a node into two or more sub-nodes **using a split criterion** and a selected feature.
Branch/Sub-Tree	A subsection of the decision tree starts at an internal node and ends at the leaf nodes
Pruning	The process of removing branches from the tree that do not provide any additional information or lead to overfitting (**Optimization**)


Entire Dataset
Root Node
Selected Feature Decisions
(Condition)
Sub-Tree
Decision Node
Decision Node
Terminal Node
Decision Node
Terminal Node
Terminal Node
Terminal Node
Terminal Node

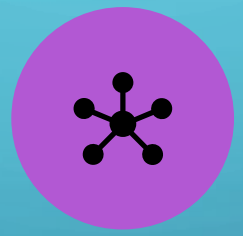
SOME NOTES YOU SHOULD REMEMBER !

Decision tree is not unique, as different ordering of internal nodes can give different decision tree.

DECISION TREE MAY BE N-ARY, N ≥ 2.



ALL NODES DRAWN WITH **CIRCLE (ELLIPSE)** ARE CALLED INTERNAL NODES.



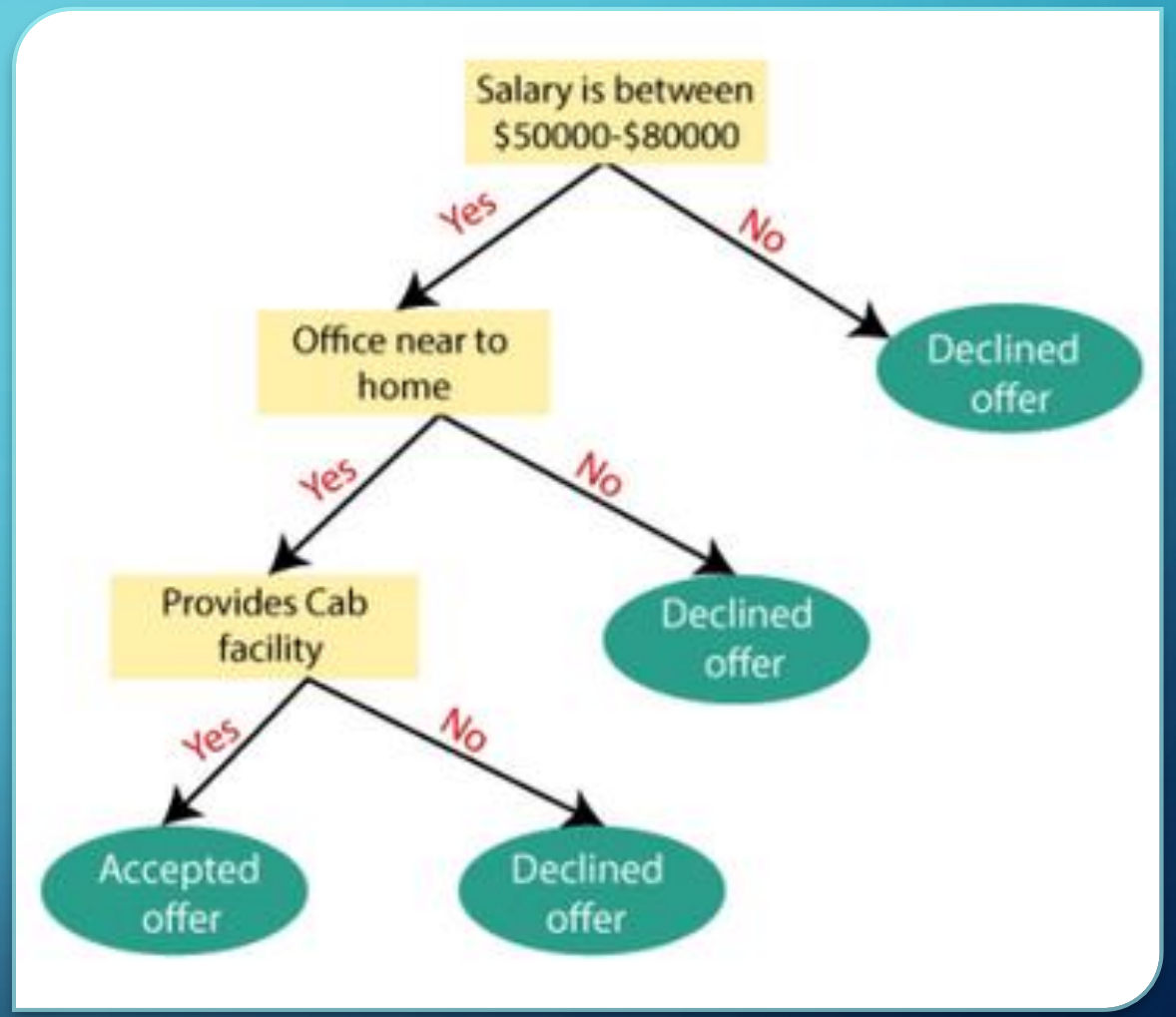
ALL NODES DRAWN WITH **SQUARE BOXES** ARE CALLED TERMINAL NODES OR LEAF NODES.

EDGES OF A NODE REPRESENT **THE OUTCOME FOR A VALUE OF THE NODE.**

IN A PATH, A NODE WITH SAME LABEL IS **NEVER REPEATED**.

EXAMPLE

- Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by **Attribute Selection Measure –ASM-**).



DECISION TREE ALGORITHM FOR CLASSIFICATION

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf nodeClassification and Regression Tree algorithm.

https://www.geeksforgeeks.org/decision-tree/

PSEUDOCODE

- Input: D : Training dataset
- Output: T : Decision tree

Steps

1. If all tuples in D belongs to the same class C_j
   Add a leaf node labeled as C_j
   Return *// Termination condition*
2. **Select** an attribute A_i (so that it is not selected twice in the same branch)
3. **Partition** $D = \{ D_1, D_2, \ldots, D_p \}$ based on p different values of A_i in D
4. For each $D_k \in D$
   Create a node and add an edge between D and D_k with label as the A_i's attribute value in D_k
5. For each $D_k \in D$
   BuildTD(D_k) *// Recursive call*
6. Stop

HOW TO HANDLE ATTRIBUTES (SPLITTING NODES) ?

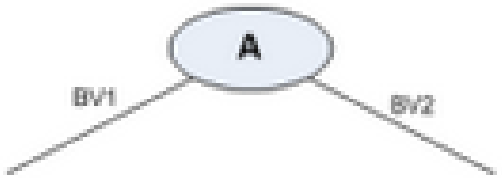
- You should know how to split nodes, for any kind of attribute

What kind of attribute we will face ?

- Binary Attribute
- Nominal/Categorical Attribute
- Ordinal Attribute
- Numerical Attribute

BINARY ATTRIBUTE

- This is the simplest case of node splitting
- The test condition for a binary attribute generates only two outcomes



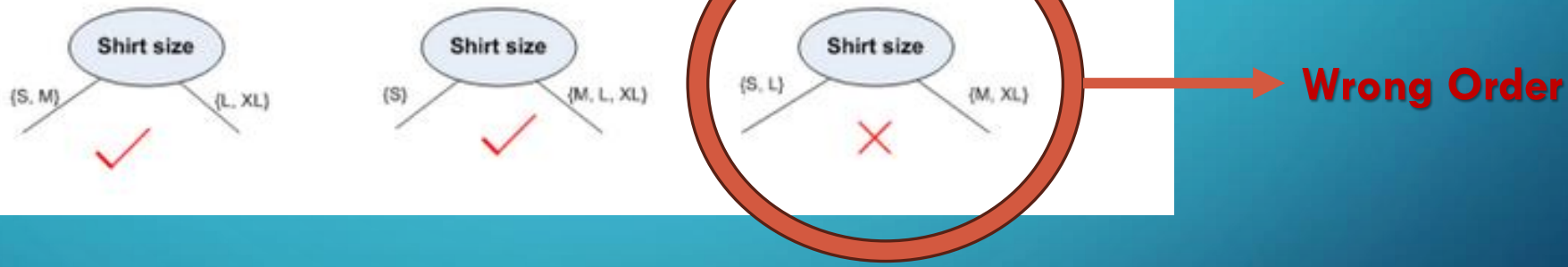
NOMINAL/CATEGORICAL ATTRIBUTE

- Since a nominal attribute can have many values, its test condition can be expressed in two ways:
  - A multi-way split
  - A binary split
- Muti-way split: Outcome depends on the number of distinct values for the corresponding attribute
- Binary splitting by grouping attribute values

ORDINAL ATTRIBUTE

- It also can be expressed in two ways:
  - A multi-way split
  - A binary split

- Muti-way split: It is same as in the case of nominal attribute
- Binary splitting attribute values should be grouped maintaining the order property of the attribute values



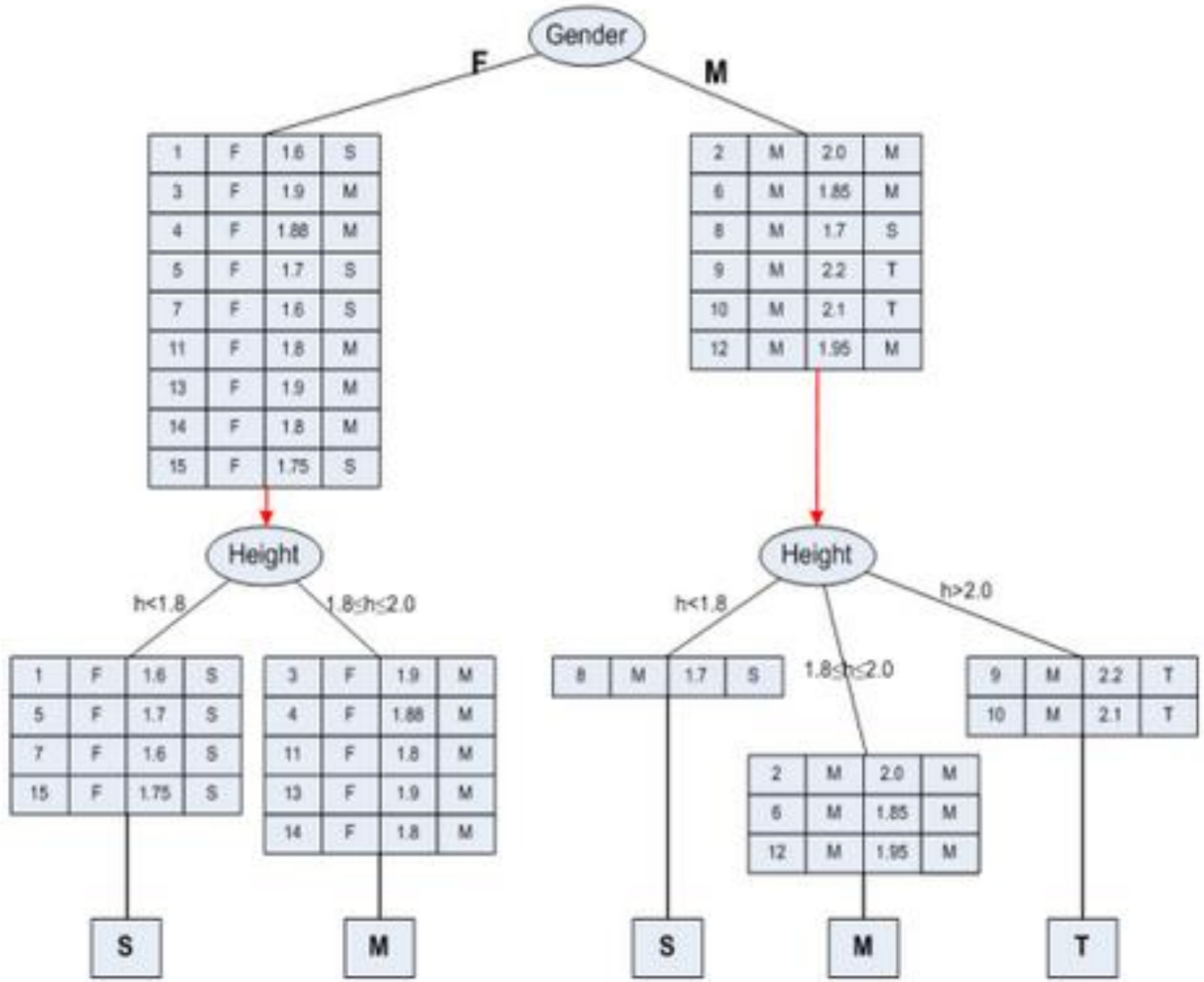
NUMERICAL ATTRIBUTE

- For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
  - Binary outcome: $A > v$ or $A \leq v$
    - In this case, decision tree induction must consider all possible split positions
  - Range query : $v_i \leq A < v_{i+1}$ for $i = 1, 2, \ldots, q$ (if q number of ranges are chosen)
    - Here, q should be decided a priori

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

EXAMPLE

- Consider the following dataset, how would you split each attribute/feature ?


Gender
F
M
1 F 1.6 S
3 F 1.9 M
4 F 1.88 M
5 F 1.7 S
7 F 1.6 S
11 F 1.8 M
13 F 1.9 M
14 F 1.8 M
15 F 1.75 S
2 M 2.0 M
6 M 1.85 M
8 M 1.7 S
9 M 2.2 T
10 M 2.1 T
12 M 1.95 M
Height
h<1.8
1.8≤h≤2.0
1 F 1.6 S
5 F 1.7 S
7 F 1.6 S
15 F 1.75 S
3 F 1.9 M
4 F 1.88 M
11 F 1.8 M
13 F 1.9 M
14 F 1.8 M
S
M
Height
h<1.8
1.8≤h≤2.0
h>2.0
8 M 1.7 S
2 M 2.0 M
6 M 1.85 M
12 M 1.95 M
9 M 2.2 T
10 M 2.1 T
S
M
T

WHAT ABOUT ATTRIBUTE SELECTION MEASURE ?

ATTRIBUTE SELECTION MEASURE (ASM)

- A tree can be "***learned***" by splitting the source set into subsets based on Attribute Selection Measures. **(The core of learning process of Decision Tree)**

- Attribute selection measure (ASM) is a criterion used in decision tree algorithms to evaluate the usefulness of different attributes for splitting a dataset.

The Goal ?

- The goal of **ASM** is to identify the attribute that will create **the most homogeneous subsets of data after the split**, thereby maximizing the information gain.

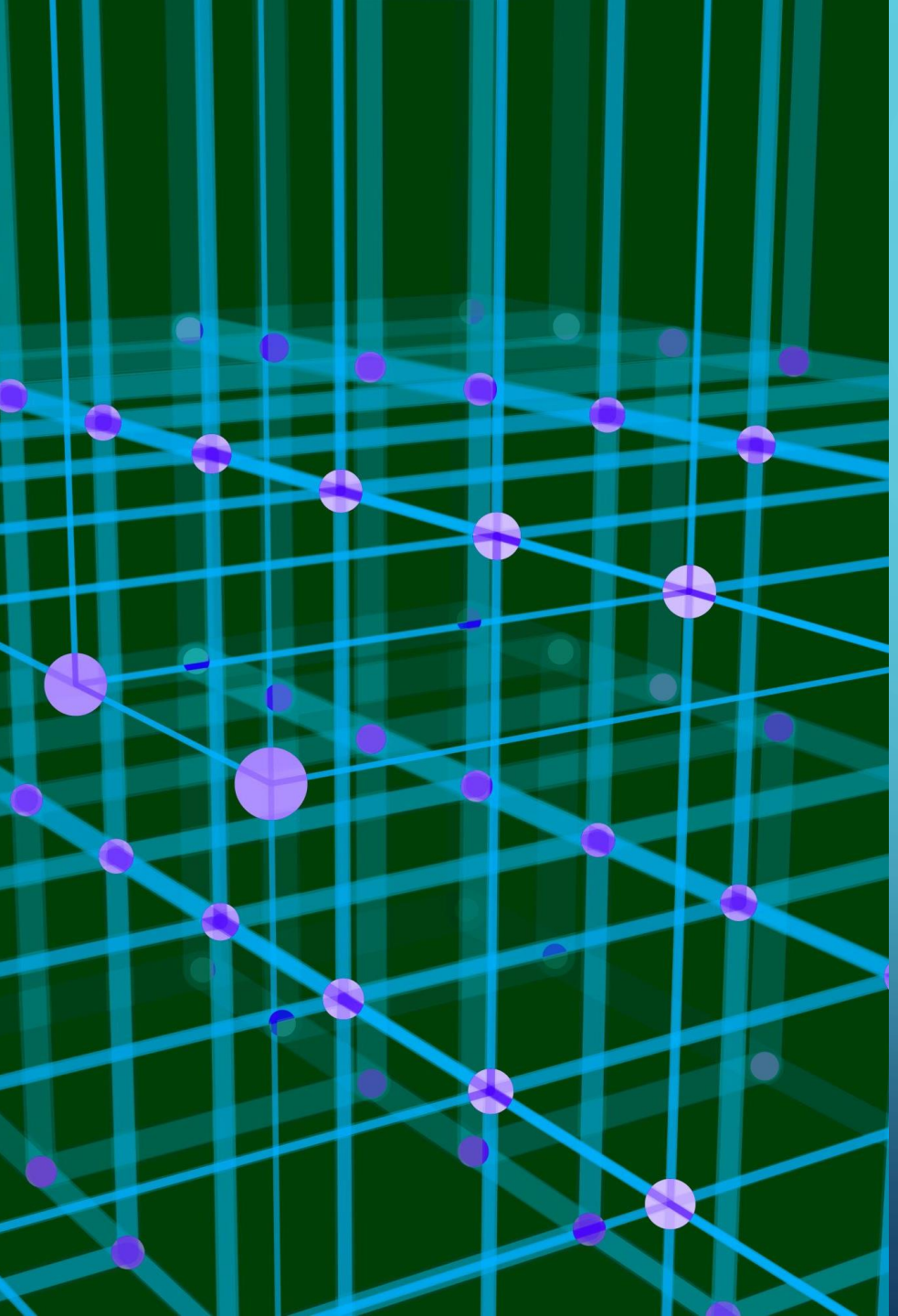
ATTRIBUTE SELECTION MEASURE (ASM) –CONT.

- This process is repeated on each derived subset in a recursive manner called ***recursive partitioning***.
- The recursion is completed when the subset at a node **all has the same value of the target variable**, or when splitting no longer adds value to the predictions.
- The construction of a decision tree classifier does not require any domain knowledge or parameter setting and therefore is appropriate for exploratory knowledge discovery.

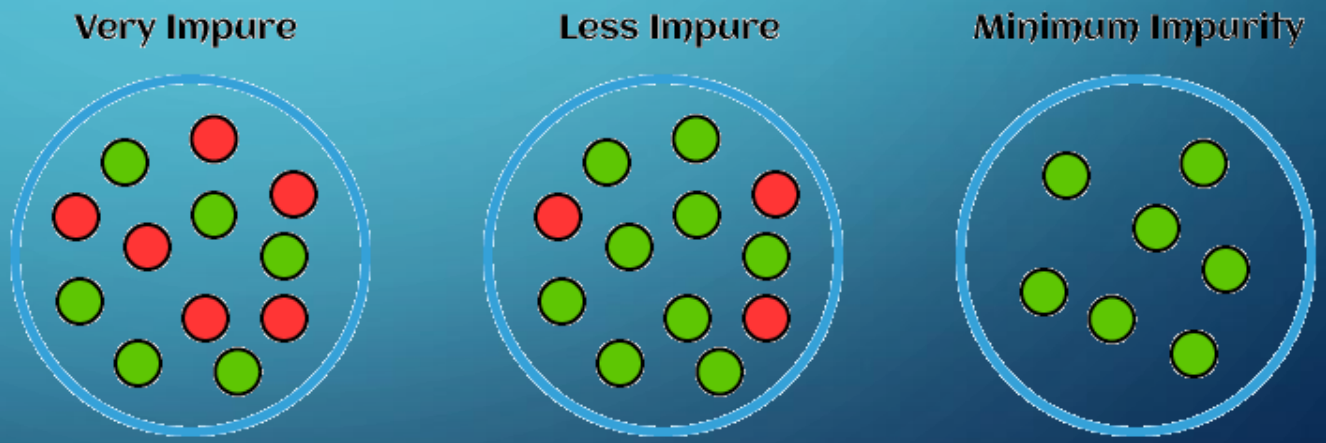
WARNING !
This makes Decision Tree able to handle High-dimensional datasets, but the learning process (ASM) could be sensitive to outliers, missing values or wrong data !

TYPES OF ASM

- Entropy
- Gini Index
- Information Gain



ENTROPY



ENTROPY

- **Entropy** is the measure of the degree of randomness or uncertainty in the dataset. In the case of classifications, It measures the randomness based on the distribution of class labels in the dataset.
- The entropy for a subset of the original dataset having K number of classes for the i^{th} node can be defined as:

$$H_i = -\sum_{k \epsilon K}^{n} p(i,k) \log_2 p(i,k)$$

Where,

•k is the particular class from K classes

•p(k) is the proportion of the data points that belong to class k to the total number of data points in dataset sample S.

•Here p(i,k) should not be equal to zero.

1. The entropy is 0 when the dataset is completely homogeneous, meaning that each instance belongs to the same class. It is the lowest entropy indicating no uncertainty in the dataset sample.
2. when the dataset is equally divided between multiple classes, the entropy is at its maximum value. Therefore, entropy is highest when the distribution of class labels is even, indicating maximum uncertainty in the dataset sample.
3. Entropy is used to evaluate the quality of a split. The goal of entropy is to select the attribute that minimizes the entropy of the resulting subsets, by splitting the dataset into more homogeneous subsets with respect to the class labels.
4. The highest information gain attribute is chosen as the splitting criterion (i.e., the reduction in entropy after splitting on that attribute), and the process is repeated recursively to build the decision tree.

IMPORTANT POINTS RELATED TO "ENTROPY"

- **Note that each node including the root node, the entropy of them will be calculated.**

INFORMATION GAIN

- **Information gain** measures **the reduction in entropy or variance** that results from splitting a dataset based on a specific property.
- It is used in decision tree algorithms to determine **the usefulness of a feature** by partitioning the dataset into more homogeneous subsets with respect to the class labels or target variable.
- The higher the information gain, the more valuable the feature is in predicting the target variable.

INFORMATION GAIN –CONT.

- **Information gain** measures the reduction in entropy or variance achieved by partitioning the dataset on attribute A.

- The attribute that **maximizes** information gain is chosen as the splitting criterion for building the decision tree.

$$\text{Information Gain(H, A)} = H - \sum \frac{|H_V|}{|H|} H_v$$

Where,
- A is the specific attribute or class label
- $|H|$ is the entropy of dataset sample S
- $|H_V|$ is the number of instances in the subset S that have the value v for attribute A

WHAT IS THE LOSS FUNCTION OF DECISION TREE ? (LEARNING PROCESS)

Attribute Selection Measure

Gini Index or Information Gain

EXERCISE

Construct the Decision tree using the following Dataset for participating rules in ECPC 2023 (8 Candidates)

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	No
Male	Computers & AI	C++	Yes
Female	Engineering	C++	No
Male	Engineering	C++	Yes
Female	Computers & AI	Python	No
Female	Engineering	Python	Yes
Female	Computers & AI	C++	Yes
Male	Engineering	Python	No

SOLUTION

$$Entropy(D1) = -\frac{4}{8}log2\left(\frac{4}{8}\right) - \frac{4}{8}log2\left(\frac{4}{8}\right) = 1$$

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	No
Male	Computers & AI	C++	Yes
Female	Engineering	C++	No
Male	Engineering	C++	Yes
Female	Computers & AI	Python	No
Female	Engineering	Python	Yes
Female	Computers & AI	C++	Yes
Male	Engineering	Python	No

SOLUTION –CONT.

1

$$Gain(D1, Gender) = Entropy(D1) - \sum_{v \in \{Male, Female\}} \frac{|Ds|}{|D|} Entropy(Ds)$$

$$1 - \frac{4}{8}\left(-\frac{2}{4}log2\left(\frac{2}{4}\right) - \frac{2}{4}log2\left(\frac{2}{4}\right)\right) - \frac{4}{8}\left(-\frac{2}{4}log2\left(\frac{2}{4}\right) - \frac{2}{4}log2\left(\frac{2}{4}\right)\right) = 0$$

2

$$Gain(D1, Faculty) = Entropy(D1) - \sum_{v \in \{CAI, Engineering\}} \frac{|Ds|}{|D|} Entropy(Ds)$$

$$1 - \frac{4}{8}\left(-\frac{2}{4}log2\left(\frac{2}{4}\right) - \frac{2}{4}log2\left(\frac{2}{4}\right)\right) - \frac{4}{8}\left(-\frac{2}{4}log2\left(\frac{2}{4}\right) - \frac{2}{4}log2\left(\frac{2}{4}\right)\right) = 0$$

3

$$Gain(D1, PL) = Entropy(D1) - \sum_{v \in \{Python, C++\}} \frac{|Ds|}{|D|} Entropy(Ds)$$

$$1 - \frac{4}{8}\left(-\frac{1}{4}log2\left(\frac{1}{4}\right) - \frac{3}{4}log2\left(\frac{3}{4}\right)\right) - \frac{4}{8}\left(-\frac{3}{4}log2\left(\frac{3}{4}\right) - \frac{1}{4}log2\left(\frac{1}{4}\right)\right) = 0.1887218755$$

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	No
Male	Computers & AI	C++	Yes
Female	Engineering	C++	No
Male	Engineering	C++	Yes
Female	Computers & AI	Python	No
Female	Engineering	Python	Yes
Female	Computers & AI	C++	Yes
Male	Engineering	Python	No

Programming Language Attribute has the highest information Gain

SOLUTION –CONT.

- **First Phase**

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	No
Male	Computers & AI	C++	Yes
Female	Engineering	C++	No
Male	Engineering	C++	Yes
Female	Computers & AI	Python	No
Female	Engineering	Python	Yes
Female	Computers & AI	C++	Yes
Male	Engineering	Python	No

Python

C++

Gender	Faculty	Class
Male	Computers & AI	No
Female	Computers & AI	No
Female	Engineering	Yes
Male	Engineering	No

Gender	Faculty	Class
Male	Computers & AI	Yes
Female	Engineering	No
Male	Engineering	Yes
Female	Computers & AI	Yes

SOLUTION –CONT.

- **Dealing with "Python Node Branch"**

$$Entropy(D2) = -\frac{3}{4}log2\left(\frac{3}{4}\right) - \frac{1}{4}log2\left(\frac{1}{4}\right) = 0.8112781245$$

Gender	Faculty	Class
Male	Computers & AI	No
Female	Computers & AI	No
Female	Engineering	Yes
Male	Engineering	No

SOLUTION –CONT.

1

$$Gain(D2, Gender) = Entropy(D2) - \sum_{v \in \{Male, Female\}} \frac{|Ds|}{|D|} Entropy(Ds)$$

$$0.8112781245 - \frac{2}{4}\left(-\frac{2}{2}log2\left(\frac{2}{2}\right)\right) - \frac{2}{4}\left(-\frac{1}{2}log2\left(\frac{1}{2}\right) - \frac{1}{2}log2\left(\frac{1}{2}\right)\right) = 0.3112781245$$

2

$$Gain(D2, Faculty) = Entropy(D2) - \sum_{v \in \{CAI, Engineering\}} \frac{|Ds|}{|D|} Entropy(Ds)$$

$$0.8112781245 - \frac{2}{4}\left(-\frac{2}{2}log2\left(\frac{2}{2}\right)\right) - \frac{2}{4}\left(-\frac{1}{2}log2\left(\frac{1}{2}\right) - \frac{1}{2}log2\left(\frac{1}{2}\right)\right) = 0.3112781245$$

Gender	Faculty	Class
Male	Computers & AI	No
Female	Computers & AI	No
Female	Engineering	Yes
Male	Engineering	No

Both features have the same Information gain, choose one of them (e.g. choose Gender)

SOLUTION –CONT.

Gender	Faculty	Class
Male	Computers & AI	No
Female	Computers & AI	No
Female	Engineering	Yes
Male	Engineering	No

Male

Faculty	Class
Computers & AI	No
Engineering	No

No

Female

Faculty	Class
Computers & AI	No
Engineering	Yes

No

Computers & AI

Engineering

Yes

CAN YOU COMPLETE THE TREE ? (DIY)

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	No
Male	Computers & AI	C++	Yes
Female	Engineering	C++	No
Male	Engineering	C++	Yes
Female	Computers & AI	Python	No
Female	Engineering	Python	Yes
Female	Computers & AI	C++	Yes
Male	Engineering	Python	No

Python

C++

Gender	Faculty	Class
Male	Computers & AI	Yes
Female	Engineering	No
Male	Engineering	Yes
Female	Computers & AI	Yes

Gender	Faculty	Class
Male	Computers & AI	No
Female	Computers & AI	No
Female	Engineering	Yes
Male	Engineering	No

Male

Female

Faculty	Class
Computers & AI	No
Engineering	No

No

Faculty	Class
Computers & AI	No
Engineering	Yes

No

Computers & AI

Engineering

Yes

TEST YOUR DECISION TREE

Classify the following participant if he/she is eligible for the ECPC 2023

Gender	Faculty	Programming Language	Class
Male	Computers & AI	Python	??

DECISION TREE ADVANTAGES

Advantages of the Decision Tree:

1. It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
2. It can be very useful for solving decision-related problems.
3. It helps to think about all the possible outcomes for a problem.
4. There is less requirement of data cleaning compared to other algorithms.

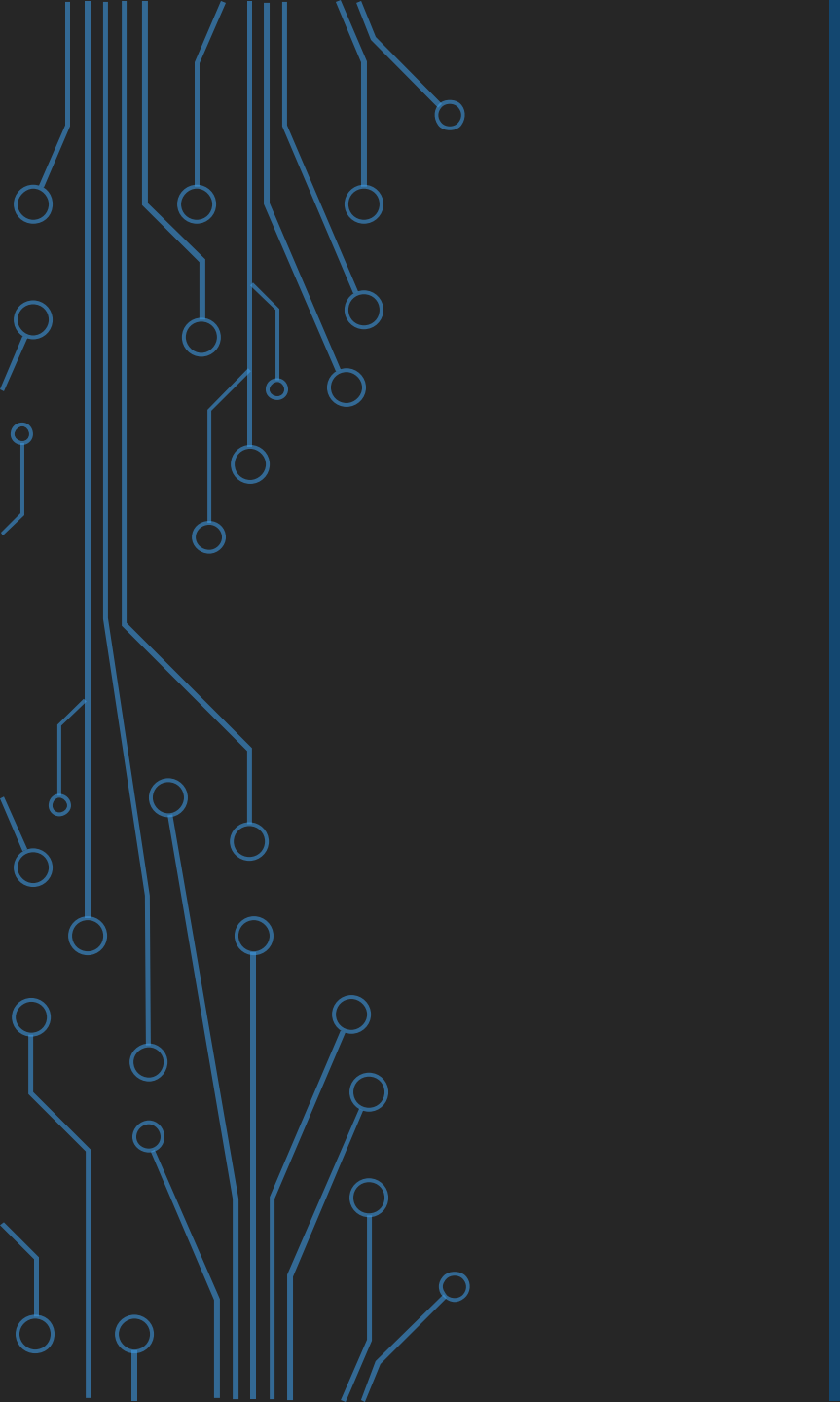
DECISION TREE DISADVANTAGES

Disadvantages of the Decision Tree:

1. The decision tree contains lots of layers, which makes it complex.
2. It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
3. For more class labels, the computational complexity of the decision tree may increase.

TIME FOR PRACTICALITY (5 MINUTES)

BREAK (10 MINUTES)



PROBABILISTIC LEARNING

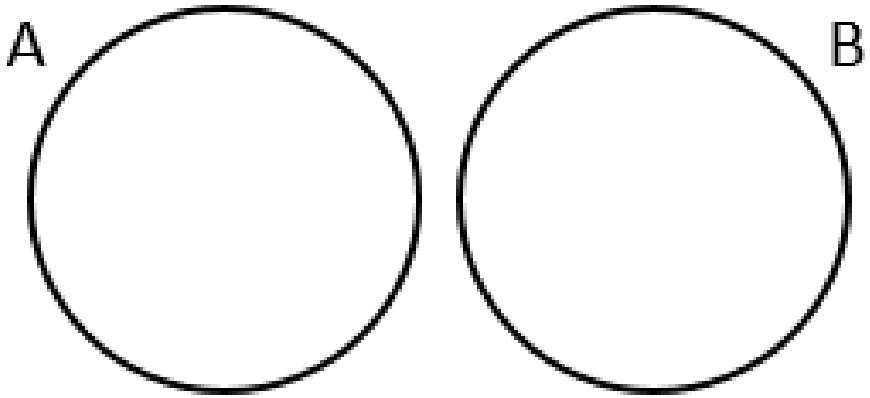
BAYESIAN METHOD

- The technique is used to describe the probability of events, and how probabilities should be revised in the light of additional information.
- These principles formed the foundation for what are now known as **Bayesian methods**.
- It suffices to say that a probability is a number between 0 and 1 (that is, between 0 percent and 100 percent), which captures the chance that an event will occur in the light of the available evidence.

BASIC CONCEPTS OF BAYESIAN METHODS

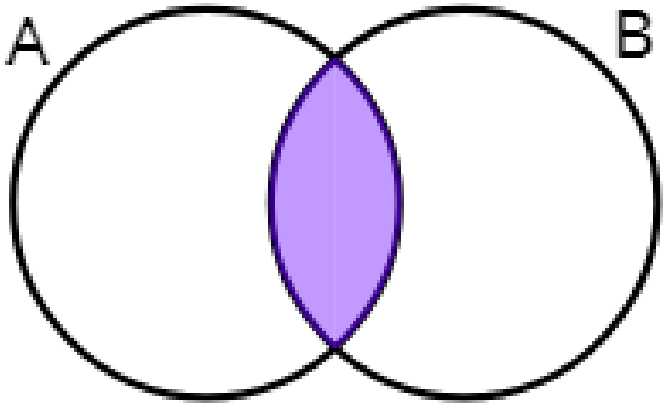
- The probability of an event is estimated from the observed data by dividing the number of trials in which the event occurred by the total number of trials.
- To denote these probabilities, we use notation in the form P(A), which signifies the probability of event A. For example, P(rain) = 0.30 and P(spam) = 0.20.
- The probability of all the possible outcomes of a trial must always sum to 1, because a trial always results in some outcome happening.
- Because an event cannot simultaneously happen and not happen, an event is always **mutually exclusive and exhaustive with its complement**, or the event comprising of the outcomes in which the event of interest does not happen.

Mutually Exclusive Events



$$P(A \text{ or } B) = P(A) + P(B)$$

Non-Mutually Exclusive Events



$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

NAÏVE BAYES

- The **Naïve Bayes** theorem is used to predict the probability of a certain class label given a set of features. It's **"naïve"** because it makes a strong assumption that **the features are conditionally independent given the class label**.

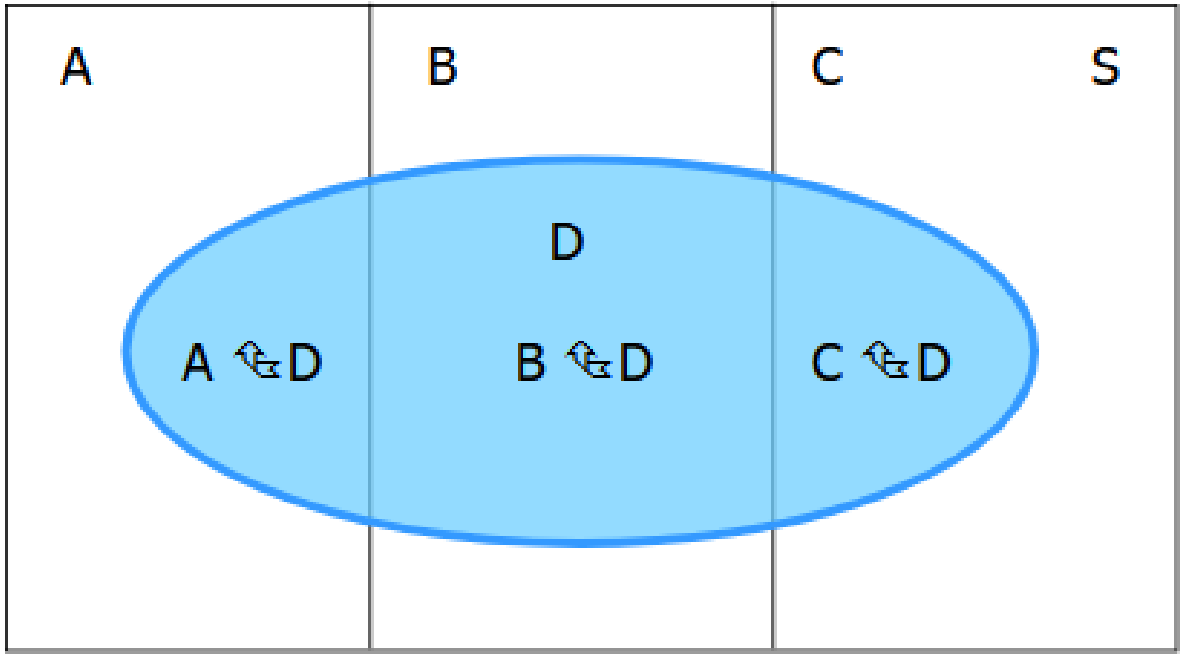
- In other words, it assumes that the presence or absence of a particular feature ***is not influenced by the presence or absence of any other feature***, given the class label.

TYPES OF NAÏVE BAYES

Gaussian Naïve Bayes: the algorithm assumes that the features (attributes) of the data follow a Gaussian (normal) distribution.

Bernoulli Naïve Bayes: each feature is treated as a binary random variable that can take one of two values: 0 (absence) or 1 (presence).

Multinomial Naïve Bayes: refers to the fact that it models the distribution of multiple categories (classes) using a multinomial distribution.



In this Venn Diagram, S is the whole sample space (everything), and D overlaps the other three sets. We will be doing more with this type of Venn.

CONDITIONAL PROBABILITY WITH NAÏVE BAYES

- The relationships between dependent events can be described using **Bayes' theorem**, as shown in the following formula.

$$p(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{p(B|A)p(A)}{P(B)}$$

- The notation *P(A|B)* is read as the probability of event *A*, given that event *B* occurred. This is known as **conditional probability**, since the probability of *A* is dependent (that is, conditional) on what happened with event *B*. Bayes' theorem tells us that our estimate of *P(A|B)* should be based on *P(A ∩ B)*, a measure of how often *A* and *B* are observed to occur together, and *P(B)*, a measure of how often *B* is observed to occur in general.

HOW THE MODEL WORKS ?

Training Phase:

- Given a labeled training dataset, the algorithm calculates the probabilities of each feature occurring for each class label.
- It estimates the conditional probabilities of feature counts for each class.

HOW THE MODEL WORKS ? (CONT.)

Prediction Phase:

- When a new document or data point needs to be classified, the algorithm calculates the conditional probabilities of each class label given the observed feature counts.
- It uses Bayes' theorem to calculate these probabilities and predicts the class label with the highest probability.

EXAMPLE

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

likelihood	Outlook w1			Temp w2			Humidity w3		Windy w4		
	Rainy w11	Overcast W12	Sunny w13	Hot w21	Mild w22	Cool w23	High w31	Normal w32	False w41	True w42	total
Play golf (yes)	2/9	4/9	3/9	2/9	4/9	3/9	3/9	6/9	6/9	3/9	9/14
Play golf (no)	3/5	0/5	2/5	2/5	2/5	1/5	4/5	1/5	2/5	3/5	5/14
Total	5/14	4/14	5/14	4/14	6/14	4/14	7/14	7/14	8/14	6/14	14

JOINT PROBABILITY DISTRIBUTION

SO, WEATHER AS THE FOLLOWING (RAINY, COOL, HIGH HUMIDITY AND WINDY) IS IT GOOD OR NOT TO PLAY GOLF ?

$$p(Play\ golf\ (no)|w11 \cap w23 \cap w31 \cap w42) = \frac{p(w11 \cap w23 \cap w31 \cap w42|Play\ golf\ (no))p(Play\ golf\ (no))}{P(w11 \cap w23 \cap w31 \cap w42)}$$

$$p(Play\ golf\ (no)|w11 \cap w23 \cap w31 \cap w42) = (3/5^*\ 1/5^*\ 4/5^*\ 3/5^*\ 5/14) = 0.020$$

$$p(Play\ golf\ (yes)|w11 \cap w23 \cap w31 \cap w42) = (2/9^*\ 3/9\ ^*\ 3/9\ ^*\ 3/9\ ^*\ 9/14) = 0.003$$

The probability of not playing golf = 0.020/(0.020+0.003 **= 0.79**

The probability of playing golf = 0.003/ (0.020+0.003)= **0.21**

The Answer is NO

TEST YOUR UNDERSTANDING

Check depend on the following if it is good to play golf or not
(Sunny, mild, Normal Humidity, Not Windy)

likelihood	Outlook w1			Temp w2			Humidity w3		Windy w4		
	Rainy w11	Overcast W12	Sunny w13	Hot w21	Mild w22	Cool w23	High w31	Normal w32	False w41	True w42	total
Play golf (yes)	2/9	4/9	3/9	2/9	4/9	3/9	3/9	6/9	6/9	3/9	9/14
Play golf (no)	3/5	0/5	2/5	2/5	2/5	1/5	4/5	1/5	2/5	3/5	5/14
Total	5/14	4/14	5/14	4/14	6/14	4/14	7/14	7/14	8/14	6/14	14

ADVANTAGES

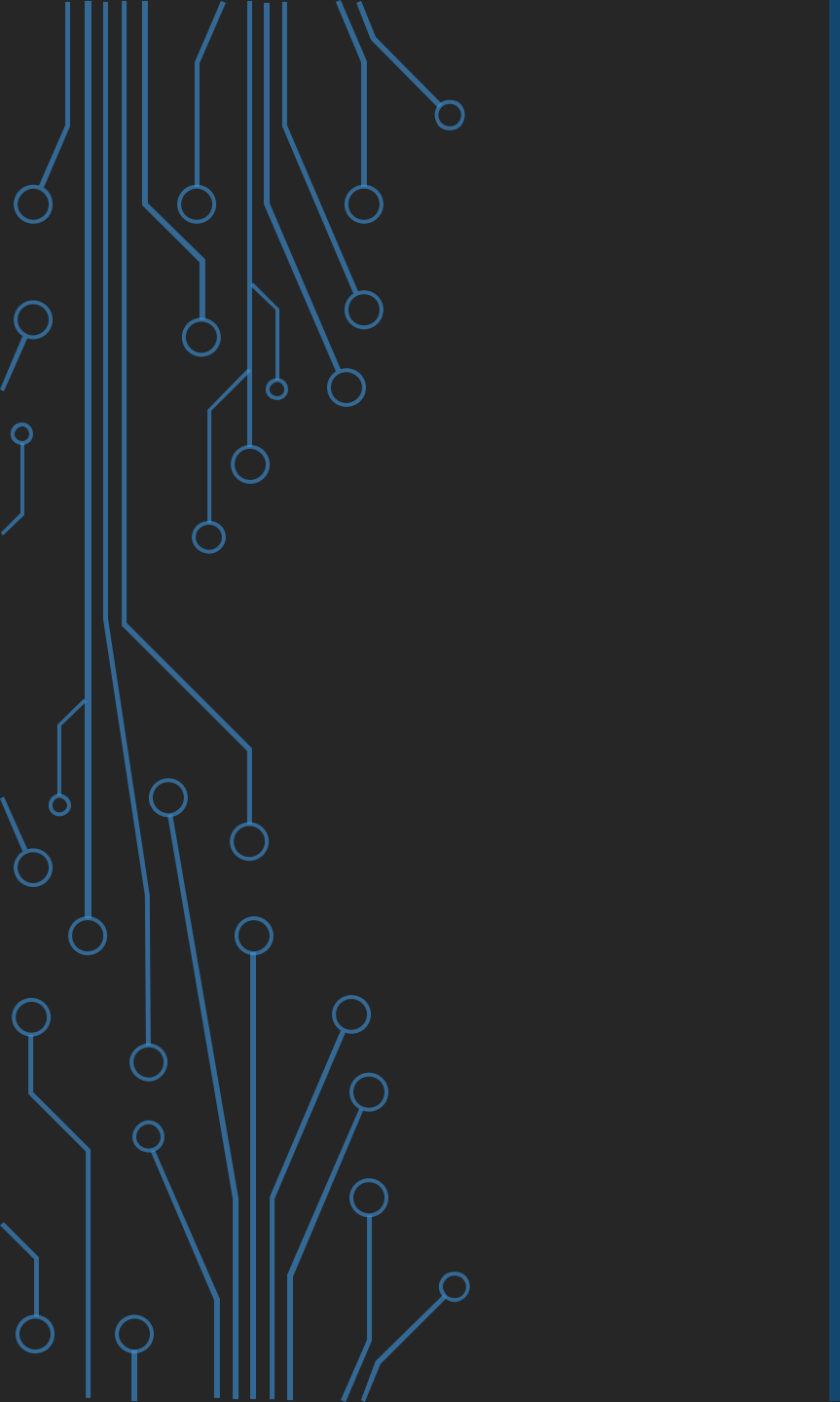
1. **Simplicity and Speed:** Naive Bayes is easy to understand and implement. It's computationally efficient and works well on large datasets. This makes it a good choice for quick prototyping and baseline models.
2. **Scalability:** Naive Bayes can handle high-dimensional data with relatively small sample sizes. It's particularly useful for tasks like text classification where the data is often sparse and high-dimensional.
3. **Real-time Prediction:** Due to its simplicity and efficiency, Naive Bayes can make real-time predictions, making it suitable for applications that require quick decisions.
4. **Strong Performance on Certain Tasks:** Despite its assumptions (naive independence), Naive Bayes can perform surprisingly well on various text classification tasks and situations where the independence assumption holds reasonably well.
5. **Handles Irrelevant Features:** Naive Bayes tends to perform well even when some irrelevant features are present in the data. It is robust to features that may not contribute to the classification decision.
6. **Interpretable Results:** Naive Bayes provides a straightforward way to interpret results and understand the importance of individual features in making predictions.

DISADVANTAGES

1. **Simplistic Assumption:** The key assumption of feature independence can limit the model's ability to capture complex relationships between features. This can lead to suboptimal performance on tasks with strong dependencies between features.
2. **Sensitive to Data Quality:** Naive Bayes can be sensitive to the quality of the training data. Inaccurate or noisy data can negatively impact its performance.
3. **Limited Expressiveness:** While effective for certain tasks, Naive Bayes may not capture subtle patterns and nuances in the data as well as more complex algorithms.
4. **Unsuitable for Continuous Data:** Naive Bayes is designed for discrete data, and its performance may degrade when applied to continuous numerical data without appropriate discretization.
5. **Requires Large Amounts of Data:** Naive Bayes' performance tends to improve with more training data. In cases where data is limited, more complex models might be necessary.
6. **Class Imbalance:** Naive Bayes may struggle with imbalanced datasets, where one class significantly outweighs the other(s). It may disproportionately favor the majority class.

TIME FOR PRACTICALITY (5 MINUTES)

BREAK (10 MINUTES)



SUPPORT-VECTOR MACHINE

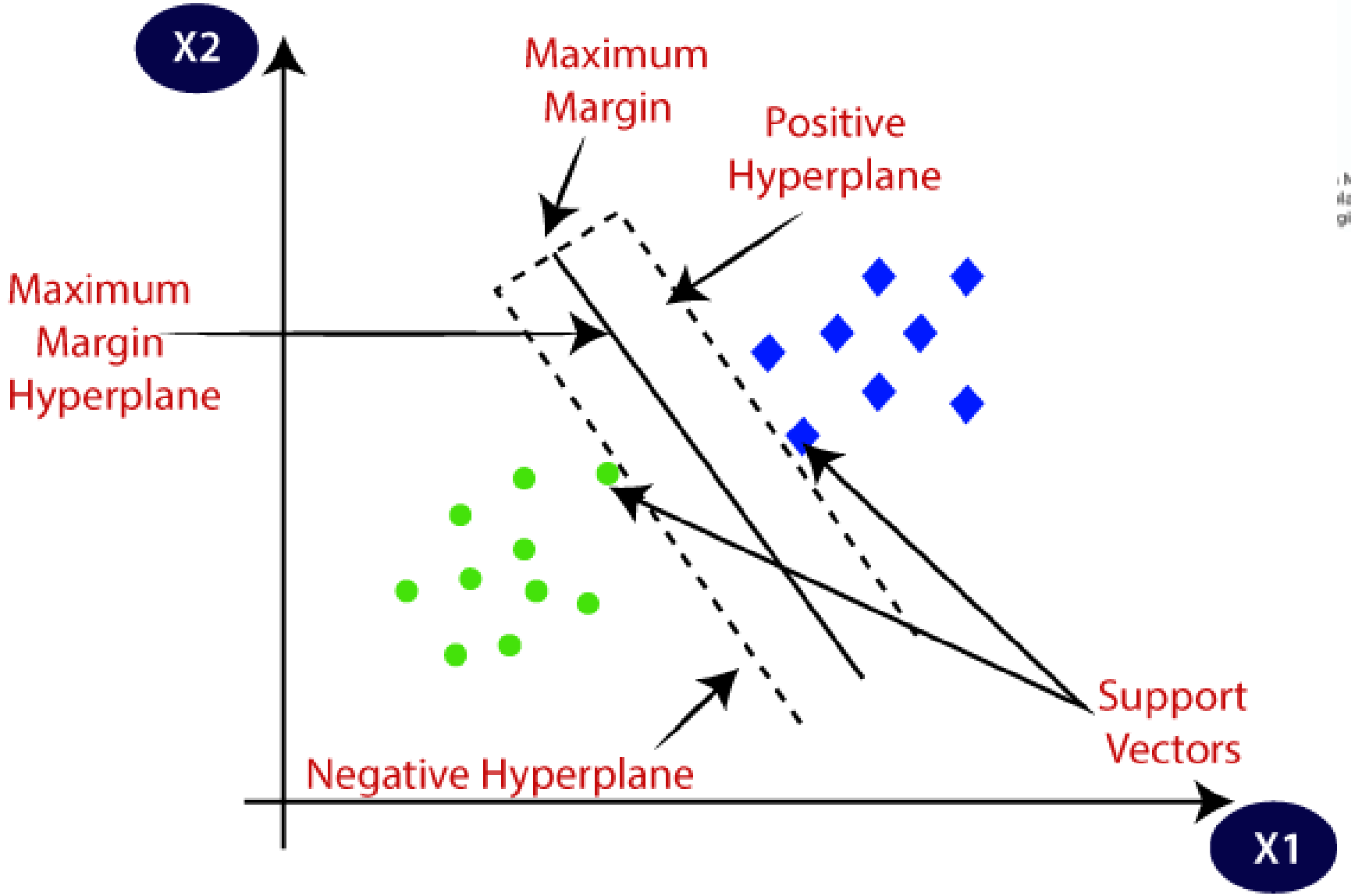
WHAT IS SUPPORT-VECTOR MACHINE ?

- A **Support Vector Machine (SVM)** is a powerful and versatile supervised machine learning algorithm used for both classification and regression tasks.

- SVMs are particularly effective in scenarios where there is a clear separation between classes or when the data is not linearly separable.

What is the main goal ?

- The main goal of an SVM is to find a hyperplane (or decision boundary) that best separates the data into different classes while maximizing the margin between the classes.


X2
Maximum
Margin
Positive
Hyperplane
Maximum
Margin
Hyperplane
Support
Vectors
Negative Hyperplane
X1

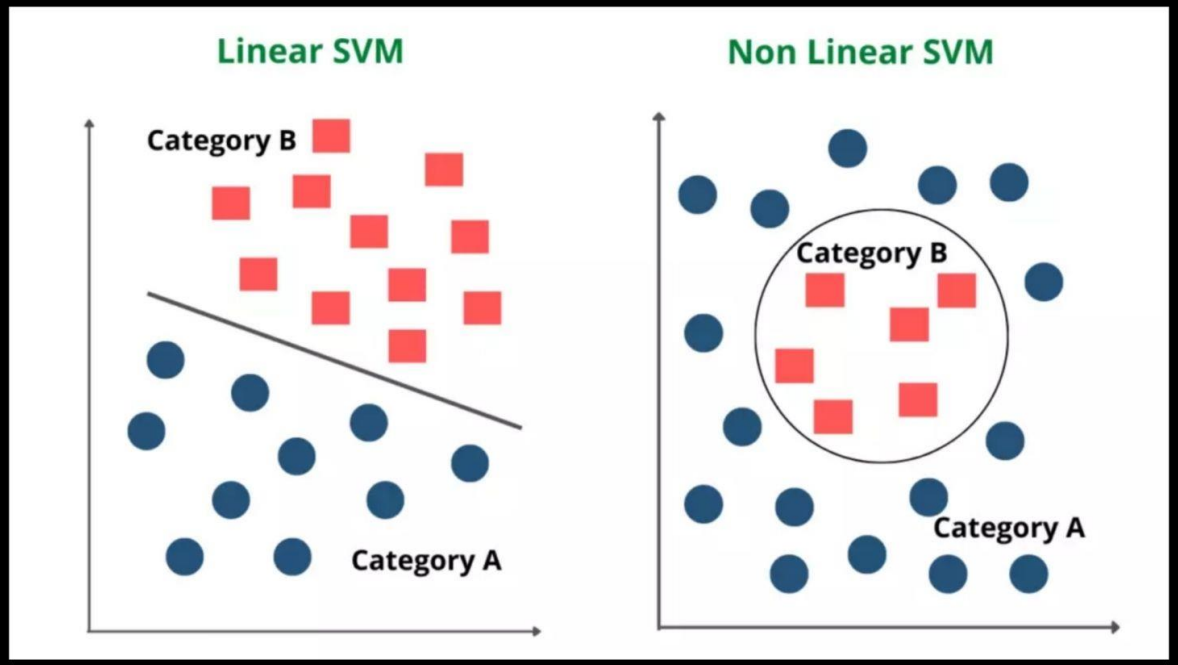
IMPORTANT TERMS

- **Hyperplane:** A hyperplane is a decision boundary that separates data points of different classes. In a binary classification problem, it's a line in two dimensions or a hyperplane in higher dimensions.
- **Support Vectors:** These are the data points that are closest to the hyperplane and have the most influence in determining its position. Support vectors define the margin and play a crucial role in SVM.
- **Margin:** The margin is the distance between the hyperplane and the nearest data points (support vectors) of each class. SVM aims to maximize this margin.
- **Kernel:** A kernel function is used to transform the data into a higher-dimensional space, allowing SVM to find a hyperplane in a transformed feature space. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid.

TYPES OF SVM

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

SVM in ML


Linear SVM
Category B
Category A
Non Linear SVM
Category B
Category A

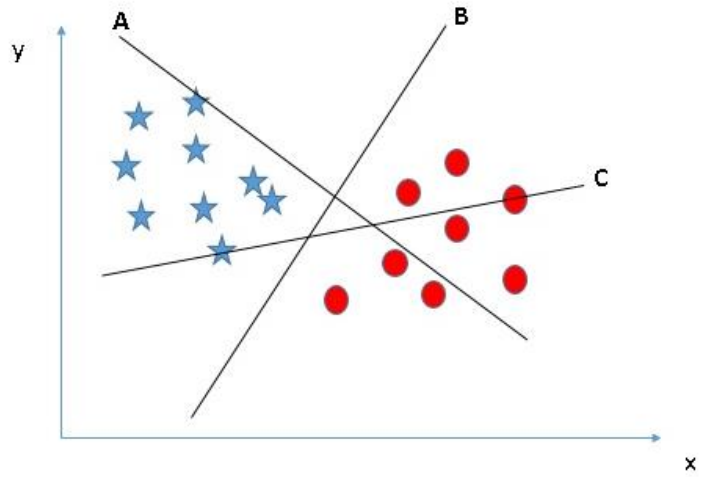
EXISTENCE OF THE KERNELS (HYPERPLANE)

The mathematical function used for the transformation is known as the ***kernel function.*** Following are the popular functions.

- Linear
- Polynomial
- Radial basis function (RBF)
- Sigmoid

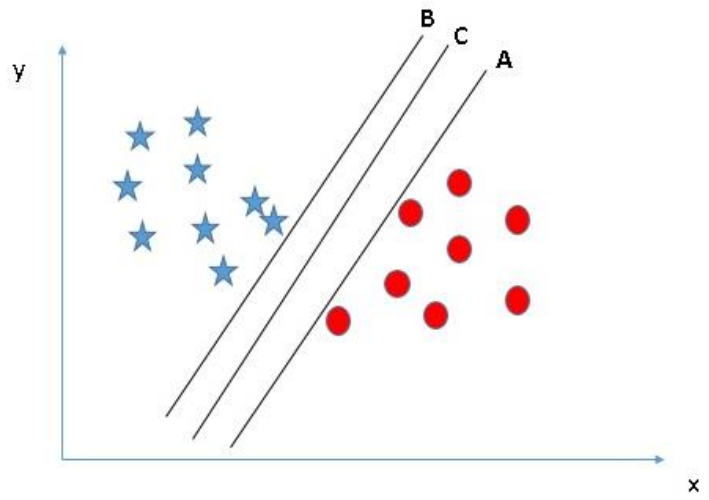
WHY "MARGIN" ?

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



WHY "MARGIN" ? (CONT.)

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B, and C), and all segregate the classes well. Now, How can we identify the right hyper-plane?

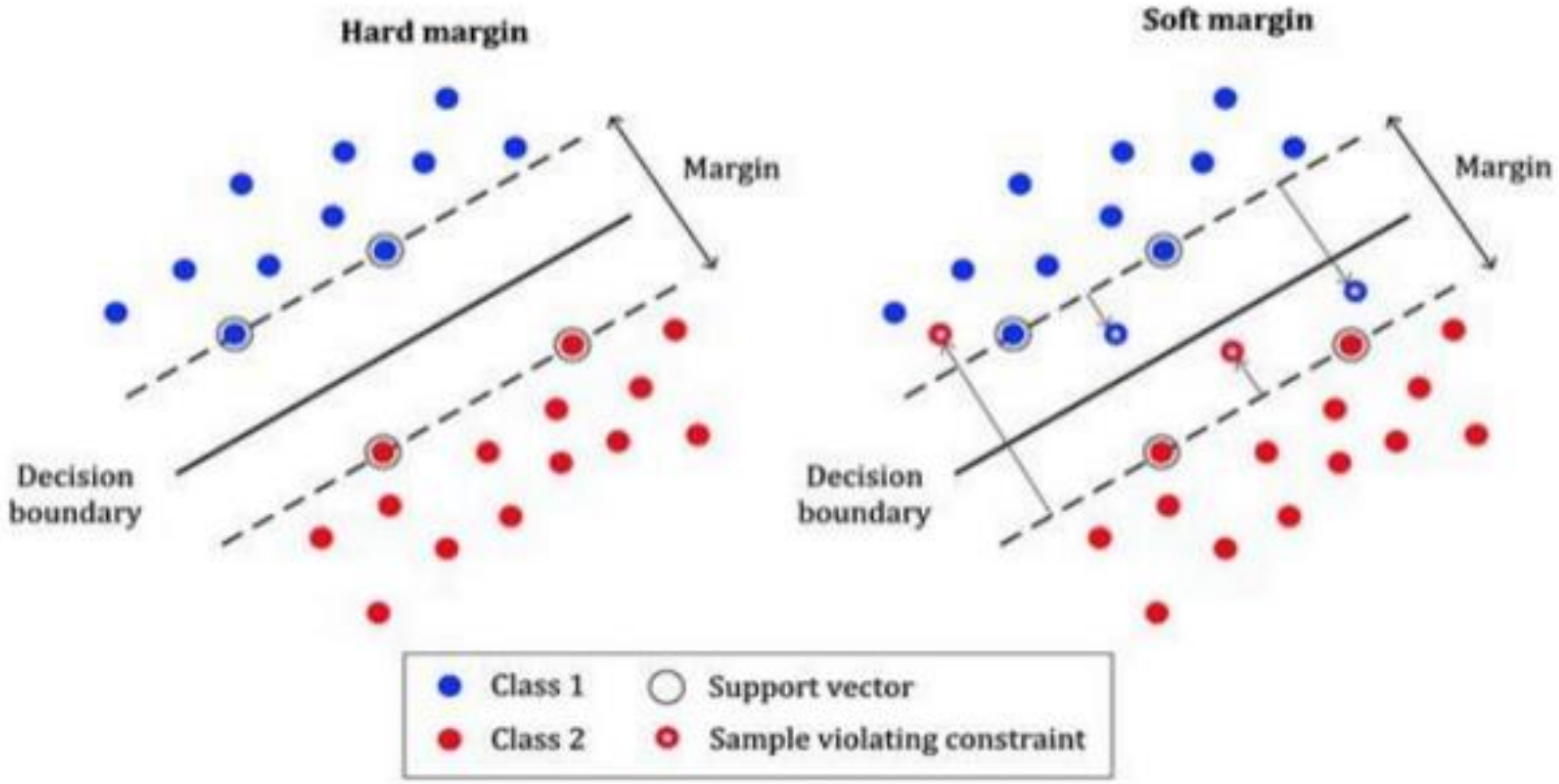


THE LEARNING PROCESS

- The distance of the vectors from the hyperplane is called the margin which is a separation of a line to the closest class points.

- We would like to choose a hyperplane that **maximizes the margin** between classes.

  - Soft Margin
  - Hard Margin



The cost function is to maximize the margin

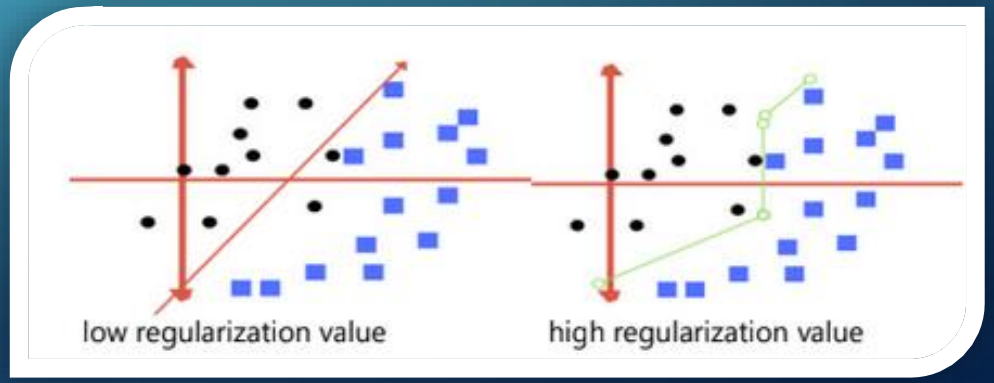
1- Soft Margin – As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay on either the incorrect side of the hyperplane and between the margin and correct side of the hyperplane.

2- Hard Margin – If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

SOFT VS. HARD MARGIN

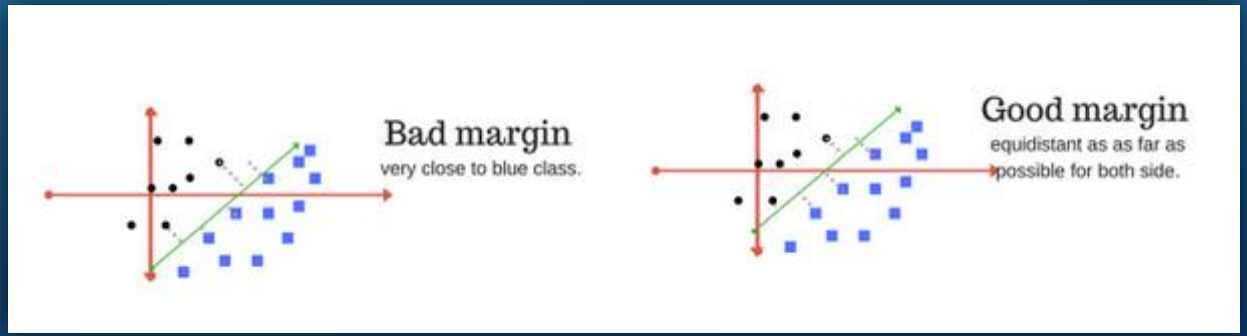
REGULARIZATION

- The Regularization parameter (often termed as C parameter in python‘s sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.



GOOD FITTING

- A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin.



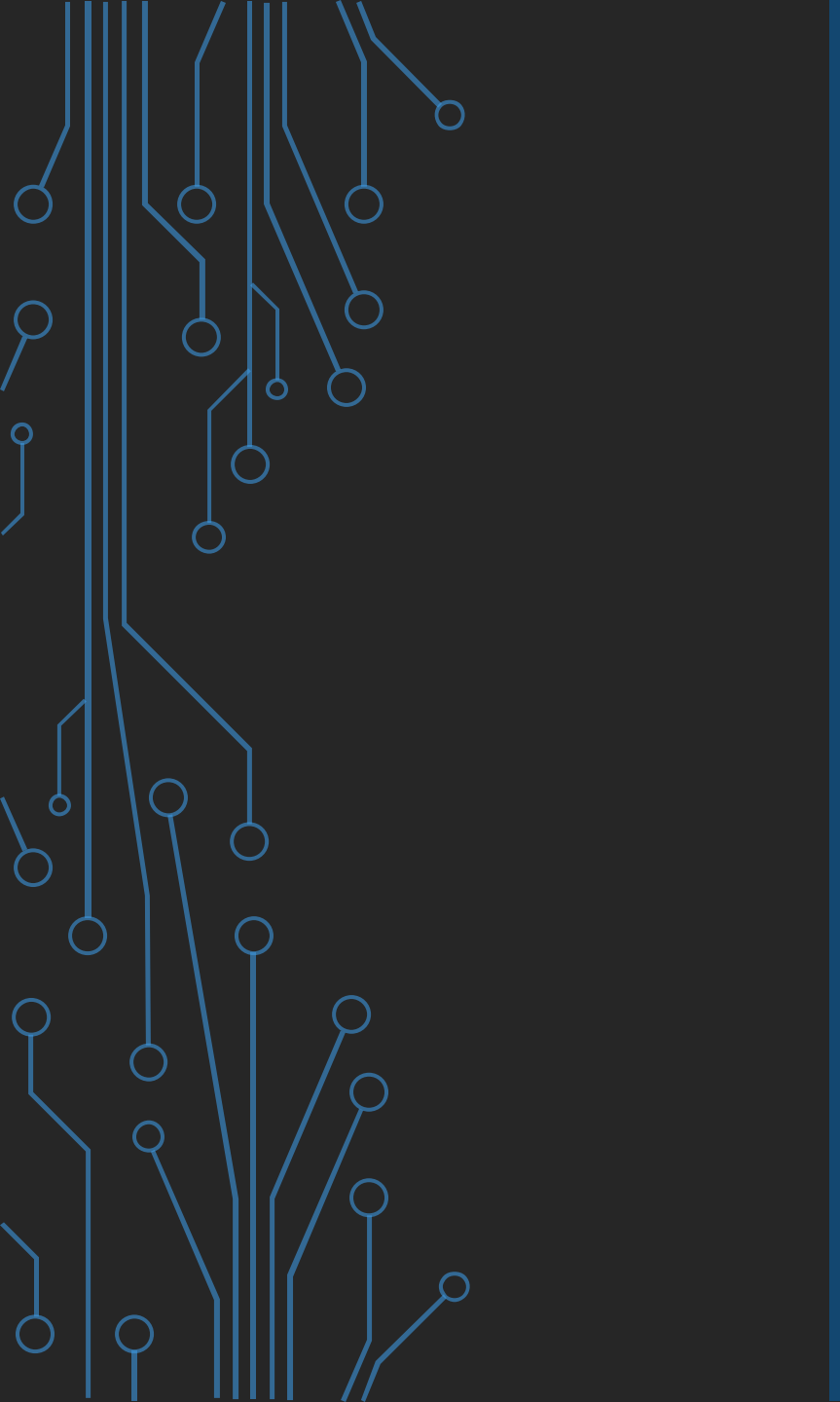
ADVANTAGES

- Effective in high-dimensional spaces and with small to medium-sized datasets.
- Works well in cases where classes are not linearly separable through the use of kernel functions.
- Tends to generalize well and avoid overfitting, especially when the regularization parameter C is properly tuned.

LIMITATIONS

- SVMs can be computationally intensive, especially for large datasets.
- Choosing the appropriate kernel and hyperparameters requires careful tuning and experimentation.
- Interpreting the model's decisions can be challenging, especially in high-dimensional spaces.
- SVMs may not perform well when dealing with noisy data or data with a high degree of overlap between classes.

TIME FOR PRACTICALITY (5 MINUTES)



EVALUATION METRICS

EVALUATION METRICS

Each Machine Learning Problem has it is own Evaluation Metrics

CLASSIFICATION EVALUATION METRICS

- **Evaluation metrics** for classification are used to assess the performance of a classification model by comparing its predictions to the actual class labels in a dataset.

- These metrics help quantify how well the model is performing and provide insights into its strengths and weaknesses.

Accuracy	Precision	Recall	F1-Score	ROC	AUC

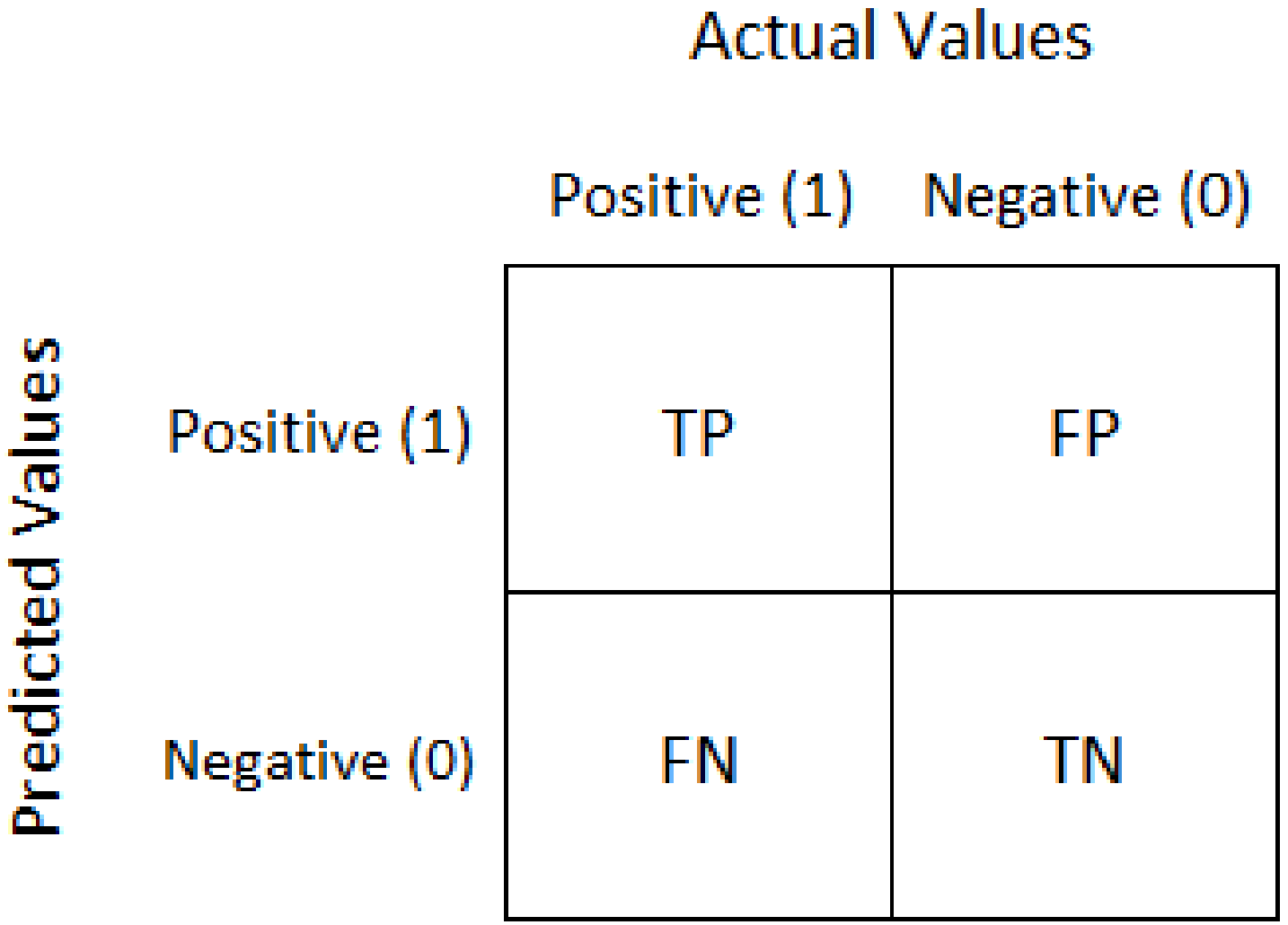
ACCURACY

Accuracy: Accuracy measures the proportion of correctly predicted instances out of the total instances in the dataset. It's a simple and intuitive metric but may not be suitable for imbalanced datasets.

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

Confusion Matrix: A confusion matrix provides a detailed breakdown of true positive, true negative, false positive, and false negative predictions. It's a useful tool for understanding the distribution of prediction errors.

CONFUSION MATRIX


Actual Values
Positive (1)
Negative (0)
Predicted Values
Positive (1)
Negative (0)
TP
FP
FN
TN

Precision: Precision measures the proportion of true positive predictions (correctly predicted positives) out of all predicted positives. It focuses on the correctness of positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Used for Medical Diagnosis

PRECISION

Recall (Sensitivity or True Positive Rate): Recall measures the proportion of true positive predictions out of all actual positives. It focuses on the completeness of positive predictions.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Used for information Retreiving

RECALL

F1-SCORE

F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is especially useful when class imbalance is present.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

SPECIFICITY

Specificity (True Negative Rate): Specificity measures the proportion of true negative predictions out of all actual negatives. It's the complement of the false positive rate.

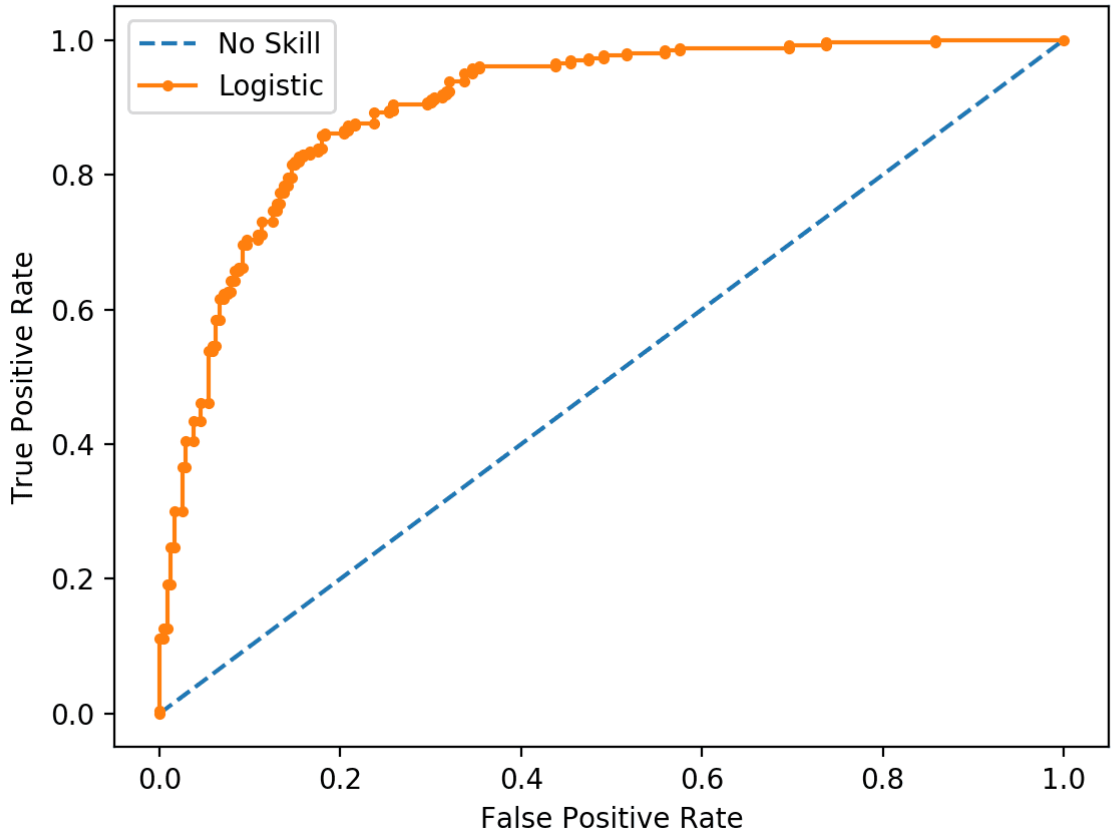
$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

RECEIVER OPERATION CHARACTERISTIC (ROC)

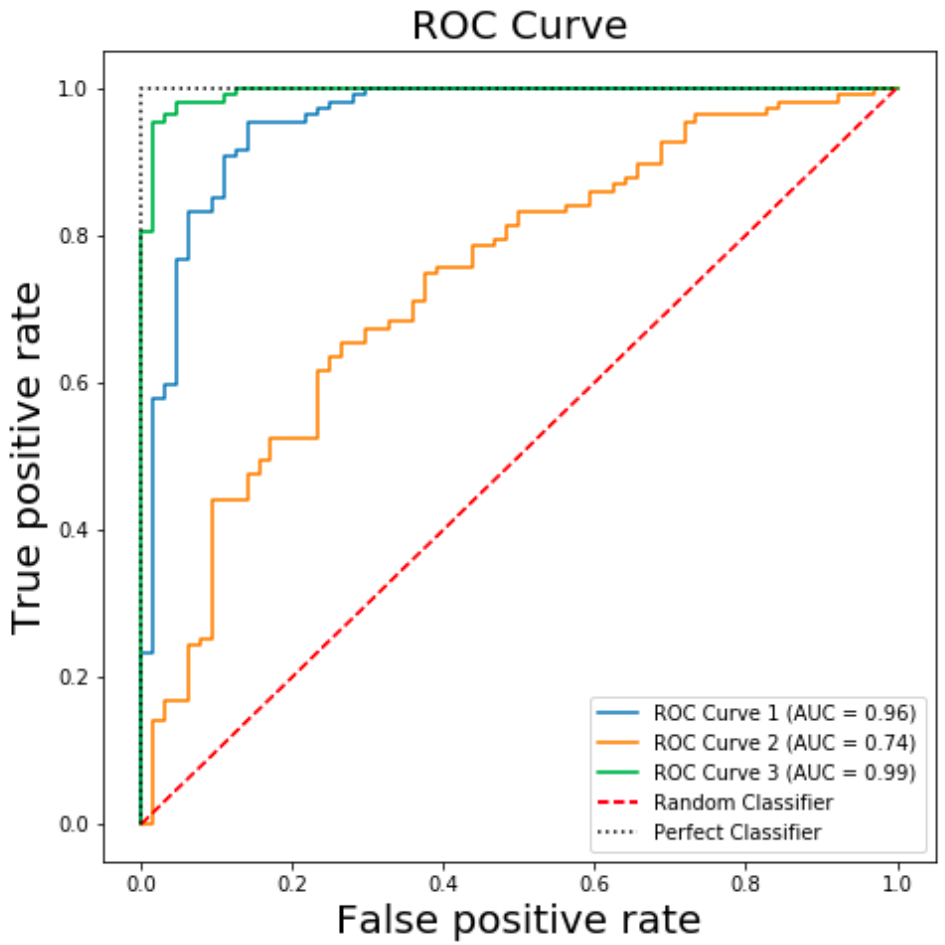
- **ROC curve** is a graphical representation of the true positive rate (recall) versus the false positive rate (1 – Specificity) for different threshold values.

- **The area under the ROC curve (AUC-ROC)** is a common metric that quantifies the overall performance of a classifier.

AUC-ROC CURVE



AUC-ROC CURVE



TIME FOR PRACTICALITY (5 MINUTES)

QUESTIONS


THANK YOU