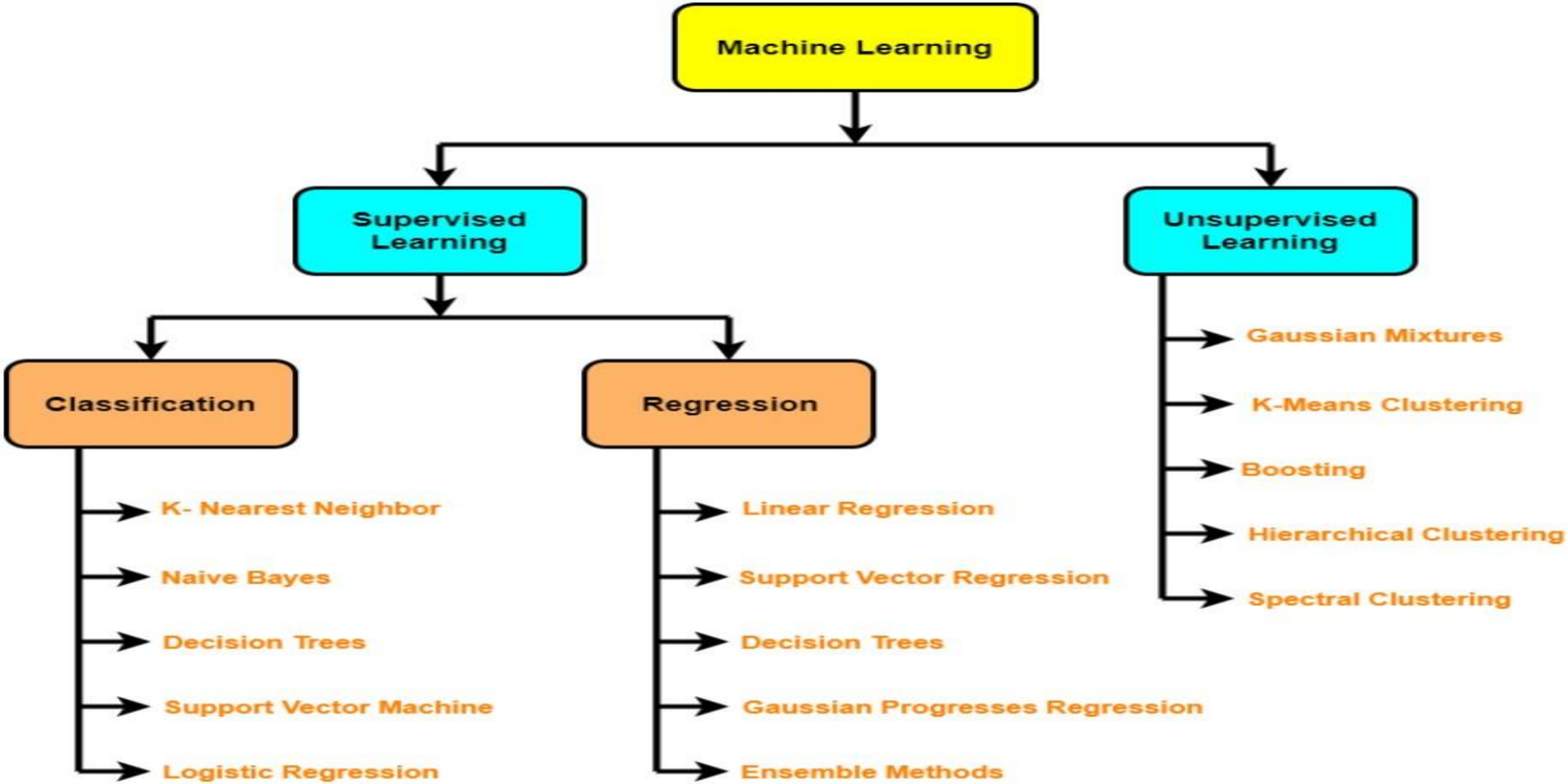
MODULE 2

Lecture	Topics
11	Classification, Logistic Regression - 1 (binary)
12	Logistic Regression - 2 (binary)
13	Nearest neighbour and K Nearest Neighbour
14	Error Analysis - Train/Test Split, validation set, Accuracy, Precision, Recall, F-measure, ROC curve, Confusion Matrix
15	Naive Bayes Classifier - 1
16	Naive Bayes Classifier - 2

17	Decision Tree: Introduction, Id3 Algorithm - 1
18	Decision Tree - Id3 Algorithm - 2
19	Decision Tree - Problem of Overfitting, Pre-pruning/post-pruning Decision Tree, Examples.
20	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 1
21	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 2
22	Support Vector Machine - KKT Condition - 3
23	Support Vector Machine - Kernel, Nonlinear Classification, and

25	Principal Component Analysis - Steps, merits, demerits, Intuition - 1
26	Principal Component Analysis - Steps, merits, demerits, Intuition - 2
27	Understanding and Implementing PCA using SVD for dimensionality reduction


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K- Nearest Neighbor
Naive Bayes
Decision Trees
Support Vector Machine
Logistic Regression
Linear Regression
Support Vector Regression
Decision Trees
Gaussian Progresses Regression
Ensemble Methods
Gaussian Mixtures
K-Means Clustering
Boosting
Hierarchical Clustering
Spectral Clustering

K-Nearest Neighbor(KNN) Algorithm

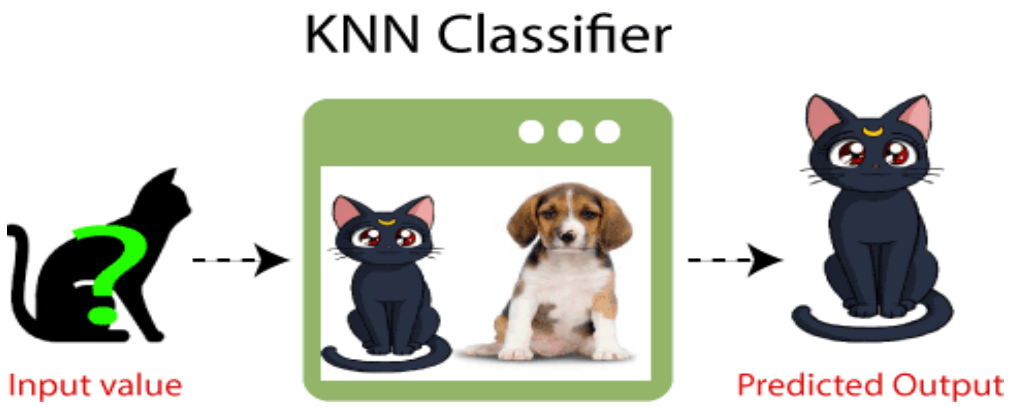
K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

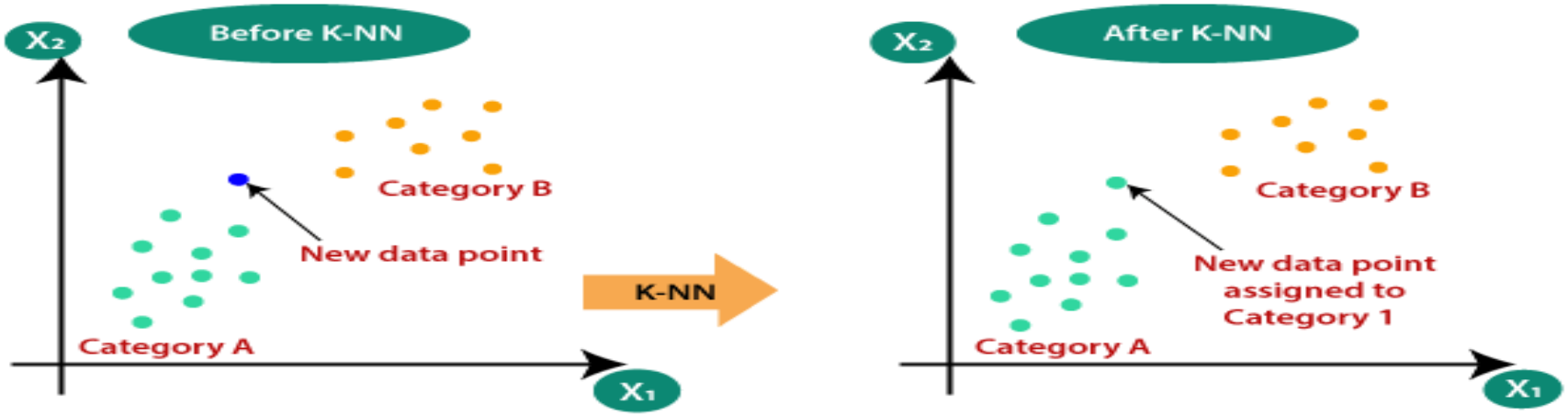
Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



K-Nearest Neighbor(KNN) Algorithm for Machine Learning

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



K-Nearest Neighbor(KNN) Algorithm for Machine Learning

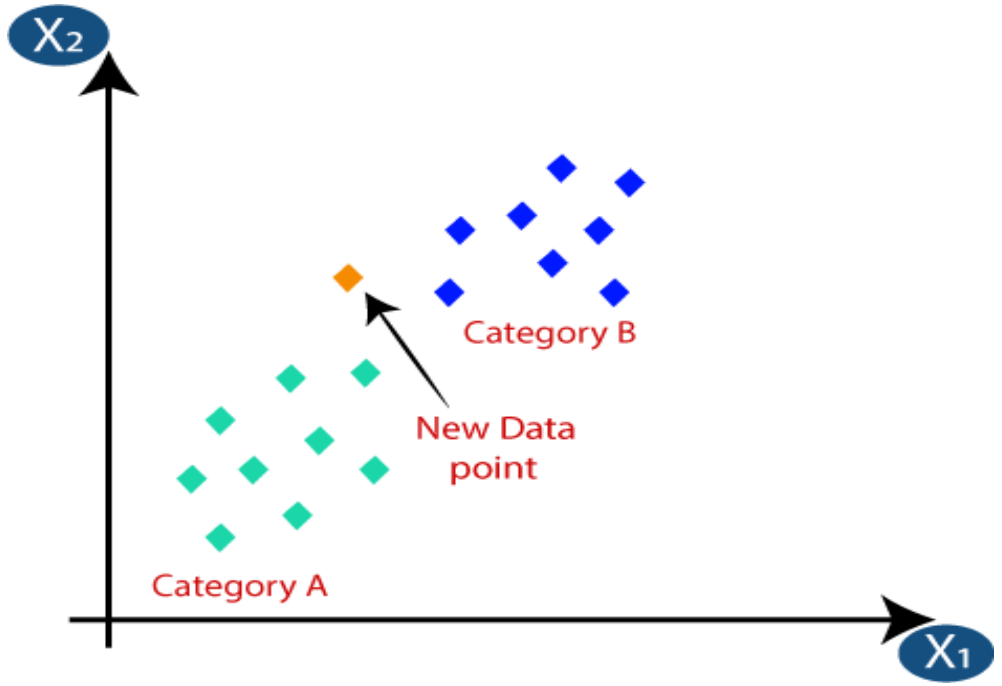
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

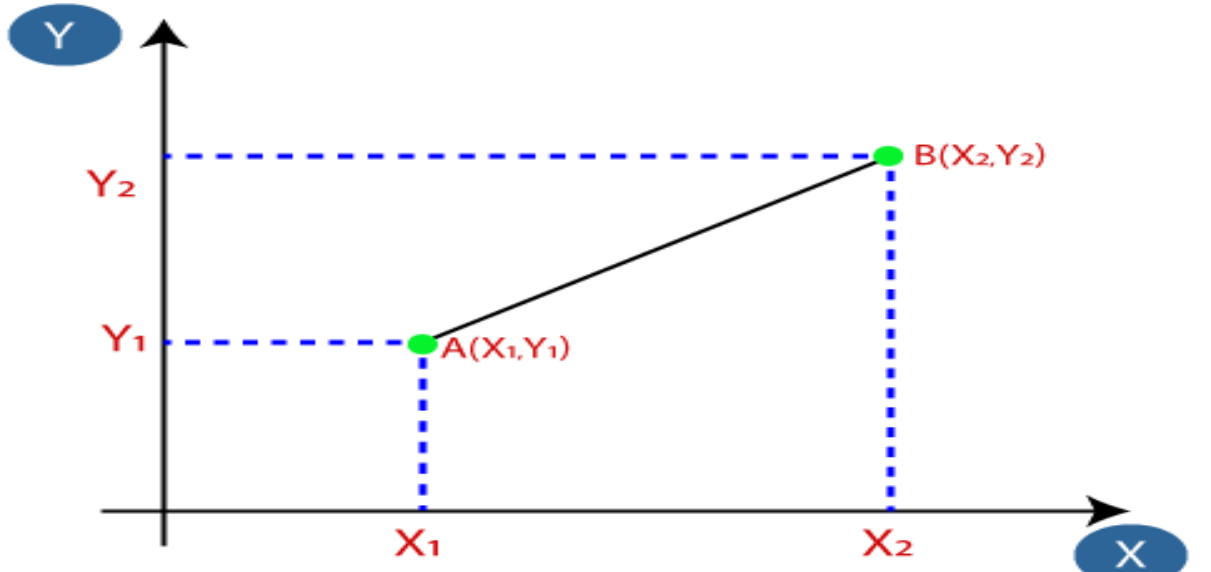
K-Nearest Neighbor(KNN) Algorithm for Machine Learning

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



K-Nearest Neighbor(KNN) Algorithm for Machine Learning

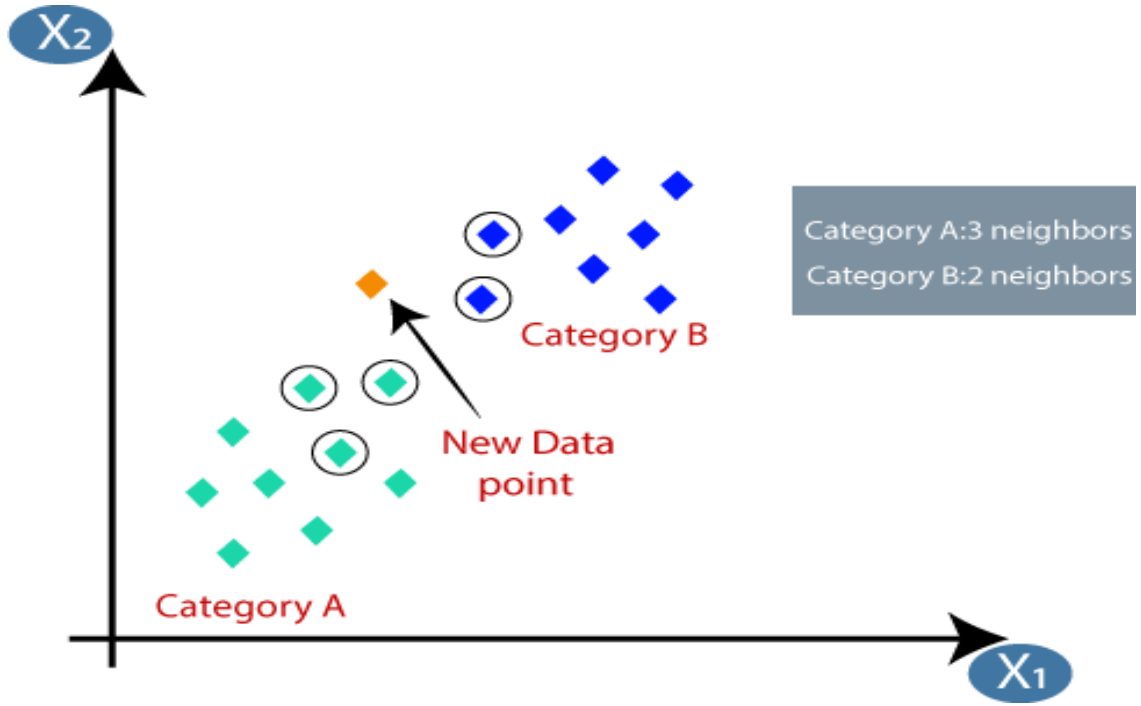
- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



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

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



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

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties. Advantages of KNN

Algorithm:

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

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

KNN Algorithm e.g.

Name	Age	Gender	sport
Ajay	32	M	Football
Mark	40	M	Neither
sara	16	F	Cricket
Zaira	34	F	Cricket
Sachin	55	M	Neither
Rahul	40	M	Cricket
Pooja	20	F	Neither
Smith	15	M	Cricket
Laxmi	55	F	Football
Michael	15	M	Football

KNN Algorithm e.g.

Name	Age	Gender	sport
Ajay	32	M	Football
Mark	40	M	Neither
sara	16	F	Cricket
Zaira	34	F	Cricket
Sachin	55	M	Neither
Rahul	40	M	Cricket
Pooja	20	F	Neither
Smith	15	M	Cricket
Laxmi	55	F	Football
Michael	15	M	Football
Angelina	5	F	?

male = 0

Female = 1

The distance equation is normally,

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

ie Euclidean Distance

(other distances → Manhattan distance
→ minkowski distance
etc.

male = 0

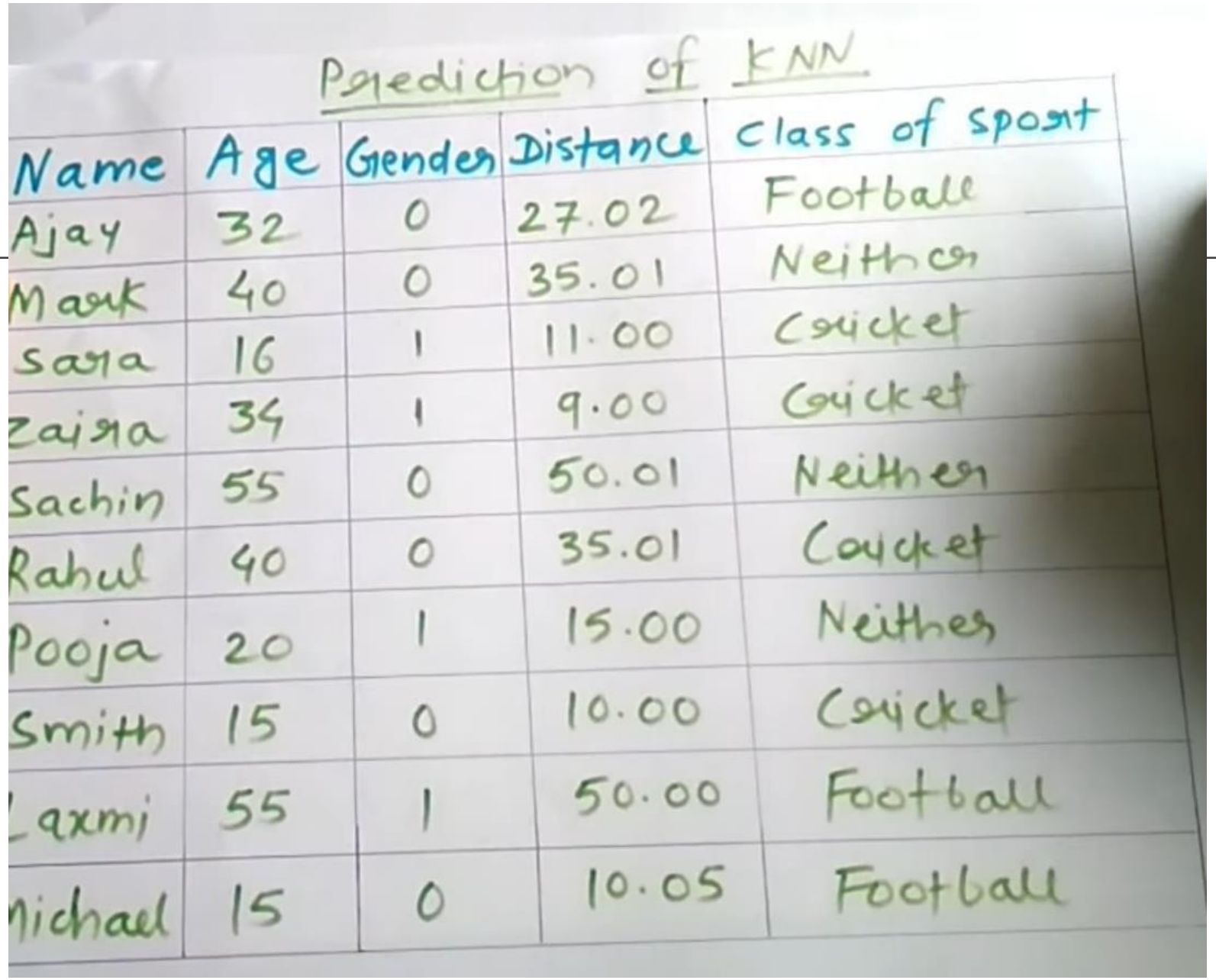
Female = 1

Ajay male = 0 Age = 32

$$= \sqrt{(5-32)^2 + (1-0)^2}$$

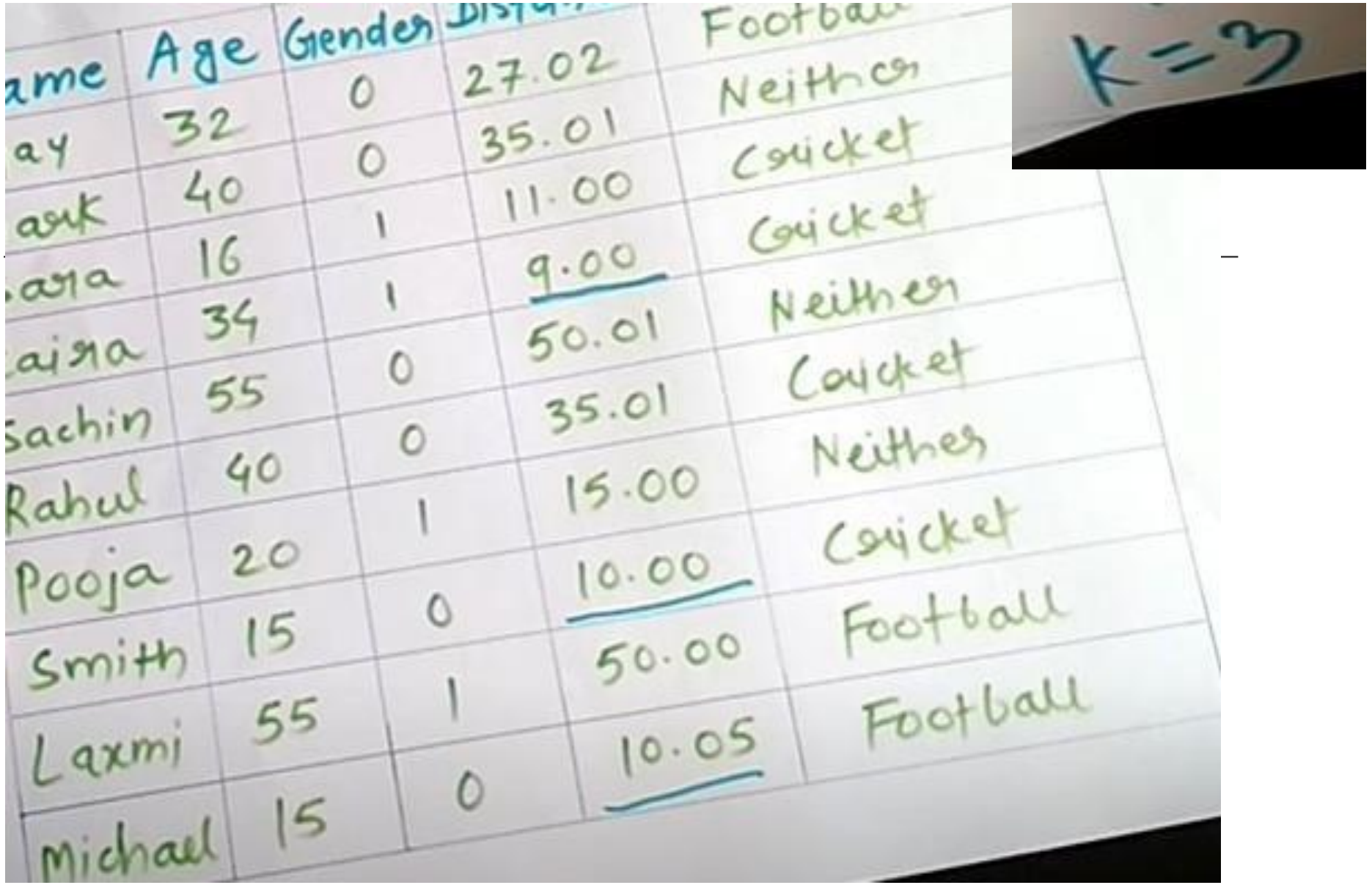
$$= \sqrt{729 + 1}$$

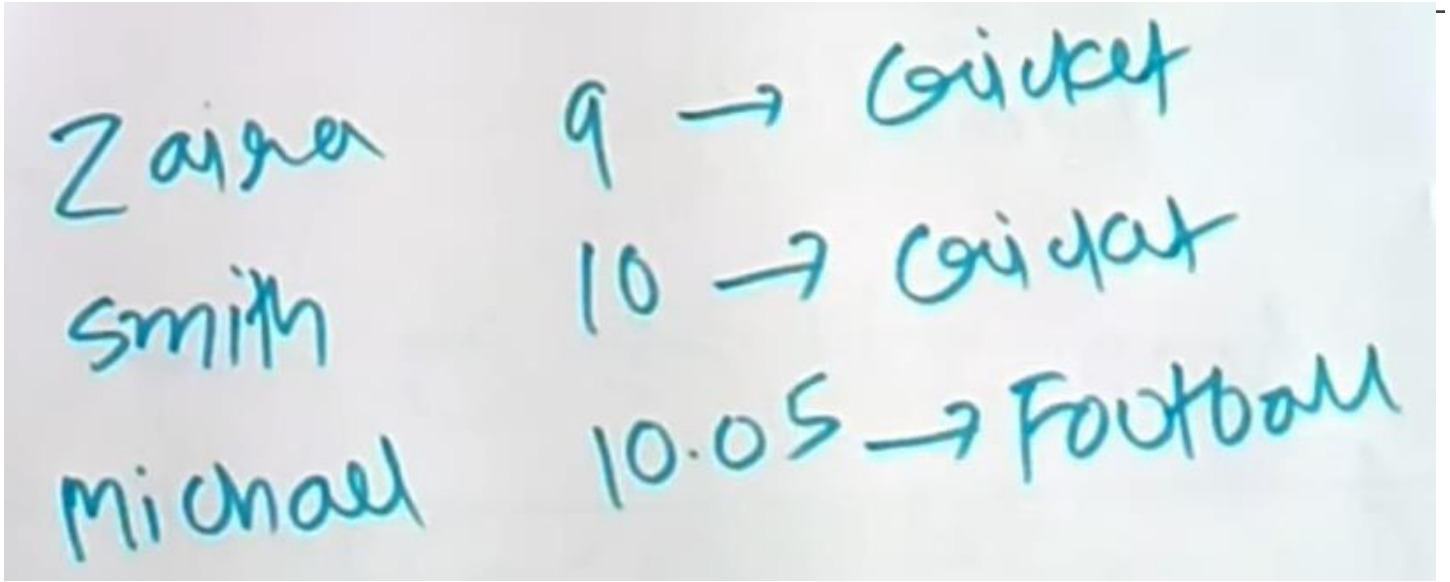
$$= 27.02$$

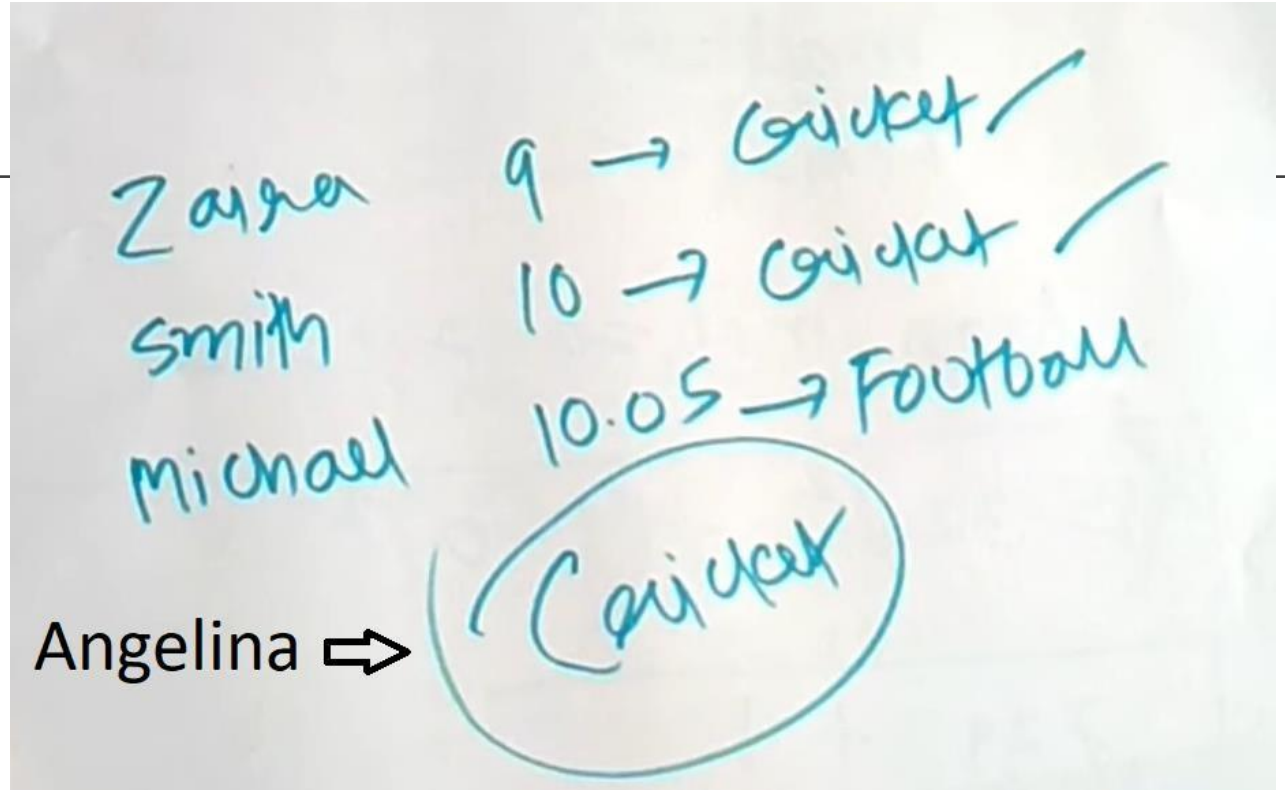


Prediction of KNN

Name	Age	Gender	Distance	Class of sport
Ajay	32	0	27.02	Football
Mark	40	0	35.01	Neither
sara	16	1	11.00	Cricket
Zaira	34	1	9.00	Cricket
Sachin	55	0	50.01	Neither
Rahul	40	0	35.01	Cricket
Pooja	20	1	15.00	Neither
Smith	15	0	10.00	Cricket
Laxmi	55	1	50.00	Football
Michael	15	0	10.05	Football


K=3
ame Age Gender
ay 32 0 27.02 Footba
ark 40 0 35.01 Neither
ara 16 1 11.00 Cricket
aira 34 1 9.00 Cricket
Sachin 55 0 50.01 Neither
Rahul 40 0 35.01 Cricket
Pooja 20 1 15.00 Neither
Smith 15 0 10.00 Cricket
Laxmi 55 1 50.00 Football
Michael 15 0 10.05 Football


Zaira 9 → Cricket
Smith 10 → Cricket
Michael 10.05 → Football


Zayra 9 → Cricket
smith 10 → Cricket
Michael 10.05 → Football
Angelina ⇨
Cricket

KNN Question

A dataset contains information about houses and their prices. Each house is described using **4 features**:

Training Data:

House	Size (sqft)	Bedrooms	Age (years)	Distance (km)	Price
A	1200	2	10	5	L
B	1500	3	5	3	H
C	800	2	20	10	L
D	1800	4	3	2	H
E	1000	3	15	7	L

A new house has the following features:

- Size = 1300 sqft
- Bedrooms = 3
- Age = 8 years
- Distance = 4 km

1. Using the K-Nearest Neighbors (KNN) algorithm with
- K = 3

Predict whether the new house belongs to **High (H)** or **Low (L)** price category.

Distance Calculations (Euclidean)

A (L): (1200,2,10,5)

$$\sqrt{(100)^2 + (1)^2 + (-2)^2 + (-1)^2} = \sqrt{10000 + 1 + 4 + 1} = \sqrt{10006} \approx 100.03$$

B (H): (1500,3,5,3)

$$\sqrt{(-200)^2 + 0^2 + 3^2 + 1^2} = \sqrt{40000 + 0 + 9 + 1} = \sqrt{40010} \approx 200.02$$

C (L): (800,2,20,10)

$$\sqrt{500^2 + 1^2 + (-12)^2 + (-6)^2} = \sqrt{250000 + 1 + 144 + 36} = \sqrt{250181} \approx 500.18$$

D (H): (1800,4,3,2)

$$\sqrt{(-500)^2 + (-1)^2 + 5^2 + 2^2} = \sqrt{250000 + 1 + 25 + 4} = \sqrt{250030} \approx 500.03$$

E (L): (1000,3,15,7)

$$\sqrt{300^2 + 0^2 + (-7)^2 + (-3)^2} = \sqrt{90000 + 0 + 49 + 9} = \sqrt{90058} \approx 300.10$$

Nearest 3 Neighbors

House	Distance	Class
A	100.03	L
B	200.02	H
E	300.10	L

Majority Voting

L, H, L → **2 L vs 1 H**

Final Answer

Predicted class = Low (L)

You are a business analyst at a retail company and want to predict the monthly sales revenue of new stores based on their size (in square feet) and the number of employees.

Dataset:

Suppose you have the following training dataset:

Store Size (sq ft)	Number of Employees	Monthly Sales Revenue ($)
1500	5	30000
2000	7	50000
2500	10	70000
3000	15	80000
3500	20	100000

Goal:

Predict the monthly sales revenue for a new store with a size of 2800 sq ft and 12 employees.

Calculate Distances:

- **For Store 1 (1500 sq ft, 5 employees):** Distance = sqrt((1500 - 2800)² + (5 - 12)²)
  Distance = sqrt((1300)² + (-7)²)
  Distance = sqrt(1690000 + 49)
  Distance ≈ 1300.02
- **For Store 2 (2000 sq ft, 7 employees):** Distance = sqrt((2000 - 2800)² + (7 - 12)²)
  Distance = sqrt((800)² + (-5)²)
  Distance = sqrt(640000 + 25)
  Distance ≈ 800.02
- **For Store 3 (2500 sq ft, 10 employees):** Distance = sqrt((2500 - 2800)² + (10 - 12)²)
  Distance = sqrt((-300)² + (-2)²)
  Distance = sqrt(90000 + 4)
  Distance ≈ 300.01

- **For Store 4 (3000 sq ft, 15 employees):** Distance = sqrt((3000 - 2800)² + (15 - 12)²)
  Distance = sqrt((200)² + (3)²)
  Distance = sqrt(40000 + 9)
  Distance ≈ 200.02
- **For Store 5 (3500 sq ft, 20 employees):** Distance = sqrt((3500 - 2800)² + (20 - 12)²)
  Distance = sqrt((700)² + (8)²)
  Distance = sqrt(490000 + 64)
  Distance ≈ 700.04

Step 1: Data Preparation

Prepare your data using the two features (store size and number of employees) to find the K nearest neighbors for the new store.

Step 2: Choosing K

Choose k = 3. This means you will consider the 3 closest neighbors to make your prediction.

Step 3: Distance Calculation

Calculate the Euclidean distance between the new store and each store in the dataset using the formula:

$$\text{Distance} = \text{sqrt}((X1 - X2)^2 + (Y1 - Y2)^2)$$

Where:

- X1 and Y1 are the features of the new store (size and employees).
- X2 and Y2 are the features of each training store.

Step 4: Finding Neighbors

Now, we have calculated the distances for each store. Here are the distances we obtained:

- Store 1: 1300.02
- Store 2: 800.02
- Store 3: 300.01
- Store 4: 200.02
- Store 5: 700.04

Now, we will select the three closest neighbors (smallest distances):

1. Store 4 (200.02)
2. Store 3 (300.01)
3. Store 5 (700.04)

Step 5: Target Value Prediction

Next, we will take the monthly sales revenues of these three nearest neighbors and calculate their average.

- Store 4 Revenue: $80,000
- Store 3 Revenue: $70,000
- Store 5 Revenue: $100,000

Step 5: Target Value Prediction

Next, we will take the monthly sales revenues of these three nearest neighbors and calculate their average.

- Store 4 Revenue: $80,000
- Store 3 Revenue: $70,000
- Store 5 Revenue: $100,000

Average Revenue = (80000 + 70000 + 100000) / 3
Average Revenue = 250000 / 3
Average Revenue ≈ 83333.33

The predicted monthly sales revenue for the new store (2800 sq ft and 12 employees) is approximately **$83,333.33**.

Common Distance Measures in KNN

1. Manhattan Distance (L1)

$$D = \sum |x_i - y_i|$$

Moves like city blocks (up–down, left–right).

Example:

Points (2,3) and (5,7)

$$|2 - 5| + |3 - 7| = 3 + 4 = 7$$

2. Euclidean Distance (L2)

$$D = \sqrt{\sum (x_i - y_i)^2}$$

Straight-line distance.

Example:

$$\sqrt{(3^2 + 4^2)} = 5$$

3. Minkowski Distance (General Form)

$$D = \left(\sum |x_i - y_i|^p \right)^{1/p}$$

- p = 1 → Manhattan
- p = 2 → Euclidean

4. Chebyshev Distance

$$D = \max(|x_i - y_i|)$$

Only the largest difference matters.

Example:

(2,5) and (6,9) → max(4,4) = 4

5. Hamming Distance

Used for categorical/binary data.

Counts mismatches.

Example:

10101 and 10011 → 2 differences

6. Cosine Distance

$$D = 1 - \frac{A \cdot B}{|A||B|}$$

Measures angle, not magnitude.

Used in text data.

7. Jaccard Distance

$$D = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Used for sets/binary features.

Distance Type	Best For
Euclidean	Continuous numeric data
Manhattan	Grid-like movement
Minkowski	General purpose
Chebyshev	Max difference matters
Hamming	Binary/categorical data
Cosine	Text / high-dimensional
Jaccard	Set/binary features

Curse of Dimensionality

The **curse of dimensionality** refers to the problems that arise when working with data that has **too many features (dimensions)**.

As the number of features increases, the data becomes:

- Very **sparse**
- Harder to analyze
- Less meaningful for distance-based methods like KNN, K-means, etc.

Effect on Machine Learning

- Distance-based models (KNN, clustering) perform poorly
- Models overfit easily
- Training becomes slow and expensive
- Visualization becomes difficult

How to Handle It

- Feature selection (remove unnecessary features)
- Dimensionality reduction (PCA, LDA, Autoencoders)
- Regularization
- Collect more data if possible

The curse of dimensionality means that as the number of features increases, data becomes sparse, distances become meaningless, and learning becomes difficult without a very large amount of data.

Simple Example

Imagine you are placing points in space.

Case 1: 1 Feature (1D)

Suppose feature = Height

Range = 0 to 10

To cover this line well, you may need only **10 points**.

```lua
0----1----2----3----4----5----6----7----8----9----10
```

Points are close and easy to compare.

Case 2: 2 Features (2D)

Features = Height and Weight

Range = 0 to 10 for both

Now space is a square:

Area = 10 × 10 = 100

To cover this space well, you may need about **100 points**.

Case 3: 3 Features (3D)

Features = Height, Weight, Age

Range = 0 to 10

Now space is a cube:

Volume = 10 × 10 × 10 = 1000

You now need about **1000 points**.

Case 4: 10 Features

Range = 0 to 10 for each feature

Space size = 10^{10}

You would need **10,000,000,000 points** to cover the space well — which is impossible in practice.

Applications of K-Nearest Neighbors (KNN)

KNN is a simple, instance-based learning algorithm mainly used for classification and regression.

1. Pattern Recognition

- Handwritten digit recognition
- Face recognition
- Signature verification

2. Recommendation Systems

- Suggesting movies, songs, or products based on similar users
- E-commerce product recommendations

3. Medical Diagnosis

- Disease classification based on patient symptoms
- Cancer detection using medical data

4. Image and Video Analysis

- Image classification
- Object recognition
- Color quantization

5. Text and Document Classification

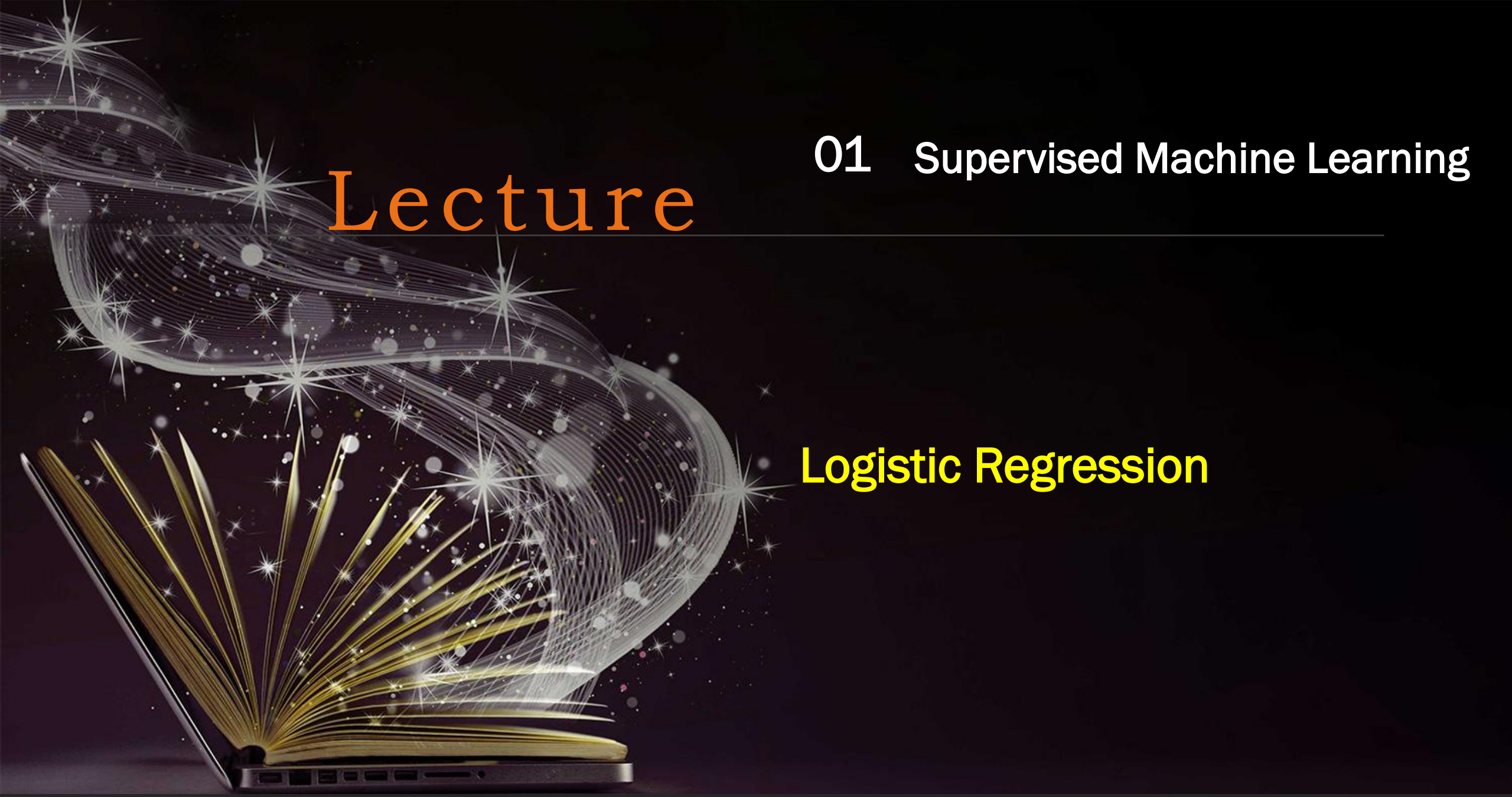
- Spam detection
- News article classification
- Language detection

6. Finance

- Credit scoring
- Risk assessment
- Fraud detection

7. Anomaly Detection

- Detecting unusual patterns in network traffic
- Identifying outliers in data

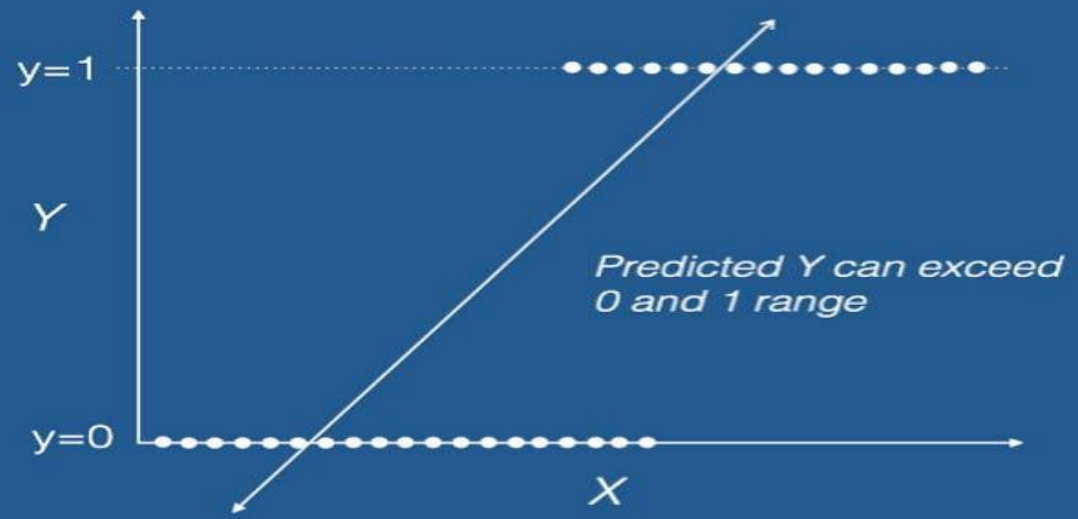


Lecture

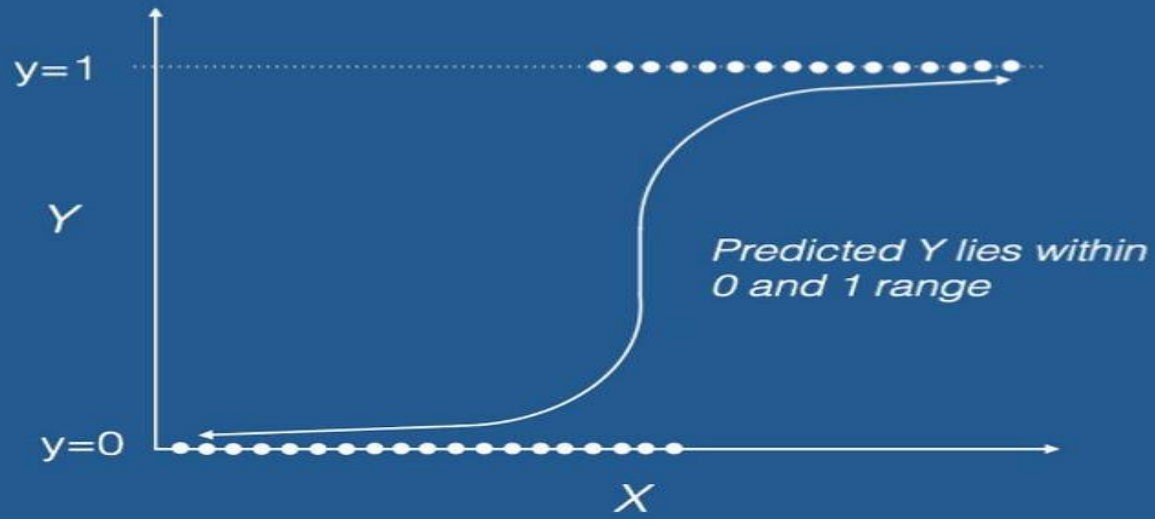
01 Supervised Machine Learning

Logistic Regression

Linear Regression



Logistic Regression



Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

Logistic regression analysis is used to examine the association of (categorical or continuous) independent variable(s) with one dichotomous dependent variable. This is in contrast to linear regression analysis in which the dependent variable is a continuous variable.

Definition of the logistic function [edit]

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a sigmoid function, which takes any real input t, and outputs a value between zero and one.[2] For the logit, this is interpreted as taking input log-odds and having output probability. The *standard* logistic function $\sigma : \mathbb{R} \to (0,1)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

A graph of the logistic function on the *t*-interval (−6,6) is shown in Figure 1.

Let us assume that t is a linear function of a single explanatory variable x (the case where t is a *linear combination* of multiple explanatory variables is treated similarly). We can then express t as follows:

$$t = \beta_0 + \beta_1 x$$

And the general logistic function $p : \mathbb{R} \to (0,1)$ can now be written as:

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

In the logistic model, $p(x)$ is interpreted as the probability of the dependent variable Y equaling a success/case rather than a failure/non-case. It is clear that the response variables Y_i are not identically distributed: $P(Y_i = 1 \mid X)$ differs from one data point X_i to another, though they are independent given design matrix X and shared parameters β.[11]

A company wants to predict whether a customer will buy a product (1 = Buy, 0 = Not Buy) based on one feature:

$x =$ number of hours spent on the website.

The logistic regression model is:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Given:

$\beta_0 = -4$

$\beta_1 = 1.2$

Question:

1. If a customer spends $x = 3$ hours on the website, calculate the probability that the customer will buy the product.
2. Based on a threshold of 0.5, predict whether the customer will buy or not.

Given model:

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

$\beta_0 = -4,\ \beta_1 = 1.2,\ x = 3$

Step 1: Compute the linear term

$$z = \beta_0 + \beta_1 x = -4 + (1.2 \times 3) = -4 + 3.6 = -0.4$$

Step 2: Apply logistic function

$$P = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{0.4}}$$

$$e^{0.4} \approx 1.4918$$

$$P = \frac{1}{1+1.4918} = \frac{1}{2.4918} \approx 0.401$$

Step 3: Prediction using threshold 0.5

$$0.401 < 0.5 \Rightarrow \text{Predict: Not Buy } (0)$$

Logistic Regression Problem Solution

- The dataset of pass or fail in an exam of 5 students is given in the table.
- Use logistic regression as classifier to answer the following questions.

1. Calculate the probability of pass for the student who studied 33 hours.

2. At least how many hours student should study that makes he will pass the course with the probability of more than 95%.

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

Assume the model suggested by the optimizer for odds of passing the course is,

$$\log(odds) = -64 + 2 * hours$$

In logistic regression, log-odds refers to the logarithm of the odds of an event occurring. The odds of an event are the ratio of the probability of the event occurring to the probability of it not occurring.

$$\log(odds) = -64 + 2 * hours$$

Odds and Log-Odds

Given a probability p of an event occurring:

$$\text{Odds} = \frac{p}{1-p}$$

The **log-odds** (also called the **logit**) is the natural logarithm of the odds:

$$\text{Log-Odds} = \log\left(\frac{p}{1-p}\right)$$

Logistic Regression and Log-Odds

In logistic regression, the relationship between the independent variables (predictors) and the dependent variable (binary outcome) is modeled in terms of log-odds. The logistic regression model can be written as:

$$\text{Log-Odds} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

where:

- β_0 is the intercept,
- $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients of the independent variables $x_1, x_2, \ldots, x_k$.

The model expresses the log-odds of the dependent variable being 1 (the event of interest) as a linear combination of the predictors. The coefficients represent the change in log-odds for a one-unit change in the corresponding predictor.

Converting Log-Odds to Probability

The logistic function (or sigmoid function) can be used to convert log-odds back to the probability p:

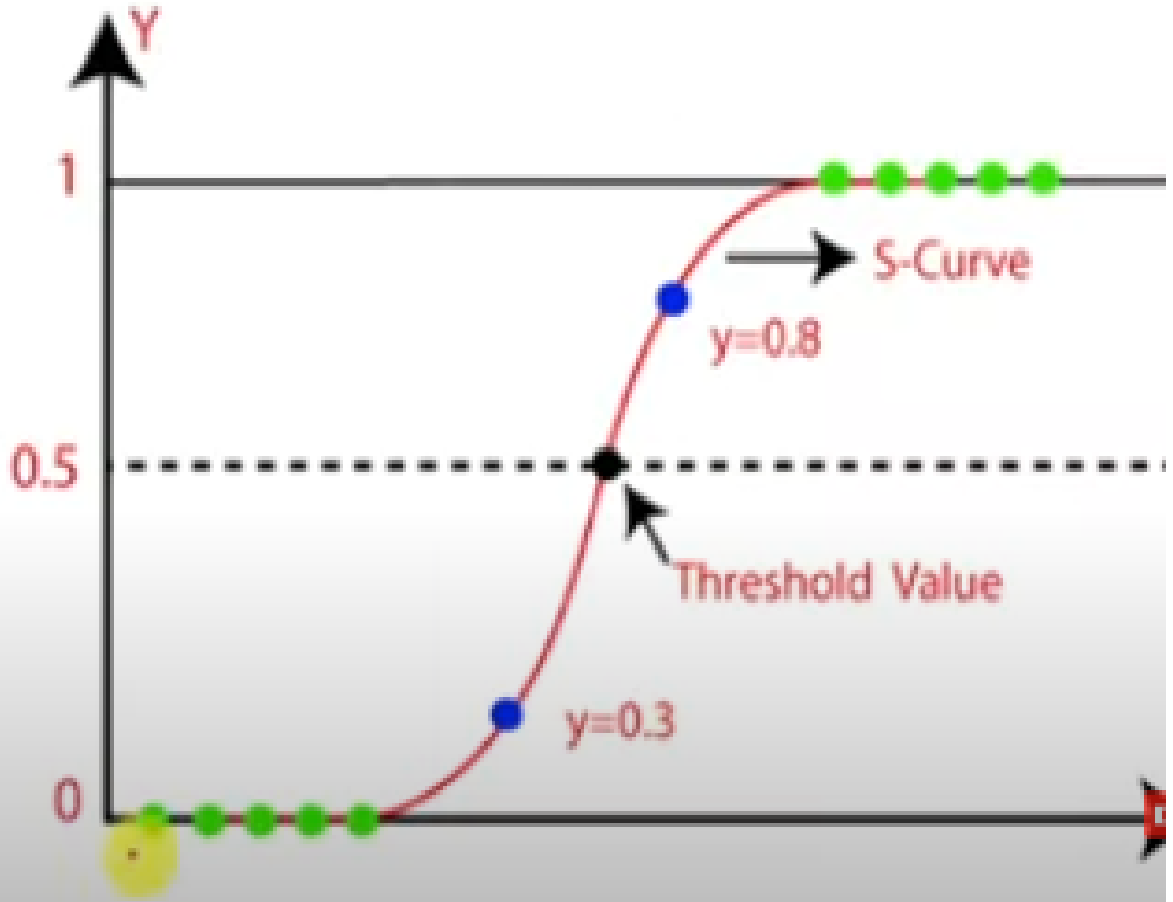
$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}}$$

This function ensures that the predicted probabilities lie between 0 and 1.

In summary, in logistic regression, the log-odds are the core concept for modeling the relationship between predictors and the binary outcome.

- We use Sigmoid Function in logistic regression

- $$s(x) = \frac{1}{1+e^{-x}}$$



1. Calculate the probability of pass for the student who studied 33 hours.

- $p = \frac{1}{1+e^{-z}}$ $\qquad s(x) = \frac{1}{1+e^{-x}}$
- $z = -64 + 2 * 33 = -64 + 66 = 2$
- $p = \frac{1}{1+e^{-2}} = 0.88$

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

$$\log(odds) = z = -64 + 2 * hours$$

That is, if student studies 33 hours, then there is **88% chance** that the student will pass the exam

2. At least how many hours student should study that makes he will pass the course with the probability of more than 95%.

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

- $\boldsymbol{p = \frac{1}{1+e^{-z}} = 0.95}$
- $0.95 * (1 + e^{-z}) = 1$
- $0.95 * e^{-z} = 1 - 0.95$
- $e^{-z} = \frac{0.05}{0.95} = 0.0526$
- $\ln(e^{-z}) = \ln(0.0526)$

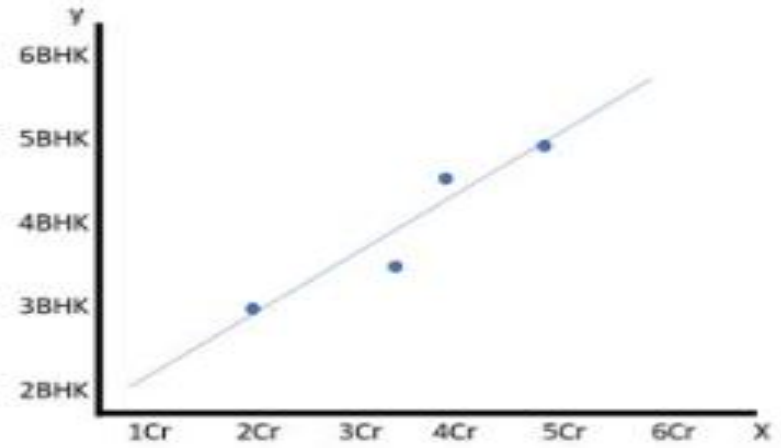
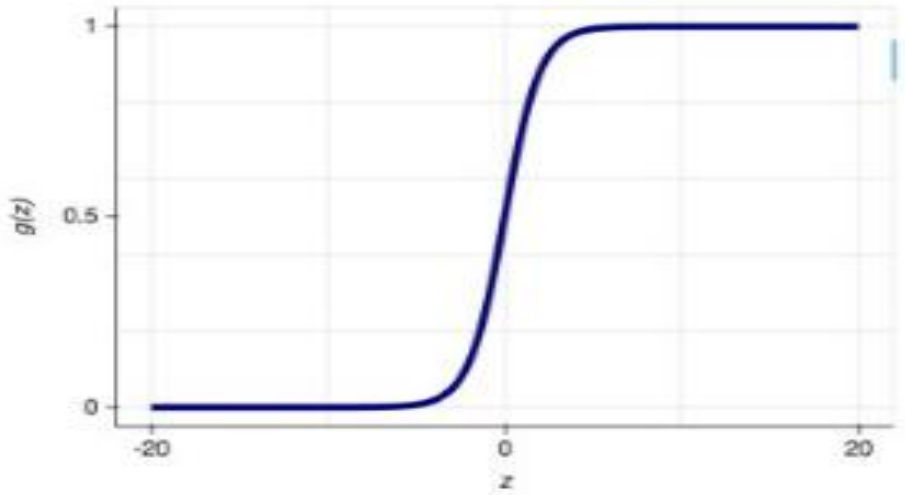
$\boldsymbol{\ln(e^x) = x}$

$-z = ln(0.0526) = -2.94$

- $z = 2.94$
- $\log(odds) = z = -64 + 2 * hours$
- $2.94 = -64 + 2 * hours$
- $2 * hours = 2.94 + 64$
- $2 * hours = 66.94$
- $hours = \frac{66.94}{2}$
- $\boldsymbol{hours = 33.47\ Hours}$

Hours Study	Pass (1) / Fail (0)
29	0
15	0
33	1
28	1
39	1

- The student should study **at least 33.47 hours**, so that he will pass the exam with more than 95% probability

Linear Regression	Logistic Regression
Target is an interval variable	Target is discrete (binary or ordinal) variable
Predicted values are the mean of the target variable at the given values of the input variable	Predicted values are the probability of the particular levels of the given values of the input variable
Solve regression problems	Solve classification problems
Example : What is the Temperature?	Example : Will it rain or not?
Graph is straight line	Graph is S-curve
	

Use Logistic Regression

- The student dataset has entrance mark based on the historic data of those who are selected or not selected.
- Based on the logistic regression, the values of the learnt parameters are $\beta_0 = 1$ and $\beta_1 = 8$.
- Assuming marks of x = 60, compute the resultant class.

If we assume the threshold value as 0.5, then it is observed that 0.44 < 0.5, therefore, the candidate with marks 60 is not selected.

answer

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\beta_0 + \beta_1 x = 481$$

$$p(x) = \frac{1}{1 + e^{-481}} = 0.44$$

Applications of Logistic Regression

Logistic Regression is mainly used for binary and multiclass classification problems.

1. Healthcare

- Disease prediction (diabetes, heart disease)
- Patient risk assessment

2. Marketing

- Customer churn prediction
- Purchase prediction
- Campaign response analysis

3. Finance

- Loan approval prediction
- Credit risk modeling
- Fraud detection

4. Education

- Student performance prediction
- Dropout prediction

5. Social Media and Web Analytics

- Click-through rate (CTR) prediction
- Ad conversion prediction

6. Cybersecurity

- Spam filtering
- Intrusion detection

7. Human Resources

- Employee attrition prediction
- Resume screening

8. Manufacturing

- Quality control (defective vs non-defective)
- Predictive maintenance

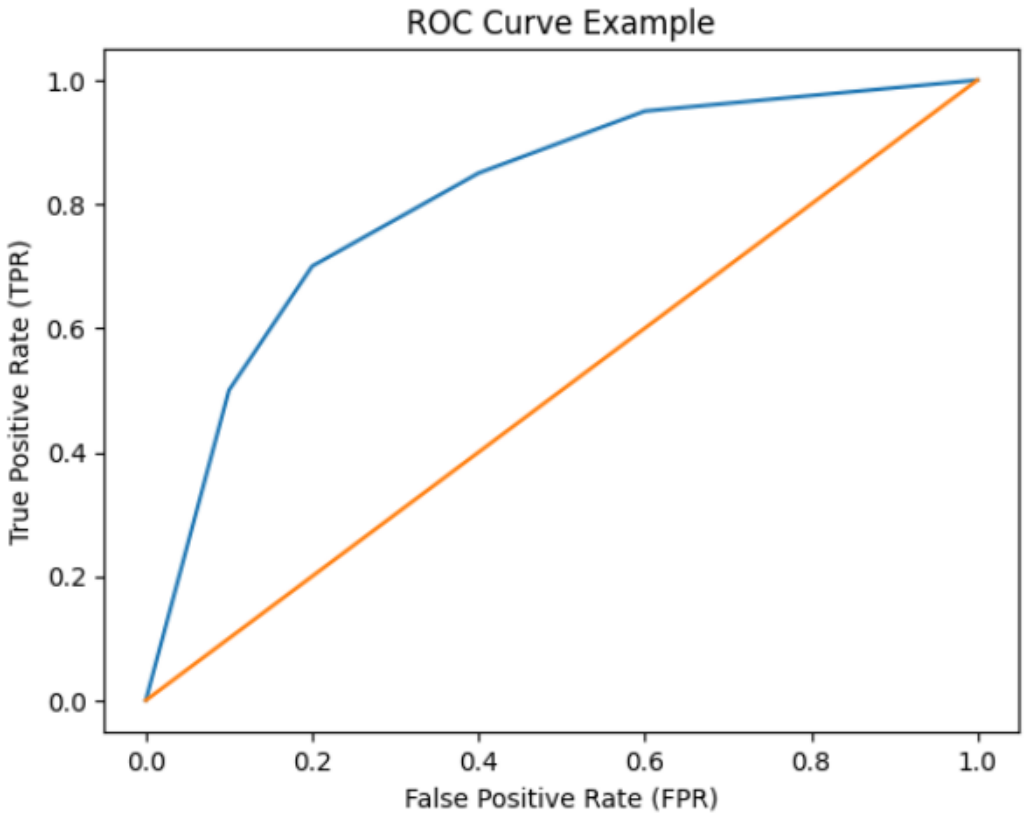
Performance Analysis

Error Analysis - Train/Test Split, validation set, Accuracy, Precision, Recall, F-measure, ROC curve, Confusion Matrix

ROC Curve (Receiver Operating Characteristic Curve) is a graphical tool used to evaluate the performance of a classification model, especially for binary classification.

It shows the relationship between:

- **True Positive Rate (TPR)** or **Sensitivity** on the Y-axis
- **False Positive Rate (FPR)** on the X-axis



ROC Curve – Graph and Explanation

You can see the ROC curve plotted above.

How to read this graph:

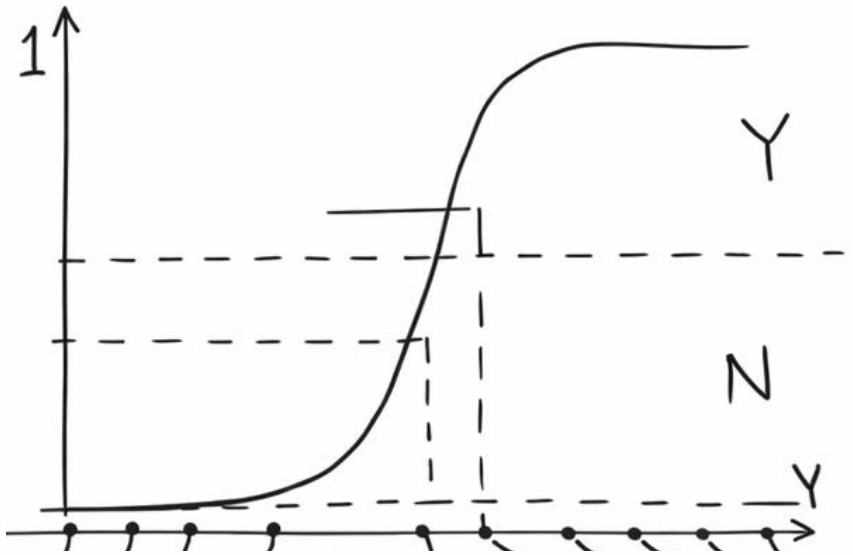
- **X-axis:** False Positive Rate (FPR)

  = FP / (FP + TN)

  → How many negatives are wrongly predicted as positive.
- **Y-axis:** True Positive Rate (TPR) or Sensitivity

  = TP / (TP + FN)

  → How many actual positives are correctly predicted.



at T= 0.5

3 TP	2 FP
2 FN	3 TN

TPR= 3/5
FPR= 2/5

at T= 0

5 TP	0 FP
0 FN	5 TN

TPR= 5/5
FPR= 0/5

at T= 1

0 TP	5 FP
5 FN	0 TN

TPR= 0
FPR= 5/5

at T= 0.75

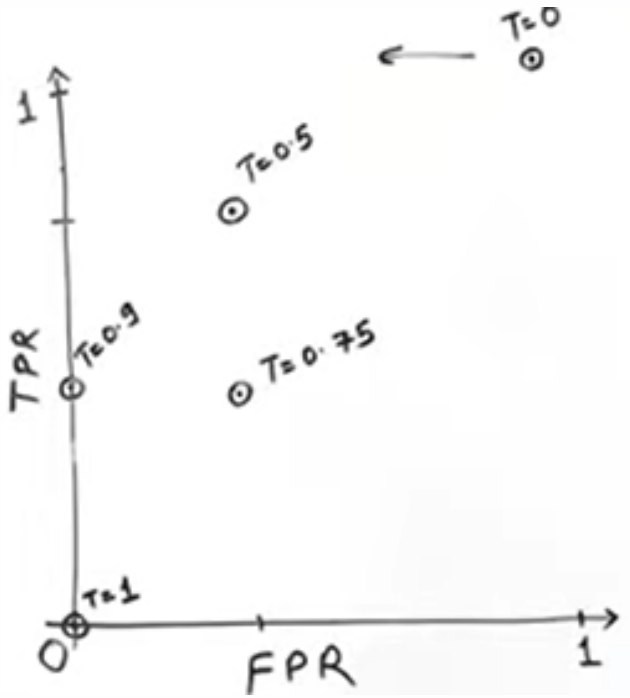
2 TP	2 FP
3 FN	3 TN

TPR= 2/5
FPR= 2/5

at T= 0.9

2 TP	0 FP
3 FN	5 TN

TPR= 2/5
FPR= 0/5



	Y	N
Y	TP	FP
N	FN	TN

$\rightarrow TPR = \frac{TP}{TP+FN}$

$\rightarrow FPR = \frac{FP}{FP+TN}$

Lines in the graph:

1. **Curved Line (Model Curve):**

   This shows the performance of a classifier at different thresholds.

   Each point corresponds to a different decision threshold.

2. **Diagonal Line:**

   This represents a **random classifier**.

   If your model lies close to this line, it is not useful.

Interpretation:

- The closer the curve is to the **top-left corner**, the better the model.
- A good model has:
  - High TPR (more correct positives)
  - Low FPR (fewer false alarms)

Interpretation:

- The closer the curve is to the **top-left corner**, the better the model.
- A good model has:
  - High TPR (more correct positives)
  - Low FPR (fewer false alarms)

AUC (Area Under Curve):

- The area under this curve is called **AUC**.
- Range: 0 to 1
  - 0.5 → Random model
  - 1.0 → Perfect model
  - Higher AUC = Better classifier

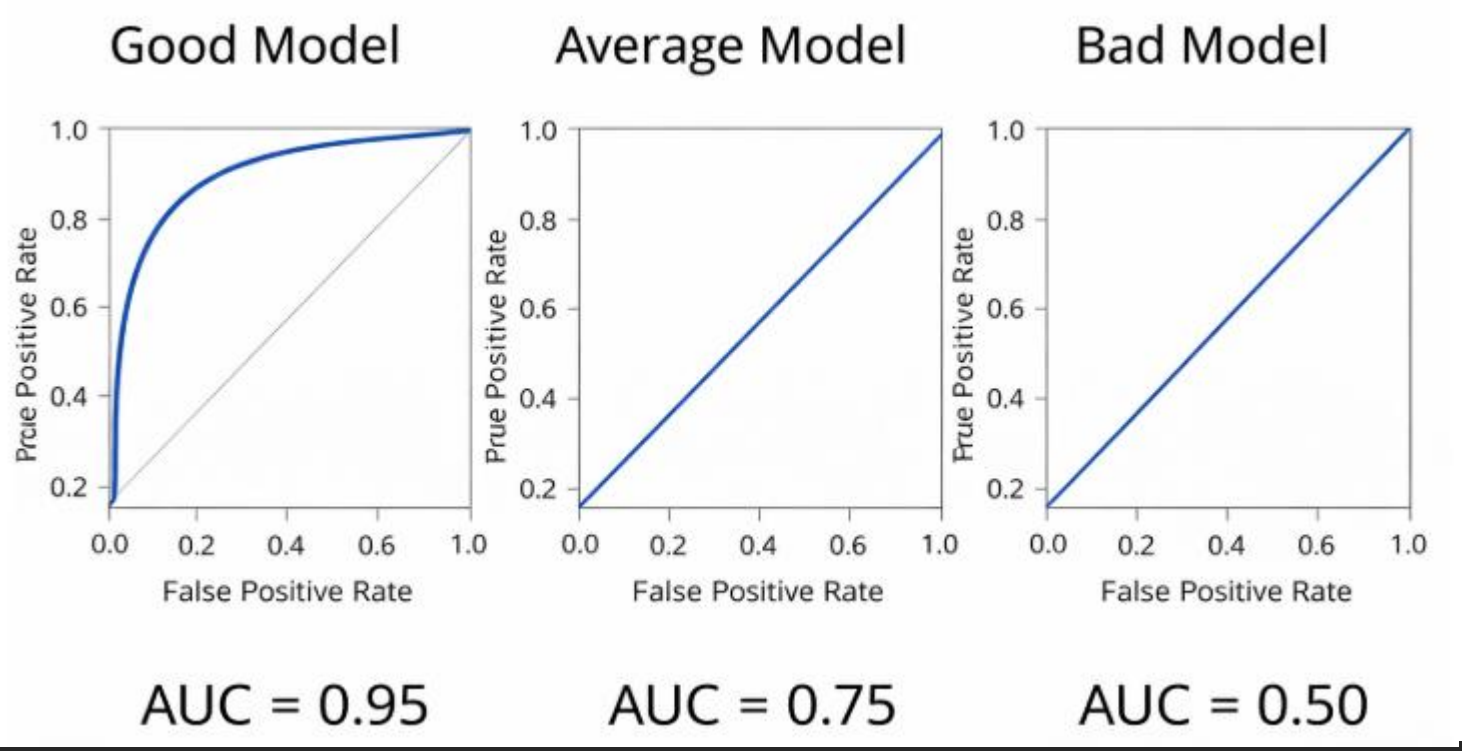
In simple words:

The ROC curve shows how well a model separates two classes by balancing:

- Catching positives correctly
- Avoiding false alarms

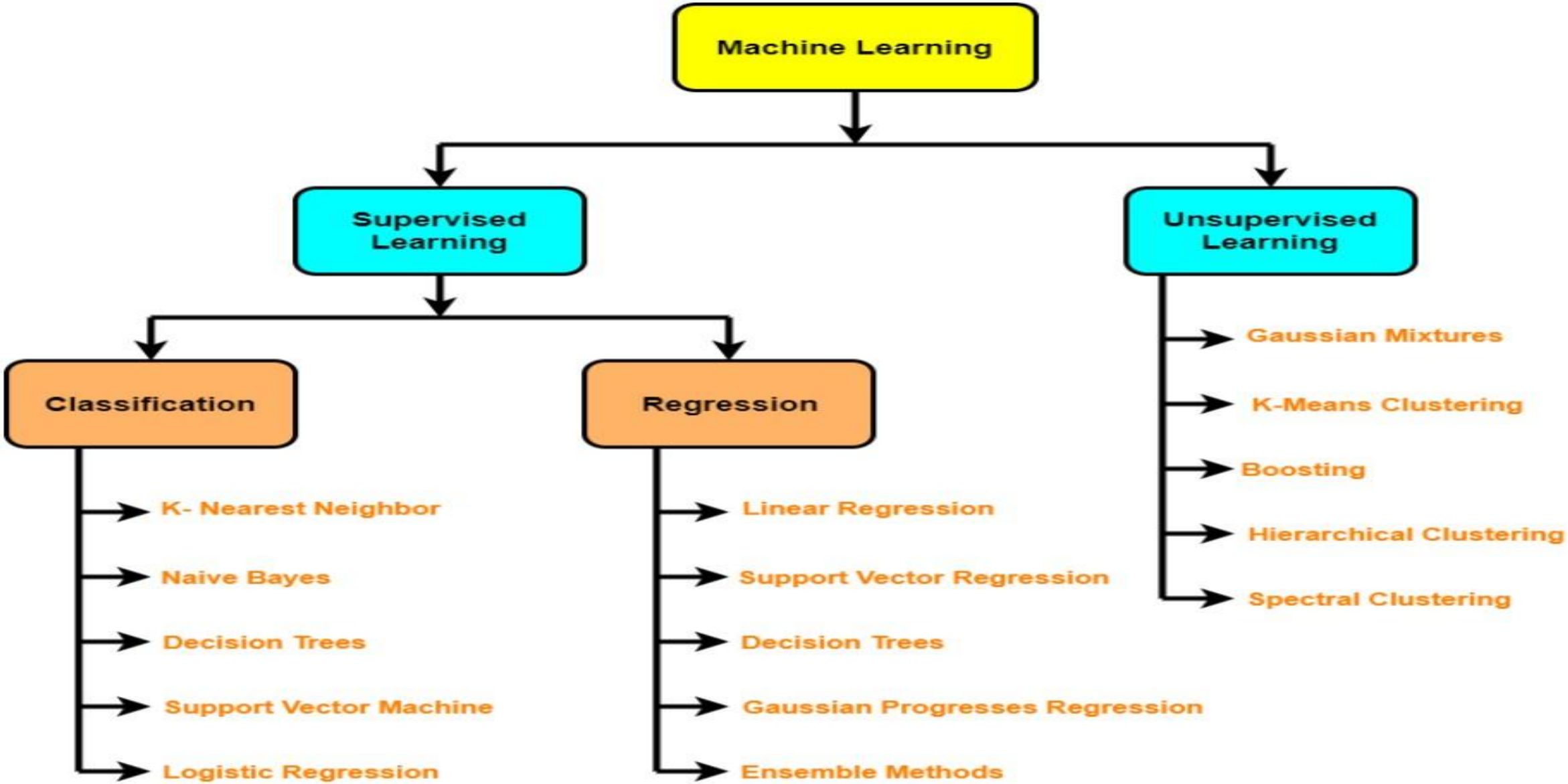
AUC (Area Under Curve):

- The area under this curve is called **AUC**.
- Range: 0 to 1
  - 0.5 → Random model
  - 1.0 → Perfect model
  - Higher AUC = Better classifier



17	Decision Tree: Introduction, Id3 Algorithm - 1
18	Decision Tree - Id3 Algorithm - 2
19	Decision Tree - Problem of Overfitting, Pre-pruning/post-pruning Decision Tree, Examples.
20	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 1
21	Support Vector Machine - Terminologies, Intuition, Learning, Derivation - 2
22	Support Vector Machine - KKT Condition - 3
23	Support Vector Machine - Kernel, Nonlinear Classification, and

25	Principal Component Analysis - Steps, merits, demerits, Intuition - 1
26	Principal Component Analysis - Steps, merits, demerits, Intuition - 2
27	Understanding and Implementing PCA using SVD for dimensionality reduction


Machine Learning
Supervised Learning
Unsupervised Learning
Classification
Regression
K- Nearest Neighbor
Naive Bayes
Decision Trees
Support Vector Machine
Logistic Regression
Linear Regression
Support Vector Regression
Decision Trees
Gaussian Progresses Regression
Ensemble Methods
Gaussian Mixtures
K-Means Clustering
Boosting
Hierarchical Clustering
Spectral Clustering

Naïve Bayes Classifier

What is Naive Bayes classifiers?

Naive Bayes classifiers are a collection of classification algorithms .

Works on the principle of conditional probability.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem.** It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

Conditional Probability

Conditional probability is the probability that an event happens **given that another event has already happened**.

It is written as:

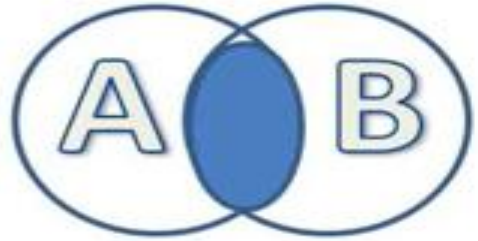
$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

This means:

Probability of A **when B is known to have occurred**.

conditional probability

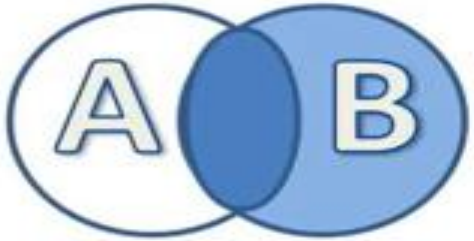
$P(A \cap B)$



Intersection of Events A and B

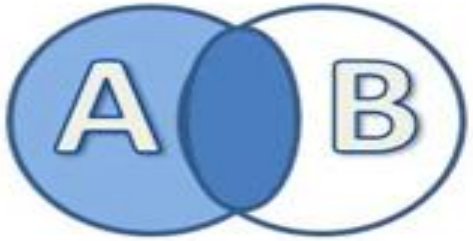
P(A ∩ B) is a probability of both events A and B occurring together.

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



A for given B

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



B for given A

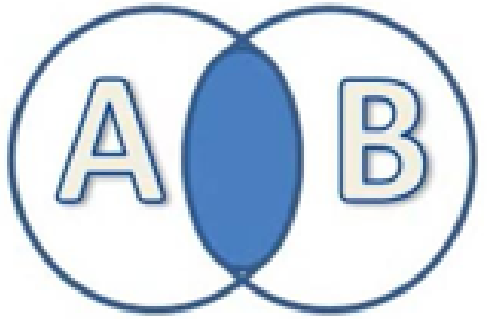
- $P(A \cap B)$ counts outcomes where both A and B happen.
- $P(B)$ counts all outcomes where B happens.

$$\frac{\text{Outcomes satisfying both A and B}}{\text{Outcomes satisfying B}}$$

What is Naive Bayes theorem?

$$P(A \cap B) = P(A)P(B \mid A)$$

$$P(A \cap B) = P(B)P(A \mid B)$$



Intersection of Events

- $P(A \cap B)$ counts outcomes where both A and B happen.

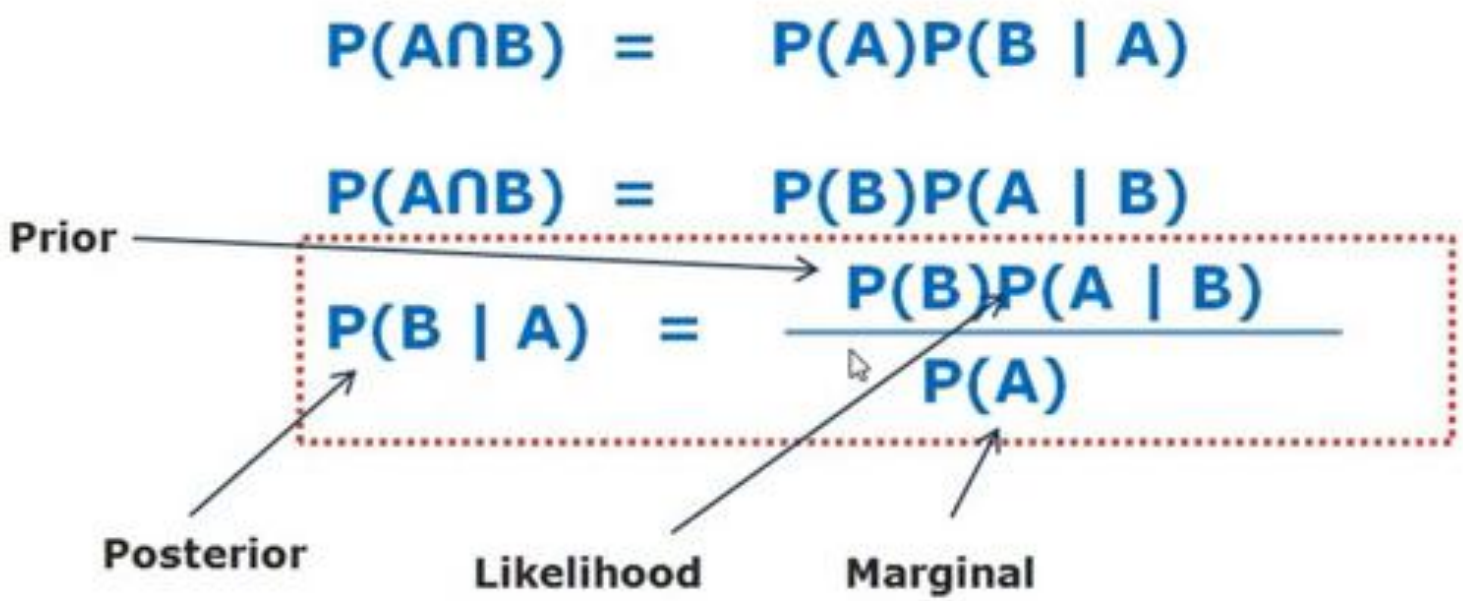
What is Naive Bayes theorem?

$$P(A \cap B) = P(B \mid A)\, P(A)$$

$$P(A \cap B) = P(A \mid B)\, P(B)$$

$$P(B \mid A)\, P(A) = P(B) P(A \mid B)$$

What is Naive Bayes theorem?



Baye's theorem gives the relationship between the probabilities of A and B, and the conditional probabilities of A given B and B given A.

Bayes' Theorem

Given a hypothesis H and evidence E, Bayes' theorem states that the relationship between the probability of the hypothesis before getting the evidence $P(H)$ and the probability of the hypothesis after getting the evidence $P(H|E)$ is

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Bayes' Theorem Proof

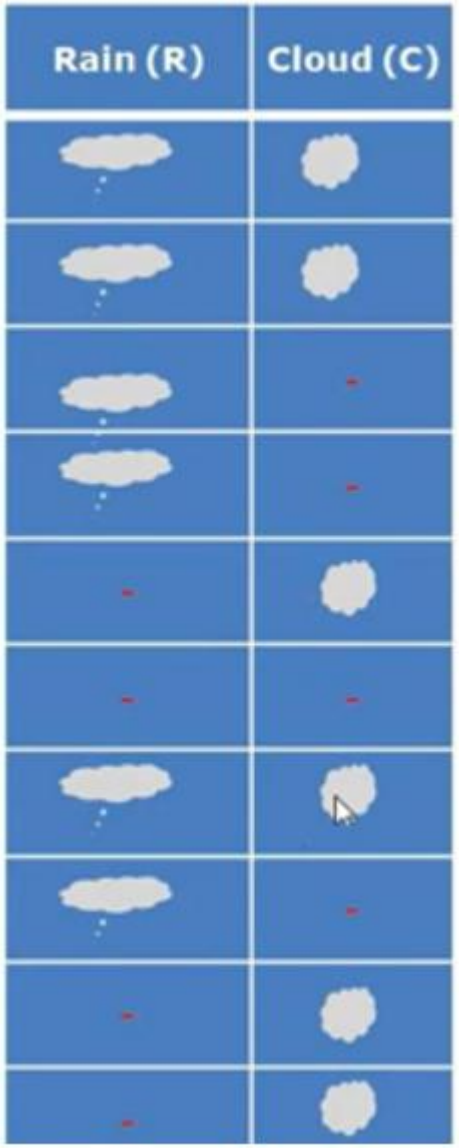
Likelihood
How probable is the evidence
Given that our hypothesis is true?

Prior
How probable was our hypothesis
Before observing the evidence?

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

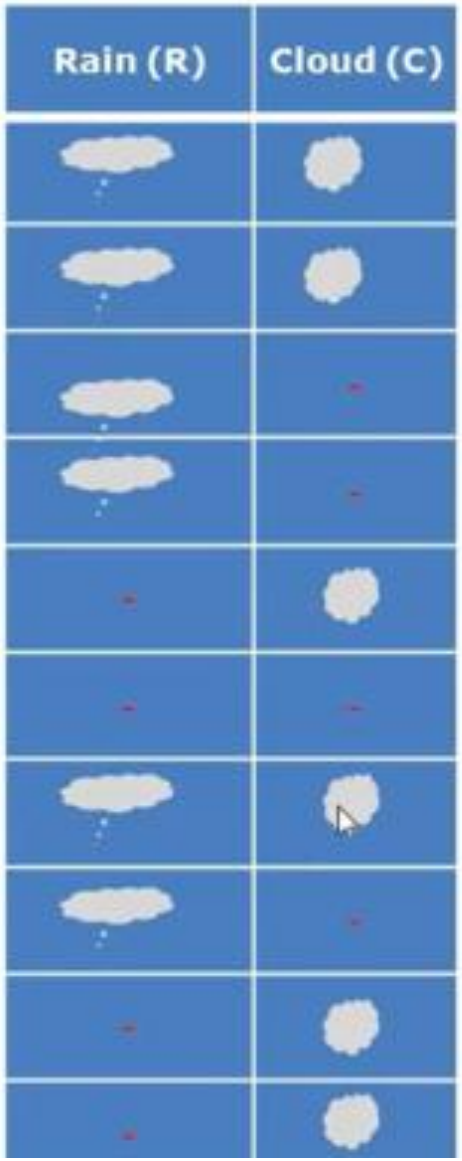
Posterior
How probable is our Hypothesis
Given the observed evidence?
(Not directly computable)

Marginal
How probable is the new evidence
Under all possible hypothesis?



$$P(R|C) =$$

Naive Bayes theorem



$$P(R|C) = \frac{P(C \mid R)\,P(R)}{P(C)} = \frac{(3/6)*(6/10)}{6/10}$$

$$P(R|C) = 0.5$$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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

NAIVE BAYES CLASSIFIER

Example - 1

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5

NAIVE BAYES CLASSIFIER – Example -1

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \prod_i \; P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot \; P(Humidity = high|v_j) P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes) \; P(sunny|yes) \; P(cool|yes) \; P(high|yes) \; P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no) \; P(sunny|no) \; P(cool|no) \; P(high|no) \; P(strong|no) = .0206$$

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes) + v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes) + v_{NB}(no)} = 0.795$$

Color	Type	Origin	Stolen?
Red	Sports	Domestic	Yes
Red	Sports	Domestic	No
Red	Sports	Domestic	Yes
Yellow	Sports	Domestic	No
Yellow	Sports	Imported	Yes
Yellow	SUV	Imported	No
Yellow	SUV	Imported	Yes
Yellow	SUV	Domestic	No
Red	SUV	Imported	No
Red	Sports	Imported	Yes

$$\boldsymbol{New\ Instance} = (\boldsymbol{Red}, \boldsymbol{SUV}, \boldsymbol{Domestic})$$

$$p(Yes) = \frac{5}{10} = 0.5$$

$$p(No) = \frac{5}{10} = 0.5$$

Color	Yes	No
Red	3/5	2/5
Yellow	2/5	3/5

Type	Yes	No
Sports	4/5	2/5
SUV	1/5	3/5

Origin	Yes	No
Domestic	2/5	3/5
Imported	3/5	2/5

$$P(Yes|New\ Instance) = p(Yes) * P(Color = Red|Yes) * P(Type = SUV|Yes) * P(Origin = Domestic|Yes)$$

$$P(Yes|New\ Instance) = \frac{5}{10} * \frac{3}{5} * \frac{1}{5} * \frac{2}{5} = \frac{3}{125} = 0.024$$

$$P(No|New\ Instance) = p(No) * P(Color = Red|No) * P(Type = SUV|No) * P(Origin = Domestic|No)$$

$$P(No|New\ Instance) = \frac{5}{10} * \frac{2}{5} * \frac{3}{5} * \frac{3}{5} = \frac{9}{125} = 0.072$$

$$P(No|New\ Instance) > P(Yes|New\ Instance)$$

Why it is Called Naive Bayes?

It is named as "Naive" because it assumes the presence of one feature does not affect other features. The "Bayes" part of the name refers to its basis in Bayes'

You are given a small dataset on fruit attributes. We want to classify a new fruit with the attributes **{Long: Yes, Sweet: Yes, Yellow: No}** as either a **Banana** or **Orange** using the Naive Bayes algorithm.

Training Data

Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

Solution Steps

The Naive Bayes formula is:

$$P(Class|Features) \propto P(Class) \times \prod P(Feature|Class)$$

We need to calculate the posterior probability for both "Banana" and "Orange" and select the class with the highest value.

1. Calculate Prior Probabilities

The prior probability is the overall probability of each class based on the training data.

- $P(\text{Banana}) = \text{Total Bananas}/\text{Total Fruits} = 500/1000 = 0.5$
- $P(\text{Orange}) = \text{Total Oranges}/\text{Total Fruits} = 300/1000 = 0.3$

2. Calculate Likelihoods (Conditional Probabilities)

Calculate the probability of each feature given the class label for the new fruit {Long: Yes, Sweet: Yes, Yellow: No}.

For Banana:

- $P(\text{Long: Yes}|\text{Banana}) = 400/500 = 0.8$
- $P(\text{Sweet: Yes}|\text{Banana}) = 350/500 = 0.7$
- $P(\text{Yellow: No}|\text{Banana}) = (500 - 450)/500 = 50/500 = 0.1$

For Orange:

- $P(\text{Long: Yes}|\text{Orange}) = 0/300 = 0.0$
- $P(\text{Sweet: Yes}|\text{Orange}) = 150/300 = 0.5$
- $P(\text{Yellow: No}|\text{Orange}) = (300 - 300)/300 = 0/300 = 0.0$

3. Calculate Unnormalized Posterior Probabilities

Multiply the prior probability by the likelihoods for each class.

- **Probability (Banana):**
  $$P(\text{Banana}) \times P(\text{Long: Yes}|\text{Banana}) \times P(\text{Sweet: Yes}|\text{Banana}) \times P(\text{Yellow: No}|\text{Banana})$$
  $$= 0.5 \times 0.8 \times 0.7 \times 0.1 = \mathbf{0.028}$$
- **Probability (Orange):**
  $$P(\text{Orange}) \times P(\text{Long: Yes}|\text{Orange}) \times P(\text{Sweet: Yes}|\text{Orange}) \times P(\text{Yellow: No}|\text{Orange})$$
  $$= 0.3 \times 0.0 \times 0.5 \times 0.0 = \mathbf{0.0}$$

4. Predict the Class

Compare the resulting probabilities. The class with the highest probability is the predicted class.

- Banana Probability: 0.028
- Orange Probability: 0.0

Since 0.028 > 0.0, the Naive Bayes classifier predicts that the fruit with attributes {Long: Yes, Sweet: Yes, Yellow: No} is a **Banana**.

Types of Naïve Bayes

Gaussian

Multinomial

Bernoulli

Bernoulli Naive Bayes

Here, the predictors are boolean variables. So, the only values you have are 'True' and 'False' (you could also have 'Yes' or 'No'). We use it when the data is according to multivariate Bernoulli distribution.

It is one of the prevalent **types of Naive Bayes model:** Its working is identical to the Multinomial classifier. However, the predictor variables are the independent Boolean variables. For example, it works as -a specific word exists or not in a document. Moreover, this model is famous for document classification tasks.

Multinomial Naive Bayes

People use this algorithm to solve document classification problems. For example, if you want to determine whether a document belongs to the 'Legal' category or 'Human Resources' category, you'd use this algorithm to sort it out. It uses the frequency of the present words as features.

This model is used when the data is multinomial distributed. Primarily, it is used for document classification problems. It denotes that a specific document belongs to which category (like Education, Politics, Sports, etc.). You can easily

understand these **types of Naive Bayes models** with an example.

Suppose there is a text document, and you want to extract all the distinctive words and prepare multiple features such that every feature signifies the word count in the document. The frequency is a feature to consider in this example. When you use multinomial Naive Bayes for this example, it neglects the non-occurrence of the features. Hence, if the e frequency is 0, the probability of occurrence of the particular feature is 0. It is one of those **types of Naive Bayes model:** that seamlessly works with text classification problems.

Gaussian Naive Bayes

If the predictors aren't discrete but have a continuous value, we assume that they are a sample from a gaussian distribution. It is among those types of Naive Bayes models that consider normal distribution. It assumes that the feature adopts a normal distribution. If predictors accept continuous values instead of discrete, the Gaussian Naive Bayes model assumes that such values are sampled through the Gaussian distribution. It is always better to first identify your problem and determine which is not a main type of Naive Bayes classifier.

Naïve Bayes Classifier

What is Naive Bayes classifiers?

Naive Bayes classifiers are a collection of classification algorithms .

Works on the principle of conditional probability.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem.** It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

Conditional Probability

Conditional probability is the probability that an event happens **given that another event has already happened**.

It is written as:

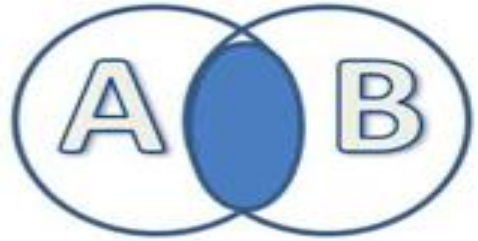
$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

This means:

Probability of A **when B is known to have occurred**.

conditional probability

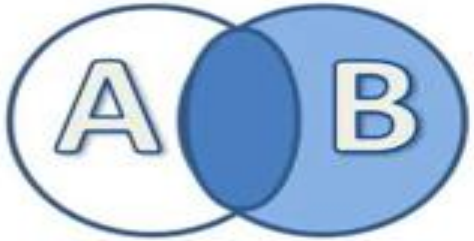
$P(A \cap B)$



Intersection of Events A and B

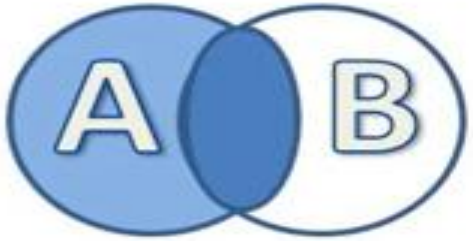
P(A ∩ B) is a probability of both events A and B occurring together.

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



A for given B

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



B for given A

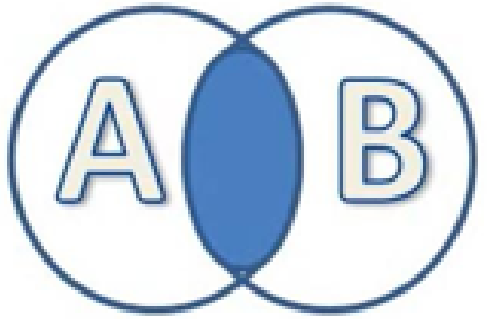
- $P(A \cap B)$ counts outcomes where both A and B happen.
- $P(B)$ counts all outcomes where B happens.

$$\frac{\text{Outcomes satisfying both A and B}}{\text{Outcomes satisfying B}}$$

What is Naive Bayes theorem?

$$P(A \cap B) = P(A)P(B \mid A)$$

$$P(A \cap B) = P(B)P(A \mid B)$$



Intersection of Events

- $P(A \cap B)$ counts outcomes where both A and B happen.

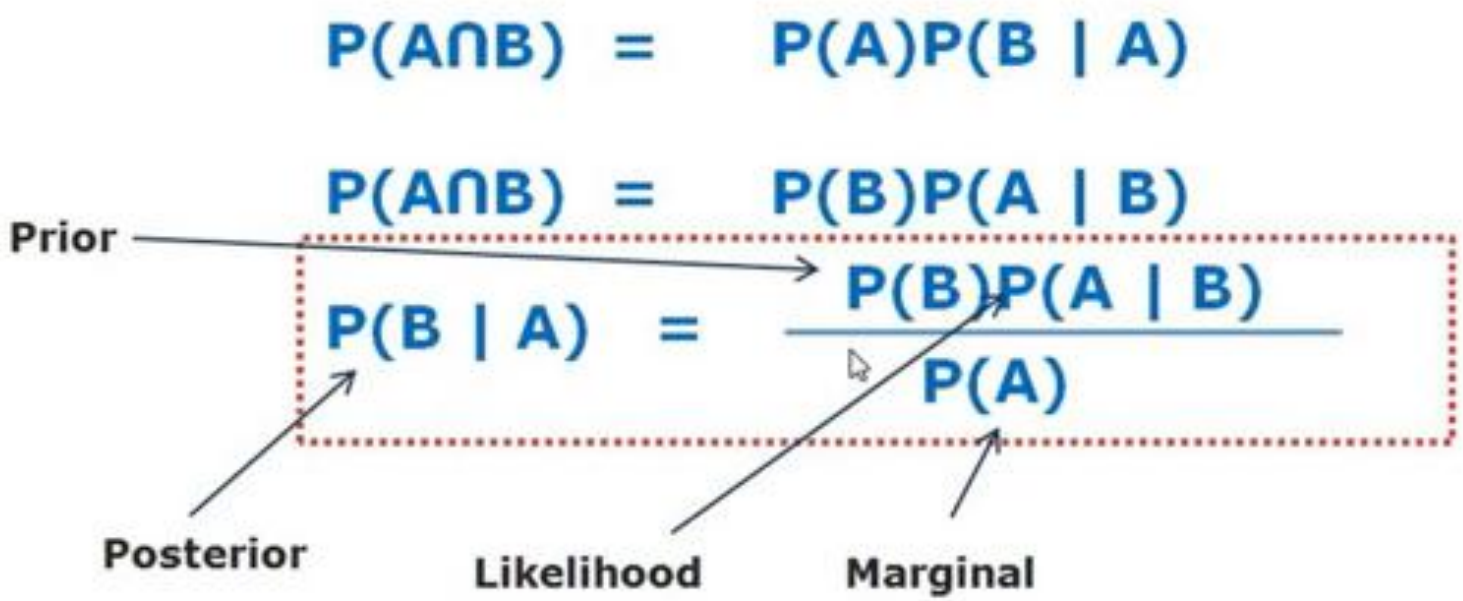
What is Naive Bayes theorem?

$$P(A \cap B) = P(B \mid A)\, P(A)$$

$$P(A \cap B) = P(A \mid B)\, P(B)$$

$$P(B \mid A)\, P(A) = P(B) P(A \mid B)$$

What is Naive Bayes theorem?



Baye's theorem gives the relationship between the probabilities of A and B, and the conditional probabilities of A given B and B given A.

Bayes' Theorem

Given a hypothesis H and evidence E, Bayes' theorem states that the relationship between the probability of the hypothesis before getting the evidence $P(H)$ and the probability of the hypothesis after getting the evidence $P(H|E)$ is

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Bayes' Theorem Proof

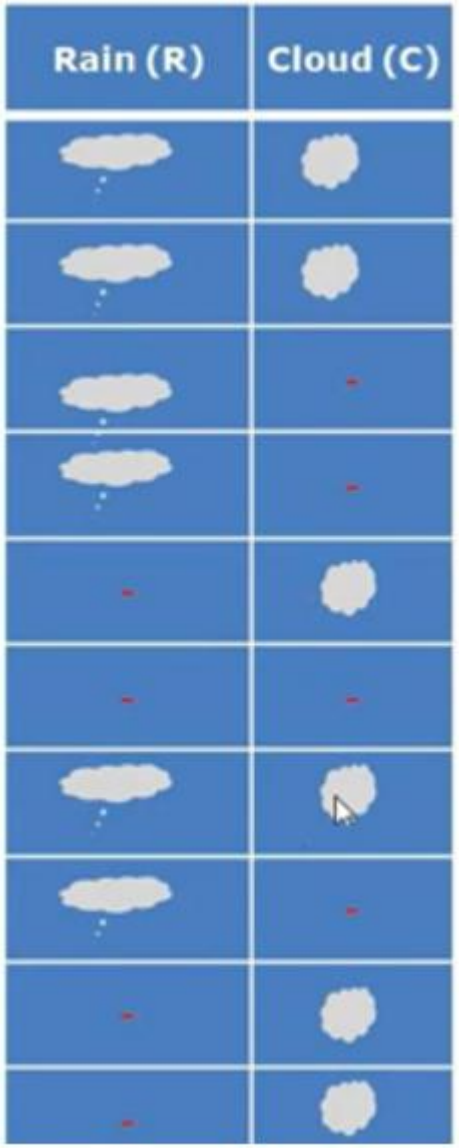
Likelihood
How probable is the evidence
Given that our hypothesis is true?

Prior
How probable was our hypothesis
Before observing the evidence?

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

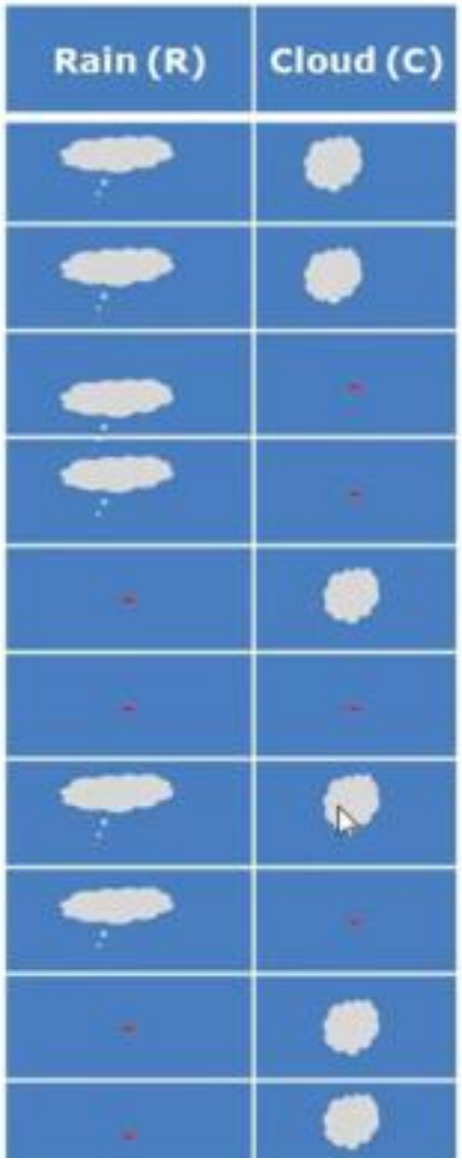
Posterior
How probable is our Hypothesis
Given the observed evidence?
(Not directly computable)

Marginal
How probable is the new evidence
Under all possible hypothesis?



$$P(R|C) =$$

Naive Bayes theorem



$$P(R|C) = \frac{P(C \mid R)\,P(R)}{P(C)} = \frac{(3/6)*(6/10)}{6/10}$$

$$P(R|C) = 0.5$$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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

NAIVE BAYES CLASSIFIER

Example - 1

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5

NAIVE BAYES CLASSIFIER – Example -1

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \prod_i \; P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot \; P(Humidity = high|v_j) P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes) \; P(sunny|yes) \; P(cool|yes) \; P(high|yes) \; P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no) \; P(sunny|no) \; P(cool|no) \; P(high|no) \; P(strong|no) = .0206$$

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes) + v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes) + v_{NB}(no)} = 0.795$$

Color	Type	Origin	Stolen?
Red	Sports	Domestic	Yes
Red	Sports	Domestic	No
Red	Sports	Domestic	Yes
Yellow	Sports	Domestic	No
Yellow	Sports	Imported	Yes
Yellow	SUV	Imported	No
Yellow	SUV	Imported	Yes
Yellow	SUV	Domestic	No
Red	SUV	Imported	No
Red	Sports	Imported	Yes

$$\boldsymbol{New\ Instance} = (\boldsymbol{Red}, \boldsymbol{SUV}, \boldsymbol{Domestic})$$

$$p(Yes) = \frac{5}{10} = 0.5$$

$$p(No) = \frac{5}{10} = 0.5$$

Color	Yes	No
Red	3/5	2/5
Yellow	2/5	3/5

Type	Yes	No
Sports	4/5	2/5
SUV	1/5	3/5

Origin	Yes	No
Domestic	2/5	3/5
Imported	3/5	2/5

$$P(Yes|New\ Instance) = p(Yes) * P(Color = Red|Yes) * P(Type = SUV|Yes) * P(Origin = Domestic|Yes)$$

$$P(Yes|New\ Instance) = \frac{5}{10} * \frac{3}{5} * \frac{1}{5} * \frac{2}{5} = \frac{3}{125} = 0.024$$

$$P(No|New\ Instance) = p(No) * P(Color = Red|No) * P(Type = SUV|No) * P(Origin = Domestic|No)$$

$$P(No|New\ Instance) = \frac{5}{10} * \frac{2}{5} * \frac{3}{5} * \frac{3}{5} = \frac{9}{125} = 0.072$$

$$P(No|New\ Instance) > P(Yes|New\ Instance)$$

Why it is Called Naive Bayes?

It is named as "Naive" because it assumes the presence of one feature does not affect other features. The "Bayes" part of the name refers to its basis in Bayes'

You are given a small dataset on fruit attributes. We want to classify a new fruit with the attributes **{Long: Yes, Sweet: Yes, Yellow: No}** as either a **Banana** or **Orange** using the Naive Bayes algorithm.

Training Data

Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

Solution Steps

The Naive Bayes formula is:

$$P(Class|Features) \propto P(Class) \times \prod P(Feature|Class)$$

We need to calculate the posterior probability for both "Banana" and "Orange" and select the class with the highest value.

1. Calculate Prior Probabilities

The prior probability is the overall probability of each class based on the training data.

- $P(\text{Banana}) = \text{Total Bananas}/\text{Total Fruits} = 500/1000 = 0.5$
- $P(\text{Orange}) = \text{Total Oranges}/\text{Total Fruits} = 300/1000 = 0.3$

2. Calculate Likelihoods (Conditional Probabilities)

Calculate the probability of each feature given the class label for the new fruit {Long: Yes, Sweet: Yes, Yellow: No}.

For Banana:

- $P(\text{Long: Yes}|\text{Banana}) = 400/500 = 0.8$
- $P(\text{Sweet: Yes}|\text{Banana}) = 350/500 = 0.7$
- $P(\text{Yellow: No}|\text{Banana}) = (500 - 450)/500 = 50/500 = 0.1$

For Orange:

- $P(\text{Long: Yes}|\text{Orange}) = 0/300 = 0.0$
- $P(\text{Sweet: Yes}|\text{Orange}) = 150/300 = 0.5$
- $P(\text{Yellow: No}|\text{Orange}) = (300 - 300)/300 = 0/300 = 0.0$

3. Calculate Unnormalized Posterior Probabilities

Multiply the prior probability by the likelihoods for each class.

- **Probability (Banana):**
  $$P(\text{Banana}) \times P(\text{Long: Yes}|\text{Banana}) \times P(\text{Sweet: Yes}|\text{Banana}) \times P(\text{Yellow: No}|\text{Banana})$$
  $$= 0.5 \times 0.8 \times 0.7 \times 0.1 = \mathbf{0.028}$$
- **Probability (Orange):**
  $$P(\text{Orange}) \times P(\text{Long: Yes}|\text{Orange}) \times P(\text{Sweet: Yes}|\text{Orange}) \times P(\text{Yellow: No}|\text{Orange})$$
  $$= 0.3 \times 0.0 \times 0.5 \times 0.0 = \mathbf{0.0}$$

4. Predict the Class

Compare the resulting probabilities. The class with the highest probability is the predicted class.

- Banana Probability: 0.028
- Orange Probability: 0.0

Since 0.028 > 0.0, the Naive Bayes classifier predicts that the fruit with attributes {Long: Yes, Sweet: Yes, Yellow: No} is a **Banana**.

Types of Naïve Bayes

Gaussian

Multinomial

Bernoulli

Bernoulli Naive Bayes

Here, the predictors are boolean variables. So, the only values you have are 'True' and 'False' (you could also have 'Yes' or 'No'). We use it when the data is according to multivariate Bernoulli distribution.

It is one of the prevalent **types of Naive Bayes model:** Its working is identical to the Multinomial classifier. However, the predictor variables are the independent Boolean variables. For example, it works as -a specific word exists or not in a document. Moreover, this model is famous for document classification tasks.

Multinomial Naive Bayes

People use this algorithm to solve document classification problems. For example, if you want to determine whether a document belongs to the 'Legal' category or 'Human Resources' category, you'd use this algorithm to sort it out. It uses the frequency of the present words as features.

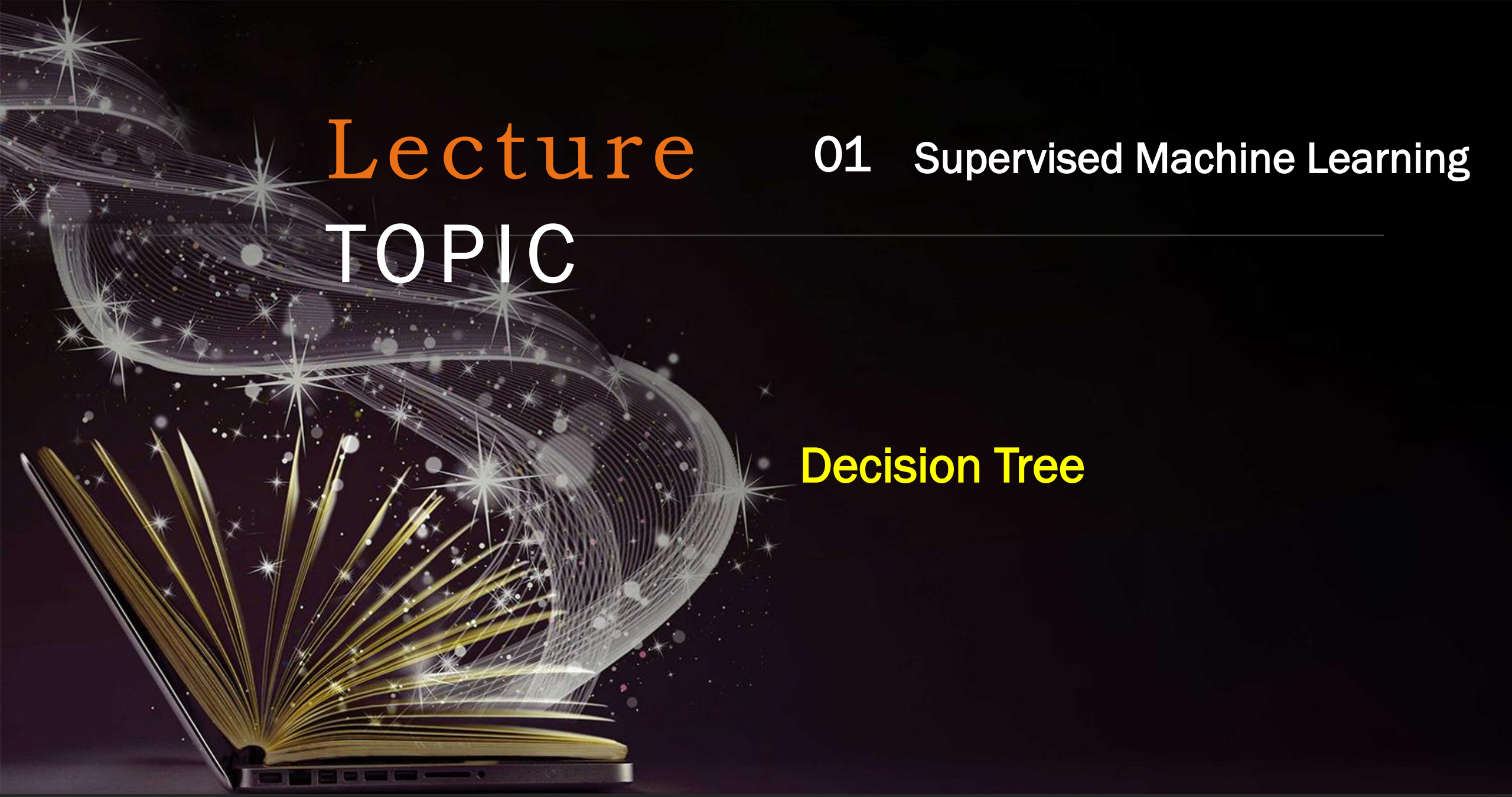
This model is used when the data is multinomial distributed. Primarily, it is used for document classification problems. It denotes that a specific document belongs to which category (like Education, Politics, Sports, etc.). You can easily

understand these **types of Naive Bayes models** with an example.

Suppose there is a text document, and you want to extract all the distinctive words and prepare multiple features such that every feature signifies the word count in the document. The frequency is a feature to consider in this example. When you use multinomial Naive Bayes for this example, it neglects the non-occurrence of the features. Hence, if the e frequency is 0, the probability of occurrence of the particular feature is 0. It is one of those **types of Naive Bayes model:** that seamlessly works with text classification problems.

Gaussian Naive Bayes

If the predictors aren't discrete but have a continuous value, we assume that they are a sample from a gaussian distribution. It is among those types of Naive Bayes models that consider normal distribution. It assumes that the feature adopts a normal distribution. If predictors accept continuous values instead of discrete, the Gaussian Naive Bayes model assumes that such values are sampled through the Gaussian distribution. It is always better to first identify your problem and determine which is not a main type of Naive Bayes classifier.



Lecture TOPIC

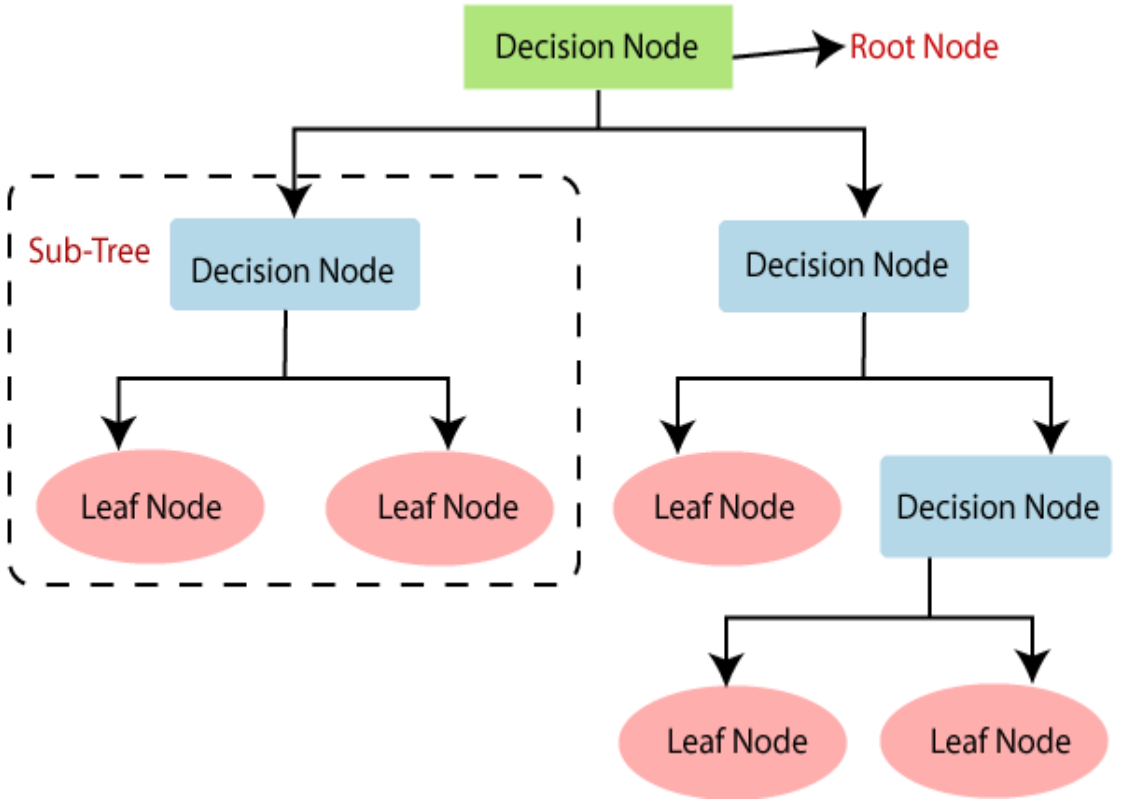
01 Supervised Machine Learning

Decision Tree

Decision Tree

Decision Tree Classification Algorithm

- 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 the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***



Decision Tree Classification Algorithm

- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**

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

Decision Tree Classification Algorithm

Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

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.

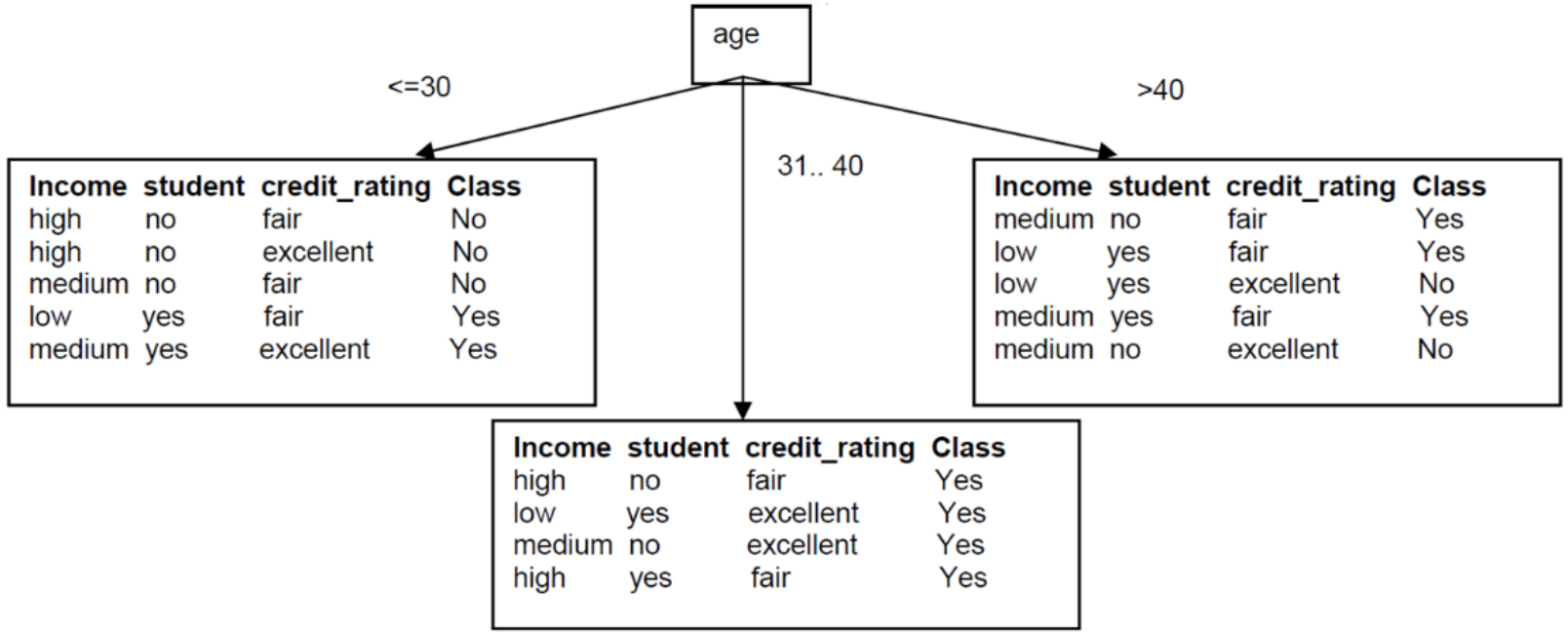
Decision Tree Classification Algorithm

- **How does the Decision Tree algorithm Work?**
- **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 node.

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes

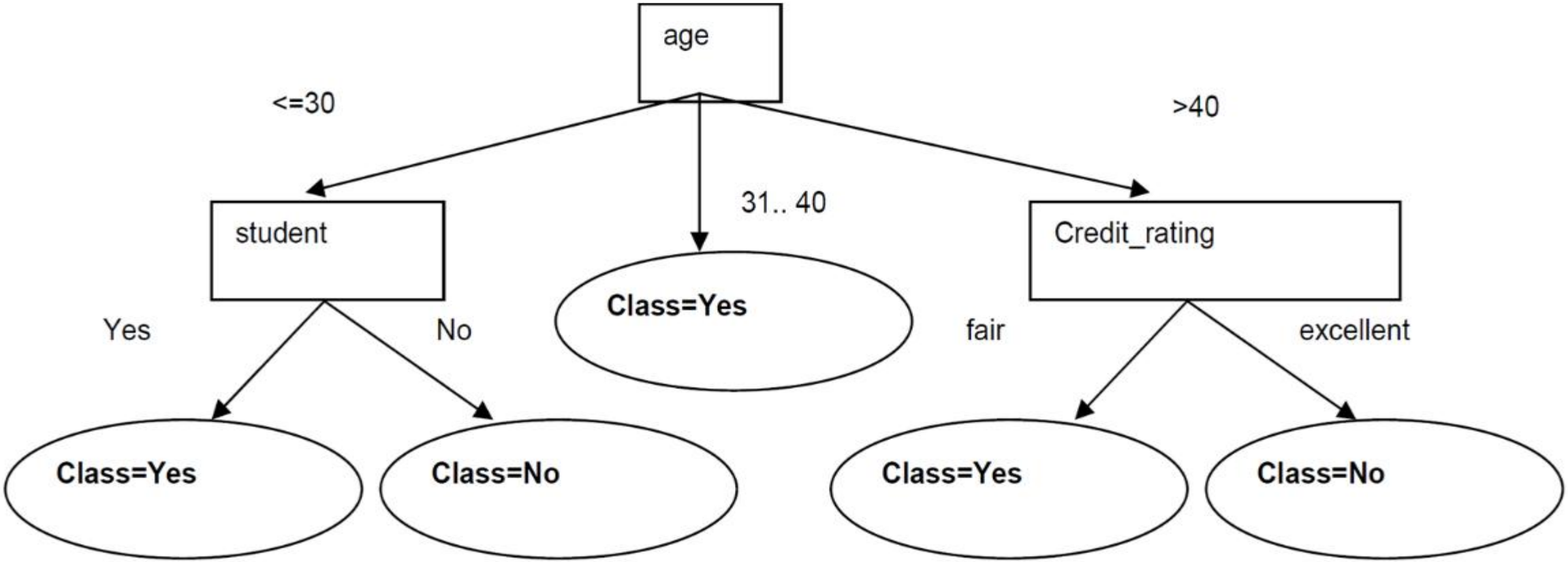
age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no

Since Age has the highest Information Gain we start splitting the dataset using the age attribute.



Decision Tree after step 1

Decision Tree



Decision Tree for Buys Computer

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:

- **Information Gain**

Entropy

Decision Tree Classification Algorithm

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation 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.
- 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. It can be calculated using the below formula:
- **Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)**

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(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

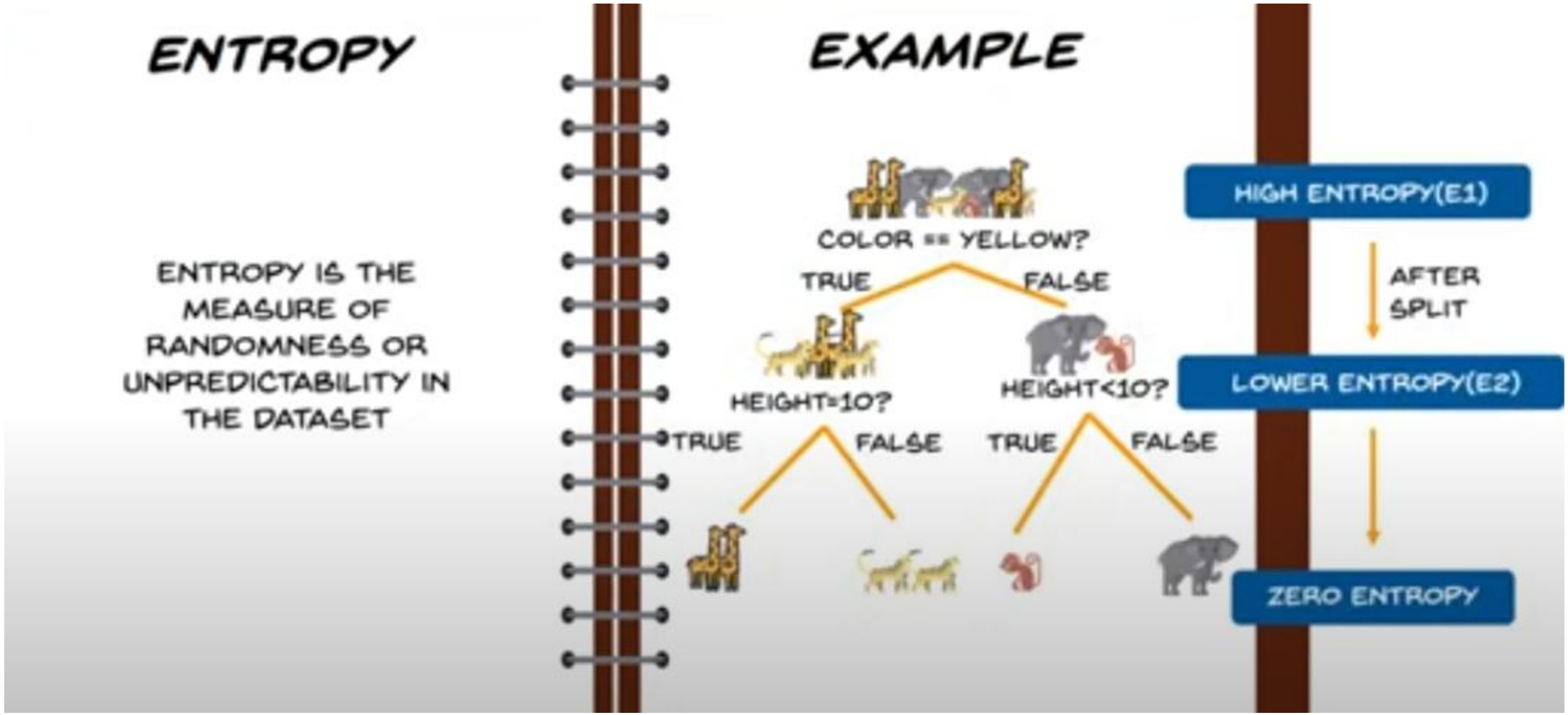
P(no)= probability of no

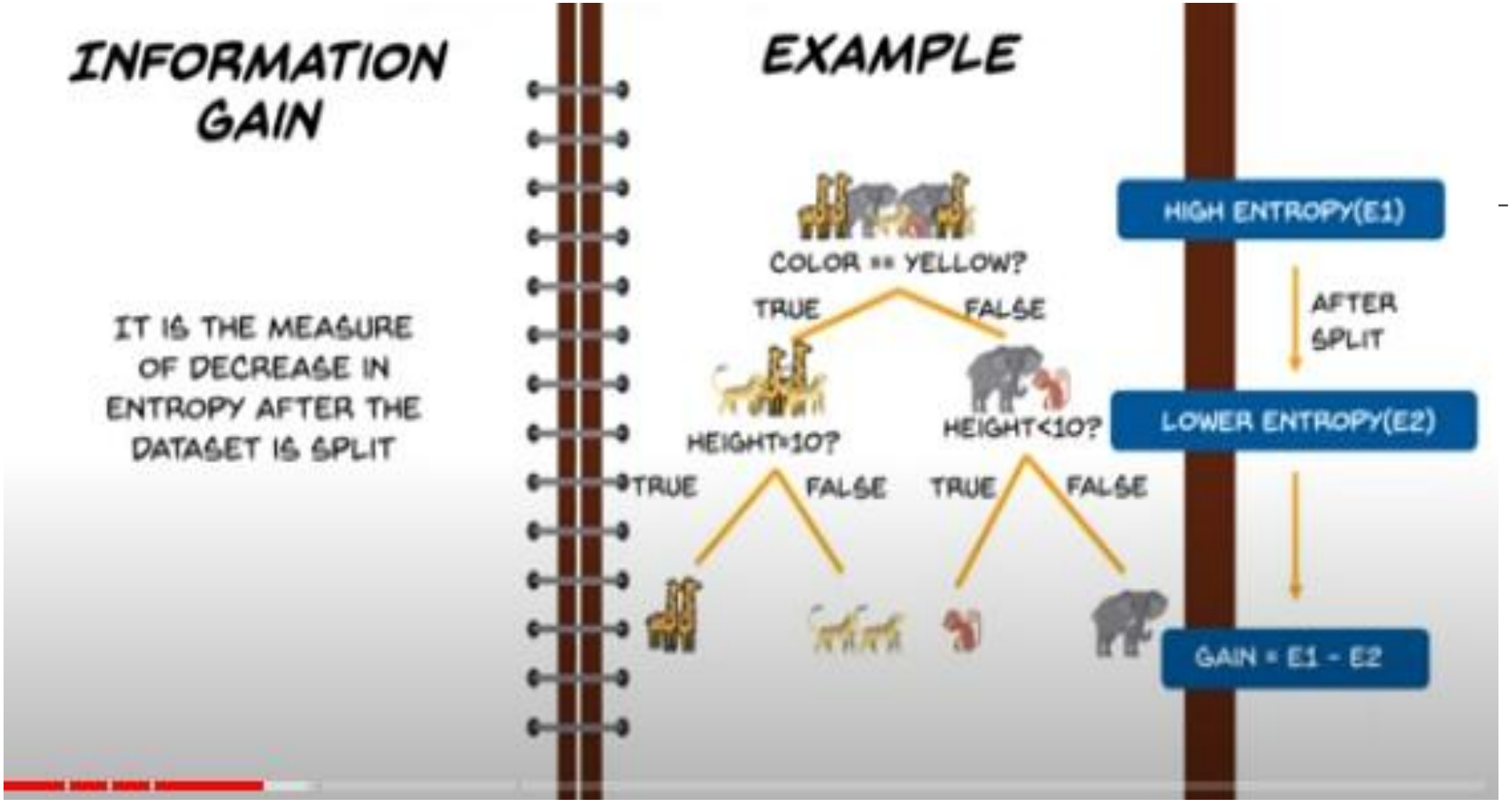
ENTROPY

ENTROPY IS THE MEASURE OF RANDOMNESS OR UNPREDICTABILITY IN THE DATASET

EXAMPLE

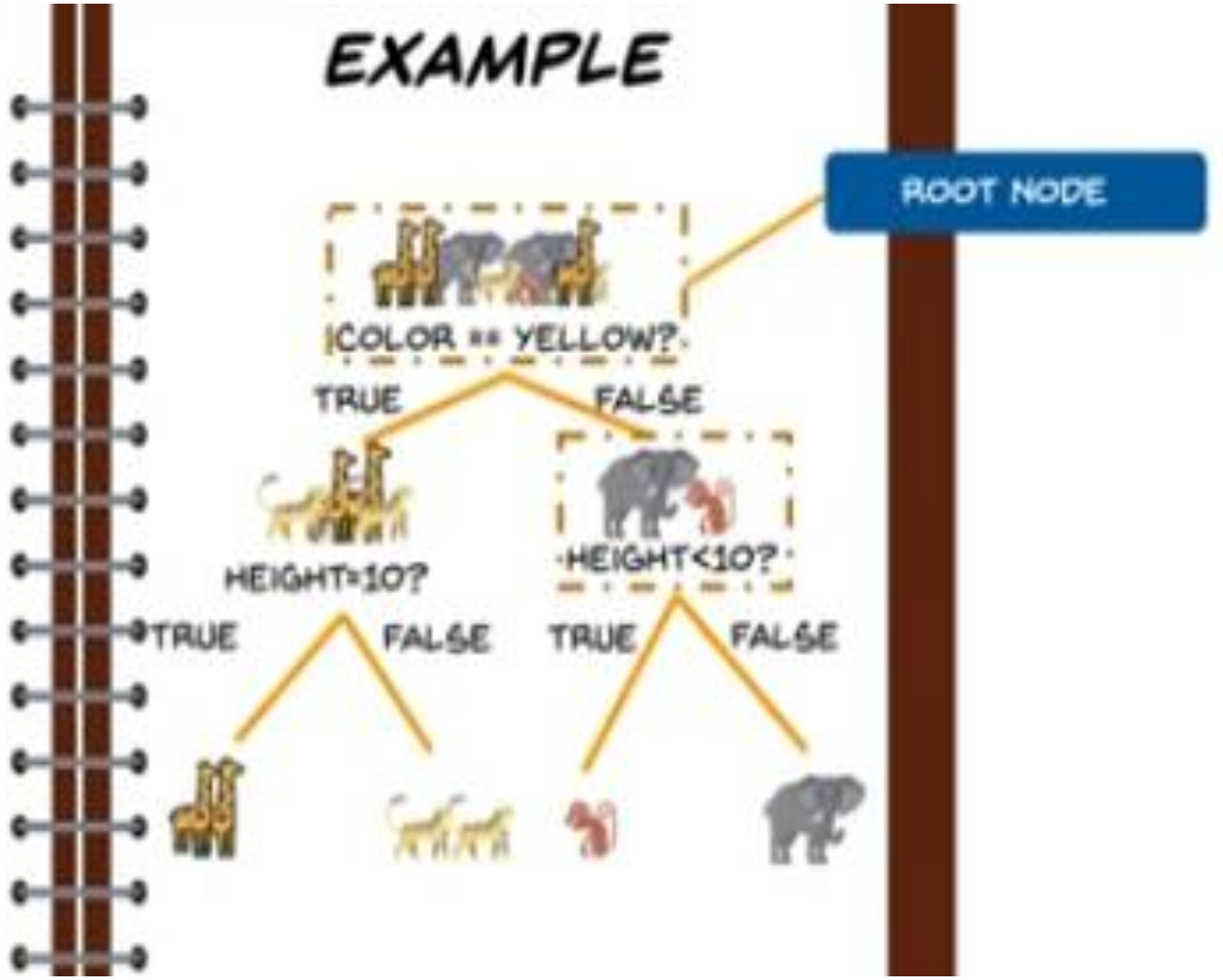
THIS DATASET HAS A VERY HIGH ENTROPY


ENTROPY
ENTROPY IS THE MEASURE OF RANDOMNESS OR UNPREDICTABILITY IN THE DATASET
EXAMPLE
COLOR == YELLOW?
TRUE
FALSE
HEIGHT=10?
HEIGHT<10?
TRUE
FALSE
TRUE
FALSE
HIGH ENTROPY(E1)
AFTER SPLIT
LOWER ENTROPY(E2)
ZERO ENTROPY


INFORMATION GAIN
IT IS THE MEASURE OF DECREASE IN ENTROPY AFTER THE DATASET IS SPLIT
EXAMPLE
COLOR == YELLOW?
TRUE
FALSE
HEIGHT>10?
HEIGHT<10?
TRUE
FALSE
TRUE
FALSE
HIGH ENTROPY(E1)
AFTER SPLIT
LOWER ENTROPY(E2)
GAIN = E1 - E2

ROOT NODE

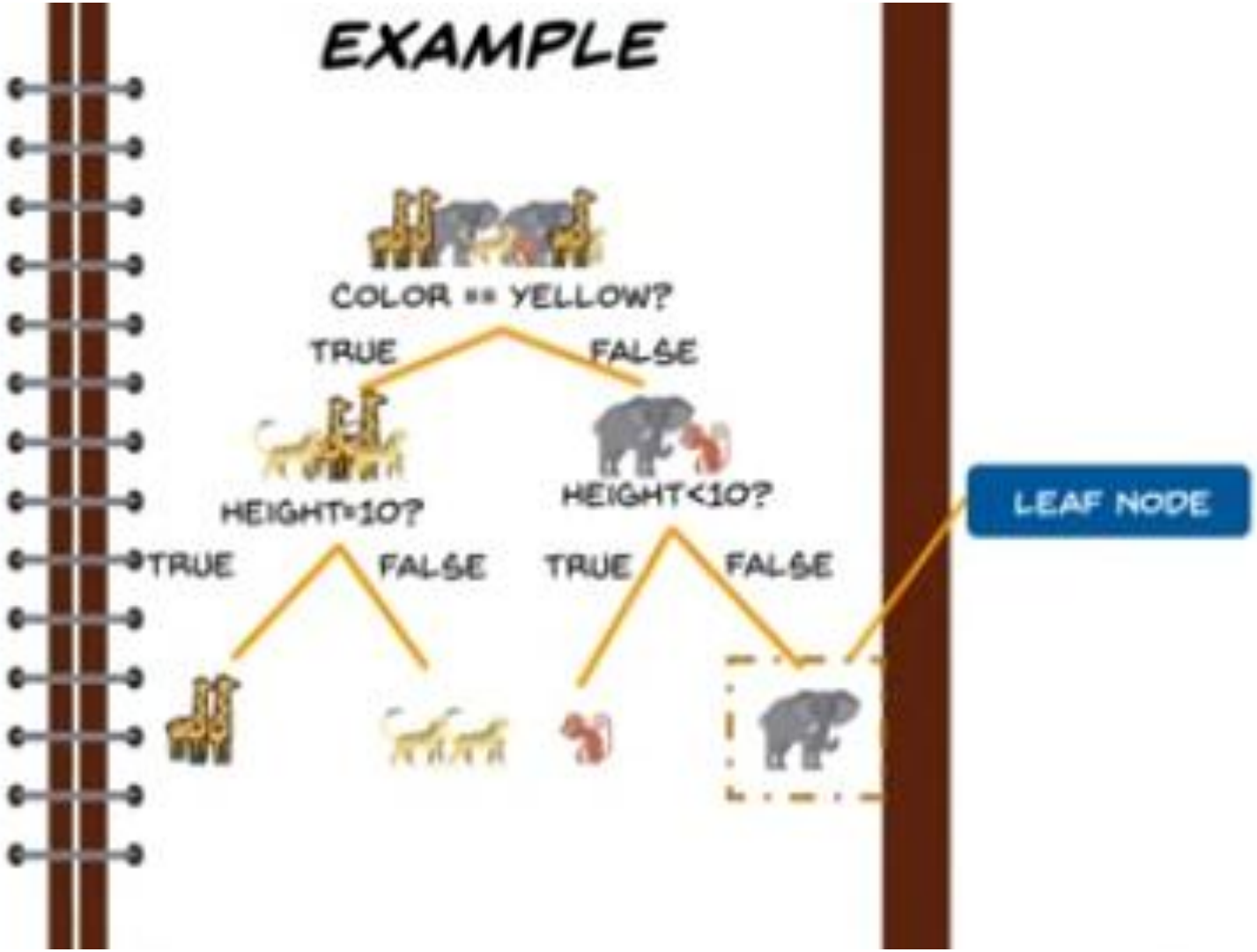
THE TOP MOST DECISION NODE IS KNOWN AS THE ROOT NODE



LEAF NODE

LEAF NODE CARRIES THE CLASSIFICATION OR THE DECISION

EXAMPLE



ENTROPY AND INFORMATION GAIN

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

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *p*.
  - *S* is a collection of training examples

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

  - p_+ the proportion of positive examples in *S*
  - p_- the proportion of negative examples in *S*

Examples

$$Entropy([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$Entropy([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) = 1/2 + 1/2 = 1$$

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

- Given entropy as a measure of the impurity in a collection of training examples, the ***information gain,*** is simply the expected reduction in entropy caused by partitioning the examples according to an attribute.
- More precisely, the information gain, ***Gain(S, A)*** of ***an*** attribute **A,** relative to a collection of examples ***S,*** is defined as,

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

- where ***Values(A)*** is the set of all possible values for attribute A, and S_v, is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$)

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

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

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

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

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

$$= Entropy(S) - (8/14) Entropy(S_{Weak}) - (6/14) Entropy(S_{Strong})$$

$$= 0.940 - (8/14)0.811 - (6/14)1.00$$

$$= 0.048$$

ID3 algorithm to create Decision Tree

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.

Invented by Ross Quinlan, ID3 uses a **top-down greedy** approach to build a decision tree. In simple words, the **top-down** approach means that we start building the tree from the top and the **greedy** approach means that at each iteration we select the best feature at the present moment to create a node.

Most generally ID3 is only used for classification problems with nominal features only.

ID3 algorithm to create Decision Tree

Day	Outlook	Temp	Humidity	Wind	PlayTennis
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

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

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

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

$$S_{Sunny} \leftarrow [2+, 3-] \qquad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$$

$$S_{Overcast} \leftarrow [4+, 0-] \qquad Entropy(S_{Overcast}) = -\frac{4}{4}log_2\frac{4}{4} - \frac{0}{4}log_2\frac{0}{4} = 0$$

$$S_{Rain} \leftarrow [3+, 2-] \qquad Entropy(S_{Rain}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$$

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

$$Gain(S, Outlook)$$

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

$$- \frac{5}{14}Entropy(S_{Rain})$$

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

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

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

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

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

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

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

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

$$Gain(S, Temp)$$

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

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

$$Gain(S, Temp) = 0.94 - \frac{4}{14} 1.0 - \frac{6}{14}\ 0.9183 - \frac{4}{14} 0.8113 = 0.0289$$

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

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

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

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

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

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

$$Gain(S, Humidity)$$

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

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

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

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

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

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

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

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

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

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

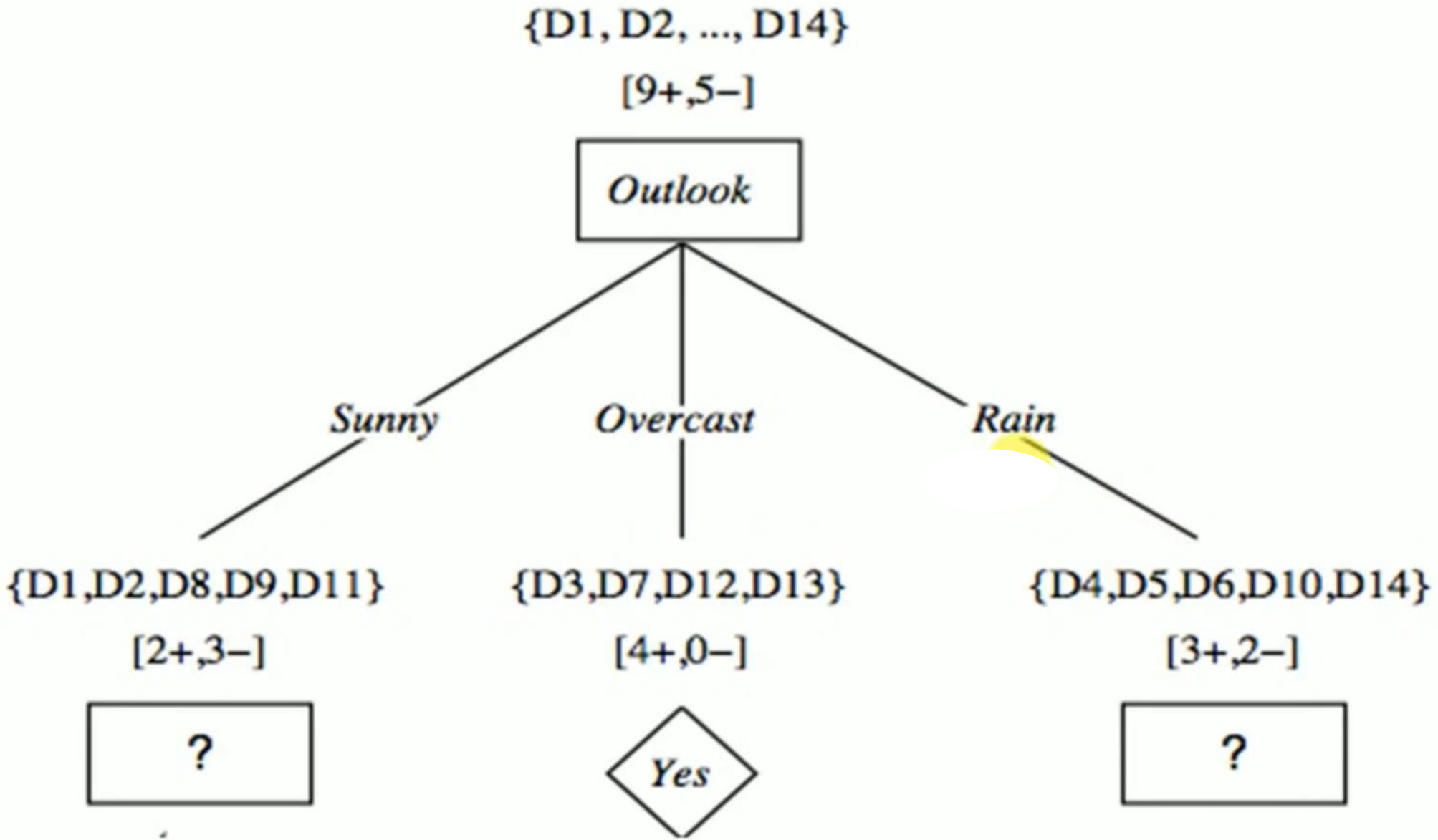
Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak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

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

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

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

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


{D1, D2, ..., D14}
[9+,5−]
Outlook
Sunny
Overcast
Rain
{D1,D2,D8,D9,D11}
[2+,3−]
?
{D3,D7,D12,D13}
[4+,0−]
Yes
{D4,D5,D6,D10,D14}
[3+,2−]
?

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

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

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

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

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

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

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

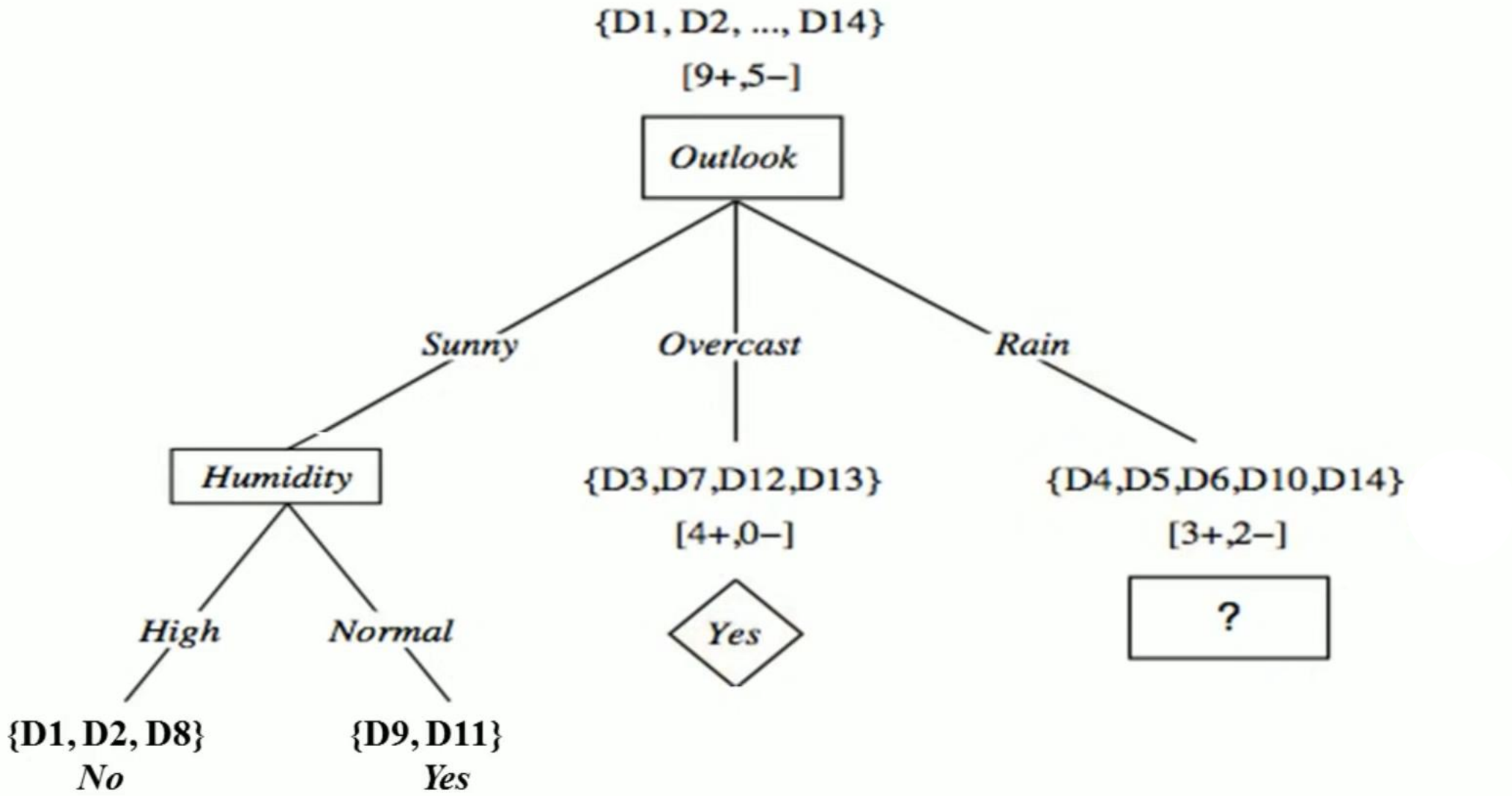
$$Gain(S_{sunny}, Wind) = 0.97 - \frac{2}{5}1.0 - \frac{3}{5}\,0.918 = 0.0192$$

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

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

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

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


{D1, D2, ..., D14}
[9+,5−]
Outlook
Sunny
Overcast
Rain
Humidity
{D3,D7,D12,D13}
[4+,0−]
{D4,D5,D6,D10,D14}
[3+,2−]
High
Normal
Yes
?
{D1, D2, D8}
No
{D9, D11}
Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

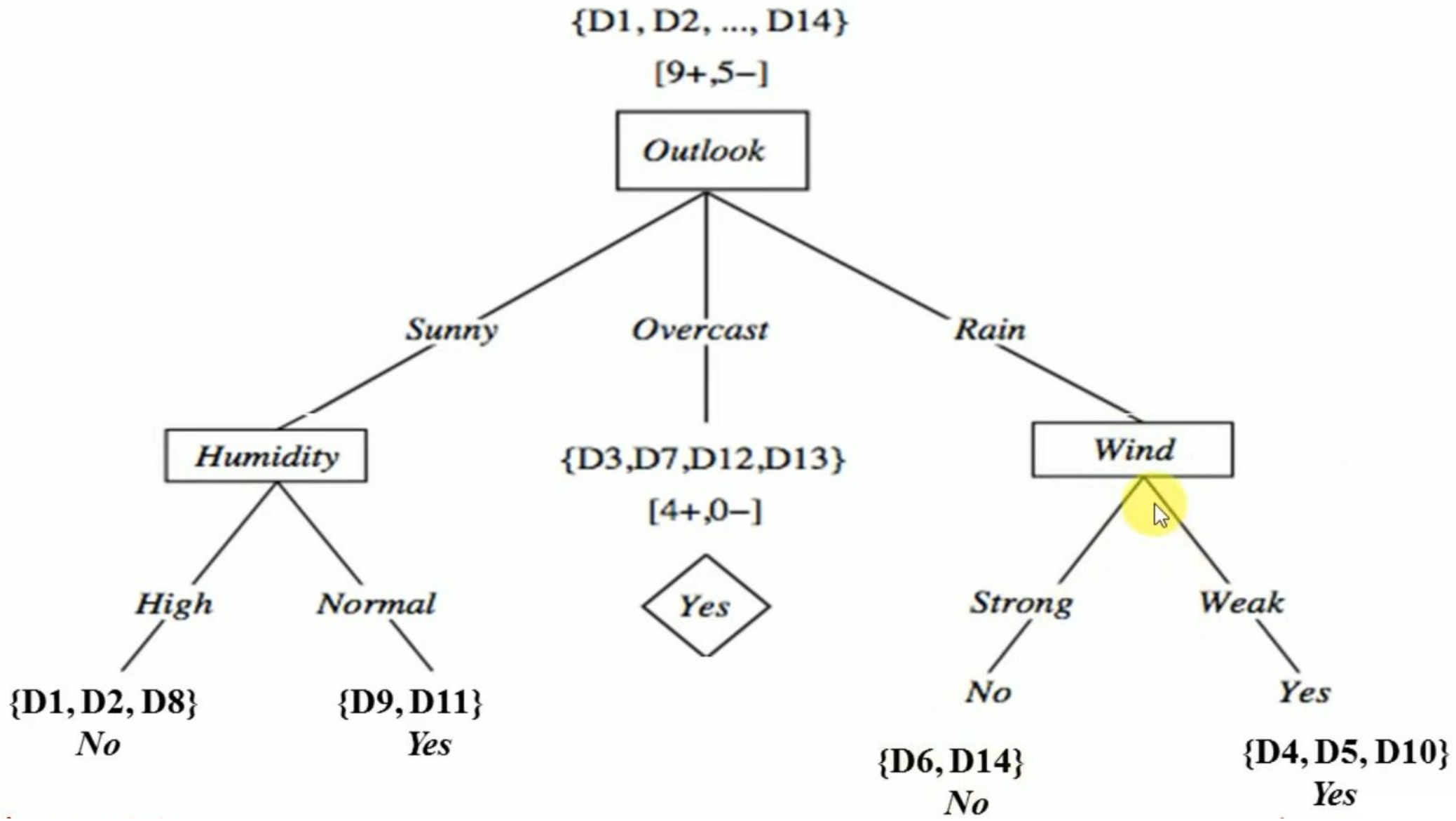
$$Gain(S_{Rain}, Wind) = 0.97 - \frac{2}{5}\ 0.0 - \frac{3}{5}0.0 = 0.97$$

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

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

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

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


{D1, D2, ..., D14}
[9+,5−]
Outlook
Sunny
Overcast
Rain
Humidity
{D3,D7,D12,D13}
[4+,0−]
Wind
High
Normal
Yes
Strong
Weak
{D1, D2, D8}
No
{D9, D11}
Yes
No
Yes
{D6, D14}
No
{D4, D5, D10}
Yes

Entropy

1. **Definition**: Entropy is a measure from information theory that quantifies the amount of impurity or disorder in a dataset. It is based on the concept of entropy from thermodynamics and information theory.
2. **Formula**: The entropy $H(S)$ for a binary classification problem is given by:

$$H(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$

   where:

   - p_1 is the proportion of positive class instances in the dataset,
   - p_2 is the proportion of negative class instances in the dataset.

3. **Range**: Entropy ranges from 0 (completely pure node, where all instances belong to a single class) to 1 (completely impure node, with an equal number of instances in each class in a binary classification).
4. **Use**: Entropy is used in the ID3 and C4.5 algorithms for selecting the best split. The decision tree aims to minimize entropy after splitting, leading to more homogeneous nodes.

Gini Index

1. **Definition:** The Gini Index (or Gini impurity) measures the probability of misclassifying a randomly chosen element if it were randomly labeled according to the distribution of labels in the dataset. It is a measure of impurity or impurity of a dataset.
2. **Formula:** The Gini Index $G(S)$ for a binary classification problem is given by:

$$G(S) = 1 - (p_1^2 + p_2^2)$$

where:

- p_1 is the proportion of positive class instances in the dataset,
- p_2 is the proportion of negative class instances in the dataset.

3. **Range:** Gini Index ranges from 0 (perfect purity) to 0.5 (maximum impurity in a binary classification). For multiclass classification, the maximum impurity can be higher.
4. **Use:** The Gini Index is used in the CART (Classification and Regression Trees) algorithm for decision trees. It selects splits by minimizing the Gini impurity, leading to purer nodes.

Example: Binary Classification with Gini Index

Suppose we have a dataset of 10 samples, and each sample belongs to either Class A or Class B. The dataset is as follows:

~~Class A: 6 samples~~

Class B: 4 samples

We want to calculate the Gini Index of this dataset to understand its impurity.

Step-by-Step Calculation

1. **Calculate the Proportions:**
   - p_A (proportion of Class A) = Number of samples in Class A / Total number of samples = $\frac{6}{10} = 0.6$
   - p_B (proportion of Class B) = Number of samples in Class B / Total number of samples = $\frac{4}{10} = 0.4$
2. **Apply the Gini Index Formula:**

The formula for the Gini Index is:

$$Gini(S) = 1 - \sum(p_i^2)$$

For a binary classification, the Gini Index becomes:

$$Gini(S) = 1 - (p_A^2 + p_B^2)$$

3. **Compute the Gini Index:**

$$Gini(S) = 1 - (0.6^2 + 0.4^2)$$

$$Gini(S) = 1 - (0.36 + 0.16)$$

$$Gini(S) = 1 - 0.52$$

$$Gini(S) = 0.48$$

- The Gini Index of 0.48 indicates the impurity of the dataset. A Gini Index of 0 would mean perfect purity (all samples belong to a single class), while a Gini Index closer to 0.5 (in binary classification) represents higher impurity.
- This value suggests that the dataset is somewhat impure since it contains both Class A and Class B samples.

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

- Compute the Gini Index for the overall collection of training examples.
- There are four possible output variables Cinema, Tennis, Stay In and Shopping.
- The data has 6 instances of Cinema, 2 instances of Tennis, 1 instance of Stay In and 1 of shopping.
- $Gini(S) = 1 - [\left(\frac{6}{10}\right)^2 + \left(\frac{2}{10}\right)^2 + \left(\frac{1}{10}\right)^2 + \left(\frac{1}{10}\right)^2] = 0.58$

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

- Computation of **Gini Index for Money** Attribute
- It has **two possible values of Rich (7 examples)** and **Poor (3 examples).**
- For **Money = Poor,** there are **3 examples with "Cinema".**
- $Gini(S) = 1 - [\left(\frac{3}{3}\right)^2] = 0$ ✓
- For **Money = Rich,** there are **2 examples with "Tennis", 3 examples with "Cinema"** and **1 example with "Stay in", "Shopping" each**
- $Gini(S) = 1 - [\left(\frac{2}{7}\right)^2 + \left(\frac{3}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2] = 0.694$
- $\boldsymbol{Weighted\ Average(Money)}$

$$= 0 * \left(\frac{3}{10}\right) + 0.694 * \left(\frac{7}{10}\right) = 0.486$$

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

$$Weighted\ Average(Weather)$$

$$= 0.444 * \left(\frac{3}{10}\right) + 0.444 * \left(\frac{3}{10}\right) + 0.375 * \left(\frac{4}{10}\right)$$

$$= 0.416$$

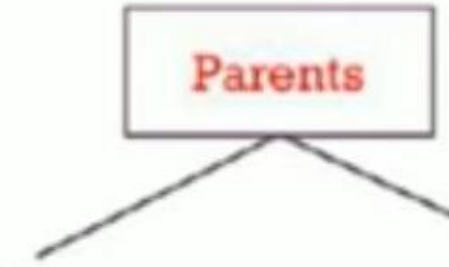
For Weather - Gini Index: 0.416

For Parents - Gini Index: 0.36

For Money - Gini Index: 0.486

Parents is selected as it has smallest Gini index.

Decision Tree using Gini Index



Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W6	Rainy	Yes	Poor	Cinema
W9	Windy	Yes	Rich	Cinema

Weekend	Weather	Parents	Money	Decision
W2	Sunny	No	Rich	Tennis
W5	Rainy	No	Rich	Stay In
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W10	Sunny	No	Rich	Tennis

Decision Tree Classification Algorithm

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Decision Tree Classification Algorithm

Advantages of the Decision Tree

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

Disadvantages of the Decision Tree

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

Support Vector Machine Algorithm

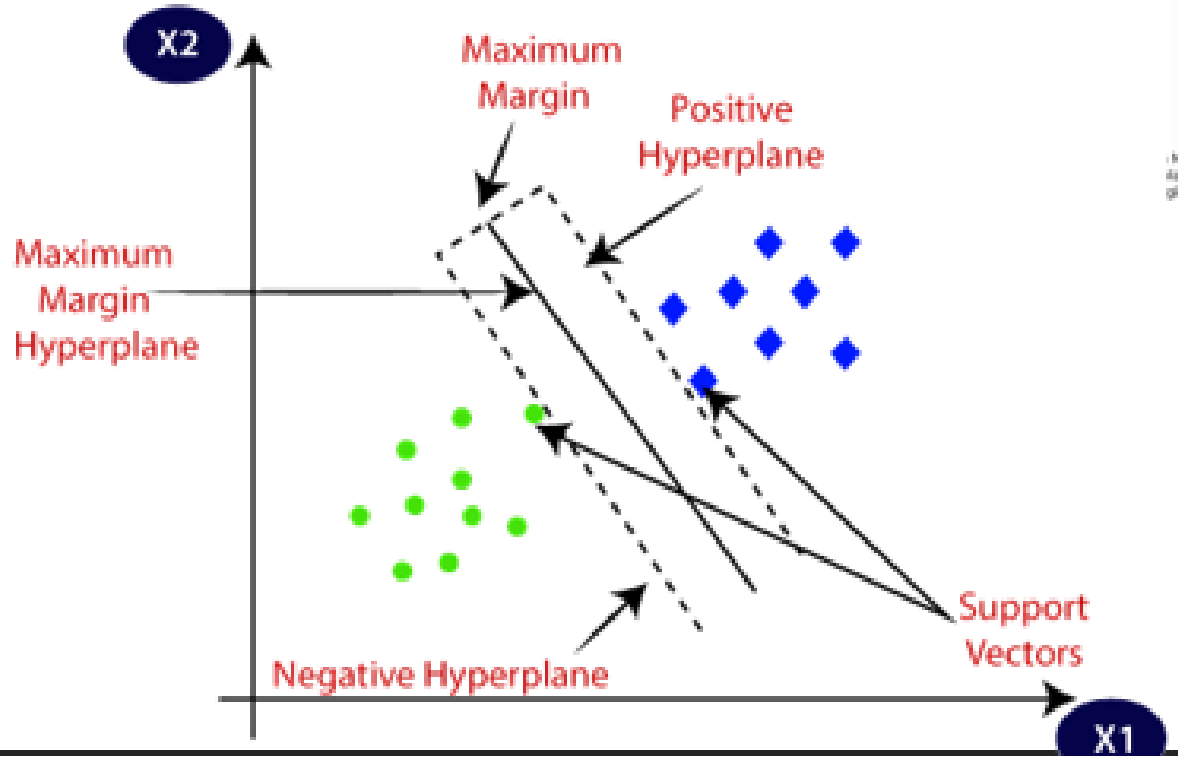
Support Vector Machine Algorithm

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.

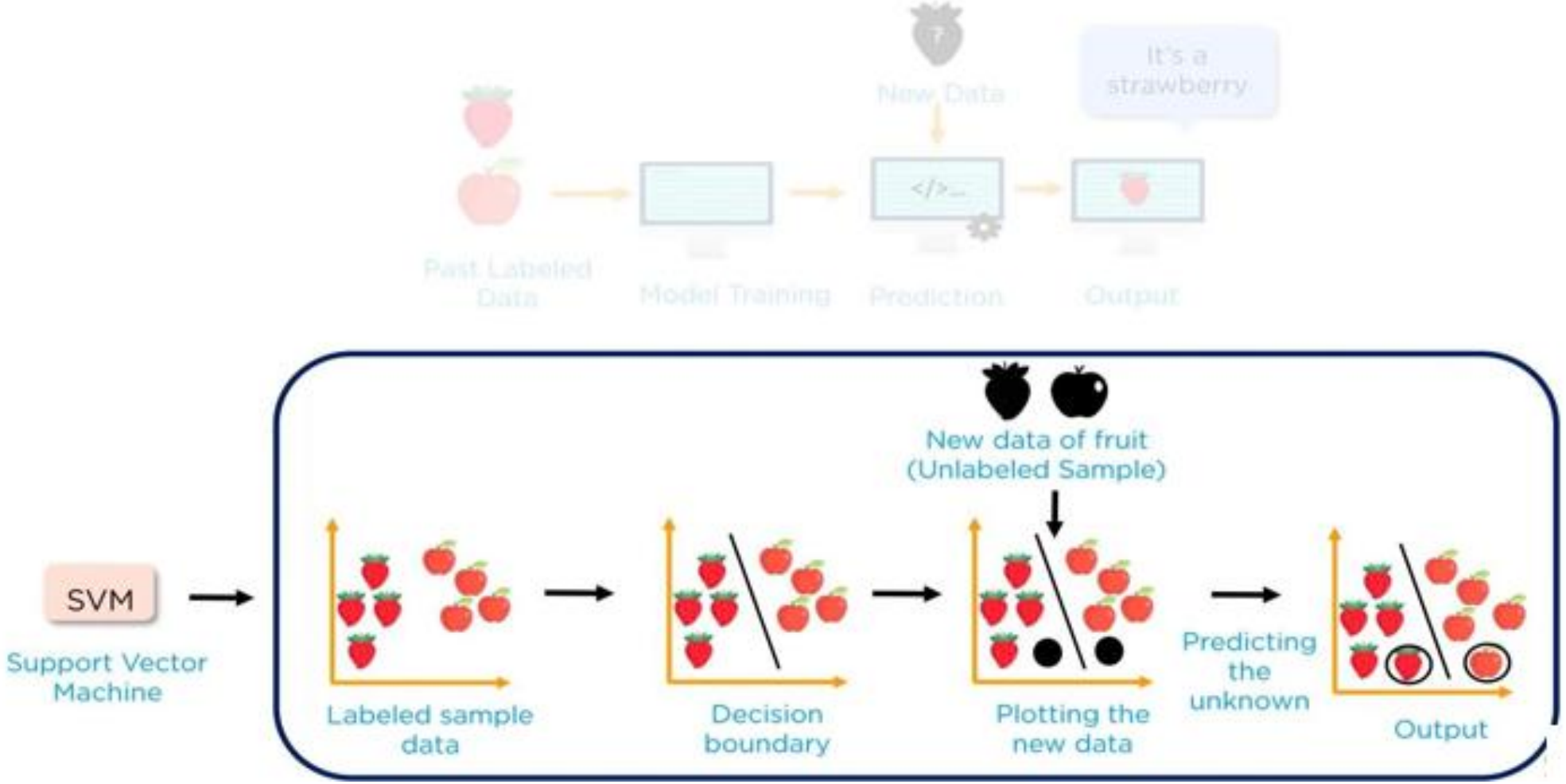
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

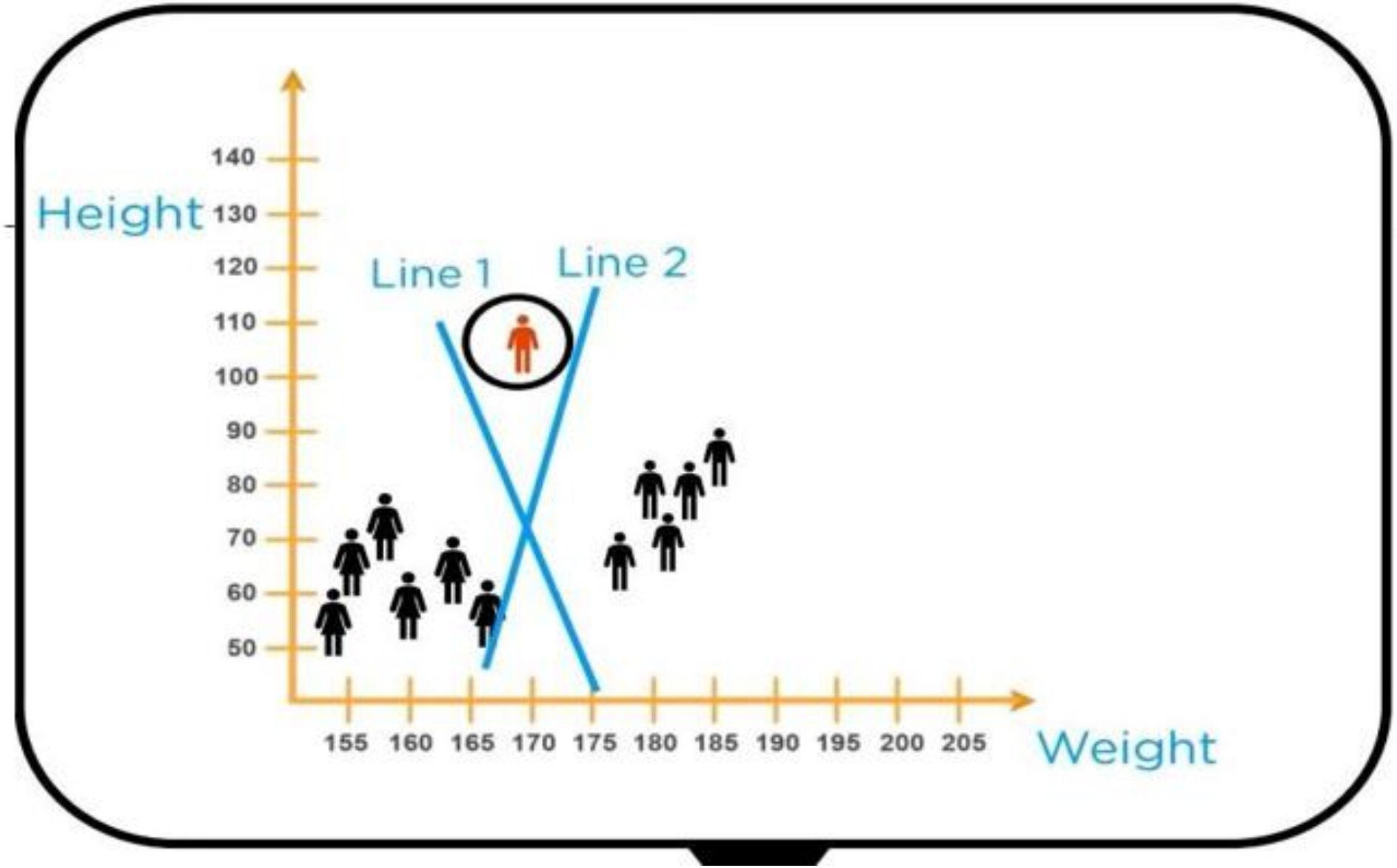
Support Vector Machine Algorithm

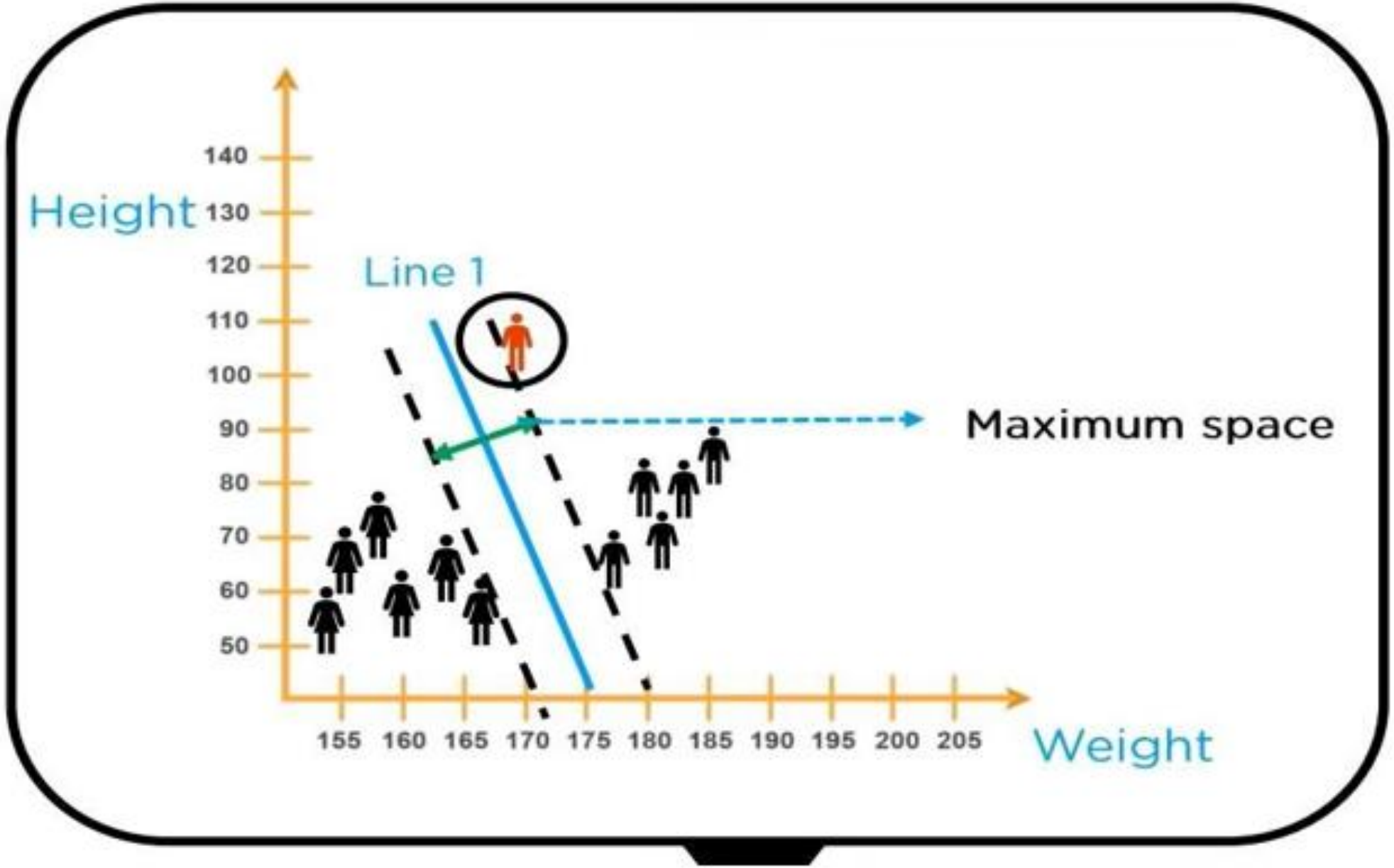
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

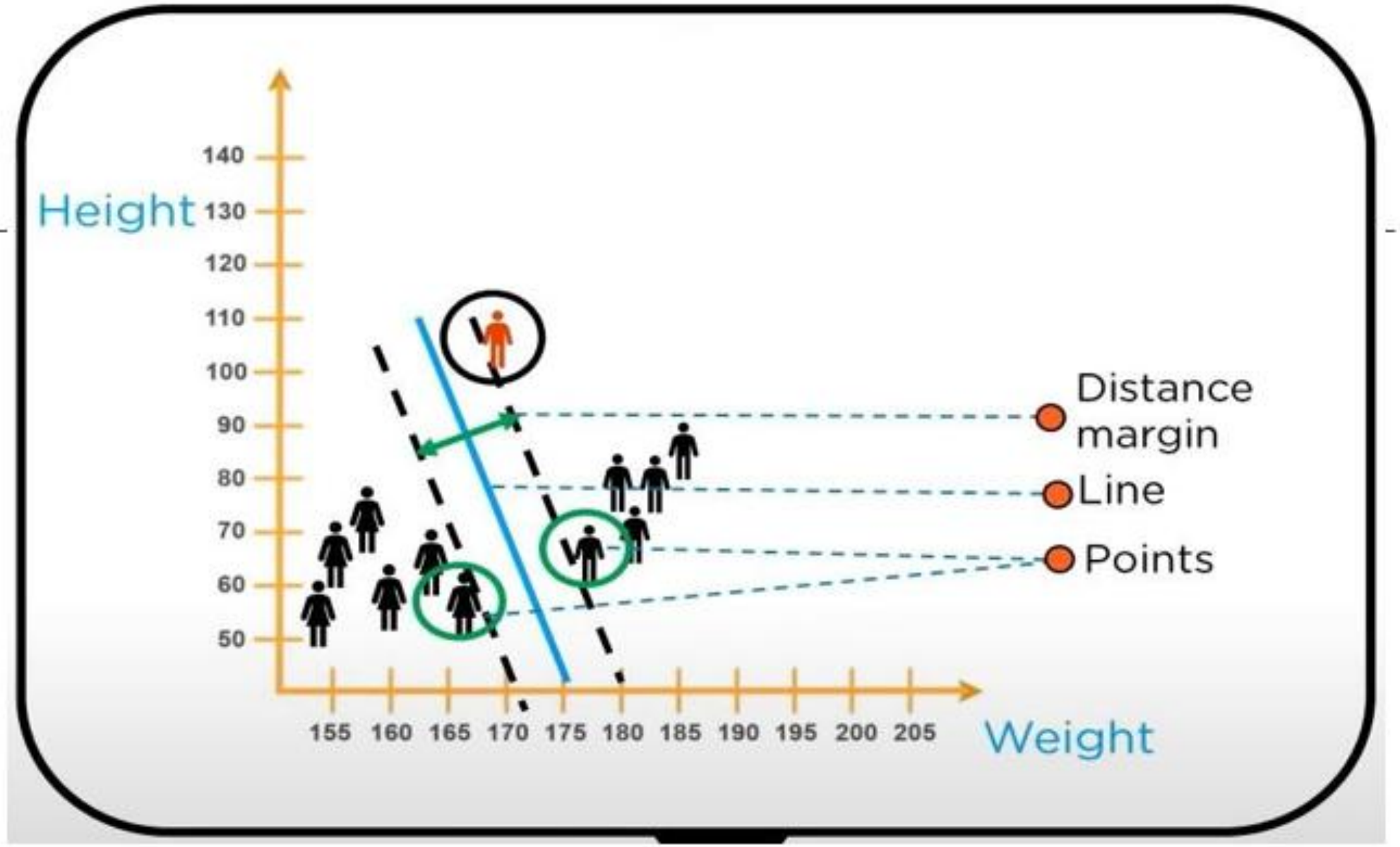


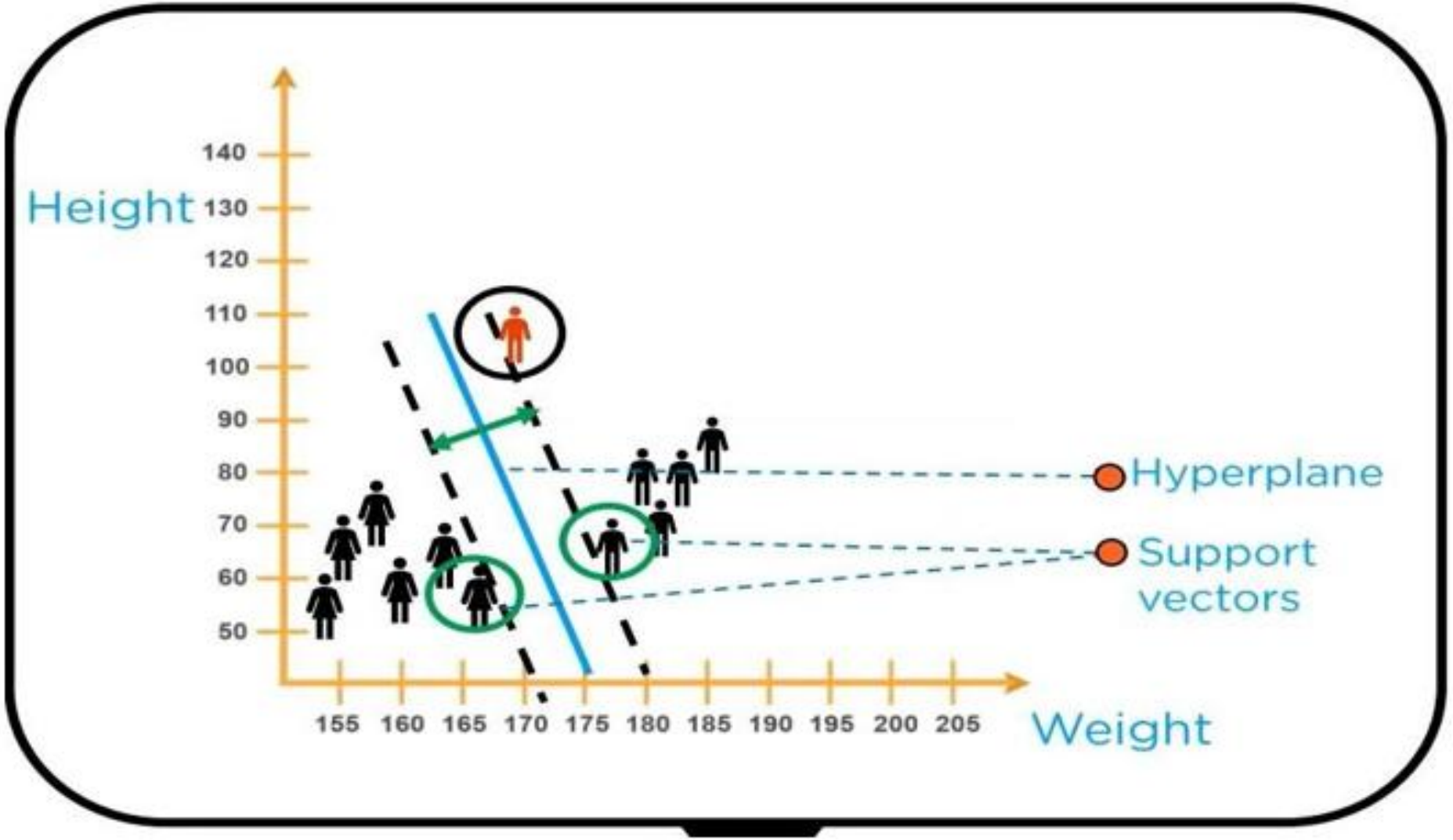
Why Support Vector Machine?

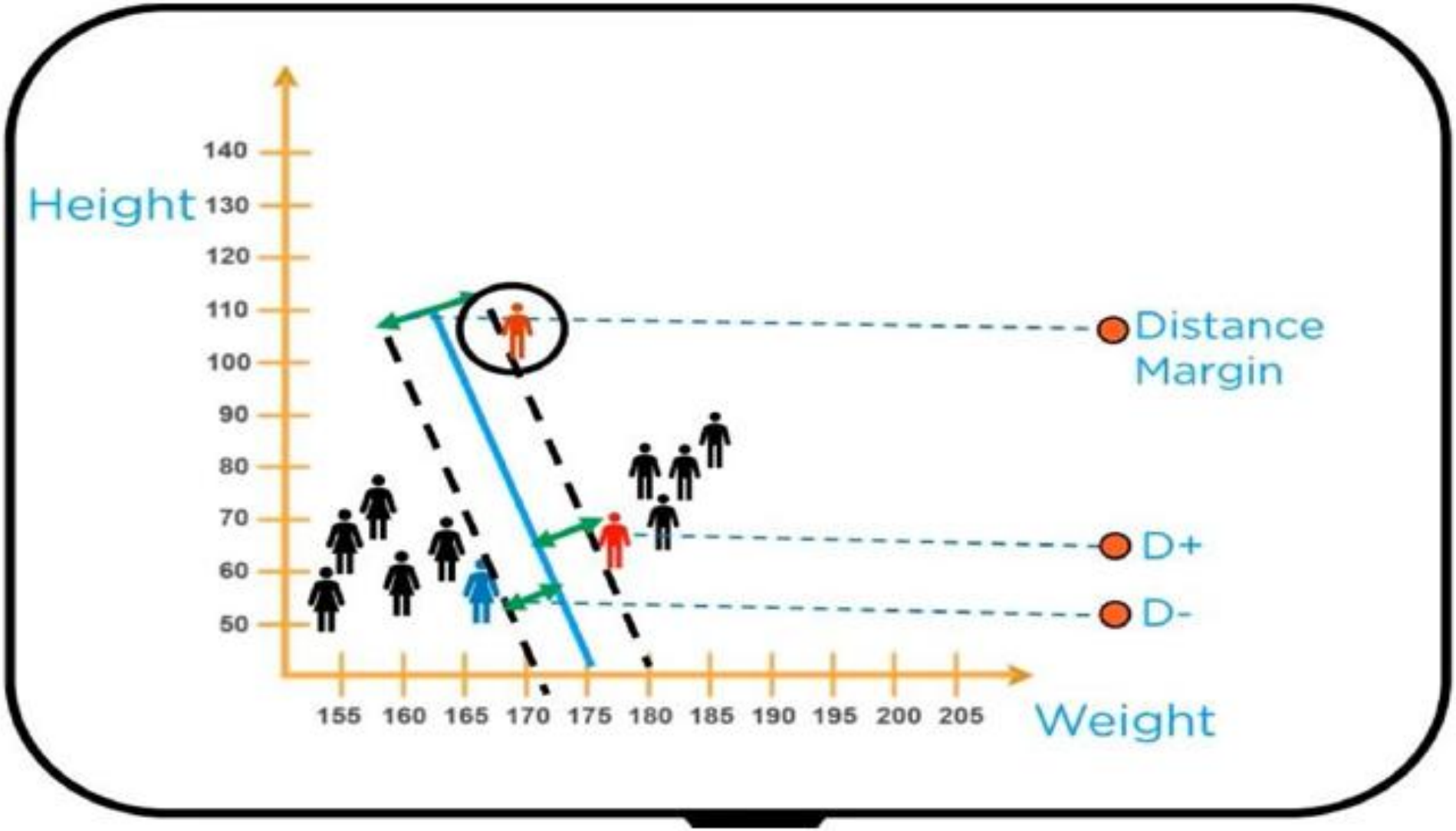



Height
140
130
120
110
100
90
80
70
60
50
Line 1
Line 2
155
160
165
170
175
180
185
190
195
200
205
Weight


Height
140
130
120
110
100
90
80
70
60
50
Line 1
Maximum space
155
160
165
170
175
180
185
190
195
200
205
Weight


Height
140
130
120
110
100
90
80
70
60
50
155
160
165
170
175
180
185
190
195
200
205
Weight
Distance margin
Line
Points


Height
140
130
120
110
100
90
80
70
60
50
155
160
165
170
175
180
185
190
195
200
205
Weight
Hyperplane
Support vectors


Height
140
130
120
110
100
90
80
70
60
50
155
160
165
170
175
180
185
190
195
200
205
Weight
Distance
Margin
D+
D-

- Support Vectors: These are the data points that are closest to the hyperplane. These points influence the orientation and position of the hyperplane.
- Margin: The margin is the distance between the hyperplane and the nearest data point of either class. SVM tries to maximize this margin to improve classification confidence.
- Hard Margin: Assumes data is perfectly separable by the hyperplane. All points must lie outside the margin.
- Soft Margin: Allows some misclassification or margin violations for non-linearly separable data.
- Kernel Function: A kernel function transforms data into a higher-dimensional space to make it linearly separable.
- Types of Kernels:
  - Linear Kernel: Suitable for linearly separable data.
  - Polynomial Kernel: For curved boundaries.
  - Radial Basis Function (RBF) Kernel: Captures complex relationships.

Support Vector Machine Algorithm

Types of SVM

SVM can be of two types:

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

Support Vector Machine Algorithm

Hyperplane and Support Vectors 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 (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

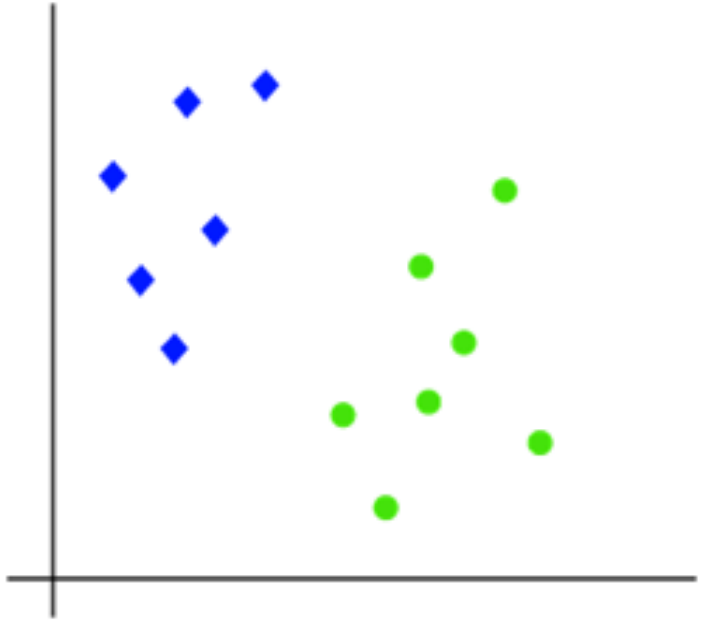
Support Vectors:

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

How does SVM works?

Linear SVM:

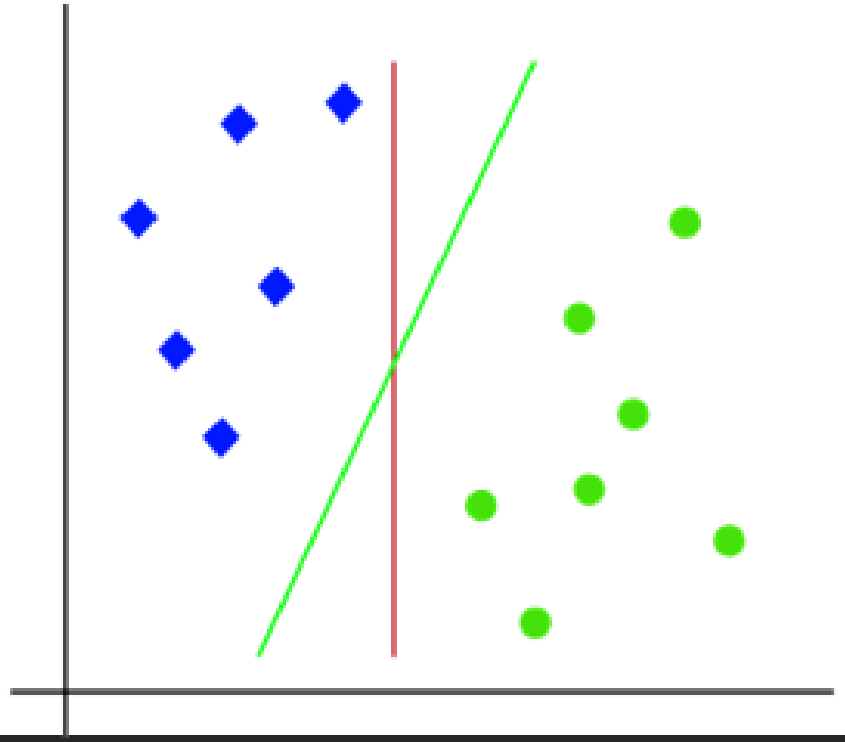
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



How does SVM works?

Linear SVM:

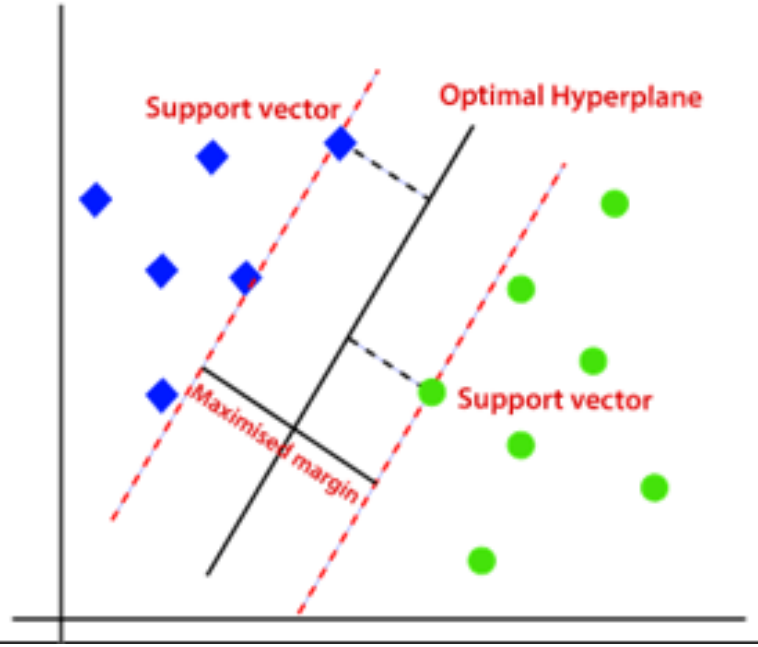
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



How does SVM works?

Linear SVM:

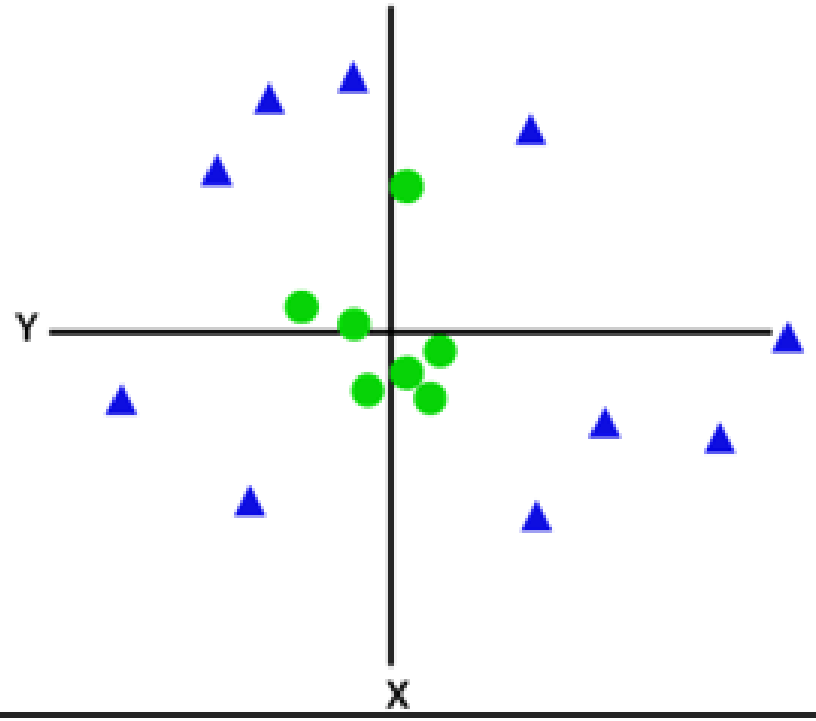
Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



How does SVM works?

Non-Linear SVM:

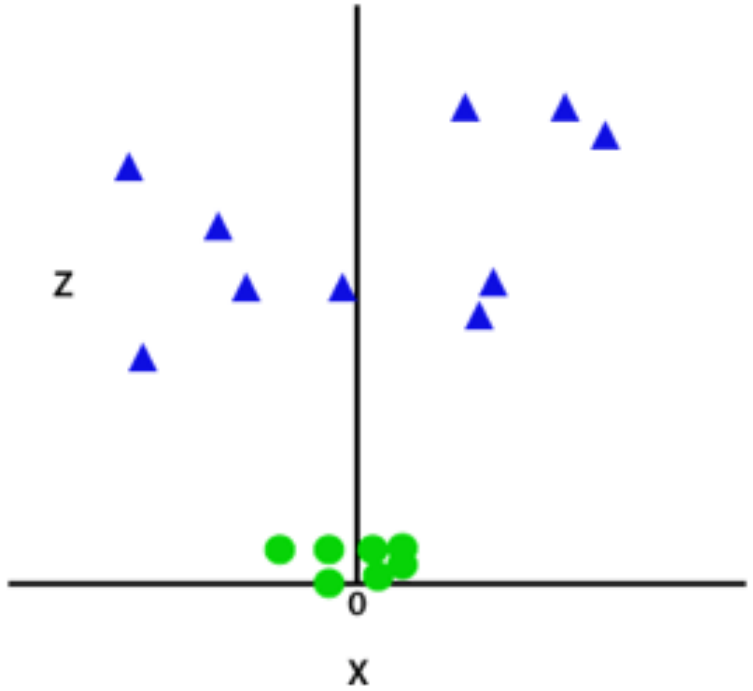
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



How does SVM works?

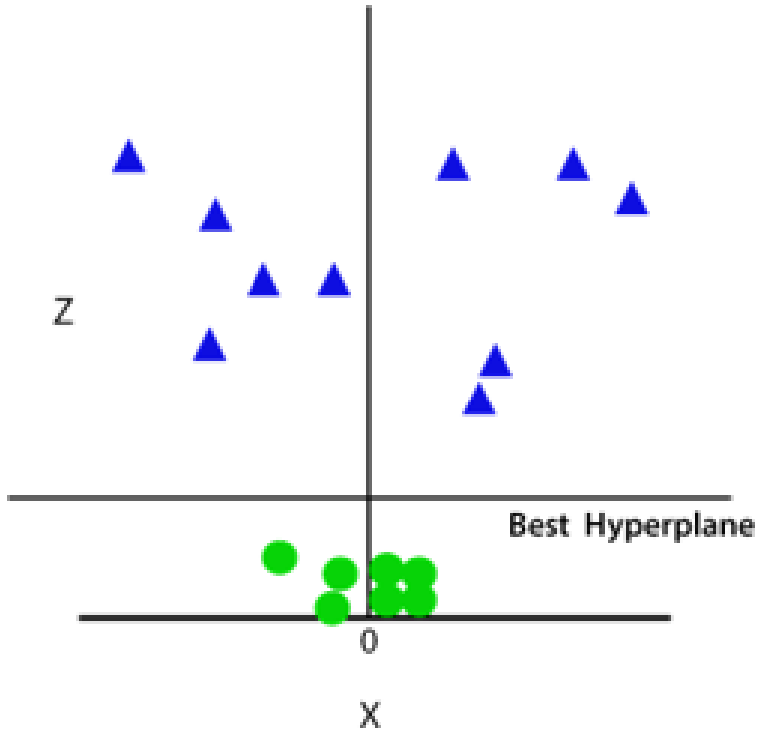
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as: $z=x^2+y^2$

By adding the third dimension, the sample space will become as below image:



How does SVM works?

So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Applications of SVM in Real World

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.

Advantages:

1. SVM works relatively well when there is a clear margin of separation between classes.
2. SVM is more effective in high dimensional spaces.
3. SVM is effective in cases where the number of dimensions is greater than the number of samples.
4. SVM is relatively memory efficient

Disadvantages:

1. SVM algorithm is not suitable for large data sets.
2. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
3. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
4. As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

For a linear regression model y=wx+b. Consider the following: Training data:

$$(1, 2), (2, 3), (3, 5)$$

Initial parameters:

$w = 0.5, b = 0.5$

Learning rate: $\alpha = 0.01$

(a) Compute the predicted values.

(b) Calculate the cost function value.

(c) Determine the updated value of w after one gradient descent step.

Model: $y = wx + b$

Training data: $(1, 2),\ (2, 3),\ (3, 5)$ Number of data points: $m = 3$

Initial parameters: $w = 0.5,\quad b = 0.5$ Learning rate: $\alpha = 0.01$

(a) Predicted values

Use the model

For $x = 1$ $\hat{y}_1 = (0.5)(1) + 0.5 = 1.0$

For $x = 2$ $\hat{y}_2 = (0.5)(2) + 0.5 = 1.5$

For $x = 3$ $\hat{y}_3 = (0.5)(3) + 0.5 = 2.0$

Predicted values: $(1.0,\ 1.5,\ 2.0)$

(b) Cost function value

Cost function for linear regression:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

Step 1: Compute errors and squared errors

Data point	y	$\hat{y}$	Error $(\hat{y} - y)$	Squared error
(1,2)	2	1.0	-1.0	1.00
(2,3)	3	1.5	-1.5	2.25
(3,5)	5	2.0	-3.0	9.00

Step 2: Sum of squared errors

$$1 + 2.25 + 9 = 12.25$$

Step 3: Apply cost formula

$$J = \frac{1}{2 \times 3} \times 12.25 = \frac{12.25}{6} = 2.0417$$

Start with the cost function $J(\theta) = \frac{1}{2m}(y - \hat{y})^2$

$\hat{y} = \theta x$

$$J(\theta) = \frac{1}{2m}(y - \theta x)^2$$

Use chain rule

$\frac{d}{d\theta}\left[\frac{1}{2m}(y - \theta x)^2\right]$

$\frac{1}{2m} \cdot \frac{d}{d\theta}(y - \theta x)^2$

$\frac{d}{d\theta}(u^2) = 2u \cdot \frac{du}{d\theta}$

$u = (y - \theta x)$

$$\frac{dJ}{d\theta} = \frac{1}{2m} \cdot 2(y - \theta x)(-x)$$

$\frac{d}{d\theta}(y - \theta x)^2 = 2(y - \theta x) \cdot \frac{d}{d\theta}(y - \theta x)$

Replace θx with $\hat{y}$:

$\frac{d}{d\theta}(y - \theta x) = -x$

$$\boxed{\frac{dJ}{d\theta} = \frac{1}{2m} \cdot 2(y - \hat{y})(-x)}$$

(c) Updated value of w after one gradient descent step

Gradient of cost function w.r.t. w:

$$\frac{\partial J}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_i$$

Step 1: Compute $(\hat{y} - y)x$

Data point	Error $(\hat{y} - y)$	x	Product
(1,2)	-1.0	1	-1.0
(2,3)	-1.5	2	-3.0
(3,5)	-3.0	3	-9.0

Step 2: Sum

$$-1 - 3 - 9 = -13$$

Step 3: Average

$$\frac{\partial J}{\partial w} = \frac{-13}{3} = -4.333$$

Step 4: Gradient descent update

$$w_{\text{new}} = w - \alpha \frac{\partial J}{\partial w}$$

$$w_{\text{new}} = 0.5 - (0.01)(-4.333)$$

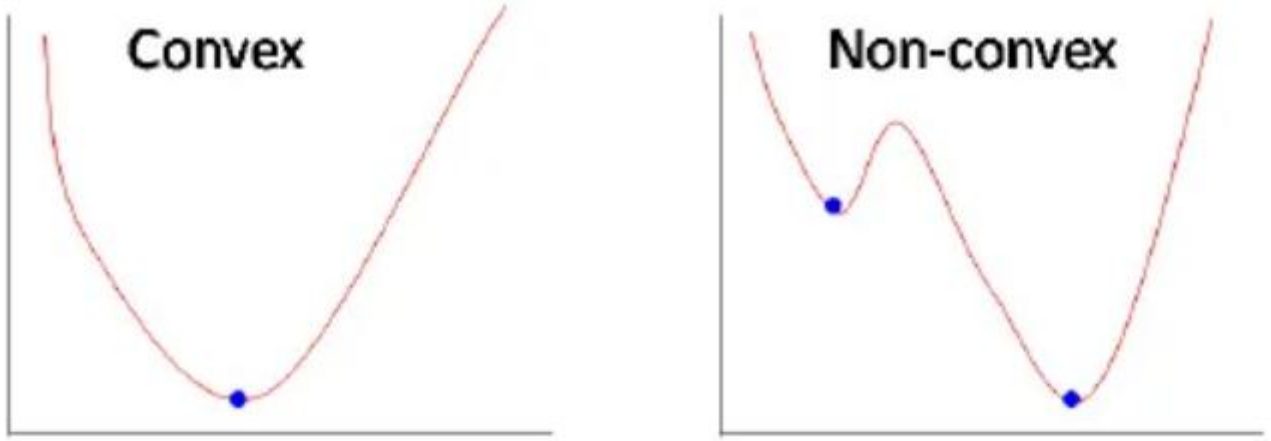
$$w_{\text{new}} = 0.5 + 0.04333 = 0.5433$$

Updated value of w:

$$\boxed{w \approx 0.543}$$

The Derivative of Cost Function for Logistic Regression

Linear regression uses **Least Squared Error** as a loss function that gives a convex loss function and then we can complete the optimization by finding its vertex as a global minimum. However for logistic regression, the hypothesis is changed, the Least Squared Error will result in a non-convex loss function with local minimums by calculating with the sigmoid function applied on raw model output.



In order to preserve the convex nature for the loss function, a log loss error function has been designed for logistic regression. The cost function is split for two cases y=1 and y=0.

For the case when we have y=1 we can observe that when hypothesis function tends to 1 the error is minimized to zero and when it tends to 0 the error is maximum. This criterion exactly follows the criterion as we wanted

In order to optimize this convex function, we can either go with gradient-descent or newtons method. For both cases, we need to derive the gradient of this complex loss function. The mathematics for deriving gradient is shown in the steps given below

The Derivative of Cost Function:

Since the hypothesis function for logistic regression is sigmoid in nature hence, The First important step is finding the gradient of the sigmoid function. We can see from the derivation below that gradient of the sigmoid function follows a certain pattern.

The function is of the form:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d(\sigma(x))}{dx} = \frac{0(1 + e^{-x}) - 1(e^{-x}(-1))}{(1 + e^{-x})^2}$$

where

$\frac{f(x)}{g(x)}$

- $f(x) = 1$
- $g(x) = 1 + e^{-x}$

$$\frac{d}{dx}\left(\frac{f}{g}\right) = \frac{f'g - fg'}{g^2}$$

- $f'(x) = \frac{d}{dx}(1) = 0$
- $g'(x) = \frac{d}{dx}(1 + e^{-x}) = e^{-x}(-1)$

$$\frac{d(\sigma(x))}{dx} = \frac{(e^{-x})}{(1+e^{-x})^2} = \frac{1\text{-}1 + (e^{-x})}{(1+e^{-x})^2} = \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}$$

$$\frac{d(\sigma(x))}{dx} = \frac{1}{1+e^{-x}} * \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))$$

Derivative of Sigmoid Function

The derivative of the sigmoid function simplifies to σ(x)(1 − σ(x)), which is computationally efficient and widely used in logistic regression and neural networks.

Consider a simple linear regression model: $y = wx$

You are given the following training dataset:

x	y
1	2
2	3
3	5

Assume:

- Initial weight $w = 1$
- Bias $b = 0$
- Regularization parameter $\lambda = 1$
- Learning rate $\alpha = 0.1$
- Number of training samples $m = 3$

(a) Compute the Ridge regression cost function.

(b) Update the weight w after one gradient descent step using Ridge regression.

(c) Compute the Lasso regression cost function.

(d) Update the weight w after one gradient descent step using Lasso regression.

Step 1: Predicted values

$$\hat{y} = wx$$

x	y	$\hat{y}$
1	2	1
2	3	2
3	5	3

Step 2: Errors and squared errors

x	y	$\hat{y}$	$\hat{y} - y$	$(\hat{y} - y)^2$
1	2	1	−1	1
2	3	2	−1	1
3	5	3	−2	4

$$\sum(\hat{y} - y)^2 = 6$$

(a) Ridge Regression Cost

Ridge cost function:

$$J_{ridge}(w) = \frac{1}{2m}\sum(\hat{y} - y)^2 + \frac{\lambda}{2m}w^2$$

Substitute values:

$$J_{ridge} = \frac{1}{6}(6) + \frac{1}{6}(1^2)$$

$$J_{ridge} = 1 + 0.1667 = 1.1667$$

(b) Ridge Weight Update

Gradient of Ridge cost:

$$\frac{\partial J}{\partial w} = \frac{1}{m}\sum(\hat{y} - y)x + \frac{\lambda}{m}w$$

Compute gradient:

$$(-1)(1) + (-1)(2) + (-2)(3) = -9$$

$$\frac{-9}{3} = -3$$

Regularization term:

$$\frac{1}{3}(1) = 0.333$$

Total gradient:

$$-3 + 0.333 = -2.667$$

Gradient descent update:

$$w_{new} = w - \alpha\frac{\partial J}{\partial w}$$

$$w_{new} = 1 - 0.1(-2.667) = 1.267$$

(c) Lasso Regression Cost

Lasso cost function:

$$J_{lasso}(w) = \frac{1}{2m}\sum(\hat{y} - y)^2 + \frac{\lambda}{2m}|w|$$

$$J_{lasso} = \frac{1}{6}(6) + \frac{1}{6}(1)$$

$$J_{lasso} = 1 + 0.1667 = 1.1667$$

(d) Lasso Weight Update

Gradient of Lasso cost:

$$\frac{\partial J}{\partial w} = \frac{1}{m}\sum(\hat{y} - y)x + \frac{\lambda}{m}\text{sign}(w)$$

Since $w = 1 > 0$:

$$\text{sign}(w) = +1$$

$$\frac{\partial J}{\partial w} = -3 + 0.333 = -2.667$$

Update rule:

$$w_{new} = 1 - 0.1(-2.667) = 1.267$$