



CSCN 8000

# Artificial Intelligence Algorithms and Mathematics

# Outline



- ✓ **General Introductions**
- ✓ **Review to Course Policies and Procedures-Applicable**
- ✓ **Introduction to AI**
- ✓ **About ML**
- ✓ **Scope of ML**
- ✓ **Machine Learning Project Approach-checklist**
- ✓ **Essential Technical Tools**
- ✓ **Case studies and Applications for AI**

# Learning Objectives



- ✓ Understand Course Policies and Procedures-Applicable.
- ✓ Comprehend the difference between AI and ML.
- ✓ List the different types of machine learning techniques.
- ✓ Understand the different feature types in ML systems.
- ✓ Discuss the strengths and weaknesses of ML.
- ✓ Understand the technical tools available in the course for implementation of ML algorithms.

# Program Handbook



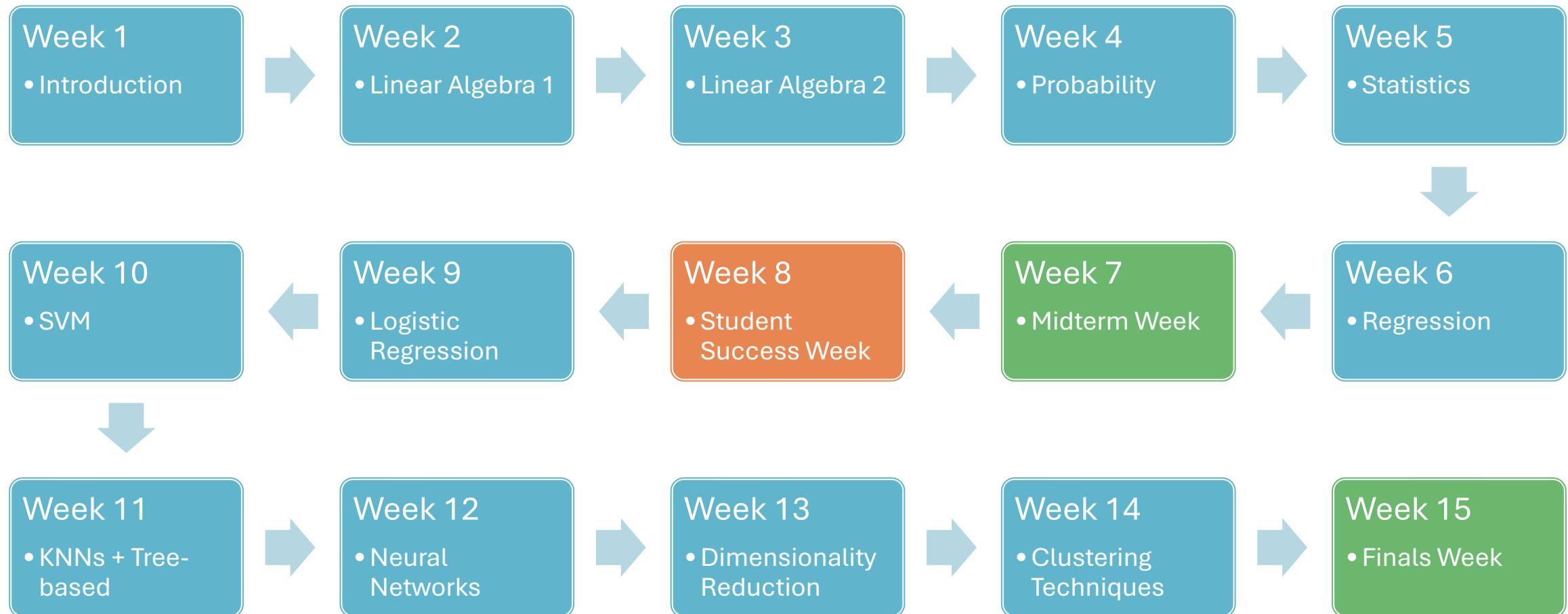
- <https://www.conestogac.on.ca/handbook/1557>

# Resources



- **Practical Statistics for Data Scientists, 2nd Edition** Peter Bruce, Andrew Bruce, Peter Gedeck
- **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition** By Aurélien Géron

# Course Content



# Course Deliverables



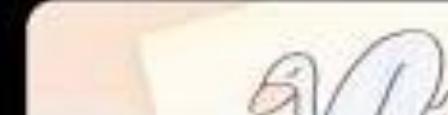
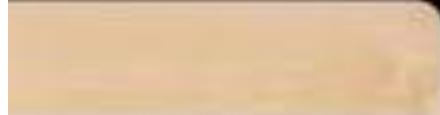
- 3 Labs → 10% each
- 3 Assignments → 10% each
- Midterm (20%):
  - Midterm Part A: Theoretical → 10%
  - Midterm Part B: Practical → 10%
- Final (20%):
  - Final Part A: Theoretical → 10%
  - Final Part B: Practical → 10%

# Online Session Rules



- Attendance is mandatory and highly encouraged to properly understand the complex mathematical topics and solve the assignments/exams.
- Cameras should always be turned on unless prior approval has been given for special circumstances.
- The online sessions are supposed to be interactive and collaborative so feel free to ask questions or open discussions whenever possible.
- All sessions will be recorded. By participating, you consent to being recorded.





# Introduction to Artificial Intelligence



- Artificial Intelligence is an area of computer science, where the goal is to enable computers and machines to perform human like tasks and simulate human behavior.
- Machine Learning is subset of AI that tries to solve a specific problem and make predictions using data.
- Data Science is a field that attempts to find patterns and draw insights from the data.
- Mostly commonly driven tool in AI is Machine Learning

# Categorization



- AI can be categorized into ANI and AGI
- ANI –Artificial Narrow Intelligence –Example : Self Driving Car, Virtual Assistants
- AGI- Mimicking the Human Ability



# What is Machine Learning



- Subdomain of computer science that focusses on algorithms which help a computer learn **from data** without explicit programming



**ChatGPT**



**BARD AI**



**Stable Diffusion**

# Types of Machine Learning



- Supervised Learning – Using Labelled Inputs – Outputs

- Example : Identifying Pictures



CAT



DOG

- Unsupervised Learning-Uses Unlabelled data to identify patterns in the data

- Example : Clustering the Pictures



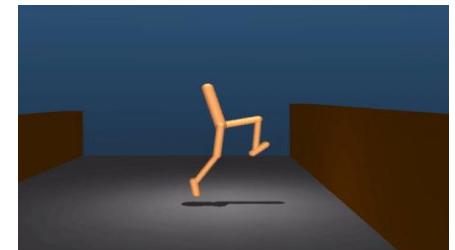
Structure 1



Structure 2

- Reinforcement Learning

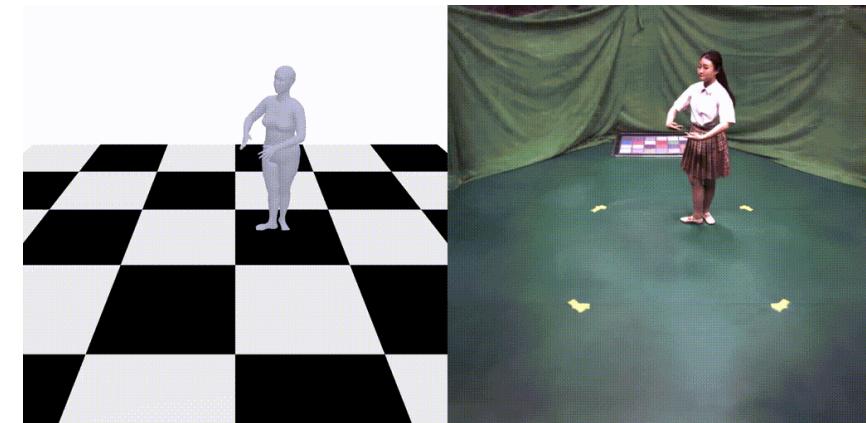
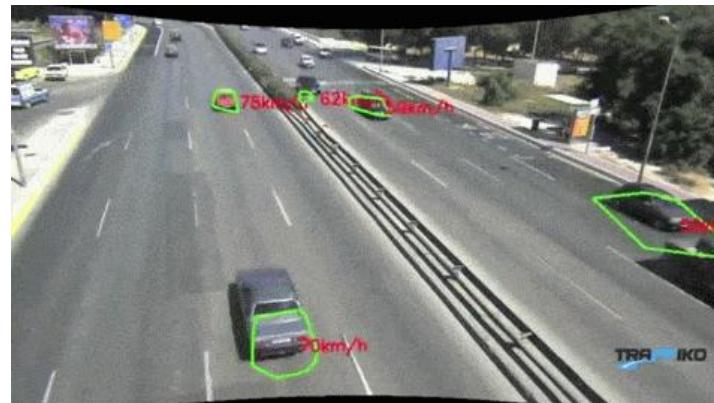
- Agent learning in interactive environment depending on rewards and penalties.



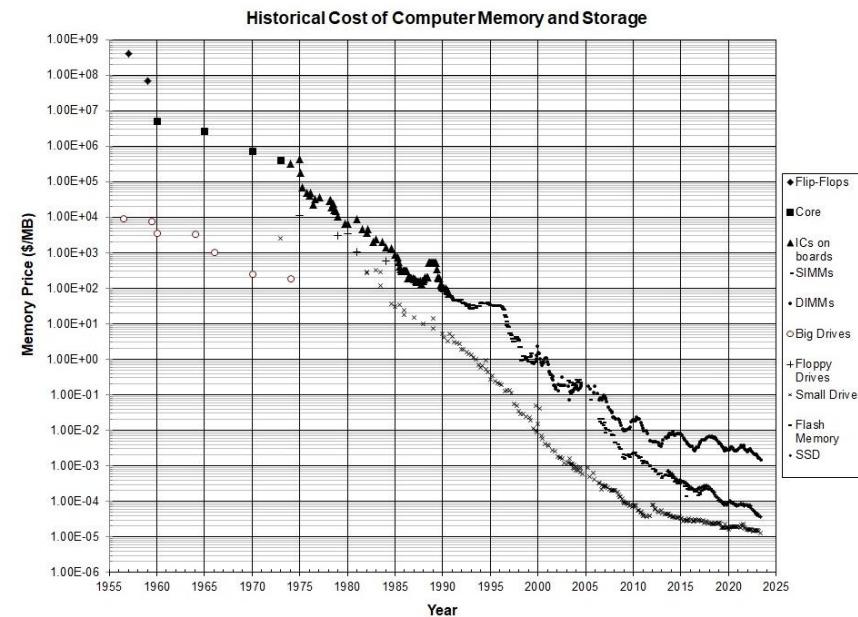
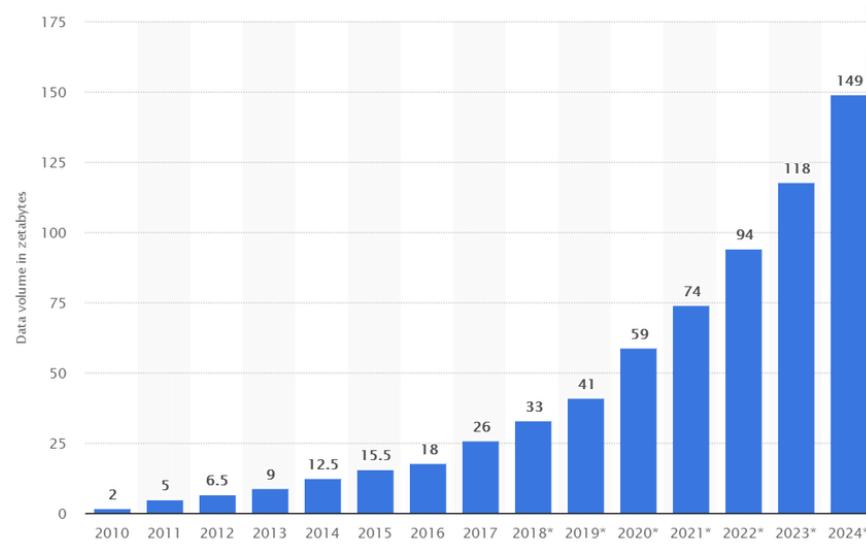
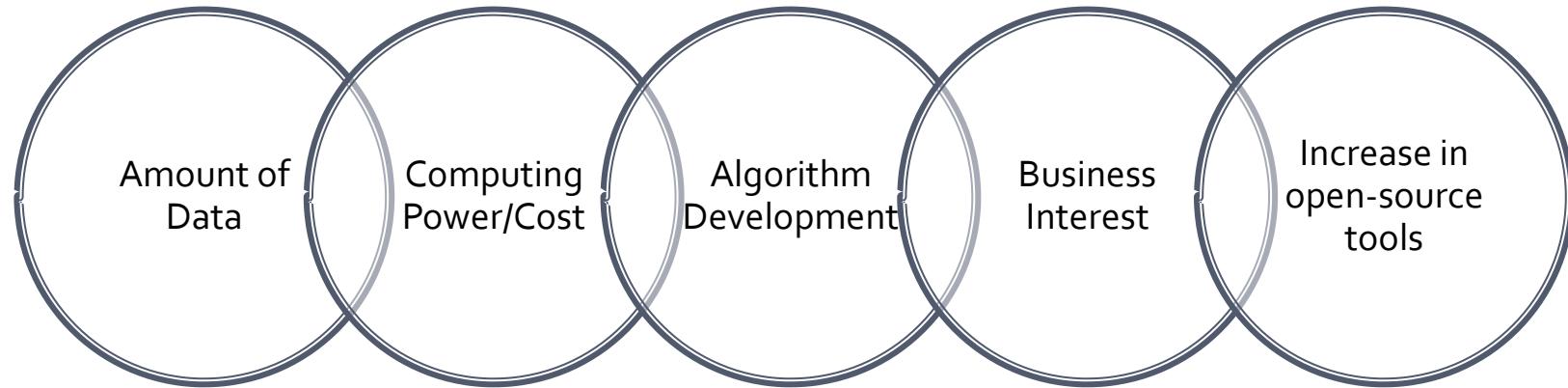
# Supervised Learning Applications



- Spam Filtering -> Input – email ,Output – Spam
- Speech Recognition -> Input – Audio ,Output – Text Transcript
- Language Translation -> Input – English ,Output – Chinese
- Self-driving Car -> Input – image, radar ,Output – Position of the cars



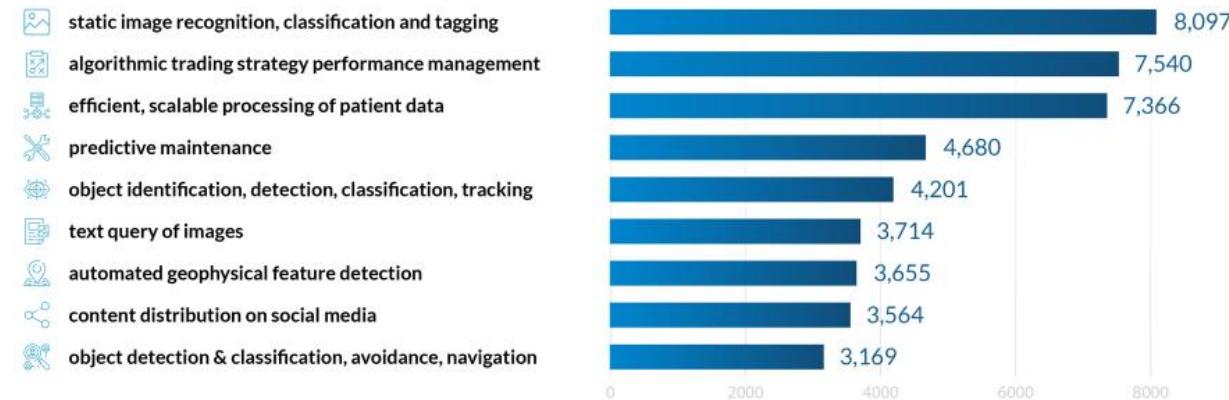
# Why is ML booming?



1

## Global AI revenue forecast by 2025, ranked by use case in millions US dollar

Source: Statista



2

## Penetration of artificial intelligence skills, by country

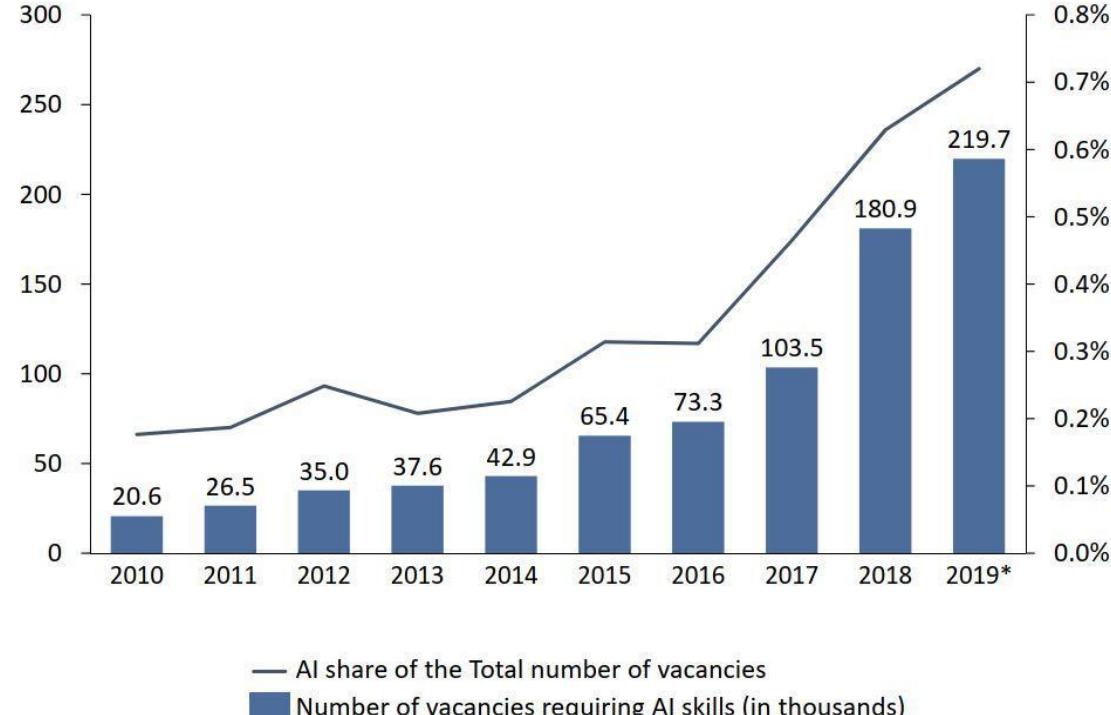
Source: Dun &amp; Bradstreet



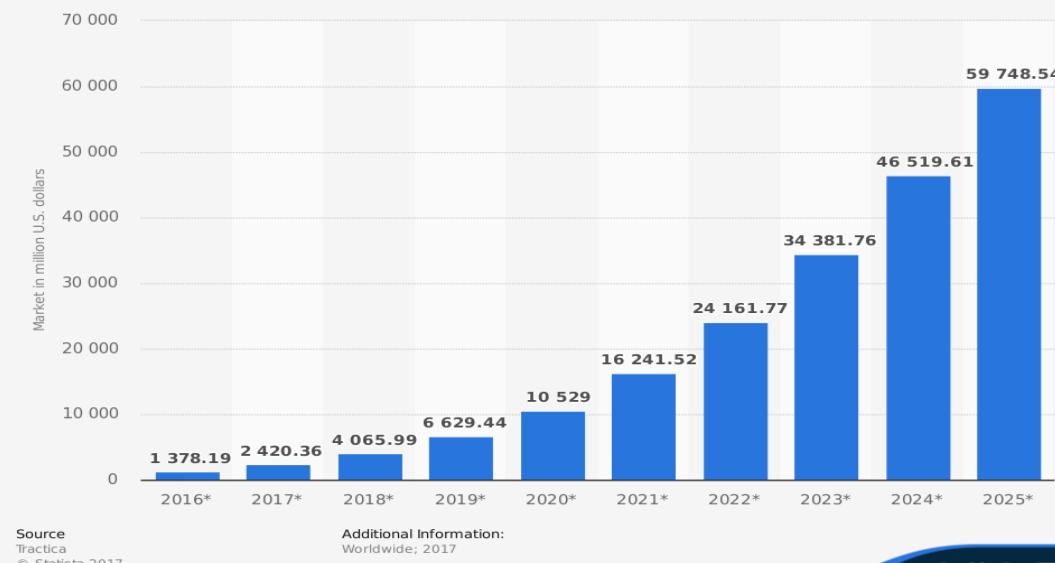
3

## Organizations deploying AI, by functional areas

Source: Medium

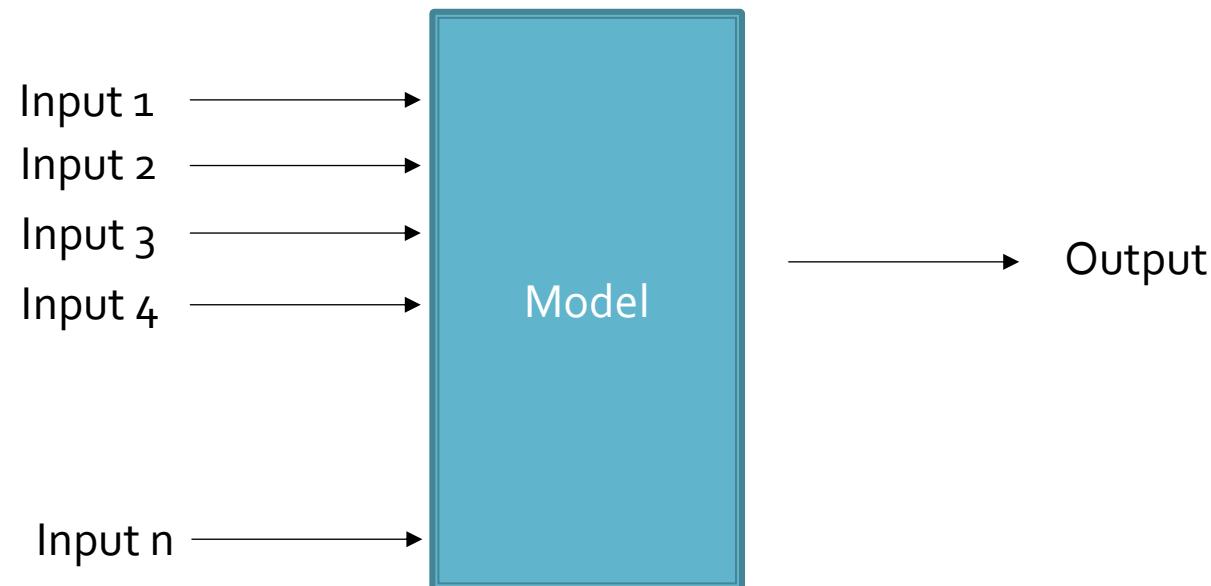


## Revenues from the artificial intelligence (AI) market worldwide, from 2016 to 2025 (in million U.S. dollars)





# Machine Learning Model



# Features: Qualitative Data Types



- Qualitative: Categorical data with finite number of categories.
  - Nominal Data: No inherent order
    - E.g. Red/Blue/Green
  - Ordinal Data: Posses meaningful order.
    - E.g. Excellent/Very Good/Good
  - Binary Data: Only two categorical values.
    - E.g. Yes/No
- One Hot Encoding: Used for processing by ML models.

[UCI Machine Learning Repository: MAGIC Gamma Telescope Data Set](#)

{USA,India ,Canada}

USA – { 1,0,0}

India – { 0,1,0}

Canada- { 0,0,1}

# Features: Quantitative Data Types



- Quantitative –Numerical Values Data that could be either discrete or continuous
  - Discrete: distinct, separate values with clear gaps
    - Example: # of Employees
  - Continuous: can take on any value within a range
    - Ratio data: has a natural order **and** a meaningful zero point
      - Example: Age
    - Interval data: has a natural order **but lacks** a meaningful zero point
      - Example: Temperature measured in Celsius

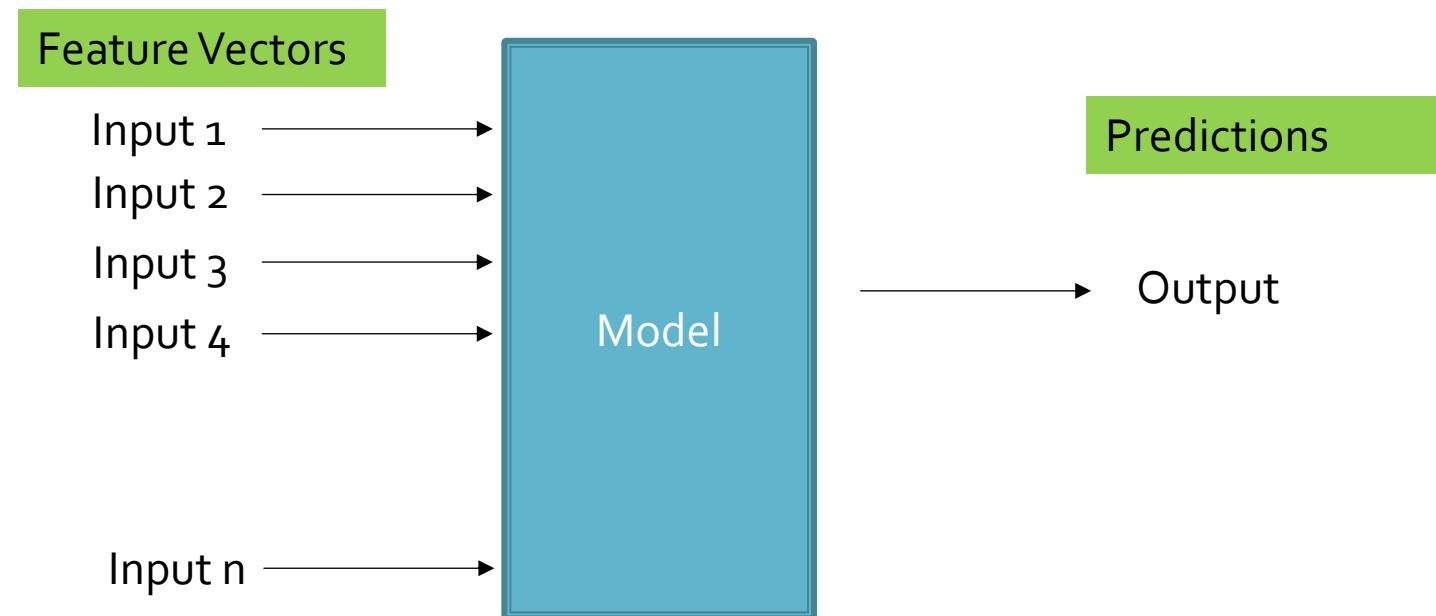
# Features: Q/A



- Bank account balance is an example of which type of data?
  - (a) Ratio
  - (b) Interval
- Which of the following is an example of nominal data?
  - a) Educational level
  - b) Types of fruits
- Is the measurement of pH (acidic or alkaline levels) considered which type of data?
  - a) Discrete
  - b) Ratio
  - c) Interval
- Which of the following is an example of interval data?
  - a) Height in centimeters
  - b) Weight in kilograms
  - c) Temperature in degrees Celsius
  - d) Distance in meters
- On a Likert scale (e.g., strongly disagree, disagree, neutral, agree, strongly agree), what type of data is obtained?
  - a) Ratio
  - b) Interval
  - (c) Ordinal



# Machine Learning Model



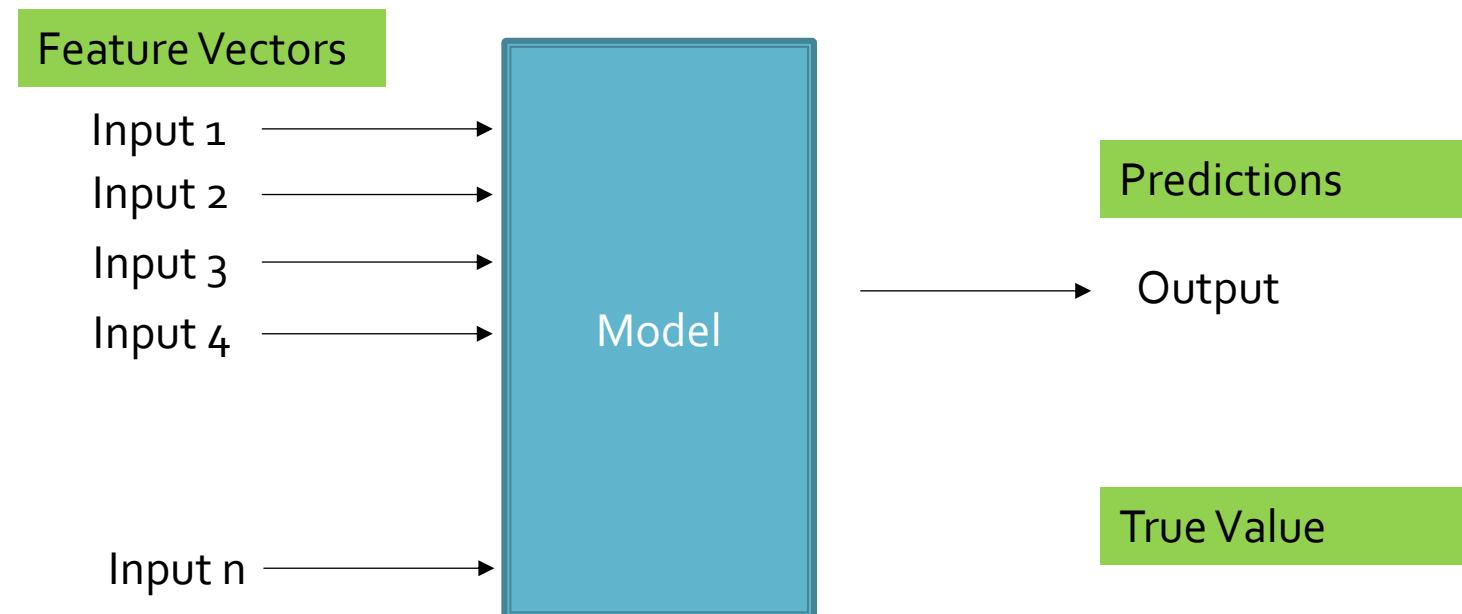
# Supervised Learning



- Classification : Predicting discrete classes /categories
  - Binary Classification(between two) ,Example : True/False, Object Identification- Cat /Not Cat
  - Multiclass classification(more than two) ,Example : Animal Species
  - Multilabel classification(more than two) ,Example : Colors in Image
- Regression :Predicting continuous values /Predicting the value that is close to the true value.
  - Examples: Price Prediction ,Temperature Prediction



# Machine Learning Model



# Data Set



- <https://www.kaggle.com/c/titanic>
- <https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents>
- [UCI Diabetes Data Set | Kaggle](https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)



- 
- Weaknesses
  - Commonsense Reasoning
  - Contextual Understanding
  - Causality and Explainability
  - Labeled Data Dependence
  - Ethical and Bias Issues
  - Robustness and Security
  - Privacy Concerns
  - Continual Learning
-

# Data Repositories



- Popular open data repositories:
  - [OpenML.org](#)
  - [Kaggle.com](#)
  - [PapersWithCode.com](#)
  - [UC Irvine Machine Learning Repository](#)
  - [Amazon's AWS datasets](#)
  - [TensorFlow datasets](#)
- Meta portals (they list open data repositories):
  - [DataPortals.org](#)
  - [OpenDataMonitor.eu](#)
- Other pages listing many popular open data repositories:
  - [Wikipedia's list of machine learning datasets](#)
  - [Quora.com](#)
  - [The datasets subreddit](#)

# AI Technical Tools



- Machine Learning Frameworks
  - PyTorch
  - Tensorflow
  - Keras
  - MXNet
  - CNTK
  - Caffe
  - PaddlePaddle
  - Scikit-Learn
  - R
  - Weka
- Research Activity
  - Arxiv
- GitHub Repositories
  - Open Source

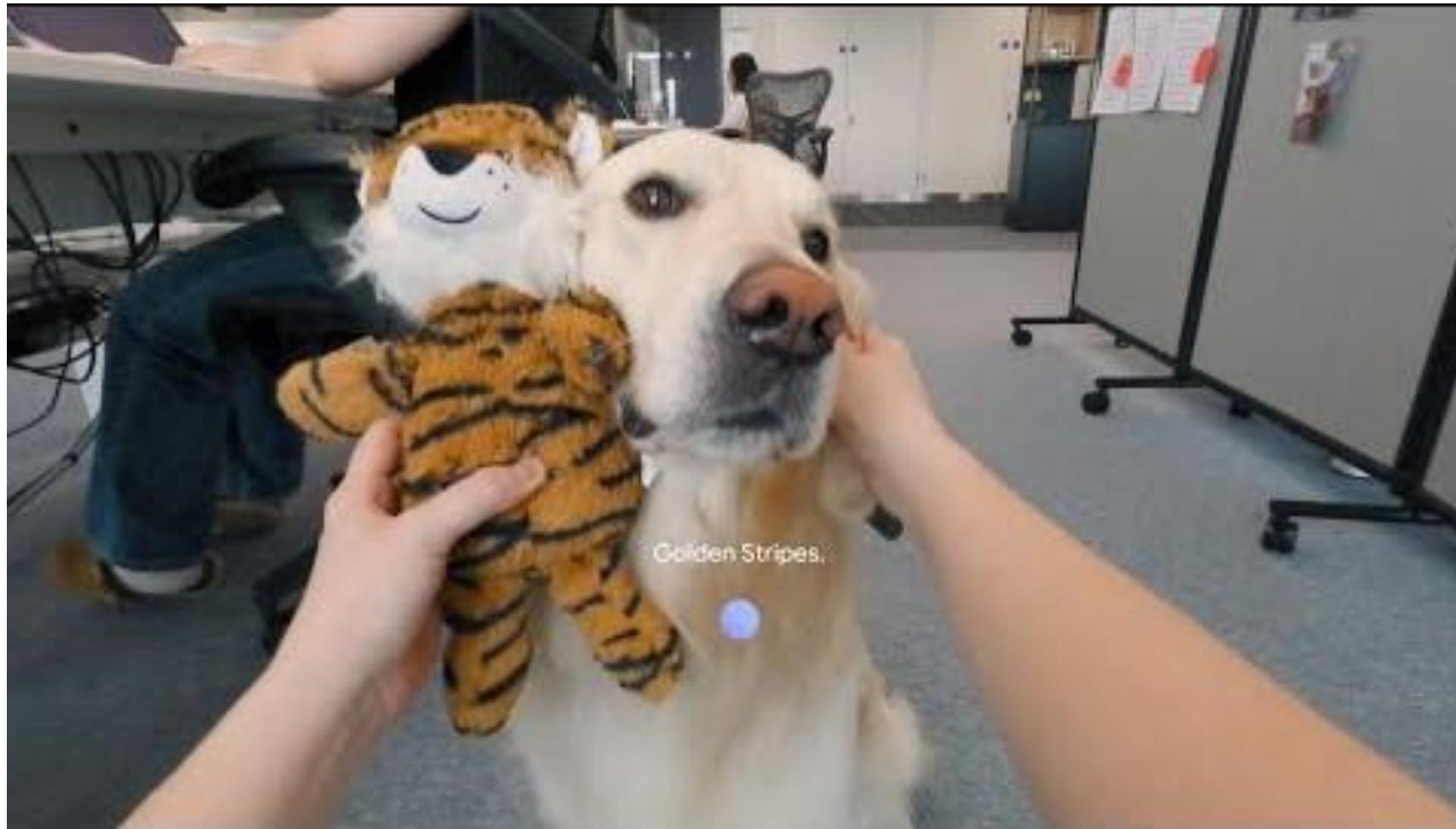
# CPU Vs GPU



- CPU –Computer Processing Unit
- GPU :Graphics Processing Unit
- Cloud Vs On Premises

# Machine Learning Project Checklist





Golden Stripes.



# Thank you!



- Open Discussion



# References



- Essential Scrum: A Practical Guide to the Most Popular Agile Process, Kenneth S. Rubin.
- Product Vision Board by Roman Pichler



Thank You  
Youssef Abdelkareem  
[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Motivation

- Vectors, Matrices, Gradients, Optimization, and Probability Distribution occurs a lot in Machine Learning.
- How and why do machine learning algorithms work?
- To build customized machine learning models.
- Debugging the machine learning algorithms.
- Understanding the maths will help you to improve the performance of the machine learning algorithms.



# Linear Algebra

---

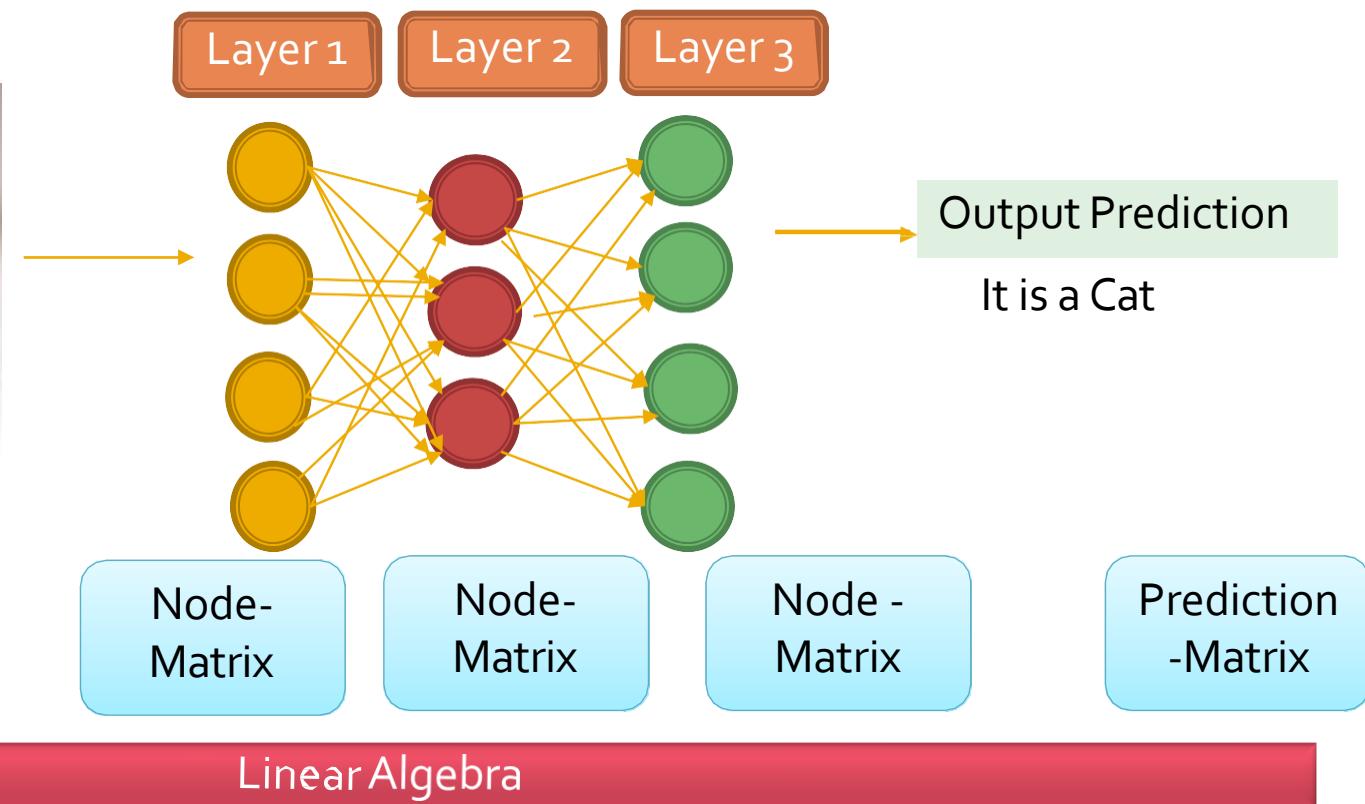
- Vectors
- Matrix
  - Inverse
  - Rank
- Linear Transformation
- Eigenvalues
- Eigenvectors
- These are used in machine learning to store and compute on data.



# Motivation to Machine Learning



- Linear Algebra most useful in machine learning
- Most popular application –Neural Networks –image recognition –using matrix operations.



# Motivation to Machine Learning



- To perform all the vector or matrix operations to go from the image on the left to the prediction on the right, we use linear algebra.
- The pixels in the image are simply values that serve as an input where the algorithm(neural networks ) performs mathematical operations to determine the output prediction.

# Vector



- A vector is a quantity defined by a magnitude and a direction. For example, a rocket's velocity is a 3-dimensional vector: its magnitude is the rocket's speed, and its direction is (hopefully) up.
- A vector can be represented by an array of numbers called *scalars*.
- Each scalar corresponds to the magnitude of the vector with regard to each dimension.
- For example, say the rocket is going up at a slight angle: it has a vertical speed of 5,000 m/s, and also a slight speed towards the East at 10 m/s, and a slight speed towards the North at 50 m/s.

$$\text{velocity} = \begin{pmatrix} 10 \\ 50 \\ 5000 \end{pmatrix}$$

# Purpose



- To represent observations and predictions.
- For example, say we built a Machine Learning system to classify videos into 3 categories (good, spam, clickbait) based on what we know about them.
- For each video, we would have a vector representing what we know about it, such as:

$$\text{video} = \begin{pmatrix} 10.5 \\ 5.2 \\ 3.25 \\ 7.0 \end{pmatrix}$$

This vector could represent a video that lasts 10.5 minutes, but only 5.2% viewers watch for more than a minute, it gets 3.25 views per day on average, and it was flagged 7 times as spam

# Purpose



- Based on this vector our Machine Learning system may predict that there is an 80% probability that it is a spam video, 18% that it is clickbait, and 2% that it is a good video.
- This could be represented as the following vector:

$$\text{class\_probabilities} = \begin{pmatrix} 0.80 \\ 0.18 \\ 0.02 \end{pmatrix}$$

# Vectors in python



- In python, a vector can be represented in many ways

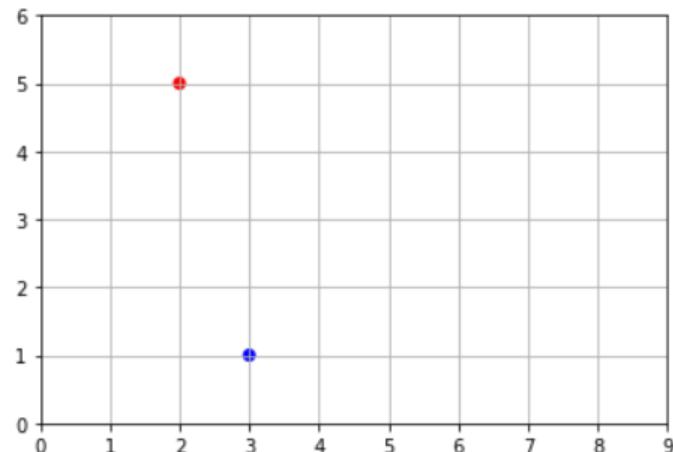
- import numpy as np
- video = np.array([10.5, 5.2, 3.25, 7.0])
- Video

Output : array([10.5 , 5.2 , 3.25, 7. ])

# Plotting vectors



- To plot vectors, we will use matplotlib, so let's start by importing
- import matplotlib.pyplot as plt
- Let's create a couple of very simple 2D vectors to plot:
  - `u = np.array([2, 5])`
  - `v = np.array([3, 1])`
- These vectors each have 2 elements, so they can easily be represented graphically on a 2D graph, for example as points:
  - `x_coords, y_coords = zip(u, v)`
  - `plt.scatter(x_coords, y_coords, color=["r", "b"])`
  - `plt.axis([0, 9, 0, 6])`
  - `plt.grid()`
  - `plt.show()`



# Arrays



- The array object in NumPy is called **ndarray** meaning 'n-dimensional array'.
- To begin with, you will use one of the most common array types: the one-dimensional array ('1-D').
- A 1-D array represents a standard list of values entirely in one dimension.
- **Remember that in NumPy, all of the elements within the array are of the same type.**

# Arrays -Examples



- Create and print a NumPy array 'a' containing the elements 1, 2, 3.
  - `a = np.array([1, 2, 3])`
  - `print(a)`
- Create an array with 3 integers, starting from the default integer 0.
  - `b = np.arange(3)`
  - `print(b)`
- NumPy function `np.linspace` is a floating point (`np.float64`).
  - `l = np.linspace(0, 100, 5)`
  - `print(l)`
- Converting and displaying as int
  - `l_int = np.linspace(0, 100, 5, dtype=int)`
  - `print(l_int)`

Output: [1 2 3]

Output: [0 1 2]

Output: [ 0. 25. 50.  
75. 100.]

Output: [ 0 25 50 75  
100]

# Arrays



- One of the advantages of using NumPy is that you can easily create arrays with built-in functions such as:
  - `np.ones()` - Returns a new array setting values to one.
  - `np.zeros()` - Returns a new array setting values to zero.
  - `np.empty()` - Returns a new uninitialized array.
  - `np.random.rand()` - Returns a new array with values chosen at random.

# Vector Operations



- Vector Addition:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

- Vector Multiplication:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}$$

- Vector Dot Product:

$$[1 \ 2 \ 3] \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = (1 * 1) + (2 * 2) + (3 * 3) = 14$$

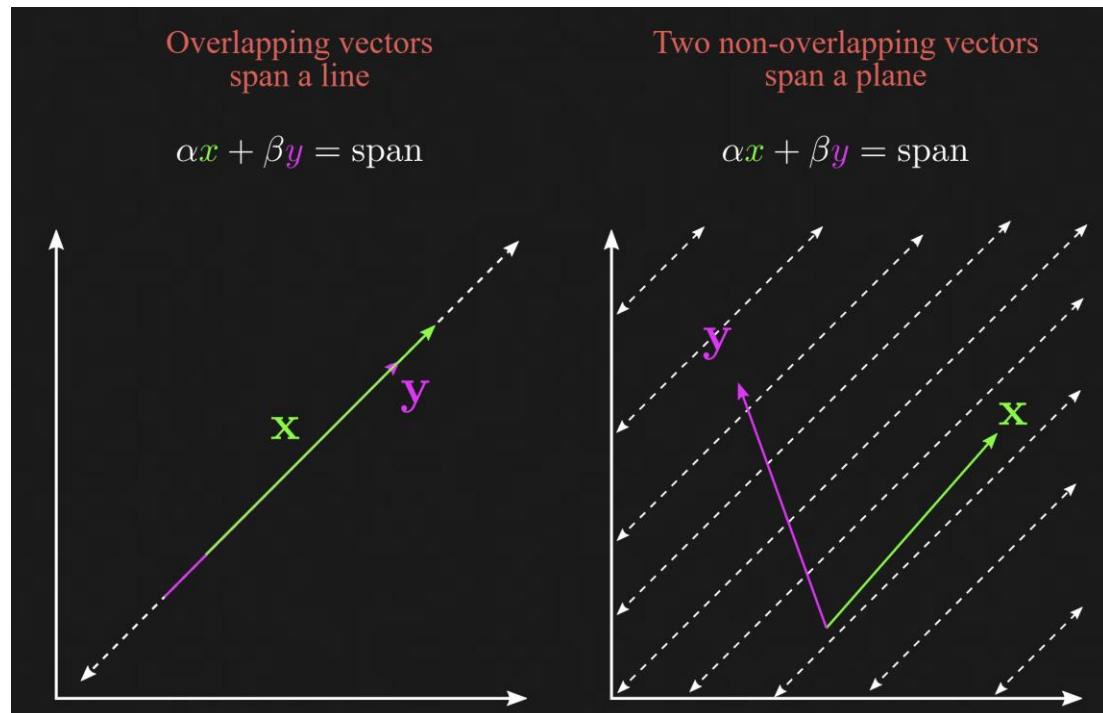
# Vector Space & Span

## ■ Vector Space:

- In its more general form, a vector space, also known as linear space, is a collection of objects that follow the rules defined for vectors in  $R^n$ .

## ■ Vector Span:

- Consider the vectors  $x$  and  $y$  and the scalars  $\alpha$  and  $\beta$ . If we take all possible linear combinations of  $\alpha x + \beta y$  we would obtain the **span** of such vectors.

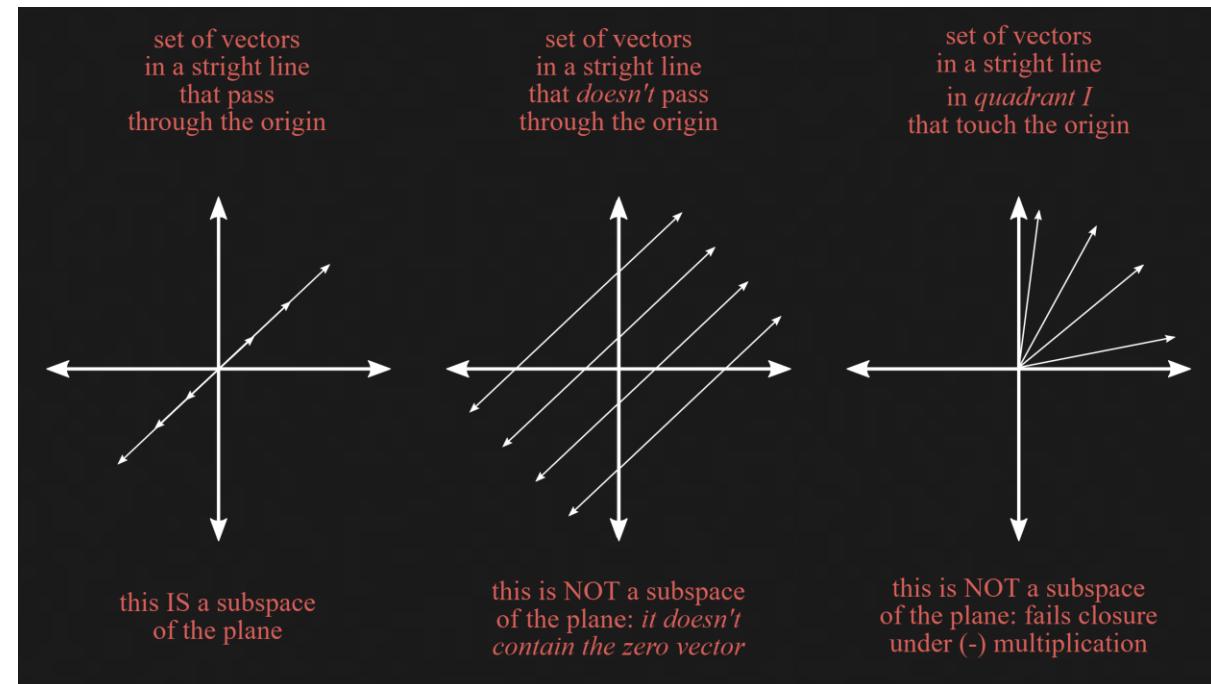


# Vector Subspace



## ■ Vector Subspace:

- A vector subspace is a vector space that lies within a larger vector space. These are also known as linear subspaces.
- Consider a subspace  $S$ . For a vector to be in the valid subspace it has to meet three conditions:
  - Contains the zero vector,  $\mathbf{0} \in S$
  - Closure under multiplication,  $\forall \alpha \in R \rightarrow \alpha \times s_i \in S$ .
  - Closure under addition,  $\forall s_i \in S \rightarrow s_1 + s_2 \in S$ .



*Is the span of  $x = [1 \ 1]$  a valid subspace of  $R^2$ ?*

# Basis of Vector Space



- The basis of a vector space is a set of vectors that satisfies two critical criteria:
  - **Linear Independence:** The vectors in the basis set are linearly independent. This means that no vector in the basis set can be written as a linear combination of the others.
  - **Spanning the Space:** The basis set of vectors spans the vector space. This means that any vector in the vector space can be expressed as a linear combination of the vectors in the basis set.
- The number of vectors in a basis set of a vector space is equal to the number of dimensions.
- What is the basis set of  $R^3$ ?

# Matrices



- With matrices, we can represent sets of variables. In this sense, a matrix is simply an ordered collection of vectors.
- Matrix-Matrix Addition:

$$A + B = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

- Matrix-Matrix Dot Product:

- $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}$

- $A \cdot B = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mp} \end{bmatrix}$

- $c_{ij} = \sum_{l=1}^n a_{il}b_{lj} \text{ where } i = 1, \dots, m, \text{ and } j = 1, \dots, p$

# The Determinant



- The determinant is a scalar value that can be computed from the elements of a square matrix. It is denoted as  $\det(A)$  or  $|A|$  for a matrix  $A$ .
- Consider Matrix: 
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$
- $Determinant = a \cdot d - b \cdot c = (1 * 4) - (3 * 2) = -2$
- Applications:
  - **Geometry:** Used in transformations, rotations, and scaling of geometric figures.
  - **Physics and Engineering:** In solving systems of linear equations which frequently arise in physics and engineering problems.
  - **Computer Graphics:** In transformations and animations.
  - **Economics:** In solving linear models and systems of equations in economic analysis.

# Matrices in python



- In python, a matrix can be represented in various ways. The simplest is just a list of python lists:
  - `[10, 20, 30],`
  - `[40, 50, 60]`

```
[[10, 20, 30], [40, 50, 60]]
```

# Matrices



- Use the NumPy library which provides optimized implementations of many matrix operations:
  - `A = np.array([[10, 20, 30], [40, 50, 60]])`
  - `A`

```
array([[10, 20, 30],  
       [40, 50, 60]])
```

# Size



- The size of a matrix is defined by its number of rows and number of columns.
- For example, consider a  $2 \times 3$  matrix: 2 rows, 3 columns. Caution: a  $3 \times 2$  matrix would have 3 rows and 2 columns.
- To get a matrix's size in NumPy:
  - `A.shape`

(2, 3)

- Caution: the `size` attribute represents the number of elements in the `ndarray`, not the matrix's size:

- `A.size`

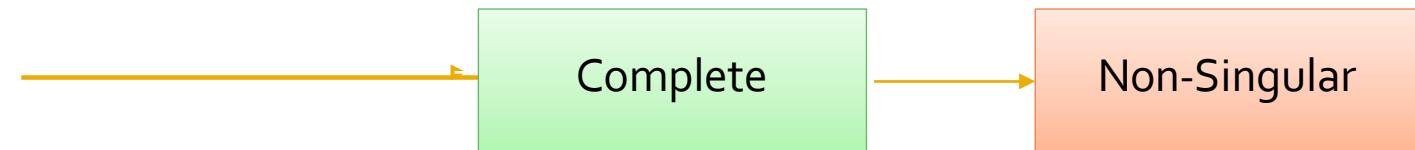
6

# Introduction to System of Linear Equations



## ■ System 1

- The apple is red
- The pear is green



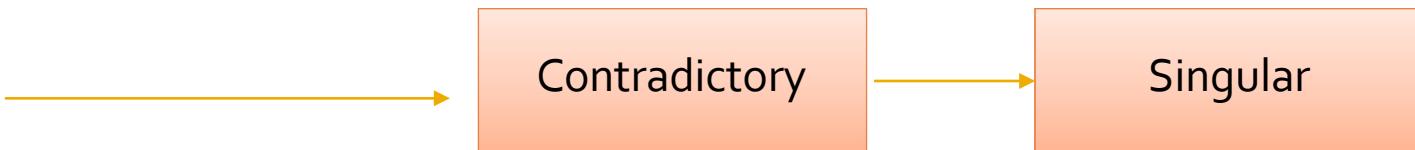
## ■ System 2

- The apple is red
- The apple is red



## ■ System 2

- The apple is red
- The apple is green



# System of Equations



- The system of equations can also be singular and non-singular, very similar to sentences.

# System of Linear Equations



## ■ Linear Equations

- $x + y = 20$
- $x + 2y = 22$

## • Non Linear Equations-complicated

- $x^2 + y^2 = 20$
- $x + 2y = 22$
- $\sin(a) + 2y = 35$
- $xy + \log(z) = 5$

# System of Equations as Lines , planes



- Plotting the different solutions as the coordinates visually on the graph.
- Example: Consider the linear equation  $3x + 4y = 10$  ,  $2x - 2y = 2$  ;
- Since the points cross at a unique solution, it is said to be nonsingular.
- If there are infinite solutions or no solutions then it is said to be singular.

# System of Linear Equations as Matrices



- Coefficients of the Linear Equations are considered elements of the matrix.
- Example :  $x+y=0$       matrix = 
$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$
$$x+2y=0$$
- Example :  $x+y=0$       matrix = 
$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$$
$$2x+2y=0$$
- Matrices can also be Singular and Non Singular

# Singular Vs Nonsingular Matrix



- **Linear dependence** between rows – Second row is multiple of the first row. Here they are linearly dependent

- For Example :  $x+y=0$

$$2x+2y=0$$

|   |   |                                      |
|---|---|--------------------------------------|
| 1 | 1 | First Row                            |
| 2 | 2 | Second Row/multiple of the first row |

- Otherwise, they are referred to be **linearly independent**

- For Example :  $x+y=0$

$$x+2y=0$$

|   |   |  |
|---|---|--|
| 1 | 1 | First Row                                  |
| 1 | 2 | Second Row/not a multiple of the first row |

# Singular vs Non-Singular Matrix



- To determine whether a system of linear equations is singular vs non-singular, one could use ***The Determinant***:
  - If the determinant of the matrix of the system is equal to zero → **Singular System**
  - If the determinant of the matrix of the system is not equal to zero → **Non-Singular System**

# Solving Systems of Linear Equations using Matrices



- NumPy linear algebra package provides a quick and reliable way to solve the system of linear equations using the function **np.linalg.solve(A, b)**.
  - $A$  is a matrix, each row of which represents one equation in the system and each column corresponds to the variable  $x_1, x_2$
  - $b$  is a 1-D array of the free (right side) coefficients.
- [More about the Package: <https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html>](https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html)

# References



- [https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what\\_is\\_machine\\_learning](https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what_is_machine_learning)

# Thank you!



- Any questions?





Thank You  
Youssef Abdelkareem  
[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000

# Linear Algebra

---

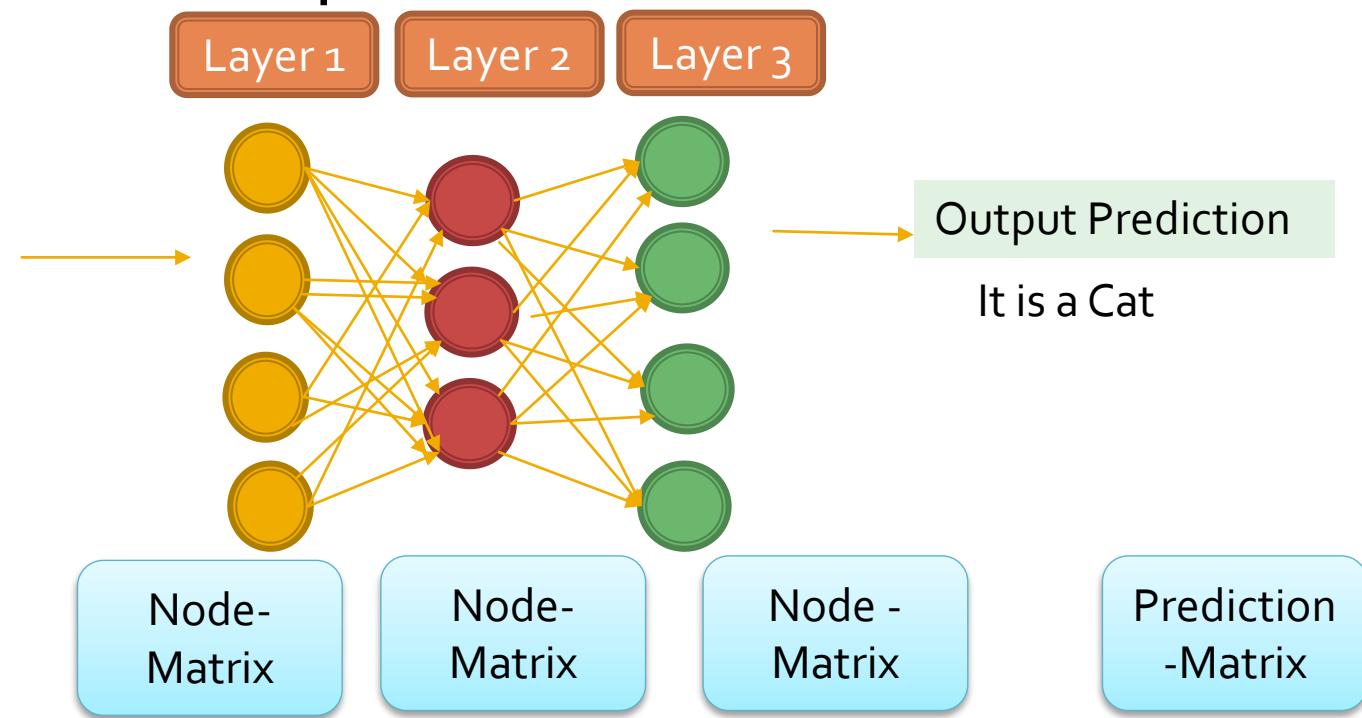
- Vectors
- Matrix
  - Inverse
  - Rank
- Linear Transformation
- Eigenvalues
- Eigenvectors
- These are used in machine learning to store and compute on data.



# Motivation to Machine Learning



- Linear Algebra most useful in machine learning
- Most popular application –Neural Networks –image recognition –using matrix operations.



# Recap: Singular vs Non-Singular



- Linear dependence between rows – Second row is multiple of the first row. Here they are linearly dependent

- For Example :  $x+y = 0$

$$2x+2y=0$$

|   |   |                                      |
|---|---|--------------------------------------|
| 1 | 1 | First Row                            |
| 2 | 2 | Second Row/multiple of the first row |

- Otherwise, they are referred to be linearly independent

- For Example :  $x+y = 0$

$$x+2y=0$$

|   |   |  |
|---|---|--|
| 1 | 1 | First Row                                  |
| 1 | 2 | Second Row/not a multiple of the first row |

# Rank of Matrix



- The rank of a matrix is the maximum number of linearly independent rows (or columns) in the matrix.
- A matrix is considered to be of **full rank** if none of its columns (or rows) can be expressed as a linear combination of the others. This implies linear independence. **[Non-Singular]**
- A matrix is **non-full rank** if its columns (or rows) are linearly dependent. In this case, at least one column (or row) can be expressed as a linear combination of the others. **[Singular]**

# Solving System of Linear Equations



- To solve the linear equations, we could get to a solved system
  - Multiplying by a constant [Scaling]
  - Adding two equations [Elimination]
  - Subtracting the equations [Elimination]
  - Substitution with values
  - For example :
    - Eq 1:  $4x + y = 16$
    - Eq 2:  $x - \frac{1}{2}y = 4$
    - Multiply Eq 2 by 2  $\rightarrow$  Eq 2:  $2x - y = 8$
    - Add Eq 1 & Eq2  $\rightarrow 6x = 24 \rightarrow x = 4$
    - Substitute  $x$  in Eq1  $\rightarrow (4 * 4) + y = 16 \rightarrow y = 0$

# Matrix Row Reduction –Gaussian Elimination



- Apply the operations to convert the matrix into a much simpler representation and solve the system.

System  
(Actual)

- $5x+y=17$
- $4x-3y=6$

|   |    |    |
|---|----|----|
| 5 | 1  | 17 |
| 4 | -3 | 6  |

Scaling  
 $R1 = \frac{1}{5}R1$

- $1x+\frac{1}{5}y = \frac{17}{5}$
- $4x-3y=6$

|   |     |      |
|---|-----|------|
| 1 | 1/5 | 17/5 |
| 4 | -3  | 6    |

Addition  
 $R2 = R2 - 4R1$

- $1x+\frac{1}{5}y = \frac{17}{5}$
- $0x-3.8y=-7.6$

|   |      |      |
|---|------|------|
| 1 | 1/5  | 17/5 |
| 0 | -3.8 | -7.6 |

Scaling  
 $R2 = \frac{1}{-3.8}R2$

- $1x+\frac{1}{5}y = \frac{17}{5}$
- $0x+1y=2$

|   |     |      |
|---|-----|------|
| 1 | 1/5 | 17/5 |
| 0 | 1   | 2    |

Substitution  
 $R2 \rightarrow R1$

- $1x+\frac{1}{5}y = \frac{17}{5}$
- $0x+1y=2$

|   |     |      |
|---|-----|------|
| 1 | 1/5 | 17/5 |
| 0 | 1   | 2    |

Row Echelon Form

# Example



## Actual System

- $5x+y=11$
- $10x+2y=22$

## Matrix form

|    |   |
|----|---|
| 5  | 1 |
| 10 | 2 |

## Manipulated system

- $$\begin{aligned}x+0.2y &= 2.2 \\ 0x+0y &= 0\end{aligned}$$

## Upper Diagonal Matrix

|   |     |
|---|-----|
| 1 | 0.2 |
| 0 | 0   |

Row Echelon  
Matrix

# Row Echelon Form



- The Row Echelon Form (REF), also known as row-reduced form, is a specific arrangement of a matrix that simplifies solving systems of linear equations. A matrix is in row echelon form if it satisfies the following conditions:
  - Zero Rows: Any row containing only zeros is at the bottom.
  - Leading Entry: In each nonzero row, the leftmost nonzero entry is **1**, and the element directly below this leading **1** is zero.
  - Leading **1s**: The leading **1** in the second row (if it exists) is to the right of the leading **1** in the first row. The leading **1** in the third row (if it exists) is to the right of the leading **1** in the second row, and so on.

|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 5 | 2 |
| 0 | 1 | 3 | 3 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |

A

|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 5 | 2 |
| 0 | 1 | 3 | 3 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |

B

# Rank of Matrix



- Number of pivot elements –Rank of matrix
- The rank of the matrix is the number of non-zero rows in the row echelon form. To find the rank, we need to perform the following steps:
  - Find the row-echelon form of the given matrix
  - Count the number of non-zero rows.



# Trace of a Matrix

- The trace of a matrix, typically denoted as  $tr(A)$  for a matrix  $A$ , is the sum of the elements on the main diagonal of the matrix.
- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- $tr(A) = 1 + 4 = 5$
- Applications:
  - Often used in optimization problems
  - Understanding the properties of certain matrices like covariance matrices



# Which Matrices have inverse?

- An invertible matrix is one that is:
  - *Full Rank*
  - *Has Determinant not equal to zero*
  - *All rows and columns are linearly independent*
  - *Non-Singular*
  - *Square Matrix*
- For example, for the following  $2 \times 2$  matrix:
  - $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
  - $A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$
  - $A A^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$

# Recall: Vector



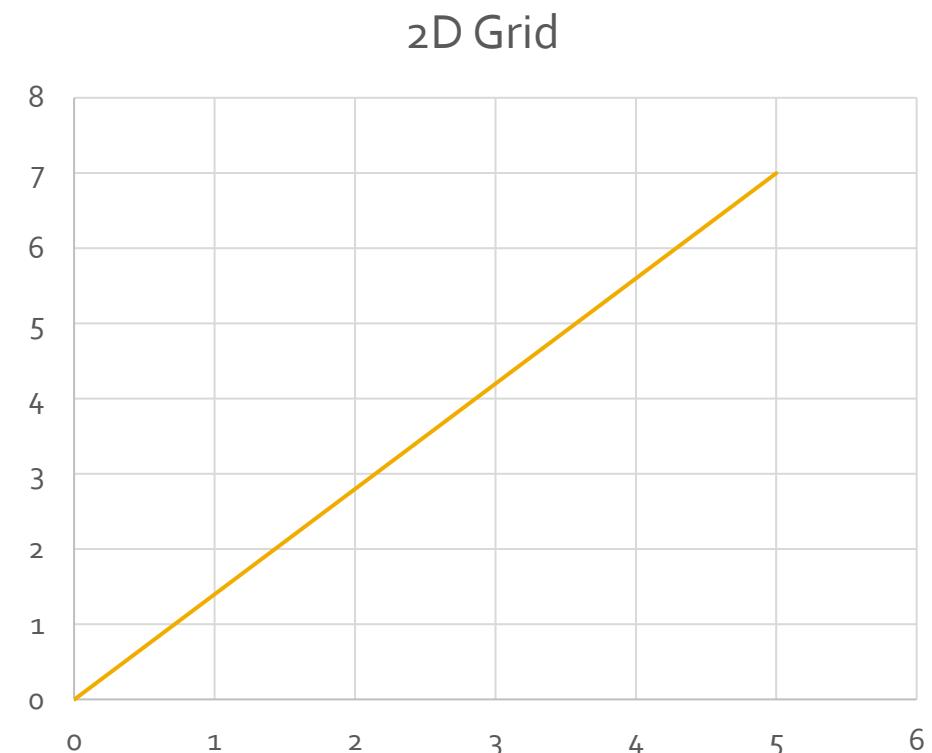
- A vector is a quantity defined by a magnitude and a direction. For example, a rocket's velocity is a 3-dimensional vector: its magnitude is the rocket's speed, and its direction is (hopefully) up.
- A vector can be represented by an array of numbers called *scalars*.
- Each scalar corresponds to the magnitude of the vector with regard to each dimension.
- For example, say the rocket is going up at a slight angle: it has a vertical speed of 5,000 m/s, and also a slight speed towards the East at 10 m/s, and a slight speed towards the North at 50 m/s.

$$\text{velocity} = \begin{pmatrix} 10 \\ 50 \\ 5000 \end{pmatrix}$$

# Norms



- **L<sub>1</sub> –norm:**
  - Formulation:  $|[a, b]| = |a| + |b|$
  - Geometric Interpretation: the distance you would travel if you could only move along the grid lines.
- **L<sub>2</sub> –norm:**
  - Formulation:  $\|[a, b]\|_2 = \sqrt{a^2 + b^2}$
  - Geometric Interpretation: measures the straight-line distance, which is the Euclidean distance between two points.





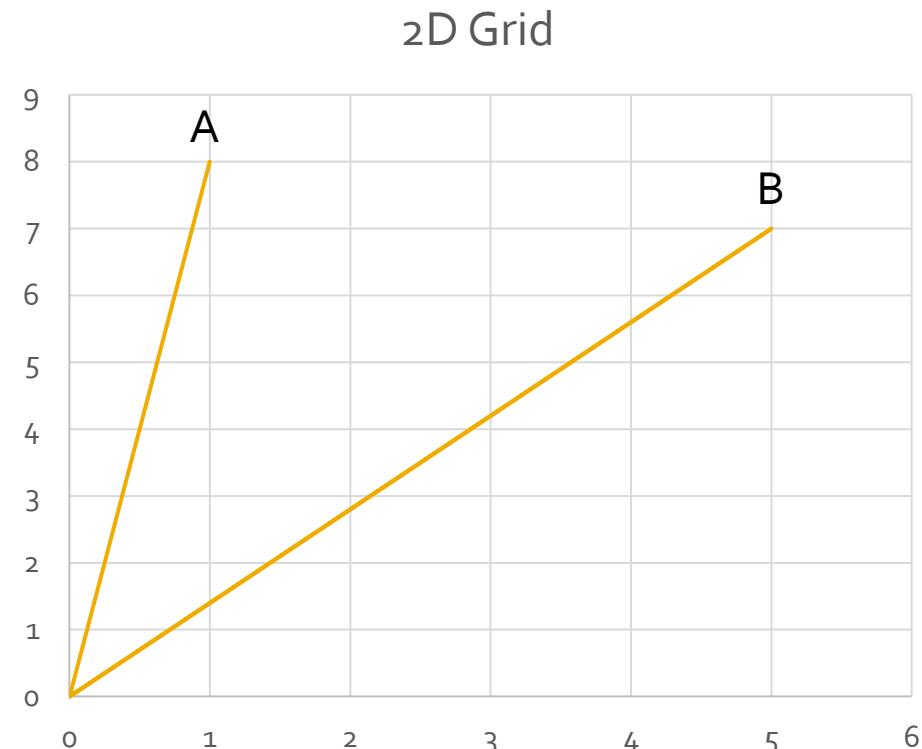
# Sum and Difference of Vectors

- Sum and Difference of Vectors :  $z = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$ 
  - Add the coordinates of vectors ,  $z + y = \begin{bmatrix} 2+1 \\ 1+4 \end{bmatrix}$
  - Subtract the coordinates of vectors ,  $z - y = \begin{bmatrix} 2-1 \\ 1-4 \end{bmatrix}$

# Distance between the Vectors



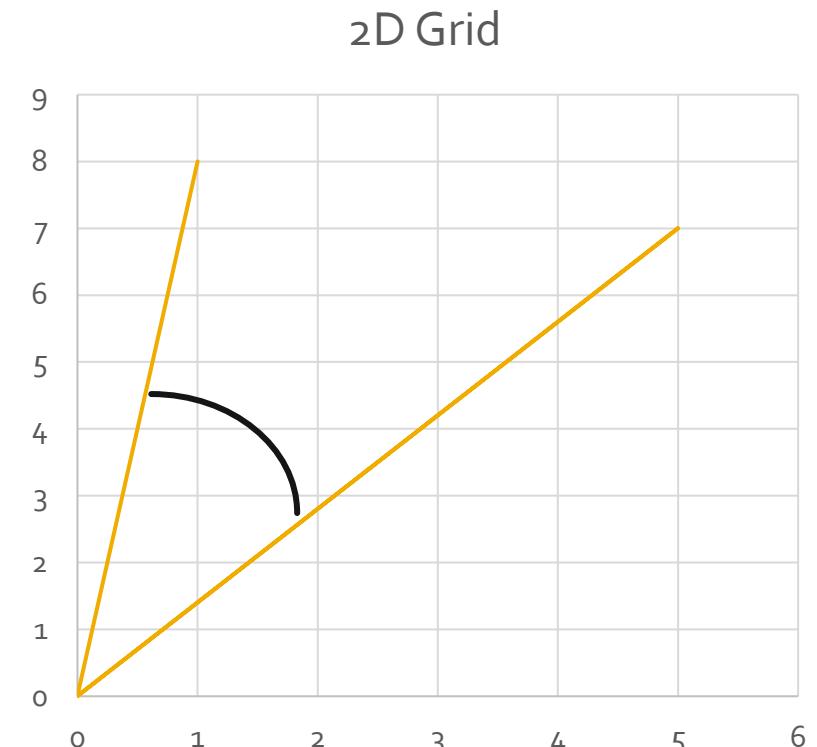
- How different are two vectors  $A = \begin{bmatrix} 1 \\ 8 \end{bmatrix}$  and  $B = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$  from each other?
  - L<sub>1</sub>-distance  $|A - B|$  :
    - $|A - B| = |1 - 5| + |8 - 7| = 5$
  - L<sub>2</sub>-distance  $||A - B||$  :
    - $||A - B|| = \sqrt{(1 - 5)^2 + (8 - 7)^2} = 4.12$



# Geometric Definition of the Dot Product



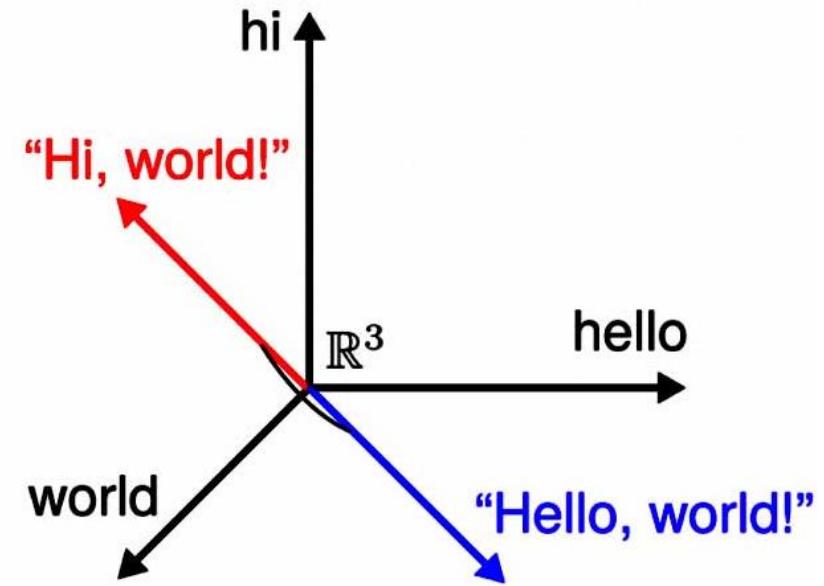
- In Euclidean space, a Euclidean vector has both magnitude and direction.
- The dot product of two vectors  $x$  and  $y$  is defined by:
  - $x \cdot y = |x||y|\cos(\theta)$ , where  $\theta$  is the angle between the two vectors.
  - $x \cdot y = x_1y_1 + x_2y_2 + x_3y_3 + \dots$
- What is the value of  $A \cdot B$ ?



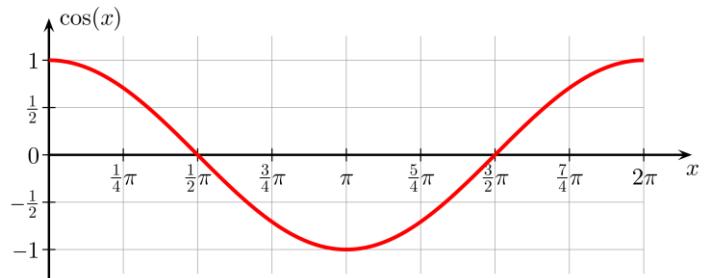
# Cosine Similarity



- Evaluates similarity in direction, regardless of magnitude.
- Commonly used in text analysis, recommendation systems, and information retrieval.
- Formulation:
  - $\cos(\theta) = \frac{x \cdot y}{|x||y|}$
- Dimensions =  $\begin{bmatrix} \text{hi} \\ \text{hello} \\ \text{world} \end{bmatrix}$
- $A = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},$
- $\text{Cos-Sim}(A, B) = \frac{1}{2}$



Cosine Similarity



# How to find a norm of a vector?



- The norm of a vector can be found using NumPy function `np.linalg.norm()`:
- `print("Norm of a vector v is", np.linalg.norm(v))`
  - Norm refers to the L<sub>2</sub> norm =  $\sqrt{a^2 + b^2}$

# Spam Classifier



| Spam | Prize | Award |
|------|-------|-------|
| Yes  | 1     | 2     |
| Yes  | 2     | 2     |
| Yes  | 0     | 2     |
| No   | 0     | 1     |
| Yes  | 2     | 2     |
| No   | 1     | 0     |

What threshold do we need to consider for the classifier to be considered as a model that does spam classifier?

# Matrix Multiplication



| Spam | Prize | Award |
|------|-------|-------|
| Yes  | 1     | 2     |
| Yes  | 2     | 2     |
| Yes  | 0     | 2     |
| No   | 0     | 1     |
| Yes  | 2     | 2     |
| No   | 1     | 0     |

**Best Threshold?**  
 $Pred = y > 1.5$

$a$

$w$

Model

$y = a \cdot w$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow y = a \cdot w = 3$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow y = a \cdot w = 1$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow y = a \cdot w = 1$$

# Linear Transformations

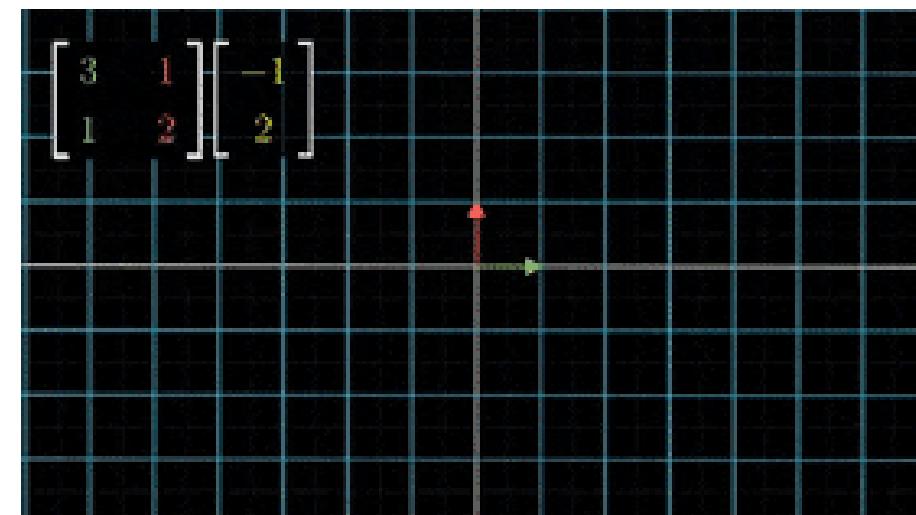


- A transformation is a function from one vector space to another that respects the underlying (linear) structure of each vector space.
- Linear Transformations
  - A transformation  $T$  is said to be linear if the following two properties are true for any scalar  $k$  and any input vectors  $u$  and  $v$ :
    - $T(kv) = k \cdot T(v)$
    - $T(u + v) = T(u) + T(v)$

# Linear Transformations as Matrices



- $A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$  *Vector* =  $\begin{bmatrix} -1 \\ 2 \end{bmatrix}$  ->in
- Matrix Multiplication,  $A @ \text{Vector} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$  ->Out



# Eigen Vectors and Eigen Values



- Given a vector  $v$ , apply linear transformation with square matrix  $A$ .

$$A \cdot v = ?$$

$$A \cdot v = \lambda v \rightarrow [\text{Special Case}]$$

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\lambda_1 = 3, \quad v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = -1, \quad v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

# Eigen Values and Eigen Vectors



- An eigenvalue of a matrix represents the scaling factor that the matrix applies to a vector in its eigenvector direction.
- If  $v$  is an eigenvector of a matrix  $A$  with eigenvalue  $\lambda$ , then  $Av = \lambda v$ .
- Matrix  $A$  stretches or compresses the vector  $v$  by a factor of  $\lambda$  in its eigenvector direction.
- Eigenvalues and eigenvectors can be found using the NumPy function `np.linalg.eig()`.
  - It returns a tuple consisting of a vector and an array.
  - The vector contains the eigenvalues.
  - The array contains the corresponding eigenvectors, one eigenvector per column

# Quick Quiz 1



- Dimensions =  $\begin{bmatrix} hi \\ hello \\ world \end{bmatrix}$
- What is the Cosine Similarity between the following pairs of sentences:
  - “hi hi” and “hello world”
  - “hello world” and “hello world”

# References



- [https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what\\_is\\_machine\\_learning](https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what_is_machine_learning)

# Thank you!

- Any questions?





**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000

# Probability

---

- Probability
- Conditional Probability
- Bayes Theorem
- Naïve Bayes Model
- Spam Filter using Naïve's Bayes





# What is Probability?

- Probability is how likely an event (E) is to occur.

$$P(E) = \frac{\text{Number of needed outcomes of event } E}{\text{Total number of outcomes in sample space } S}$$

- Example 1: Action: Flip Coin, Sample Space (S) = {Heads, Tails}
  - Sample Space Size = 2
  - What is the probability for the event (E) Head to occur?
    - $P(H) = \frac{1}{2}$
- Example 2: Action: Flip 2 Coins, Sample Space (S) = {HH, HT, TH, TT}
  - What is the probability for the event (E) = HH to occur?
    - $P(HH) = \frac{1}{4}$
- Example 3: Action: Roll Dice, Sample Space (S) = {1, 2, 3, 4, 5, 6}
  - What is the probability for getting 1?
    - $P(1) = \frac{1}{6}$
  - What is the probability for getting an even number?
    - $P(Even) = \frac{3}{6} = \frac{1}{2}$



# Complement of a Probability

- $P(A') = 1 - P(A)$

$$P(Tails) = 1 - P(Heads) = 1 - \frac{1}{2} = \frac{1}{2}$$

$$P(Odd\ Number\ in\ Dice) = ?$$

# Joint Events



- Joint Events are ones that could share some outcomes
- Example: Suppose, in a college, students can play different sports. Assume the total number of students is 10. Number of students who play soccer (S) is 3, Number of students who play basketball (B) is 4, student who play both S and B is 2.
  - $P(S) = 0.3, P(B) = 0.4$
  - $P(\text{Soccer and Basketball}) = P(S \cap B) = ?$ 
    - $P(S \cap B) = \frac{2}{10}$
  - $P(\text{Soccer or Basketball}) = P(S \cup B) = P(S) + P(B) - P(S \cap B)$ 
    - $P(S \cup B) = P(S) + P(B) - P(S \cap B) = \frac{1}{2}$

# Disjoint Events



- Disjoint Events are ones that could **not** share any outcome.
- Example: Suppose, in a college, students can play only one sport. Assume the total number of students is 10. Number of students who play soccer (S) is 3, Number of students who play basketball (B) is 4.
  - $P(S) = 0.3, P(B) = 0.4$
  - $P(\text{Soccer and Basketball}) = P(S \cap B) = ?$ 
    - $P(S \cap B) = 0$
  - $P(\text{Soccer or Basketball}) = P(S \cup B) = ?$ 
    - $P(S \cup B) = P(S) + P(B) - P(S \cap B) = 0.7$

# Independent Events

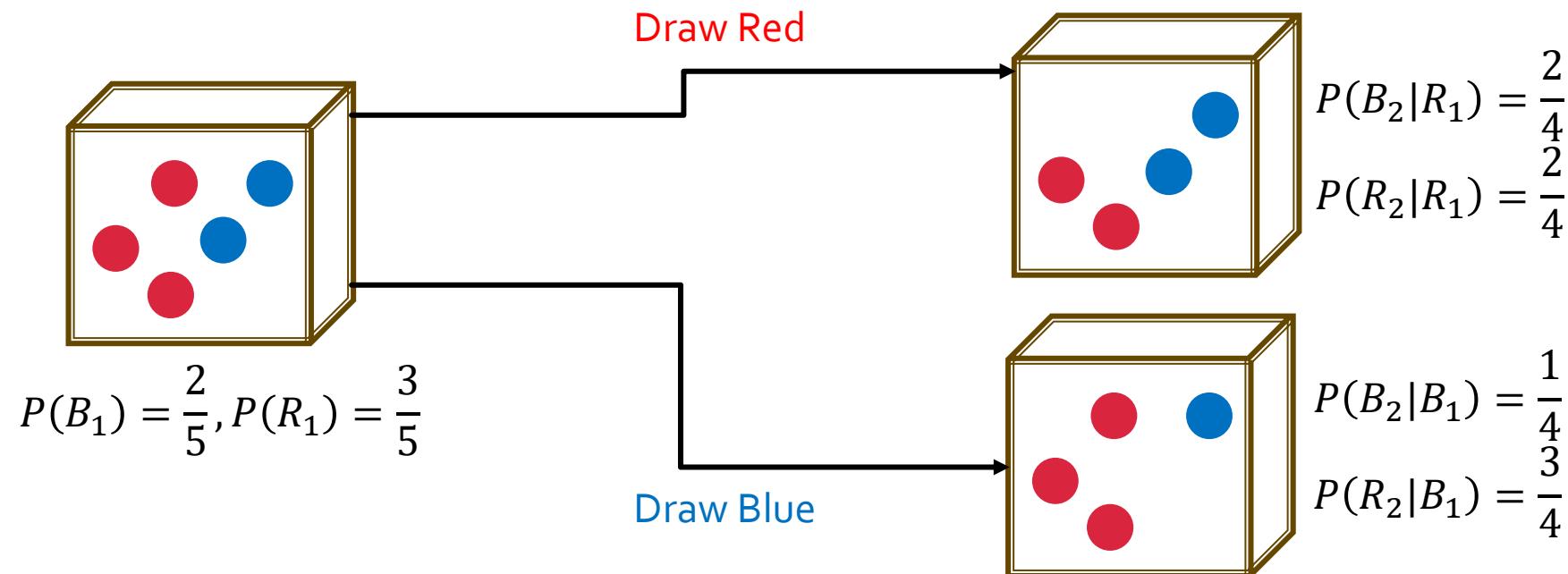


- Occurrence of one event does not affect the occurrence of another following event.
- Example: Flipping a coin twice
  - Event A: Getting Heads on First Coin Flip
  - Event B: Getting Heads on Second Coin Flip
  - Are A and B dependent?
  - What is the probability of A **and** B?
    - $P(A \cap B) = P(A) * P(B)$

# Conditional Probability



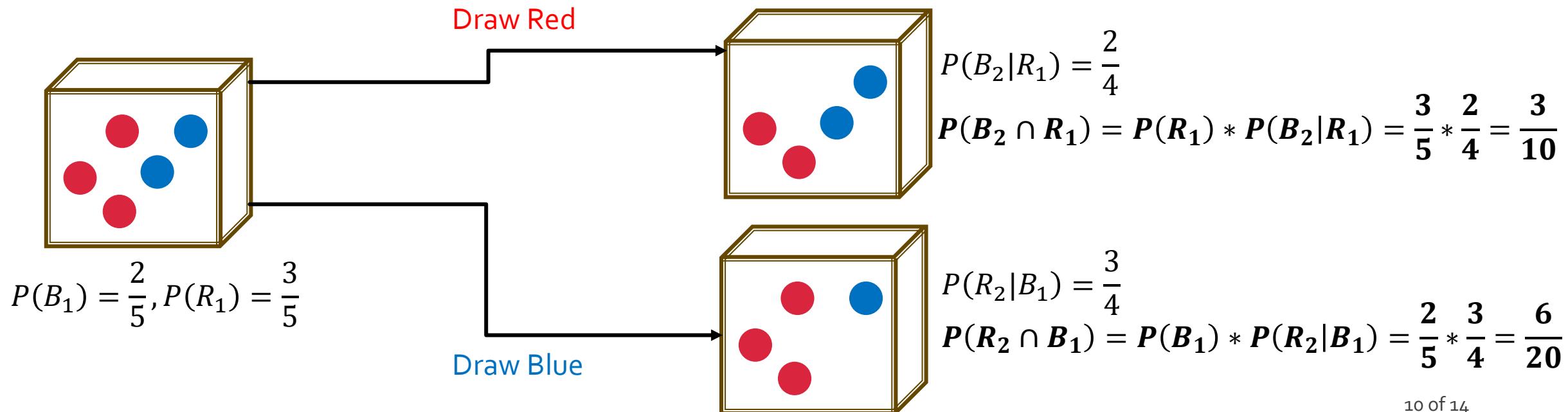
- Probability of event happening given that another event already occurred.
- Example: Suppose we have a box containing 3 red marbles (R) and 2 blue marbles (B). We want to find the probability of drawing a marble on the second draw, given that we drew a marble on the first draw.



# Dependent Events



- Probability of event happening given that another event already occurred.
- Example: Suppose we have a box containing 3 red marbles (R) and 2 blue marbles (B). We want to find the probability of drawing a marble on the second draw, given that we drew a marble on the first draw.



# Recap of Probability Rules



- Complement :  $P(A') = 1 - P(A)$
- Union (OR):  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- Intersection (AND):
  - If A and B are independent:
    - $P(A \cap B) = P(A) * P(B)$
    - $P(A \cap B|C) = P(A|C) * P(B|C)$
  - If B depends on A:  $P(A \cap B) = P(A) * P(B|A)$
- If A depends on a set of events  $\{B_1, B_2, B_3, \dots\}$ :
  - $P(A) = P(B_1) * P(A|B_1) + P(B_2) * P(A|B_2) + P(B_3) * P(A|B_3) + \dots$

# Bayes Theorem



$$P(A|B) = \frac{\underbrace{P(B|A) * P(A)}_{Posterior}}{\underbrace{P(B)}_{Prior}}$$

The equation for Bayes Theorem is shown. The numerator is the product of the likelihood  $P(B|A)$  and the prior  $P(A)$ , grouped by a brace labeled "Posterior". The denominator is the probability  $P(B)$ , which is grouped by a brace labeled "Prior".

# Bayes Theorem – Spam Filter



- Suppose we have a dataset for emails being Spam or Not based on the occurrence of the word “Award” in the email.

- Target: What is the probability of Spam given the award feature  $P(S|A)$ ?

$$P(S|A) = \frac{P(A|S)*P(S)}{P(A)}$$

$$P(S) = \frac{4}{6}$$

$$P(A|S) = \frac{2}{4}$$

$$P(A) = P(S) * P(A|S) + P(S') * P(A|S')$$
$$= \frac{4}{6} * \frac{2}{4} + \frac{1}{3} * \frac{1}{2} = \frac{1}{2}$$

$$P(S|A) = \frac{\frac{2}{4} * \frac{4}{6}}{\frac{1}{2}} = \frac{2}{3}$$

| Spam (S) | AWARD (A) |
|----------|-----------|
| Yes      | No        |
| Yes      | Yes       |
| No       | No        |
| Yes      | Yes       |
| No       | Yes       |
| Yes      | No        |

# Bayes Theorem – Spam Filter



- Suppose we have a dataset for emails being Spam or Not based on the occurrence of the words “Award” and “Gift” in the email.
- Target: What is the probability of Spam given the award and prize features  $P(S|A, G)$ ?

$$P(S|A \cap G) = \frac{P(A \cap G|S) * P(S)}{P(A \cap G)}$$

$$P(S|A \cap G) = \frac{[P(A|S) * P(G|S)] * P(S)}{P(A) * P(G)} \quad \left. \right\} \text{Assume Independence!}$$

$$P(S|A \cap G) = \frac{\left[ \frac{1}{2} * \frac{1}{4} \right] * \frac{4}{6}}{\frac{1}{2} * \frac{2}{6}} = \frac{1}{2}$$

| Spam (S) | AWARD (A) | GIFT (G) |
|----------|-----------|----------|
| Yes      | No        | Yes      |
| Yes      | Yes       | No       |
| No       | No        | No       |
| Yes      | Yes       | No       |
| No       | Yes       | Yes      |
| Yes      | No        | No       |

# Naïve Bayes Theorem



- Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the “naive” assumption of ***conditional independence*** between every pair of features given the value of the class variable.
- General Naive Bayes Formulation:

$$P(y|x_1, x_2, x_3, \dots) = \frac{P(x_1, x_2, x_3, \dots|y)*P(y)}{P(x_1, x_2, x_3, \dots)} = \frac{[P(x_1|y)*P(x_2|y)*\dots]*P(y)}{P(x_1)*P(x_2)*\dots}$$

- What if it's a multi-class classification problem?
  - Compare  $P(y_1|x_1, x_2, x_3, \dots)$ ,  $P(y_2|x_1, x_2, x_3, \dots)$ ,  $P(y_3|x_1, x_2, x_3, \dots)$ , ...

# Random Variables



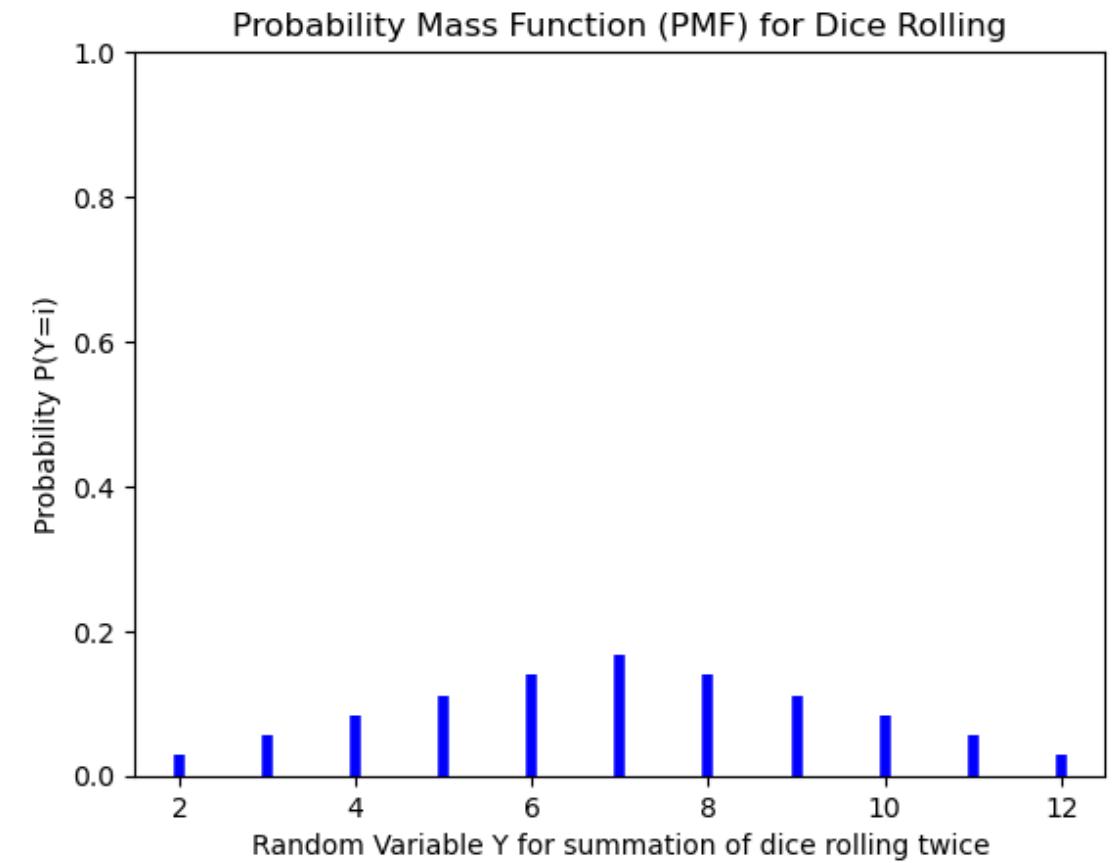
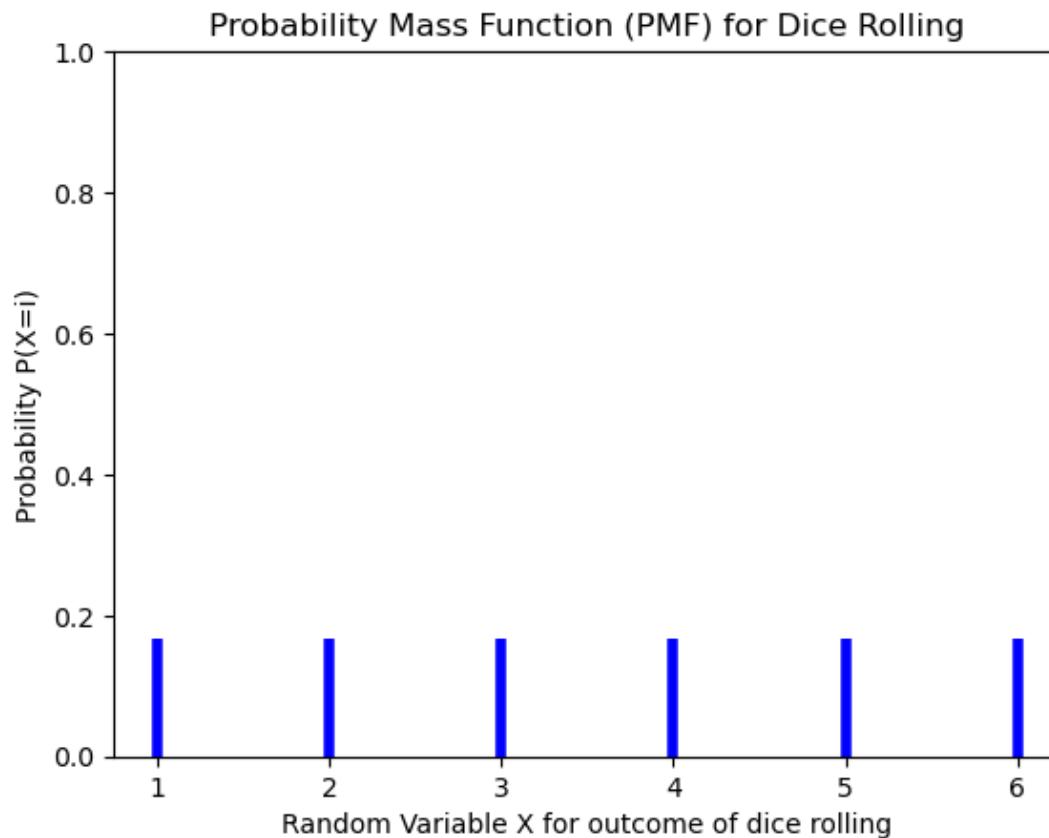
- Random variables are a way to assign numerical values to outcomes of a random process (experiment), making it easier to perform calculations and analyze probabilities.
- Recall the Dice rolling experiment with sample space  $\{1,2,3,4,5,6\}$ .
  - Case 1: Define a random variable  $X$  that represents the value of rolling the dice **once**.
    - $P(X = 1) = P(X = 2) = P(X = 3) = \dots = \frac{1}{6}$
  - Case 2: Define a random variable  $Y$  that represents the summation of dice values after **two rolls**.
    - $P(Y = 2) = P(X_1 = 1 \text{ and } X_2 = 1) = \frac{1}{6} * \frac{1}{6} = \frac{1}{36}$
    - $P(Y = 3) = P(X_1 = 2 \text{ and } X_2 = 1) + P(X_1 = 1 \text{ and } X_2 = 2) = \frac{1}{18}$
    - $P(Y = 1) = ?$

# Discrete and Continuous Random Variables



- Discrete random variables:
  - Can take on only a countable number of distinct values.
  - Example: random variable  $X$  representing number of heads after flipping coin *three* times.
- Continuous random variables:
  - Can take on any value within a given range (infinite values).
  - Example: random variable  $X$  representing the height of a randomly selected person.

# Plotting Discrete Random Variables



*Probability Mass Function*

# Probability Mass Function

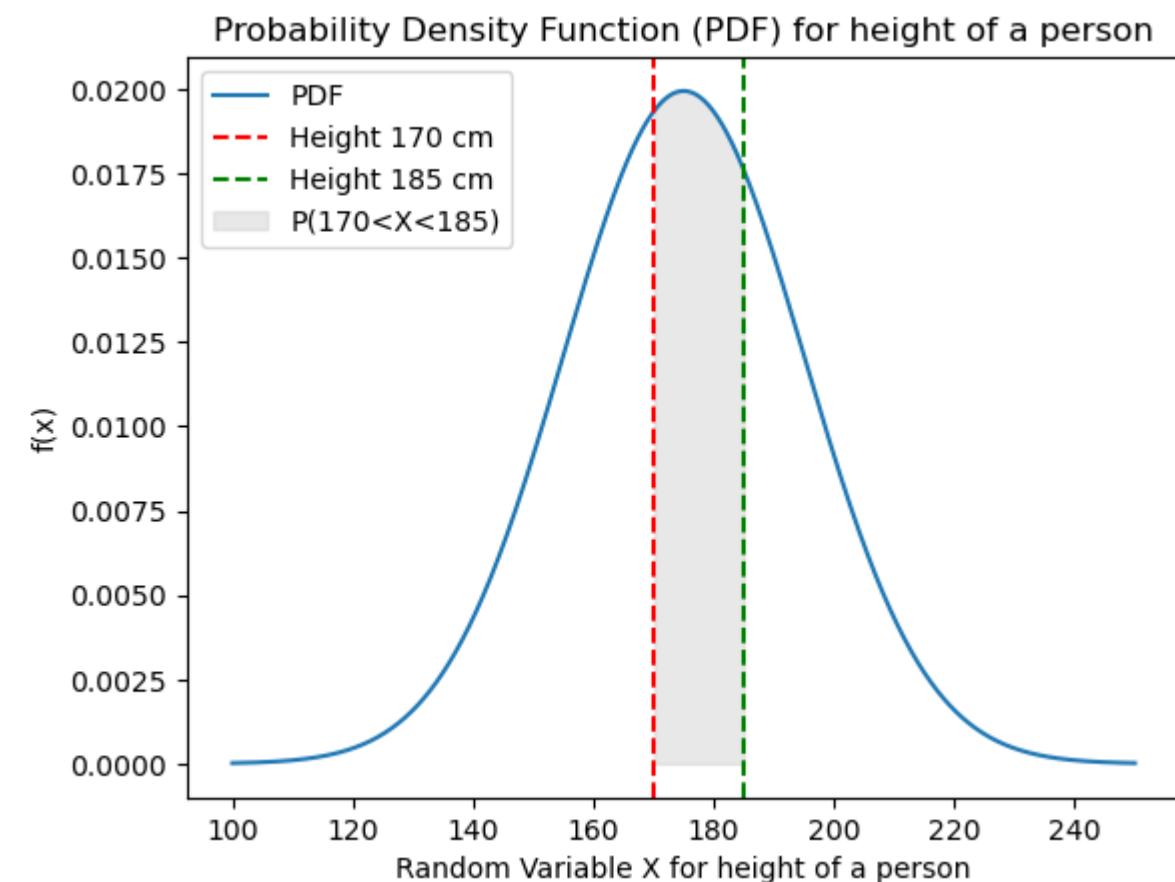
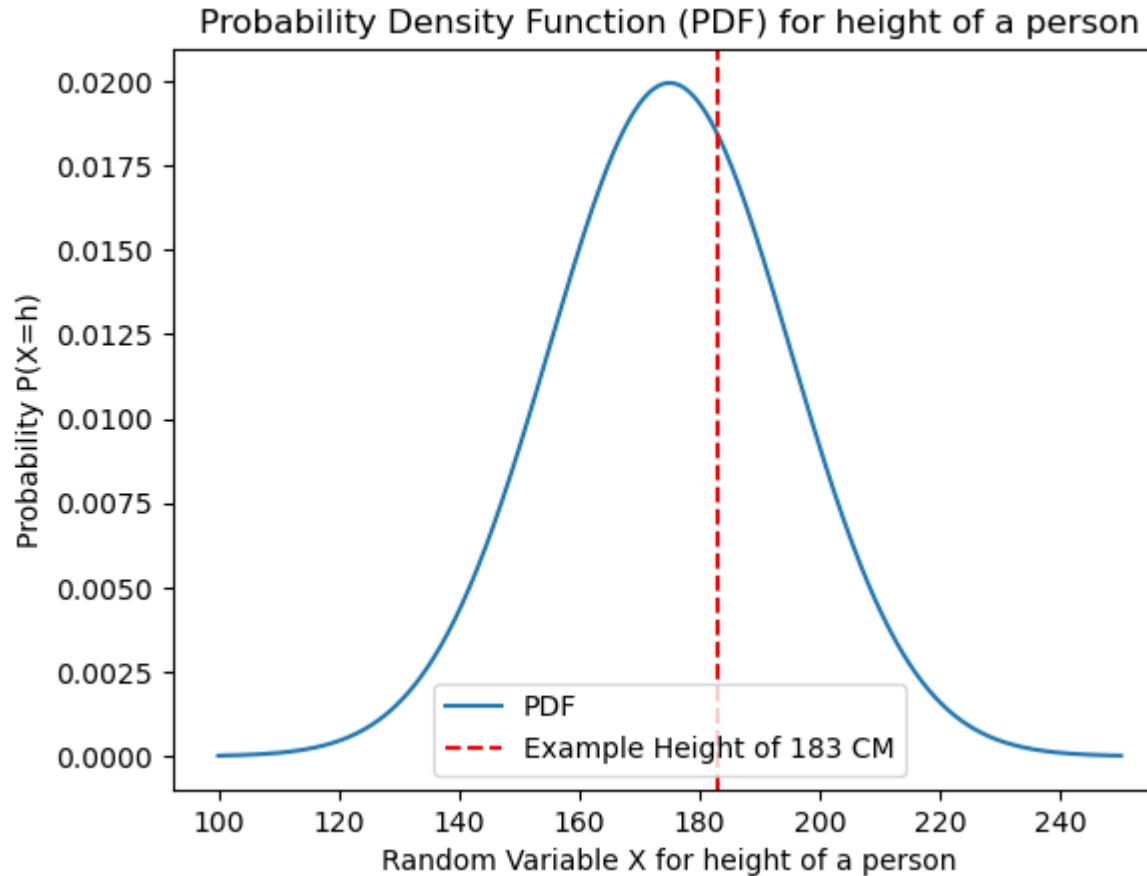


- PMF gives you the probabilities for the discrete random variables, also called discrete density function.
- Ex: Dice Possible Outcomes  $\{1, 2, 3, 4, 5, 6\}$  -> Sample Space
- For each of them, look at the probability they happen. This forms a *probability distribution*
- All discrete random variables can be modeled by their probability mass function, also called PMF.
- Probability Mass Function :

$$p_X(x) \geq 0$$

$$\sum_x p_X(x) = 1$$

# Plotting Continuous Random Variables

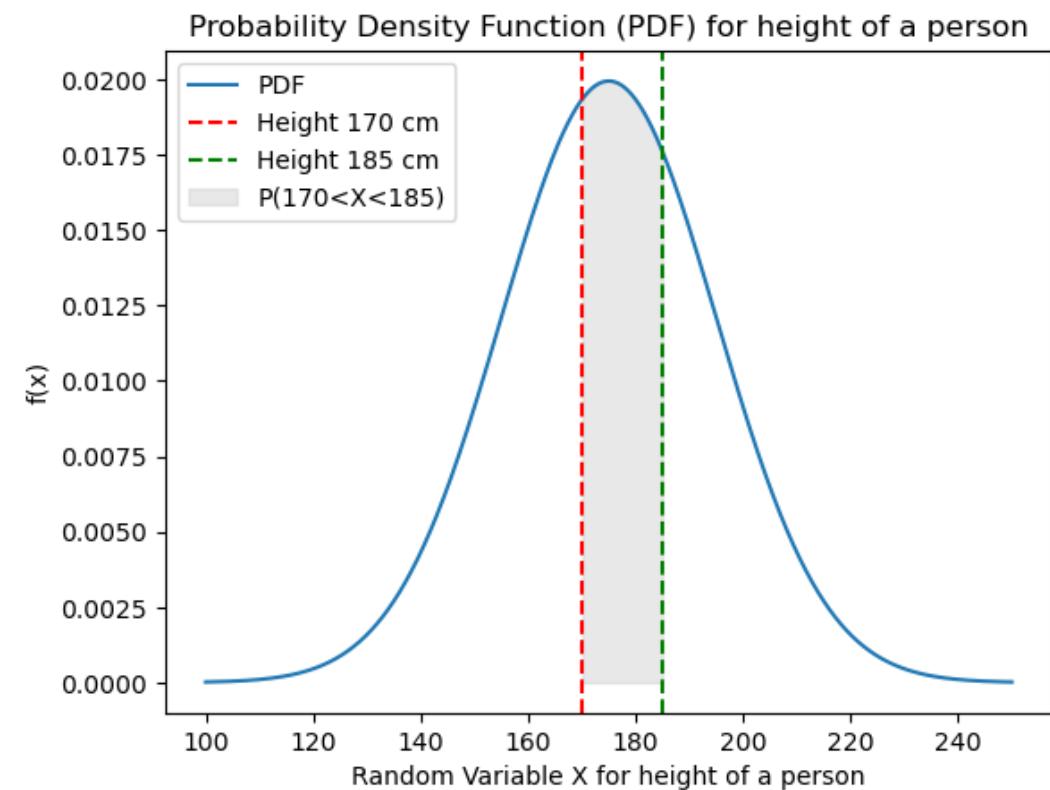


*Probability Density Function*

# Probability Density Function



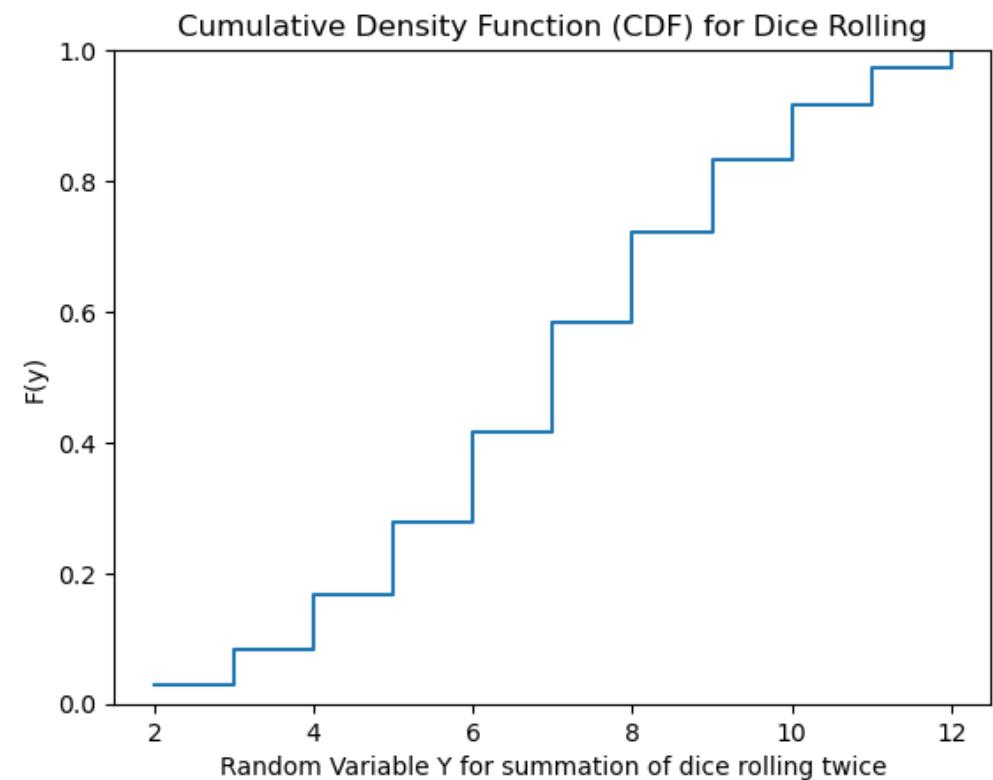
- For a continuous random variable  $X$ , the Probability Density Function, denoted as  $f(x)$ , represents the relative likelihood of the variable taking on a particular value  $x$ .
- Needs to satisfy two properties:
  - Non-Negativity:
    - $f(x) \geq 0$ , for all  $x$  in range of  $X$
  - Area Under Curve (AUC):
    - $P(a < X < b) = \int_a^b f(x) dx$
    - $P(-\infty < X < \infty) = \int_{-\infty}^{\infty} f(x) dx = 1$



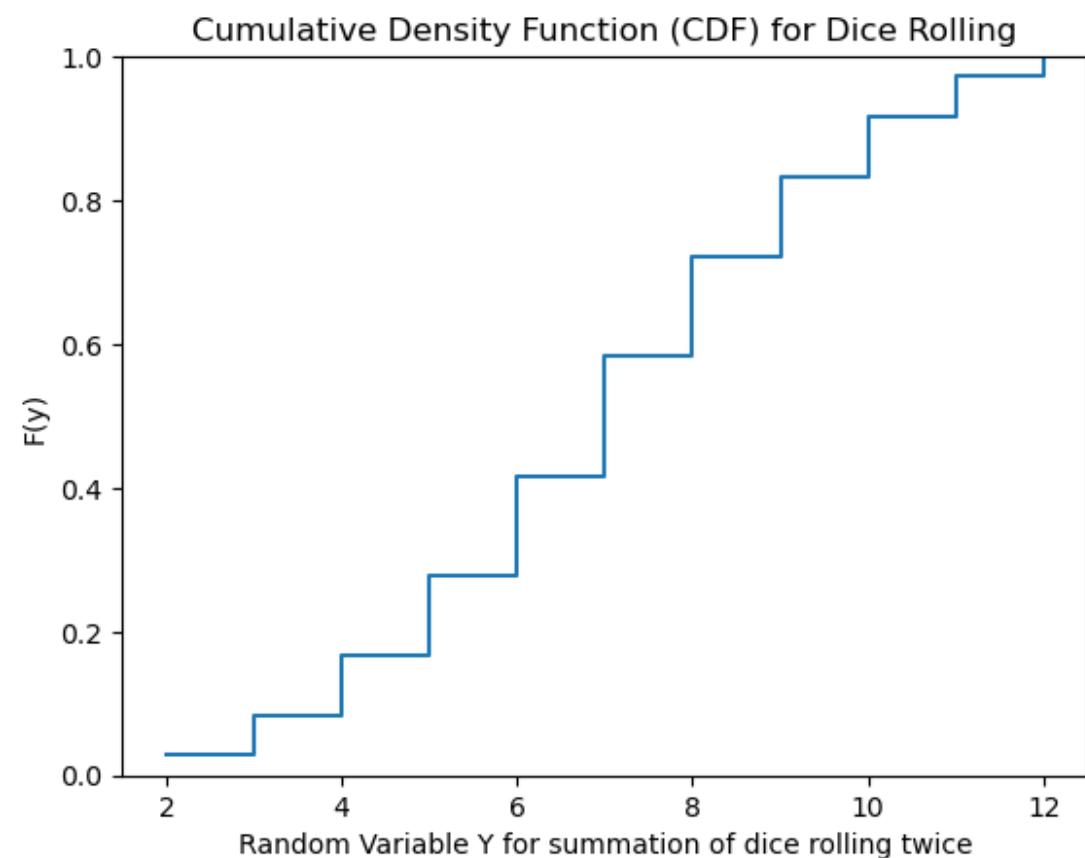
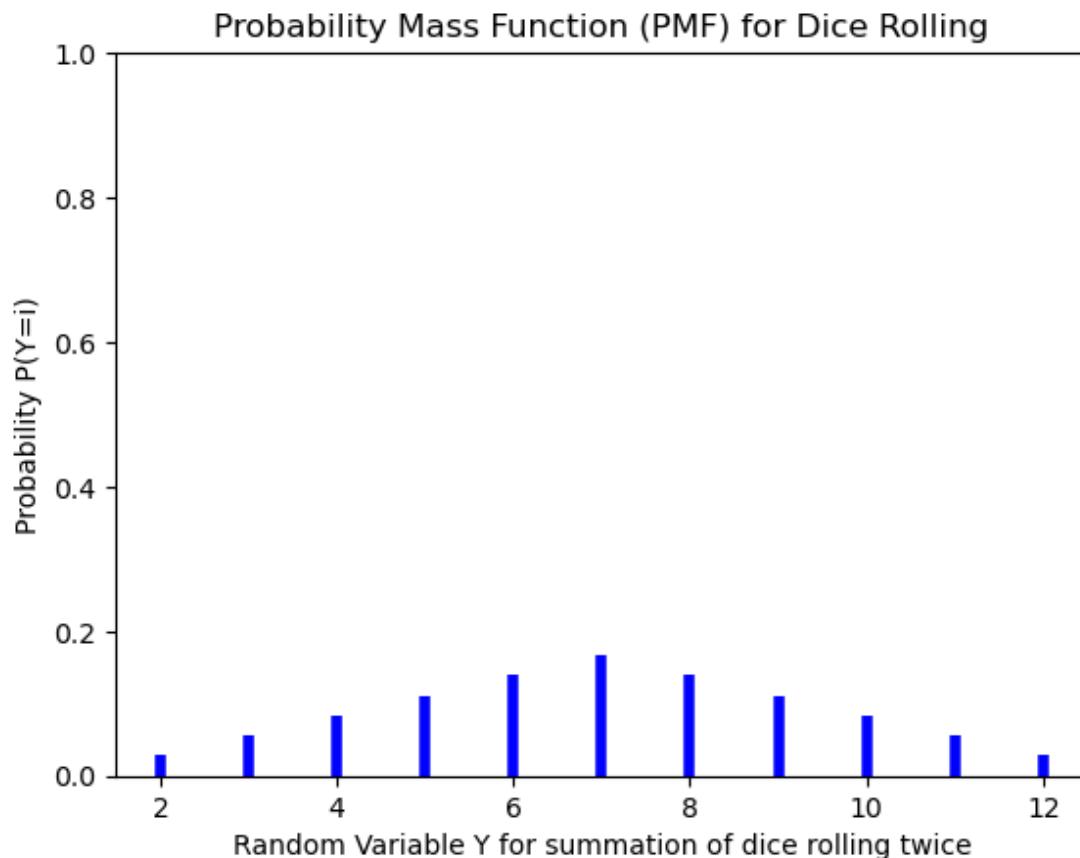
# Cumulative Distribution Function (CDF)



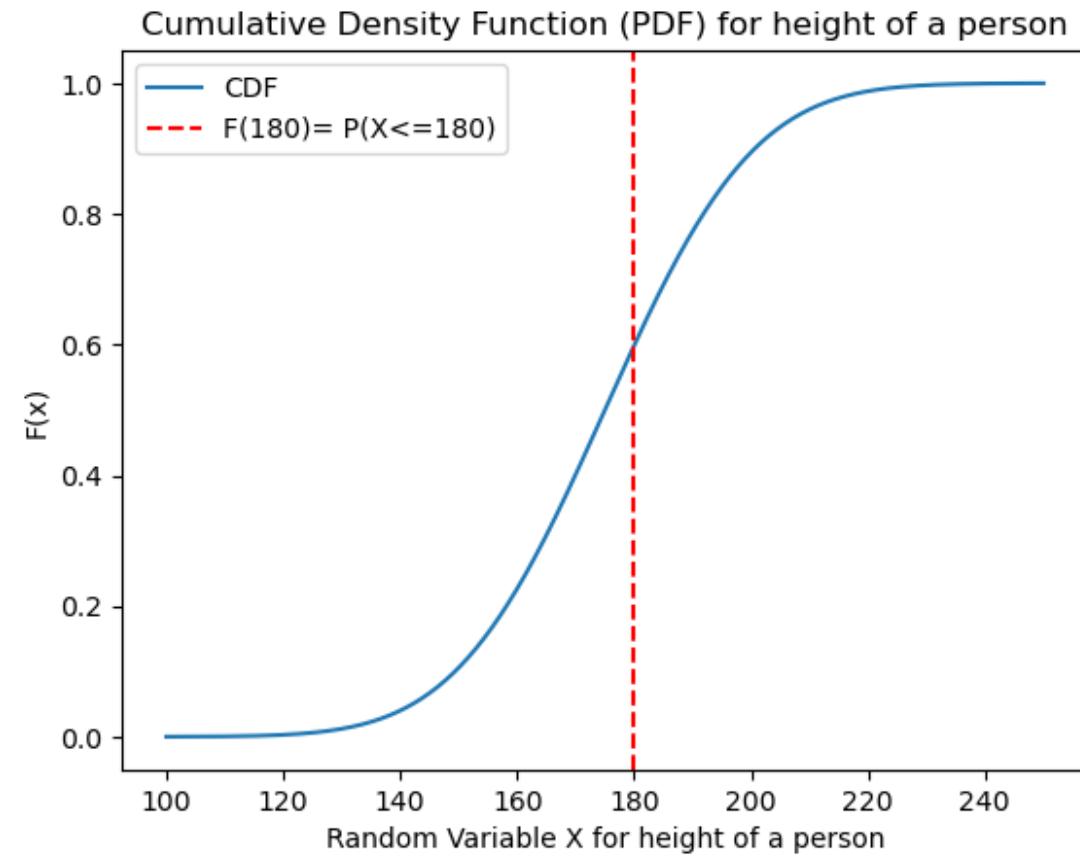
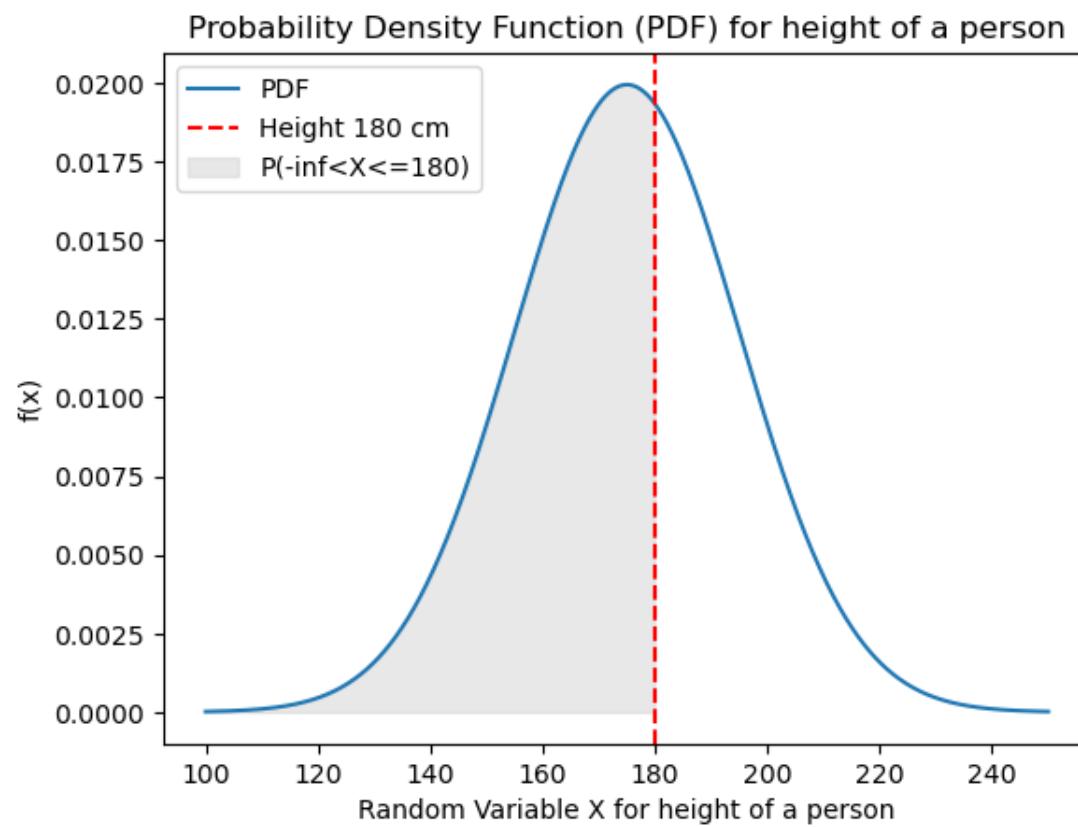
- CDF provides information about the probability that a random variable takes on a value less than or equal to a specified point.
- For Discrete Random Variables:
  - $F(x) = P(X \leq x) = \sum_{x_i \leq x} P(X = x_i)$
- For Continuous Random Variables:
  - $F(X) = P(X \leq x) = \int_{-\infty}^x f(t) dt$



# Cumulative Distribution Function (CDF)



# Cumulative Distribution Function (CDF)



# Recall: Naïve Bayes Theorem



- General Naïve Bayes Formulation:

$$P(y|x_1, x_2, x_3, \dots) = \frac{P(x_1, x_2, x_3, \dots | y) * P(y)}{P(x_1, x_2, x_3, \dots)} =$$
$$\frac{[P(x_1|y) * P(x_2|y) * \dots] * P(y)}{P(x_1) * P(x_2) * \dots}$$

- How did we calculate  $P(x_1|y)$  in the Spam Filter?

# Bernoulli Naïive Bayes



- Suited for binary or binarized data where each feature represents the presence or absence of a certain characteristic.
- Primarily used for binary data (Bernoulli Distribution)
- $P(x_1|y)$  is estimated by counting occurrences of True/False values in the feature.

| Spam (S) | AWARD (A) | GIFT (G) |
|----------|-----------|----------|
| Yes      | No        | Yes      |
| Yes      | Yes       | No       |
| No       | No        | No       |
| Yes      | Yes       | No       |
| No       | Yes       | Yes      |
| Yes      | No        | No       |

# MultiNomial Naiive Bayes



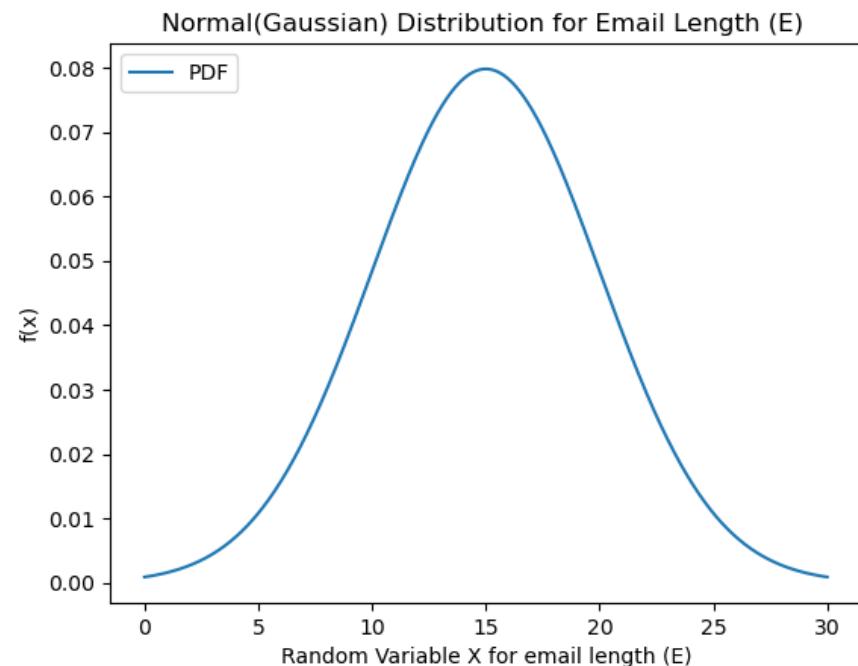
- It estimates the probability of observing counts among a fixed, discrete set of categories.
- Primarily used for discrete data with multinomial distribution.
- $P(x_1|y)$  is estimated by counting occurrences of each value in the given feature .

| Spam (S) | IMPORTANT WORD (G) |
|----------|--------------------|
| Yes      | AWARD              |
| Yes      | GIFT               |
| No       | GIFT               |
| Yes      | AWARD              |
| No       | GIFT               |
| Yes      | AWARD              |

# Gaussian Naive Bayes



- Assumes that the probability of features given the class follows a Gaussian (normal) distribution.
- Assumes that features are continuous and can take any real value.
- $P(x_1|y) = f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$



| Spam (S) | Email Length (E) |
|----------|------------------|
| Yes      | 10.5             |
| Yes      | 12.4             |
| No       | 13.5             |
| Yes      | 17.8             |
| No       | 20.5             |
| Yes      | 3.56             |

# References



- [https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what\\_is\\_machine\\_learning](https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/cho1.html#what_is_machine_learning)
- [https://scikit-learn.org/stable/modules/naive\\_bayes.html#gaussian-naive-bayes](https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes)

# Thank you!

- Any questions?





**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000

# Statistics

---

- Statistics
  - Descriptive Vs. Inferential statistics
- Population and Sample
- Sampling
- Measure of Central Tendency
- Measure of Dispersion
- Normal Distribution
- Z Score



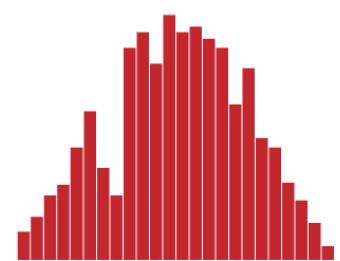
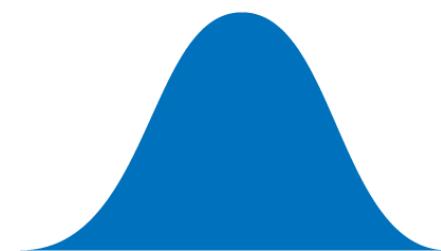
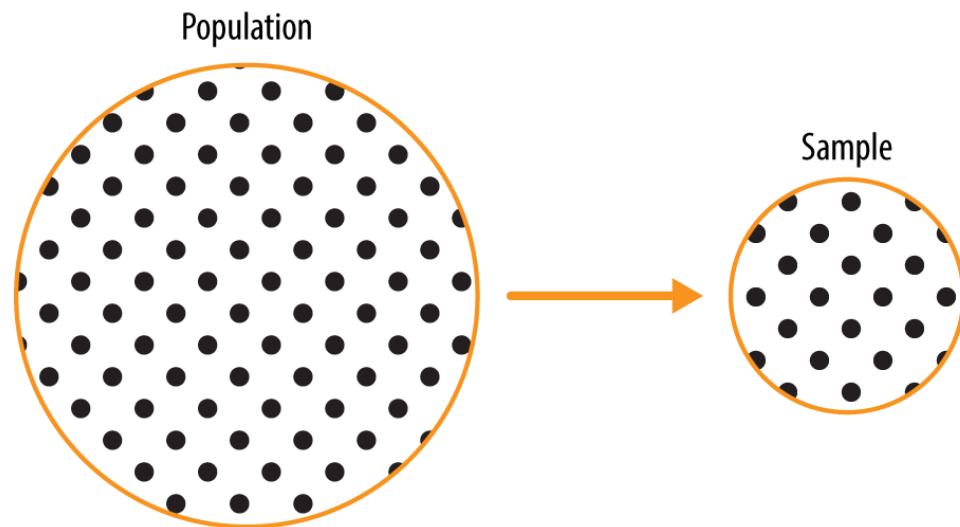


- Science of collecting, organizing and analyzing the data.
- Descriptive Statistics:
  - Used to summarize and describe the main features of a dataset.
  - They provide a clear and concise overview of the data, helping to understand its characteristics.
  - Examples: Mean, median, mode, etc.
- Inferential Statistics:
  - Make inferences or draw conclusions about a population based on a sample of data.
  - Examples: Hypothesis testing, confidence intervals, etc.

# Population and Sample



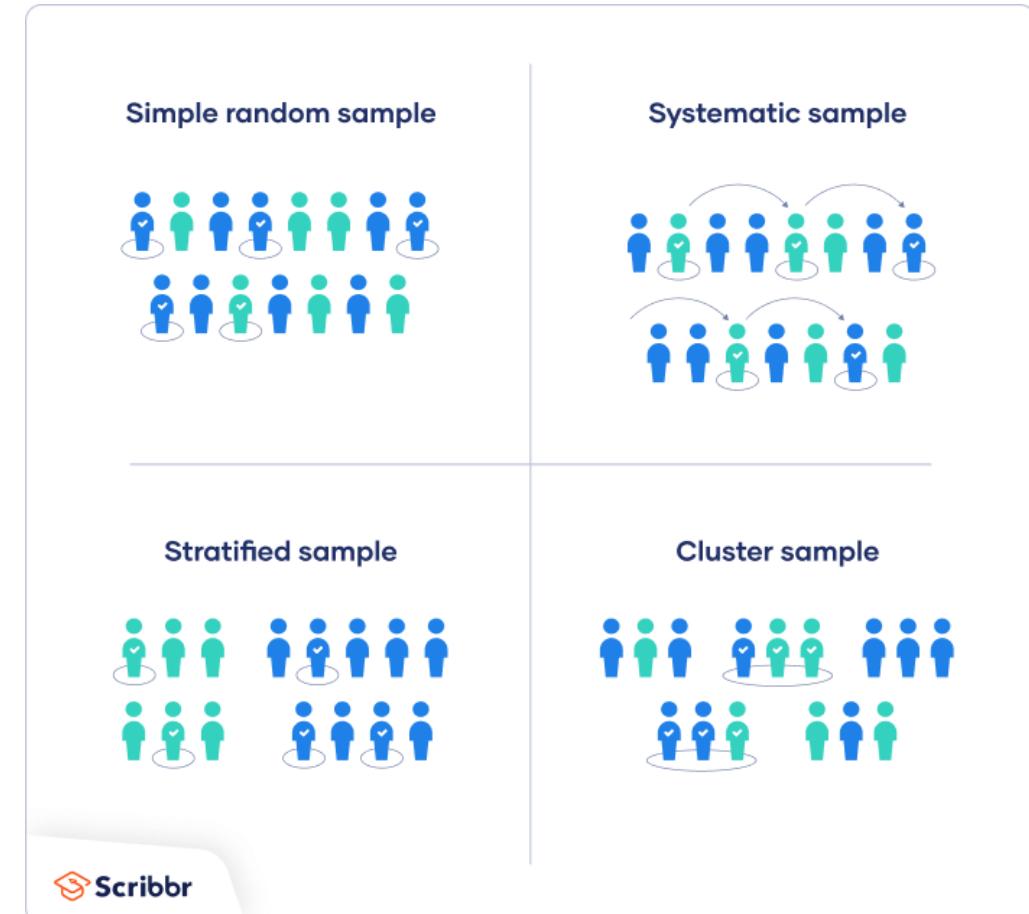
Population – Entire Data (N)  
Sample – Small set of data (n)



# How to generate the samples?



- Simple Random Sampling:
  - Every member of the population has an equal chance of being selected for sample.
  - Best representation of population.
- Stratified Sampling:
  - The population is split into nonoverlapping groups then randomly sample from each group.
  - Beneficial for handling imbalanced datasets.
- Systematic Sampling:
  - From the population, we select the nth individual.
- Convenience Sampling:
  - From the population, we select the sample data that is expertise in the specific domain.



# Measures of Central Tendency



- Central Tendency – the measure used to determine the center of distribution of data.
- Mean(  $\mu$  ) is the average of the population (  $N$  ) or sample(  $n$  ).

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$

- Examples:
  - $S = \{1,1,2,2,3,4,5,1,2,1,3\}$ ,  $\mu(S) = \frac{25}{11} = 2.3$
  - $S = \{1,1,2,2,3,4,5,1,2,1,3,100\}$ ,  $\mu(S) = \frac{350}{11} = 11.3$
- Drawback: Affected by outliers

# Measures of Central Tendency



| Name of the Person | Earning |
|--------------------|---------|
| Sean               | 30000   |
| Carl               | 15000   |
| John               | 20000   |
| Ranny              | 15250   |
| Dansh              | 17500   |
| Tony               | 15500   |
| James              | 10000   |
| Lisa               | 15000   |

Mean:  
13,8250

| Name of the Person | Earning |
|--------------------|---------|
| Sean               | 30000   |
| Carl               | 15000   |
| John               | 20000   |
| Ranny              | 15250   |
| Dansh              | 17500   |
| Tony               | 15500   |
| James              | 10000   |
| Lisa               | 150000  |

Mean:  
1,638,250

Outlier

# Measures of Central Tendency



- Median: It is the middle value in a sorted dataset.
  - Sort the numbers ascendingly
  - Find the center number
    - If odd number of samples → Take single center number as median.
    - If even number of samples → Take average of the two middle numbers.
- Examples:
  - $S = \{1,1,1,1,2,2,2,3,3,4,5\}$ ,
    - $median(S) = 2$
  - $S = \{1,1,1,1,2,2,2,3,3,4,5,100\}$ ,
    - $median(S) = \frac{2+2}{2} = 2$
- Pros: Not affected by outliers

# Measures of Central Tendency

| Name of the Person | Earning |
|--------------------|---------|
| Sean               | 30000   |
| Carl               | 15000   |
| John               | 20000   |
| Ranny              | 15250   |
| Dansh              | 17500   |
| Tony               | 15500   |
| James              | 10000   |
| Lisa               | 15000   |

 *Sorting*

| Name of the Person | Earning |
|--------------------|---------|
| James              | 10000   |
| Carl               | 15000   |
| Ranny              | 15250   |
| Tony               | 15500   |
| Dansh              | 17500   |
| John               | 20000   |
| Sean               | 30000   |
| Lisa               | 150000  |

Median:  
16,500

# Measures of Central Tendency

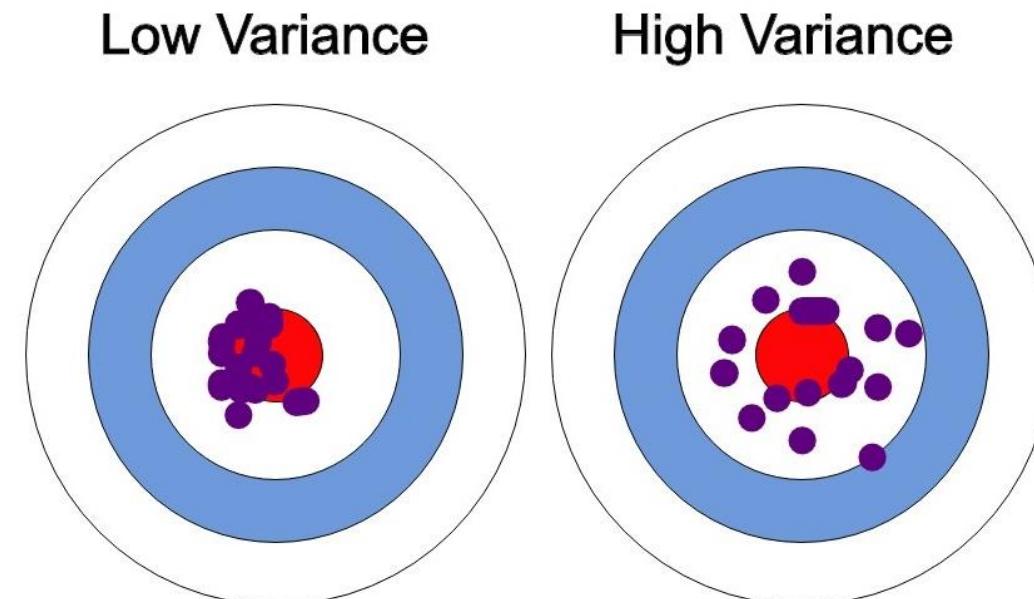


- Mode: the value that occurs most frequently in a dataset.
- Examples:
  - $S = \{1,1,2,7,3,4,5,1,8,1,10\}$ ,
    - $mode(S) = 1 \rightarrow Unimodal$
  - $S = \{1,1,2,2,3,4,5,1,2,1,6\}$ 
    - $mode(S) = \{1,2\} \rightarrow Multimodal$
- Usage:
  - Could fill the missing values in a column using the mode value.

# Measures of Dispersion



- Dispersion – How well spread the data points are?
  - Variance
  - Standard Deviation



# Measures of Dispersion



- Variance: Variance is the average of the squared differences from the Mean.
  - Population Variance -  $\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$
  - Sample Variance -  $s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2 \rightarrow$  Bessel's Correction
- Example:
  - $S = \{1, 2, 2, 3, 4, 6\}$
  - $\mu = 3$
  - $s^2 = \frac{(1-3)^2}{5} + \frac{(2-3)^2}{5} + \dots = 3.2$

# Measures of Dispersion



- Standard Deviation: the square root of the variance. It provides a measure of the average distance between each data point and the mean.

$$\sigma = \sqrt{\sigma^2}$$

- Example:
  - $S = \{1, 2, 2, 3, 4, 6\}$ 
    - $\mu = 3$
    - $s^2 = \frac{(1-3)^2}{5} + \frac{(2-3)^2}{5} + \dots = 3.2$
    - $\sigma = \sqrt{3.2} = 1.78$
- What if we want to study how two variables vary compared to each other?

# Covariance

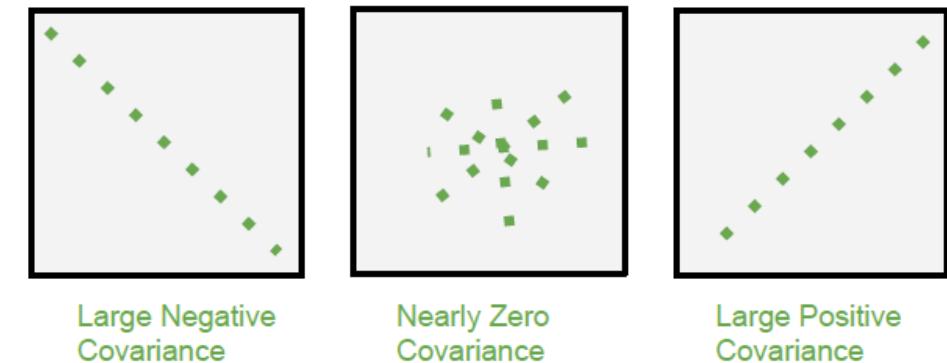


- **Covariance:** It indicates the direction of the linear relationship between two variables. It tells us whether both variables tend to increase or decrease at the same time.

- Positive Covariance: the two variables tend to move in the same direction: as one increases, the other tends to increase, and vice-versa.
- Negative Covariance: the two variables tend to move in opposite directions: as one increases, the other tends to decrease, and vice versa.
- Zero Covariance: No linear relationship exists.

- Main Drawback:

- It can range from negative infinity to positive infinity.
- The magnitude (how large the relationship is) is hard to interpret because it depends on the units of the variables.



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

# Correlation



- **Correlation:** is a standardized version of covariance that measures the strength and direction of the linear relationship between two variables. It provides a value between -1 and 1:

- A correlation of 1 means there's a perfect positive linear relationship (as one variable increases, the other does too).
- A correlation of -1 means there's a perfect negative linear relationship (as one variable increases, the other decreases).
- A correlation of 0 means there is no linear relationship between the variables.

- Formulation:

- $$\text{Corr}(x, y) = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y}$$

| Hours Studied (x) | Exam Score (y) |
|-------------------|----------------|
| 1                 | 2              |
| 2                 | 4              |
| 3                 | 5              |
| 4                 | 4              |
| 5                 | 5              |

$$\text{Cov}(x, y) = ?$$

$$\text{Corr}(x, y) = ?$$

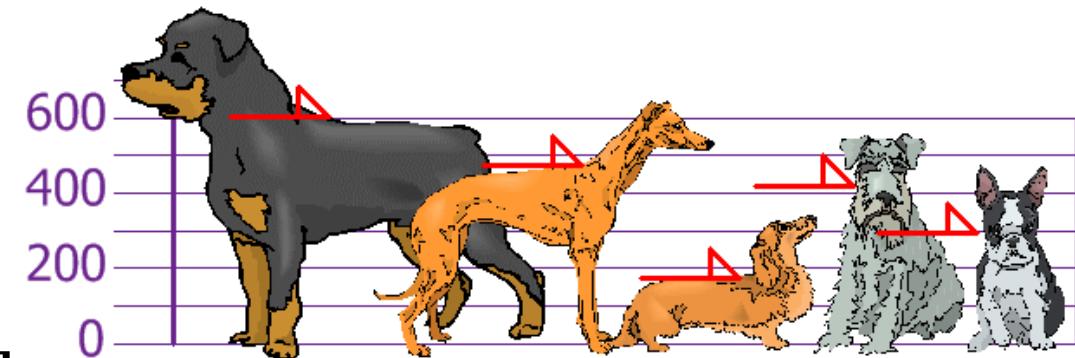
$$\text{Cov}(x, y) = -0.5$$

$$\text{Corr}(x, y) = -0.28$$

# Example



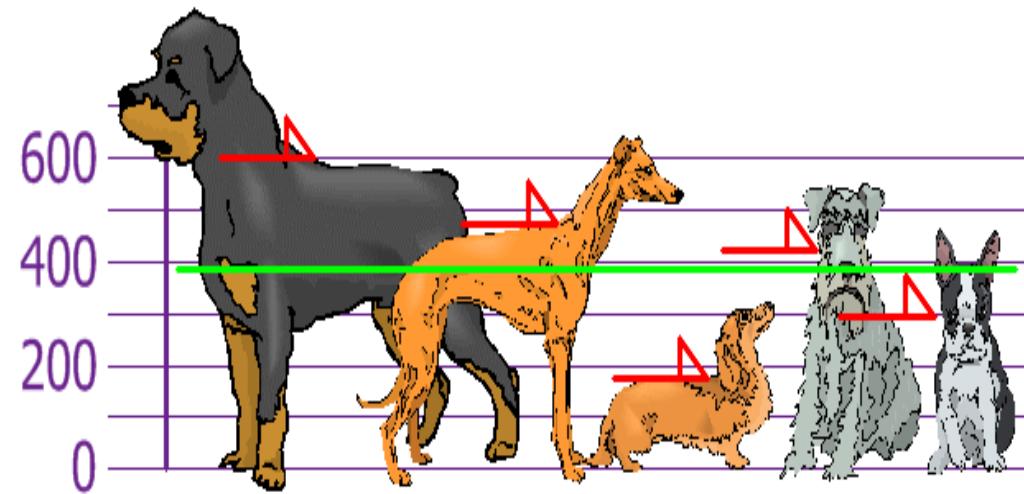
- You and your friends have just measured the heights of your dogs (in millimeters). The heights (at the shoulders) are:  
 $H = [600 \text{ mm}, 470 \text{ mm}, 170 \text{ mm}, 430 \text{ mm} \text{ and } 300 \text{ mm}]$



- Find out the Mean, the Variance, and the Standard Deviation.

# Example: Mean

- $\mu(H) = \frac{600 + 470 + 170 + 430 + 300}{5}$
- $\mu(H) = \frac{1970}{5} = 394$





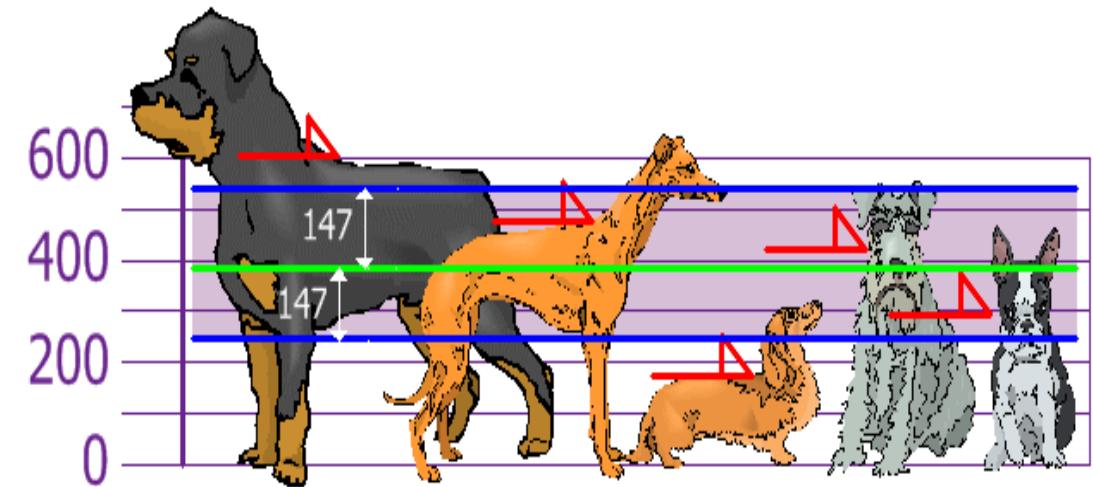
# Example: Variance

- $\sigma^2 = \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5}$
- $\sigma^2 = \frac{42436 + 5776 + 50176 + 1296 + 8836}{5}$
- $\sigma^2 = \frac{108520}{5} = 21704$
- $\sigma^2 = \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5}$
- $\sigma^2 = \frac{42436 + 5776 + 50176 + 1296 + 8836}{5}$
- $\sigma^2 = \frac{108520}{5} = 21704$

# Example: Standard Deviation



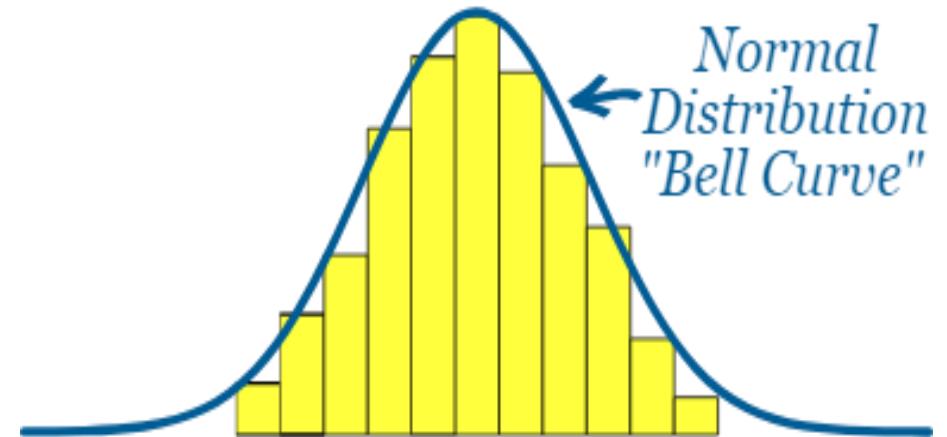
- $\sigma = \sqrt{\sigma^2} = \sqrt{21704} = 147 \text{ mm}$



# Normal distribution



- Data tends to be around a central value with no bias left or right, and it gets close to a "Normal Distribution" like this:



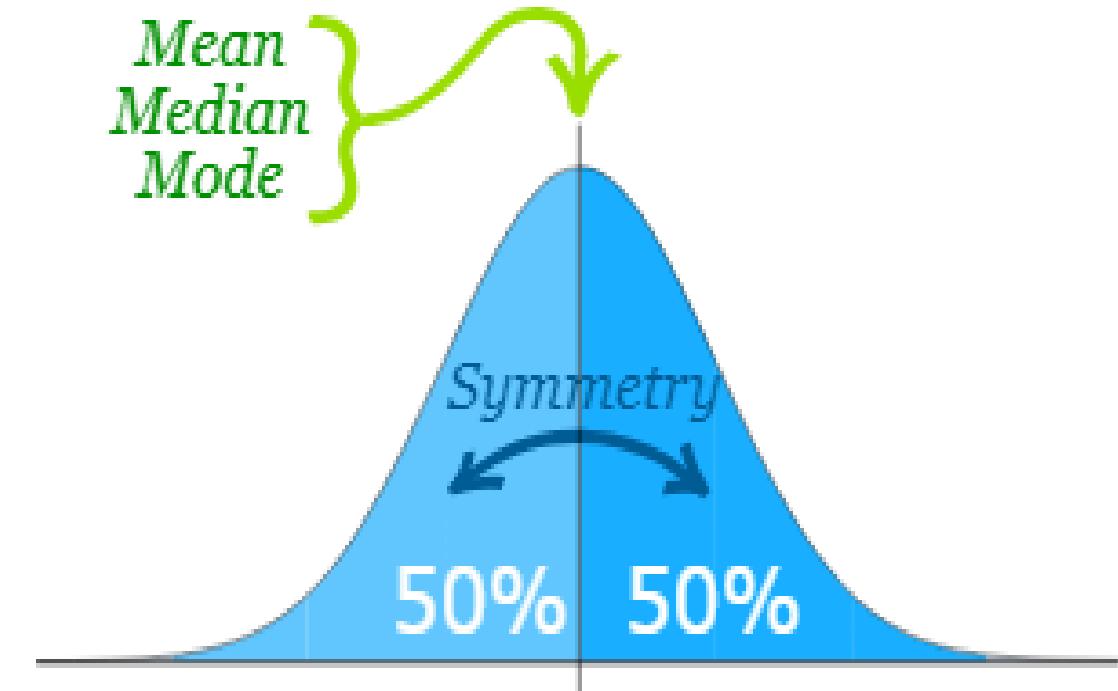
The blue curve is a Normal Distribution. The yellow histogram shows some data that follows it closely, but not perfectly. It is often called a "Bell Curve"

# Normal Distribution



- The Normal Distribution has:

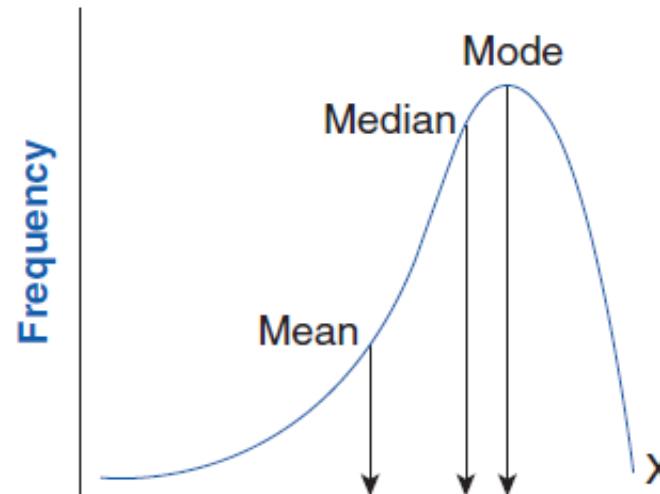
- mean = median = mode
- symmetry about the center
- 50% of values less than the mean
- and 50% greater than the mean



# Normal vs Skewed Distributions

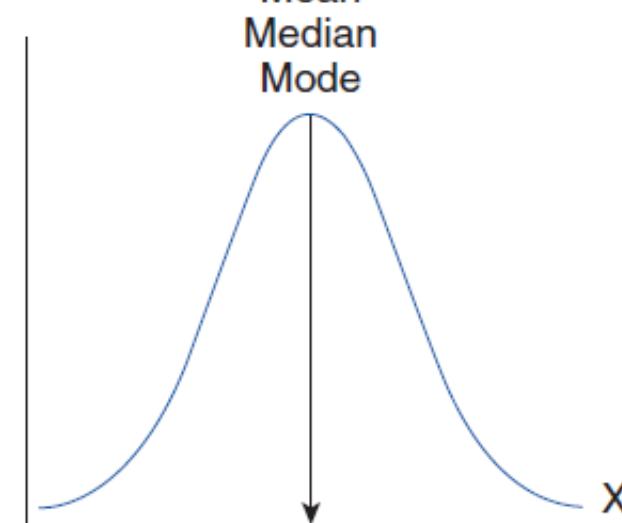


(a) Negatively skewed



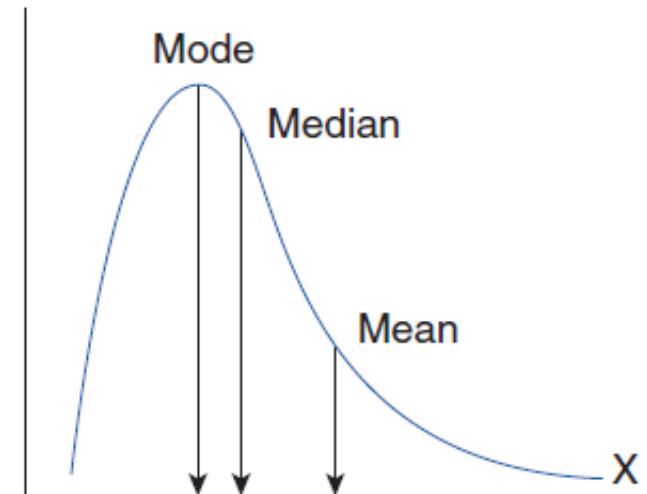
Negative direction

(b) Normal (no skew)



The normal curve represents a perfectly symmetrical distribution

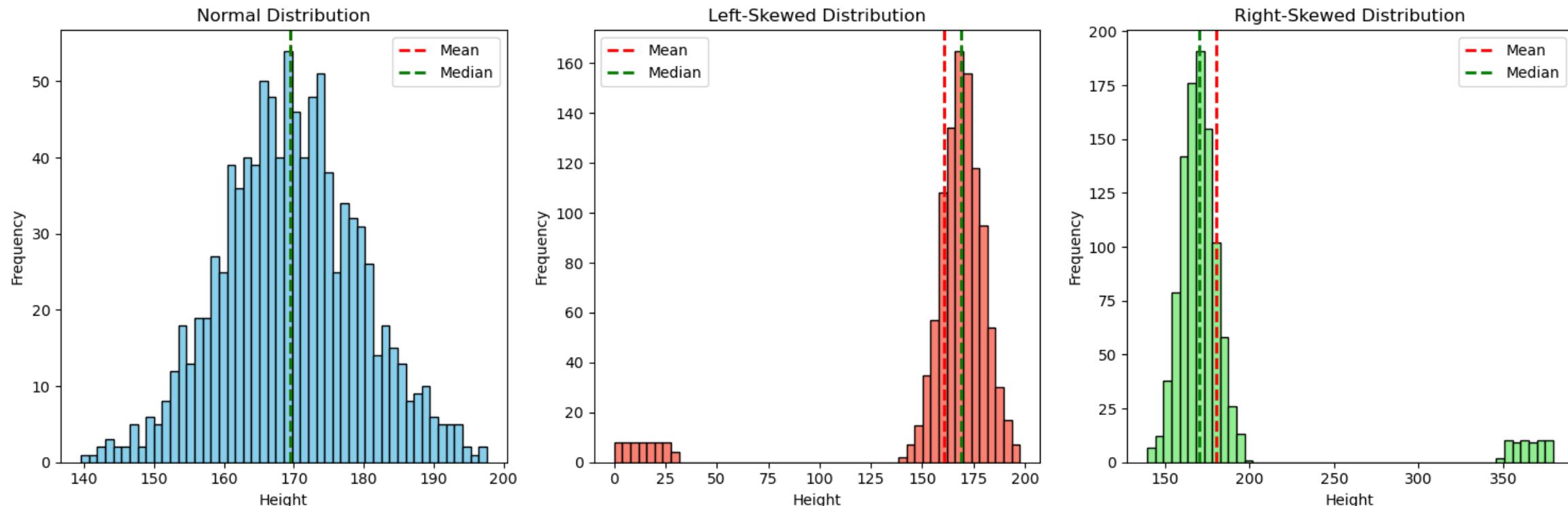
(c) Positively skewed



Positive direction

- In Skewed distributions the **median** is usually utilized instead of the **mean** as a representative center of tendency.

# Normal vs Skewed Distributions: Example

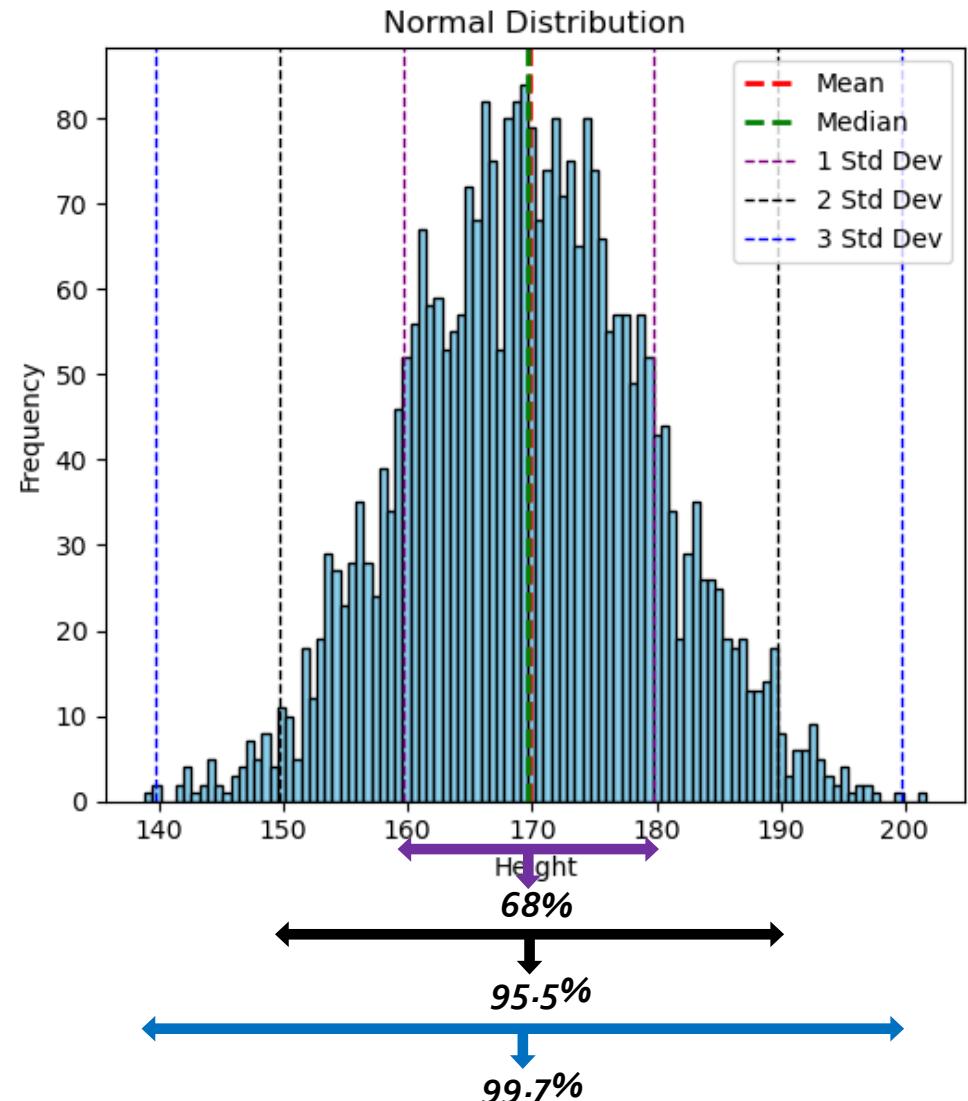


- For this Height feature in a diagnosis prediction dataset, outliers caused skewness of distribution.
- The median is least affected by this compared to the mean.

# Gaussian /Normal Distribution



- In a normal distribution:
  - 68% of all values are within 1 standard deviation from mean
  - 95.5% of all values are within 2 standard deviations from mean
  - 99.7% of all values are within 3 standard deviations from mean

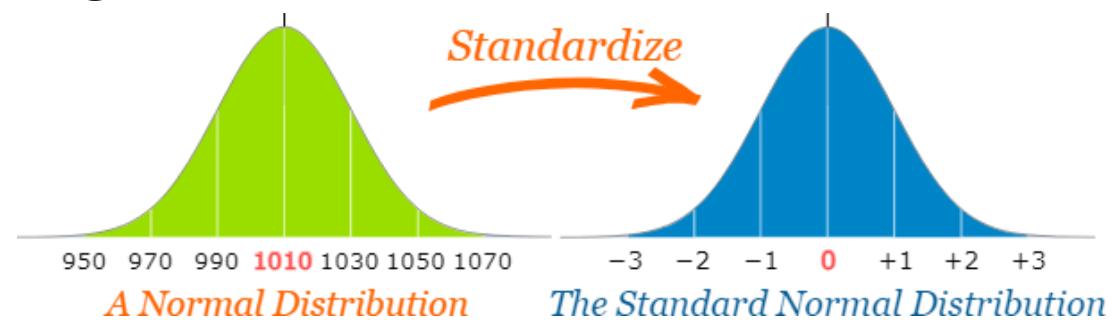


# Z-score / Standard Score



- The number of standard deviations from the mean is also called the "Standard Score", "sigma" or "z-score".
- To convert a value to a Standard Score ("z-score"):
  - first subtract the mean,
  - then divide by the Standard Deviation
  - And doing that is called "Standardizing":
- Formula:

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



# Z-score: Example



- Convert the following dataset values to Z-score representation.

| Name of the Person | Earning |
|--------------------|---------|
| James              | 10000   |
| Carl               | 15000   |
| Ranny              | 15250   |
| Tony               | 15500   |
| Dansh              | 17500   |
| John               | 20000   |
| Sean               | 30000   |
| Lisa               | 150000  |

$$\mu = 34156.25, \sigma = 44117.55$$

| Name of the Person | Earning |
|--------------------|---------|
| James              | -0.548  |
| Carl               | -0.434  |
| Ranny              | -0.429  |
| Tony               | -0.423  |
| Dansh              | -0.378  |
| John               | -0.321  |
| Sean               | -0.094  |
| Lisa               | 2.626   |

# Outlier Removal: Z-Score

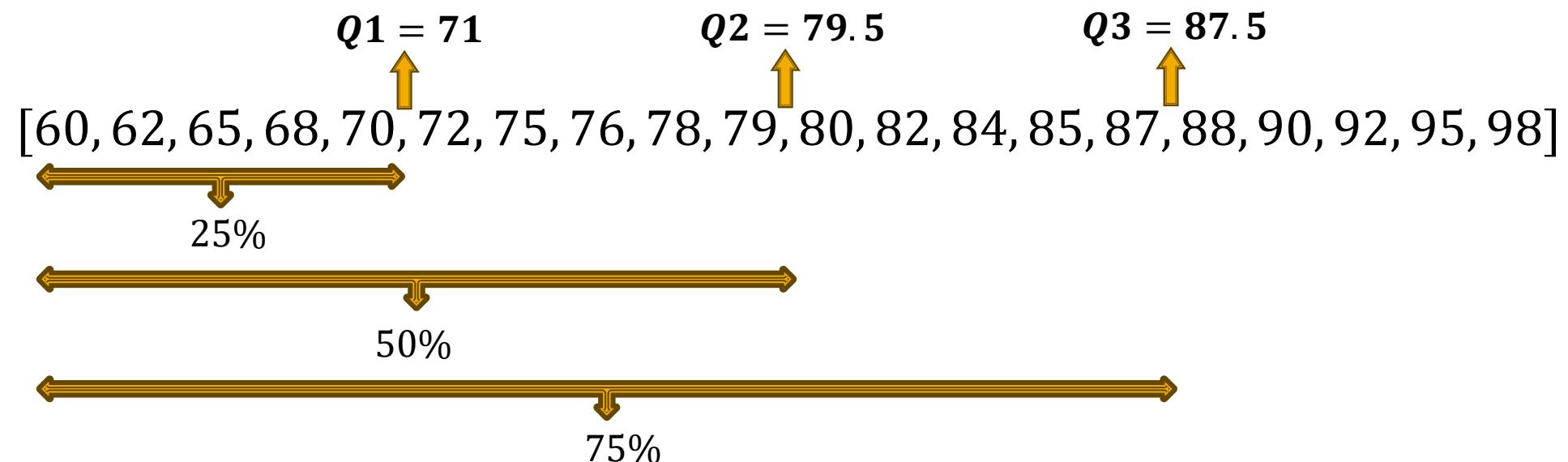


- Data points that have a z-score above a certain threshold  $\tau$  could be labeled as outliers.
- Steps:
  - Normalize the feature to Z-score representation.
  - Based on threshold  $\tau$ , any value larger than  $\tau$  is an outlier
  - Remove the outliers.
- $\tau$  usually is set to a number between 2 and 3, but it could be any arbitrary value.

# Percentiles and Quartiles



- Assume we have a dataset of 20 student exam scores.  
[75, 80, 65, 90, 85, 70, 88, 92, 78, 68, 95, 60, 72, 82, 98, 76, 84, 62, 79, 87]
- Let's start by sorting the values ascendingly.



# Percentiles and Quartiles

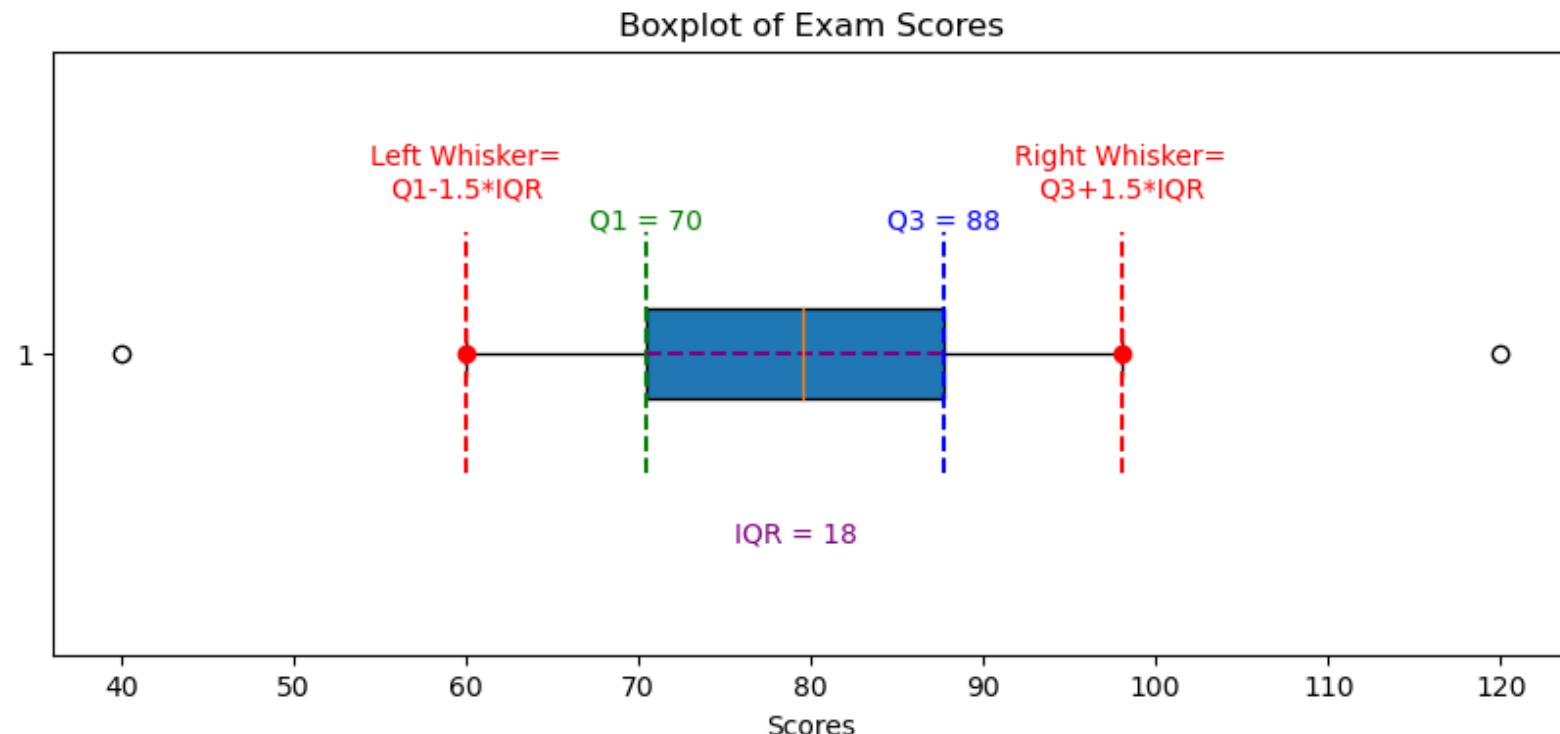


- Quartiles divide a dataset into four equal parts. The three quartiles, denoted as  $Q_1$ ,  $Q_2$ , and  $Q_3$ , represent the following:
  - **$Q_1$  (First Quartile):** This is the value below which 25% of the data falls. It is the median of the lower half of the data set.
  - **$Q_2$  (Second Quartile or Median):** This is the value below which 50% of the data falls. It is the median of the entire dataset.
  - **$Q_3$  (Third Quartile):** This is the value below which 75% of the data falls. It is the median of the upper half of the data set.
- Percentiles are similar to quartiles but are more general. Quartiles specifically divide data into four parts, while percentiles divide data into  $n$  equal parts, where  $n$  is any whole number.

# Plotting Quartiles: Boxplots



$[40, 60, 62, 65, 68, 70, 72, 75, 76, 78, 79, 80, 82, 84, 85, 87, 88, 90, 92, 95, 98, 120]$



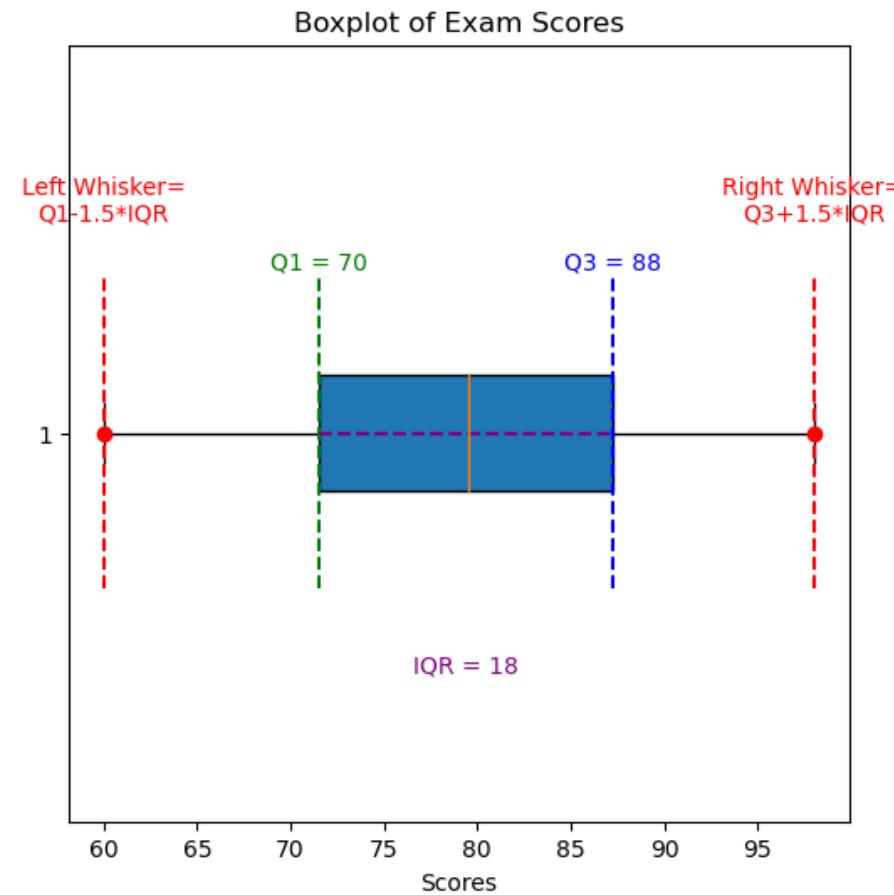
$$\text{Left Whisker} = Q1 - 1.5 * \text{IQR}$$

$$\text{Right Whisker} = Q3 + 1.5 * \text{IQR}$$

# Plotting Quartiles: Boxplots



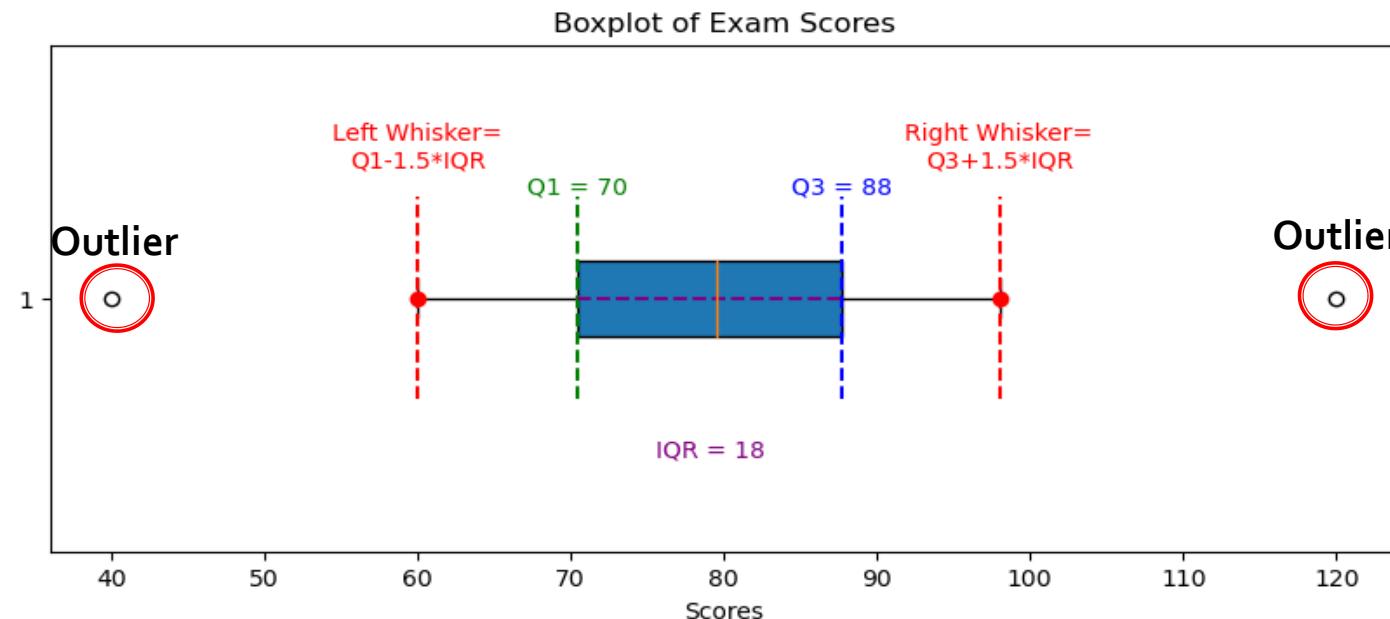
- **Box Plot:** is a graphical representation of a dataset's distribution. It displays a summary of a set of data values including the minimum, first quartile, median, third quartile, and maximum.
- **Left Whisker:** It represents the lower bound of the data within 1.5 times the interquartile range (IQR) below the first quartile (Q1).
- **Right Whisker:** It represents the upper bound of the data within 1.5 times the IQR above the third quartile (Q3).
- **Inter-Quartile Range (IQR):** it represents the spread of the middle 50% of the data. It is calculated as the difference between the third quartile (Q3) and the first quartile



# Outlier Removal: Boxplots



- Data points that have values  $x$  above the right whisker or below the left whisker  $\rightarrow$  labeled as outliers.
- In other words:
  - If  $x > Q3 + 1.5 * IQR$  or  $x < Q1 - 1.5 * IQR \rightarrow$  Outlier



# References



- <https://learning.oreilly.com/library/view/practical-statistics-for/9781491952955/cho6.html>
- <https://www.mathsisfun.com/data/standard-deviation.html>

# Thank you!

- Any questions?





**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Statistics

---

- Continue on statistics:
  - Data preprocessing
  - Feature normalization
  - Data Encoding
  - Feature Engineering
- Linear Regression
- Regression Evaluation Metrics



# Data Preprocessing

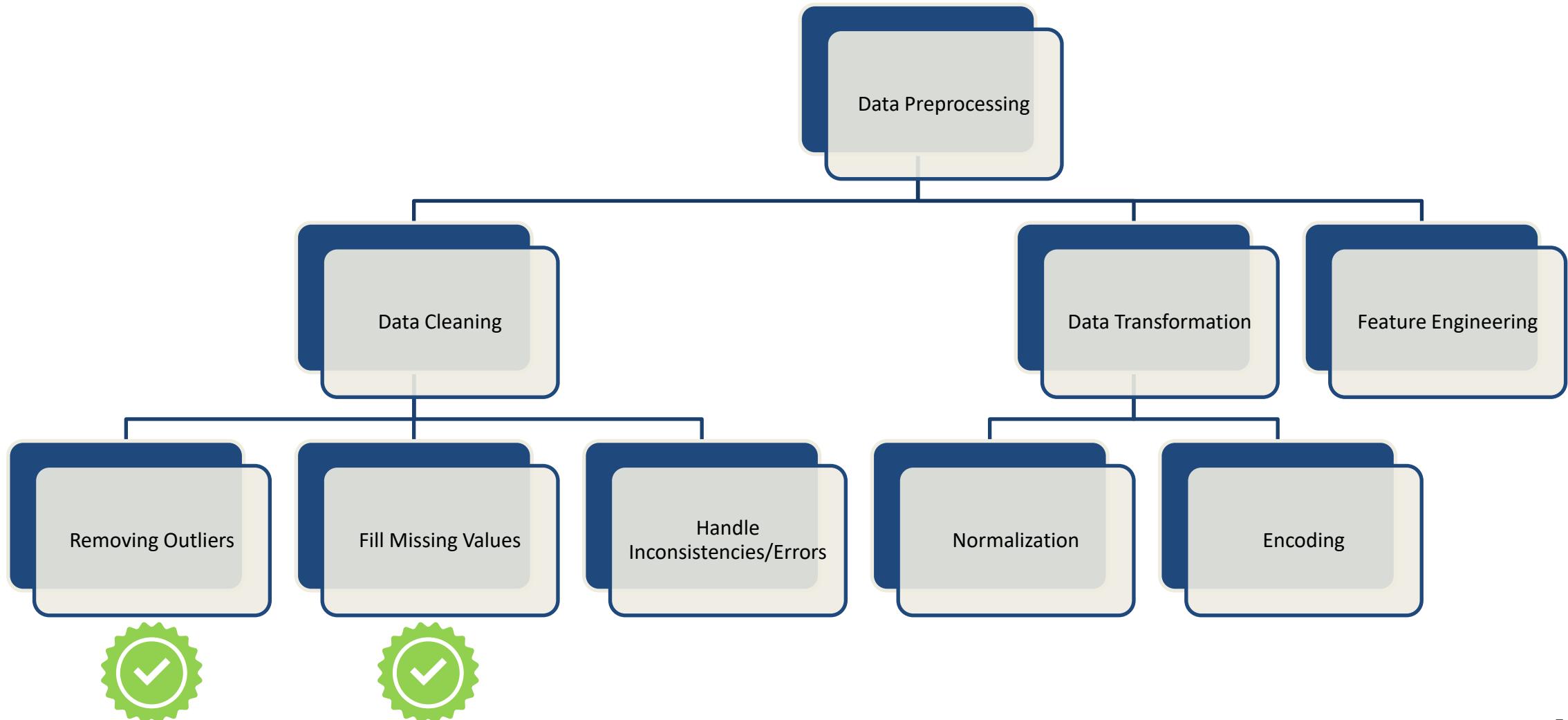


Garbage in, Garbage out

Low-quality data will lead to low-quality and misleading analysis results

(No matter how sophisticated the model is!)

# Data Preprocessing



# Handling Inconsistencies



- A crucial step to correct inconsistencies in the data via fuzzy joins, regular expressions or other methods.

| patientCity | Value Count |
|-------------|-------------|
| Guelph      | 1662        |
| Kitchener   | 1247        |
| Waterloo    | 793         |
| Cambridge   | 330         |
| Fergus      | 204         |
| KITCHENER   | 10          |
| KITTChener  | 21          |
| Geulph      | 12          |
| GUELPH      | 23          |
| WATERLO     | 9           |
| FRGUS       | 13          |

Before

| patientCity | Value Count |
|-------------|-------------|
| Guelph      | 1697        |
| Kitchener   | 1278        |
| Waterloo    | 802         |
| Cambridge   | 330         |
| Fergus      | 217         |

After

# Data Normalization



- A crucial step in preparing data for machine learning algorithms. It helps to ensure that features are on a similar scale, which can lead to more stable and faster convergence during training.
- Allows us to make sure that no variable dominates the other variable.
- There are several ways to perform feature scaling in machine learning.



# Min-Max Scaling

- Definition:
  - Values are shifted and rescaled to range from 0 to 1.
- Formulation:
  - $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$
- Implementation:
  - Sklearn provides MinMaxScaler for this.

| Pros   | Cons  |
|--|---|
| <ul style="list-style-type: none"><li>• Keeps variable relationships intact</li><li>• Suitable for algorithms requiring similar scales</li></ul> | <ul style="list-style-type: none"><li>• Sensitive to outliers</li></ul> |

|    | House  | Cost (\$) | Size (sq. ft) | Cost Scaled | Size Scaled |
|----|--------|-----------|---------------|-------------|-------------|
| 1  | 250000 | 2000      | 0.25          | 0.142       |             |
| 2  | 300000 | 2200      | 0.375         | 0.21        |             |
| 3  | 200000 | 1800      | 0.0           | 0.0         |             |
| 4  | 400000 | 2500      | 0.75          | 0.375       |             |
| 5  | 150000 | 1500      | 0.125         | 0.071       |             |
| 6  | 450000 | 2800      | 1.0           | 1.0         |             |
| 7  | 350000 | 2100      | 0.625         | 0.25        |             |
| 8  | 275000 | 1900      | 0.3125        | 0.118       |             |
| 9  | 325000 | 2300      | 0.5           | 0.429       |             |
| 10 | 275000 | 1600      | 0.3125        | 0.071       |             |

# Z-score Normalization



- Definition:
  - Scales the data to have a mean of 0 and a standard deviation of 1.
- Formulation:
  - $X_{norm} = \frac{x - \mu}{\sigma}$
- Implementation:
  - Sklearn provides StandardScaler for this.

| Pros  |
|---|
| • Maintains shape of original distribution. |
| • Less sensitive to outliers.               |

| Cons   |
|--|
| • Not suitable if needs to maintain the mean and standard deviation. |

| House | Cost (\$) | Size (sq. ft) | Cost Scaled | Size Scaled |
|-------|-----------|---------------|-------------|-------------|
| 1     | 250000    | 2000          | -0.588      | -0.226      |
| 2     | 300000    | 2200          | 0.117       | 0.159       |
| 3     | 200000    | 1800          | -1.293      | -0.543      |
| 4     | 400000    | 2500          | 1.822       | 1.063       |
| 5     | 150000    | 1500          | -1.949      | -1.351      |
| 6     | 450000    | 2800          | 2.117       | 1.866       |
| 7     | 350000    | 2100          | 0.823       | 0.523       |
| 8     | 275000    | 1900          | -0.411      | -0.098      |
| 9     | 325000    | 2300          | 0.529       | 0.764       |
| 10    | 275000    | 1600          | -0.411      | -1.072      |

# Robust Scaling



- **Definition:**
  - Uses the median and the interquartile range (IQR) instead of the mean and standard deviation.
- **Formulation:**
  - $$X_{norm} = \frac{x - \text{median}}{IQR}$$
- **Implementation:**
  - Sklearn provides RobustScaler for this.

| Pros  |
|---|
| <ul style="list-style-type: none"><li>• Least sensitive to outliers compared to Min-Max/Z-score</li></ul> |

| Cons  |
|---|
| <ul style="list-style-type: none"><li>• Still influenced by extreme outliers.</li></ul> |

| House | Cost (\$) | Size (sq. ft) | Cost Scaled | Size Scaled |
|-------|-----------|---------------|-------------|-------------|
| 1     | 250000    | 2000          | -0.25       | 0.0         |
| 2     | 300000    | 2200          | 0.25        | 0.2857      |
| 3     | 200000    | 1800          | -0.75       | -0.2857     |
| 4     | 400000    | 2500          | 1.25        | 0.5714      |
| 5     | 150000    | 1500          | -1.25       | -0.5714     |
| 6     | 450000    | 2800          | 1.5         | 1.1429      |
| 7     | 350000    | 2100          | 0.75        | 0.4286      |
| 8     | 275000    | 1900          | 0.0         | 0.1429      |
| 9     | 325000    | 2300          | 0.5         | 0.8571      |
| 10    | 275000    | 1600          | 0.0         | -0.4286     |

# Box Cox Transformation



- Definition:
  - Used to stabilize the variance and make the data more normally distributed.
- Formulation:

$$X_{norm} = \begin{cases} \log(X), & \text{if } \lambda = 0 \\ \frac{X^\lambda - 1}{\lambda}, & \text{otherwise} \end{cases}$$

- Implementation:
  - Scipy.stats provides boxcox() for this.

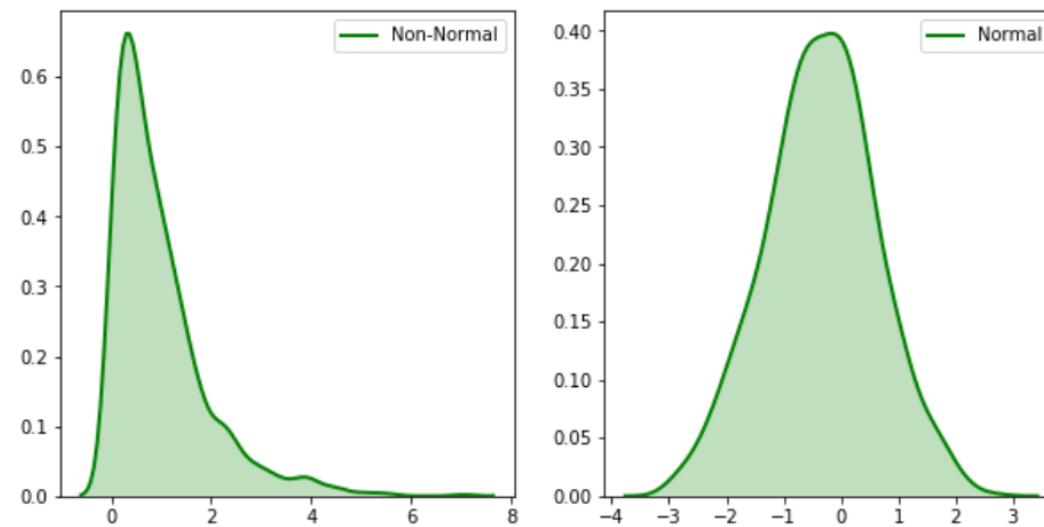
## Pros

- Best if the algorithm requires normal distributions

## Cons

- Assumes that all values are strictly positive

Lambda value used for Transformation: 0.30656155175590766



# Data Encoding



- Its primary purpose is to transform categorical variables, which represent qualitative attributes, into a numerical format that can be effectively utilized by mathematical models.
- This conversion is imperative because most machine learning algorithms are designed to operate on numerical data
- There are several ways to perform data encoding in machine learning.



# Label Encoding

- Assigns a unique integer to each category.
- Suitable for ordinal categorical variables with a clear order.
- May introduce unintended ordinal relationships.

| Sample | Education Level   |
|--------|-------------------|
| 1      | High School       |
| 2      | Bachelor's Degree |
| 3      | Master's Degree   |
| 4      | High School       |
| 5      | PhD               |
| 6      | Bachelor's Degree |
| 7      | High School       |
| 8      | Master's Degree   |
| 9      | Bachelor's Degree |
| 10     | High School       |

| Sample | Encoded Education Level |
|--------|-------------------------|
| 1      | 0                       |
| 2      | 1                       |
| 3      | 2                       |
| 4      | 0                       |
| 5      | 3                       |
| 6      | 1                       |
| 7      | 0                       |
| 8      | 2                       |
| 9      | 1                       |
| 10     | 0                       |

# One-Hot Encoding



- Represents each category as a binary vector.
- Suitable for nominal categorical variables.
- Avoids the assumption of ordinality between categories.
- Can lead to high-dimensional data if there are many categories.

| Sample | Favorite Color |
|--------|----------------|
| 1      | Red            |
| 2      | Blue           |
| 3      | Green          |
| 4      | Red            |
| 5      | Blue           |
| 6      | Green          |
| 7      | Red            |
| 8      | Blue           |
| 9      | Green          |
| 10     | Red            |

| Sample | Red | Blue | Green |
|--------|-----|------|-------|
| 1      | 1   | 0    | 0     |
| 2      | 0   | 1    | 0     |
| 3      | 0   | 0    | 1     |
| 4      | 1   | 0    | 0     |
| 5      | 0   | 1    | 0     |
| 6      | 0   | 0    | 1     |
| 7      | 1   | 0    | 0     |
| 8      | 0   | 1    | 0     |
| 9      | 0   | 0    | 1     |
| 10     | 1   | 0    | 0     |

# Feature Engineering

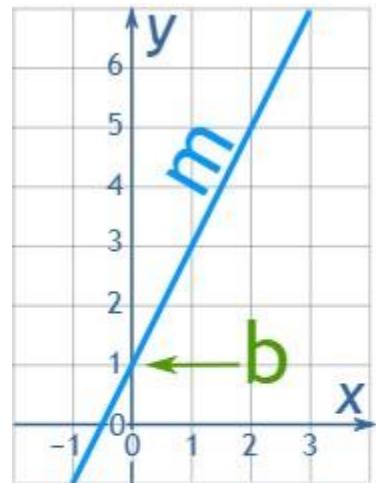


- Involves creating new features or modifying existing ones to improve the performance of machine learning models.
- Example:
  - Combine two or more existing features to create new ones
  - Create summary statistics (i.e. fill with mean, median per another categorical feature)

| Height (cm) | Weight (kg) | BMI   |
|-------------|-------------|-------|
| 165         | 70          | 25.71 |
| 170         | 68          | 23.53 |
| 155         | 60          | 24.97 |
| 180         | 75          | 23.15 |
| 160         | 65          | 25.39 |
| 175         | 72          | 23.51 |

# Machine Learning Algorithms

# Recall Equation of Line



$$y = mx + b$$

Slope or  
Gradient

**y** value when **x=0**  
(see Y Intercept)

**y** = how far up

**x** = how far along

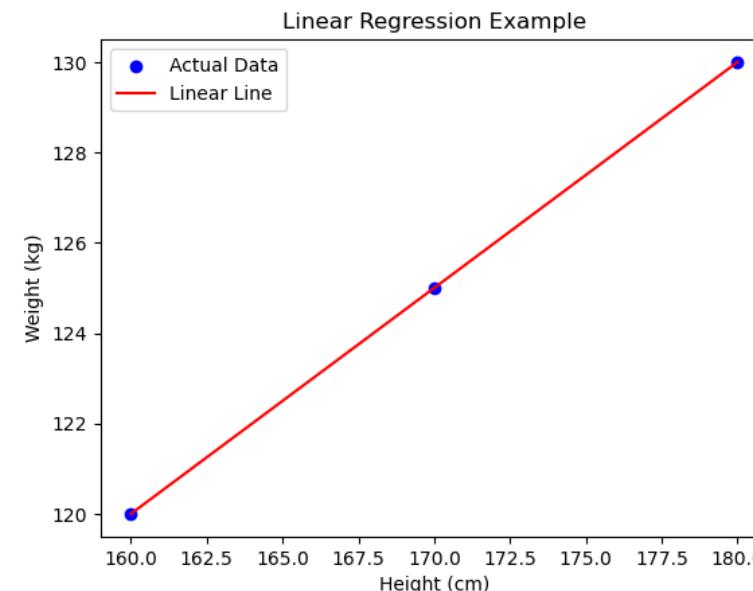
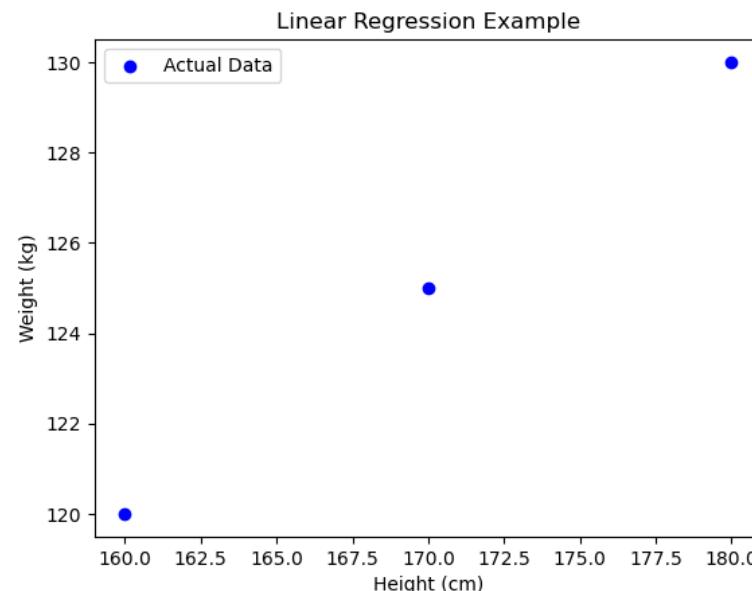
**m** = Slope or Gradient (how steep the line is)

**b** = value of **y** when **x=0**

# Linear Regression: Formulation



- Assume we have a set of three 2D points where the x-axis represents Height and y-axis represents Weight, such that  $[(160,120), (170,125), (180,130)]$ . Can we directly compute the equation of line passing through the three points?

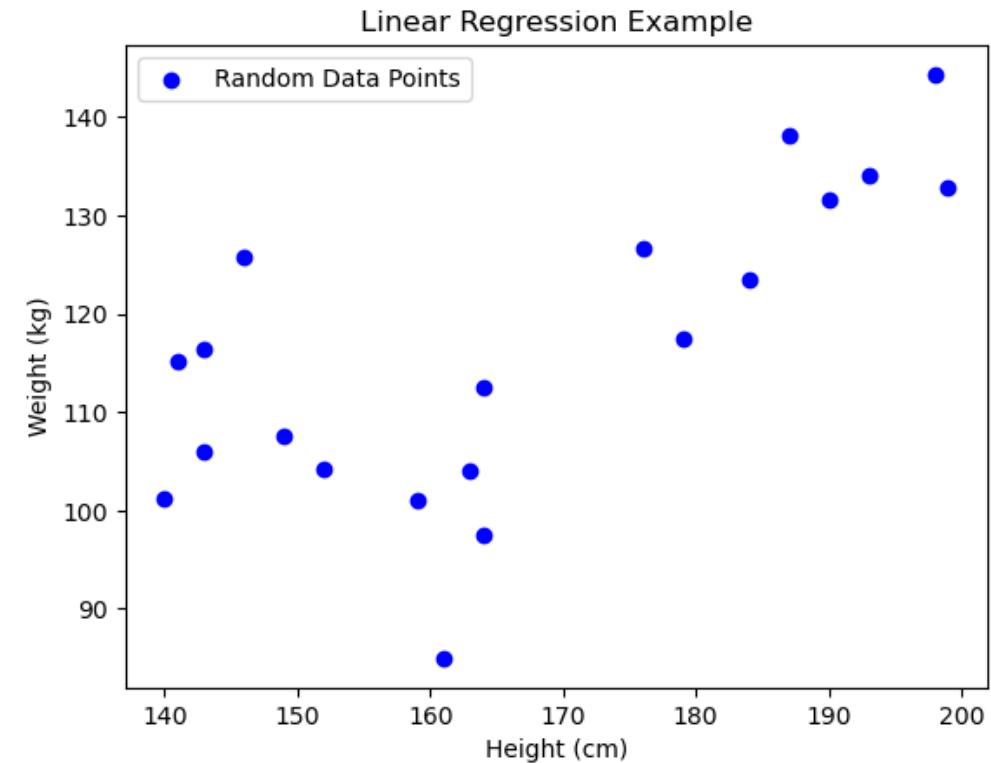


$$y = \frac{1}{2}x + 40$$

# Linear Regression: Formulation



- Assume we have a set of 20 randomly scattered 2D points where the x-axis represents Height and y-axis represents Weight. Can we directly compute the equation of line passing through **all the points**?



$$y = m x + c$$

$$m = ?, c = ?$$

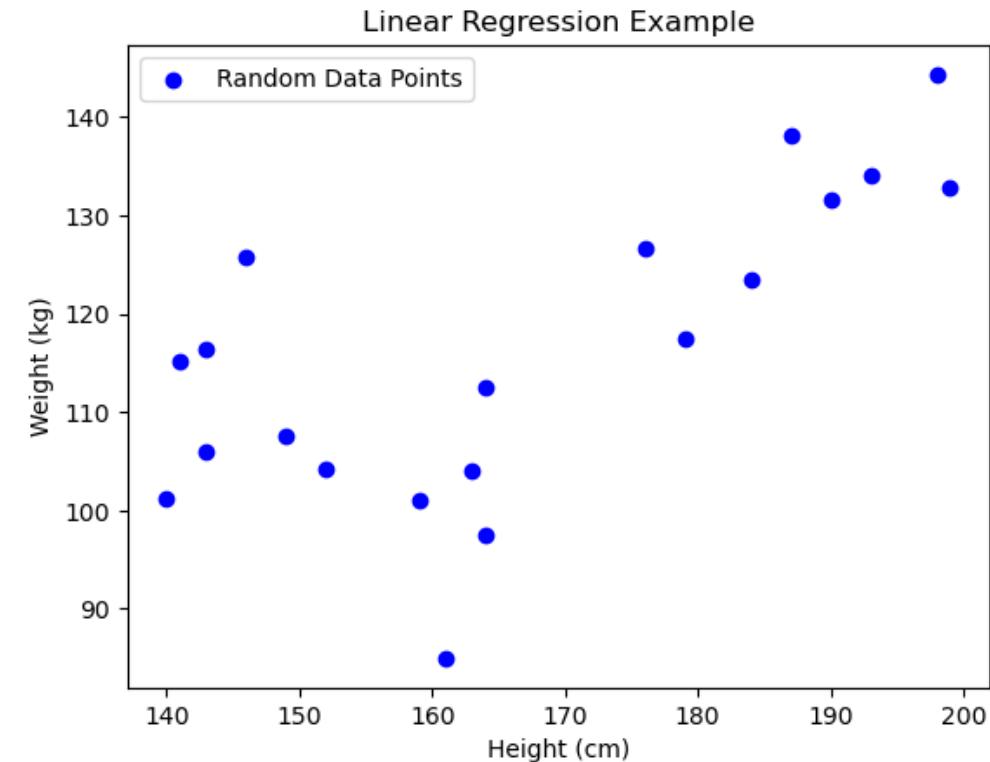
# Linear Regression: Formulation



- Linear regression is a supervised learning algorithm which allows us to find the **best fit** line/hyperplane passing through the set of available data points.
- The predicted **best fit line** equation corresponds to predicting a continuous variable  $\hat{y}$  given input features  $x$ , such that:

$$\hat{y} = \vec{w} \vec{x} + b$$

- $\vec{w}, b$  are the missing parameters that need to be estimated to get the best fit line equation.

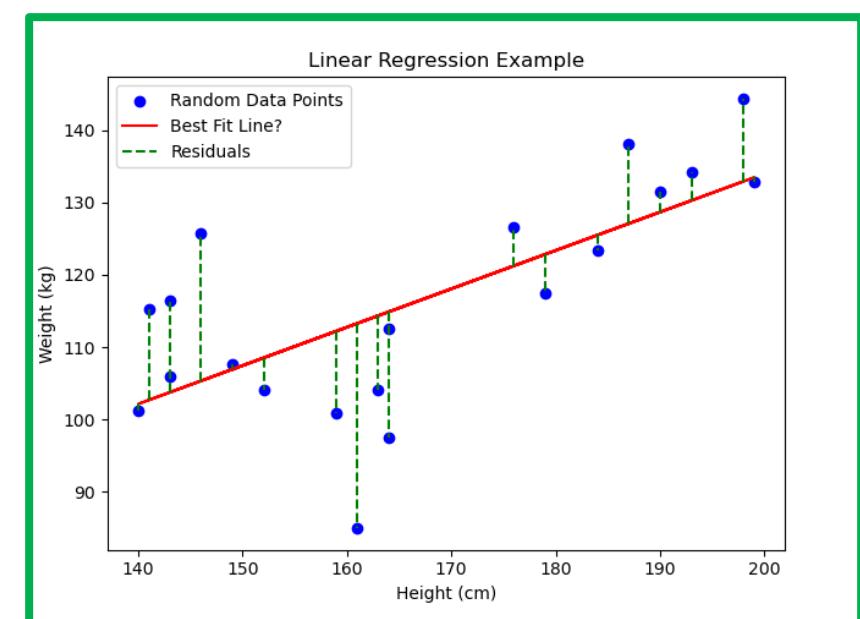
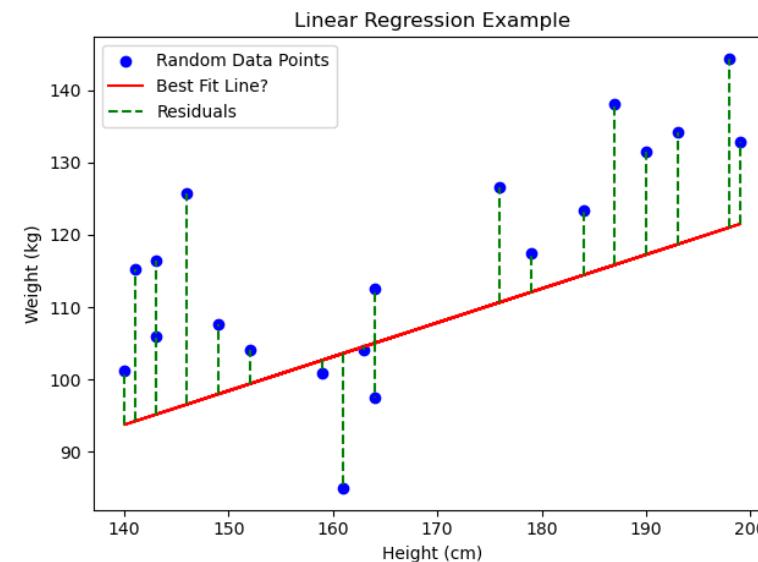
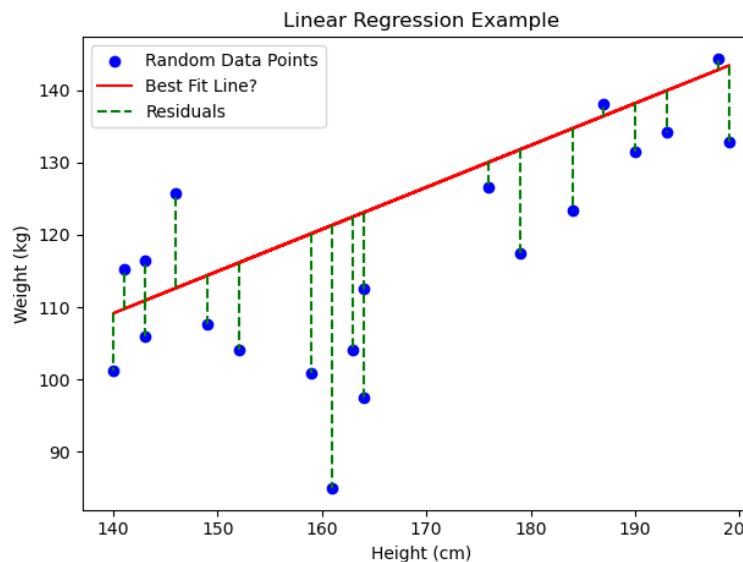


$$w = ?, b = ?$$

# Linear Regression: Cost Function



- By definition, the **best fit** line is one that has the minimum distances (residuals) between itself and all the data points available.
- Which of these could be the **best fit** line?



# Linear Regression: Cost Function



- By definition, the **best fit** line is one that has the minimum distances (residuals) between itself and all the data points available.
- To find the best fit line, we need to **minimize** the average of the squared distances between the predictions  $\hat{y}$  and the actual output  $y$ , such that:

$$L(\vec{w}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \frac{1}{N} \sum_{i=1}^N (y_i - (\vec{w} \cdot \vec{x}_i + \mathbf{b}))^2$$

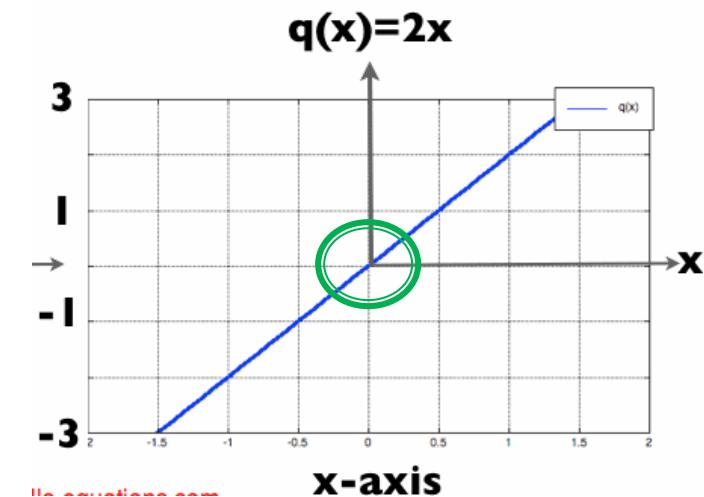
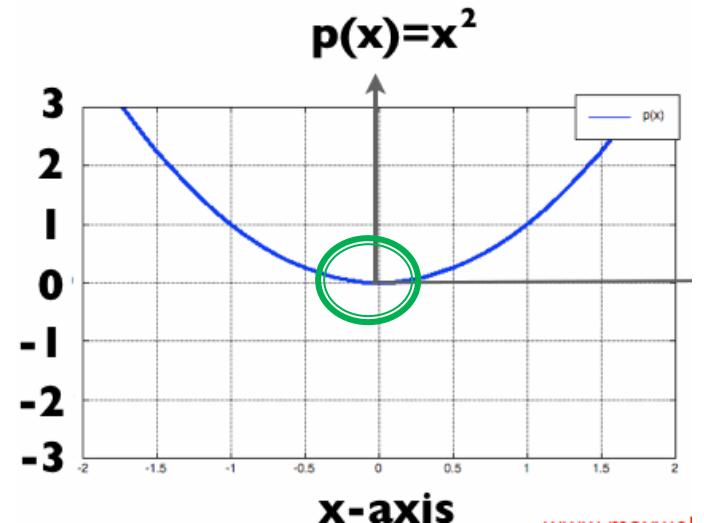
- $L$  is usually referred to as “Loss Function” or “Cost Function”.
- Our target is to find the value of  $\vec{w}$  and  $\mathbf{b}$  at which  $L$  is **minimum**.

# Linear Regression: Loss Minimization



- Given a function  $f(x)$ , how to get the  $x$  value at which  $f(x)$  is **minimum** ?
- In general, one should get the  $x$ -value at which the derivative (differentiation) of  $f(x)$  with respect to  $x$  is **equal to zero**, such that,

$$\frac{d(f(x))}{dx} = 0$$



# Linear Regression: Loss Minimization



$$L(\vec{w}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^N (y_i - (\vec{w} \cdot \vec{x}_i + \mathbf{b}))^2$$

- Get the values of  $\vec{w}, \mathbf{b}$  at which  $L(\vec{w}, \mathbf{b})$  is minimum  $\rightarrow$  Get the values of  $\vec{w}, \mathbf{b}$  at which  $\frac{d(L)}{d\vec{w}} = 0$ , and  $\frac{d(L)}{d\mathbf{b}} = 0$ .
- For linear regression,  $\frac{d(L)}{d\vec{w}} = 0$  and  $\frac{d(L)}{d\mathbf{b}} = 0$  both have **closed-form** solutions that can be derived by making  $\vec{w}$  and  $\mathbf{b}$  the subjects of their equations.
- In this case, the closed-form solution corresponds to:

$$\begin{bmatrix} \vec{w} \\ \mathbf{b} \end{bmatrix} = (X^T X)^{-1} X^T y$$

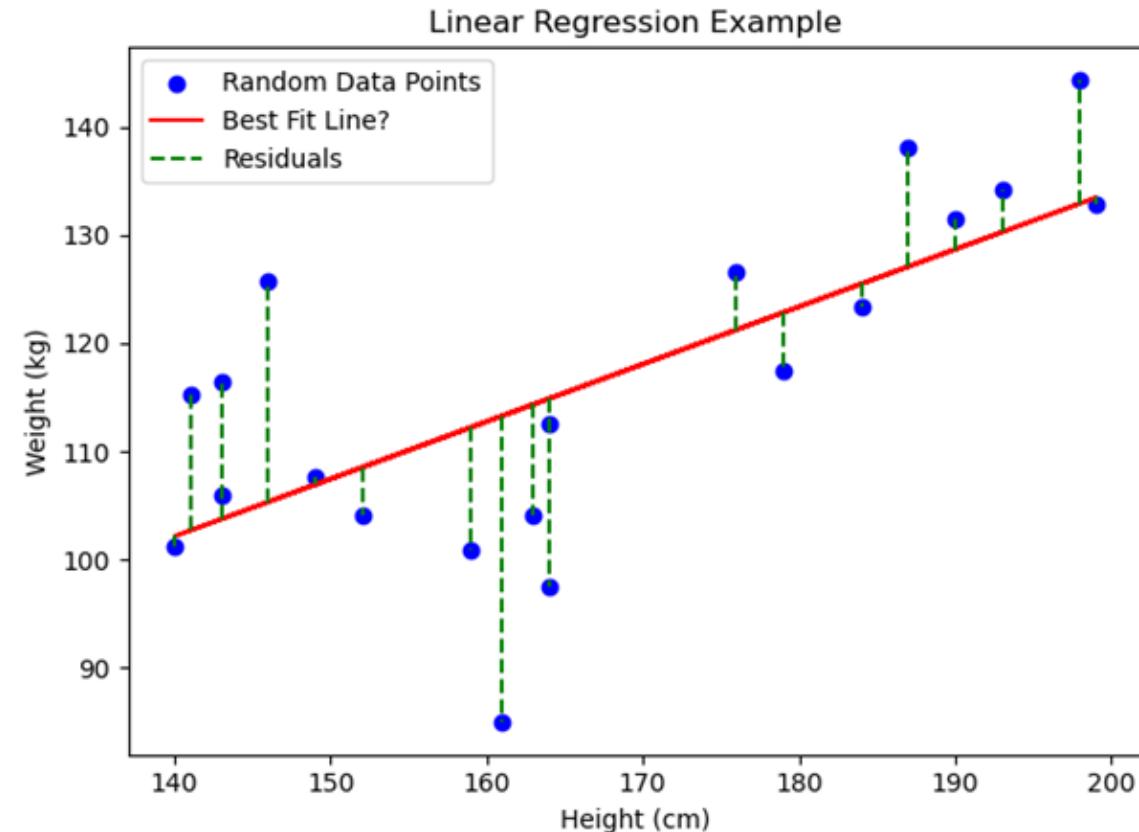
- $X$  is a matrix where each row represents a data point and each column represents a feature,  $y$  is a vector of needed output.

# Linear Regression: Loss Minimization



- There are other machine learning algorithms for which their cost functions don't have a closed-form solution.
- In other words, we cannot set  $\vec{w}$  and  $b$  the subject of their equations  $\frac{d(L)}{d\vec{w}} = 0$  and  $\frac{d(L)}{db} = 0$ , respectively.
- For that reason, we utilize iterative optimization approaches like the famous ***Gradient Descent*** algorithm.

# Linear Regression: Solution



$$\hat{y} = w_1 x + b$$

$$w = 0.533, b = 27.94$$

# Linear Regression: Single vs Multiple Variables

## Linear Regression: Single Variable

$$\hat{y} = \beta_0 + \beta_1 x$$

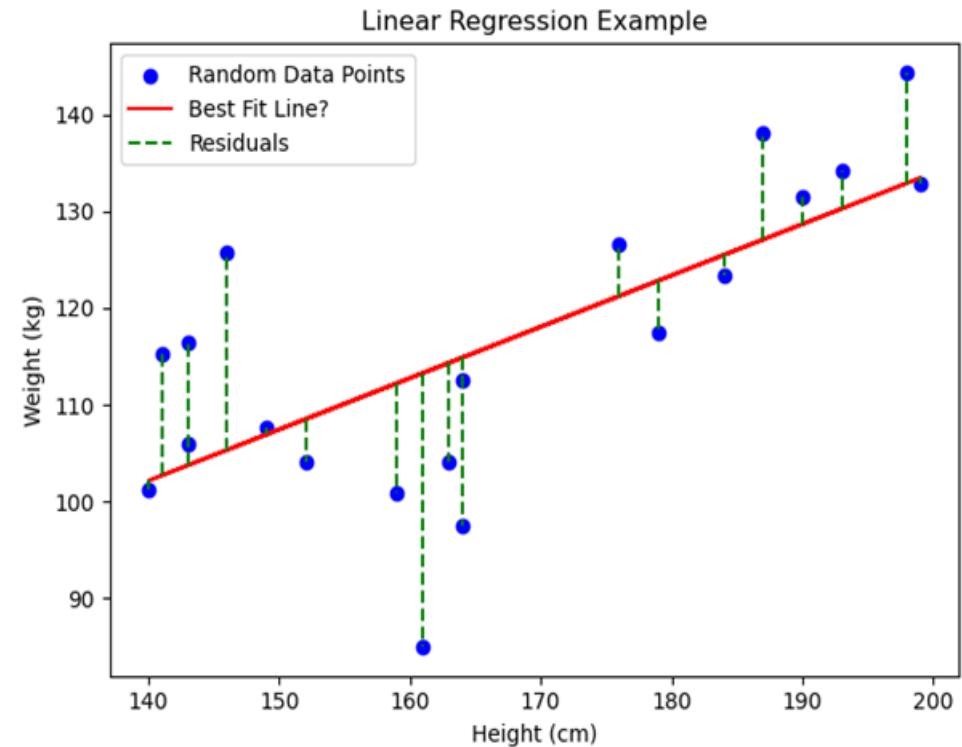
## Linear Regression: Multiple Variables

$$\boxed{\hat{y}} = \beta_0 + \beta_1 \boxed{x_1} + \dots + \beta_p \boxed{x_p}$$

# Evaluation Metrics for Regression Models



- Difference between the actual value and the model's estimate a **residual or error**.
- Evaluation metrics are measurements that take our collection of residuals and condense them into a *single* value that represents the predictive ability of our model.
  - Mean Absolute Error (MAE)
  - Mean Square Error (MSE)
  - Mean Absolute Percentage Error (MAPE)
  - Mean Percentage Error (MPE)



# Mean Absolute Error

## ■ Formulation:

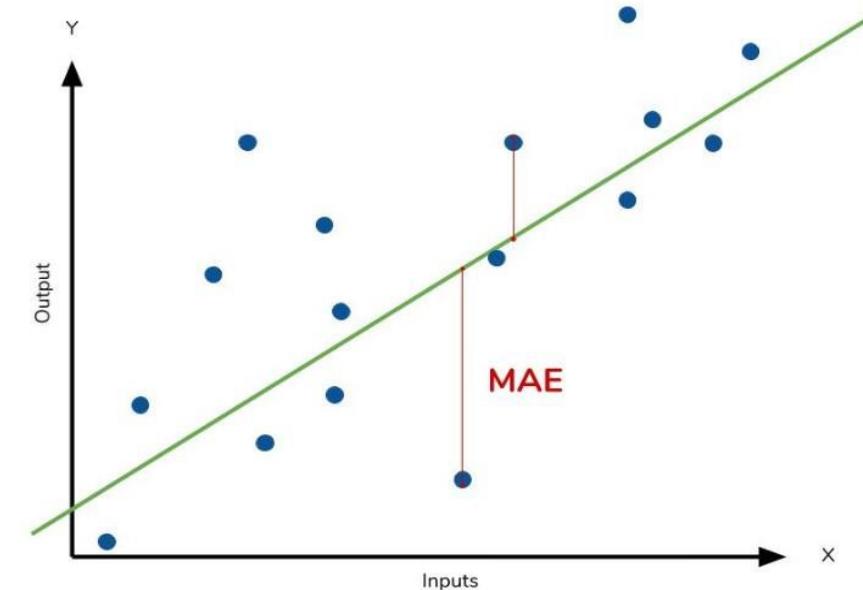
$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

### Pros

- Easy to understand and interpret
- Not sensitive to outliers, as it treats all errors equally

### Cons

- Doesn't punish large errors as much as MSE, which may be a drawback if you want to heavily penalize outliers.



# Mean Squared Error



## ■ Formulation:

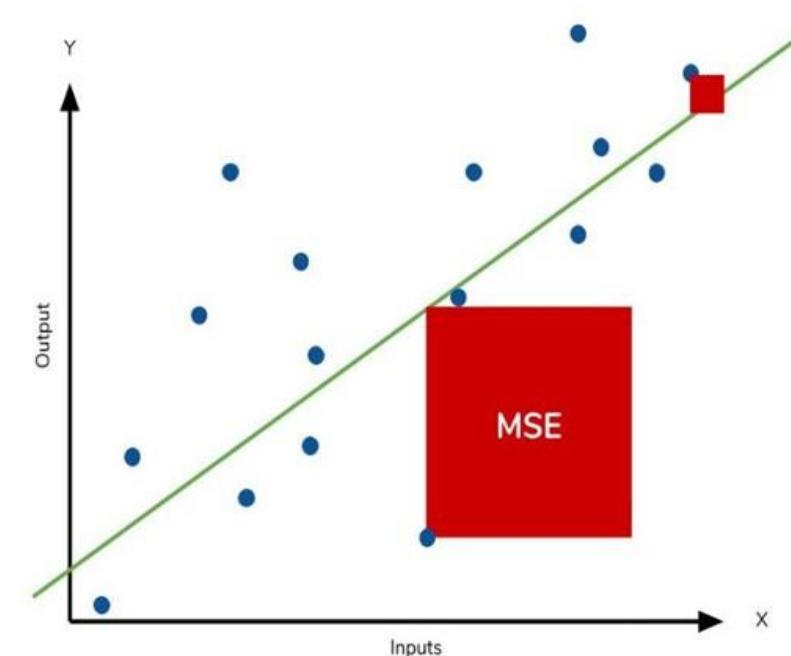
$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

### Pros

- It is differentiable, making it possible to reach closed-form solutions.

### Cons

- Sensitive to outliers and gives more weight to larger errors.



# Mean Absolute Percentage Error



## Formulation:

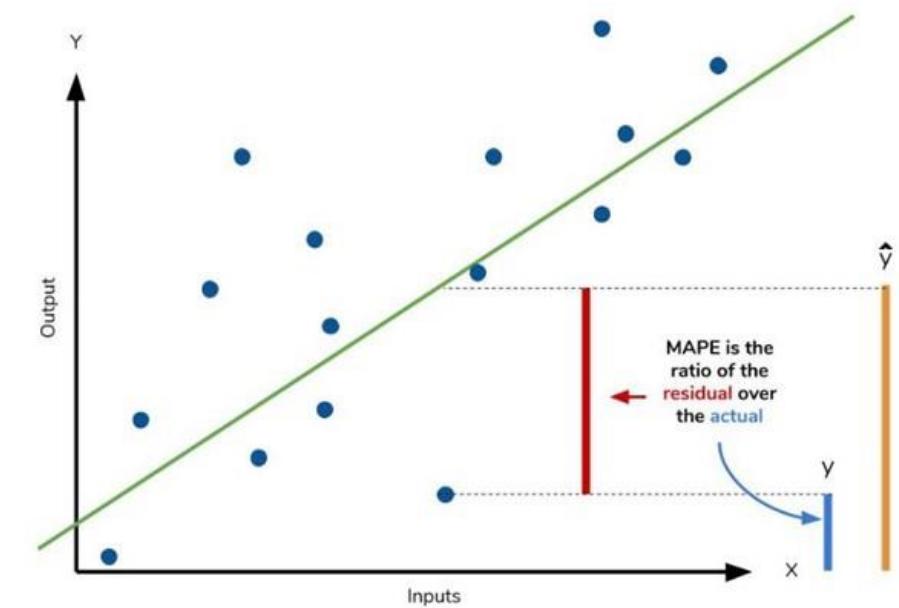
$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| * 100$$

### Pros

- Expresses errors as a percentage of the actual values, which can be more intuitive
- Gives an idea of the relative size of the error.

### Cons

- Problematic when actual values are close to zero



# Mean Percentage Error



## Formulation:

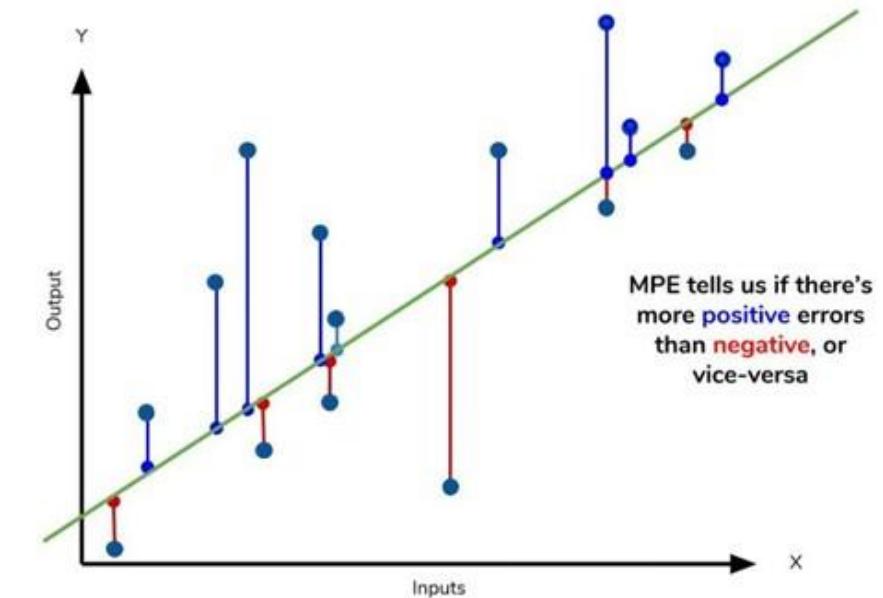
$$MPE = \frac{1}{N} \sum_{i=1}^N \frac{y_i - \hat{y}_i}{y_i} * 100$$

### Pros

- It gives a sense of the direction (overestimation or underestimation) of the errors.

### Cons

- Problematic when actual values are close to zero



# Summary



| Acronym | Full Name                      | Residual Operation? | Robust To Outliers? |
|---------|--------------------------------|---------------------|---------------------|
| MAE     | Mean Absolute Error            | Absolute Value      | Yes                 |
| MSE     | Mean Squared Error             | Square              | No                  |
| RMSE    | Root Mean Squared Error        | Square              | No                  |
| MAPE    | Mean Absolute Percentage Error | Absolute Value      | Yes                 |
| MPE     | Mean Percentage Error          | N/A                 | Yes                 |

# References



- <https://learning.oreilly.com/library/view/practical-statistics-for/9781491952955/cho6.html>
- <https://www.mathsisfun.com/data/standard-deviation.html>

# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Classification

---

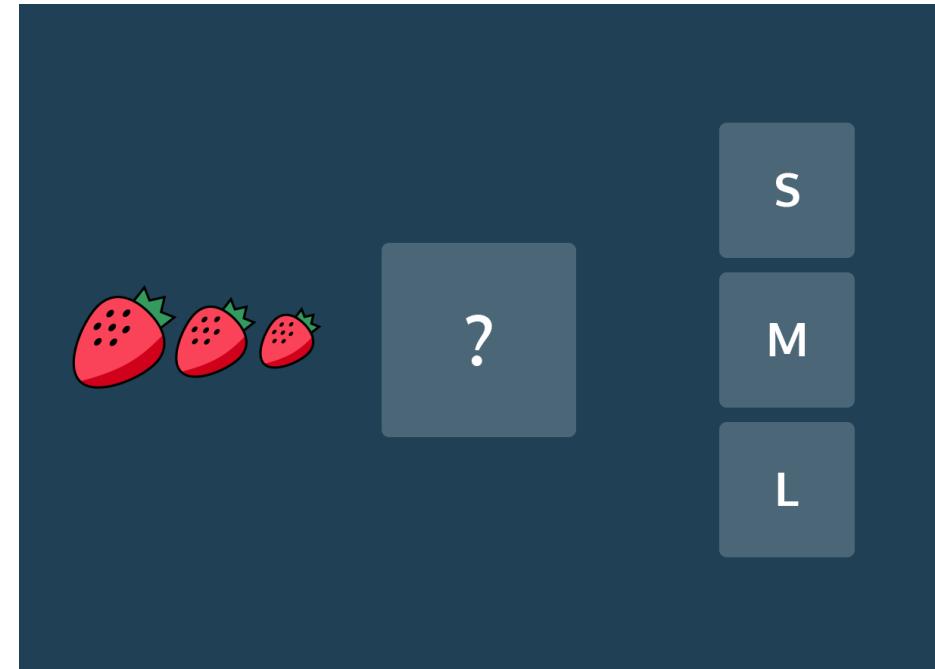
- Introduce Classification
- Classification algorithms: Logistic Regression
- Gradient Descent
- Linear vs Non-Linear methods
- Evaluation methods for classification methods



# Classification Algorithms



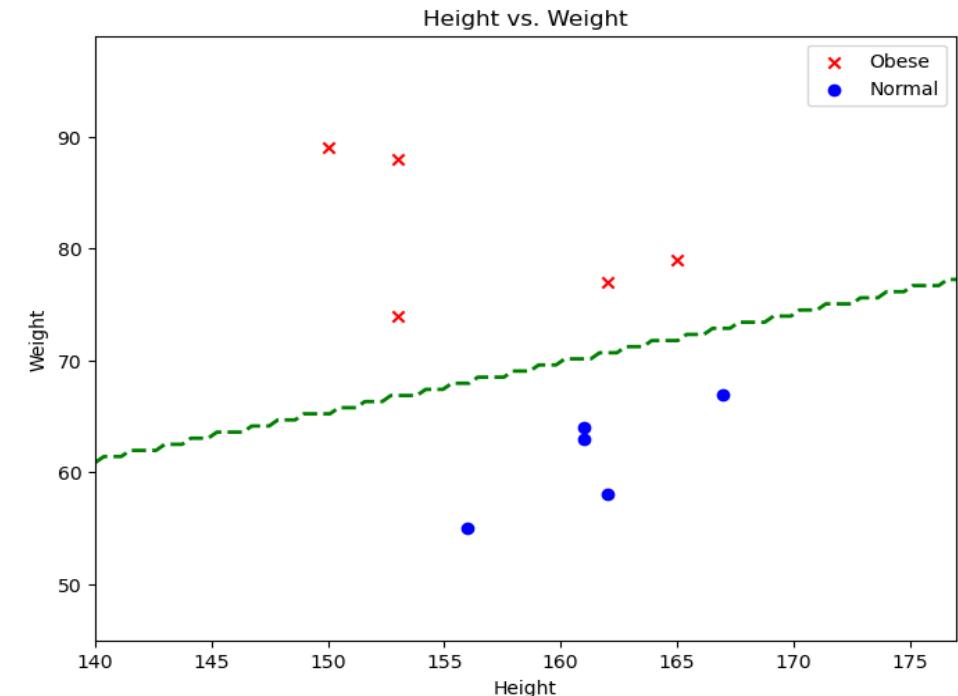
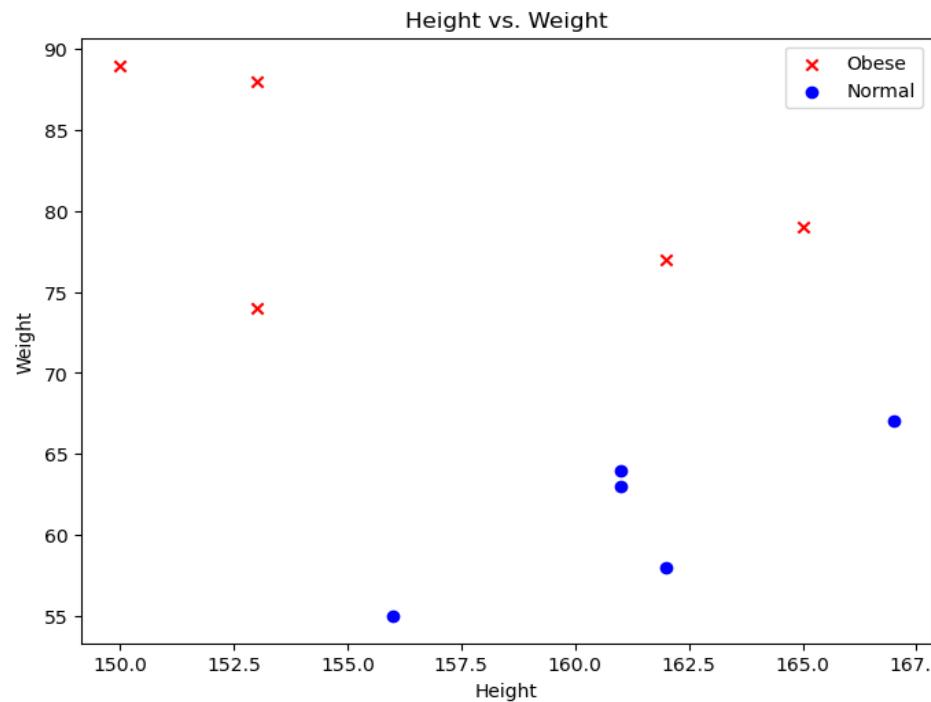
- Classification : Predicting discrete classes /categories
  - Binary Classification(between two)
    - Example : True/False, Object Identification- Cat /Not Cat
  - Multiclass classification(more than two)
    - Example : Animal Species
  - Multilabel classification(more than two)
    - Example : Colors in Image



# Decision Boundary Definition



- Assume we have 10 different people with different weight and height values. 5 of those people are labeled as “Obese” and 5 are labeled as “Normal”. The plot of those people is shown below. Is it possible to **separate** the two classes with a single line?



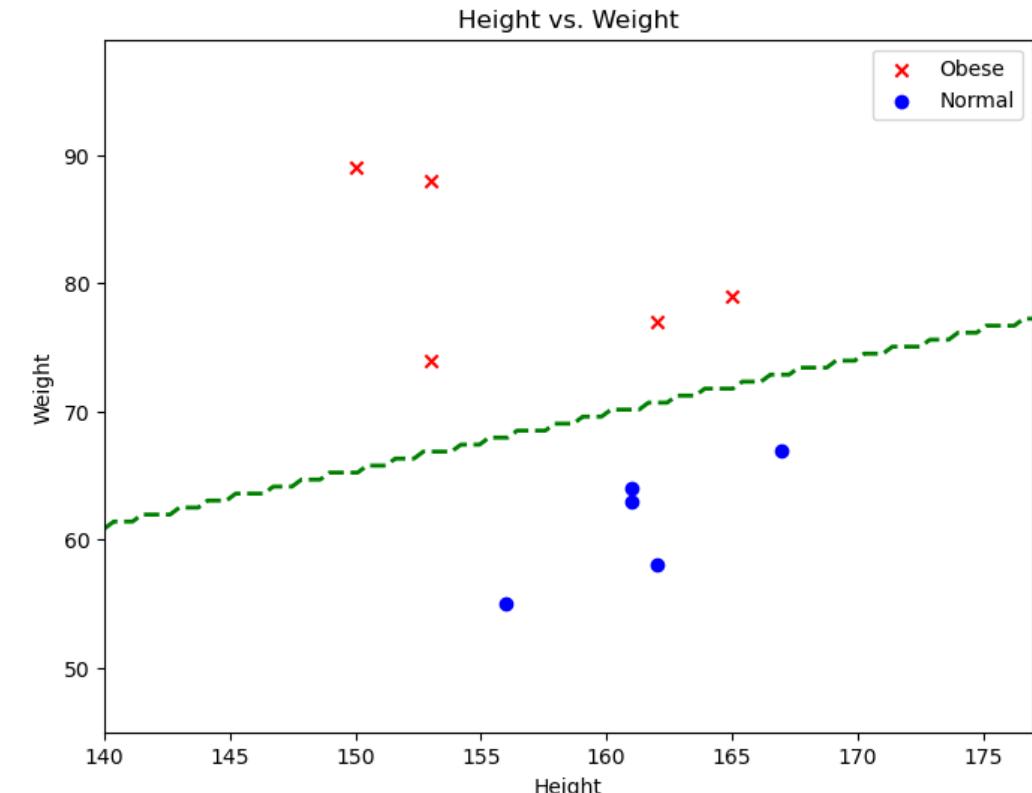
# Decision Boundary Definition



- The decision boundary in a binary classification problem is a hypersurface that separates the feature space into regions corresponding to different classes.
- The decision boundary only exists as a line or hyperplane if the classes are ***linearly separable***
- The equation of the linear decision boundary is as follows:

$$\hat{y} = \vec{w} \vec{x} + b$$

- $\vec{x}$  represents ***all*** input features.
  - $\hat{y}$  represents estimated class a point belongs to.
  - $y$  represents the label of the point  $\{0,1\}$
- 
- $\vec{w}, b$  are learnable parameters to be estimated

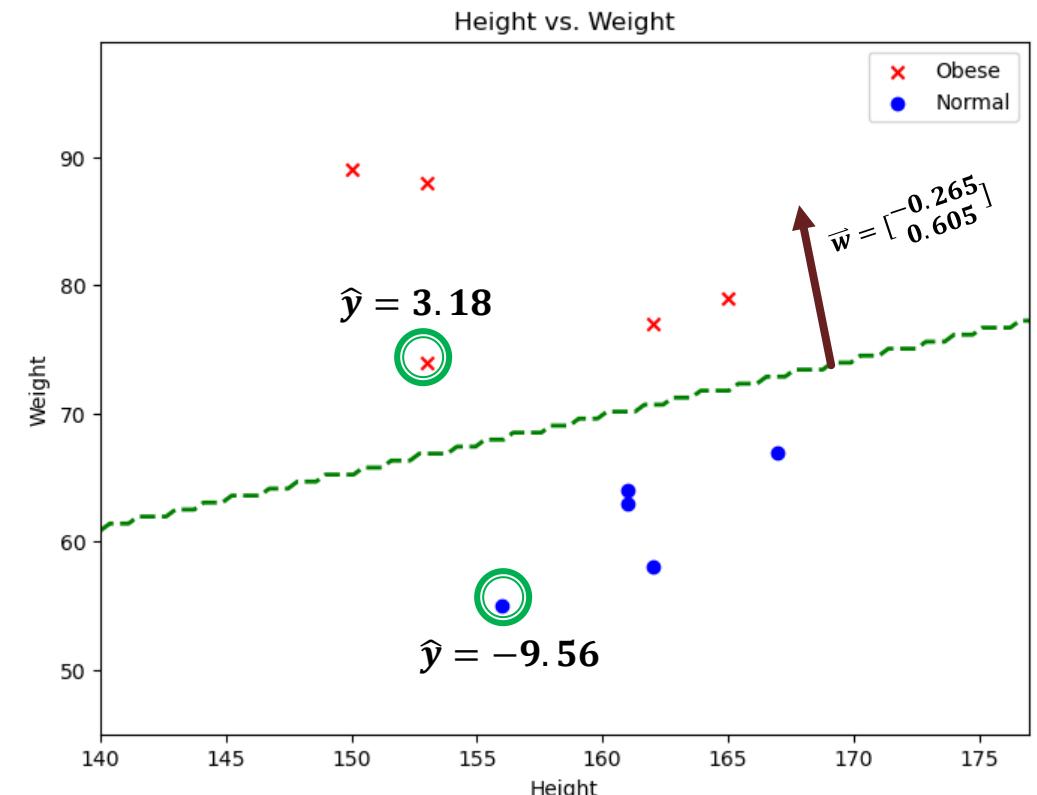


$$\hat{y} = \vec{w} \vec{x} + b$$

# Decision Boundary Characteristics



- Any point lying above the decision boundary  $\rightarrow \hat{y}$  is positive
- Any point lying below the decision boundary  $\rightarrow \hat{y}$  is negative
- Any point lying on the decision boundary  $\rightarrow \hat{y}$  is zero
- $\vec{w}$  is always perpendicular on the decision boundary.
- In any classification problem, our target is to find the decision boundary by estimating  $\vec{w}, b$ .
- Each classification algorithm has different ways (i.e. threshold) to classify the classes by matching  $\hat{y}$  to  $y$ .



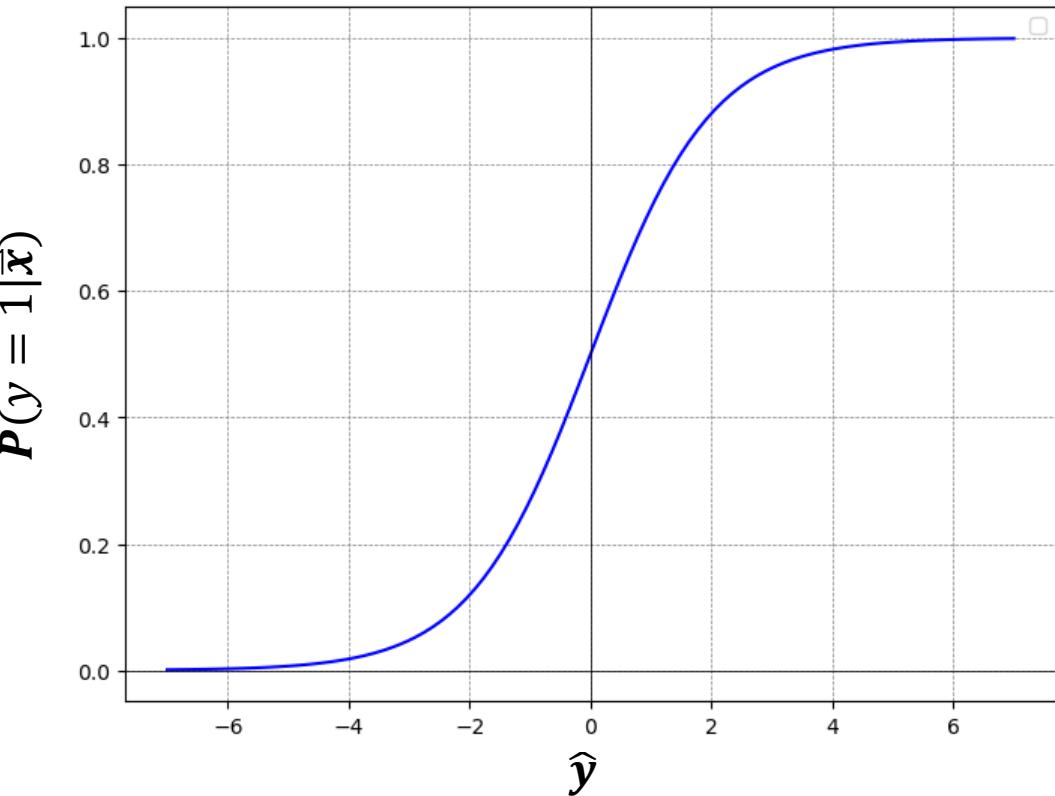
$$\hat{y} = w_1 * \text{height} + w_2 * \text{weight} + b$$
$$\hat{y} = -0.265 * \text{height} + 0.6054 * \text{weight}$$

# Logistic Regression



- Is there a way to use the value of  $\hat{y}$  to get the **probability** that a point belong to the positive class ( $y = 1$ )?
  - We need to find a function that can transform the positive and negative values of  $\hat{y}$  to a probability in range 0 – 1.

*Sigmoid Function*



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

# Logistic Regression



- Logistic regression is a classification algorithm used to predict the probability that a point belongs to a certain class  $P(y = 1|\vec{x})$
- $$P(y = 1|\vec{x}) = \frac{1}{1+e^{-\hat{y}}} = \frac{1}{1+e^{-(\vec{w}^T \vec{x} + b)}}$$
- In binary classification, each point  $\vec{x}$  has a label  $y$  that could be equal to 0 or 1.
- We want to represent the probability of observing the **correct** label of the point denoted as  $P(y|\vec{x})$ .

# Logistic Regression



- We want to represent the probability of observing the **correct** label of the point denoted as  $P(y|\vec{x})$ .

$$P(y|\vec{x}) = \begin{cases} P(y = 1|\vec{x}) \text{ if } \vec{x} \text{ belongs to +ve} \\ 1 - P(y = 1|\vec{x}) \text{ if } \vec{x} \text{ belongs to -ve} \end{cases}$$

- Is there a way to represent it in one equation only to handle both cases?

$$P(y|\vec{x}) = P(y = 1|\vec{x})^y * [1 - P(y = 1|\vec{x})]^{1-y}$$

# Logistic Regression



- In a dataset with  $N$  number of datapoints, what is the probability of observing the correct label  $y$  for ***all the points*** in the dataset?

$$P(y|\vec{x}) = \prod_{i=1}^N P(y_i = 1|\vec{x}_i)^{y_i} * [1 - P(y_i = 1|\vec{x}_i)]^{1-y_i}$$

- The target of the logistic regression model is to ***maximize*** the probability of observing the correct label  $y$  for ***all the points*** in the dataset → maximize  $[P(y|\vec{x})]$

# Logistic Regression



- ML algorithms favor ***minimizing*** the target functions and including **summations** rather than multiplications to help in the differentiation.

- Solution:
    - Apply ***log*** on the target function (replaces multiplication with summation)
    - Add ***-ve sign*** to the target function to be minimized rather than maximized.

$$P(y|\vec{x}) = \prod_{i=1}^N P(y_i = 1|\vec{x}_i)^{y_i} * [1 - P(y_i = 1|\vec{x}_i)]^{1-y_i}$$

$$L(\vec{w}, b) = - \sum_{i=1}^N \log(P(y_i = 1|\vec{x}_i))y_i + (1 - y_i)\log[1 - P(y_i = 1|\vec{x}_i)]$$

Cost function to be minimized

# Logistic Regression: Loss Minimization



$$L(\vec{w}, \mathbf{b}) = - \sum_{i=1}^N \log(P(y_i = 1 | \vec{x}_i))y_i + (1 - y_i) \log[1 - P(y_i = 1 | \vec{x}_i)]$$

$$P(y = 1 | \vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + \mathbf{b})}}$$

- Get the values of  $\vec{w}, \mathbf{b}$  at which  $L(\vec{w}, \mathbf{b})$  is minimum  $\rightarrow$  Get the values of  $\vec{w}, \mathbf{b}$  at which  $\frac{d(L)}{d\vec{w}} = 0$ , and  $\frac{d(L)}{d\mathbf{b}} = 0$ .
- For logistic regression,  $\frac{d(L)}{d\vec{w}} = 0$  and  $\frac{d(L)}{d\mathbf{b}} = 0$  **do not have** closed form solutions that can be derived by making  $\vec{w}$  and  $\mathbf{b}$  the subjects of their equations.
- For that reason, we utilize iterative optimization approaches like the famous **Gradient Descent** algorithm.

# Logistic Regression: Loss Minimization



$$L(\vec{w}, \vec{b}) = - \sum_{i=1}^N \log(P(y_i = 1 | \vec{x}_i))y_i + (1 - y_i) \log[1 - P(y_i = 1 | \vec{x}_i)]$$

$$P(y = 1 | \vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + \vec{b})}}$$

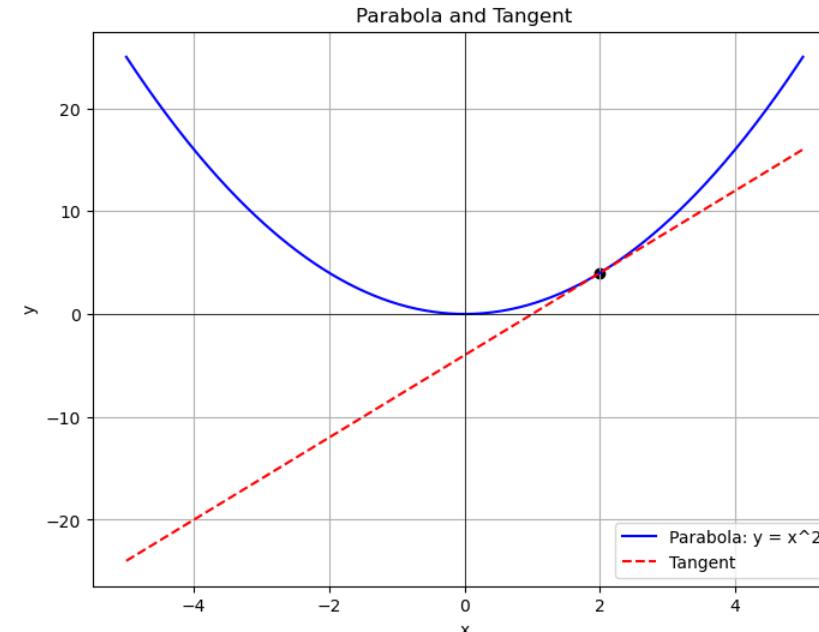
- We utilize iterative optimization approaches like the famous ***Gradient Descent*** algorithm to minimize  $L(\vec{w}, \vec{b})$ .

$$\frac{dL}{d\vec{w}} = \sum_{i=1}^N \left( y_i - \frac{e^{\vec{w} \cdot \vec{x}_i + \vec{b}}}{1 + e^{\vec{w} \cdot \vec{x}_i + \vec{b}}} \right) \vec{x}_i$$

# Gradient Descent



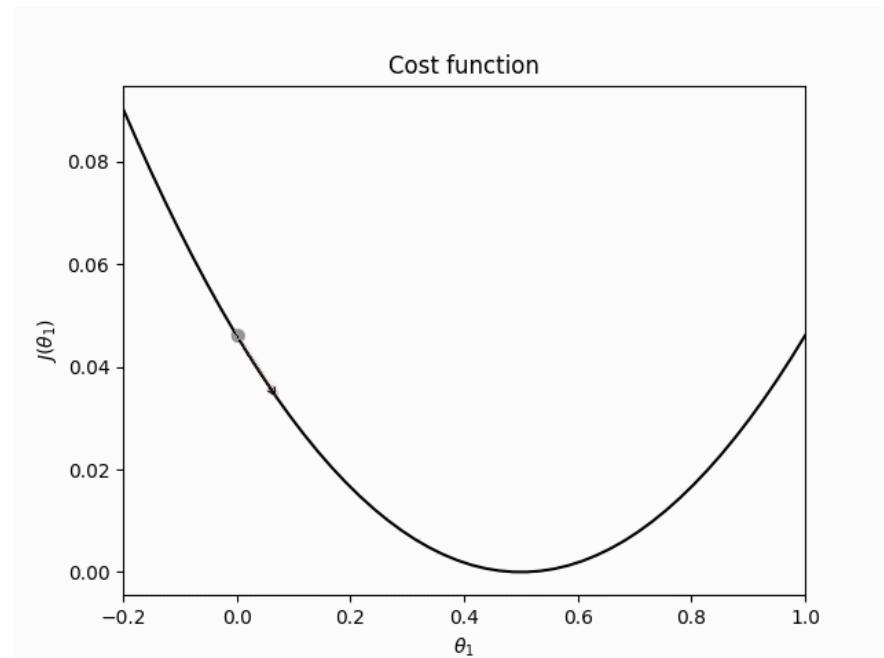
- Given a parabola in 2D, how to get the gradient at any point along the curve?
  - Get the gradient of the ***tangent*** of the curve at the point.
  - The ***tangent*** to a curve at a specific point is a straight line that just "touches" the curve at that point.



# Gradient Descent



- Now assume the x-axis represents the weights to be estimated  $\vec{w}$  and the y-axis represents a loss (cost) function  $L(\vec{w})$ .
- Our target is to find the value of  $\vec{w}$  at which  $L(\vec{w})$  is minimum.
- To achieve this iteratively starting from some random value of  $\vec{w}$ , we need to keep moving in the **negative** direction of the gradient until reaching the minimum value.



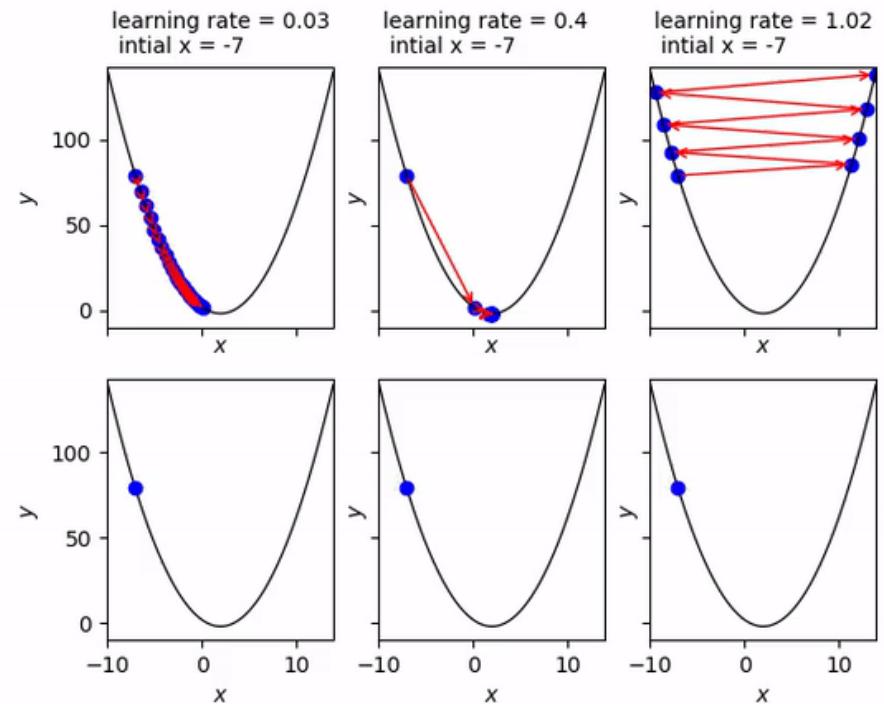
# Gradient Descent



- In other words, gradient descent keeps updating the value of  $\vec{w}$  in the **negative** direction of the gradient by a certain **step size** until reaching the minimum value such that:

$$\vec{w}_{t+1} = \vec{w}_t - \delta \frac{dL}{d\vec{w}}$$

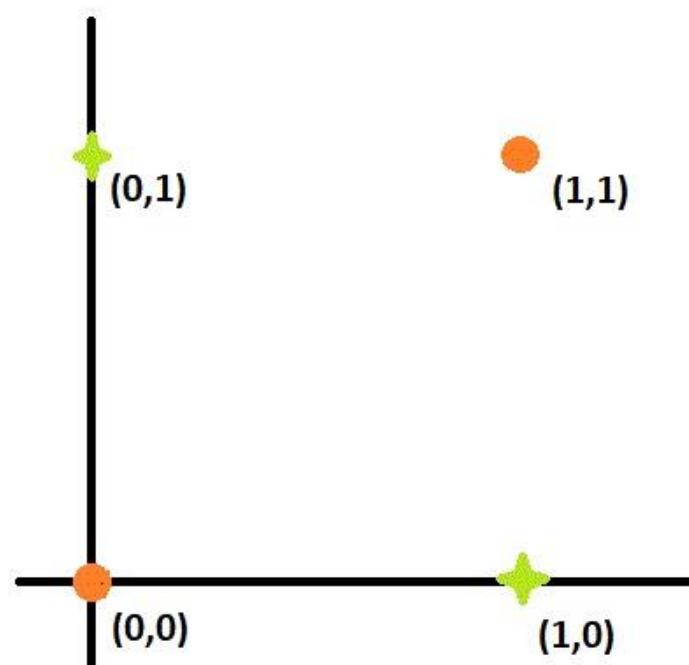
- The step size ( $\delta$ ) is a manually defined hyperparameter called the learning rate and it controls the magnitude of the change done for  $\vec{w}$  in each step.
  - If we set  $\delta$  too high  $\rightarrow$  Might not converge to minimum
  - If we set  $\delta$  too low  $\rightarrow$  Might take a very long time to converge to minimum



# Linearly vs Non-Linearly Separable



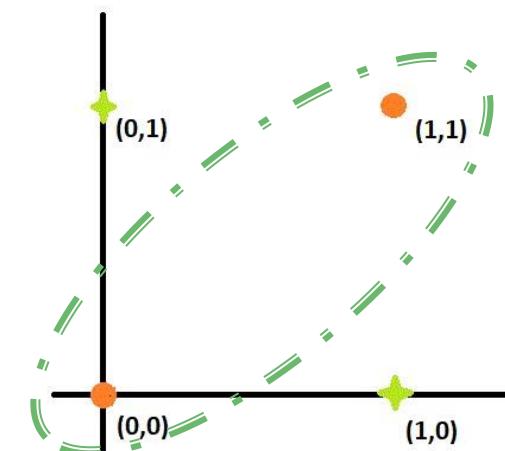
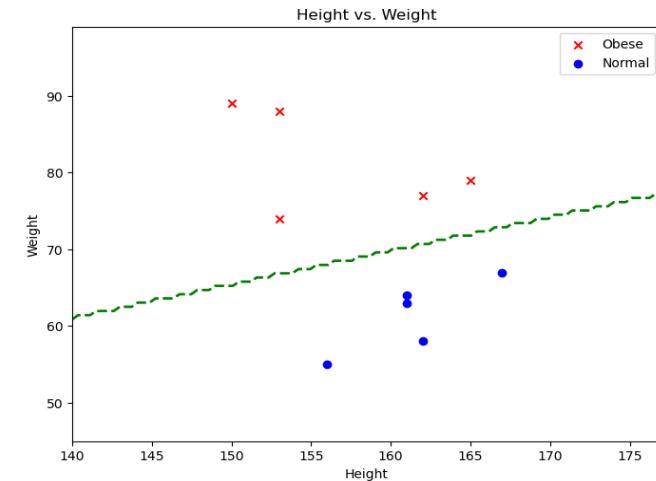
- Is it possible to have a linear decision boundary correctly separating the following classes?



# Linearly vs Non-Linearly Separable



- In a **linearly separable problem**, it is possible to draw a straight line (or hyperplane in higher dimensions) that completely separates the classes or categories of data points.
  - Linearly separable problems can be effectively solved using linear models such as logistic regression.
- In a **non-linearly separable problem**, it is not possible to draw a straight line or hyperplane to separate the classes. The data points are intermingled in a way that a linear boundary cannot accurately classify them.
  - The decision boundary in a non-linearly separable problem is a complex curve or surface.
  - Non-linearly separable problems require more complex models that can learn and represent non-linear relationships.



# Evaluation Metrics for Classification



- The evaluation metrics for classification problems measure how accurate are the predicted classes compared to the ground truth classes.
- They give us a *single* value that represents the classification ability of our model.
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - Specificity
  - AUC

|           |          | Actual         |                |
|-----------|----------|----------------|----------------|
|           |          | Positive       | Negative       |
| Predicted | Positive | True Positive  | False Positive |
|           | Negative | False Negative | True Negative  |

# Confusion Matrix



- Assume we have a spam classification model that is already trained and evaluated on 100 data points. There are 50 points in the positive class (Spam) and 50 in the negative class (Not Spam). When you evaluated the model, the following output was found:
  - 40 **spam** points were predicted as **spam**
  - 10 **spam** points were predicted as **not spam**
  - 35 **not spam** points were predicted as **not spam**
  - 15 **not spam** points were predicted as **spam**.

|            |          | Actual   |          | Prediction |
|------------|----------|----------|----------|------------|
|            |          | Positive | Negative |            |
| Prediction | Positive | 40       | 15       | 55         |
|            | Negative | 10       | 35       | 45         |
|            |          | 50       | 50       | 100        |

|            |          | Actual              |                     | Prediction |
|------------|----------|---------------------|---------------------|------------|
|            |          | Positive            | Negative            |            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |            |
|            | Negative | False Negative (FN) | True Negative (TN)  |            |

# Confusion Matrix



- A confusion matrix is a tabular representation used in machine learning and statistics to assess the performance of a classification model. It provides a detailed breakdown of the model's predictions compared to the actual outcomes in a classification problem.

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Accuracy



## ■ Definition:

- Accuracy is the measure of how correctly the model predicts all classes, both positive and negative. It calculates the ratio of correctly predicted instances to the total instances.

## ■ Formulation:

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

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Precision



## ■ Definition:

- Precision measures the proportion of true positive predictions out of all positive predictions made by the model.
- In other words, from all the positives predicted by the model, how many were correct?
- Important in search algorithms when the retrieved links need to be mostly relevant.

## ■ Formulation:

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

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Recall (Sensitivity / True Positive Rate)



## ■ Definition:

- Recall measures the proportion of true positive predictions out of all actual positives in the dataset.
- In other words, from all the positives in the dataset, how many did the model predict correctly?
- Important in medical screenings, where missing a true positive in this context can delay treatment.

## ■ Formulation:

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

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Specificity (True Negative Rate)



## ■ Definition:

- Specificity measures the proportion of true negatives out of all actual negatives.
- In other words, from all the negatives in the dataset, how many did the model predict correctly?
- Important in spam filtering, where marking a legitimate email as spam can lead to important messages being overlooked or missed.

## ■ Formulation:

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

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Example



- Assume we have a dataset of 95 Not Spam (-ve) points and 5 Spam (+ve) points. You trained a model on this dataset and upon evaluation it predicted 99 data points to be Not Spam and 1 Spam point to be Spam.

- $Accuracy = \frac{96}{100} = 96\%$

- $Precision = \frac{1}{1} = 100\%$

- $Recall = \frac{1}{5} = 20\%$

- Is accuracy a good representative of the performance of the model?

- In imbalanced datasets, accuracy is not a reliable measure of performance. We need another measure mixing both precision and recall reliably in one metric.

|            |          | Actual   |          | Total |
|------------|----------|----------|----------|-------|
|            |          | Positive | Negative |       |
| Prediction | Positive | 1        | 0        | 1     |
|            | Negative | 4        | 95       | 99    |
|            |          | 5        | 95       | 100   |

# F1-Score



## ■ Definition:

- F1-Score is a metric that combines precision and recall into a single value. It is the harmonic mean of precision and recall, providing a balanced measure of a model's performance.
- Particularly useful in situations where both false positives and false negatives are critical or we have imbalanced datasets.

## ■ Formulation:

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

|            |          | Actual              |                     |
|------------|----------|---------------------|---------------------|
|            |          | Positive            | Negative            |
| Prediction | Positive | True Positive (TP)  | False Positive (FP) |
|            | Negative | False Negative (FN) | True Negative (TN)  |

# Example



- Assume we have a dataset of 95 Not Spam (-ve) points and 5 Spam (+ve) points. You trained a model on this dataset and upon evaluation it predicted 99 data points to be Not Spam and 1 Spam point to be Spam.

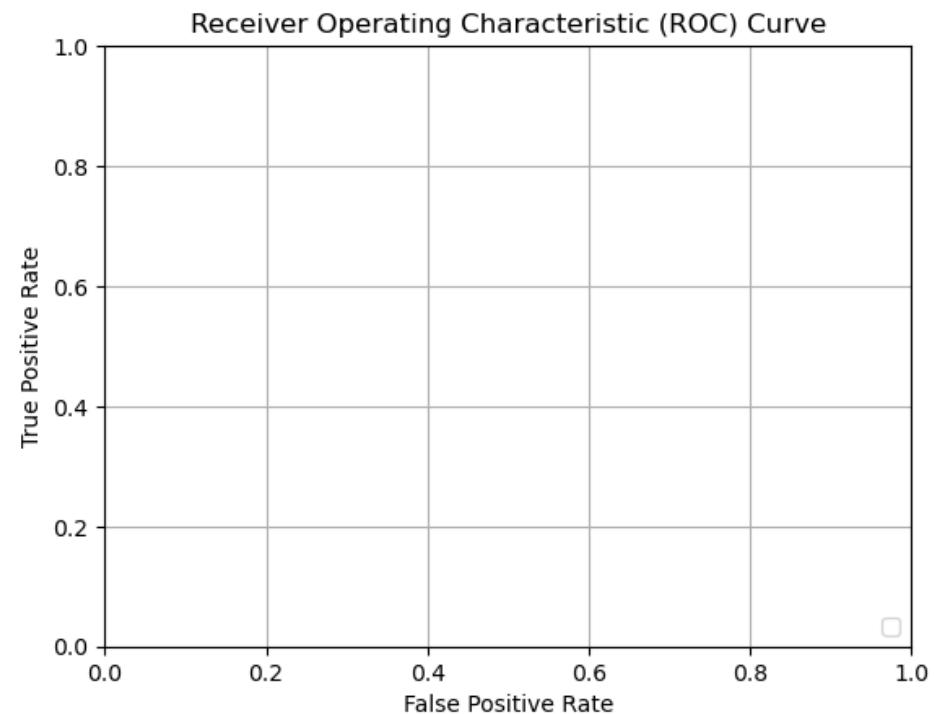
- $Accuracy = \frac{96}{100} = 96\%$
- $Precision = \frac{1}{1} = 100\%$
- $Recall = \frac{1}{5} = 20\%$
- $F1 - Score = \frac{2*20*100}{20+100} = 33.3\%$

|            |          | Actual   |          | Total |
|------------|----------|----------|----------|-------|
|            |          | Positive | Negative |       |
| Prediction | Positive | 1        | 0        | 1     |
|            | Negative | 4        | 95       | 99    |
|            |          | 5        | 95       | 100   |

# ROC Curve



- Assume we have a model that can tell if a person is sick or not. But, like any model, it's not perfect. Sometimes it says a healthy (-ve) person is sick (+ve), and sometimes it says a sick person (+ve) is healthy (-ve).
- Our model produces a probability from 0 to 1 that a person is sick.
- We can manually set a threshold on the output probability. If the probability is higher than the threshold, the person is labeled as sick.
- We want to figure out how good this model really is. We will plot a 2D graph for this purpose.
  - X-axis represents False Positive Rate (FPR) → How many of the healthy people do we label as sick.
  - Y-axis represents True Positive Rate (TPR) → How many of the sick people do we label as sick.

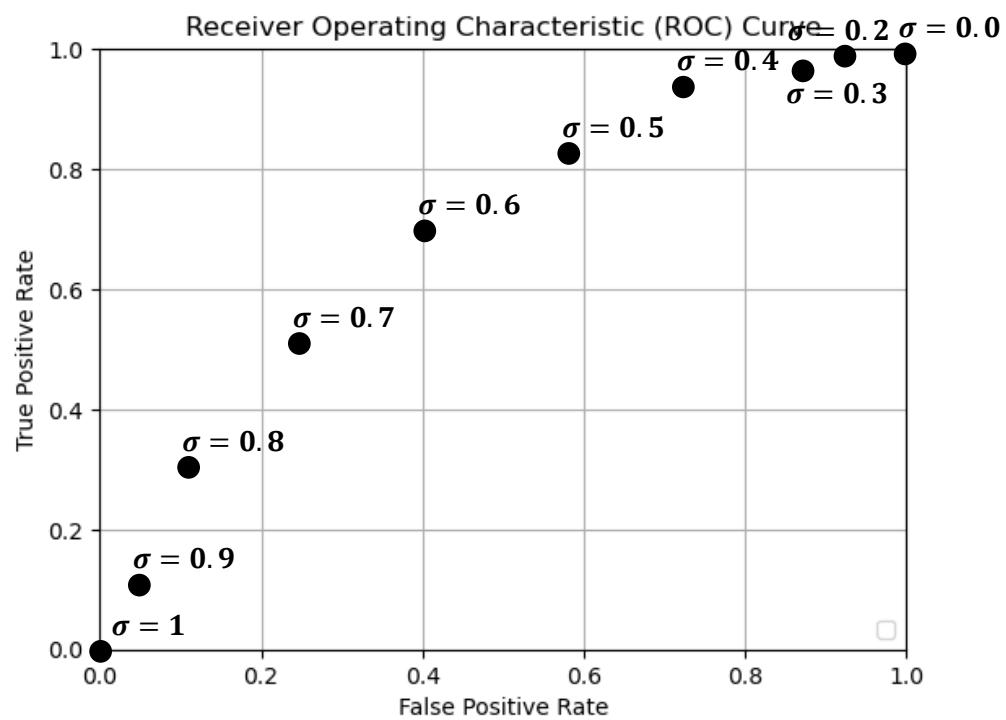


$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{TN + FP}$$

# ROC Curve



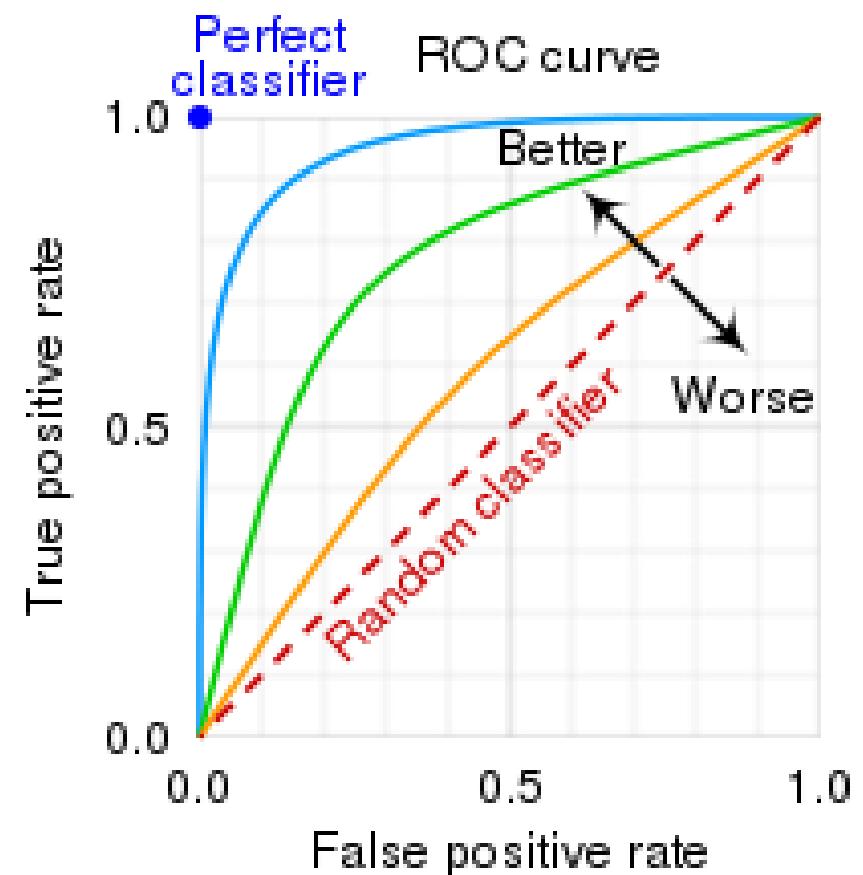
- We can manually set a threshold  $\sigma$  on the output probability. If the probability is higher than the threshold, the person is labeled as sick. Assuming we have a non-perfect classifier.
  - If we set  $\sigma = 1 \rightarrow TPR \approx 0, FPR \approx 0$
  - If we set  $\sigma = 0.9 \rightarrow TPR \approx 0.1, FPR \approx 0.05$
  - If we set  $\sigma = 0.8 \rightarrow TPR \approx 0.3, FPR \approx 0.1$
  - If we set  $\sigma = 0.7 \rightarrow TPR \approx 0.55, FPR \approx 0.25$
  - If we set  $\sigma = 0.6 \rightarrow TPR \approx 0.65, FPR \approx 0.40$
  - ....
  - If we set  $\sigma = 0.2 \rightarrow TPR \approx 0.97, FPR \approx 0.85$
  - If we set  $\sigma = 0.1 \rightarrow TPR \approx 0.99, FPR \approx 0.92$
  - If we set  $\sigma = 0.0 \rightarrow TPR \approx 1.0, FPR \approx 1.0$



# ROC Curve



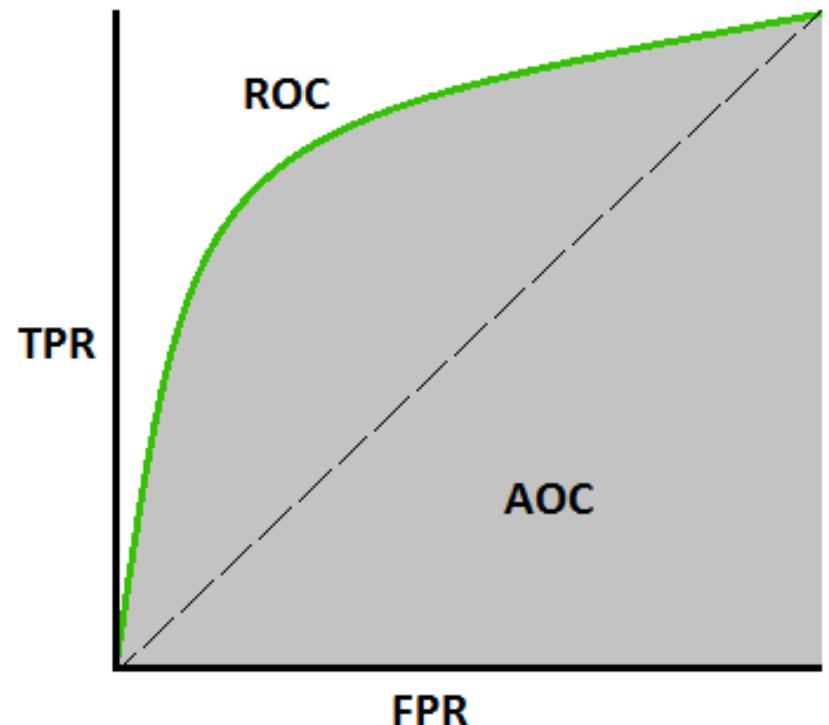
- A random classifier will always have a 50% chance to label a point in the positive class → Identity Line
- A better classifier will have higher true positives compared to false positives in the higher thresholds → A more steep curve
- A worse classifier will have lower true positives compared to false positives in the higher thresholds → A less steep curve
- A perfect classifier will have 100% true positives in all thresholds, and false positives increase from 0 to 100% as threshold increases.



# Area Under ROC Curve (AUC)



- Definition:
  - The AUC provides a single scalar value that summarizes the model's performance across various threshold settings.
  - Gives an understanding on how confident the model is in distinguishing between the classes.
  - A higher AUC means the model predicts the positive classes with high confidence (probability).
  - Useful when we have imbalanced datasets and care about the confidence of the model in classifying the positive class.
- Formulation:
  - *Compute the area under the ROC curve*



# Activity



- Given the available diabetes.csv dataset. Train a Logistic Regression model to predict whether a person has diabetes or not.
  - Train and evaluate on the whole dataset (no need to split)
  - Evaluate the model performance by printing the classification\_report
  - Plot the ROC Curve by predicting the probabilities using the predict\_proba function and extracting the positive class
  - Print the AUC

# References



- <https://learning.oreilly.com/library/view/practical-statistics-for/9781491952955/cho6.html>
- <https://www.mathsisfun.com/data/standard-deviation.html>

# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Classification

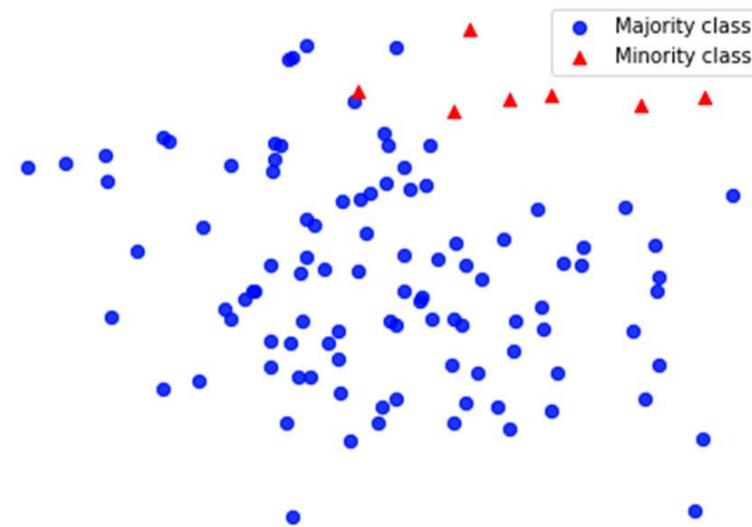
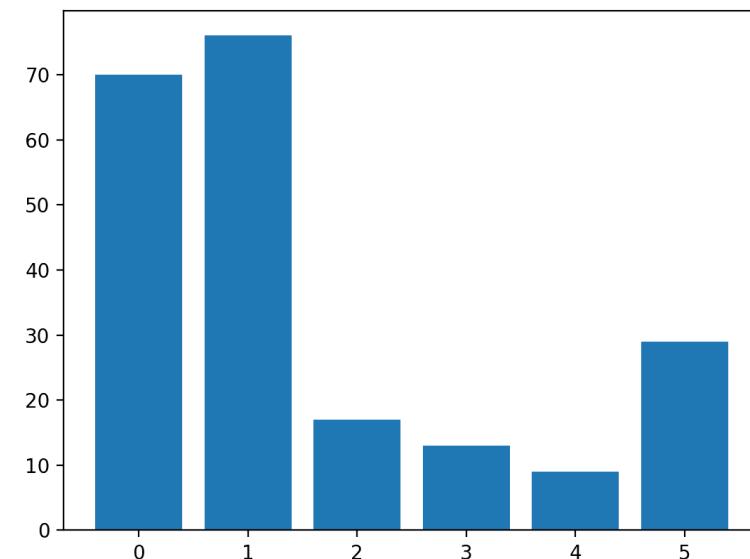
- Handling Class Imbalances
  - Support Vector Machines



# Handling Class Imbalances



- In classification problems, classes can be highly imbalanced where some classes have very high frequency compared to others.
- This can result in poor performance on the minority class.



# Class Weights



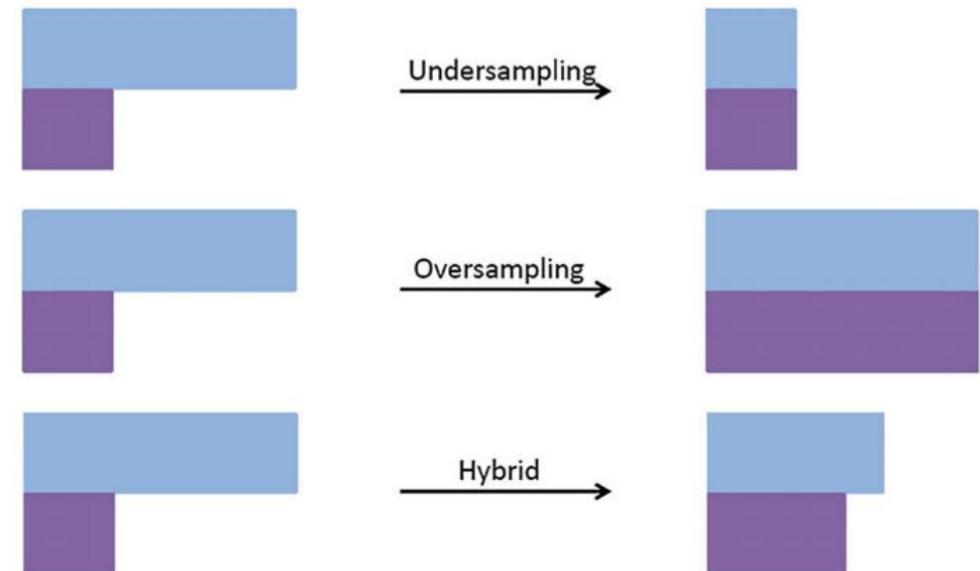
- One solution to handle class imbalances is ***Class Weights***
- Class weights basically assign higher weights to the samples from the minority class and lower weights to the samples from the majority class during the training process.
- This gives the algorithm more emphasis on correctly classifying the minority class, effectively balancing the impact of each class on the model's learning process.

|            |          | With Class Weighting |          |        |          |          |       |
|------------|----------|----------------------|----------|--------|----------|----------|-------|
|            |          | Sawtooth             | Sine     | Square | Triangle |          |       |
| True Class | Sawtooth | 21                   |          | 2      |          | 91.3%    | 8.7%  |
|            | Sine     | 1                    | 35       |        | 5        | 85.4%    | 14.6% |
|            | Square   |                      |          | 19     |          | 100.0%   |       |
|            | Triangle | 1                    | 6        |        | 6        | 46.2%    | 53.8% |
|            |          | Sawtooth             | Sine     | Square | Triangle |          |       |
|            |          | Predicted Class      |          |        |          |          |       |
|            |          | Sawtooth             | 20       |        | 3        | 87.0%    | 13.0% |
|            |          | Sine                 | 1        | 38     | 2        | 92.7%    | 7.3%  |
| True Class |          | Square               |          | 19     |          | 100.0%   |       |
|            |          | Triangle             | 1        | 9      |          | 23.1%    | 76.9% |
|            |          | Predicted Class      | Sawtooth | Sine   | Square   | Triangle |       |
|            |          |                      |          |        |          |          |       |

# Over/Under Sampling



- Oversampling and undersampling are approaches that aim to balance the representation of different classes in the training dataset, which can lead to better performance of the model, especially for the minority class.
- Oversampling:
  - Increase the number of instances of the minority class by generating synthetic samples or duplicating existing ones.
  - Examples: Random Oversampling / SMOTE / ADASYN
- Undersampling:
  - Reduce the number of instances of the majority class to achieve a better class balance.
  - Examples: Random/Tomek Links/ ENNs.



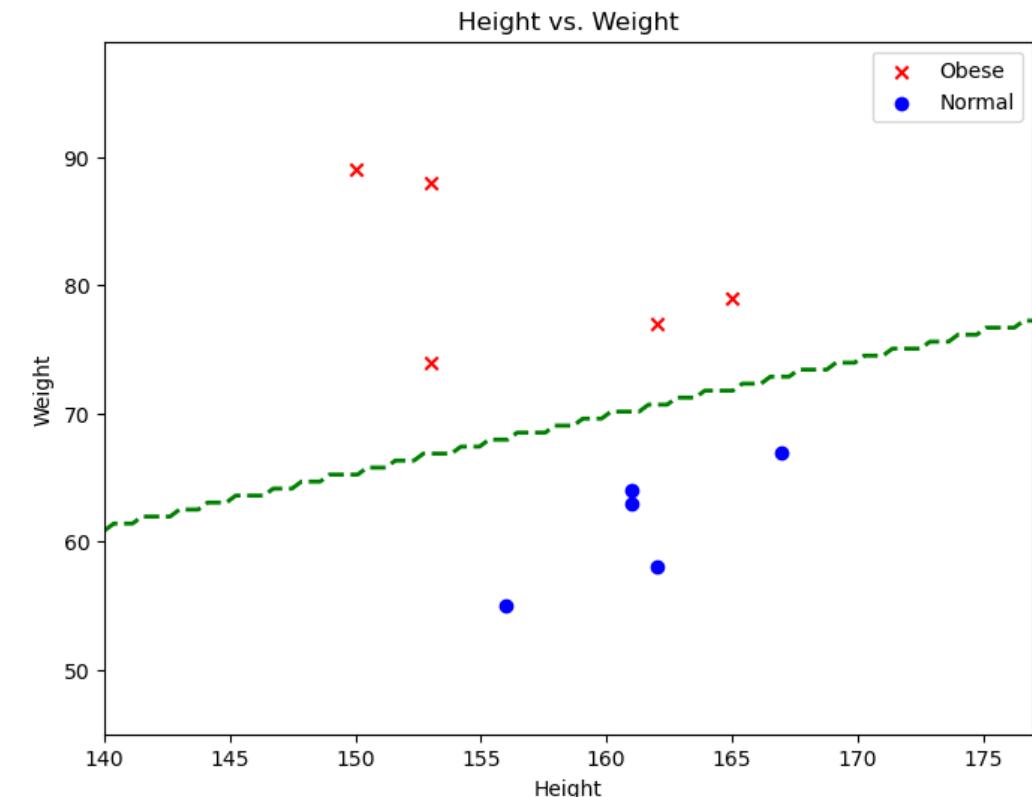
# Recall: Decision Boundary Definition



- The decision boundary in a binary classification problem is a hypersurface that separates the feature space into regions corresponding to different classes.
- The decision boundary only exists as a line or hyperplane if the classes are ***linearly separable***
- The equation of the linear decision boundary is as follows:

$$\hat{y} = \vec{w} \vec{x} + b$$

- $\vec{x}$  represents all input features.
- $\hat{y}$  represents estimated class a point belongs to.
- $y$  represents the label of the point  $\{0,1\}$
- $\vec{w}, b$  are learnable parameters to be estimated

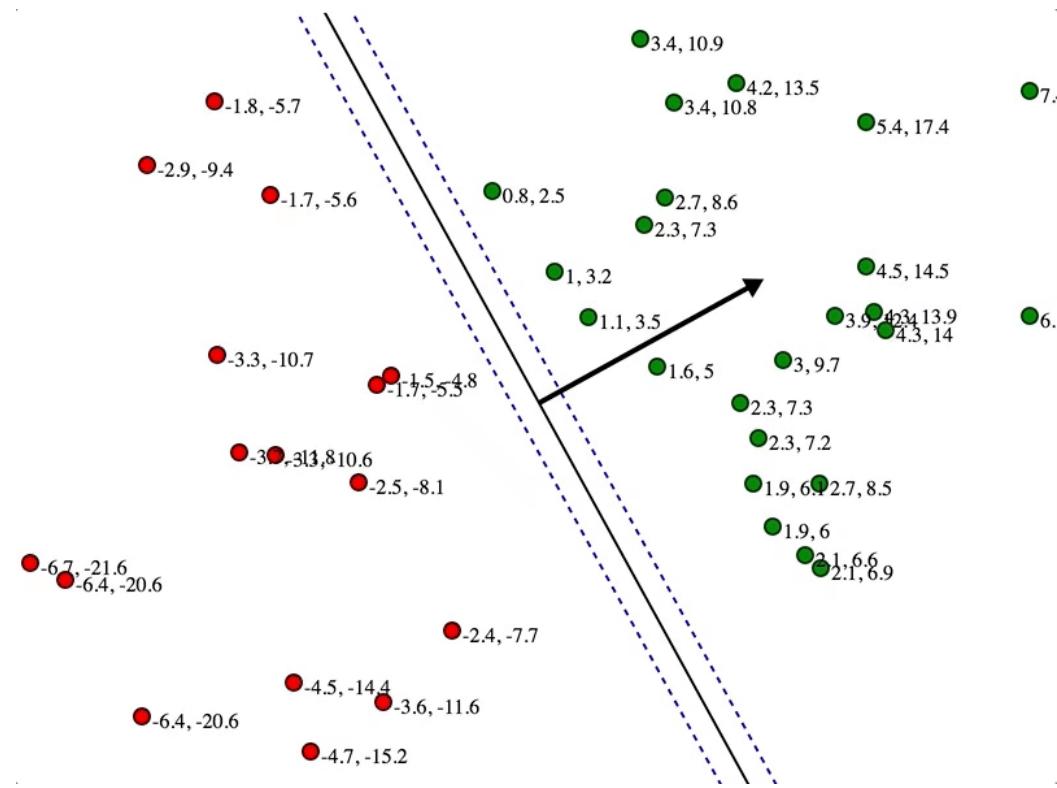


$$\hat{y} = \vec{w} \vec{x} + b$$

# Support Vector Machines (SVM)



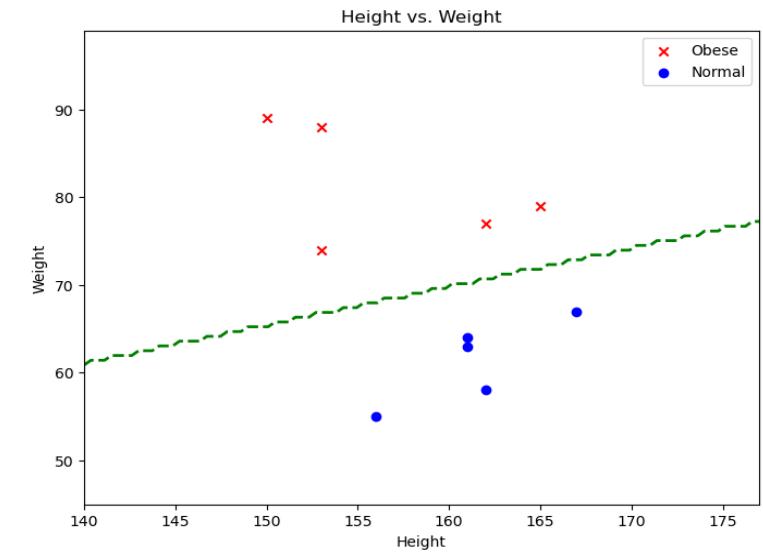
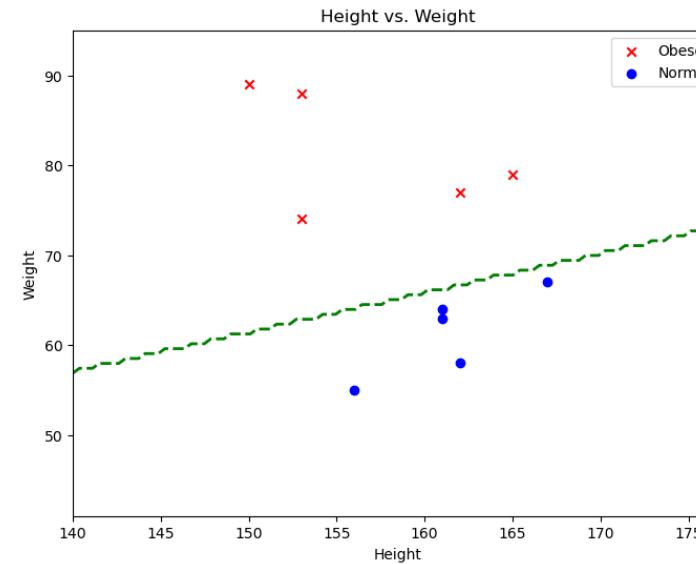
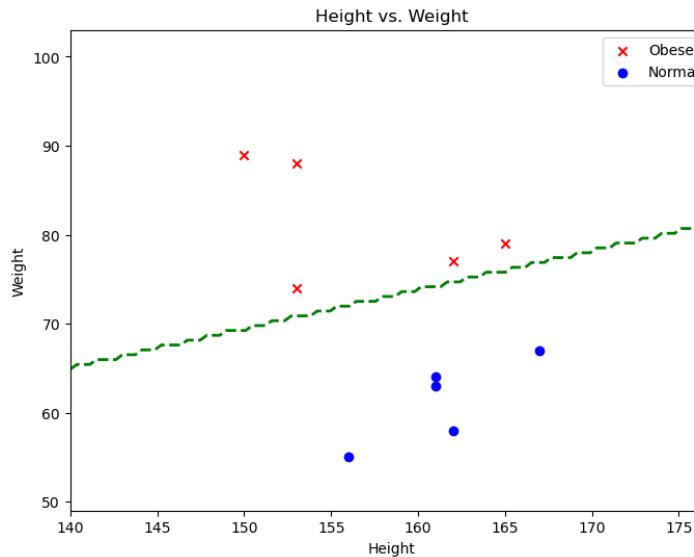
- Support Vector Machines (SVMs) are a powerful class of supervised machine learning algorithms used for both classification and regression tasks. They work by finding the optimal hyperplane that best separates different classes in a dataset.
- In order to reach the optimal hyperplane, several assumptions are made by the SVM algorithm. We will go through each assumption until reaching the final formulation of the SVM.



# Unbiased Decision Boundary



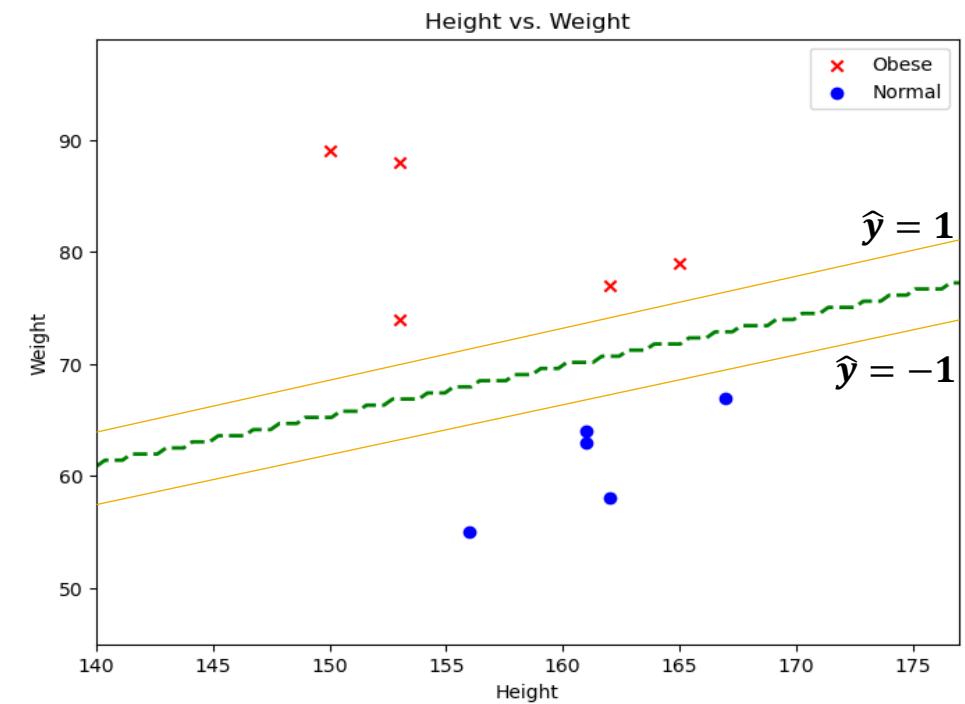
- Assume we have 10 different people with different weight and height values. 5 of those people are labeled as "Obese" and 5 are labeled as "Normal". The plot of those people is shown below. Which figure represents an unbiased decision boundary in the middle of the classes?



# Unbiased Decision Boundary



- Our first target is that we want an unbiased decision boundary lying directly in the middle of the two classes.
- To achieve that, we will first assume that all the points in each class have  $\hat{y}$  equal to at least  $\pm 1$ .  $\rightarrow$  **Ass. 1**
- We will also assume that each point in the classes have labels  $y \in \{1, -1\}$   $\rightarrow$  **Ass. 2**

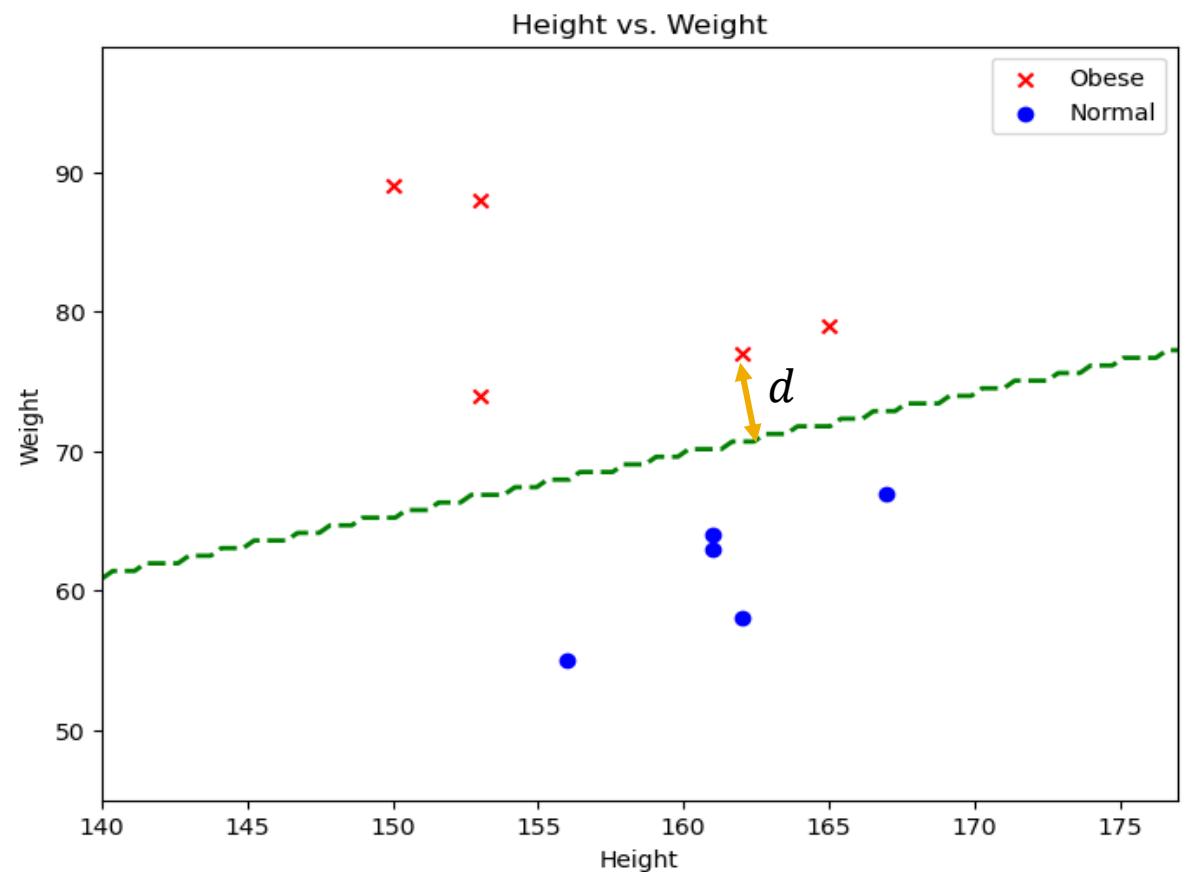


# Distance between point and boundary



- For any decision boundary  $(\vec{w} \vec{x} + b)$ , we represent the distance  $d$  between the boundary and a point  $\vec{x}$ , as follows:

$$d = \frac{(\vec{w} \vec{x} + b)}{\|\vec{w}\|} = \frac{\hat{y}}{\|\vec{w}\|}$$

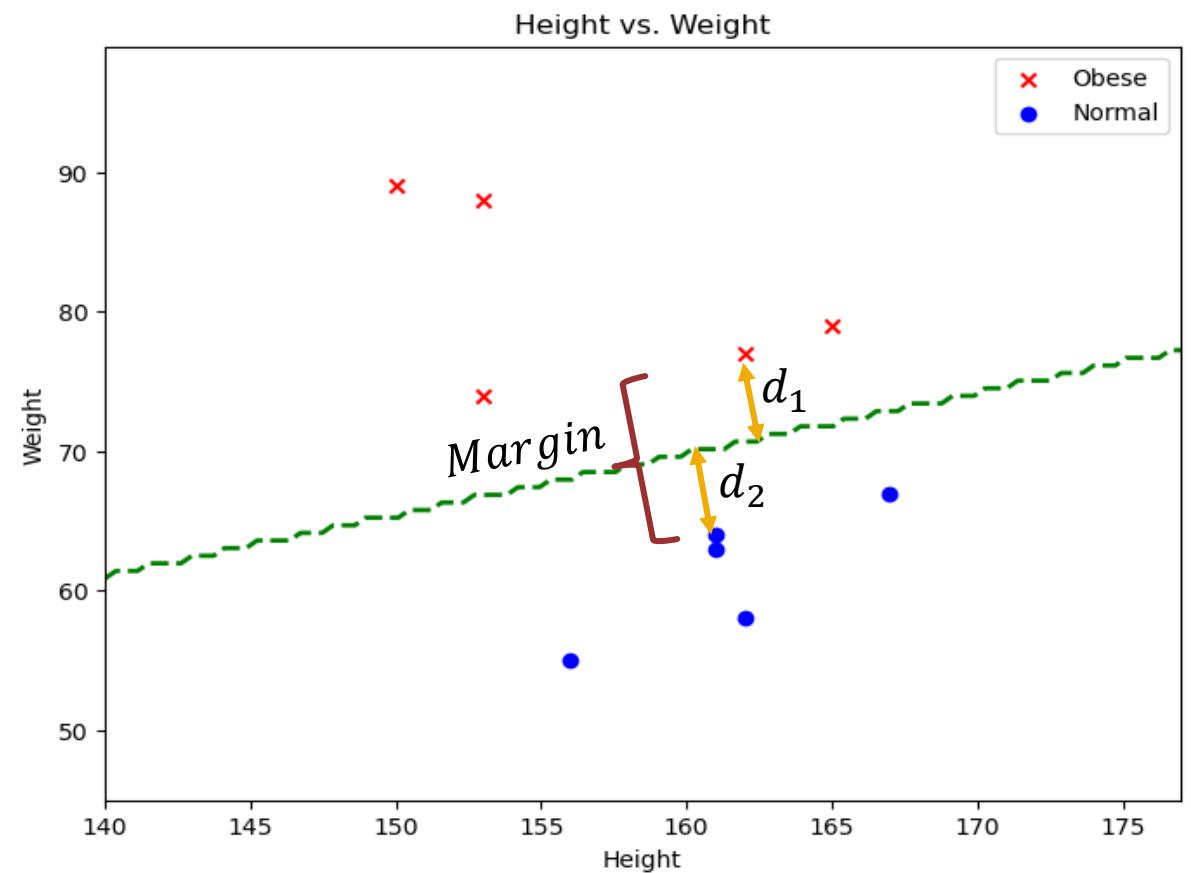


# Margin



- The margin represents the total distance between the decision boundary and the closest points in each class, where  $Margin = d_1 + d_2$ .

$$Margin = \left| \frac{\hat{y}_1}{\|\vec{w}\|} \right| + \left| \frac{\hat{y}_2}{\|\vec{w}\|} \right|$$



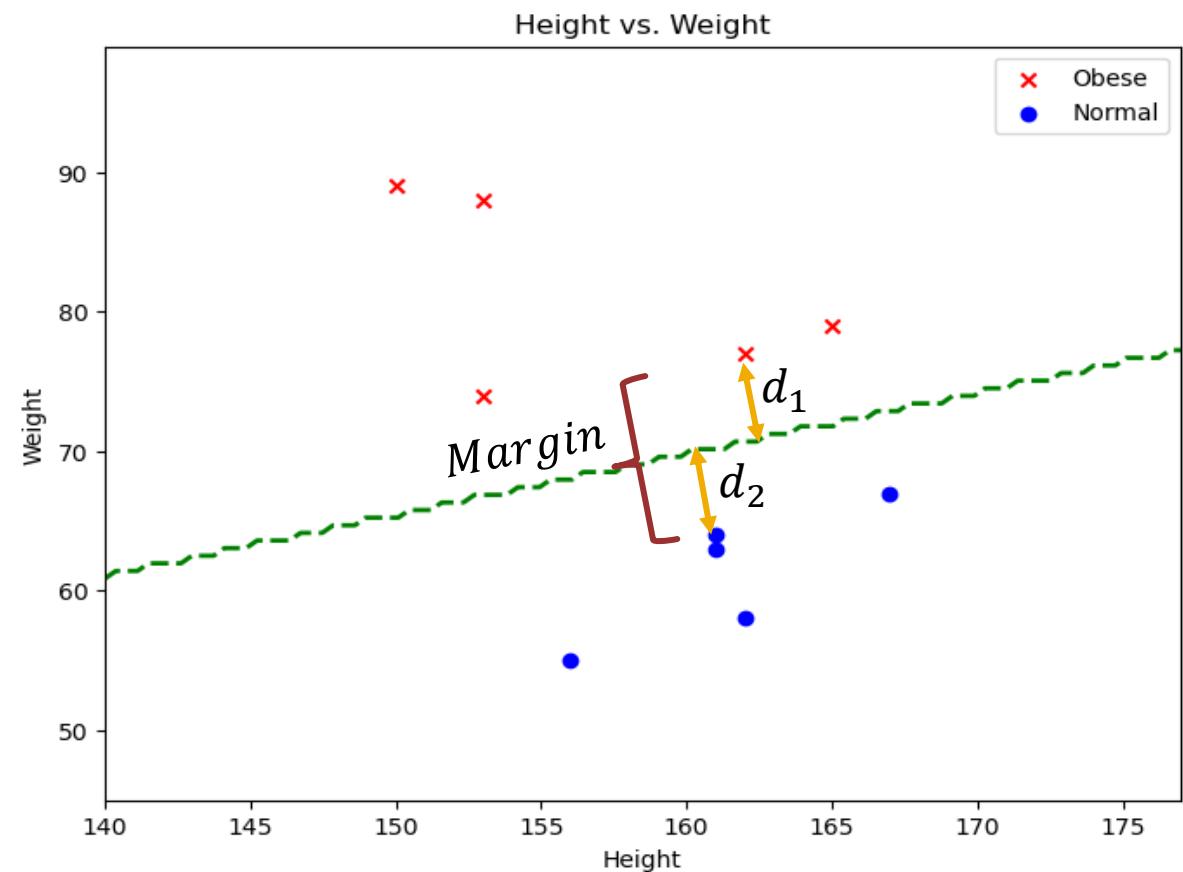
# Margin



- The margin represents the total distance between the decision boundary and the closest points in each class, where  $Margin = d_1 + d_2$ .

$$Margin = \left| \frac{\hat{y}_1}{\|\vec{w}\|} \right| + \left| \frac{\hat{y}_2}{\|\vec{w}\|} \right|$$

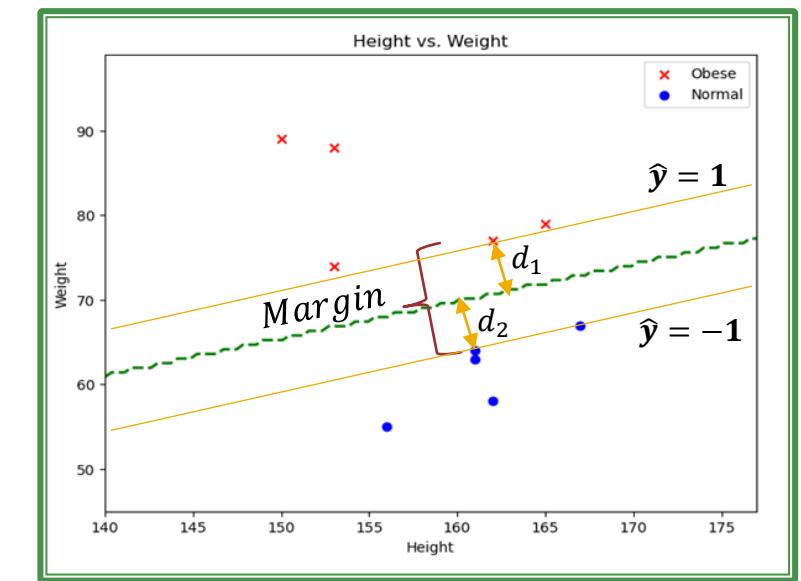
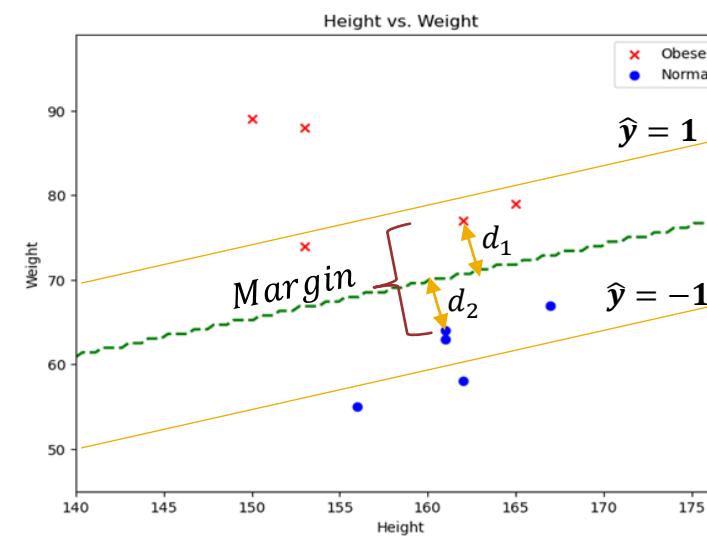
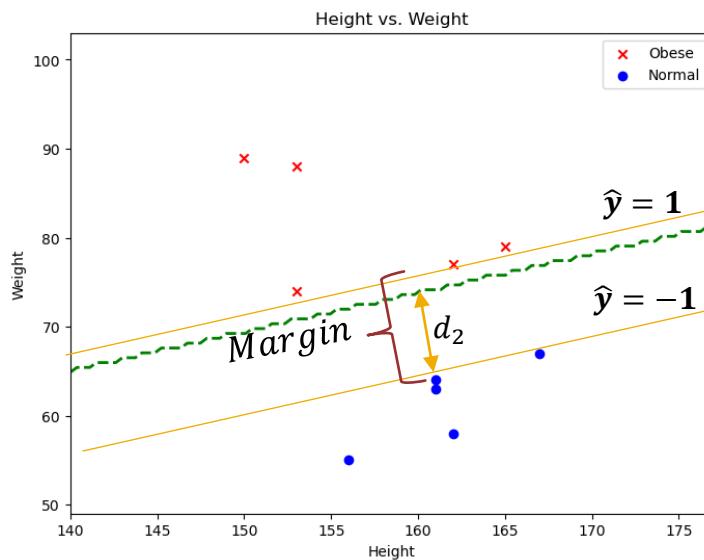
- Our target is to find the maximum possible margin by maximizing both  $d_1$  **AND**  $d_2$ .



# Margin



- Which of the following figures follows Ass. 1 and achieves maximum possible margin by maximizing both  $d_1$  **AND**  $d_2$ ?

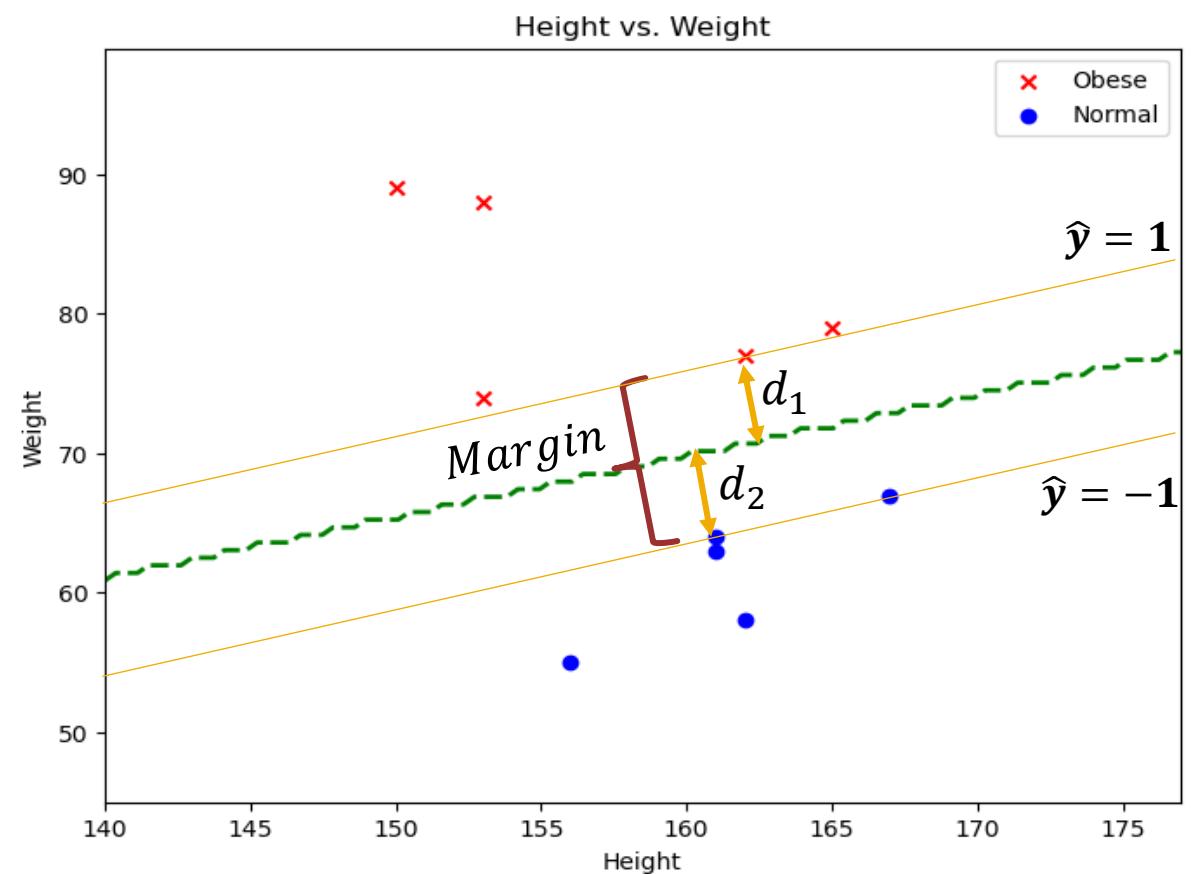


# SVM Formulation



- To have the maximum possible values of **both**  $d_1$  **AND**  $d_2$  to maximize the margin while abiding by Ass. 1  $\rightarrow$  Assume that the closest points to the boundary from each class have  $\hat{y}$  exactly equal to  $\pm 1$ . (Ass. 3)
- The margin now will be equal to:

$$\begin{aligned} \text{Margin} &= \left| \frac{\hat{y}_1}{\|\vec{w}\|} \right| + \left| \frac{\hat{y}_2}{\|\vec{w}\|} \right| \\ &= \frac{1}{\|\vec{w}\|} + \frac{1}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} \end{aligned}$$

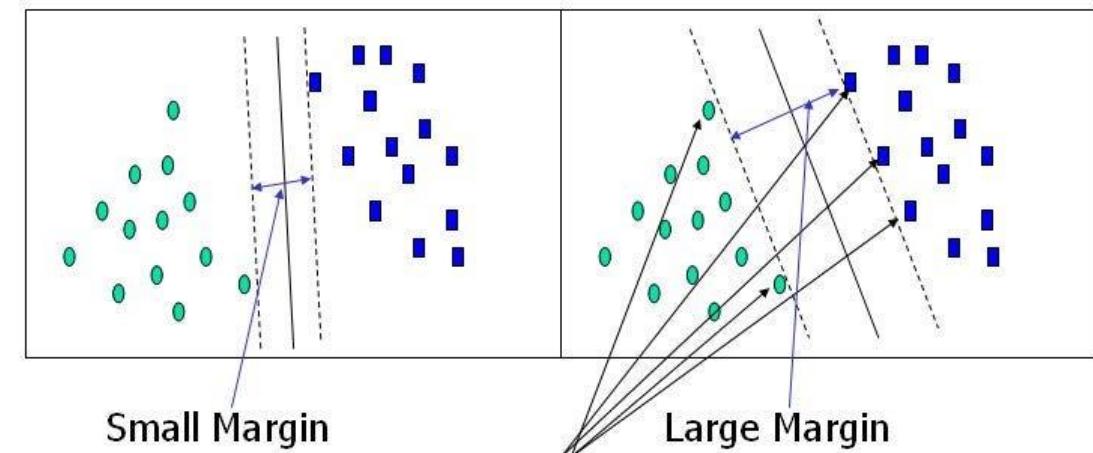


# SVM Formulation



## ■ Target of SVM:

- Maximize the margin (to have the best centered decision boundary)
- Assume that the closest points to the boundary from each class have  $\hat{y}$  exactly equal to  $\pm 1 \rightarrow \text{Ass.3}$ 
  - If we follow Ass. 3, then Ass.1 is already satisfied properly.
  - Ensures that all points are correctly classified and boundary is centered.



# SVM Formulation



- In mathematical form, the SVM target can be rewritten as follows:

$$\underset{\text{maximize}}{\frac{1}{2}} \frac{1}{\|\vec{w}\|}, \xrightarrow{\text{Margin}}$$

*such that,  $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$  for all points*  $\xrightarrow{\text{Ass. 3 and Ass. 1}}$

- It's preferred to do minimization rather than maximization, and to have differentiable terms:

- Take reciprocal
- Use  $\|\vec{w}\|^2$  instead of  $\|\vec{w}\|$

$$\underset{\text{minimize}}{\frac{1}{2}} \|\vec{w}\|^2,$$

*such that,  $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$  for all points*

Target Function

# SVM Loss Function



$$\begin{aligned} & \text{minimize} \frac{1}{2} \|\vec{w}\|^2, \\ & \text{such that, } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \end{aligned}$$

- To transform this condition-based target function to one loss (cost) function to be minimized, a concept called Lagrange Multipliers is used.

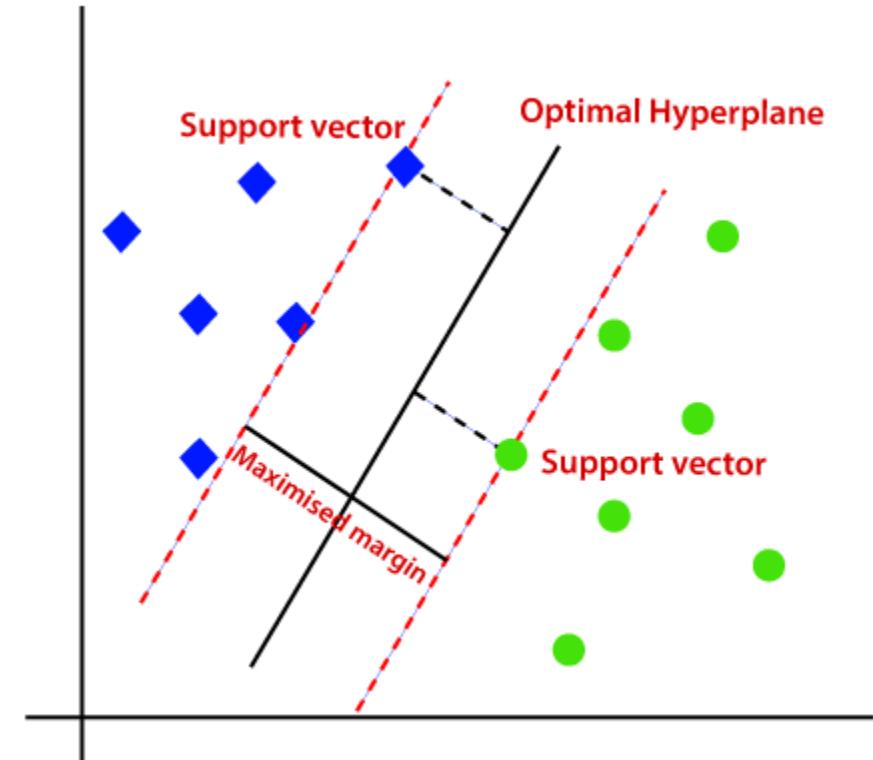
$$L(\vec{w}, b, \alpha_i) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^N \alpha_i (y_i(\vec{w} \cdot \vec{x}_i + b) - 1)$$

- This loss function can be minimized using Gradient Descent to reach an optimal decision boundary that follows the constraints and assumptions of SVM.

# Support Vectors



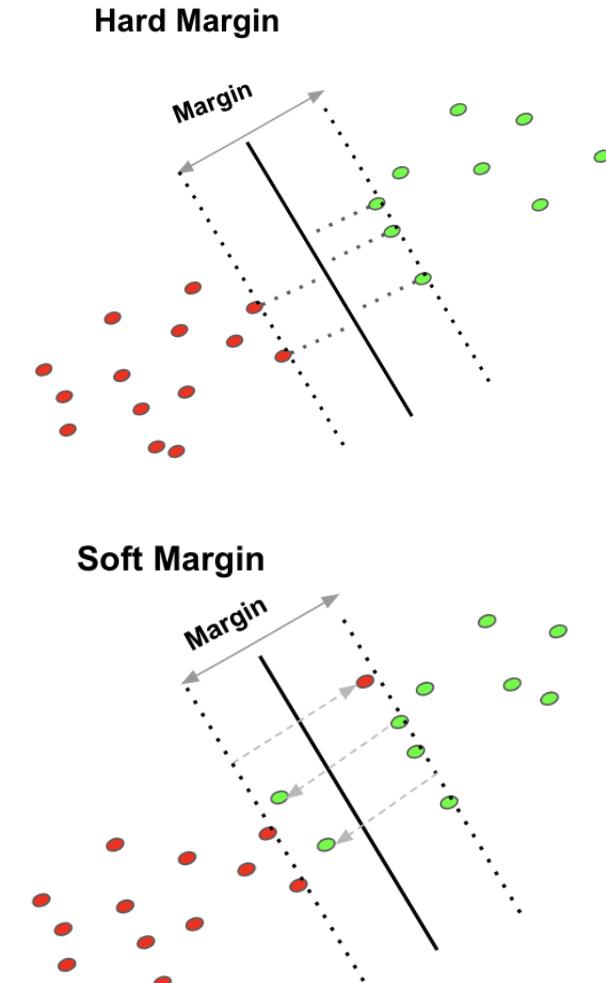
- The target function of the SVM algorithm is based on maximizing the margin which depends **only** on the closest points to the decision boundary.
- We call those closest points the **support vectors**.
- In other words, the decision boundary in an SVM algorithm is only affected by the support vectors, any other point in the dataset doesn't affect the final decision boundary.



# Hard vs Soft Margin



- The formulation we defined so far assumes a **Hard Margin** which only works if the data is perfectly linearly separable.
  - In other words, maximizing the margin while classifying **all** the points correctly only works if perfect linear separation exists.
- In practice, datasets are not perfectly linearly separable and have some noise. To make the SVM more robust, we need to define a **Soft Margin** instead.
  - The soft margin allows some points to be misclassified by lying inside the margin or on the incorrect side of the decision boundary.



# Soft Margin



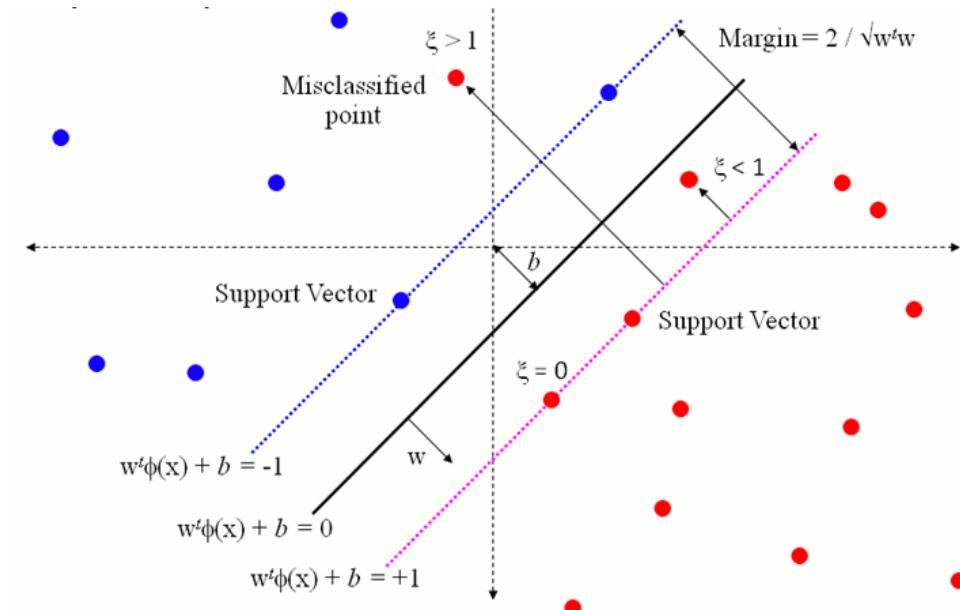
- To achieve a Soft Margin, we introduce a **slack variable**  $\xi_i$  for each point which represents the misclassification error of every point.

$$\xi_i = \begin{cases} 0, & \text{if } x_i \text{ is beyond correct margin} \\ 1 - y_i \hat{y}_i, & \text{if } x_i \text{ opposite to margin} \end{cases}$$

- In one equation:

$$\xi_i = \max(0, 1 - y_i \hat{y}_i)$$

- Points that lie far from their correct margin in the opposite side have higher  $\xi_i$  indicating higher error.



# Soft Margin: Target Function



## ■ Hard Margin Target Function:

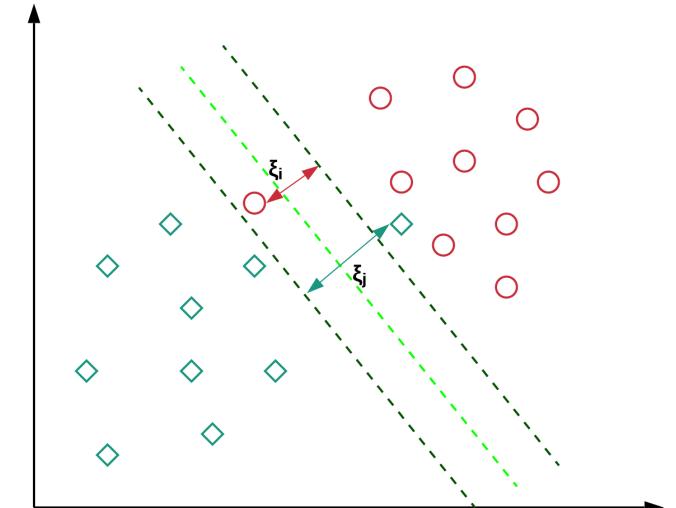
- Maximizes the margin only
- Constrains all points to strictly lie in the correct margin area.

$$\begin{aligned} & \text{minimize} \frac{1}{2} \|\vec{w}\|^2, \\ & \text{such that, } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \end{aligned}$$

## ■ Soft Margin Target Function:

- Maximizes the margin
- All points are not constrained to strictly lie in the correct margin area.
- Minimizes the total misclassification error of all points after multiplying by a constant scalar value  $C$ .

$$\begin{aligned} & \text{minimize} \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^N \xi_i, \\ & \text{such that, } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \end{aligned}$$



# Soft Margin: C Hyperparameter

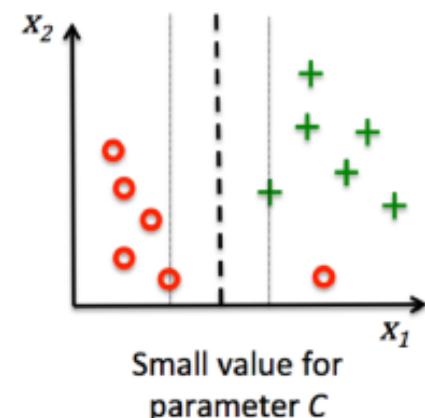
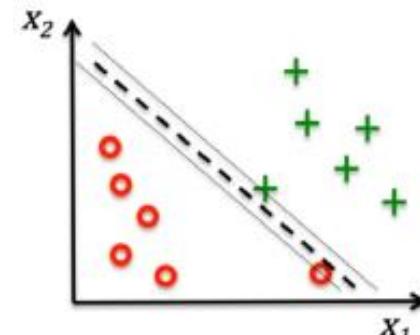


- Soft Margin Target Function:

$$\text{minimize} \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^N \xi_i,$$

such that,  $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$

- The  $C$  value is a manually defined hyperparameter and is the most important parameter in the SVM algorithm.
- It affects how strict is the SVM algorithm in allowing misclassification errors. In other words, it controls the trade-off between having a maximized margin and minimum misclassified points.
  - Very Large  $C$  value → Prioritize minimizing errors → Very narrow margin.
  - Very Low  $C$  value → Prioritize maximizing the margin → More misclassified points



# Soft Margin: Loss Minimization



$$L(\vec{w}, \mathbf{b}, \alpha_i) = \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i (\vec{w} \cdot \vec{x}_i + \mathbf{b}) - 1 - \xi_i)$$

- As done in Logistic Regression, get the values of  $\vec{w}, \mathbf{b}$  at which  $L(\vec{w}, \mathbf{b})$  is minimum  $\rightarrow$  Get the values of  $\vec{w}, \mathbf{b}$  at which  $\frac{d(L)}{d\vec{w}} = 0$ , and  $\frac{d(L)}{d\mathbf{b}} = 0$ .
- After getting the derivatives, we substitute back in the loss function to get the final objective function (called the dual formulation) that could be solved using Gradient Descent.

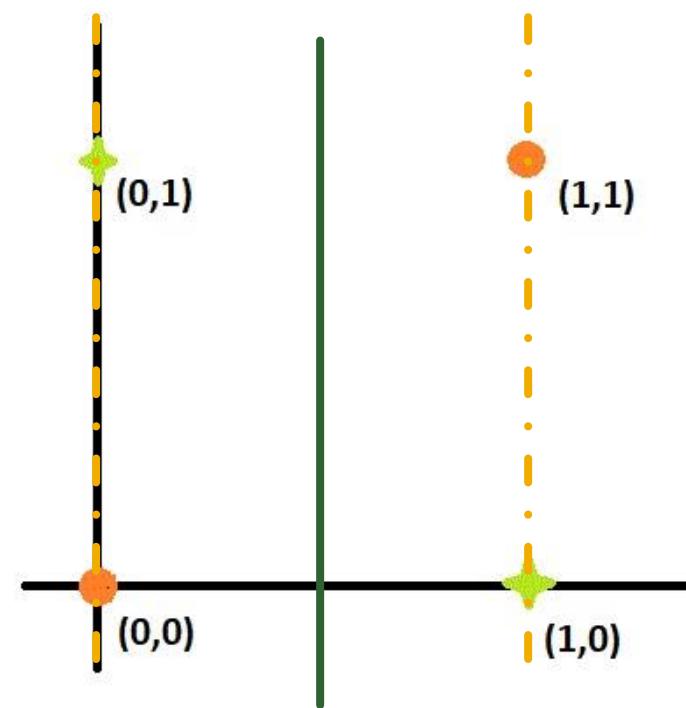
$$\begin{aligned} \max \quad & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \end{aligned}$$

**Dual Formulation**

# Non-Linearly Separable



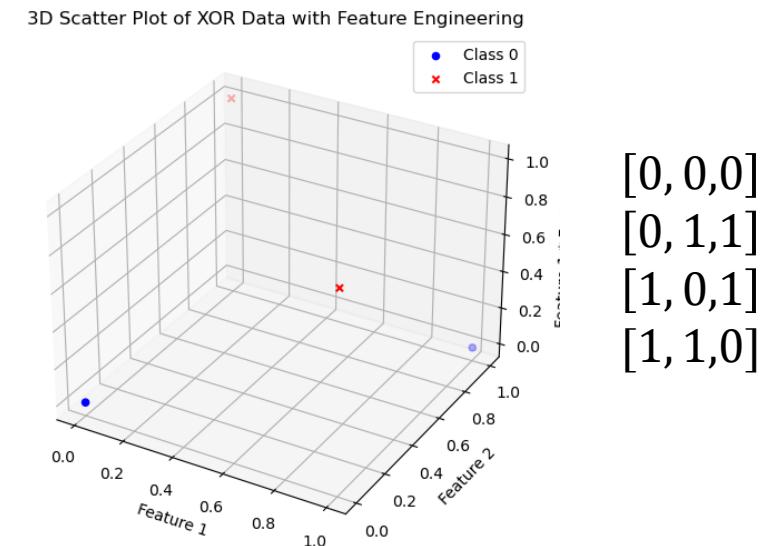
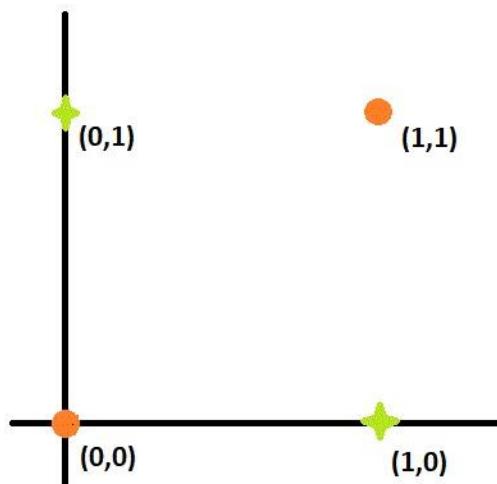
- Recall the XOR problem below. Can the Soft Margin SVM algorithm correctly solve the problem?
- The Soft Margin SVM only gives us the ability to be robust to noise in the dataset by allowing some points to be misclassified.
- It doesn't result in a non-linear decision boundary which is needed to solve the XOR problem.



# Non-Linearly Separable



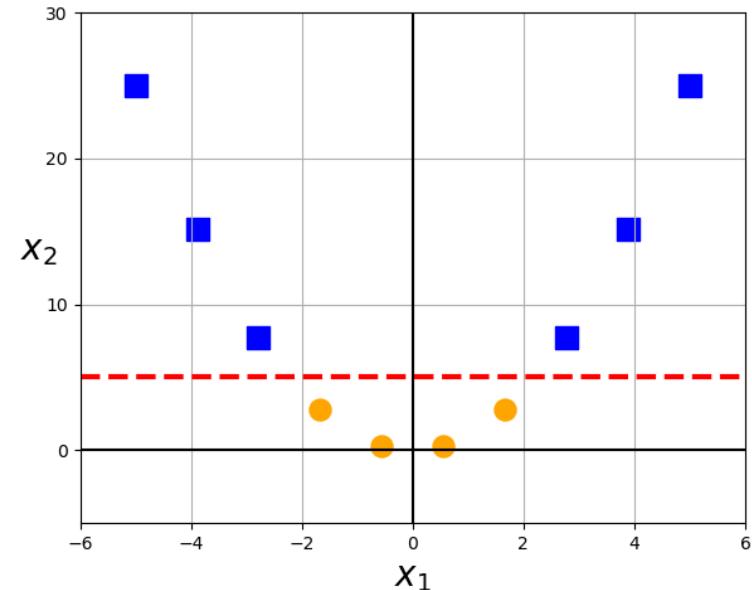
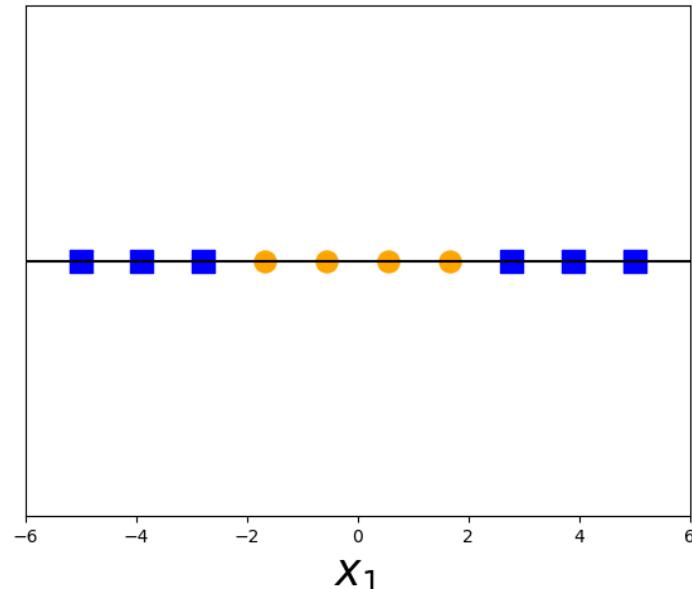
- Non-linearly separable datasets could actually be linearly separable in a **higher** dimensional space.
- In the XOR problem, we could add a third dimension representing the operation  $x \text{ } XOR \text{ } y$ . The problem then becomes linearly separable in the higher dimension (3D).



# Another Example



- Find a non-linear transformation on the 1-D dataset below to transform it to a linearly separable version in a higher dimensional space.
- Solution: In 2D Space,  $y = x^2$



# Kernels



- A kernel function can be used to raise data to higher dimensional feature spaces.
- They are basically functions that take two vectors as input and returns the equivalent for the dot product of their ***transformed*** versions in higher dimensional spaces.

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

- Examples:

- Polynomial Kernel:

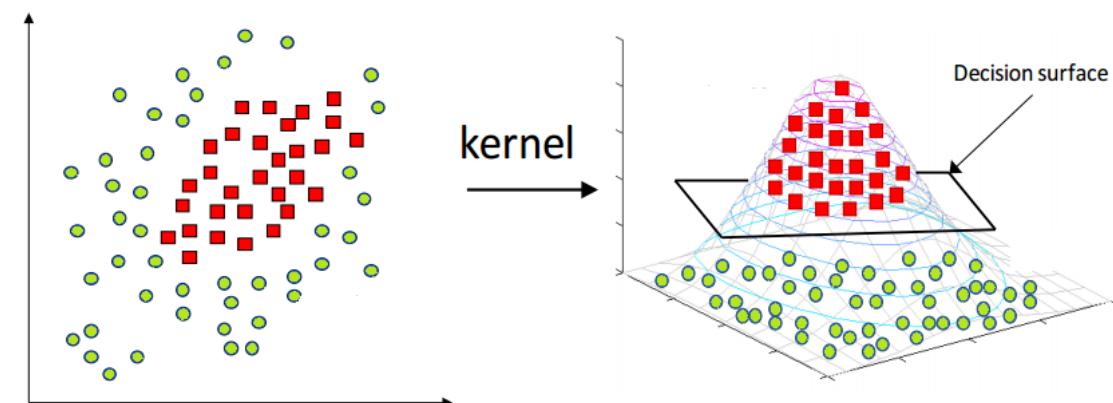
$$k(\mathbf{x}, \mathbf{y}) = [1 + (\mathbf{x}^T \cdot \mathbf{y})]^d$$

- Gaussian Radial Basis Function:

$$k(\mathbf{x}, \mathbf{y}) = \exp(-|\mathbf{x} - \mathbf{y}|^2 / 2\sigma^2)$$

- Sigmoid Functions:

$$k(\mathbf{x}, \mathbf{y}) = \tanh(c\mathbf{x}^T \cdot \mathbf{y} + h)$$



# Kernel Trick

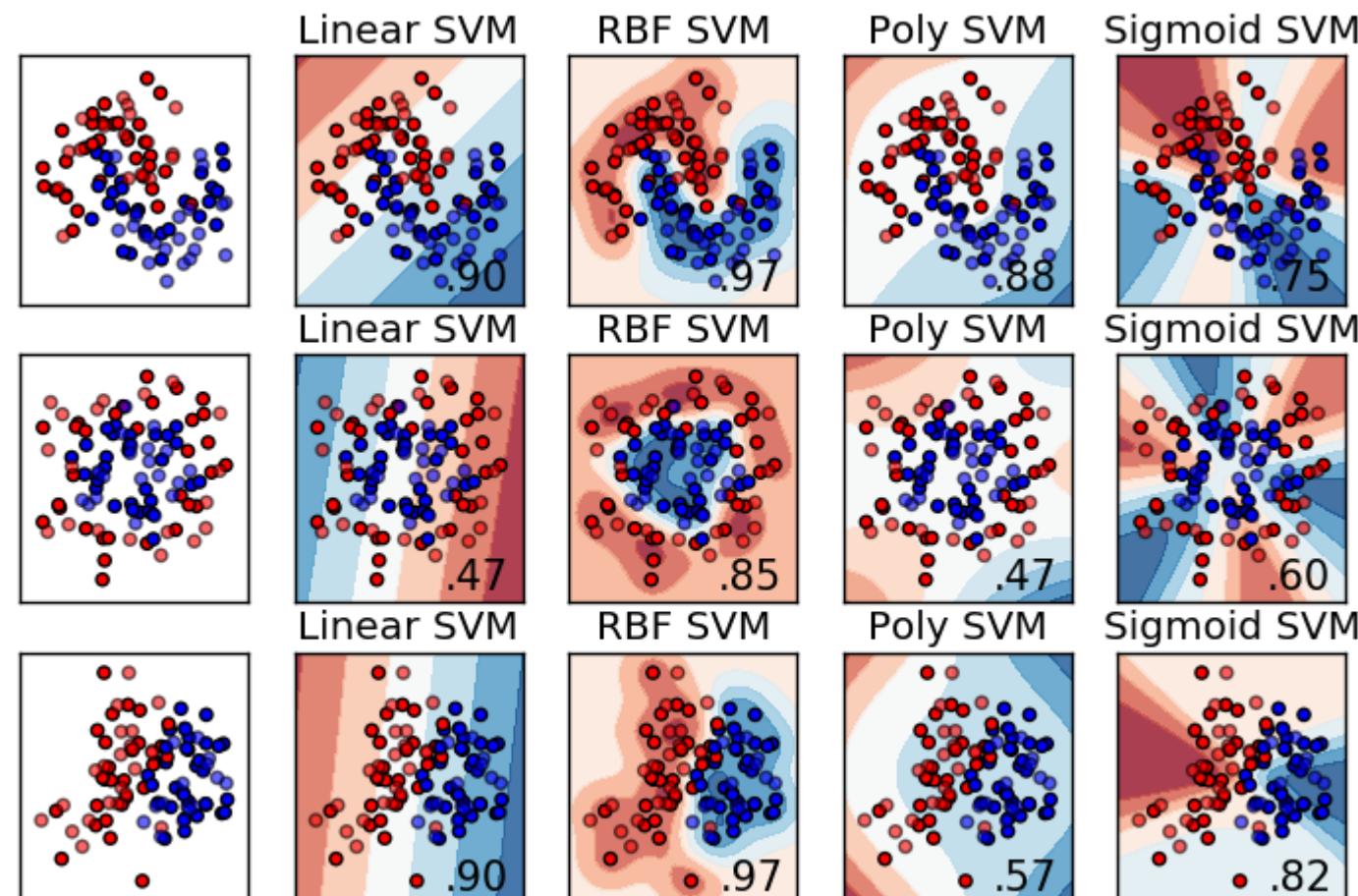


- Since kernels basically mimic the dot product operation in higher dimensionality spaces, how can they be used in SVM?
- Recall the dual formulation which is the core objective function to be optimized in SVMs:

$$\begin{aligned} \max \quad & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \end{aligned}$$

- The dual formulation already contains a dot product between the points in the original feature space  $\rightarrow x_i^T \cdot x_j$
- The Kernel Trick replaces this dot product with a kernel function  $k(x_i, x_j)$ . This allows us to solve the objective function (dual formulation) in **higher** dimensional feature space.
- If a dataset was non-linearly separable in the original dimensions, it could potentially be linearly separable in higher dimensionalities.

# Kernel Trick



# Extra Resources



- More information regarding deriving the dual formulation and Lagrangian multipliers:
  - <https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/#lagrange-multiplier>

# Thank you!



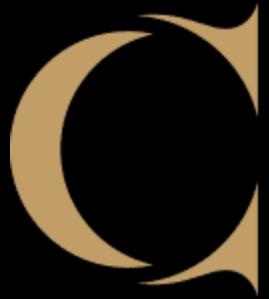
- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



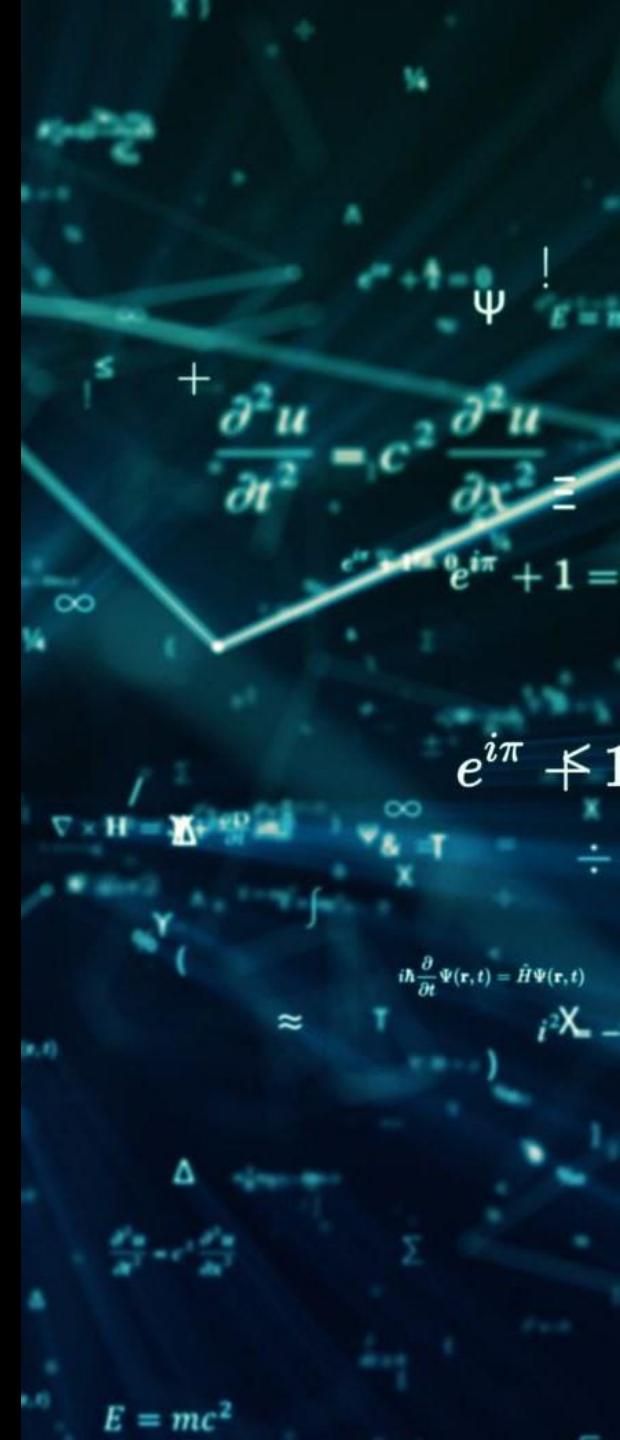
**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Classification

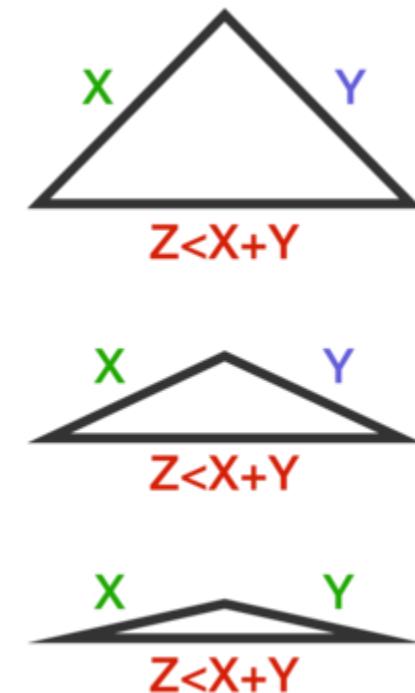
- KNN
  - Decision Trees
  - Tree Ensembles



# Distance Metrics



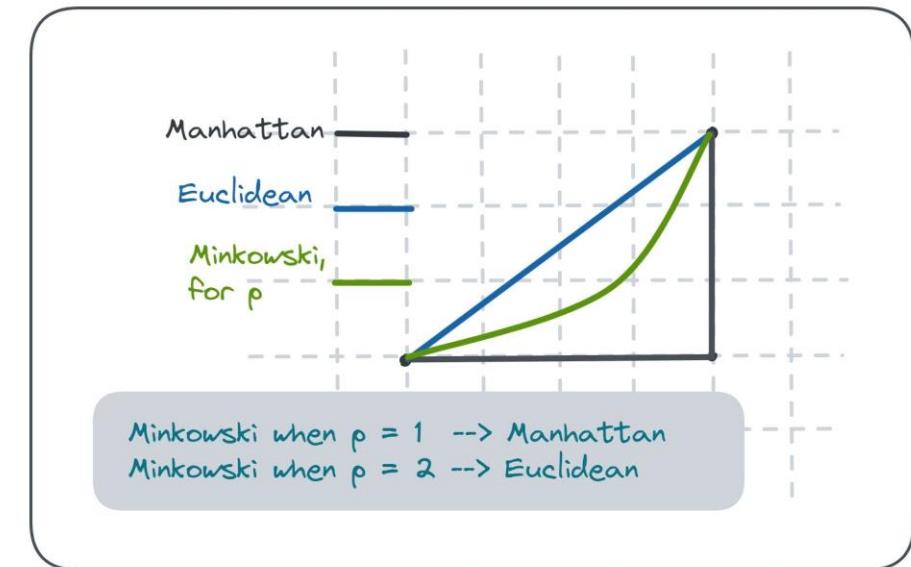
- Given two data points  $a$  and  $b$  in a dataset with  $M$  features, we need to quantify how similar/dissimilar those points are from each other.
- A distance metric  $d(a, b)$  measures how far or close two points are from each other. All distance metrics must follow the following criteria:
  - Non-negativity:  $d(a, b) > 0$
  - Identity: if  $d(a, b) = 0$ , then  $a = b$
  - Symmetry:  $d(a, b) = d(b, a)$
  - Triangular Inequality:  $d(a, b) \leq d(a, c) + d(b, c)$



# Distance Metrics



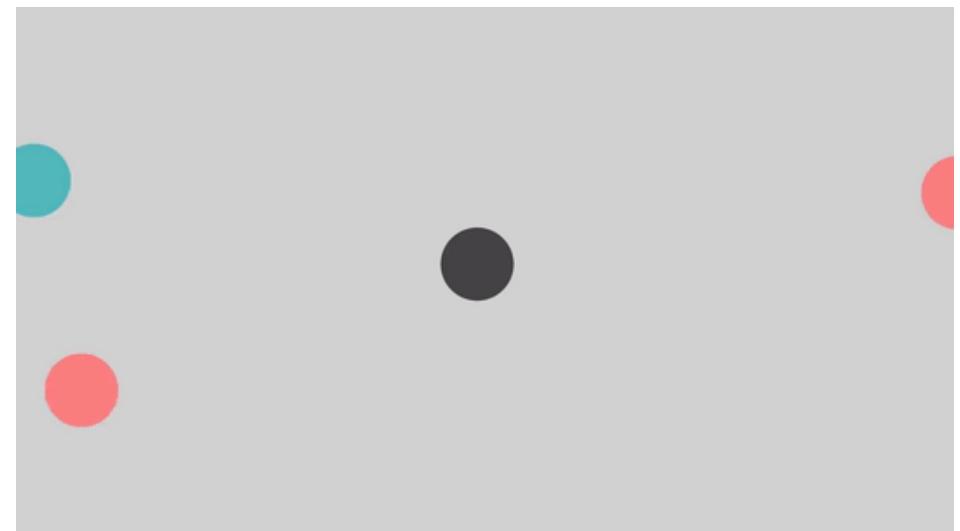
- Euclidean Distance:
  - Computes Length of the straight line between two points.
  - Formulation:  $d(a, b) = \sqrt{\sum_{i=1}^M (a[i] - b[i])^2}$
  - Drawbacks: Could be sensitive to outliers.
- Manhattan Distance:
  - Computes orthogonal (grid line) distances between two points.
  - Formulation:  $d(a, b) = \sum_{i=1}^M |a[i] - b[i]|$
  - Pros: Less sensitive to outliers.
- Minkowski Distance:
  - Generalization of both Euclidean and Manhattan distance between two points.
  - Formulation:  $d(a, b) = \left( \sum_{i=1}^M |a[i] - b[i]|^p \right)^{1/p}$
  - Pros: Allows for tuning sensitivity to different dimensions through the parameter  $p$ .



# K-Nearest Neighbours (KNN)



- KNN is a **non-linear** machine learning algorithm that makes predictions by finding the K training examples in the dataset that are closest (in terms of some distance metric) to the input data point.
- It uses the labels of those K neighbors to make a prediction.
- K is a manually defined hyperparameter that determines the number of neighbors to observe for each new point.

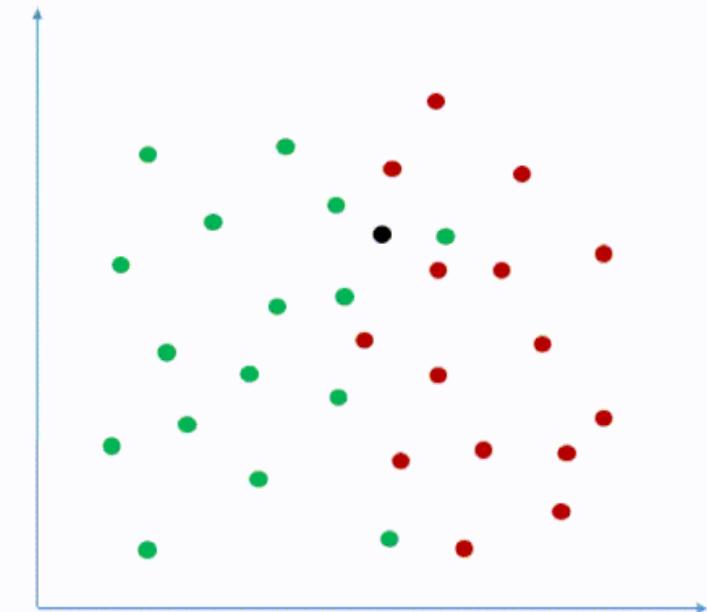


# K-Nearest Neighbours (KNN)



- Given a new query point  $x$  to be labeled, a dataset with  $N$  data points and a hyperparameter  $K$ :
  - Calculate distance  $d_i(x, n_i)$  between point  $x$  and all points  $n_i$  in the dataset.
  - Determine the nearest  $K$  points to the point  $x$  to be labeled  $\rightarrow K$  points with the smallest distances  $d_i(x, n_i)$ .
  - The point  $x$  will be labeled based on the majority label of the  $K$  nearest neighbours.

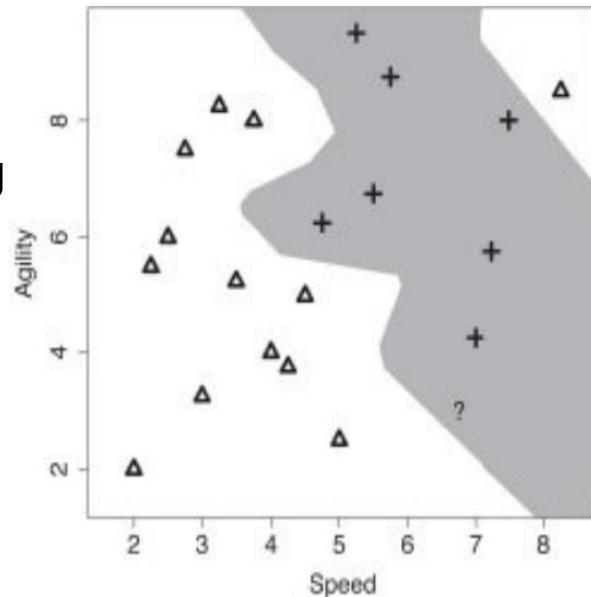
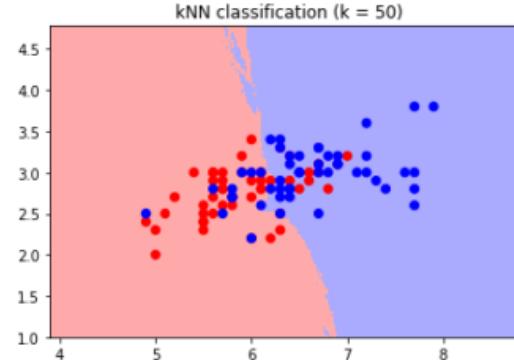
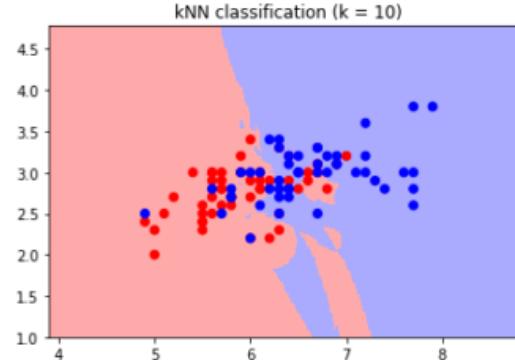
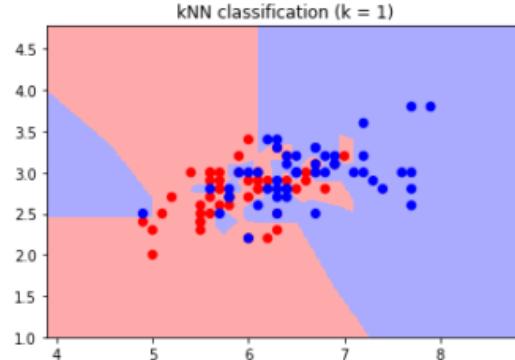
K-Nearest Neighbors Classification



# K-Nearest Neighbours (KNN)



- The choice of the hyperparameter K significantly influences the shape and flexibility of the decision boundary in K-NN.
  - Smaller K:
    - The decision boundary can be highly irregular
    - The model may be sensitive to outliers and local variations in the data.
    - Can lead to overfitting, capturing noise in the data rather than the underlying patterns.
  - Larger K:
    - The decision boundary tends to be smoother
    - Provides a more generalized view of the data, potentially avoiding overfitting

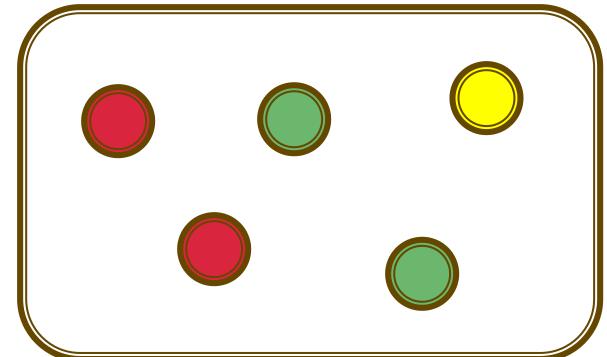


# Tree-Based Algorithms

# Entropy



- Entropy ( $H$ ) is used to represent *the amount of uncertainty* that a particular event will occur.
- For example, consider a box with two red balls and two green balls and two yellow balls.
  - Assuming no yellow balls, What is the probability of drawing a red ball?
    - Probability  $\rightarrow 50\%$
    - Uncertainty (intuitively)  $\rightarrow High$
  - What is the probability of drawing a colored ball?
    - Probability  $\rightarrow 100\%$
    - Uncertainty (intuitively)  $\rightarrow Zero$
  - What is the probability of drawing a red or green ball?
    - Probability  $\rightarrow 80\%$
    - Uncertainty (intuitively)  $\rightarrow Low$
  - What is the probability of drawing a purple ball?
    - Probability  $\rightarrow 0\%$
    - Uncertainty (intuitively)  $\rightarrow Infinitely High$



# Entropy



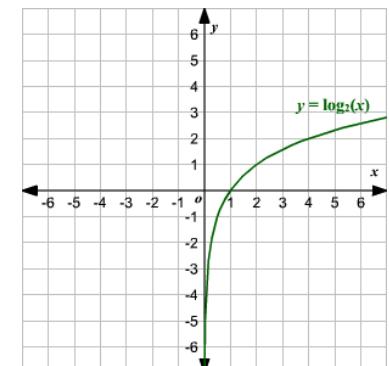
- Given a probability  $p$  that an event will occur, can we mathematically formulate the entropy/uncertainty ( $H$ ) ?

- If  $p = 1.0 \rightarrow \text{Entropy} = 0$
- If  $p = 0.0 \rightarrow \text{Entropy} \rightarrow \infty$
- If  $p = 0.5 \rightarrow \text{Entropy} = 1.0$
- If  $p > 0.5 \rightarrow \text{Entropy} < 1.0$
- If  $p < 0.5 \rightarrow \text{Entropy} > 1.0$

}  $H = \log(p)$  ?

Need to handle -ve  
 $\log(p)$  to have +ve  
Entropy

$$H = -\log(p)$$



# Entropy



- For the previous box example, is there a way to quantify the overall entropy/uncertainty ( $H$ ) of the whole box?

- $H(Box) = P(R)H(R) + P(G)H(G) + P(Y)H(Y)$

- $H(Box) = \frac{2}{5} * -\log\left(\frac{2}{5}\right) + \frac{2}{5} * -\log\left(\frac{2}{5}\right) + \frac{1}{5} * -\log\left(\frac{1}{5}\right)$

- $H(Box) = 2.3218$

- What if the box had no yellow balls?

- $H(Box) = P(R)H(R) + P(G)H(G)$

- $H(Box) = \frac{1}{2} * -\log\left(\frac{1}{2}\right) + \frac{1}{2} * -\log\left(\frac{1}{2}\right)$

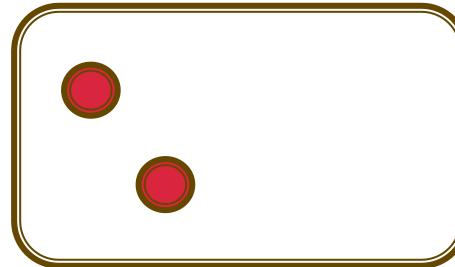
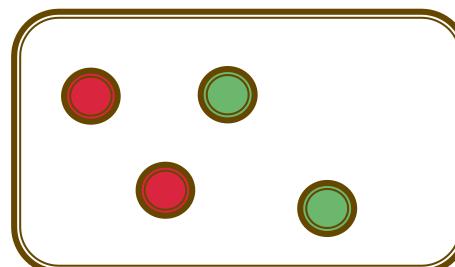
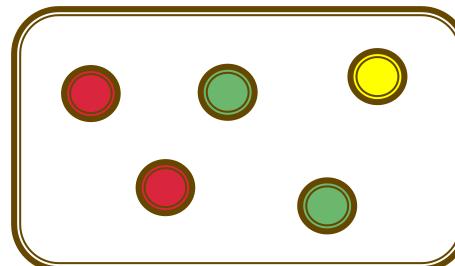
- $H(Box) = 1.0$

- What if the box had only red balls?

- $H(Box) = P(R)H(R)$

- $H(Box) = \frac{1}{1} * -\log\left(\frac{1}{1}\right)$

- $H(Box) = 0.0$



# Entropy



- Generally, for a dataset with  $N$  classes, the entropy of the full dataset represents that amount of impurity in the labels. It is formulated as follows:

$$H(\text{Dataset}) = - \sum_{i=1}^N P(c_i) * \log_2 P(c_i)$$

- This formulation is called Shannon's model of entropy and represents the foundation of information theory, providing a great measure of impurity.

# Decision Trees: Intuition



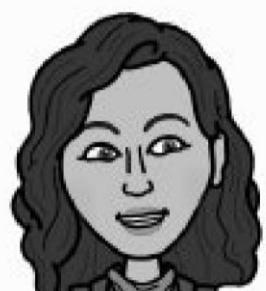
- Assume we have 4 different people with the features shown below. Can we determine (classify) the name of each person by asking a sequence of questions based on the features?



Brian



John



Aphra



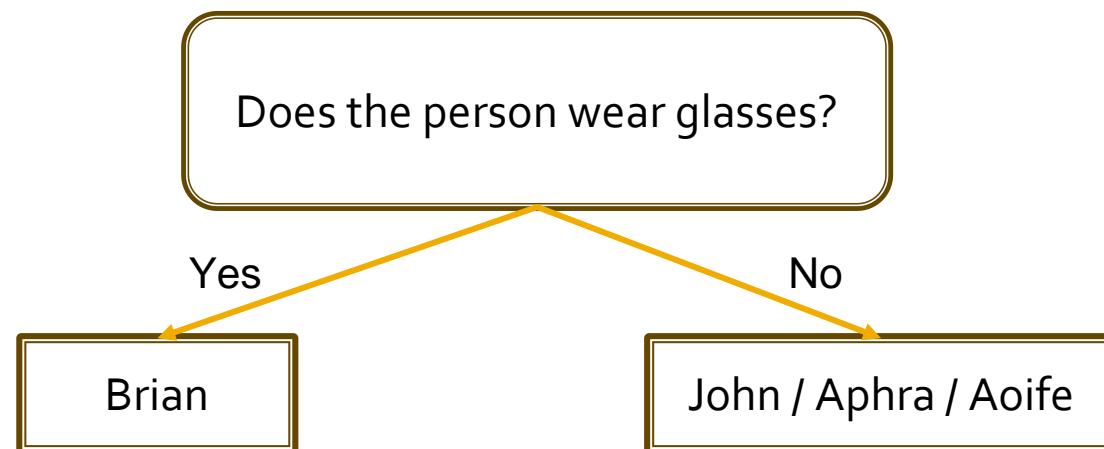
Aoife

| Man | Long Hair | Glasses | Name  |
|-----|-----------|---------|-------|
| Yes | No        | Yes     | Brian |
| Yes | No        | No      | John  |
| No  | Yes       | No      | Aphra |
| No  | No        | No      | Aoife |

# Decision Trees: Intuition



- Let's start by asking the following question: "*Does the person wear glasses?*"



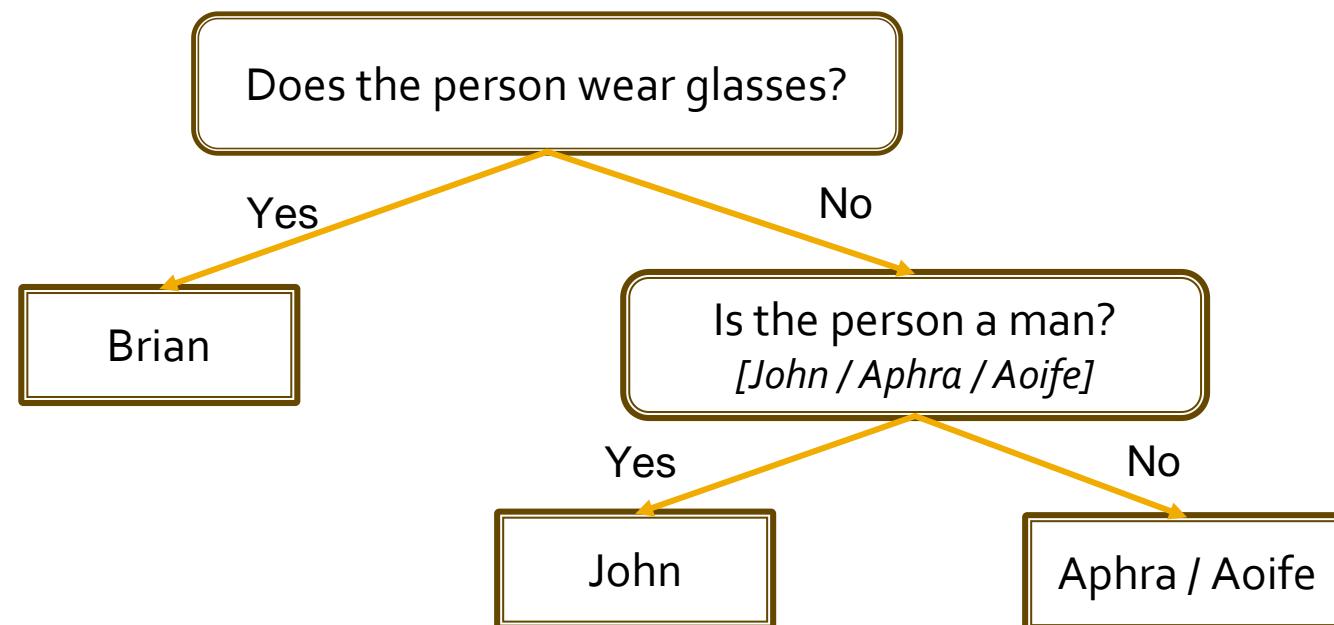
| Man | Long Hair | Glasses | Name  |
|-----|-----------|---------|-------|
| Yes | No        | Yes     | Brian |
| Yes | No        | No      | John  |
| No  | Yes       | No      | Aphra |
| No  | No        | No      | Aoife |

- "Brian" was correctly classified, three names are remaining for us.

# Decision Trees: Intuition



- For the three remaining people, we can ask about the gender:  
*“Is the person a man?”*



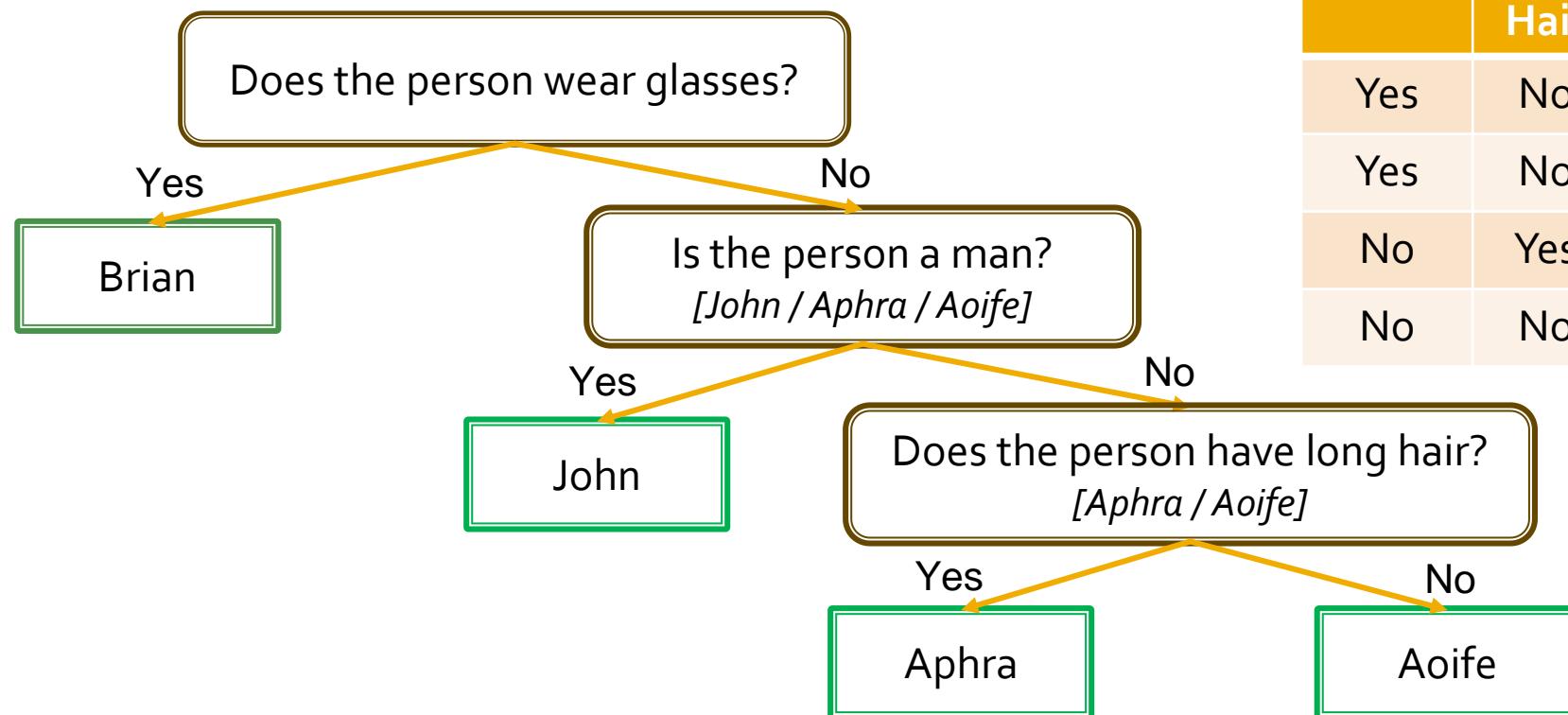
| Man | Long Hair | Glasses | Name  |
|-----|-----------|---------|-------|
| Yes | No        | Yes     | Brian |
| Yes | No        | No      | John  |
| No  | Yes       | No      | Aphra |
| No  | No        | No      | Aoife |

- “Brian” and “John” are now correctly classified, two names are remaining for us.

# Decision Trees: Intuition



- For the two remaining people, we can ask about the hair:  
*“Does the person have long hair?”*

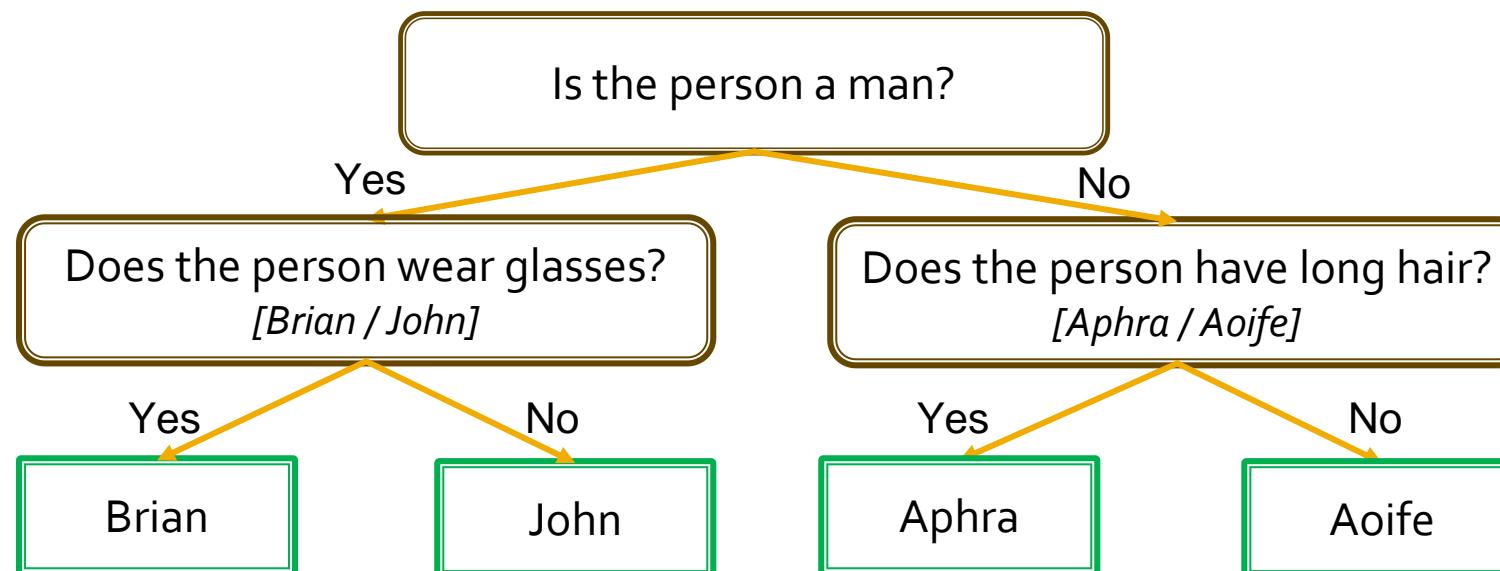


| Man | Long Hair | Glasses | Name  |
|-----|-----------|---------|-------|
| Yes | No        | Yes     | Brian |
| Yes | No        | No      | John  |
| No  | Yes       | No      | Aphra |
| No  | No        | No      | Aoife |

# Decision Trees: Intuition



- Is it possible to reach the same classification with less question sequence length per name?



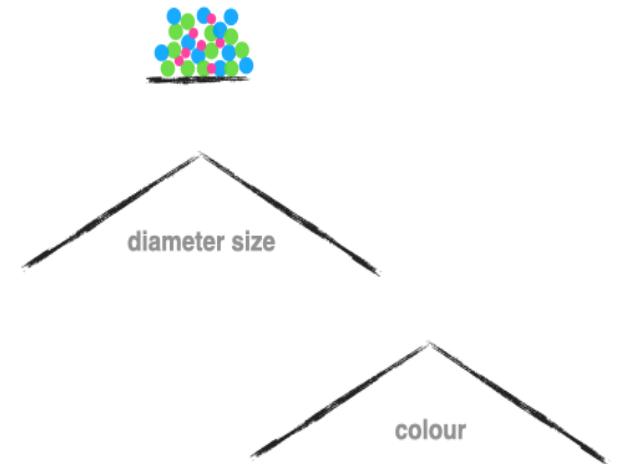
| Man | Long Hair | Glasses | Name  |
|-----|-----------|---------|-------|
| Yes | No        | Yes     | Brian |
| Yes | No        | No      | John  |
| No  | Yes       | No      | Aphra |
| No  | No        | No      | Aoife |

- Depending on which feature we choose first, we could end up building a shorter tree with less questions to classify each person.
- **Is there a systematic way we can always choose the best feature (question) in every step to reduce the total number of questions per label?**

# Decision Tree



- The decision tree is a **non-linear** supervised machine learning algorithm.
- It recursively splits the dataset into subsets based on the most significant attribute at each level.
- It keeps on splitting the dataset based on different attributes, creating a tree-like structure.
- The main goal is to reach a final tree that is as shallow as possible and has homogenous classes in the leaf nodes (bottom-most nodes of the tree).



# Decision Tree



- Consider the following dataset  $S$  which predicts whether a person should go for a picnic or not based on weather features.
- Assume we want to build a **decision tree** to do this classification.
- Which is the best feature that we should start splitting by?
  - To calculate the best feature, we will use the concept of **Entropy** to build a new metric called **Information Gain**.

| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Hot      | High    | Weak     | No          |
| Mild     | Normal  | Strong   | No          |
| Hot      | High    | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | High    | Weak     | No          |
| Cool     | Normal  | Strong   | Yes         |
| Mild     | High    | Weak     | Yes         |
| Cool     | Normal  | Weak     | Yes         |

# Information Gain



- **Information Gain (IG)** is the reduction in entropy  $H$  (impurity) achieved by splitting a set into subsets based on a particular feature  $F$ .
- In other words, it measures how much will the total impurity of the classes decrease after splitting the dataset based on a particular feature.
- Mathematically:

$$IG(F) = H(\text{Dataset}) - \text{Entropy of Dataset after split on } F$$

$$IG(F) = H(\text{Dataset}) - \sum_{v \text{ in } \text{Values}(F)} P(F = v) * H(\text{Dataset} | F = v)$$

- Generally, we want to choose the feature the gives us the **highest** information gain. This is because it is the feature leading to the highest decrease in impurity in the classes.

# Decision Tree



- Let's calculate the Information Gain for the Temp/Hum/Wind Features and choose the best one accordingly.

- $$H(S) = -[P(O = Yes) * \log(O = Yes) + P(O = No) * \log(O = No)]$$

$$H(S) = -\left[\frac{5}{9} * \log\left(\frac{5}{9}\right) + \frac{4}{9} * \log\left(\frac{4}{9}\right)\right] = 0.991$$

- $$IG(T) = H(S) - [P(T = Hot) * H(S|T = Hot) + P(T = Mild) * H(S|T = Mild) + P(T = Cool) * H(S|T = Cool)]$$

- $$IG(T) = 0.991 - \left[\frac{3}{9} * \left(-\left[\frac{1}{3} * \log\left(\frac{1}{3}\right) + \frac{2}{3} * \log\left(\frac{2}{3}\right)\right]\right) + \frac{3}{9} * \left(-\left[\frac{1}{3} * \log\left(\frac{1}{3}\right) + \frac{2}{3} * \log\left(\frac{2}{3}\right)\right]\right) + \frac{3}{9} * \left(-\left[\frac{2}{3} * \log\left(\frac{2}{3}\right) + \frac{1}{3} * \log\left(\frac{1}{3}\right)\right]\right)\right] = 0.0727$$

| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Hot      | High    | Weak     | No          |
| Mild     | Normal  | Strong   | No          |
| Hot      | High    | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | High    | Weak     | No          |
| Cool     | Normal  | Strong   | Yes         |
| Mild     | High    | Weak     | No          |
| Cool     | Normal  | Weak     | Yes         |

# Decision Tree



- Let's calculate the Information Gain for the Temp/Hum/Wind Features and choose the best one accordingly.

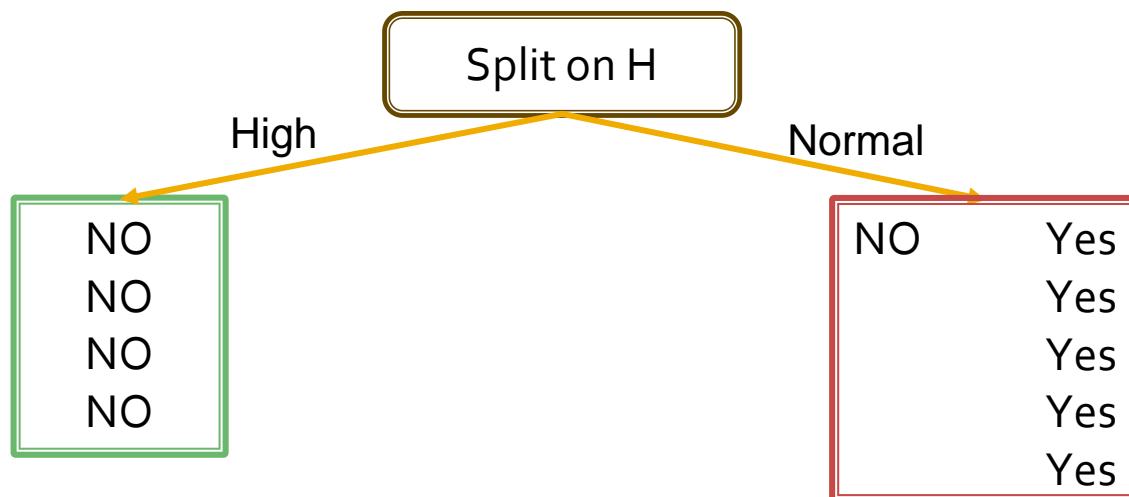
- $$IG(H) = H(S) - [P(H = High) * H(S|H = High) + P(H = Normal) * H(S|H = Normal)]$$
- $$IG(H) = 0.991 - \left[ \frac{4}{9} * \left( - \left[ \frac{0}{4} * \log \left( \frac{0}{4} \right) + \frac{4}{4} * \log \left( \frac{4}{4} \right) \right] \right) + \frac{5}{9} * \left( - \left[ \frac{4}{5} * \log \left( \frac{4}{5} \right) + \frac{1}{5} * \log \left( \frac{1}{5} \right) \right] \right) \right] = 1.1699$$
- $$IG(W) = H(S) - [P(W = Weak) * H(S|W = Weak) + P(W = Strong) * H(S|W = Strong)]$$
- $$IG(W) = 0.991 - \left[ \frac{6}{9} * \left( - \left[ \frac{3}{6} * \log \left( \frac{3}{6} \right) + \frac{3}{6} * \log \left( \frac{3}{6} \right) \right] \right) + \frac{3}{9} * \left( - \left[ \frac{1}{3} * \log \left( \frac{1}{3} \right) + \frac{2}{3} * \log \left( \frac{2}{3} \right) \right] \right) \right] = 0.9183$$

| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Hot      | High    | Weak     | No          |
| Mild     | Normal  | Strong   | No          |
| Hot      | High    | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | High    | Weak     | No          |
| Cool     | Normal  | Strong   | Yes         |
| Mild     | High    | Weak     | No          |
| Cool     | Normal  | Weak     | Yes         |

# Decision Tree



- Based on the previous calculations, it looks like Humidity (H) is the best feature to split on in the initial step as it has the highest Information Gain.



| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Hot      | High    | Weak     | No          |
| Mild     | Normal  | Strong   | No          |
| Hot      | High    | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | High    | Weak     | No          |
| Cool     | Normal  | Strong   | Yes         |
| Mild     | High    | Weak     | No          |
| Cool     | Normal  | Weak     | Yes         |

- “High” branch is now fully homogenous, no need to further split it.
- “Normal” branch is not yet homogenous, we need to split it further. We will repeat the same process to choose between “Wind” and “Temp” features.

# Decision Tree



- Our dataset now is only consisting of the split where “*Hum = Normal*”. We will choose between *T* and *W*.
- $H(S|H = \text{Normal}) = -[P(O = \text{Yes}) * \log(O = \text{Yes}) + P(O = \text{No}) * \log(O = \text{No})]$
- $H(S|H = \text{Normal}) = -\left[\frac{4}{5} * \log\left(\frac{4}{5}\right) + \frac{1}{5} * \log\left(\frac{1}{5}\right)\right] = 0.720$
- $IG(T) = H(S|H = \text{Normal}) - [P(T = \text{Hot}) * H(S|T = \text{Hot}) + P(T = \text{Mild}) * H(S|T = \text{Mild}) + P(T = \text{Cool}) * H(S|T = \text{Cool})]$
- $IG(T) = 0.720 - \left[\frac{1}{5} * \left(-\left[\frac{1}{1} * \log\left(\frac{1}{1}\right) + \frac{0}{1} * \log\left(\frac{0}{1}\right)\right]\right) + \frac{2}{5} * \left(-\left[\frac{1}{2} * \log\left(\frac{1}{2}\right) + \frac{1}{2} * \log\left(\frac{1}{2}\right)\right]\right) + \frac{2}{5} * \left(-\left[\frac{2}{2} * \log\left(\frac{2}{2}\right) + \frac{0}{2} * \log\left(\frac{0}{2}\right)\right]\right)\right] = 0.320$

| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Mild     | Normal  | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | Normal  | Strong   | Yes         |
| Cool     | Normal  | Weak     | Yes         |

# Decision Tree



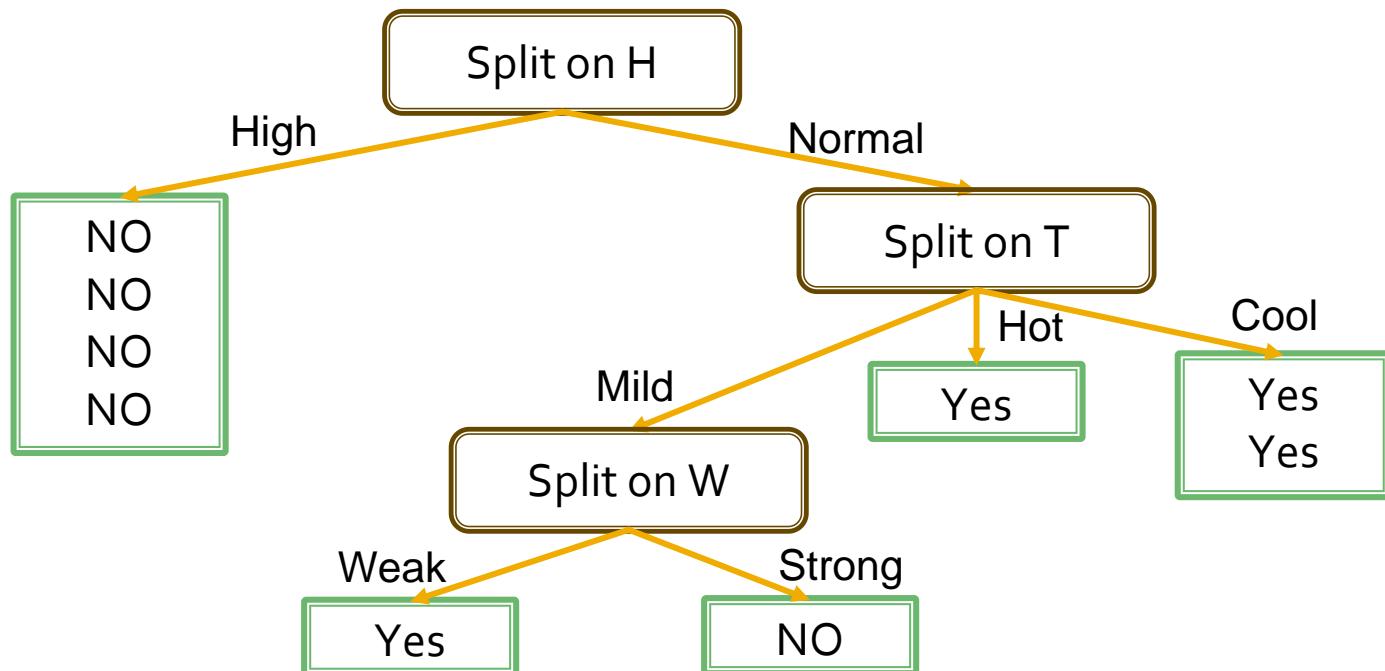
- $IG(W) = H(S|H = Normal) - [P(W = Weak) * H(S|W = Weak) + P(W = Strong) * H(S|W = Strong)]$
- $IG(W) = 0.720 - \left[ \frac{3}{5} * \left( - \left[ \frac{3}{3} * \log\left(\frac{3}{3}\right) + \frac{0}{3} * \log\left(\frac{0}{3}\right) \right] \right) + \frac{2}{5} * \left( - \left[ \frac{1}{2} * \log\left(\frac{1}{2}\right) + \frac{1}{2} * \log\left(\frac{1}{2}\right) \right] \right) \right] = 0.320$

| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Mild     | Normal  | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | Normal  | Strong   | Yes         |
| Cool     | Normal  | Weak     | Yes         |

# Decision Tree



- Based on the previous calculations, it looks like both Temp and Wind have equal Information Gains, we can choose any of them to split on. Let's go with Temp. "Mild" branch only has Wind to split on, we go with it.



| Temp (T) | Hum (H) | Wind (W) | Outside (O) |
|----------|---------|----------|-------------|
| Hot      | High    | Weak     | No          |
| Mild     | Normal  | Strong   | No          |
| Hot      | High    | Strong   | No          |
| Mild     | Normal  | Weak     | Yes         |
| Hot      | Normal  | Weak     | Yes         |
| Cool     | High    | Weak     | No          |
| Cool     | Normal  | Strong   | Yes         |
| Mild     | High    | Weak     | No          |
| Cool     | Normal  | Weak     | Yes         |

- The resulting Decision Tree has homogenous leaf nodes, therefore no further splitting needs to be done.
- We are guaranteed that the final decision tree is the "shallowest" one possible, since we used highest Information gains while splitting.

# Decision Tree: ID3 Algorithm

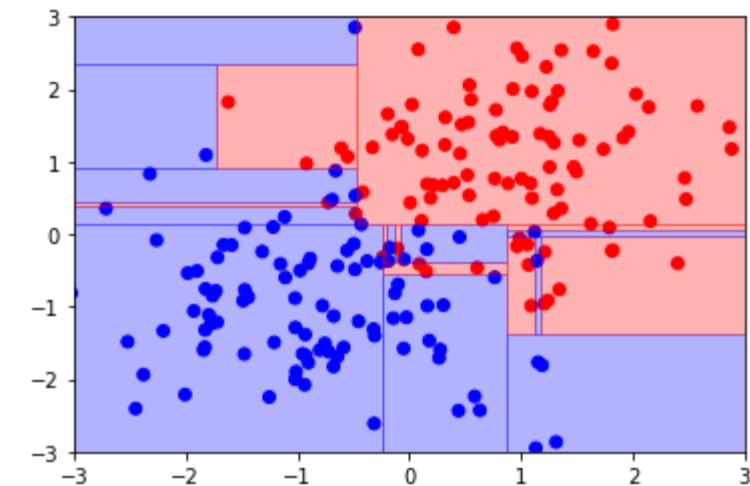


- The previous steps we followed are defined as the **ID3 algorithm**, which is a classic algorithm for building decision trees. The steps of the algorithm are as follows:
  - Calculate the entropy of every attribute of the data set  $S$ .
  - Partition ("split") the set  $S$  into subsets using the attribute for which the resulting entropy after splitting has maximum information gain.
  - Make the selected attribute an internal node and divide the subset of records that have the value of that attribute into smaller subsets.
  - Recurse on each subset until one of the following conditions holds:
    - Every element in the subset belongs to the same class.
    - There are no more attributes to be selected.
    - There are no examples in the subset.
- Other algorithms like the *C4.5* build on the *ID3* algorithm to handle continuous numerical features, missing features, and generate a smaller overall tree.

# Decision Tree: Decision Boundary



- The decision boundary in a decision tree represents the regions in the input feature space where the decision tree assigns a particular class label or output value.
- In other words, it partitions the feature space into regions, and each region is associated with a different class label or output value.
- In 2D, it can be visualized as a set of lines or curves that separate different classes.
- A very deep decision tree can impose problems as it could overfit on the training data and capture noise and fluctuations in the data.
  - It would also lead to an overly complex decision boundary that doesn't capture the real underlying patterns.
- Sometimes, we would want to impose external **stopping criteria** to enforce having a shallow tree which would generalize better to new data and capture the underlying patterns.



# Decision Tree: Stopping Criteria



- Stopping criteria are imposed on decision trees to prevent them from becoming too deep, a condition known as overfitting. Overfitting occurs when a decision tree learns the training data too well, capturing noise. Examples of different stopping criteria:

- Maximum Depth:**

- Stop growing the tree when a specified maximum depth is reached.

- Minimum Samples per Leaf:**

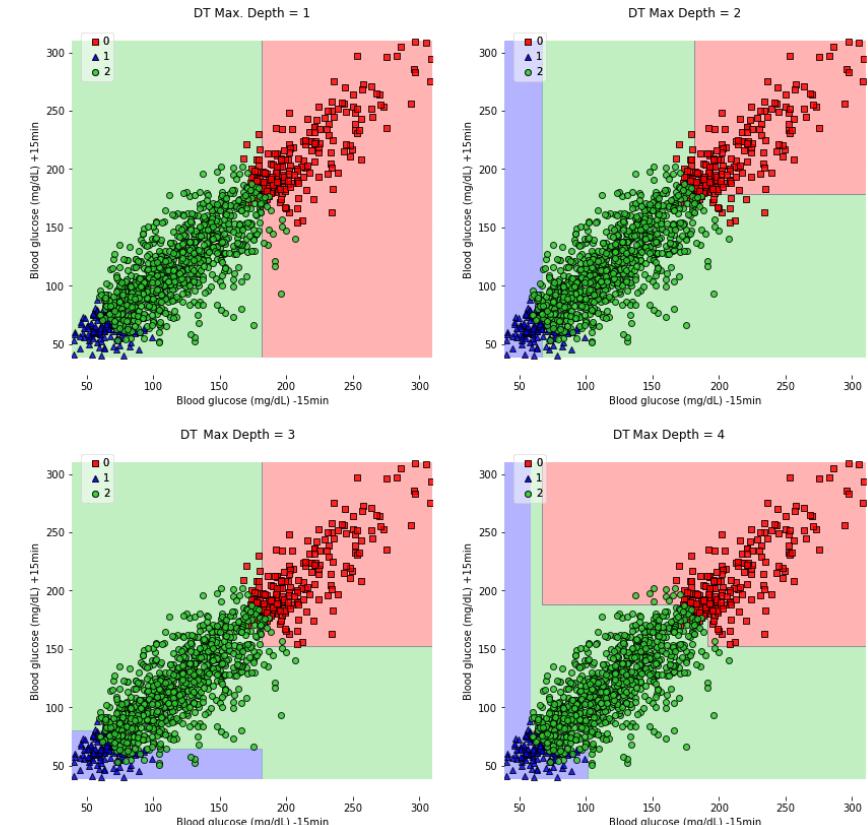
- Stop growing the tree when the number of samples in a leaf node falls below a specified threshold.

- Impurity Threshold:**

- Stop splitting nodes when the impurity (Information Gain) falls below a specified threshold.

- Pruning:**

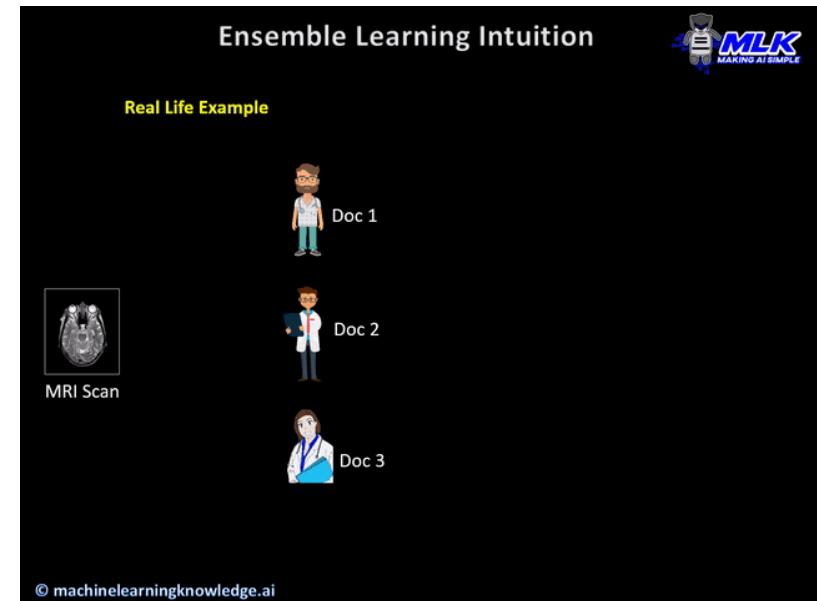
- Grow the tree fully, and then prune it by removing branches that do not significantly improve performance on a validation set.



# Ensemble Methods



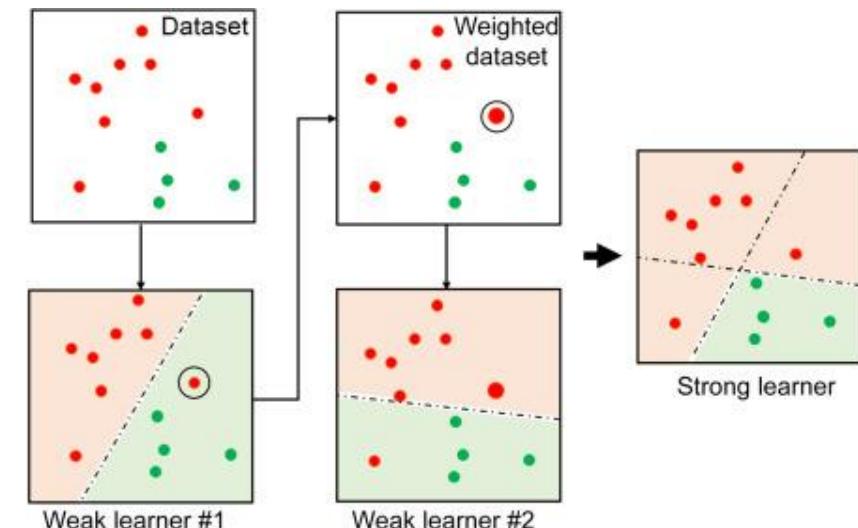
- Given a specific problem at hand, it is believed that that a committee of experts working together on a problem are more likely to solve it successfully than a single expert working alone.
- A **model ensemble** generate a set of models and then make predictions by aggregating the outputs of these models, rather than creating a single model only.
- Given a large population of independent models, an ensemble can be very accurate even if the individual models in the ensemble perform only marginally better than random guessing.
- The two famous types of ensemble algorithms are: **Boosting** and **Bagging**.



# Boosting Algorithms



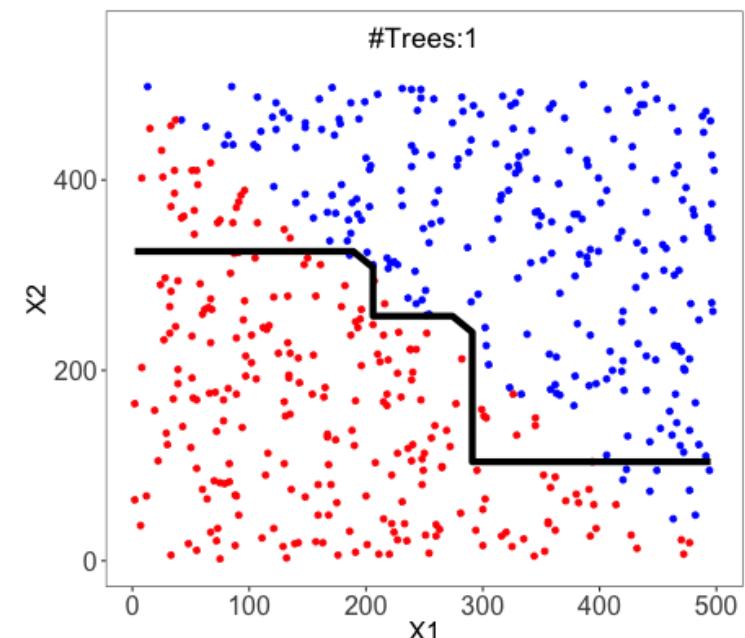
- **Boosting** is an ensemble learning technique that combines multiple weak learners to create a strong learner.
- It aims to improve predictive performance by sequentially training models, with each subsequent model giving more weight to instances that were misclassified by the previous models.
- Steps of a typical boosting algorithm are as follows:
  - Assign equal weights to all data points in the training set.
  - Train a weak learner (usually a simple model, e.g., a shallow decision tree) on the training data, considering the instance weights.
  - Compute the error of the weak learner, emphasizing misclassified instances.
  - Increase the weights of misclassified instances to make them more influential in the next iteration.
  - Train another weak learner on the updated dataset, giving higher importance to previously misclassified instances.
  - Repeat steps 3-5 for a predefined number of iterations or until a satisfactory performance is achieved.
  - Combine the predictions of all weak learners to produce the final strong model.
- Examples of Tree-based Boosting Models: AdaBoost / Gradient Boosting (GBM) / XGBoost / LightGBM



# Bagging Algorithms



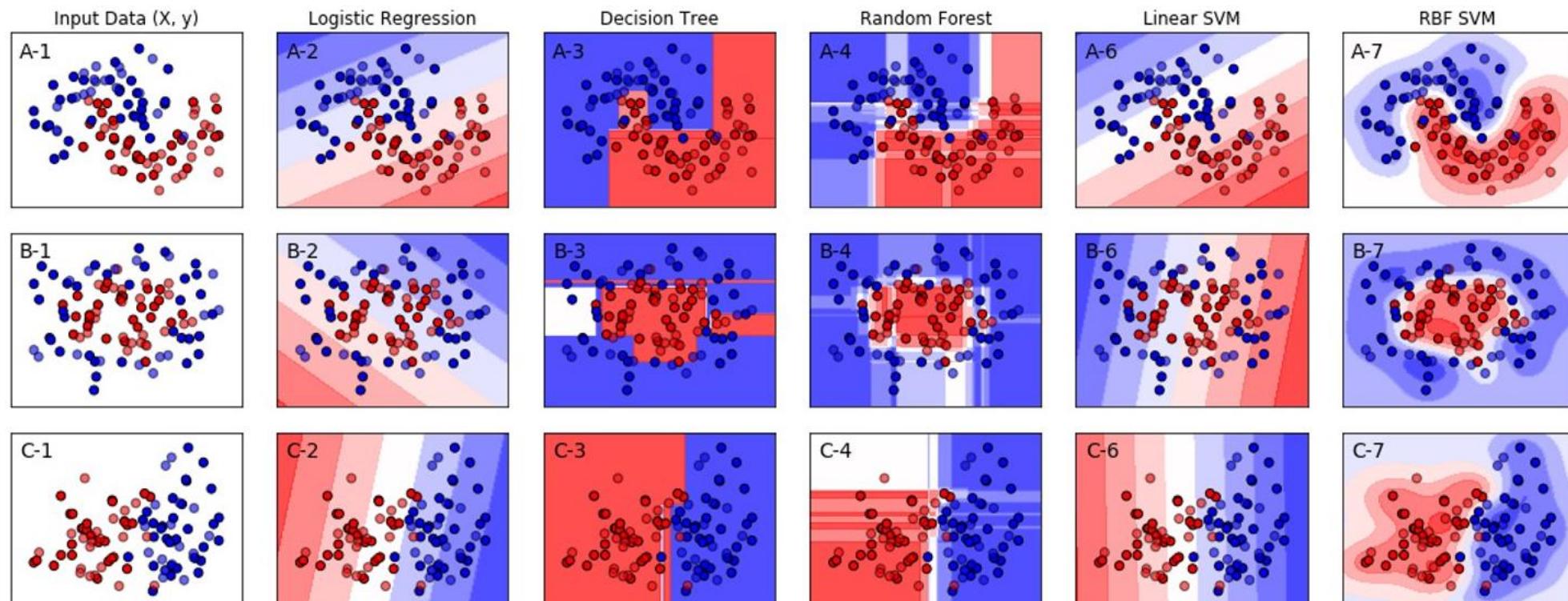
- **Bagging (Bootstrap Aggregating)** is an ensemble learning technique that combines multiple base learners.
- Each model in the ensemble is trained on a random sample of the dataset where, importantly, each random sample is the same size as the dataset and **sampling with replacement** is used.
- These type of methods are well-suited for Decision Trees since single trees are very sensitive to changes in the dataset. Therefore, having multiple trees trained on different samples of the dataset could result in a more powerful ensemble model.
- Steps of a Bagging approach are simple:
  - Generate multiple random samples with replacement from the training dataset.
  - Train a base learner (e.g., decision tree) on each bootstrap sample independently.
  - Combine predictions from all base models by voting for classification.
- The vanilla bagging approach mentioned above is called a “Random Forest” model.
- Extremely randomized trees (Extra Trees) take it a step further by also randomizing the feature to split on in every step. In other words, each weak learner (Decision Tree) doesn’t choose the best feature with highest Information Gain, but chooses it randomly.



# Comparison of Models So Far



## Comparing Classification Methods



Example adapted from Scikit-learn open-source developer guide, Code source: Gaël Varoquaux, Andreas Müller; 2018  
[https://scikit-learn.org/stable/auto\\_examples/classification/plot\\_classifier\\_comparison.html](https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)

# Extra Resources



- More information regarding the Decision Trees:
  - Chapter 4: John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. 2015. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press.

# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Classification

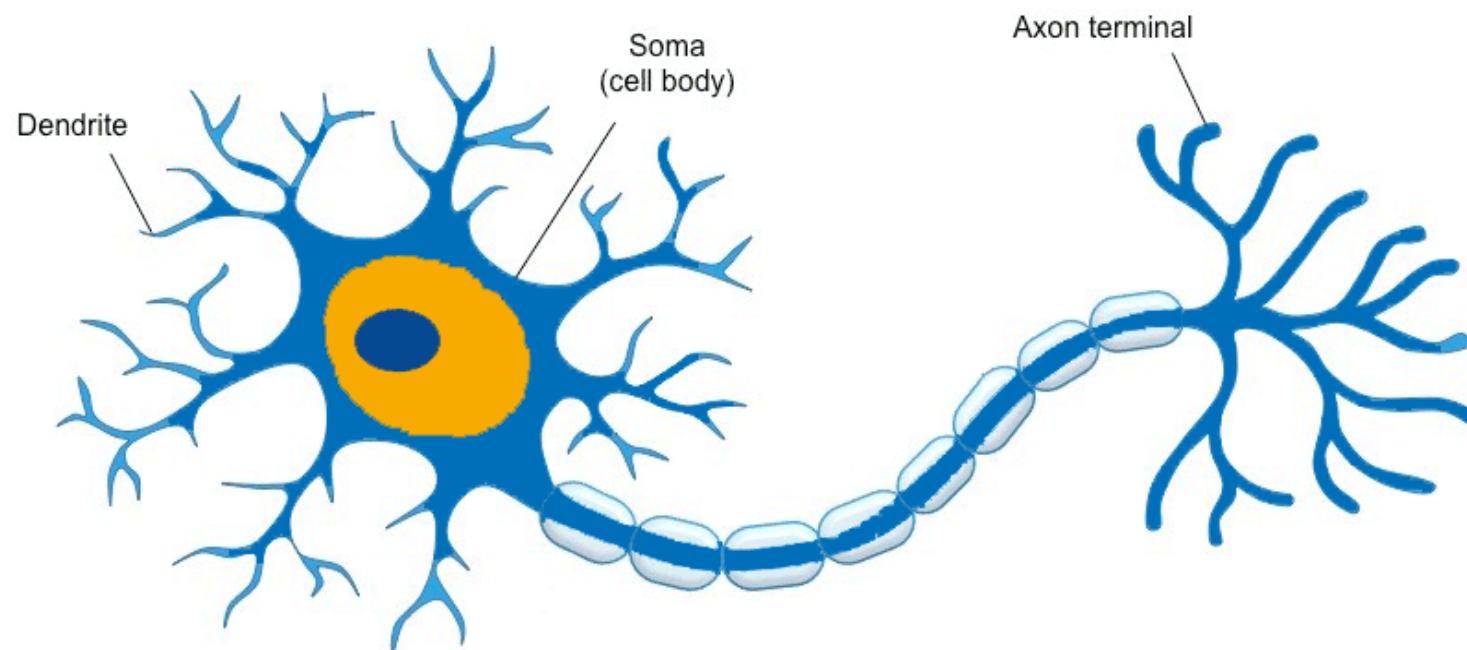
- Perceptron
  - Multi-Layered Perceptrons
  - Activation Functions
  - Backpropagation
  - Bias vs Variance
  - Solutions for Overfitting



# Biological Neuron



- A human brain has billions of neurons. Neurons are interconnected nerve cells in the human brain that are involved in processing and transmitting chemical and electrical signals.
- Dendrites receive input signals, soma cells processes them, and axon terminal produce the output signals.



# Artificial Neurons



- An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes **inputs**, **weights** them separately, **sums them up** and passes this sum through a **nonlinear function** to produce output.

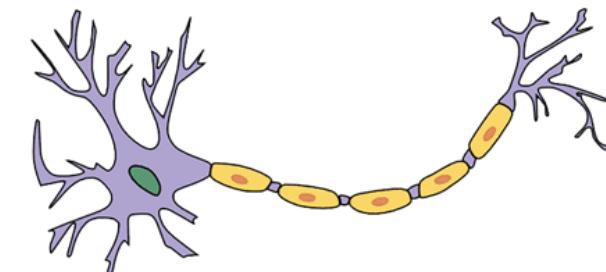


Fig: Biological Neuron

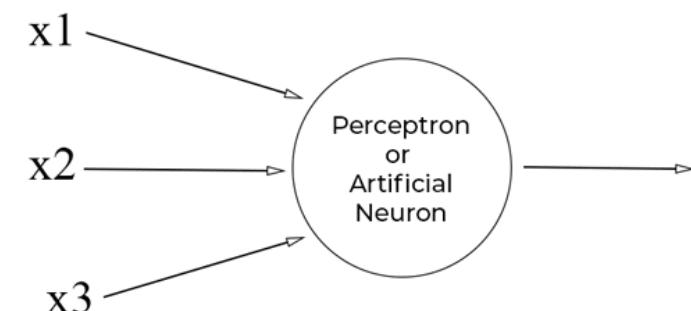
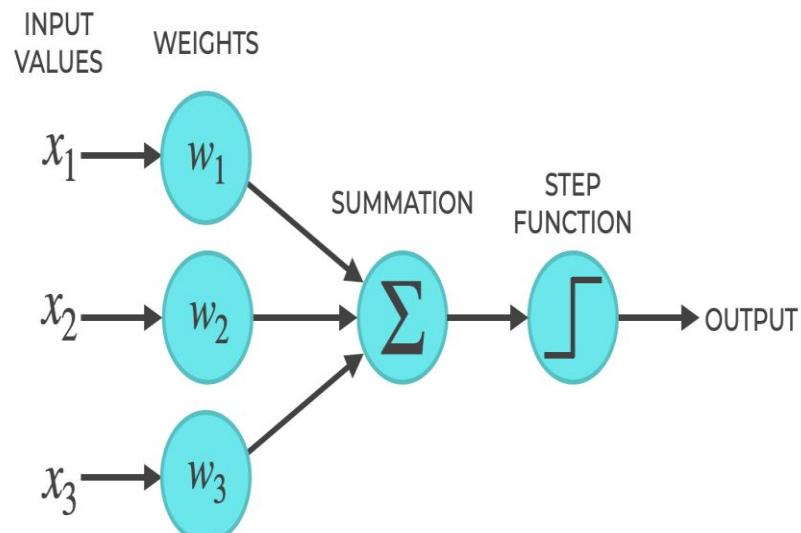


Fig: Artificial Neuron

# Perceptron



- The perceptron is an artificial neuron that carries out binary classification with supervised learning.
- The components of a perceptron are Input signals  $\vec{x}$ , Weights  $\vec{w}$ , Bias term  $\mathbf{b}$ , Step function  $\delta$ , and Output values. Mathematically:  
$$\text{Output} = \delta(\vec{w} \cdot \vec{x} + \mathbf{b})$$
- Does this look similar to a formulation we've seen before?



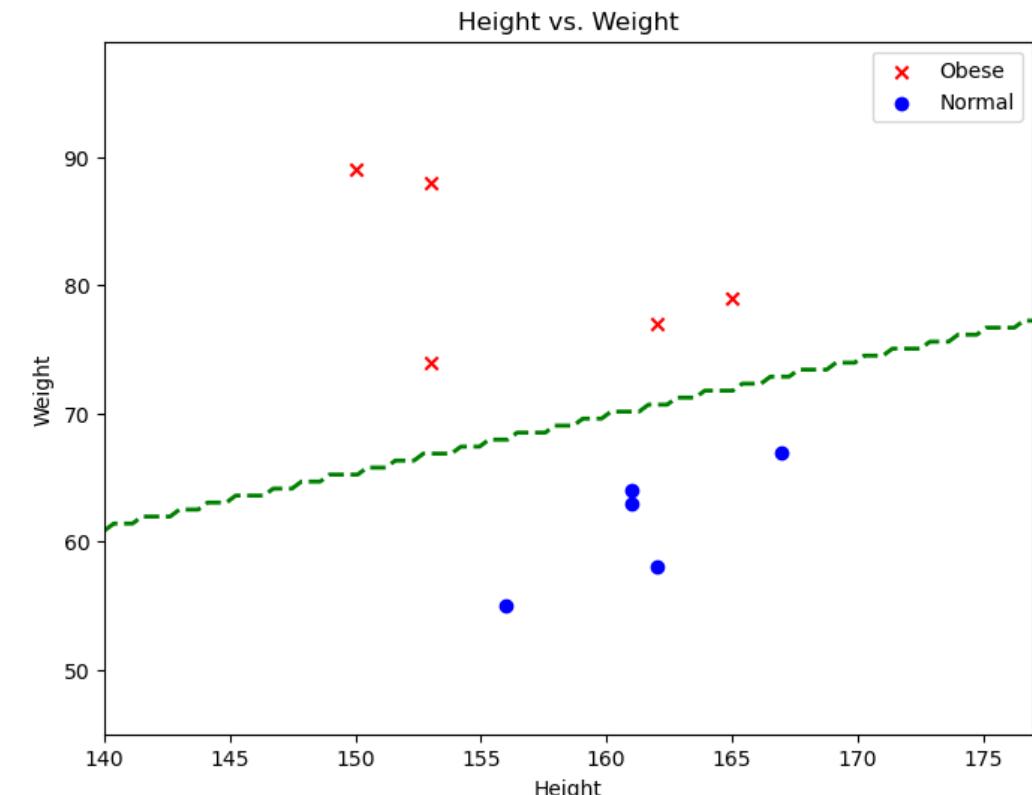
# Recall: Decision Boundary Definition



- The decision boundary in a binary classification problem is a hypersurface that separates the feature space into regions corresponding to different classes.
- The decision boundary only exists as a line or hyperplane if the classes are ***linearly separable***
- The equation of the linear decision boundary is as follows:

$$\hat{y} = \vec{w} \vec{x} + b$$

- $\vec{x}$  represents all input features.
- $\hat{y}$  represents estimated class a point belongs to.
- $y$  represents the label of the point  $\{-1, 1\}$
- $\vec{w}, b$  are learnable parameters to be estimated



$$\hat{y} = \vec{w} \vec{x} + b$$

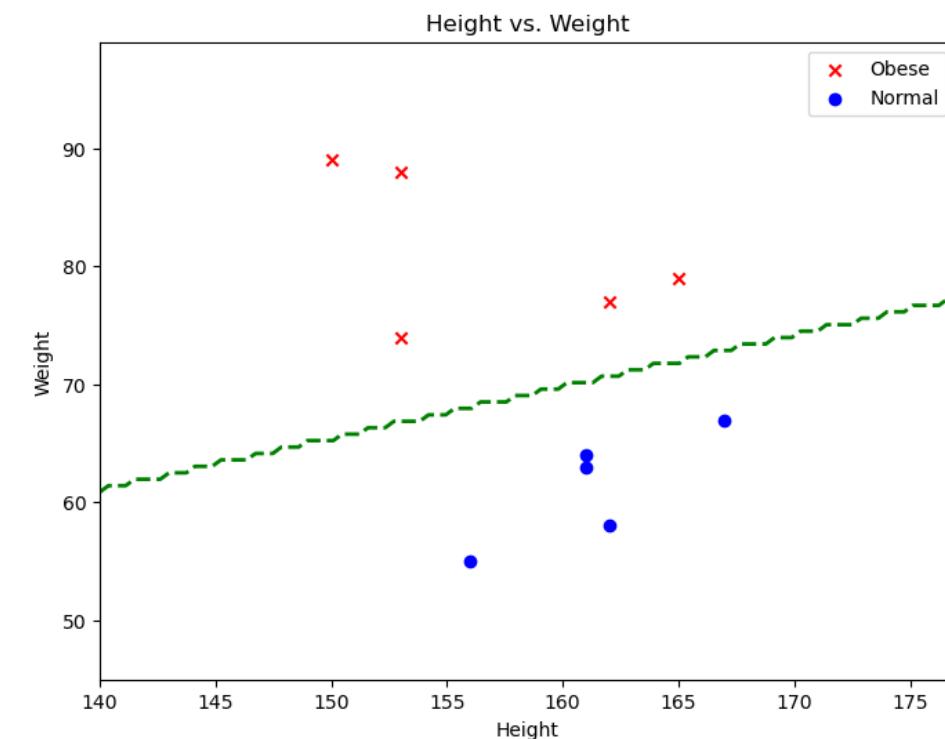
# Perceptron



- The perceptron learns a *linear* decision boundary to separate between classes.

$$\hat{y} = \vec{w} \cdot \vec{x} + b$$

- The perceptron assumes that each point in the classes have labels  $y \in \{1, -1\}$
- As we know,  $\hat{y}$  is a continuous value where  $\hat{y} > 0$  if above boundary and  $\hat{y} < 0$  if below boundary.
- We need a way to transform  $\hat{y}$  to binary values  $\in \{1, -1\}$ .



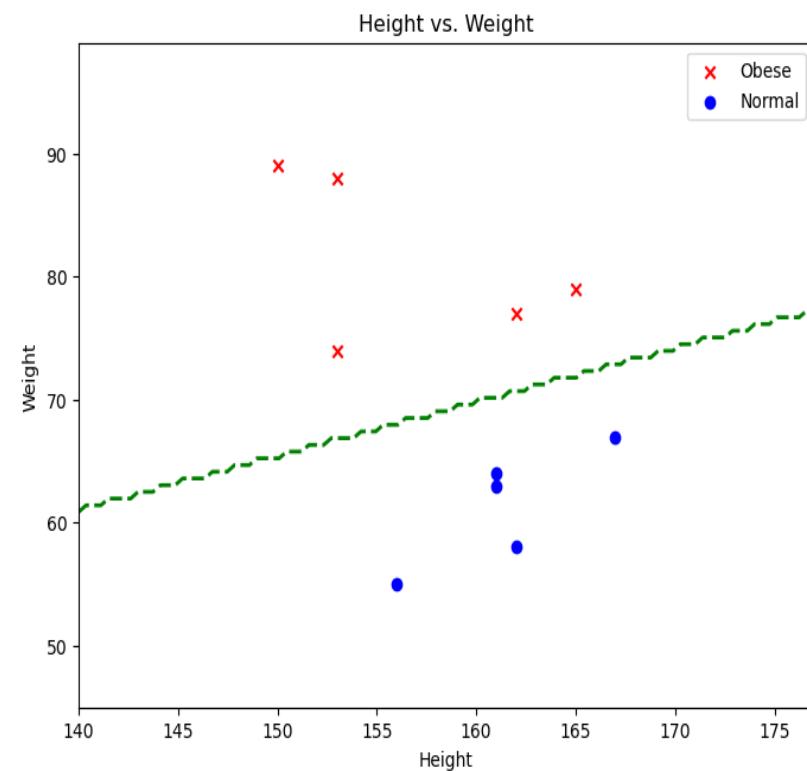
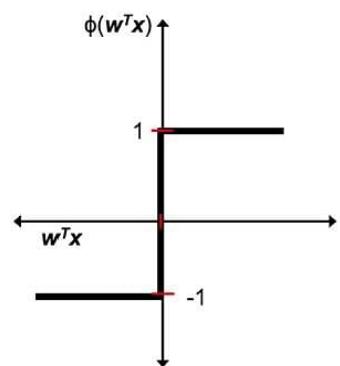
# Perceptron



- To achieve this, we need a way to transform  $\hat{y}$  to binary values  $\in \{1, -1\}$ .
- We utilize a simple sign function formulated as follows:

$$\phi(\hat{y}) = \begin{cases} 1, & \text{if } \hat{y} \geq 0 \\ -1, & \text{if } \hat{y} < 0 \end{cases}$$

- For all correctly classified points  
 $\rightarrow y_i \phi(\hat{y}_i) = 1 \rightarrow y_i \hat{y}_i \geq 0$
- For all incorrectly classified points  
 $\rightarrow y_i \phi(\hat{y}_i) = -1 \rightarrow y_i \hat{y}_i < 0$



# Perceptron



- For all incorrectly classified points  $\rightarrow y_i \phi(\hat{y}_i) = -1 \rightarrow y_i \hat{y}_i < 0$
- Target: Classify ***all*** points ***correctly***.
- To achieve this, we will formulate an error function only for incorrectly classified points  $\vec{x}_i$ , such as:

$$L(\vec{w}, \mathbf{b}) = -y_i \hat{y}_i = -y_i (\vec{w} \cdot \vec{x}_i + b)$$

- This error function quantifies how far is the misclassified point from the decision boundary.
  - The negative sign ensures that the Loss function is always positive.
- This loss function can be minimized using Gradient Descent to reach an optimal decision boundary.

# Perceptron



$$L(\vec{w}, \mathbf{b}) = -y_i \hat{y}_i = -y_i (\vec{w} \cdot \vec{x}_i + b)$$

- To get the values of  $\vec{w}, \mathbf{b}$  at which  $L(\vec{w}, \mathbf{b})$  is minimum using Gradient Descent, we need to compute  $\frac{d(L)}{d\vec{w}}$  and  $\frac{d(L)}{db}$ .
- $\frac{d(L)}{d\vec{w}} = -y_i \vec{x}_i, \quad \frac{d(L)}{db} = -y_i$
- Gradient Descent formulation for step size  $\delta$  and one misclassified point  $\vec{x}_i$  :
  - $\vec{w}_{t+1} = \vec{w}_t - \delta \frac{dL}{d\vec{w}} = \vec{w}_t + \delta y_i \vec{x}_i$
  - $b_{t+1} = b_t - \delta \frac{dL}{db} = b_t + \delta y_i$

# Perceptron

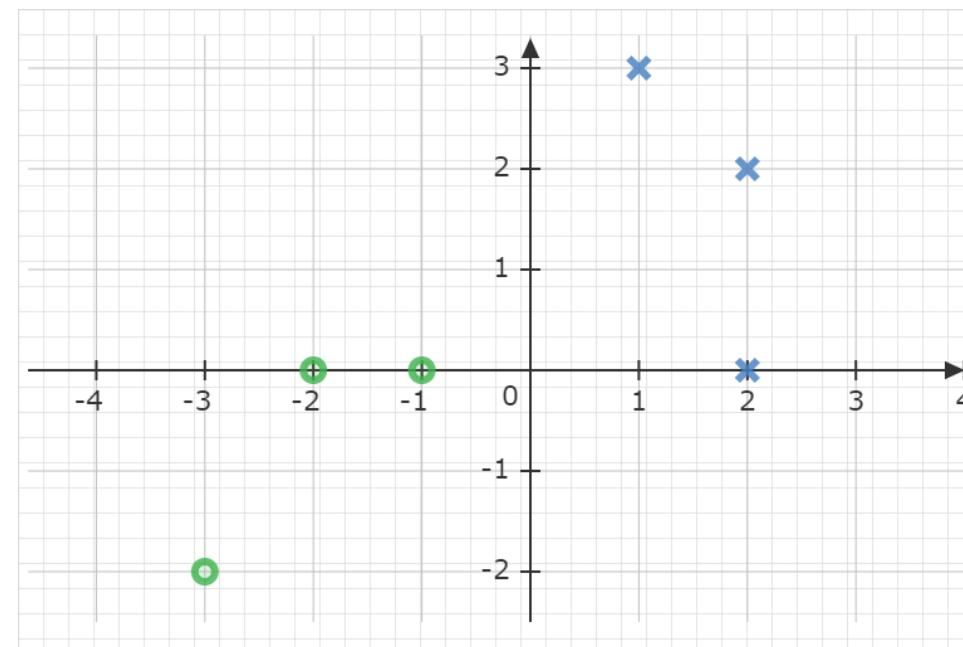


- The steps of the perceptron learning algorithm are as follows:
  - Initialize  $\vec{w}_0, b_0$  to random initial values
  - Use the rule  $y_i \hat{y}_i < 0$  to identify points that are misclassified given the current weight and bias.
  - Choose one misclassified point  $x_i$ .
  - Update the weight and bias using the following update rules:
    - $w_{t+1} = w_t - \delta \frac{dL}{dw} = \vec{w}_t + \delta y_i x_i$
    - $b_{t+1} = b_t - \delta \frac{dL}{db} = b_t + \delta y_i$
  - Repeat Steps 2-4 until all points are correctly classified.
- The previous algorithm is guaranteed to converge to a solution if all the points are **strictly linearly separable**.

# Perceptron: Example



- Given the following dataset, we want to use the Perceptron algorithm to find the decision boundary.
- Blue points have  $y_i = 1$ .
- Green points have  $y_i = -1$ .



# Perceptron: Example



- Start by initializing the weight and bias in step 0:

- $\overrightarrow{w_0} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, b_0 = -4$

- Check which points are misclassified:

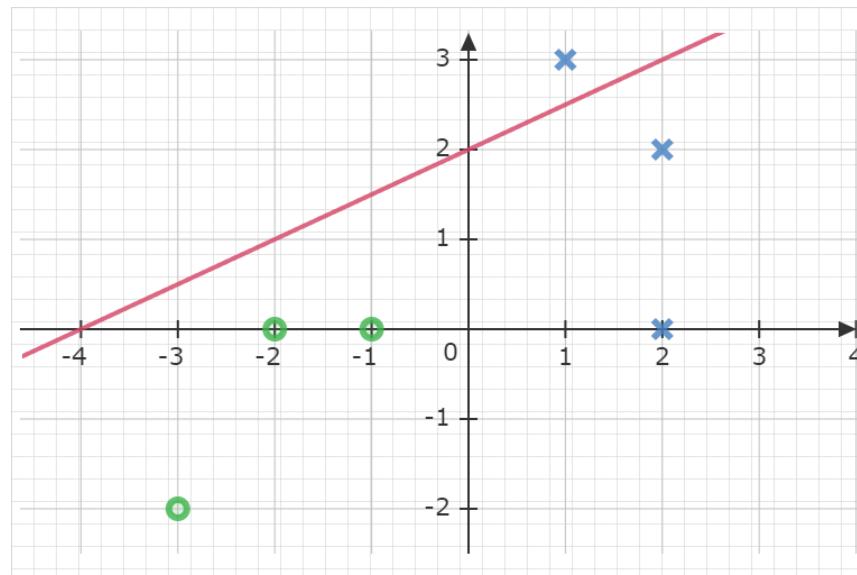
- $\begin{bmatrix} 1 \\ 3 \end{bmatrix} \rightarrow +1((-1 * 1) + (3 * 2) - 4) = 1 > 0$

- $\begin{bmatrix} 2 \\ 2 \end{bmatrix} \rightarrow +1((-1 * 2) + (2 * 2) - 4) = -2 < 0$

- Update weights using point  $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$  using  $\delta = 1$ :

- $\overrightarrow{w_1} = \begin{bmatrix} -1 \\ 2 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$

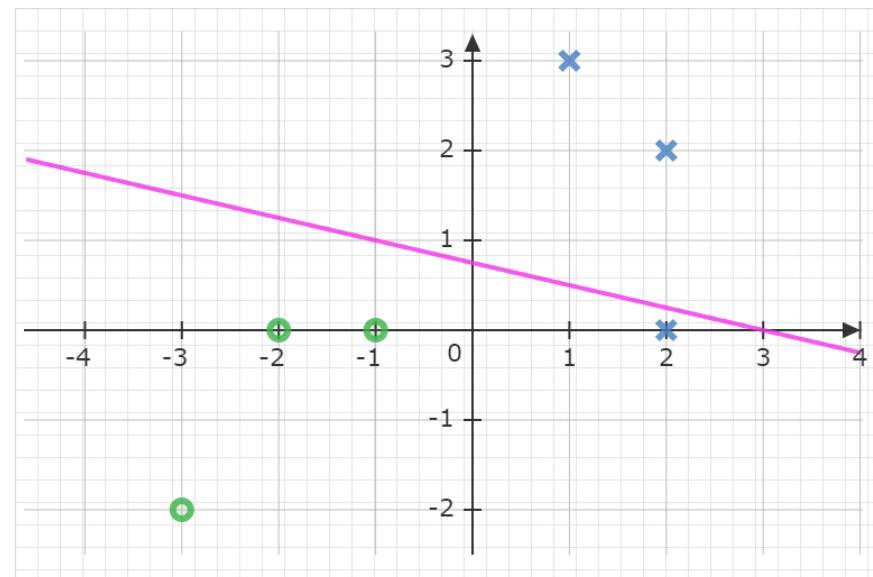
- $b_1 = -4 + 1 = -3$



# Perceptron: Example



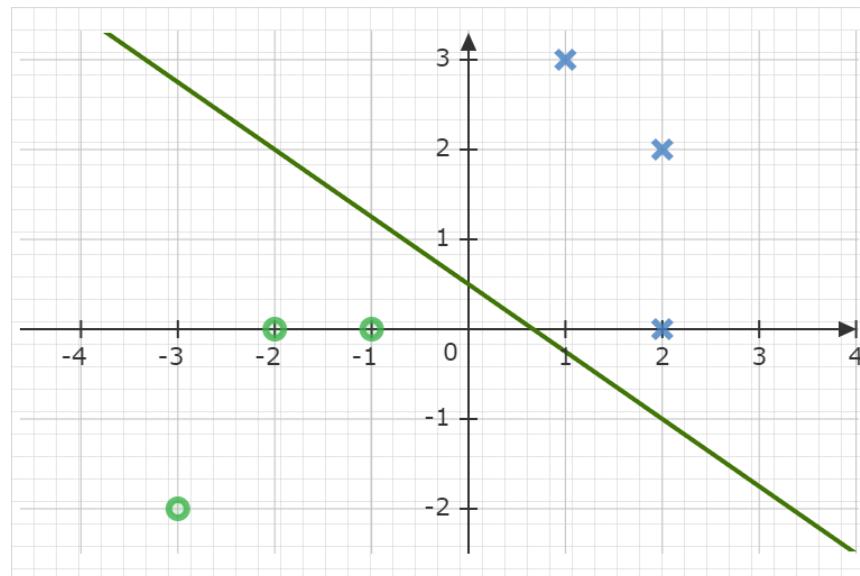
- $\overrightarrow{w_1} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, b_1 = -3$
- Check which points are misclassified:
  - $\begin{bmatrix} 2 \\ 0 \end{bmatrix} \rightarrow +1((2 * 1) + (0 * 4) - 3) = -1 < 0$
- Update weights using point  $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$  using  $\delta = 1$ :
  - $\overrightarrow{w_2} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$
  - $b_2 = -3 + 1 = -2$



# Perceptron: Example



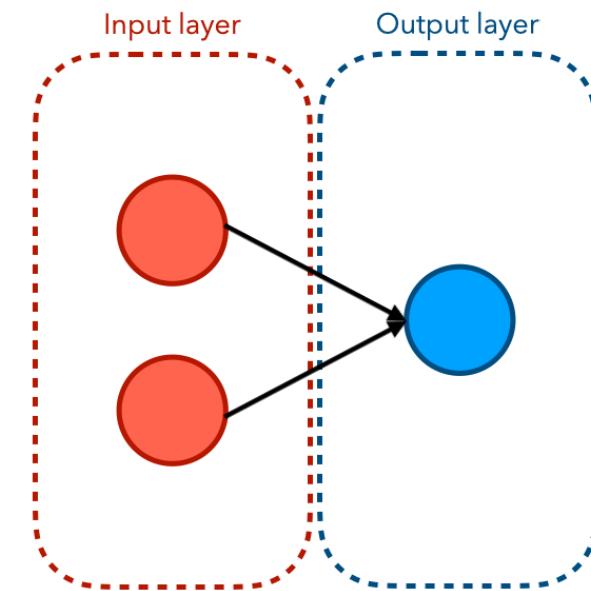
- $\overrightarrow{w}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, b_2 = -2$
- Check which points are misclassified:
  - All points are correctly classified with  $y_i \hat{y}_i \geq 0$



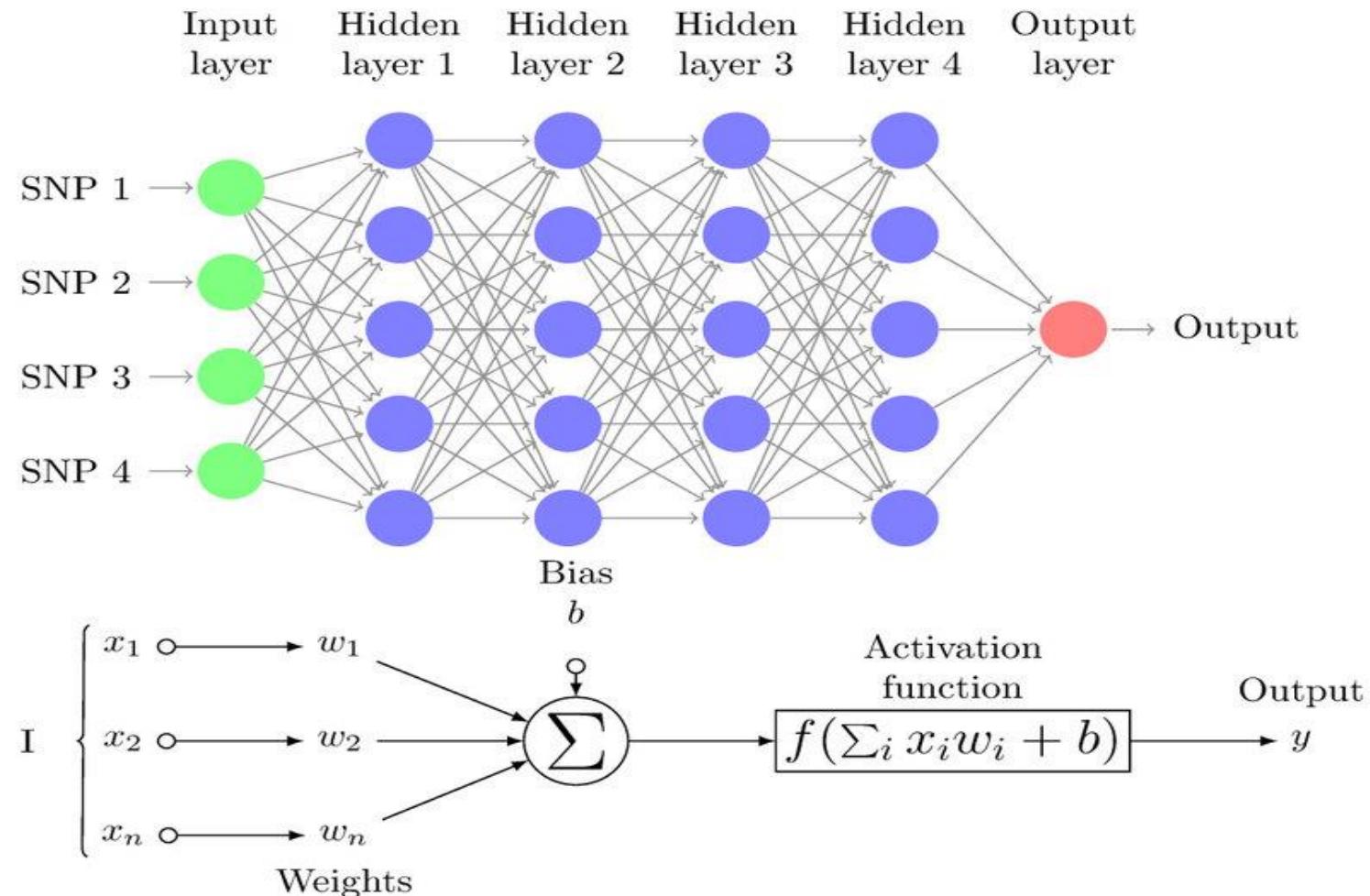
# Multi-Layer Perceptron



- The perceptron is only capable of generating a linear decision boundary and cannot be used to solve complex classification problems.
- To have a more powerful classification algorithm, we can stack a series of perceptrons on each other creating a **multi-layer perceptron**.
- Each hidden layer consists of a series of perceptrons (neurons) receiving input from previous layer and delivering output to the next layer.



# Multi-layer Perceptron



# Multi-layer Perceptron



- Each neuron (perceptron)  $i$  in the hidden layer  $l$  with  $N^l$  neurons must do the following operations:
  - Multiply the outputs from the previous layer by their corresponding weights  $\rightarrow \mathbf{w}_i^l * \mathbf{o}_i^{l-1}$
  - Sum the multiplications between the weights and the outputs and add the bias term  $\rightarrow \left[ \sum_{i=1}^{N^l} \mathbf{w}_i^l * \mathbf{o}_i^{l-1} \right] + \mathbf{b}_i^l$
  - Pass the summation to an activation function  $f$  to produce the output of the current neuron  $\rightarrow \mathbf{o}_i^l = f(\left[ \sum_{i=1}^{N^l} \mathbf{w}_i^l * \mathbf{o}_i^{l-1} \right] + \mathbf{b}_i^l)$
- The activation function is the most important component of the MLP. Without an activation function, the MLP will still only be able to learn a linear decision boundary similar to a single perceptron.
- There are different types of activation functions that could be used in an MLP.

# Types of Activation Functions

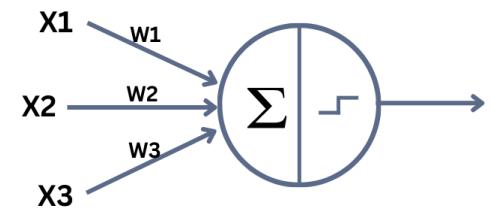


| Name   | Plot | Equation   | Derivative  |
|--|------|--|---|
| Identity   |      | $f(x) = x$   | $f'(x) = 1$   |
| Binary step  |      | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$               | $f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$             |
| Logistic (a. k. a Soft step)                             |      | $f(x) = \frac{1}{1 + e^{-x}}$  | $f'(x) = f(x)(1 - f(x))$  |
| TanH   |      | $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$  | $f'(x) = 1 - f(x)^2$  |
| ArcTan   |      | $f(x) = \tan^{-1}(x)$  | $f'(x) = \frac{1}{x^2 + 1}$   |
| Rectified Linear Unit (ReLU)                             |      | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$               | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$             |
| Parameteric Rectified Linear Unit (PReLU) <sup>[2]</sup> |      | $f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$        | $f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$        |
| Exponential Linear Unit (ELU) <sup>[3]</sup>             |      | $f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| SoftPlus   |      | $f(x) = \log_e(1 + e^x)$   | $f'(x) = \frac{1}{1 + e^{-x}}$  |

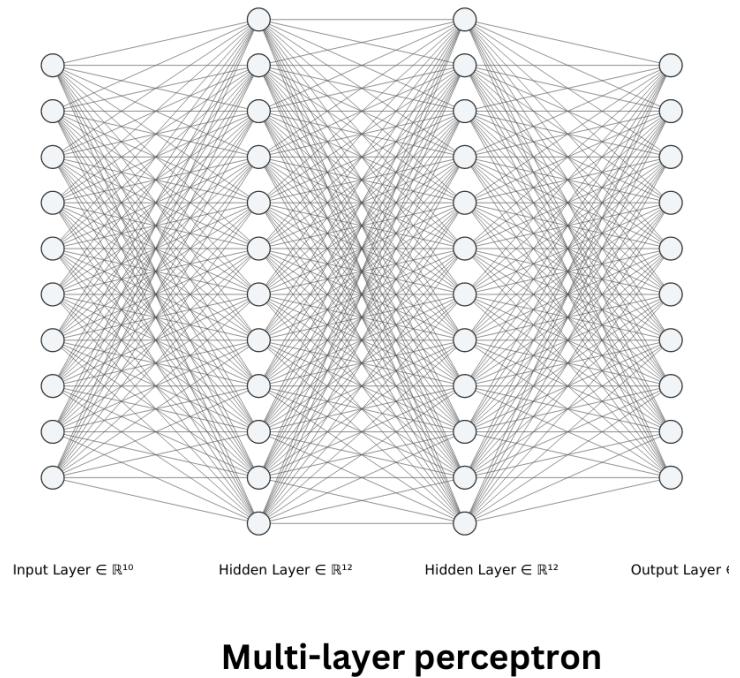
# Multi-layer Perceptron: Learning Approach



- In a single-layer perceptron, we were able to learn the optimal  $\vec{w}$ ,  $\mathbf{b}$  parameters by formulating  $L(\vec{w}, \mathbf{b})$ , computing  $\frac{d(L)}{d\vec{w}}$  and  $\frac{d(L)}{d\mathbf{b}}$  and utilizing Gradient Descent.
- In an MLP, could the same exact approach be used to reach the optimal parameters?
  - It's more complicated since we have different parameters (weight and bias) for each neuron and each neuron passes its output to the following layer.
  - $L(\vec{w}, \mathbf{b})$  is calculated after the output layer, we need a way to calculate  $\frac{d(L)}{d\vec{w}}$  and  $\frac{d(L)}{d\mathbf{b}}$  for each neuron in the previous layers.



Single-layer perceptron

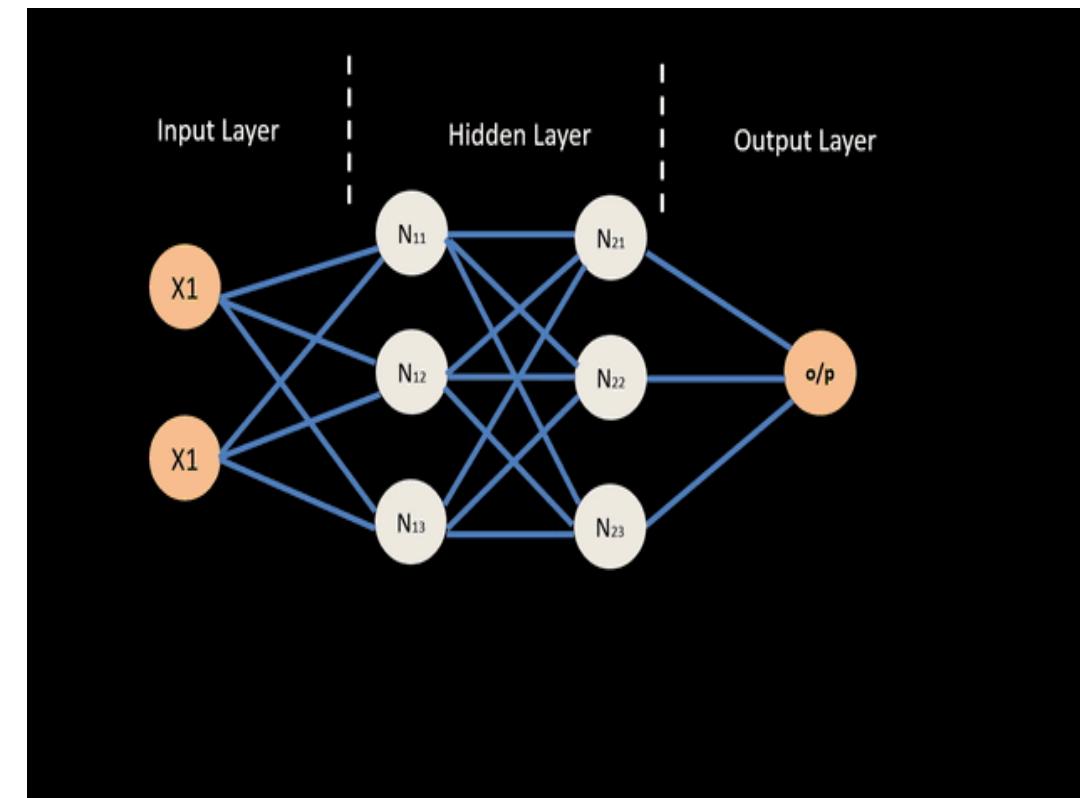


Multi-layer perceptron

# Backpropagation



- Backpropagation is a gradient-based optimization algorithm that adjusts the weights of the MLP to minimize the difference between the predicted and actual outputs.
  - Forward Pass:
    - Input data is passed through the network layer by layer
    - The final output is compared to the target output.
    - The error (the difference between the predicted and actual outputs) is calculated.
  - Backward Pass:
    - The gradient of the error with respect to the weights is computed layer by layer starting from the output layer and moving backward towards the input layer.
    - The computed gradients are then used to update the weights in the direction that minimizes the error.
    - *But how are we able to propagate the gradient of the error to all the previous layers? → Chain Rule*



# Backpropagation: Chain Rule



- Assume we have an MLP with 1 hidden layer and 1 output layer:



- To calculate derivative of Loss with respect to the first layer:

$$\frac{d(Loss)}{d(Layer_1)} = \frac{d(Loss)}{d(Output)} * \frac{d(Output)}{d(Layer_1)} \rightarrow \textbf{Chain Rule}$$

- The chain rule is the core rule used in backpropagation. We use it to recursively calculate the derivative of the loss starting from the output layer and going backwards until the input layer.

# Recall: MLP Steps

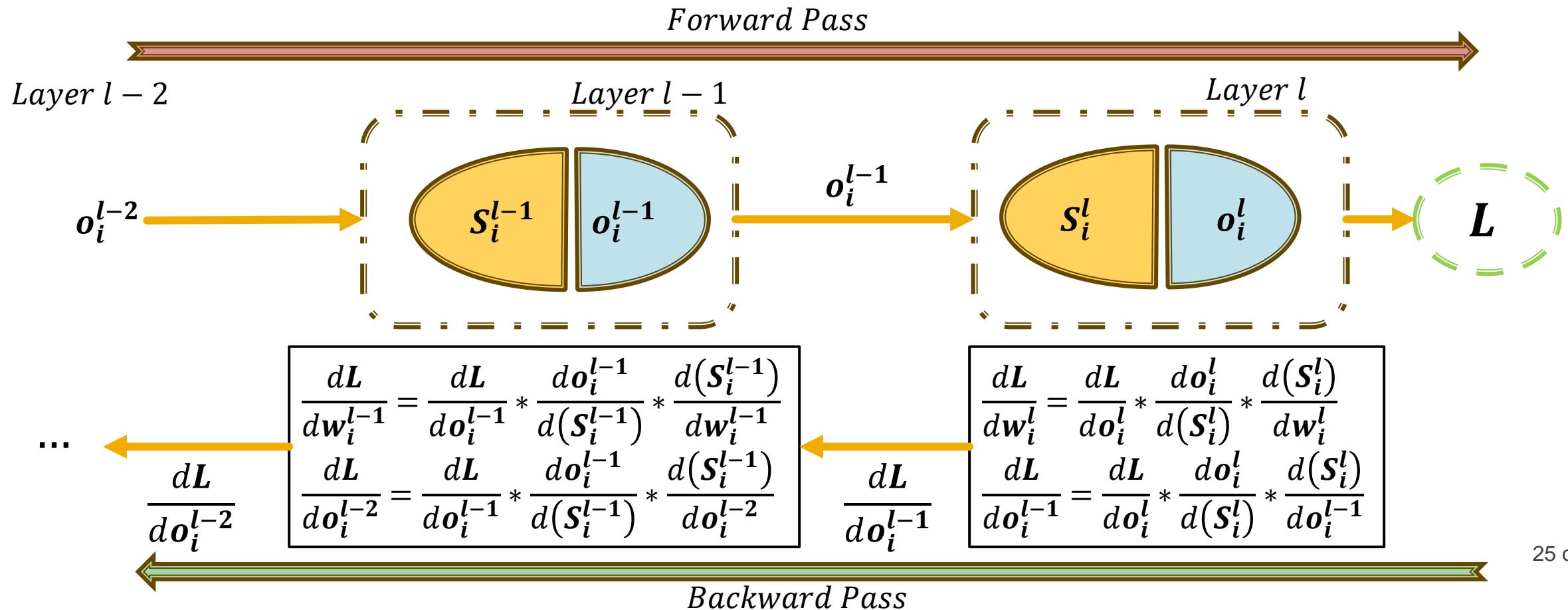


- Each neuron (perceptron)  $i$  in the hidden layer  $l$  with  $N^l$  neurons must do the following operations:
  - Multiply the outputs from the previous layer by their corresponding weights  $\rightarrow w_i^l * o_i^{l-1}$
  - Sum the multiplications between the weights and the outputs and add the bias term  $\rightarrow \left[ \sum_{i=1}^{N^l} w_i^l * o_i^{l-1} \right] + b_i^l$
  - Pass the summation to an activation function  $f$  to produce the output of the current neuron  $\rightarrow o_i^l = f(\left[ \sum_{i=1}^{N^l} w_i^l * o_i^{l-1} \right] + b_i^l)$

# Backpropagation



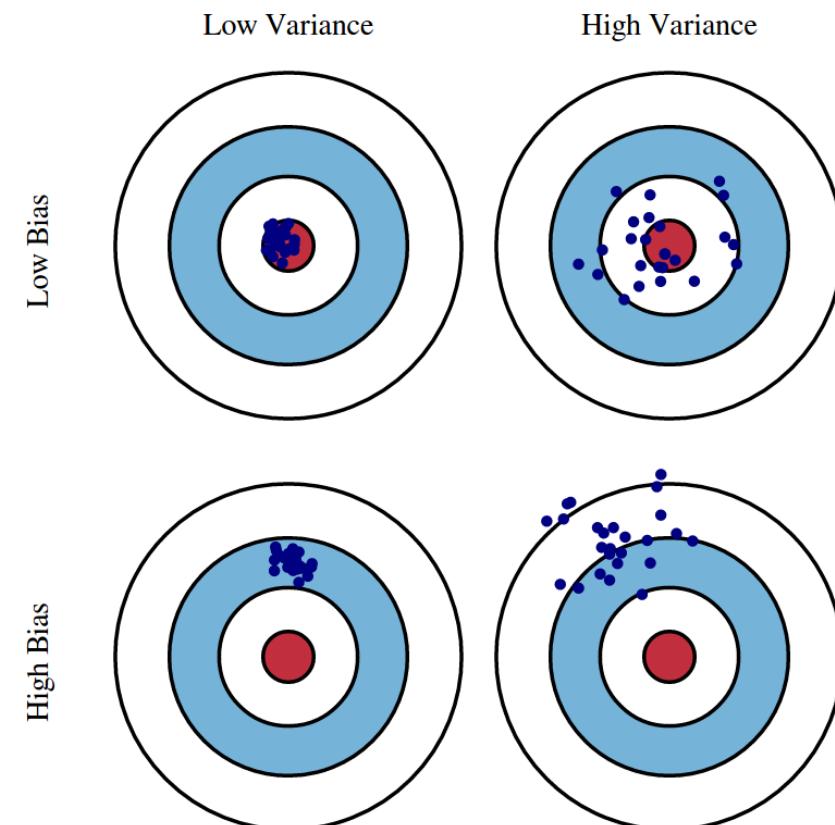
- $$S_i^l = \left[ \sum_{i=1}^{N^l} w_i^l * o_i^{l-1} \right] + b_i^l \quad \rightarrow \quad o_i^l = f(S_i^l)$$



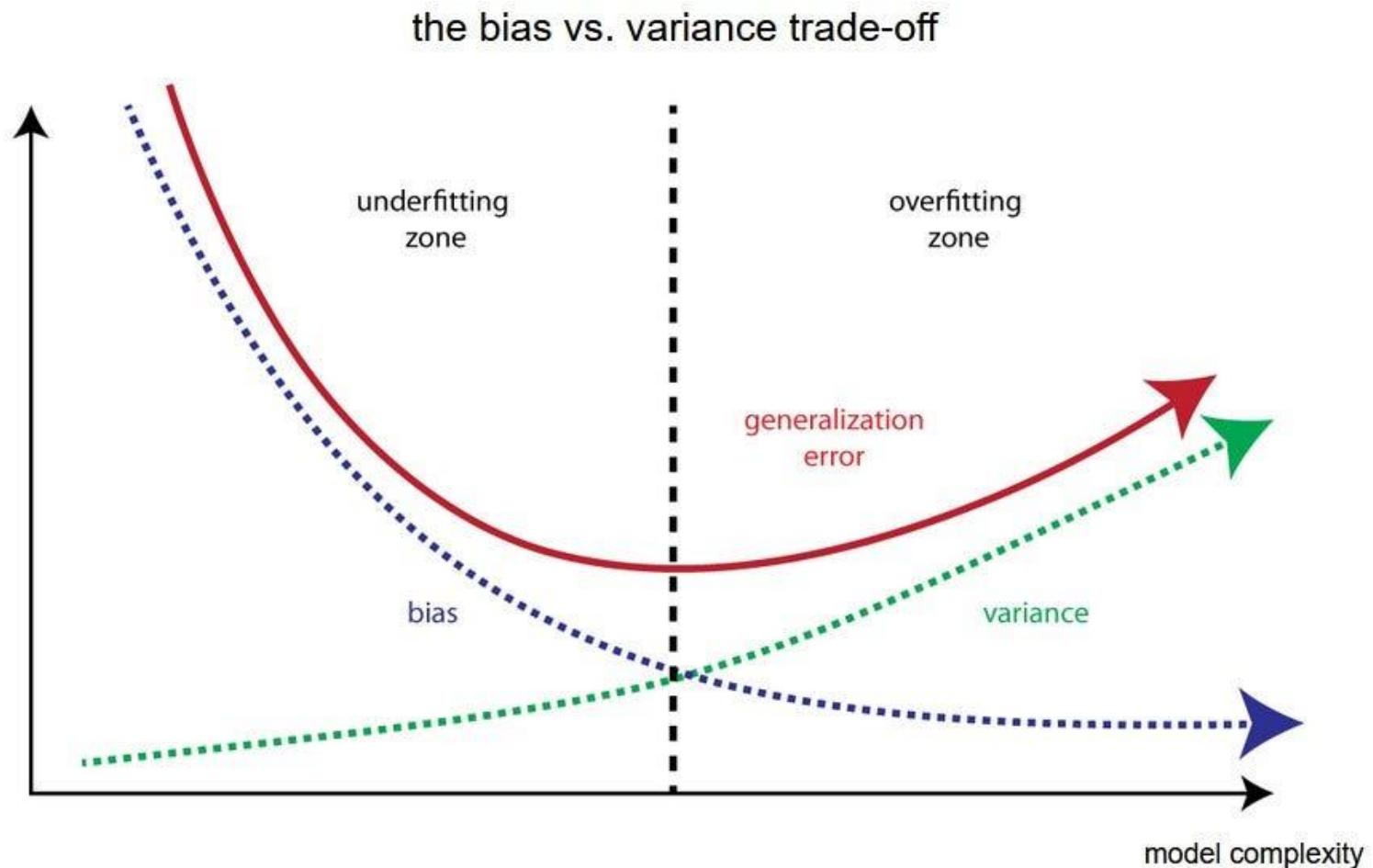
# Bias vs Variance Tradeoff



- The bias-variance tradeoff is a fundamental concept in machine learning that helps to understand the tradeoff between the simplicity and flexibility of a model.
- **Bias:** Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.
  - High Bias: Occurs when the model is too simple for the problem leading to incorrect predictions → Underfitting
  - Low Bias: Occurs when the model is sufficiently complex to generate correct predictions
- **Variance:** Refers to the model's sensitivity to the specific training data on which it was trained.
  - High Variance: Occurs when the model is too complex where it captures noise in the training data → Overfitting
  - Low Variance: Occurs when the model correctly captures the underlying patterns in the training set and generalizes well to new test data.
- An optimal model will have low bias and low variance.



# Bias vs Variance Tradeoff



# Ways to reduce overfitting



- Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations
- To reduce the overfitting, a general rule of thumb is to try to decrease the model complexity, but not too much to avoid underfitting.
- Specifically, techniques that can be used include:
  - **Cross-Validation:** Assess the model's performance on multiple subsets of the data to ensure good generalization to different portions of the dataset.
  - **Increase Dataset size:** A larger dataset can provide more diverse examples and help the model generalize better.
  - **Early Stopping:** Stop training once the performance on the validation set starts to degrade, preventing the model from overfitting the training data.
  - **Dropout:** random neurons are "dropped out" (ignored) during each training iteration. This helps prevent co-adaptation of neurons and reduces overfitting.
  - **Data Augmentation:** Increase the size of the training dataset by applying various transformations to the existing data
  - **Regularization?**

# Regularization

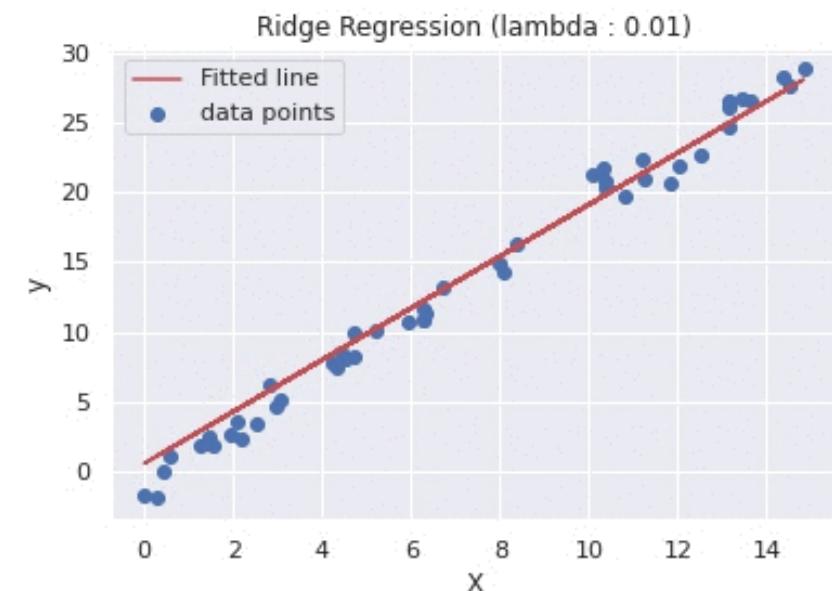


- The intuition is that one way to decrease the overall model complexity is to reduce the magnitude of the learned weights  $\vec{w}$ . This way we won't be assigning very high weights to certain features over the others → Simpler model

- This could be achieved by adding a regularization term to the loss function of any machine learning algorithm, such that,

$$L_{reg}(\vec{w}, b) = L(\vec{w}, b) + [\beta * \text{Regularization Term}]$$

- This will allow the model to not only prioritize minimizing the cost term, but also minimizing the magnitude of the weights (controlled by the hyperparameter  $\beta$ )
- There are different types of regularization terms that could be used.



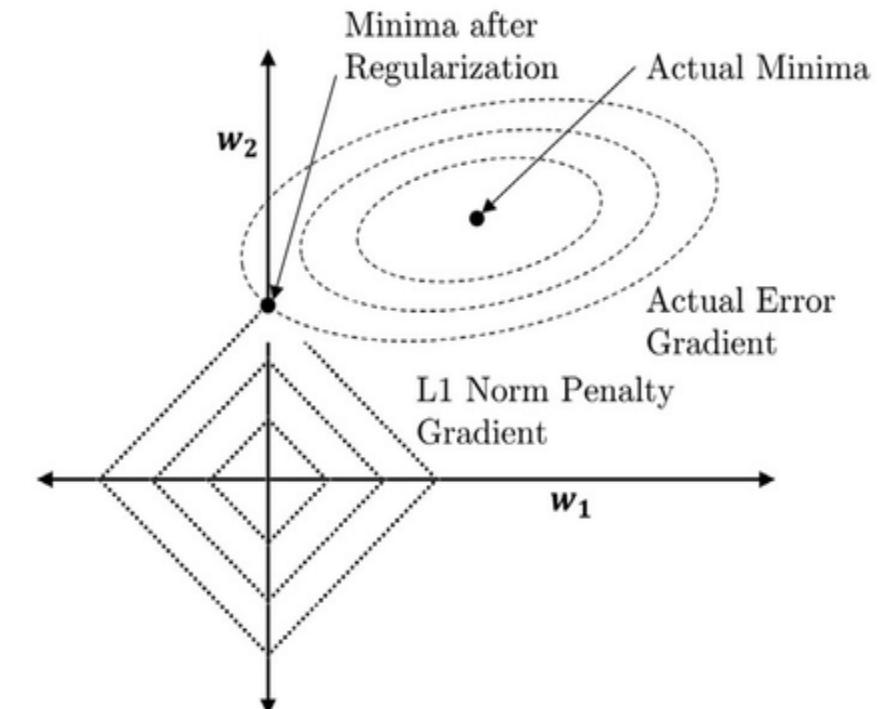
# L<sub>1</sub> Regularization (Lasso)



- L<sub>1</sub> regularization introduces a penalty term proportional to the absolute values of the model parameters.
- Mathematically,

$$L_{reg}(\vec{w}, b) = L(\vec{w}, b) + \left[ \beta * \sum_p |\vec{w}_p| \right]$$

- This encourages the model to prefer a sparse set of features, effectively driving some of them to exactly zero.
- Not only does it help prevent overfitting but also performs feature selection by excluding less relevant features.



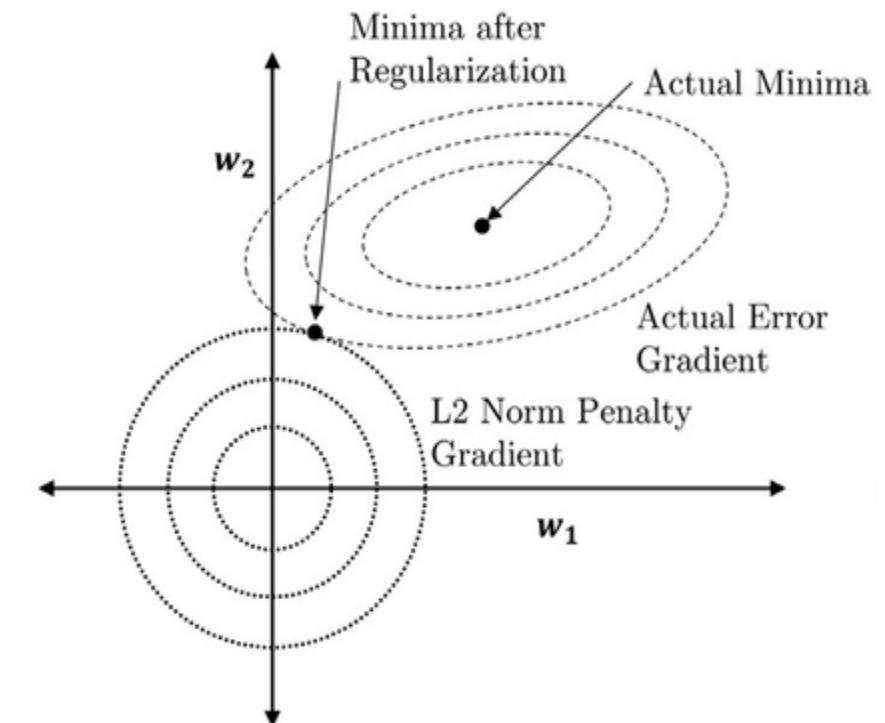
# L<sub>2</sub> Regularization (Ridge)



- L<sub>2</sub> regularization introduces a penalty term proportional to the square of the model parameters.
- Mathematically,

$$L_{reg}(\vec{w}, b) = L(\vec{w}, b) + \left[ \beta * \sum_p (\vec{w}_p)^2 \right]$$

- This discourages the weights from becoming too large, preventing individual features from dominating the model.
- It distributes the importance of features more evenly, leading to a smoother model.



# Extra Resources



- More information regarding the Bias and Variance:
  - <https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html>

# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



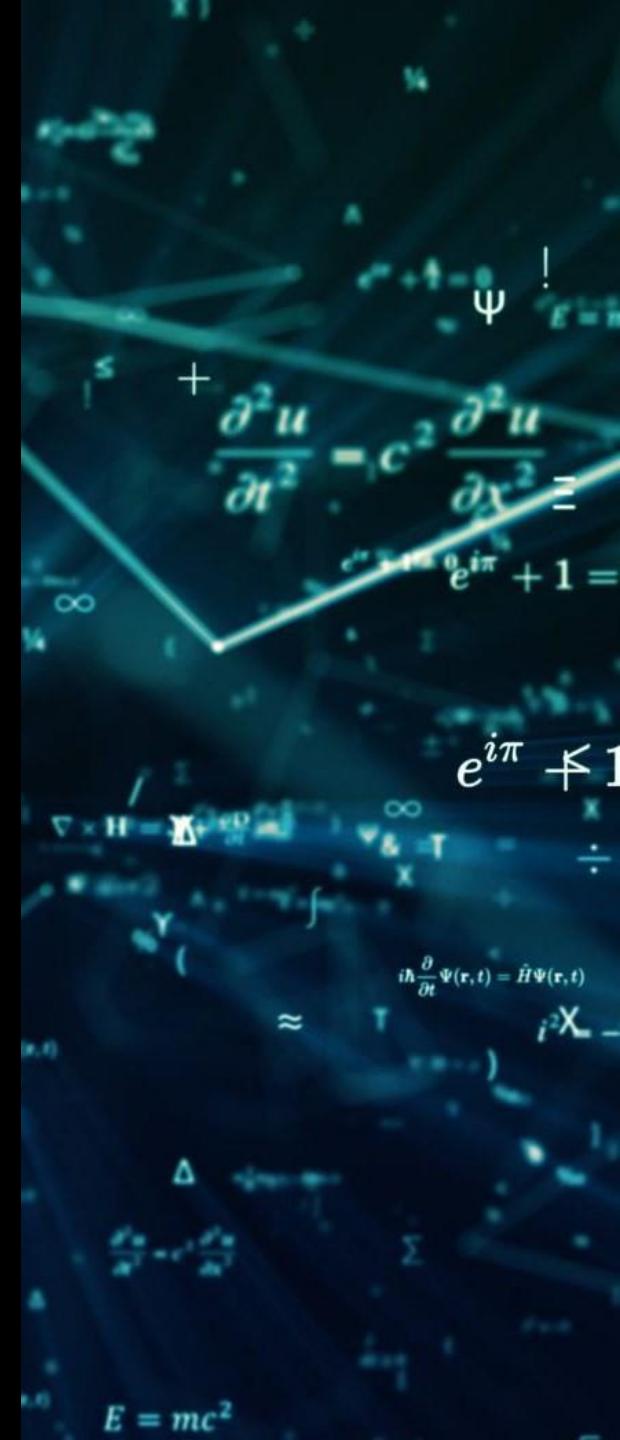
**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Classification

---

- Solutions for Overfitting
- PCA
- FDA/LDA



# Ways to reduce overfitting



- Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations
- To reduce the overfitting, a general rule of thumb is to try to decrease the model complexity, but not too much to avoid underfitting.
- Specifically, techniques that can be used include:
  - **Cross-Validation:** Assess the model's performance on multiple subsets of the data to ensure good generalization to different portions of the dataset.
  - **Increase Dataset size:** A larger dataset can provide more diverse examples and help the model generalize better.
  - **Early Stopping:** Stop training once the performance on the validation set starts to degrade, preventing the model from overfitting the training data.
  - **Dropout:** random neurons are "dropped out" (ignored) during each training iteration. This helps prevent co-adaptation of neurons and reduces overfitting.
  - **Data Augmentation:** Increase the size of the training dataset by applying various transformations to the existing data
  - **Regularization?**

# Regularization

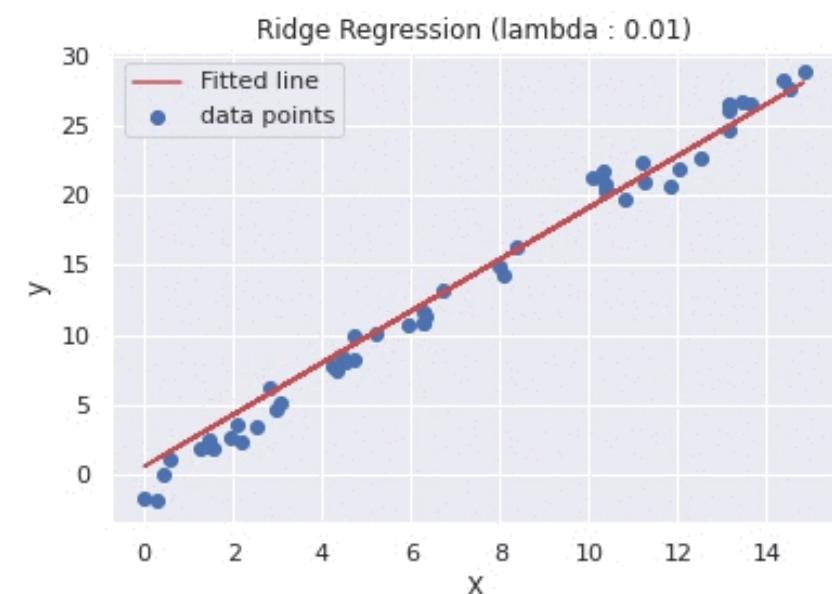


- The intuition is that one way to decrease the overall model complexity is to reduce the magnitude of the learned weights  $\vec{w}$ . This way we won't be assigning very high weights to certain features over the others → Simpler model

- This could be achieved by adding a regularization term to the loss function of any machine learning algorithm, such that,

$$L_{reg}(\vec{w}, \mathbf{b}) = L(\vec{w}, \mathbf{b}) + [\beta * \text{Regularization Term}]$$

- This will allow the model to not only prioritize minimizing the cost term, but also minimizing the magnitude of the weights (controlled by the hyperparameter  $\beta$ )
- There are different types of regularization terms that could be used.



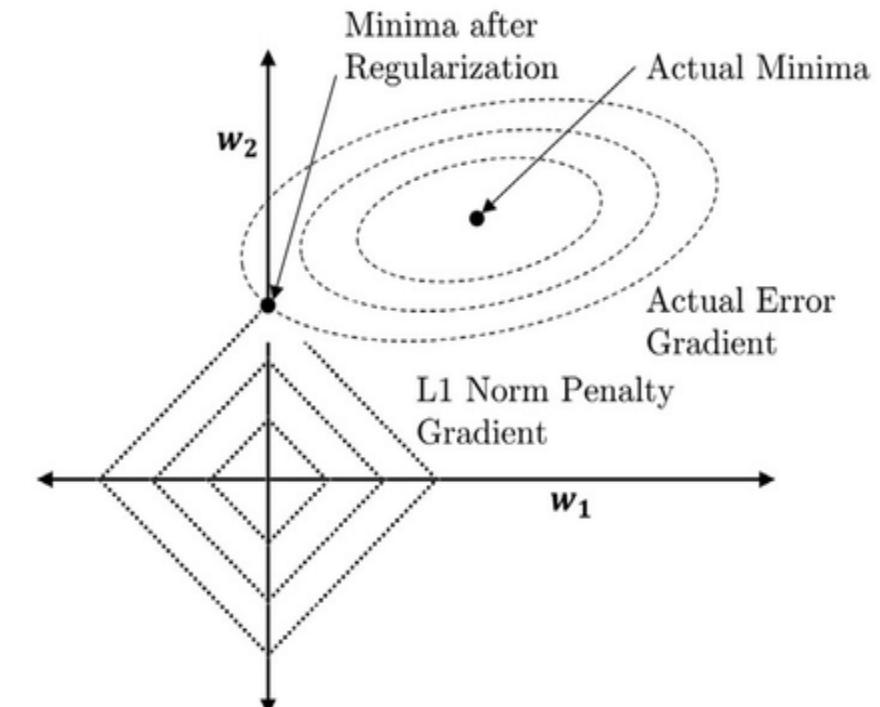
# L1 Regularization (Lasso)



- L<sub>1</sub> regularization introduces a penalty term proportional to the absolute values of the model parameters.
- Mathematically,

$$L_{reg}(\vec{w}, b) = L(\vec{w}, b) + \left[ \beta * \sum_p |\vec{w}_p| \right]$$

- This encourages the model to prefer a sparse set of features, effectively driving some of them to exactly zero.
- Not only does it help prevent overfitting but also performs feature selection by excluding less relevant features.



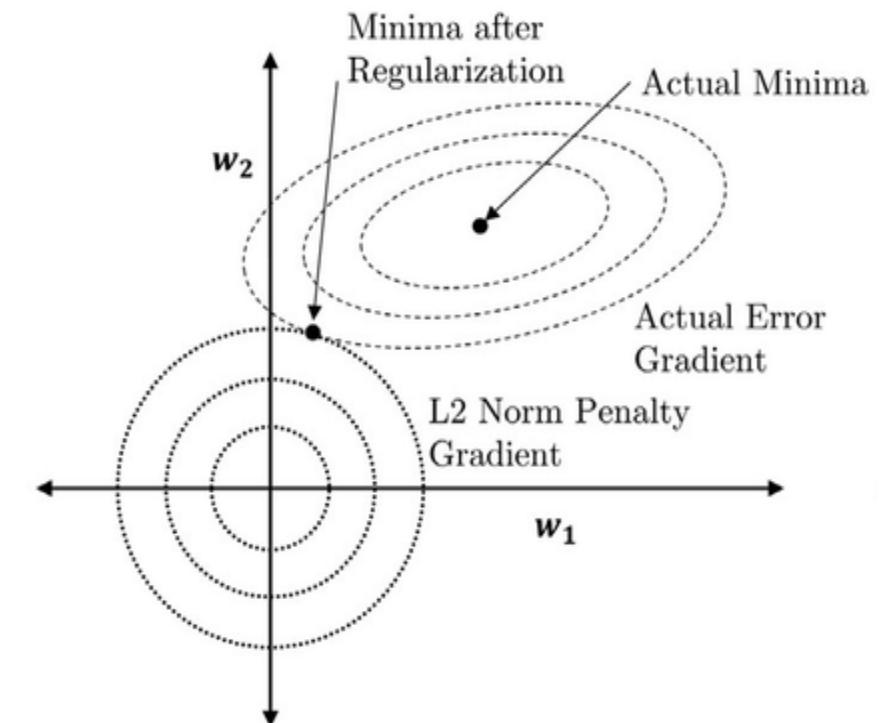
# L<sub>2</sub> Regularization (Ridge)



- L<sub>2</sub> regularization introduces a penalty term proportional to the square of the model parameters.
- Mathematically,

$$L_{reg}(\vec{w}, b) = L(\vec{w}, b) + \left[ \beta * \sum_p (\vec{w}_p)^2 \right]$$

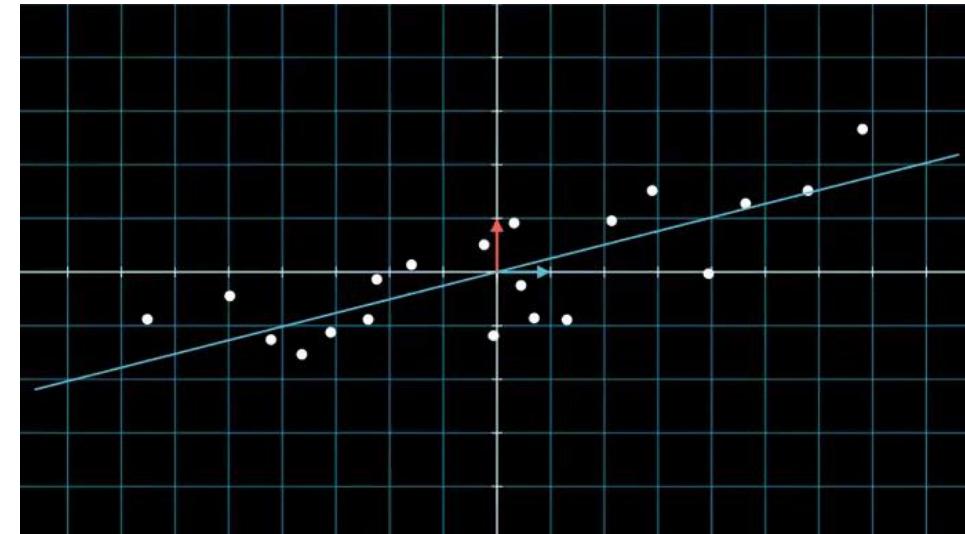
- This discourages the weights from becoming too large, preventing individual features from dominating the model.
- It distributes the importance of features more evenly, leading to a smoother model.



# Dimensionality Reduction



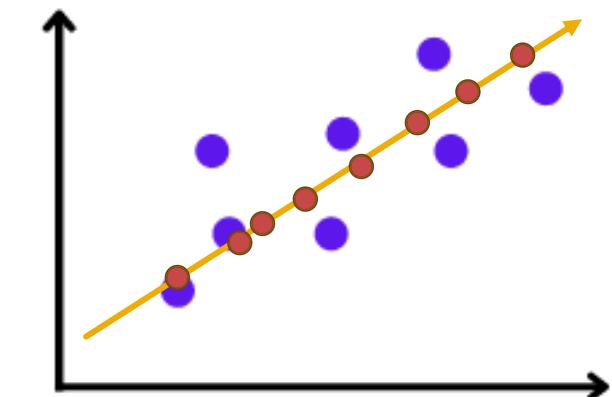
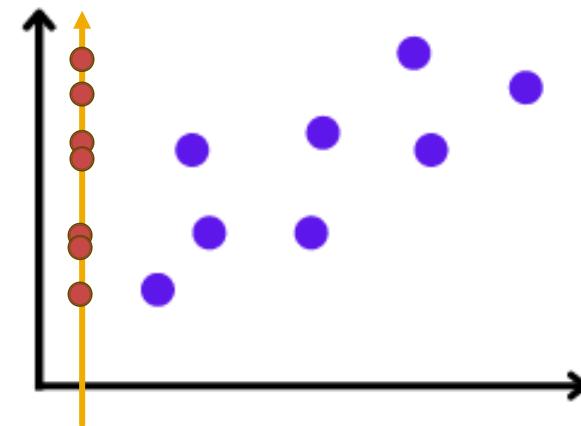
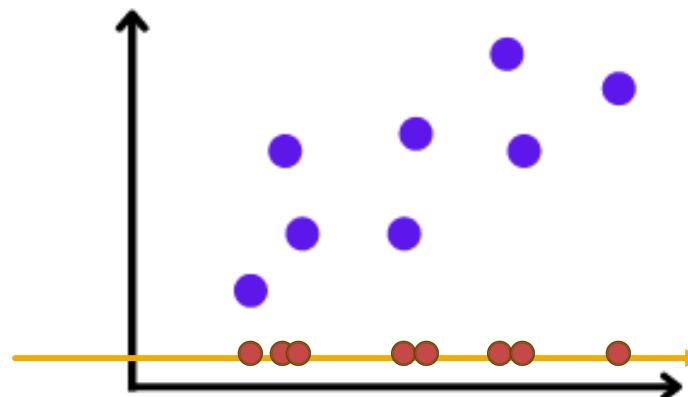
- The efficiency of ML methods depends crucially on the choice of features that are used to characterize data points.
- Target → have a small number of highly relevant features to characterize data points.
- Dimensionality Reduction techniques reduce the number of input variables or features in a dataset while retaining its essential characteristics
- Benefits of dimensionality reduction:
  - Reduce excessive resource requirements
  - Reduce the probability of overfitting
  - Make data visualizations easier



# Principal Component Analysis (PCA)



- Intuitively, if we want to project the following 2D points to only one dimension, which one of the following best preserves the shape of the dataset?

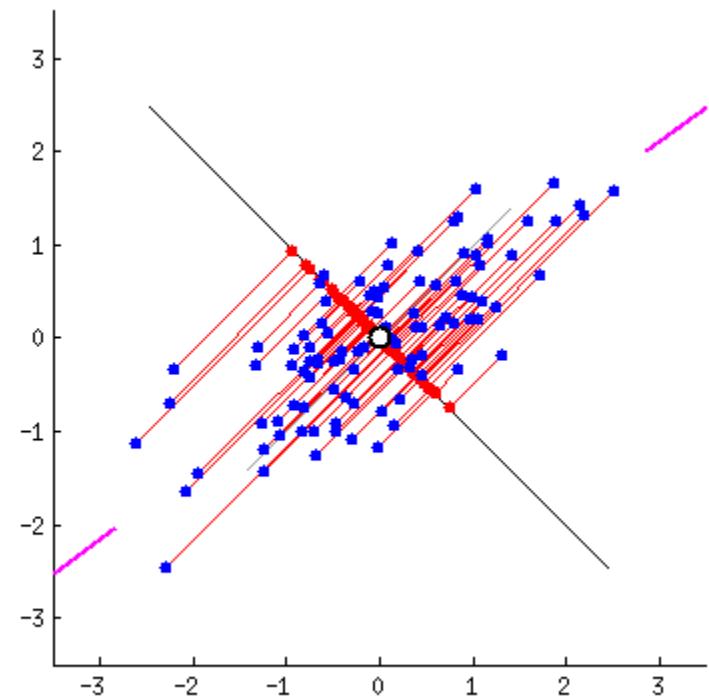


- Generally, it looks like in the best dimension, the projected data points have the largest possible variance.

# Principal Component Analysis (PCA)



- The PCA is a dimensionality reduction technique that transforms high-dimensional data into a new coordinate system (principal components/axes).
- Target → **Maximize** the variance of projected data.
- Doesn't need any class labels (unsupervised)
- The projected data must obey certain properties:
  - Each new feature (principal component/axis) is a linear combination of the original features (axes)
  - Each principal component (axis) must be perpendicular to all other principal components (axes).
  - Each principal component (axis) must be a unit vector (magnitude = 1)



# Principal Component Analysis (PCA)



- **Covariance Matrix**: provides important information about the relationships between pairs of variables in a dataset.
- Given a dataset  $X \in R^{D*N}$  with  $D$  features and  $N$  rows, the covariance matrix will have size  $D * D$  can be computed as follows:

$$C(X) = \frac{1}{N} (X - \bar{X})(X - \bar{X})^T$$

- $\bar{X}$  represents the mean of each feature in  $X$ .
- Cell  $C_{ij}$  represents the co-variance between feature  $i$  and feature  $j$ .
- If one feature increases/decreases and the other feature increases/decreases at the same time  $\rightarrow C_{ij} \neq 0$ .
- If the features are not correlated  $\rightarrow C_{ij} = 0$ .

# Eigen Vectors and Eigen Values



- Given a vector  $v$ , apply linear transformation with square matrix  $A$ .

$$A \cdot v = ?$$

$$A \cdot v = \lambda v \rightarrow [\text{Special Case}]$$

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\lambda_1 = 3, \quad v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = -1, \quad v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

# Principal Component Analysis (PCA)



- Given the initial dataset with  $D$  features, we want to get the first axis of the new coordinate space.
- We will apply a linear transformation  $u \in R^{D*1}$  on each original data point  $x \in R^{D*1}$ , such that the value  $z$  of the projected point at the new axis is formulated as,

$$z = u^T x$$

- Applying the same equation on the full dataset  $X \in R^{D*N}$ , such that,

$$Z = u^T X$$

- Target  $\rightarrow$  **Maximize** the variance of projected data (1D)  $\rightarrow$  **Maximize** the covariance matrix of projected data (Multi-Dimensional)

# Principal Component Analysis (PCA)



- Target  $\rightarrow$  **Maximize** the covariance matrix of projected data
- $Cov(Z) = \frac{1}{N}(Z - \bar{Z})(Z - \bar{Z})^T$
- $Cov(Z) = \frac{1}{N}(u^T X - u^T \bar{X})(u^T X - u^T \bar{X})^T$
- $Cov(Z) = \frac{1}{N}(u^T(X - \bar{X}))(u^T(X - \bar{X}))^T \rightarrow$  Expand 2<sup>nd</sup> Transpose
- $Cov(Z) = \frac{1}{N}u^T(X - \bar{X})(X - \bar{X})^T u = u^T[\frac{1}{N}(X - \bar{X})(X - \bar{X})^T]u$
- $Cov(Z) = u^T S u, \text{ where } S = Cov(X)$

# Principal Component Analysis (PCA)



*maximize  $u^T S u$ ,  
such that  $u^T u = 1$*

- The constraint guarantees that the axis  $u$  is a unit vector.
- By borrowing the concept of Lagrangian Multipliers, we will translate the equation to a loss function to be ***minimized***, such that,

$$L(u, \lambda) = -(u^T S u - \lambda(u^T u - 1))$$

- To minimize the loss with respect to  $u \rightarrow$  Solve  $\frac{dL}{du} = 0$

# Matrix/Vector Rules and Derivatives



| Rule   | Comments  |
|--|---|
| $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$  | order is reversed, everything is transposed                       |
| $(\mathbf{a}^T \mathbf{B} \mathbf{c})^T = \mathbf{c}^T \mathbf{B}^T \mathbf{a}$              | as above  |
| $\mathbf{a}^T \mathbf{b} = \mathbf{b}^T \mathbf{a}$  | (the result is a scalar, and the transpose of a scalar is itself) |
| $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$                            | multiplication is distributive                                    |
| $(\mathbf{a} + \mathbf{b})^T \mathbf{C} = \mathbf{a}^T \mathbf{C} + \mathbf{b}^T \mathbf{C}$ | as above, with vectors  |
| $\mathbf{AB} \neq \mathbf{BA}$   | multiplication is <b>not</b> commutative                          |

| Scalar derivative                | Vector derivative   |
|----------------------------------|---|
| $f(x) \rightarrow \frac{df}{dx}$ | $f(\mathbf{x}) \rightarrow \frac{df}{d\mathbf{x}}$                      |
| $bx \rightarrow b$               | $\mathbf{x}^T \mathbf{B} \rightarrow \mathbf{B}$                        |
| $bx \rightarrow b$               | $\mathbf{x}^T \mathbf{b} \rightarrow \mathbf{b}$                        |
| $x^2 \rightarrow 2x$             | $\mathbf{x}^T \mathbf{x} \rightarrow 2\mathbf{x}$                       |
| $bx^2 \rightarrow 2bx$           | $\mathbf{x}^T \mathbf{B} \mathbf{x} \rightarrow 2\mathbf{B} \mathbf{x}$ |

# Principal Component Analysis (PCA)



$$L(u, \lambda) = -(u^T S u - \lambda(u^T u - 1))$$

- To minimize the loss with respect to  $u \rightarrow$  Solve  $\frac{dL}{du} = 0$
- $\frac{dL}{du} = -(2S u - 2\lambda u) = 0$

$$S u = \lambda u \rightarrow \text{Result of } \frac{dl}{du} = 0$$

- We reach a formulation exactly similar to the one of eigenvalues and eigenvectors.
- In other words,  $u$  is considered an eigenvector of the covariance matrix  $S$  of the original data and  $\lambda$  is the associated eigenvalue.

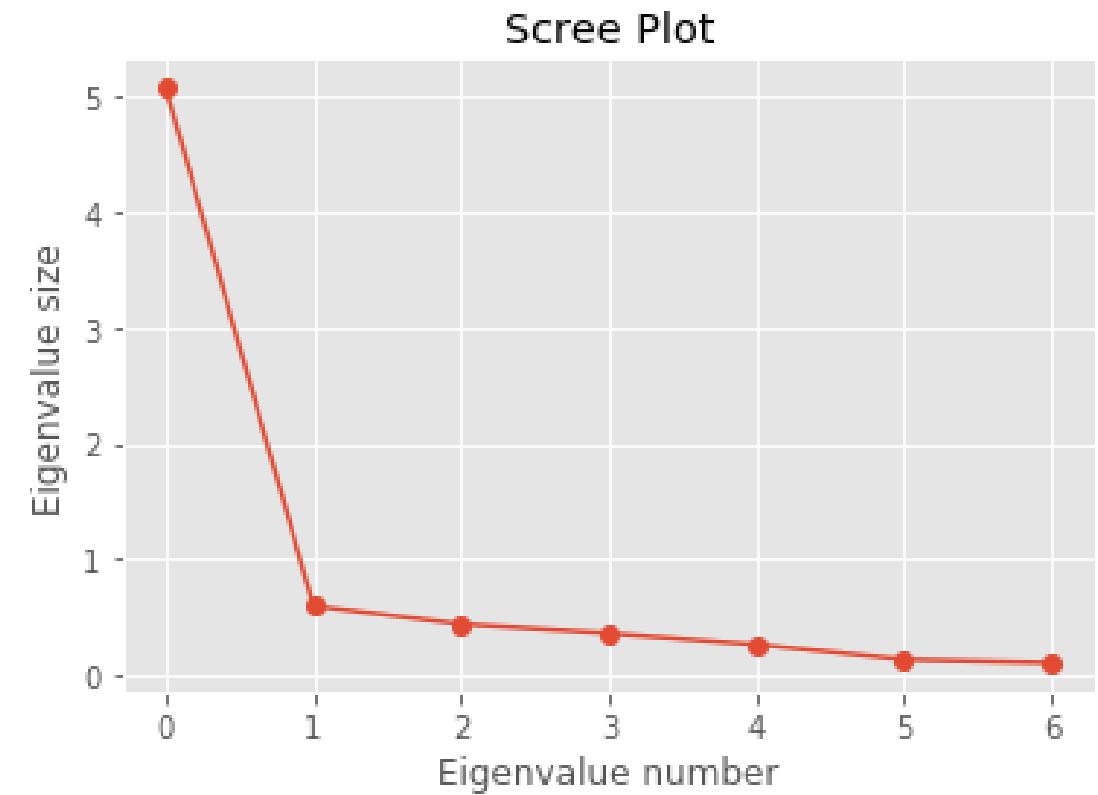
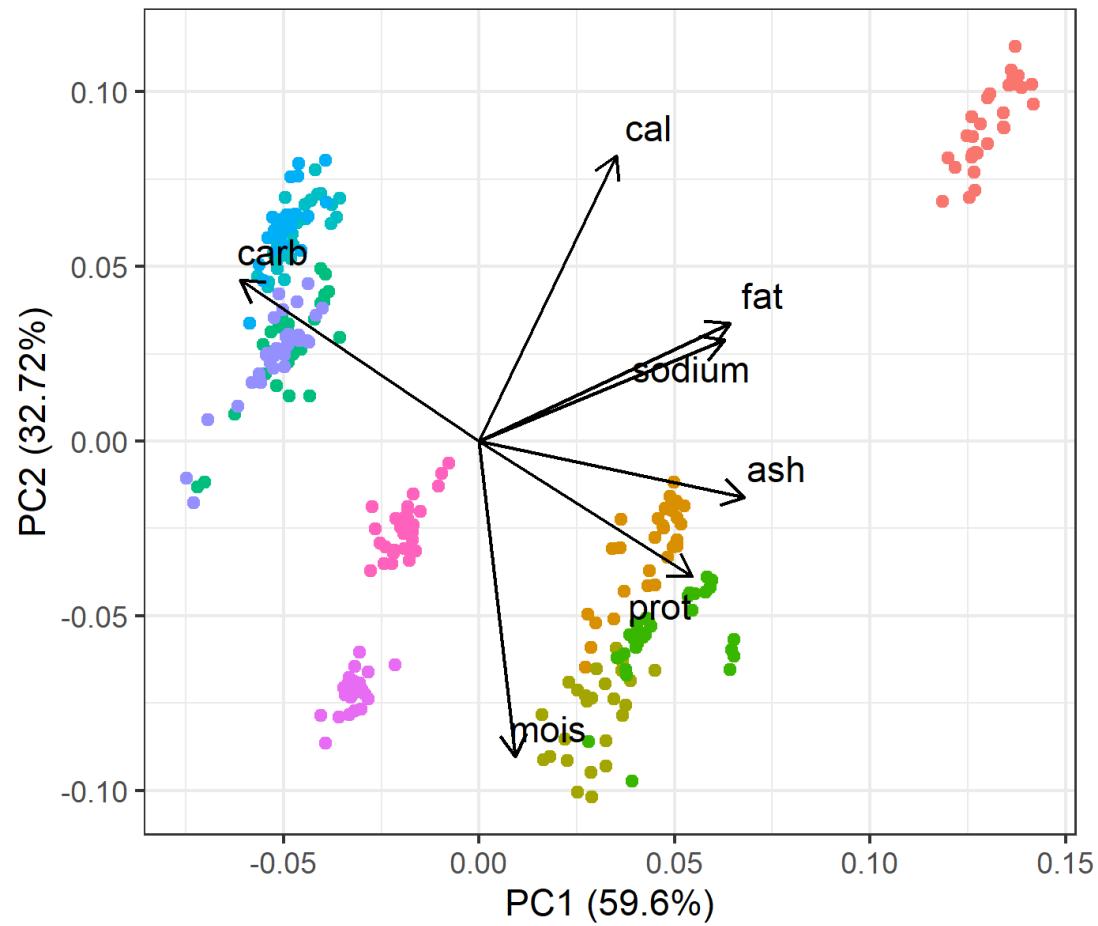
# Principal Component Analysis (PCA)



$$S\mathbf{u} = \lambda\mathbf{u}$$

- To project original data  $X \in R^{D*N}$  to  $P$  features, where  $P \leq D$ :
  - Calculate the Covariance Matrix  $S = \frac{1}{N}(X - \bar{X})(X - \bar{X})^T$
  - Get all the possible eigenvalues and eigenvectors of  $S$ .
  - Sort the eigenvalues in descending order:
    - The Largest eigenvalue corresponds to the (eigenvector) axis with highest variance of data projected on that axis.
    - Lower eigenvalues correspond to axes that are worse in preserving the characteristics of the data.
  - Get the highest  $P$  eigenvalues and their eigenvectors.
  - Construct full matrix  $U \in R^{P*D}$  by stacking all chosen eigenvectors vertically (row-wise)
  - Get the final full projected dataset  $Z \in R^{P*N} \rightarrow Z = UX^T$

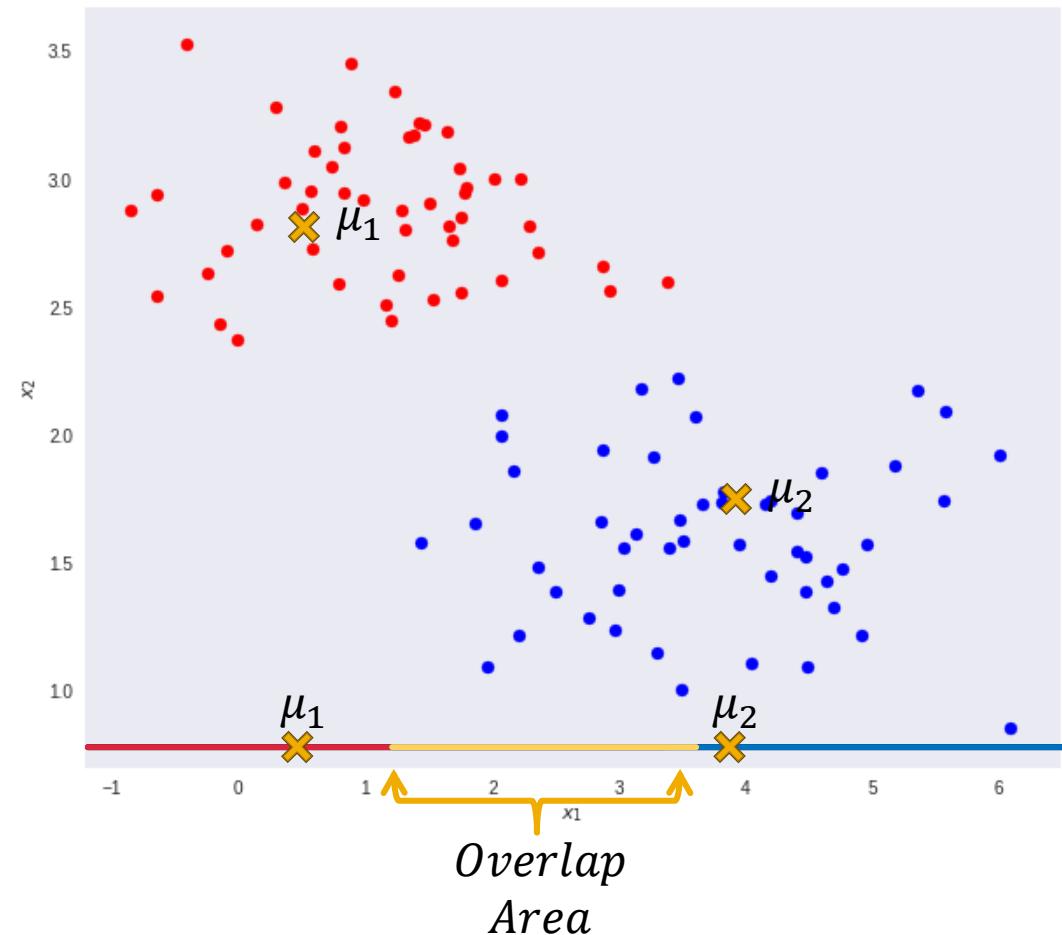
# Principal Component Analysis (PCA)



# Fisher Discriminant Analysis (FDA)



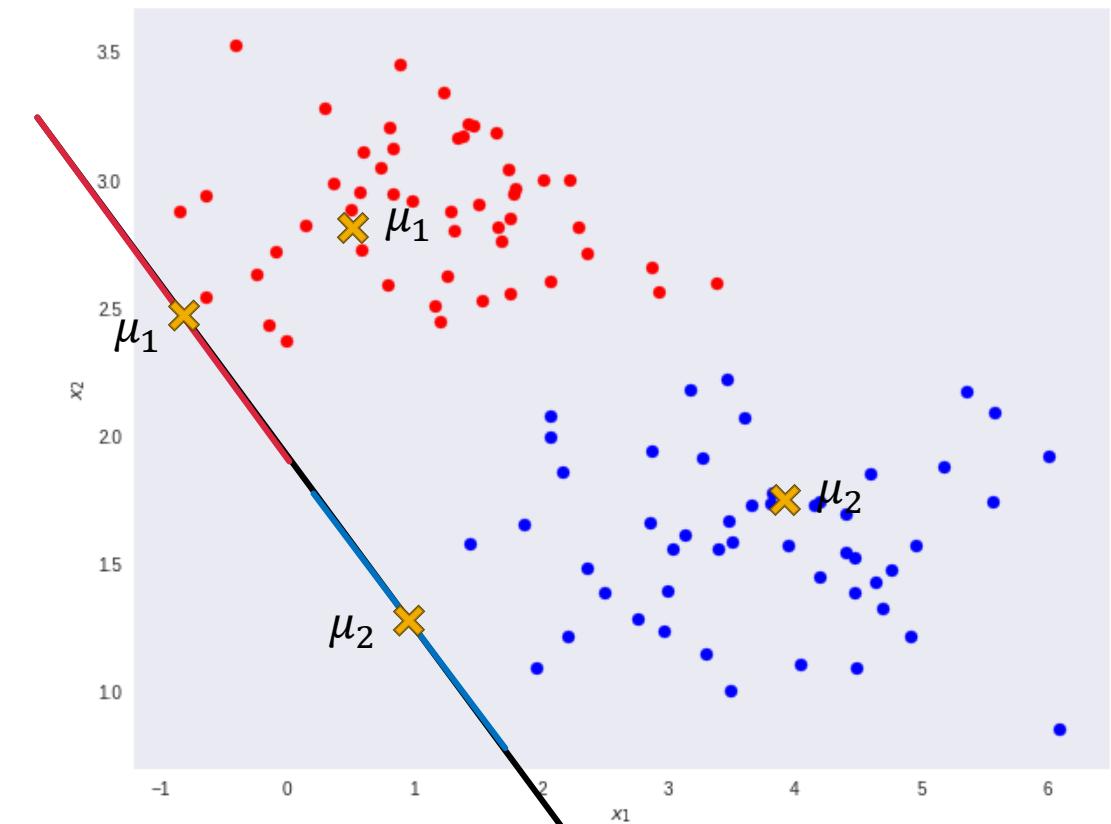
- Assume we want to project our features to fewer dimensions while maintaining the separability of our classes.
- A possible approach would be to **maximize** the distance between the centers (means) of the projected classes.
- In the following example, does the proposed axis maintain the best separability between the classes?



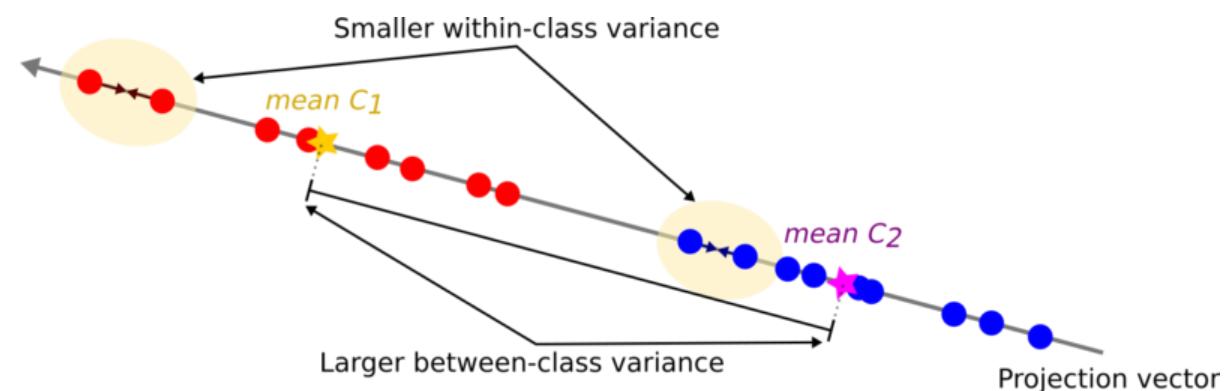
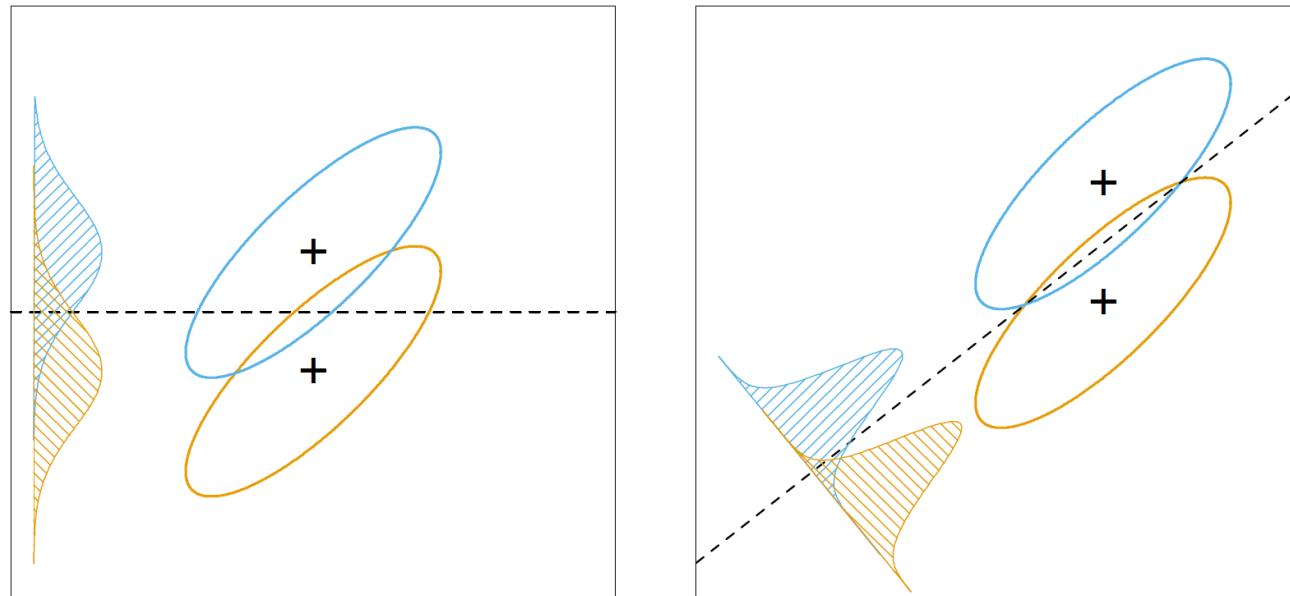
# Fisher Discriminant Analysis (FDA)



- It looks like maximizing the distance between means of projected classes is not enough if the classes are wide-spread with high variance.
- An additional constraint could be imposed by minimizing the within-class variance of the projected data.
- Combining the two constraints leads to a new axis (dimensionality) that maintains the separability of the original data with minimum or no overlap.



# Fisher Discriminant Analysis (FDA)



# Fisher Discriminant Analysis (FDA)



- Fisher's Linear Discriminant Analysis (FDA) is a linear dimensionality reduction technique that aims to project high-dimensional data into a lower-dimensional space while maximizing the separation between classes.
- This is achieved through two targets:
  - **T<sub>1</sub>: Maximizing Inter-Class Variance (Distance between class means)**
  - **T<sub>2</sub>: Minimizing the Intra-Class Variance (Within-class variance).**

# Fisher Discriminant Analysis (FDA)



- Assume that we have two classes to be projected.
- We will apply a linear transformation  $u \in R^{D*1}$  on each original data point  $x_{\{0,1\}} \in R^{D*1}$  belonging to classes 0 and 1, such that the value  $z$  of the projected point at the new axis is formulated as,

$$z = u^T X$$

- The means of the points in each class are formulated as,

$$\mu_0 = \frac{1}{N_0} \sum_{i=0}^{N_0} x_0^i,$$

$$\mu_1 = \frac{1}{N_1} \sum_{i=0}^{N_1} x_1^i$$

# Fisher Discriminant Analysis (FDA)



- $\mu_0 = \frac{1}{N_0} \sum_{i=0}^{N_0} x_0^i, \quad \mu_1 = \frac{1}{N_1} \sum_{i=0}^{N_1} x_1^i$
- **For Target 1:** Distance between the projected means is formulated as:
  - $(u^T \mu_0 - u^T \mu_1)^2 = (u^T \mu_0 - u^T \mu_1)^T (u^T \mu_0 - u^T \mu_1)$
  - $(u^T \mu_0 - u^T \mu_1)^2 = (\mu_0 - \mu_1)^T u u^T (\mu_0 - \mu_1)$
  - $(u^T \mu_0 - u^T \mu_1)^2 = u^T (\mu_0 - \mu_1) (\mu_0 - \mu_1)^T u$
  - $(u^T \mu_0 - u^T \mu_1)^2 = \mathbf{u}^T \mathbf{S}_B \mathbf{u}, \quad S_B = (\mu_0 - \mu_1) (\mu_0 - \mu_1)^T$
  - Where  $S_B$  represents the distance between class means before projection.

# Fisher Discriminant Analysis (FDA)



- Recall from PCA  $\rightarrow Cov(Z) = u^T S u$ , where  $S = Cov(X)$
- **For Target 2:** Within class-variance for the two classes can be formulated as:
  - $Cov(Z_0 + Z_1) = Cov(Z_0) + Cov(Z_1)$
  - $Cov(Z_0 + Z_1) = u^T S_0 u + u^T S_1 u$ , where  $S_0 = Cov(X_0), S_1 = Cov(X_1)$
  - $Cov(Z_0 + Z_1) = u^T (S_0 + S_1) u = \mathbf{u}^T \mathbf{S}_W \mathbf{u}$ ,  $S_W = S_0 + S_1$
  - Where  $S_W$  represents the within-class variance of the two classes altogether.

# Fisher Discriminant Analysis (FDA)



- To Achieve both **Target 1** and **Target 2**, our target could be formulated as follows:

$$\underset{u}{\text{maximize}} \frac{u^T S_B u}{u^T S_W u}$$

- Recall that our new axis needs to be unit vector (from PCA), to enforce it we can formulate the target as follows:

$$\underset{u}{\text{maximize}} \quad u^T S_B u, \quad \text{s.t.} \quad u^T S_W u = 1$$

- To formulate it as a loss function, we will borrow the concepts from Lagrangian Multipliers:

$$L(u, \lambda) = - \left( u^T S_B u - \lambda(u^T S_W u - 1) \right)$$

- The negative sign is added to minimize rather than maximize

# Fisher Discriminant Analysis (FDA)



$$L(u, \lambda) = - \left( u^T S_B u - \lambda(u^T S_W u - 1) \right)$$

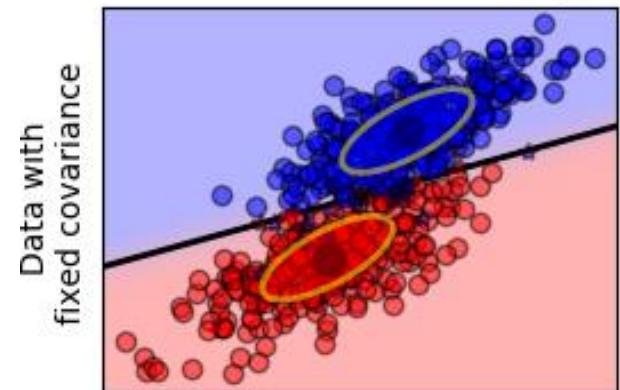
- To minimize the loss with respect to  $u \rightarrow$  Solve  $\frac{dL}{du} = 0$
- $\frac{dL}{du} = 2S_B u - 2\lambda S_W u = 0$
- $S_B u = \lambda S_W u \rightarrow [S_W^{-1} S_B] u = \lambda u$
- We reach a formulation exactly similar to the one of eigenvalues and eigenvectors.
- In other words,  $u$  is considered an eigenvector of the matrix  $S_W^{-1} S_B$  calculated from the original data and  $\lambda$  is the associated eigenvalue.
- To transform the full dataset, follow the same steps as PCA, but with calculating  $S_W^{-1} S_B$  in the first step instead.

# LDA vs FDA

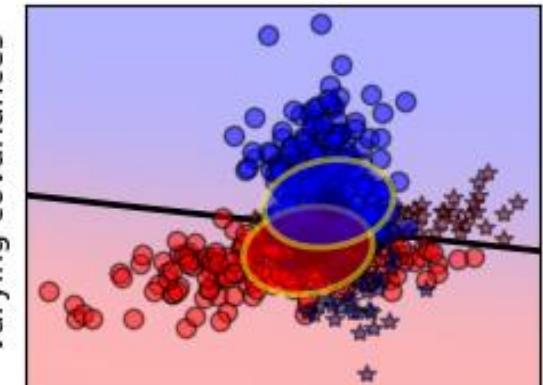


- Both Linear Discriminant Analysis (LDA) and FDA refer to the same technique which aims to project the data to lower dimensions while maximizing the class separability.
- LDA is the direct extension of FDA to work with two **or more** classes.
- LDA is not only doing dimensionality reduction, but also computes the *linear decision boundary* between the classes in the projected space.
- LDA makes important assumptions about the shape of the data:
  - All classes follow a gaussian (normal) distribution
  - All classes have equal (identical) covariance matrices.
- If any of those assumptions doesn't hold, LDA won't perform well in classification or dimensionality reduction.

Linear Discriminant Analysis



Data with varying covariances



# Extra Resources



- More information regarding the Matrix/Vector Derivatives:
  - <https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>

# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**



# Artificial Intelligence Algorithms and Mathematics

---

CSCN 8000



# Unsupervised

---

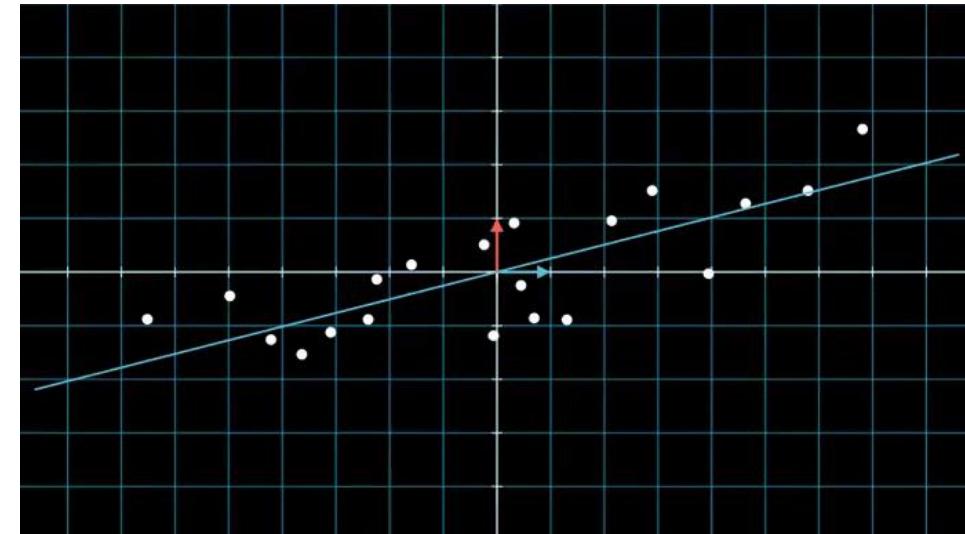
- FDA/LDA
- K-Means
- Hierarchical Clustering



# Recall: Dimensionality Reduction



- The efficiency of ML methods depends crucially on the choice of features that are used to characterize data points.
- Target → have a small number of highly relevant features to characterize data points.
- Dimensionality Reduction techniques reduce the number of input variables or features in a dataset while retaining its essential characteristics
- Benefits of dimensionality reduction:
  - Reduce excessive resource requirements
  - Reduce the probability of overfitting
  - Make data visualizations easier



# Principal Component Analysis (PCA)



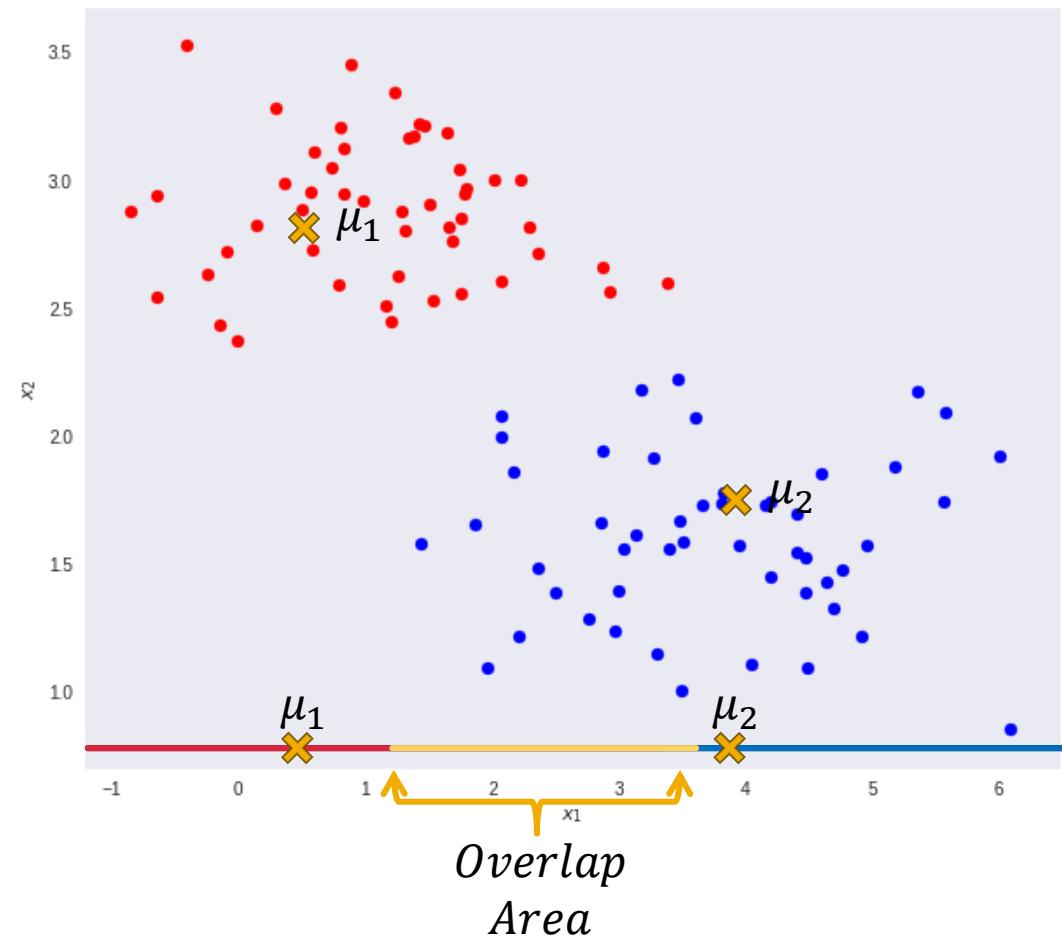
$$S\mathbf{u} = \lambda\mathbf{u}$$

- To project original data  $X \in R^{D*N}$  to  $P$  features, where  $P \leq D$ :
  - Calculate the Covariance Matrix  $S = \frac{1}{N}(X - \bar{X})(X - \bar{X})^T$
  - Get all the possible eigenvalues and eigenvectors of  $S$ .
  - Sort the eigenvalues in descending order:
    - The Largest eigenvalue corresponds to the (eigenvector) axis with highest variance of data projected on that axis.
    - Lower eigenvalues correspond to axes that are worse in preserving the characteristics of the data.
  - Get the highest  $P$  eigenvalues and their eigenvectors.
  - Construct full matrix  $U \in R^{P*D}$  by stacking all chosen eigenvectors vertically (row-wise)
  - Get the final full projected dataset  $Z \in R^{P*N} \rightarrow Z = UX^T$

# Fisher Discriminant Analysis (FDA)



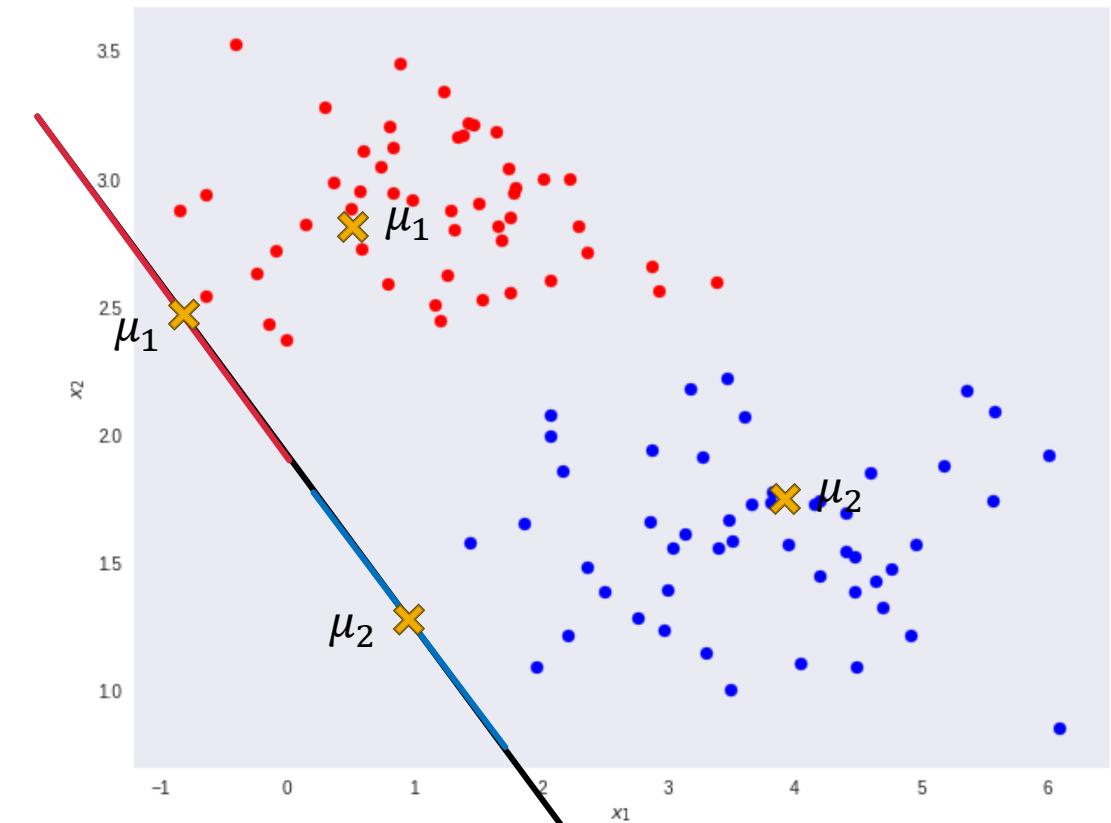
- Assume we want to project our features to fewer dimensions while maintaining the separability of our classes.
- A possible approach would be to **maximize** the distance between the centers (means) of the projected classes.
- In the following example, does the proposed axis maintain the best separability between the classes?



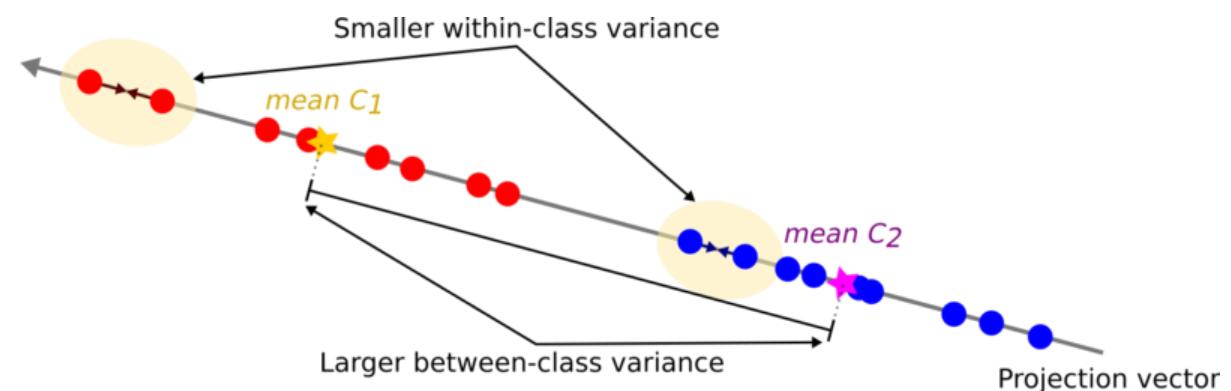
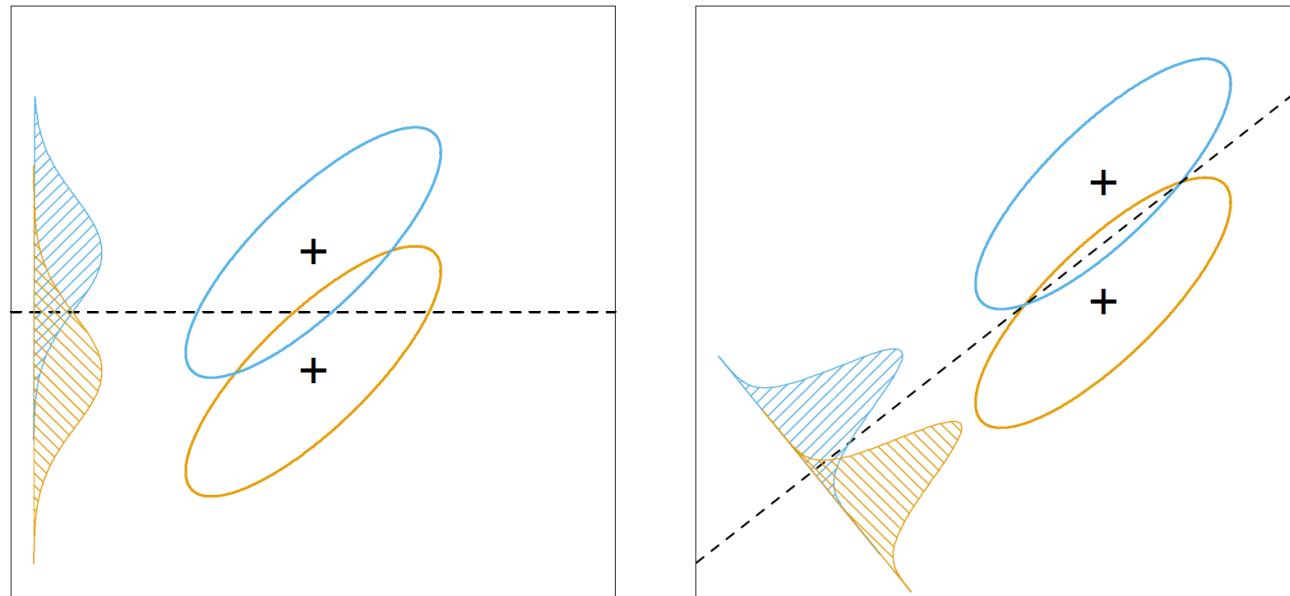
# Fisher Discriminant Analysis (FDA)



- It looks like maximizing the distance between means of projected classes is not enough if the classes are wide-spread with high variance.
- An additional constraint could be imposed by minimizing the within-class variance of the projected data.
- Combining the two constraints leads to a new axis (dimensionality) that maintains the separability of the original data with minimum or no overlap.



# Fisher Discriminant Analysis (FDA)



# Fisher Discriminant Analysis (FDA)



- Fisher's Linear Discriminant Analysis (FDA) is a linear dimensionality reduction technique that aims to project high-dimensional data into a lower-dimensional space while maximizing the separation between classes.
- This is achieved through two targets:
  - **T<sub>1</sub>: Maximizing Inter-Class Variance (Distance between class means)**
  - **T<sub>2</sub>: Minimizing the Intra-Class Variance (Within-class variance).**

# Fisher Discriminant Analysis (FDA)



- Assume that we have two classes to be projected.
- We will apply a linear transformation  $u \in R^{D*1}$  on each original data point  $x_{\{0,1\}} \in R^{D*1}$  belonging to classes 0 and 1, such that the value  $z$  of the projected point at the new axis is formulated as,

$$z = u^T X$$

- The means of the points in each class are formulated as,

$$\mu_0 = \frac{1}{N_0} \sum_{i=0}^{N_0} x_0^i,$$

$$\mu_1 = \frac{1}{N_1} \sum_{i=0}^{N_1} x_1^i$$

# Fisher Discriminant Analysis (FDA)



- $\mu_0 = \frac{1}{N_0} \sum_{i=0}^{N_0} x_0^i, \quad \mu_1 = \frac{1}{N_1} \sum_{i=0}^{N_1} x_1^i$
- **For Target 1:** Distance between the projected means is formulated as:
  - $(u^T \mu_0 - u^T \mu_1)^2 = (u^T \mu_0 - u^T \mu_1)^T (u^T \mu_0 - u^T \mu_1)$
  - $(u^T \mu_0 - u^T \mu_1)^2 = (\mu_0 - \mu_1)^T u u^T (\mu_0 - \mu_1)$
  - $(u^T \mu_0 - u^T \mu_1)^2 = u^T (\mu_0 - \mu_1) (\mu_0 - \mu_1)^T u$
  - $(u^T \mu_0 - u^T \mu_1)^2 = \mathbf{u}^T \mathbf{S}_B \mathbf{u}, \quad S_B = (\mu_0 - \mu_1) (\mu_0 - \mu_1)^T$
  - Where  $S_B$  represents the distance between class means before projection.

# Fisher Discriminant Analysis (FDA)



- Recall from PCA  $\rightarrow Cov(Z) = u^T S u$ , where  $S = Cov(X)$
- **For Target 2:** Within class-variance for the two classes can be formulated as:
  - $Cov(Z_0 + Z_1) = Cov(Z_0) + Cov(Z_1)$
  - $Cov(Z_0 + Z_1) = u^T S_0 u + u^T S_1 u$ , where  $S_0 = Cov(X_0), S_1 = Cov(X_1)$
  - $Cov(Z_0 + Z_1) = u^T (S_0 + S_1) u = \mathbf{u}^T \mathbf{S}_W \mathbf{u}$ ,  $S_W = S_0 + S_1$
  - Where  $S_W$  represents the within-class variance of the two classes altogether.

# Fisher Discriminant Analysis (FDA)



- To Achieve both **Target 1** and **Target 2**, our target could be formulated as follows:

$$\underset{u}{\text{maximize}} \frac{u^T S_B u}{u^T S_W u}$$

- Recall that our new axis needs to be unit vector (from PCA), to enforce it we can formulate the target as follows:

$$\underset{u}{\text{maximize}} \quad u^T S_B u, \quad \text{s.t.} \quad u^T S_W u = 1$$

- To formulate it as a loss function, we will borrow the concepts from Lagrangian Multipliers:

$$L(u, \lambda) = - \left( u^T S_B u - \lambda(u^T S_W u - 1) \right)$$

- The negative sign is added to minimize rather than maximize

# Fisher Discriminant Analysis (FDA)



$$L(u, \lambda) = - \left( u^T S_B u - \lambda(u^T S_W u - 1) \right)$$

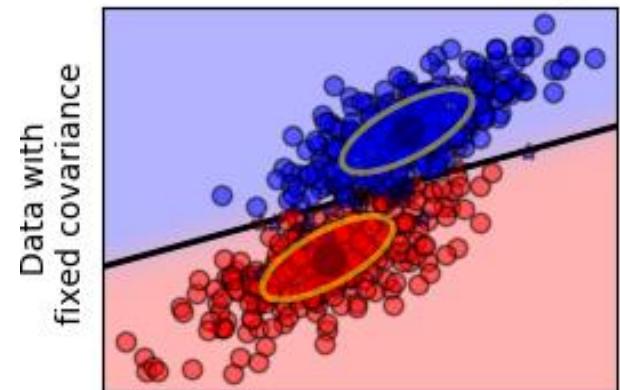
- To minimize the loss with respect to  $u \rightarrow$  Solve  $\frac{dL}{du} = 0$
- $\frac{dL}{du} = 2S_B u - 2\lambda S_W u = 0$
- $S_B u = \lambda S_W u \rightarrow [S_W^{-1} S_B] u = \lambda u$
- We reach a formulation exactly similar to the one of eigenvalues and eigenvectors.
- In other words,  $u$  is considered an eigenvector of the matrix  $S_W^{-1} S_B$  calculated from the original data and  $\lambda$  is the associated eigenvalue.
- To transform the full dataset, follow the same steps as PCA, but with calculating  $S_W^{-1} S_B$  in the first step instead.

# LDA vs FDA

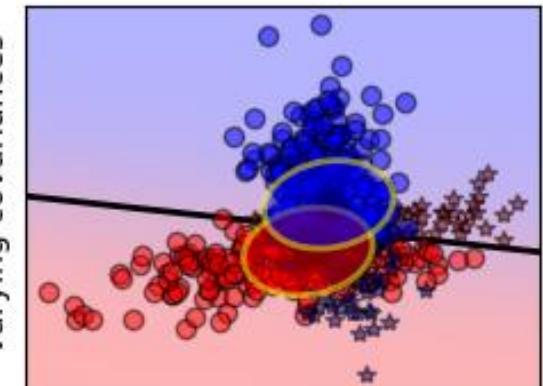


- Both Linear Discriminant Analysis (LDA) and FDA refer to the same technique which aims to project the data to lower dimensions while maximizing the class separability.
- LDA is the direct extension of FDA to work with two **or more** classes.
- LDA is not only doing dimensionality reduction, but also computes the *linear decision boundary* between the classes in the projected space.
- LDA makes important assumptions about the shape of the data:
  - All classes follow a gaussian (normal) distribution
  - All classes have equal (identical) covariance matrices.
- If any of those assumptions doesn't hold, LDA won't perform well in classification or dimensionality reduction.

Linear Discriminant Analysis



Data with varying covariances

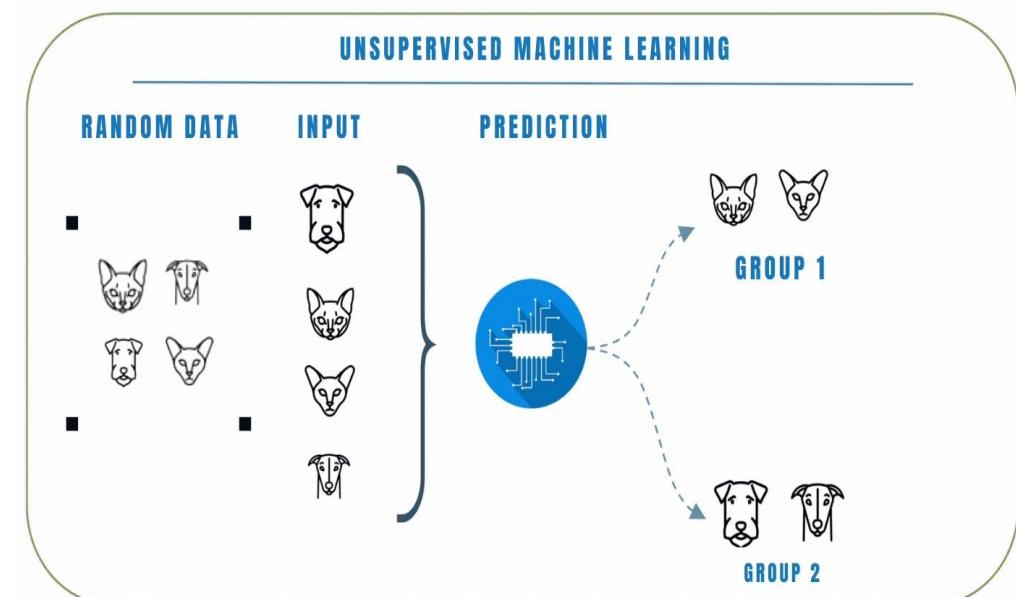


# Unsupervised Learning

# Unsupervised Learning



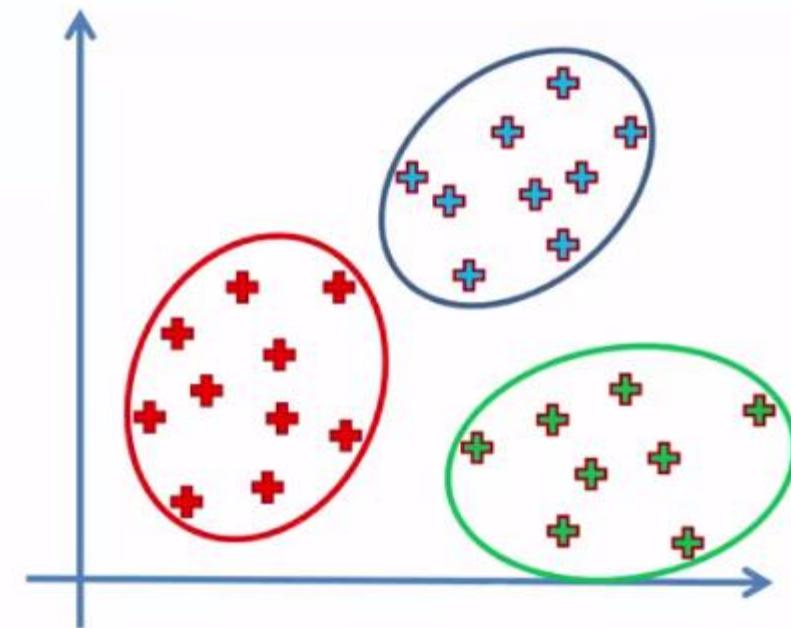
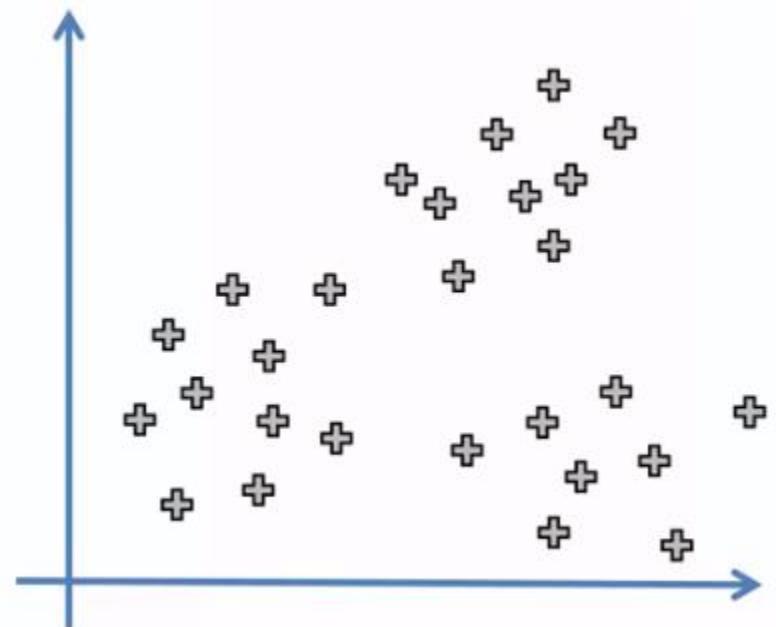
- Unsupervised learning is a type of machine learning where the algorithm is given data without explicit instructions on what to do with it.
- The system tries to learn the patterns and the structure of the data without any labeled responses to guide the learning process.
- Benefits:
  - Uncover hidden patterns and structures
  - Adaptable to various types of data without the need for labeled examples
  - Valuable tool for exploratory data analysis



# K-Means Clustering



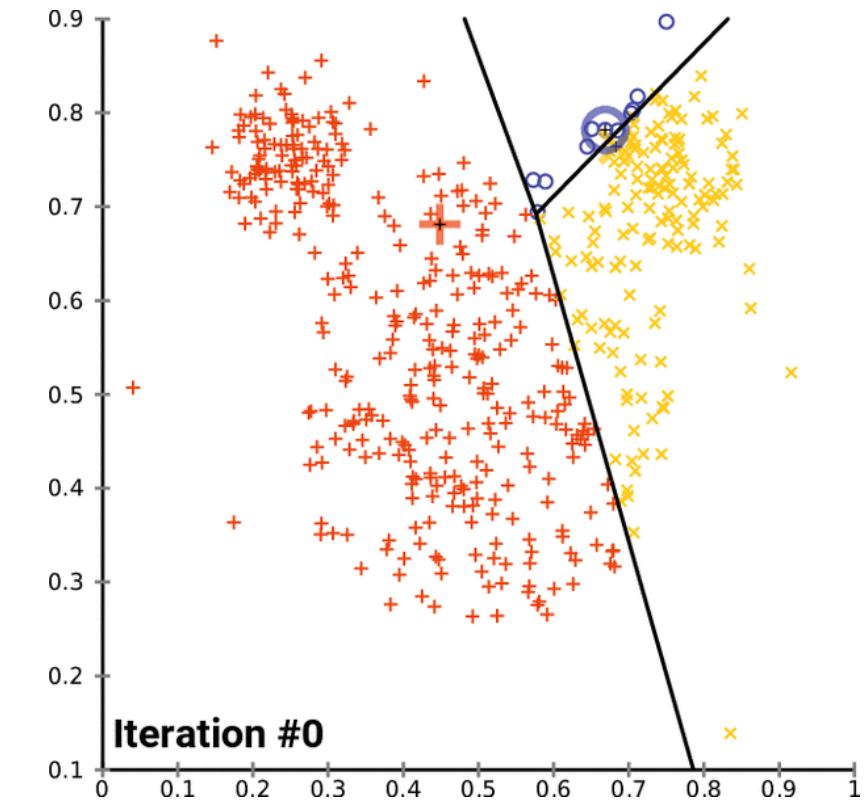
- Given the following dataset, can we group the points into 3 meaningful clusters that are sufficiently far from each other?



# K-Means Clustering



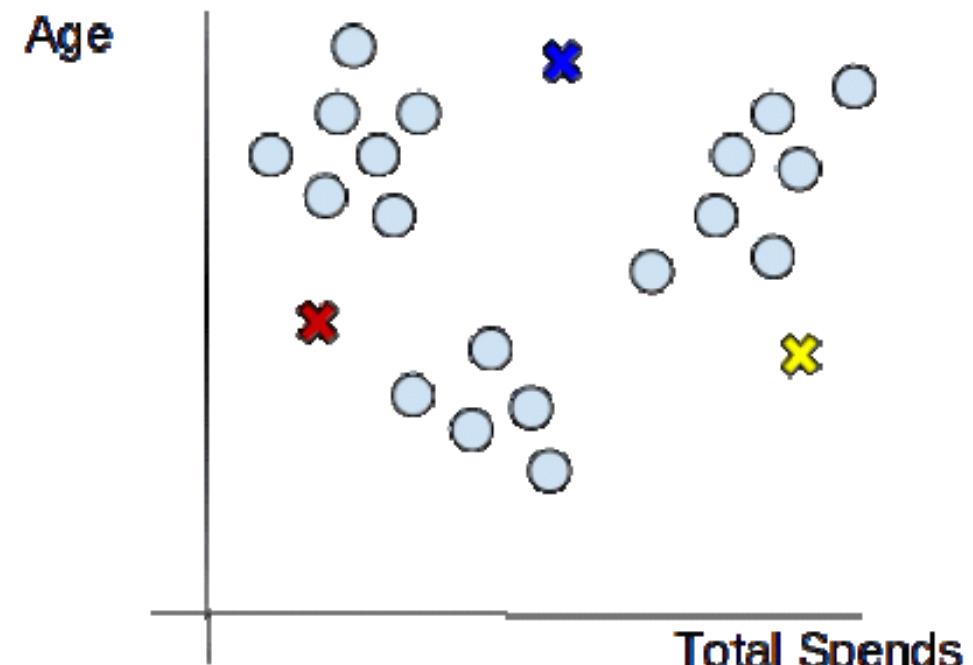
- K-Means is a popular unsupervised machine learning algorithm used for clustering data into groups or clusters based on similarity.
- The primary goal of K-Means is to partition data points into K clusters, where each point belongs to the cluster with the nearest mean.
- K is a hyperparameter manually set to determine the number of clusters.



# K-Means Clustering



- Assume we want to cluster the following dataset into 3 clusters where  $K = 3$ .
- Step 1: Initialize Clusters:
  - Choose a strategy to initialize the means (centers) of the 3 clusters.
    - A popular strategy is just to choose 3 random points to define the cluster means.
    - Other methods include: Naive Shardin and K-Means++



# K-Means Clustering

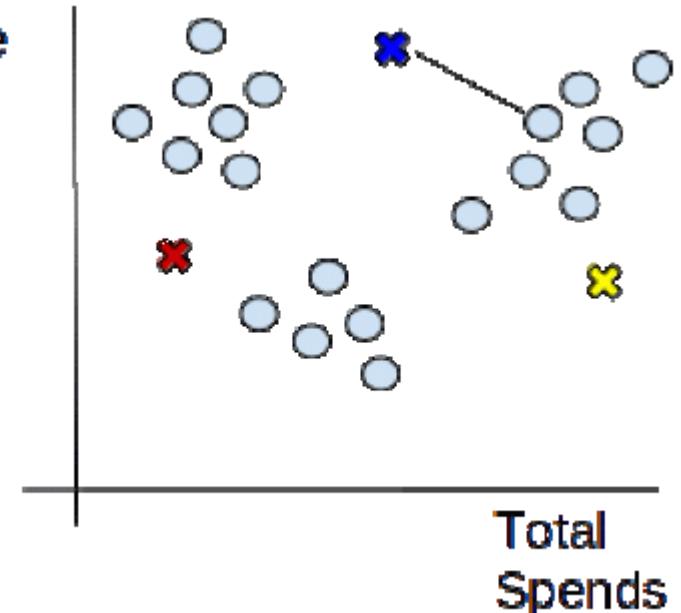


## ■ Step 2: Assign Points to Clusters:

- For each point in the dataset, calculate the distance between the points and the K-Cluster means (centers).

$$\text{▪ } d(x_i, \mu_k) = \sqrt{\sum_{d=1}^D (x_i^d - \mu_k^d)^2}$$

- Assign the point to the cluster closest to it (smallest distance).
- Repeat this step until all points in the dataset are assigned successfully to a cluster.



# K-Means Clustering

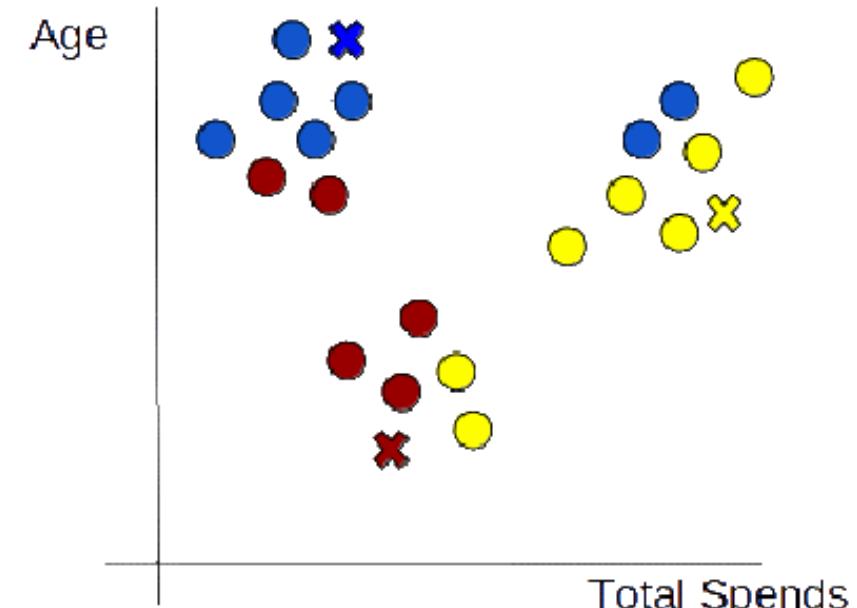


## ■ Step 3: Re-calculate Cluster Means:

- Now since different points belong to each cluster, we need to recalculate the cluster means (centers), such that:

$$\mu_k = \frac{1}{n_k} \sum_{i=0}^{n_k} x_i$$

- Where  $\mu_k$  represents the mean of cluster  $k$  as the mean of the points belonging to the cluster.
- $n_k$  represents the number of points belonging to cluster  $k$ .

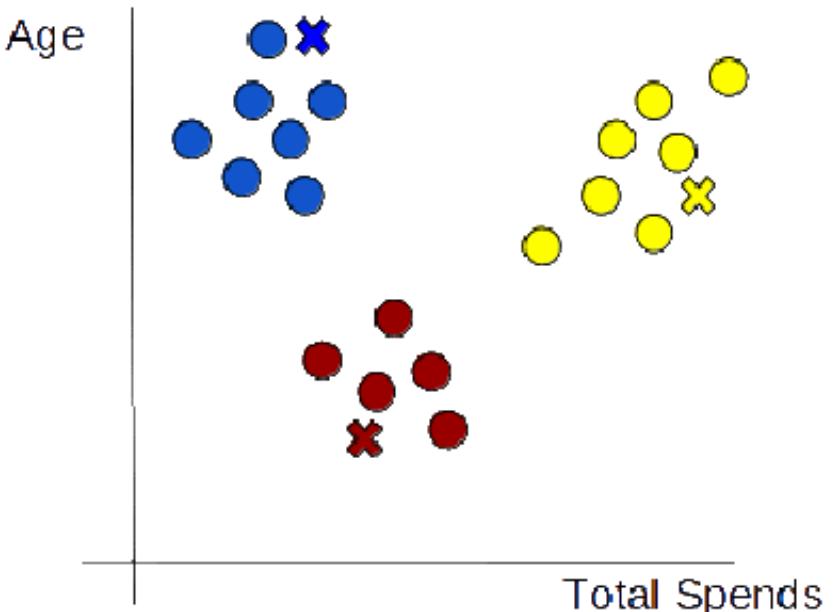


# K-Means Clustering



- Step 4: Repeat steps 2 and 3:

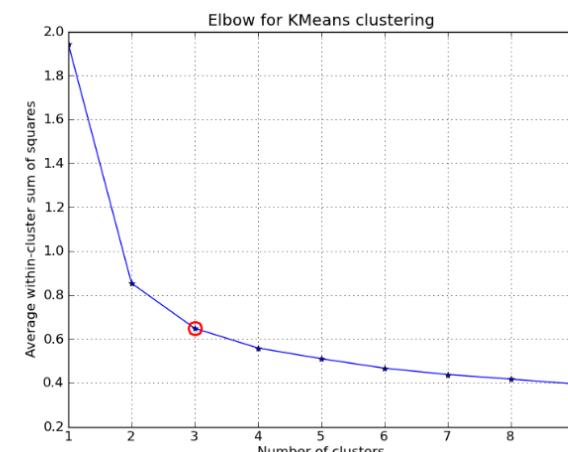
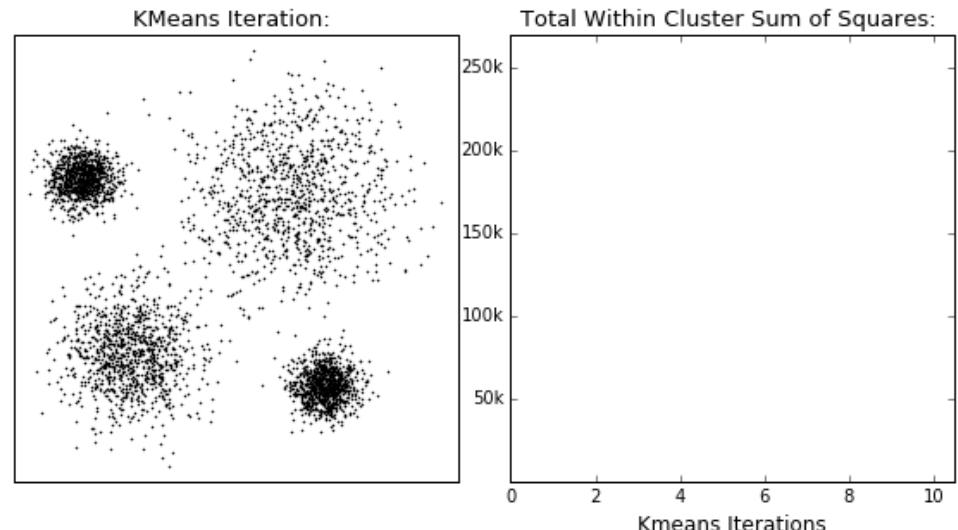
- Repeat steps 2 and 3 until the assignment of points to clusters is not changing anymore (saturation).
- The final assignment of points to clusters will define the optimal  $K$  for the current dataset.



# K-Means Clustering



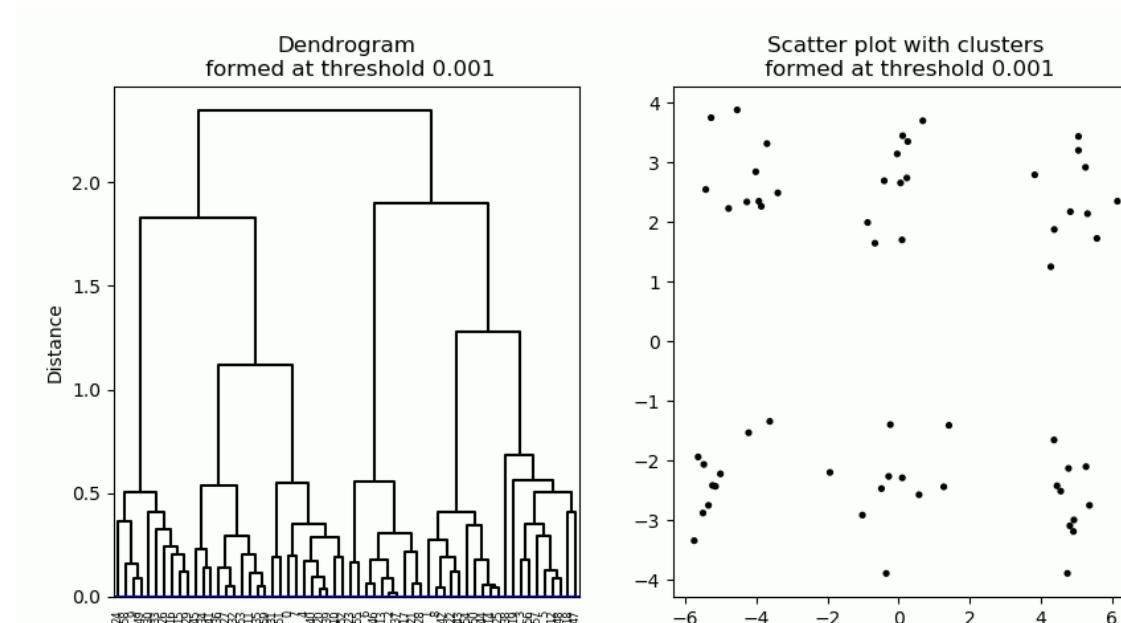
- How to choose optimal  $K$ ?
  - One method is called the Elbow Plot:
    - Calculate the Within-Cluster-Sum-of-Squares (WCSS) for different values of  $K$ :
$$WCSS_K = \sum_{i=1}^K \sum_{j=1}^{n_i} (x_j - \mu_i)^2$$
    - Where  $n_i$  represents the number of points in cluster  $i$ .
    - The WCSS tells us how spread the points in each cluster are. Lower WCSS means better clusters (more compact).
    - Plot the WCSS for different  $K$  values and choose the  $K$  value where an inflection point (elbow) occurs.



# Hierarchical Clustering



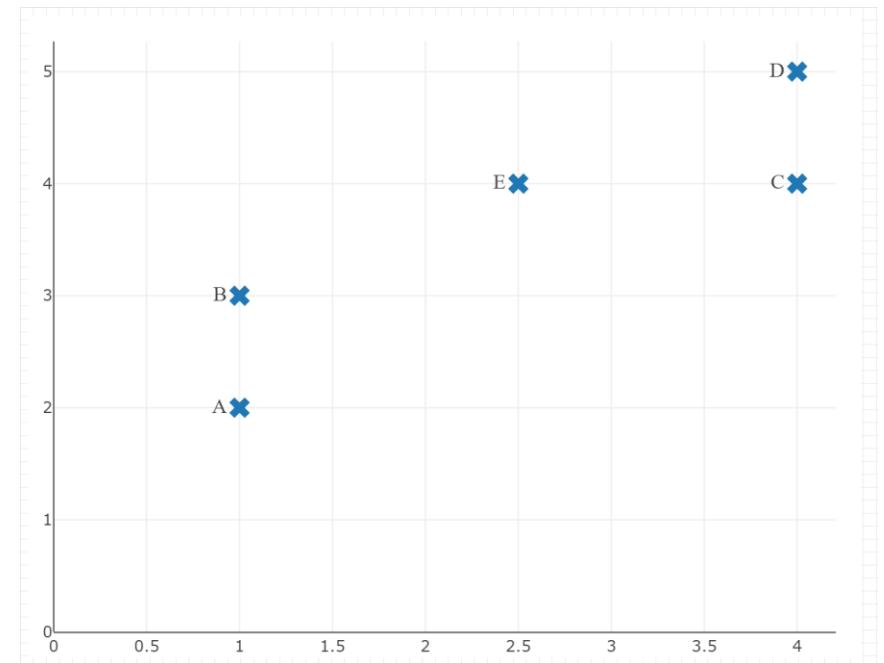
- Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters.
- There are two main types of hierarchical clustering: **Agglomerative** and **Divisive**.



# Agglomerative Hierarchical Clustering



- This method is "bottom-up," meaning it starts with each data point as a separate cluster and iteratively merges them into larger clusters.
- Assume that we have the example dataset with 5 points, we will use Agglomerative clustering to combine them into clusters.

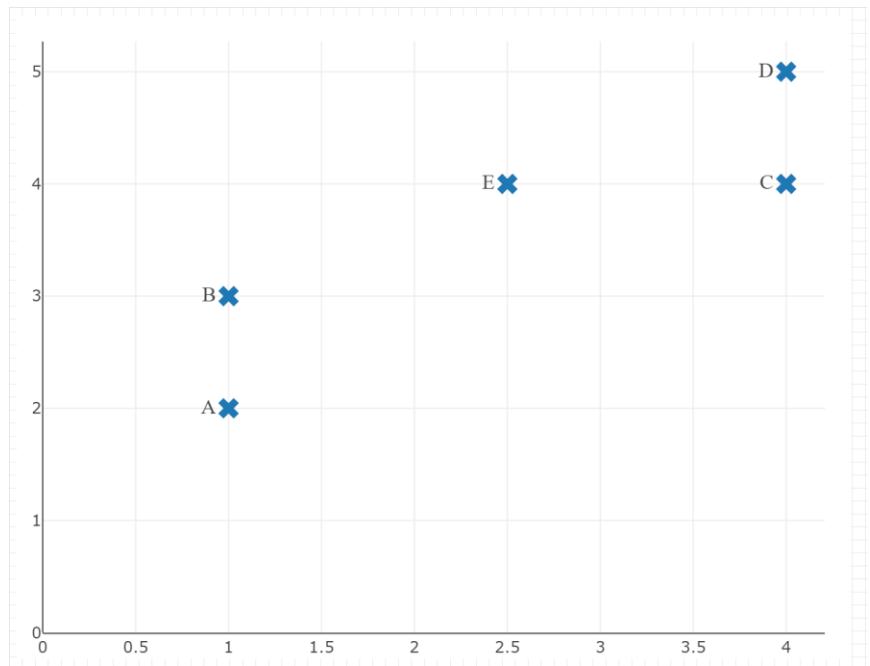


# Agglomerative Hierarchical Clustering



## ■ Step 1: Initialization:

- All points are treated as their own clusters. So we start with  $N$  clusters.
- In our example, we start with 5 clusters.



# Linkage Criterions



- They determine how the distance between clusters is measured, which directly influences how the clusters are formed. The types are:

- **Single Linkage (Nearest Neighbor):** Uses the minimum distance between members of the two clusters.

- $$d(A, B) = \min_{\{a \in A, b \in B\}} d(a, b)$$

- **Complete Linkage:** Uses the maximum distance between members of the two clusters.

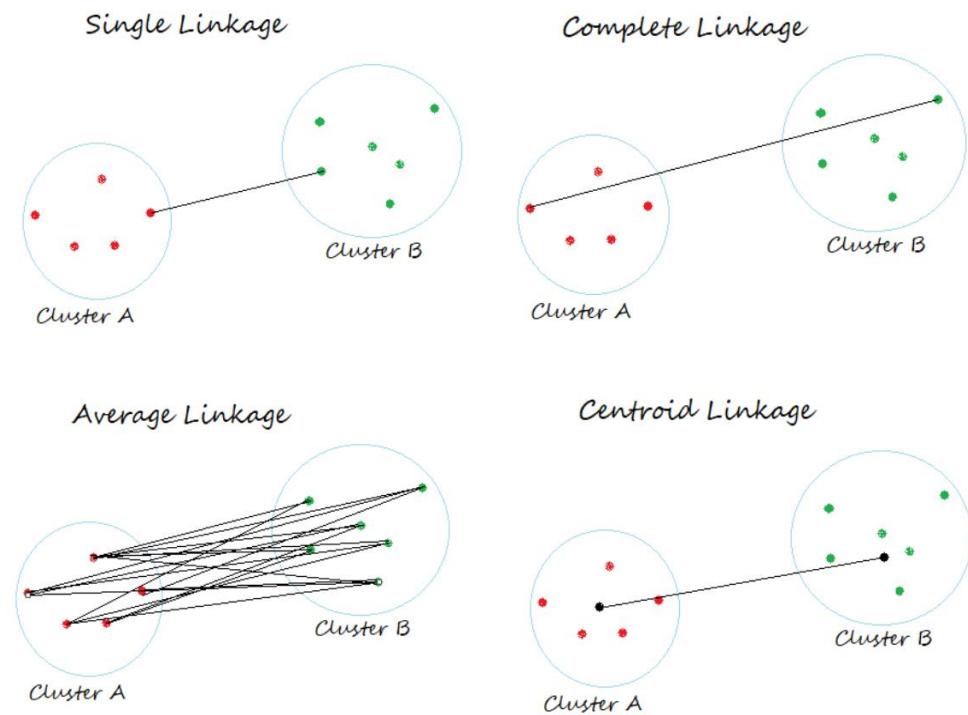
- $$d(A, B) = \max_{\{a \in A, b \in B\}} d(a, b)$$

- **Average Linkage:** Uses the average distance between all pairs of members in the two clusters.

- $$d(A, B) = \frac{1}{N_A N_B} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

- **Centroid Linkage:** Uses the distance between cluster means.

- $$d(A, B) = \|\mu_A - \mu_B\|^2$$

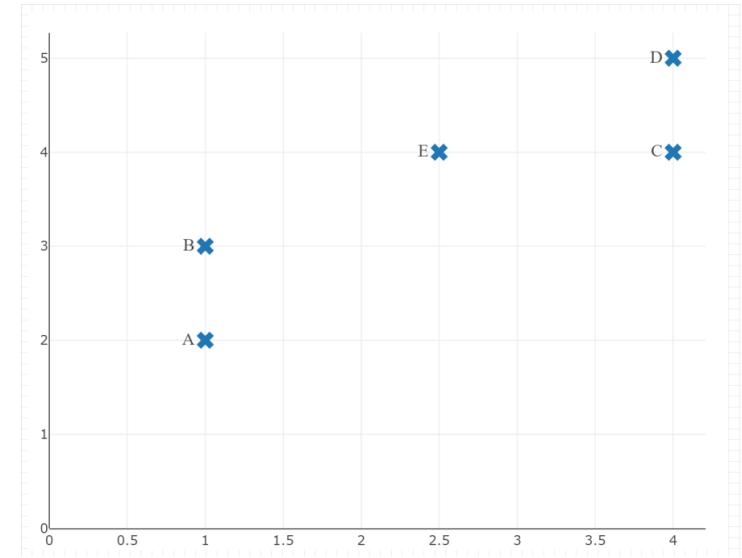


# Agglomerative Hierarchical Clustering



## ■ Step 2: Distance Matrix Computation:

- Calculate a similarity or distance matrix that measures the distances between all pairs of data points. This matrix is  $N * N$  size.
- You could use any distance metric discussed in class
  - Euclidean is the most famous
  - We will use Manhattan in this example
- Any of the linkage criterion could also be used, we will use Complete Linkage in this example.



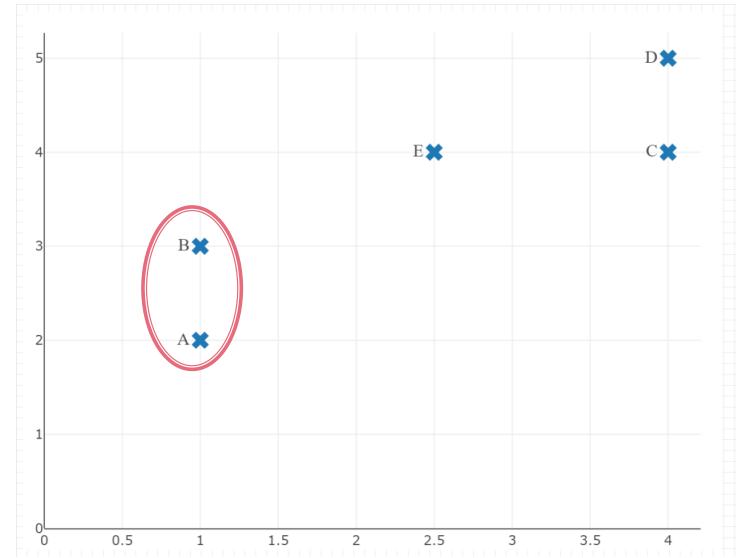
|   | A   | B   | C   | D   | E |
|---|-----|-----|-----|-----|---|
| A |     |     |     |     |   |
| B | 1   |     |     |     |   |
| C | 5   | 4   |     |     |   |
| D | 6   | 5   | 1   |     |   |
| E | 3.5 | 2.5 | 1.5 | 2.5 |   |

# Agglomerative Hierarchical Clustering



## ■ Step 3: Merge Closest Cluster:

- Find the pair of clusters that are closest to each other based on the chosen distance metric and linkage criterion.
- In our example, A-B and C-D show the minimum distances, we could choose any of them to be merged. We'll go with A-B

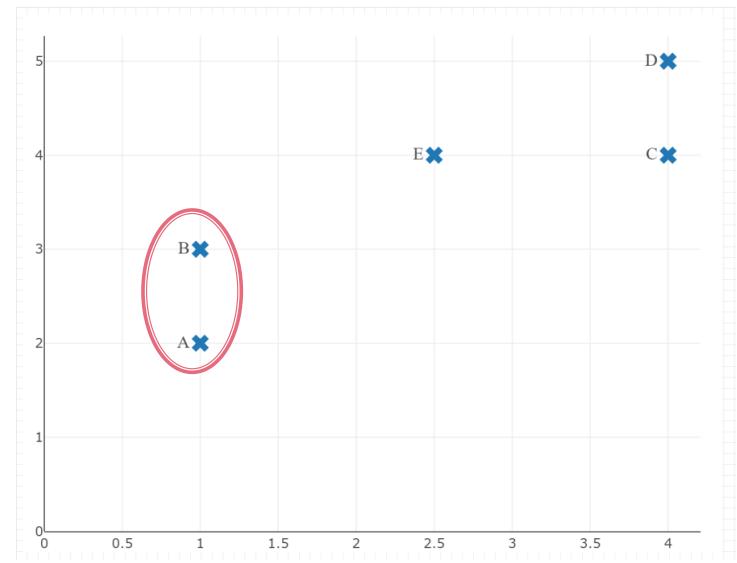


|   | A   | B   | C   | D   | E |
|---|-----|-----|-----|-----|---|
| A |     |     |     |     |   |
| B | 1   |     |     |     |   |
| C | 5   | 4   |     |     |   |
| D | 6   | 5   | 1   |     |   |
| E | 3.5 | 2.5 | 1.5 | 2.5 |   |

# Agglomerative Hierarchical Clustering



- Step 4: Recalculate Distance Metric:
  - Recalculate the distance matrix with the new cluster.
  - Recall that we're using complete linkage and Manhattan distance.

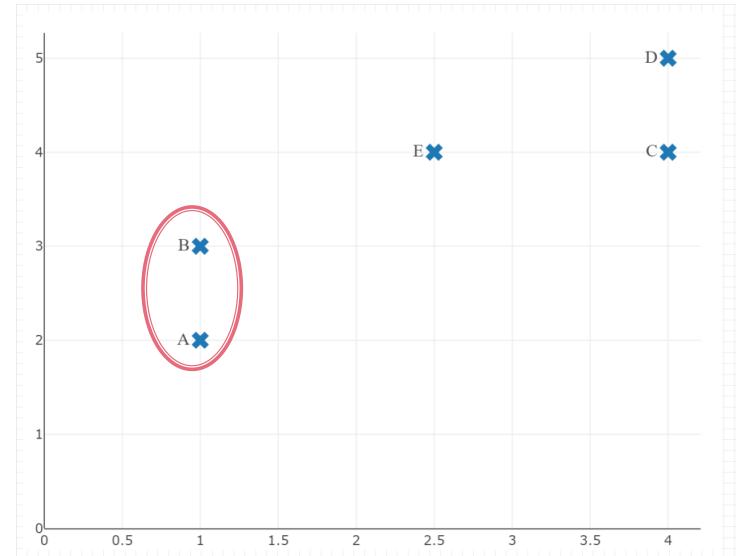


|     | A-B | C   | D   | E |
|-----|-----|-----|-----|---|
| A-B |     |     |     |   |
| C   | 5   |     |     |   |
| D   | 6   | 1   |     |   |
| E   | 3.5 | 1.5 | 2.5 |   |

# Agglomerative Hierarchical Clustering

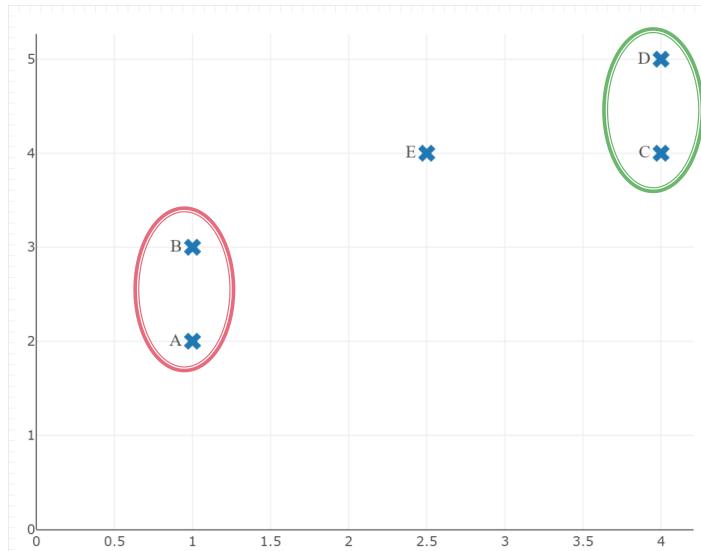


- Step 5: Repeat Steps 3 and 4:
  - Repeat steps 3 and 4 until all points are merged into one cluster.

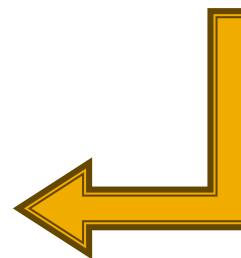
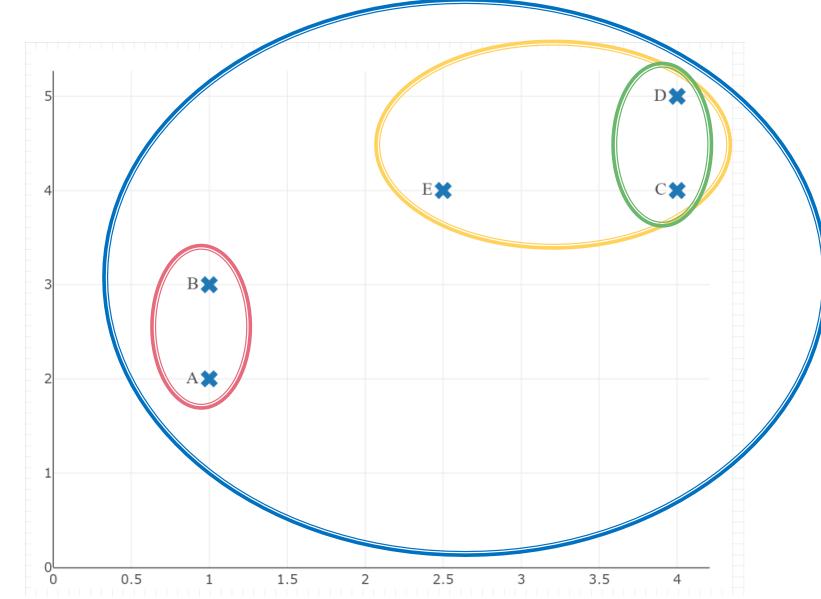
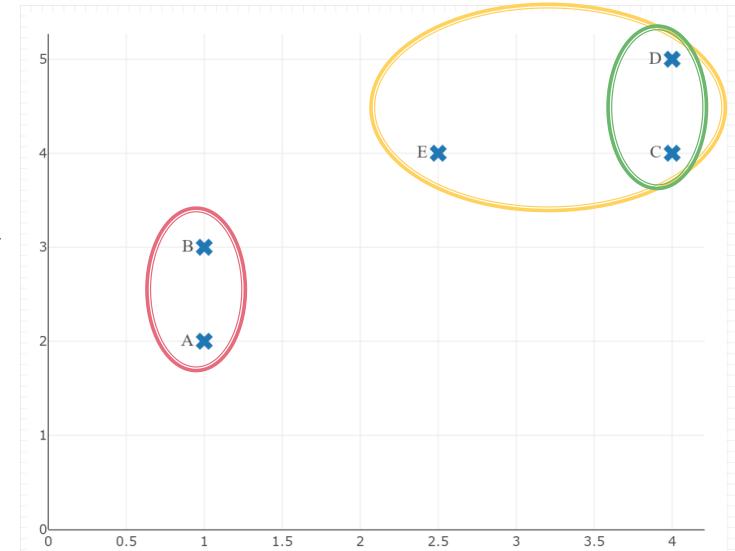


|     | A-B | C   | D   | E |
|-----|-----|-----|-----|---|
| A-B |     |     |     |   |
| C   | 5   |     |     |   |
| D   | 6   | 1   |     |   |
| E   | 3.5 | 1.5 | 2.5 |   |

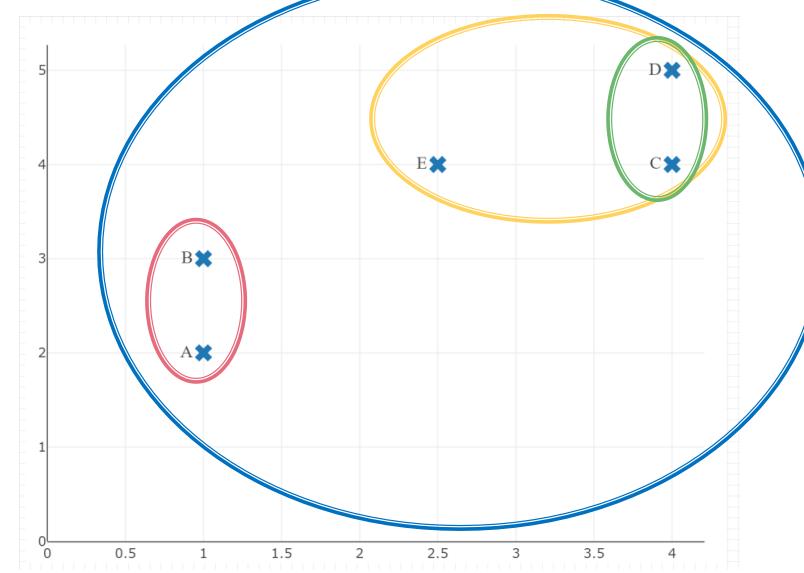
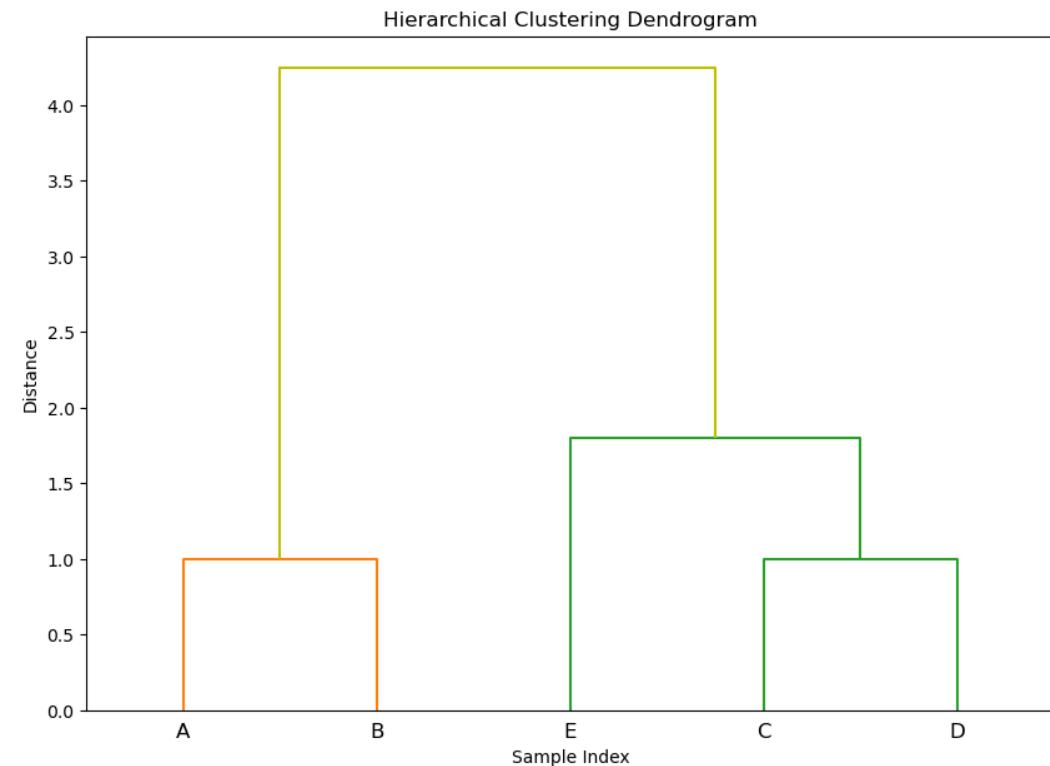
# Agglomerative Hierarchical Clustering



|     |     |     |   |
|-----|-----|-----|---|
|     | A-B | C-D | E |
| A-B |     |     |   |
| C-D | 6   |     |   |
| E   | 3.5 | 2.5 |   |



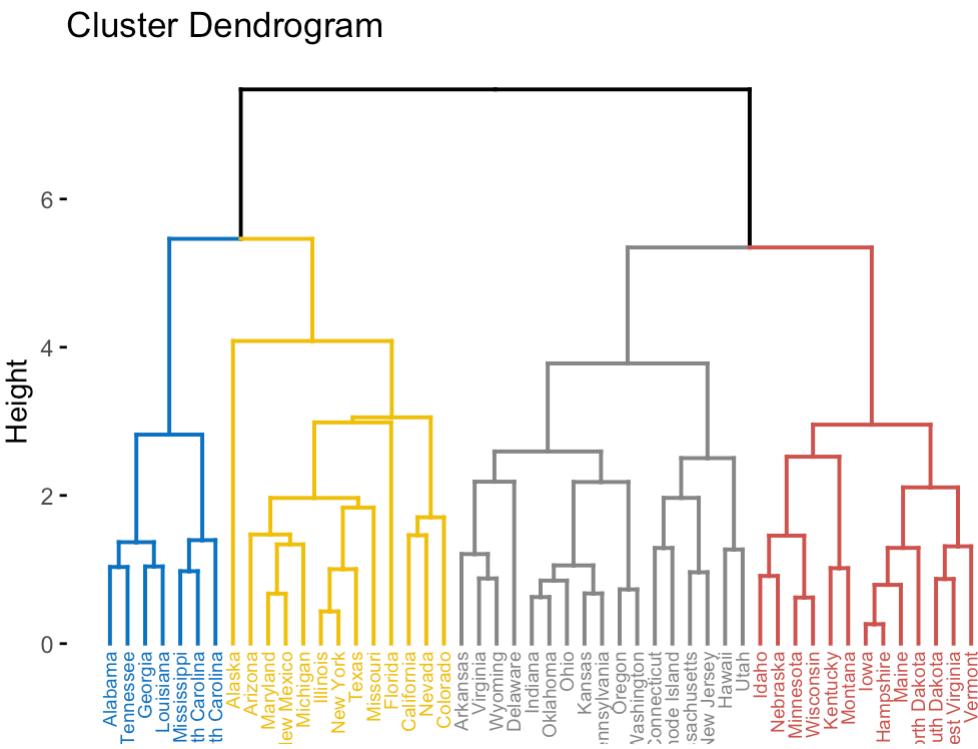
# Agglomerative Hierarchical Clustering



# Divisive Hierarchical Clustering



- This method is a "top-down" approach to cluster analysis. It begins with all data points in a single cluster and iteratively splits them into smaller clusters.
  - Steps:
    - Initialization:
      - Start with one large cluster that includes all data points.
    - Cluster Splitting:
      - At each step, split a cluster into smaller clusters.
      - The splitting is typically based on a criterion that identifies the 'least similar' members of the cluster.
    - Iterative Division:
      - Continue the process of splitting clusters at each step.
      - This process is repeated recursively until each data point forms its own cluster or a specified number of clusters is reached.
    - Result:
      - The result is often visualized as a dendrogram, which shows the hierarchical relationship between clusters and the order in which splits occurred.



# Thank you!



- Any questions?



# Disclaimer



Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution



**Thank You**  
**Youssef Abdelkareem**  
**[yabdelkareem@conestogac.on.ca](mailto:yabdelkareem@conestogac.on.ca)**