WM-ASDS 22: Machine Learning[1]

Dr. Md. Rezaul Karim
PhD(KULeuven & UHasselt), MS(Biostatistics, UHasselt), MS(Statistics, JU)
Professor, Department of Statistics and Data Science
Jahangirnagar University (JU), Savar, Dhaka - 1342, Bangladesh
Mobile: 01912605556, Email: rezaul@juniv.edu

Spring - 2024

Lecture Outline I

1 Chapter 1: Introduction

1.1 Textbook and References

1.2 Course Assessments

1.3 Problem & Motivation

1.4 What is Machine Learning?

1.5 What is Data?

1.6 About dataset

1.7 Types of Machine Learning

1.8 Challenges of machine learning

1.9 Practical Session

Lecture Outline II

2 Chapter 2: Python Overview and Data Exploration

2.1 What is Python?

2.2 Why should you learn Python?

2.3 Getting Python!

2.4 Jupyter Notebook

2.5 Pandas: Series

2.6 Multiple indexes

2.7 Pandas: Filtering

2.8 Pandas: Functions

2.9 Pandas: Series to dataframe

Lecture Outline III

2.10 Pandas: DataFrame

2.11 Data Processing

2.12 Data Concatenation

2.13 Data Grouping

2.14 Properties

2.15 Columns or fields

2.16 Rows or records

2.17 Methods

2.18 Reading Data

2.19 Importing Data

Lecture Outline IV

2.20 Sorting in `Pandas` Module

2.21 Covariance and Correlation Analysis

2.22 Linear Regression Analysis

2.23 Polynomial Regression

2.24 Multiple Regression

2.25 Example: Boston Housing Dataset

3 Chapter 3: Naïve Bayes Classification

3.1 Problem & Motivation

3.2 Bayes Theorem

3.3 Conditional Probability

Lecture Outline V

3.4 Naive Bayes Classification

3.5 Practical Example

3.6 Practical Session: Naive Bayes Classification in Python and R

4 Chapter 4: Decision Tree, CART Algorithm and Random Forest

4.1 Problem & Motivation

4.2 Decision Trees

4.3 Important Terminology related to Decision trees

4.4 Advantages of ID3 Algorithm

4.5 Limitations of ID3 Algorithm

4.6 C4.5 and 5.0 Algorithms

Lecture Outline VI

4.7 C5.0 Algorithm

4.8 Classification and regression trees (CART) Algorithm

4.9 Random Forest

4.10 Python code: Decision Tree, CART Analysis and Random Forest

4.11 R code: Decision Trees

5 Chapter 5: K-Nearest Neighbors (KNN) Algorithm (and Quiz #1)

5.1 K-Nearest Neighbor (KNN) Algorithm

5.2 How does KNN work?

5.3 Distance Metrics Used in KNN Algorithm

5.4 Example

Lecture Outline VII

5.5 Advantages and Disadvantages of KNN Algorithm

5.6 Python Code: KNN algorithm

6 Chapter 6: Evaluating a Machine Learning Model

6.1 Machine Learning Model Evaluation

6.2 Validation (or Holdout) Approach

6.3 ROC Curve and AUC Curve for classification

6.4 Cross Validation

6.5 Stratified k-fold cross-validation

6.6 Python code for k fold cross-validation.

Lecture Outline VIII

7 Chapter 7: Regression Analysis for Prediction

7.1 Problems & Motivation

7.2 Problem & Motivation

7.3 Scatter Diagram

7.4 Correlation Coefficient

7.5 Regression Analysis

7.6 Concept of Regression Analysis

7.7 Types of parametric regression analysis

7.8 Simple Linear Regression Model

7.9 A Probabilistic View of Linear Regression

Lecture Outline IX

7.10 Distributional Assumptions Underlying Regression

7.11 Simple Linear Regression Model

7.12 Multiple Linear Regression Model

7.13 Coefficient of Determination (R^2)

7.14 Adjusted R^2

7.15 The t-test for Individually Significance

8 Chapter 8: Logistic Regression Classifier

8.1 Problem & Motivation

8.2 Scatter Diagram

8.3 Correlation Coefficient

Lecture Outline X

8.4 Regression Analysis

8.5 Types of parametric regression analysis

8.6 Logistic Regression

8.7 Interpreting the Logistic Regression Equation

8.8 Multinomial Logistic Regression

8.9 Multinomial Logistic Regression Classifier

8.10 Geometric Intuition of Logistic Classifiers

8.11 Python code: Logistic Regression Classifier

8.12 R Code: Logistic Regression Analysis & Classification

9 Chapter 9: Support Vector Machine (SMV)

Lecture Outline XI

9.1 Problem & Motivation

9.2 Basic Algebra

9.3 SVM: Basic Concept

9.4 The Support Vector Classifier

9.5 Optimal Separating Hyperplanes

9.6 Computing the Support Vector Classifier

9.7 Support Vector Machines and Kernels

9.8 Applications of SVM

9.9 Python Code: Support Vector Machine

9.10 R Code: Support Vector Machine

Lecture Outline XII

10 Chapter 10: Feature Selection Techniques

10.1 Filter Methods

10.2 Wrapper Methods

10.3 Embedded Methods

10.4 Feature Selection Statistics

10.5 Boruta Feature Selection Algorithm

11 Chapter 11: Neural Network (NN) Modelling

11.1 Problem & Motivation

11.2 Basic Structure of ANN

Lecture Outline XIII

11.3 What is a neuron?

11.4 Activation Functions in NN

11.5 Perceptron

11.6 Multi-Layered Perceptron Model

11.7 Feed Forward Neural Networks

11.8 Backpropagation Algorithm

11.9 Overfitting vs. underfitting

11.10 Disadvantages of ANN

11.11 Vanishing Gradient Problem

11.12 Exploding Gradient Problem in Neural Network

Lecture Outline XIV

11.13 Drop Out Layers in Multi Neural Network

11.14 Python Code: Deep Neural Network

11.15 Example: Handwritten digits detection by Artificial Neural Network

12 Chapter 12: Clustering

12.1 Problem & Motivation

12.2 Hierarchical Clustering

12.3 The k-Means Clustering:Example (step by step)

12.4 Python Code: Clustering

12.5 R Code: Clustering

Lecture Outline XV

13 Chapter 13: Parallel Ensemble Learning — Bagging Methods

13.1 Problem & Motivation

13.2 Ensemble Learning

13.3 Bootstrapping

13.4 Bagging

13.5 Random Forest

13.6 Python Code: Bagging methods

13.7 R Code: Boasting Methods

14 Chapter 14: Sequential Ensemble Learning — Boosting Methods

Lecture Outline XVI

14.1 Problem & Motivation

14.2 Boosting Methods

14.3 AdaBoost Algorithm

14.4 Boosted Regression Tree

14.5 Python Code: Boasting methods

14.6 R Code: Boasting Methods

Chapter 1: Introduction

1 Chapter 1: Introduction

1.1 Textbook and References

1.2 Course Assessments

1.3 Problem & Motivation

1.4 What is Machine Learning?

1.5 What is Data?

1.6 About dataset

1.7 Types of Machine Learning

1.8 Challenges of machine learning

1.9 Practical Session

Prerequisites of the Course

- Fundamental knowledge of probability and linear algebra.
- Knowledge of Calculus, especially derivatives of single variable and multivariate functions.
- The ability to code in any computer language, especially in Python language or R language.

Textbook and References

Textbook

1. James G., Witten D., Hastie T., Tibshirani R., Taylor J. (2023): *An Introduction to Statistical Learning with Applications in Python*, 1st edition, Springer.
2. Andreas C. Müller, Sarah Guido (2016): *Introduction to Machine Learning with Python: A Guide for Data Scientists*, O'Reilly Media.
3. Hastie T., Tibshirani R., Friedman J. (2009): *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 3rd edition, Elsevier.

Reference list

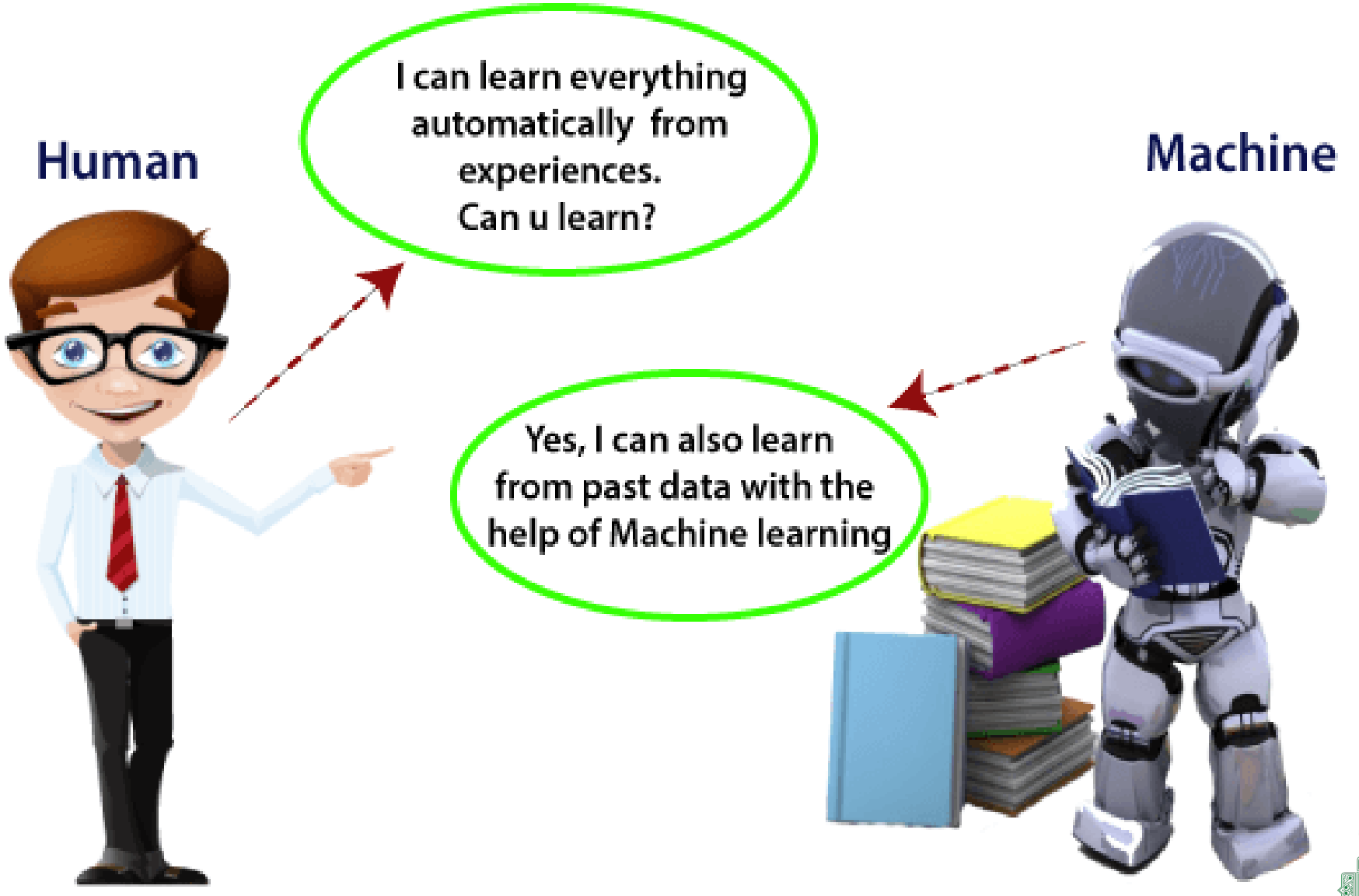
1. Dunham, M. H. (2003): *Data Mining: Introductory and Advanced Topics*, 1st edition, Pearson.
2. Zhao, Y. (2002): *R and Data Mining - Examples and Case Studies*, Academic Press, Elsevier

Course Assessments

Our department suggested implementing the following strategies in course assessments:

- Two (2) Quizzes and two (2) assignments, one midterm, and a final exam will be the components of Assessments.
  - Assignment 10%
  - Quiz 20% (Class Performance & Class Test)
  - Lab Test (with Attendance) 10%
  - Midterm 25%
  - Final 35%
- Midterm will be taken after the 6th lecture (in cases it can be extended upto 8th lecture) course contents are covered in classes.
- An average of two quizzes and an average of two assignments will be considered.

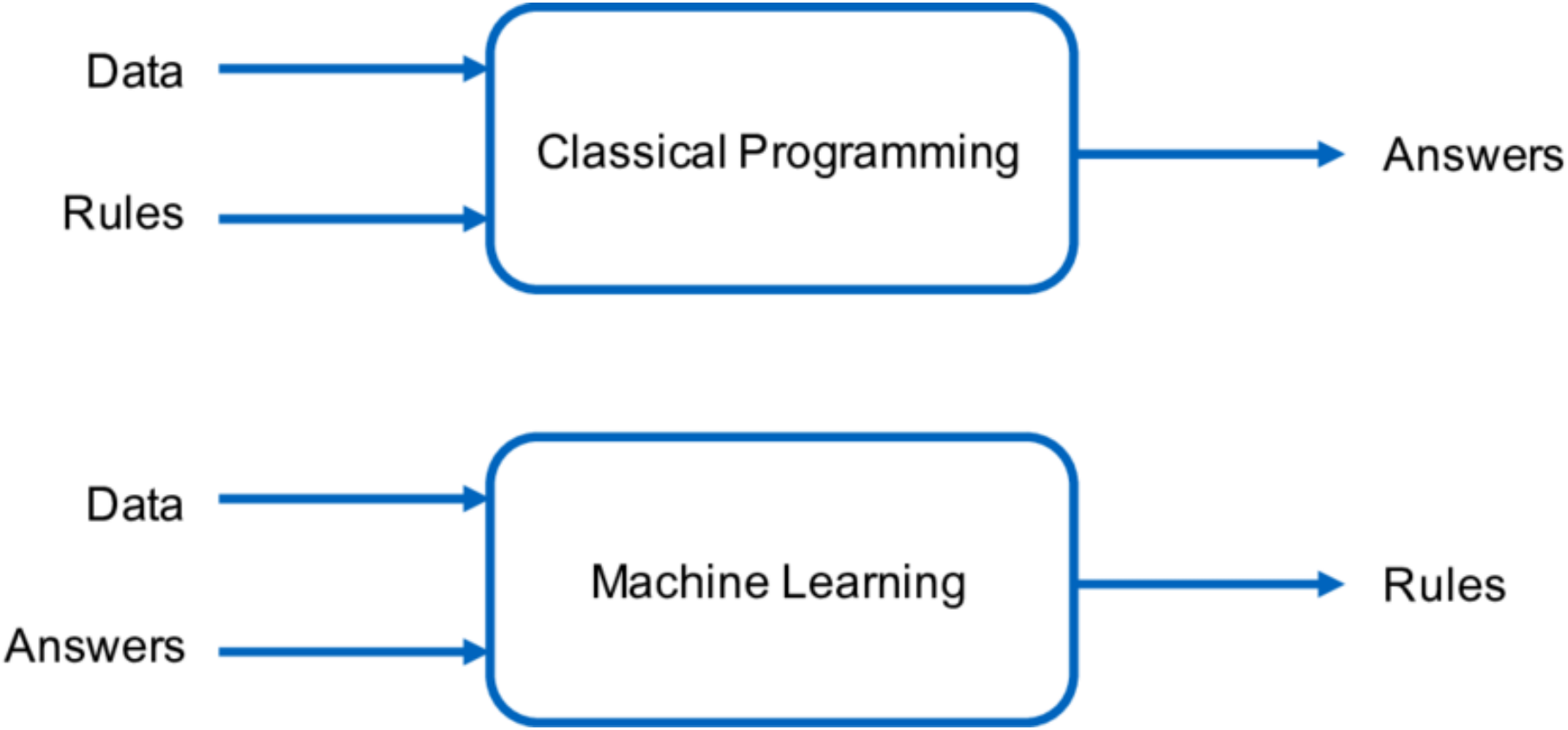
What is Machine Learning?



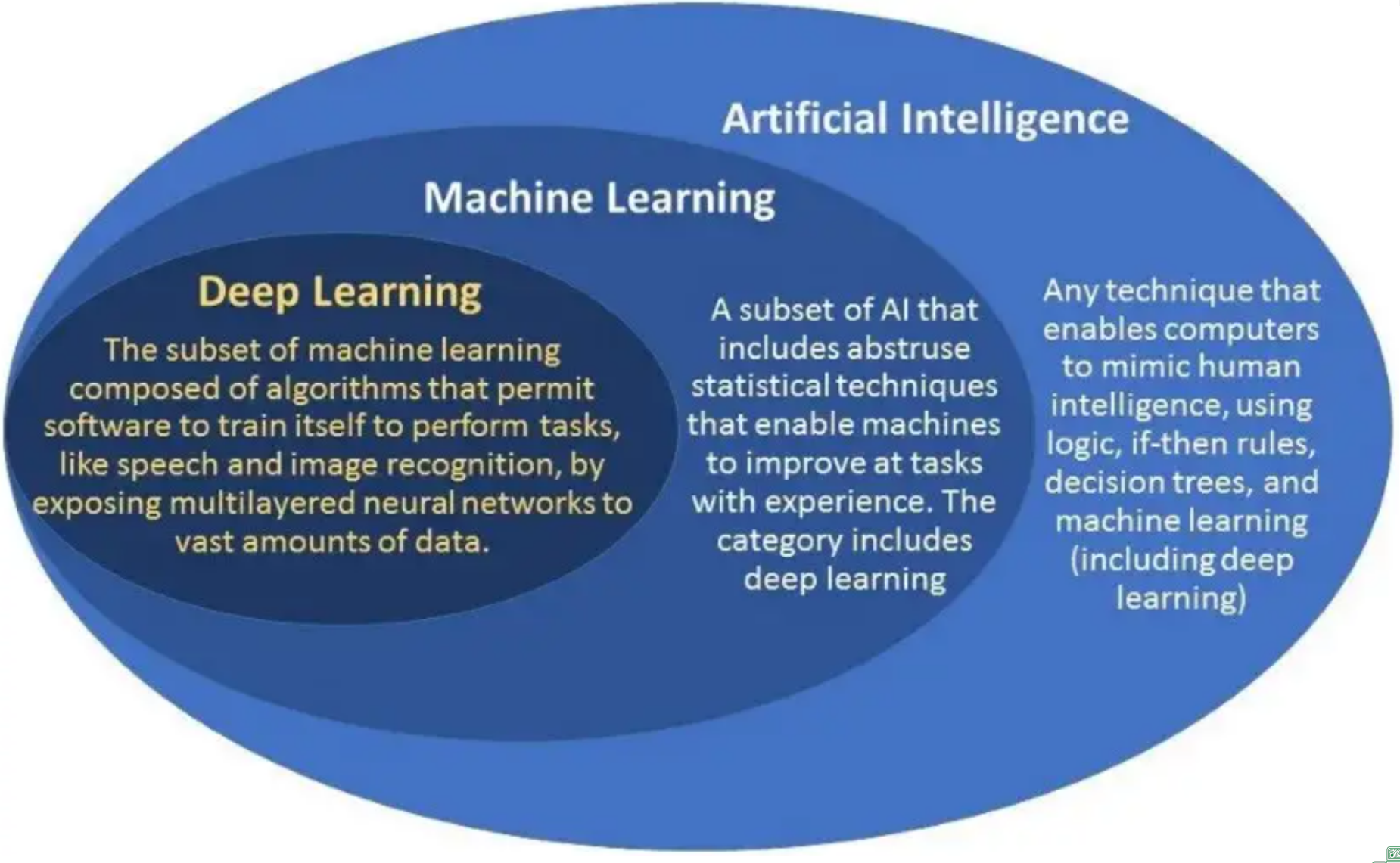
What is Machine Learning?

- Machine Learning (ML) is the kind of programming technology which gives computers the capability to automatically learn from past data without being explicitly programmed. This means in other words that these programs change their behaviour by learning from data.
- It is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy
- ML algorithms use historical data as input to predict new output values
- Currently, it is being used for various tasks such as
  - image recognition
  - speech recognition
  - email filtering
  - Facebook auto-tagging
  - recommender system
  - ...

Tradition Programming vs Machine Learning



Machine learning vs. Deep learning



Why is machine learning important?

- Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products.
- Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations.
- Machine learning has become a significant competitive differentiator for many companies.

Who is the father of machine learning?

- Geoffrey Everest Hinton is credited as the father of machine learning.
- Geoffrey Everest Hinton is the father of Deep Learning. Hinton's paper described the first deep learning algorithm that can achieve human-level performance on difficult and complex pattern recognition tasks.
- Considered by many to be the father of machine learning, Arthur Samuel created the first self-learning program in 1959 which could play checkers at a high level.
- Yoshua Bengio is a renowned computer scientist from Canada who has made significant contributions to the development of deep learning and artificial neural networks.
- John McCarthy is considered one of the pioneers of artificial intelligence, along with other notable figures such as Alan Turing, Marvin Minsky, Allen Newell, and Herbert A. Simon.

What is Data?

Data

- data are individual units of information
- these facts and figures are collected, analyzed, and summarized for presentation and interpretation
- all the data collected in a particular study are referred to as the data set for the study

Datum

- datum (a singular form of "data") is a single piece of information

Raw data

- data are collected by survey, census etc.
- it is known as *ungrouped data*

Note: Always we have raw data. We have to process or make data summaries by various statistical techniques (we will learn all by Chapters 2-3).

Types of data

1. quantitative data —ex: Family size, Score, Age
   - continuous data
     - Most numerical data in biomedical research are continuous or can be viewed as continuous. For instance, if we conduct a survey on the health and nutritional status of 7-year-old boys in a less developed region in 2020, the measurement results of their heights (cm), weights (kg), and hemoglobin (g/L) can be viewed as continuous data because their values can assume, in theory, any value in a certain range
   - discrete data
     - Discrete data occur when the data can only take certain values. The possible values of discrete data are generally integers. For instance, if we also collect data on the number of cases of cold (0, 1, 2, ...) in 2020 for the 7-year-old boys, then they are discrete data.
2. qualitative data (also called categorical data)
   - nominal-level data (ex: Gender, ID number, cell number, blood type)
   - ordinal-level data (Cancer stage, Grade, the treatment effect of a disease — cured, effective, improved, ineffective, and deteriorated.)

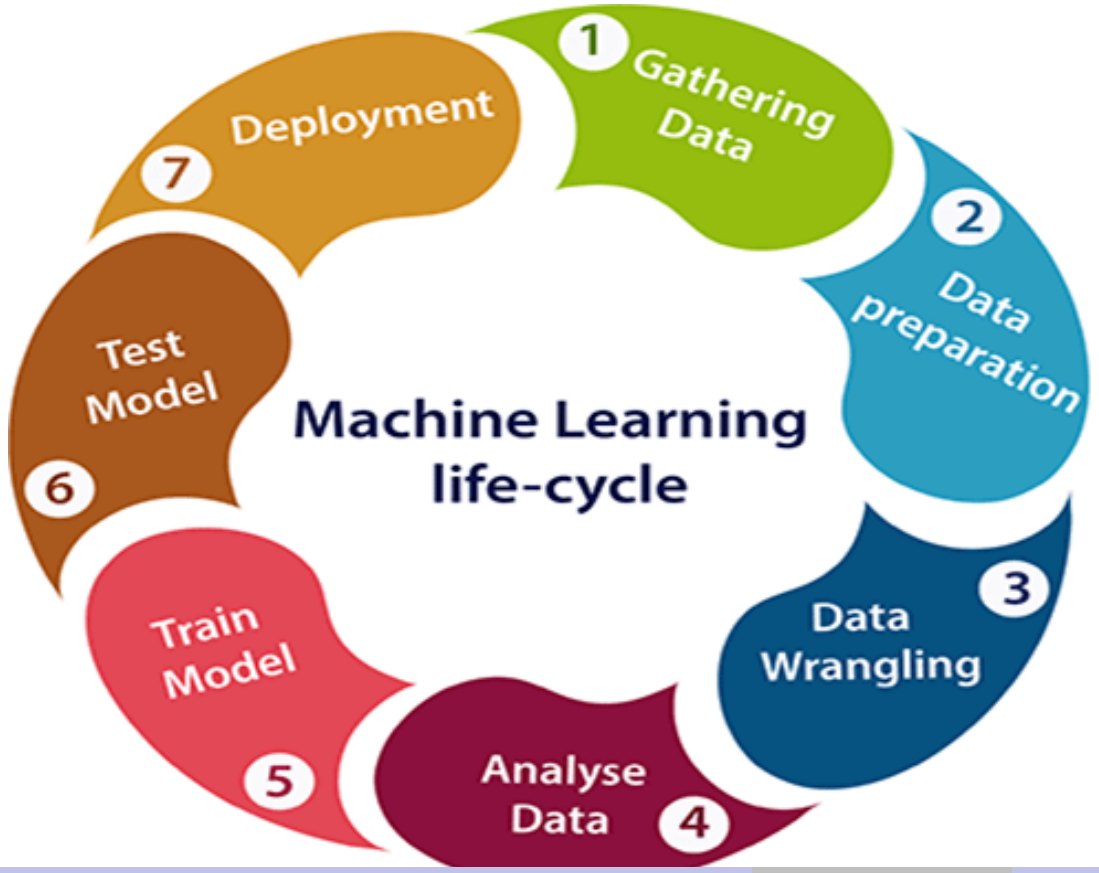
Classroom Exercise

What kind of data is the following?

1. hypertensive (1 for hypertensive and 0 for not hypertensive)
2. degrees of hypertension
   - Normal: Below SBP 120/DBP 80.
   - Elevated: SBP 120 to 129/DBP less than 80.
   - Stage 1 high blood pressure: SBP 130 to 139/DBP 80 to 89.
   - Stage 2 high blood pressure: SBP 140 and above/DBP 90 and above.
   - Hypertension crisis: SBP above 180/DBP above 120.
3. degree of the blood pressure (low, normal, high)
4. number of white blood cells (WBC) in 1 mm^3 of blood
5. hemoglobin level
6. number of leaves on a plant
7. number of giraffes visiting a water hole
8. number of eggs laid by a grasshopper

Machine learning Life cycle (How does Machine Learning work?)

The main purpose of the life cycle is to find a solution to the problem or project. Machine learning life cycle involves seven major steps, which are given below:



1. Gathering Data:
   - Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.
   - The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.
   - This step includes the below tasks:
     - identify various data sources
     - collect data
     - integrate the data obtained from different sources

2. Data preparation
   - After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.
   - In this step, first, we put all data together, and then randomize the ordering of data.
   - This step can be further divided into two processes:
     - Data exploration:
       • It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.
       • A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
     - Data pre-processing:
       • Now the next step is preprocessing of data for its analysis.

3. Data Wrangling
   - Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.
   - It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:
     - Missing Values
     - Duplicate data
     - Invalid data
     - Noise

     So, we use various filtering techniques to clean the data.
   - It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

4. Data Analysis
   Now the cleaned and prepared data is passed on to the analysis step. This step involves:
   - Selection of analytical techniques
   - Building models
   - Review the result

   The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model.

5. Train Model
   - We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Test Model
   - Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.
   - Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7. Deployment
   - The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.
   - If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system.
   - But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Train and Test datasets in Machine Learning

- Train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.
- There is no fixed rule while selecting the size of the training set and testing set. Its all about trial and error, so try out different ratios 80-20, 70-30, 65-35 and pick one that gives the best performance result.
- The amount of data you need depends both on the complexity of your problem and on the complexity of your chosen algorithm.
- For splitting the dataset, we can use the `train_test_split` function of `scikit-learn`. The bellow line of code can be used to split dataset:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y,
    test_size= 0.2, random_state=42)
```

Python Code Explanation

Explanation:

In the first line of the above code, we have imported the train_test_split function from the **sklearn** library.

In the second line, we have used four variables, which are

- x_train: It is used to represent features for the training data
- x_test: It is used to represent features for testing data
- y_train: It is used to represent dependent variables for training data
- y_test: It is used to represent independent variable for testing data
- In the train_test_split() function, we have passed four parameters. Which first two are for arrays of data, and test_size is for specifying the size of the test set. The test_size may be .5, .3, or .2, which tells the dividing ratio of training and testing sets.

Example 2: Python code for Iris Dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split
# Data Loading
url='https://gist.githubusercontent.com/curran/a08a1080b88344b0c
/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv'
df = pd.read_csv(url)
df.head()
x = df[['sepal_length', 'sepal_width', 'petal_length',
    'petal_width']]
y = df['species']
x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size=0.25, random_state =42)
```

Random Number Seed

- A random number seed is used to initialize a pseudorandom number generator (PRNG). A PRNG is a deterministic algorithm that produces a sequence of numbers that appears to be random. The sequence is determined by an initial value, which is the seed.
- The linear congruential generator (LCG) equation is given by:

$$X_{n+1} = (aX_n + c) \mod m$$

In this equation:

- X_{n+1} is the next pseudorandom number in the sequence.
- X_n is the current pseudorandom number in the sequence.
- a is the multiplier. a should be a primitive root modulo m. A primitive root is an integer a such that the powers of a generate all the positive integers less than m without repetition.
- c is the increment. Small values of c are often preferred.
- m is the modulus. m should be a large prime number or a power of 2 to ensure a long period.

- Choosing appropriate values for a, c and m is crucial to obtaining a good quality pseudorandom sequence. Poor choices can result in short periods (the sequence repeats quickly) or undesirable correlations between generated numbers.
- Consider the linear congruential generator (LCG) with the parameters:

$$a = 7, \quad c = 3, \quad m = 10, \quad X_0 = 1$$

The LCG equation is:

$$X_{n+1} = (7X_n + 3) \mod 10$$

Calculating the first few values of the pseudorandom sequence:

$$X_1 = (7 \times 1 + 3) \mod 10 = 0$$

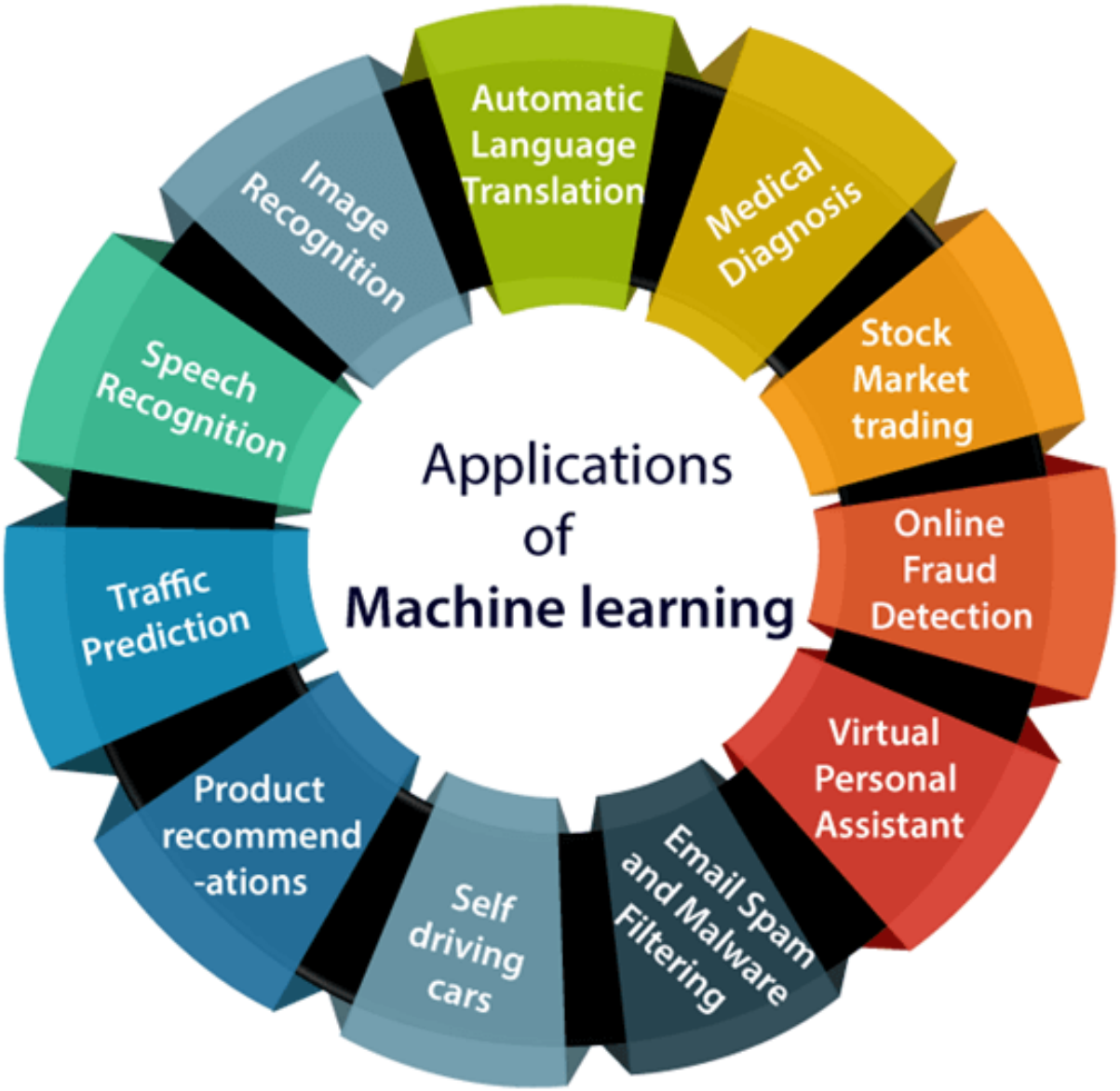
$$X_2 = (7 \times 0 + 3) \mod 10 = 3$$

$$X_3 = (7 \times 3 + 3) \mod 10 = 4$$

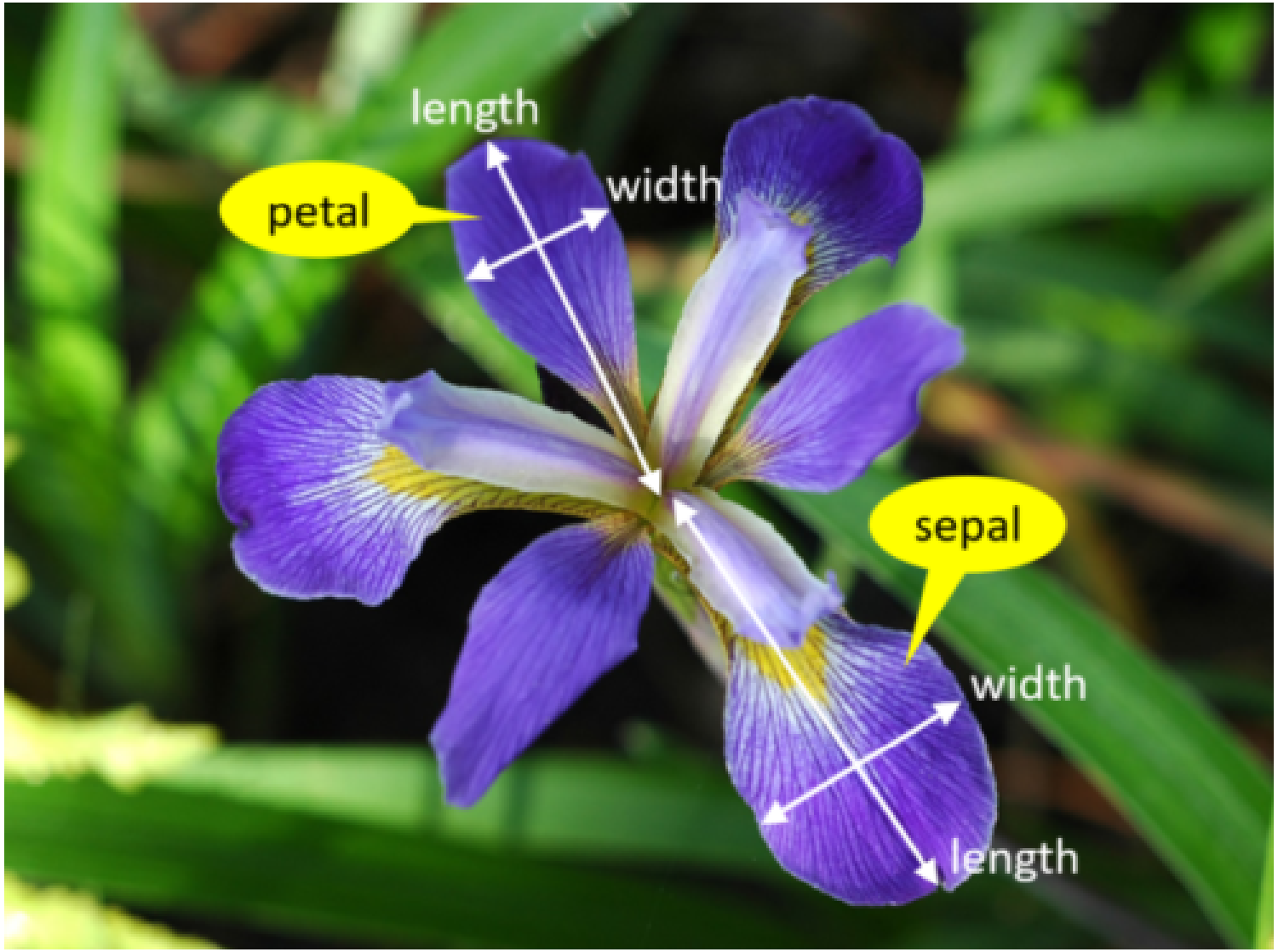
$$X_4 = (7 \times 4 + 3) \mod 10 = 1$$

And so on.

Applications of Machine learning



What's the Name of This Flower?



Iris Flower

Data Matrix for iris data

	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

What are the different types of machine learning?

There are four basic approaches:

1. supervised learning
2. unsupervised learning
3. semi-supervised learning
4. reinforcement learning

The type of algorithm data scientists choose to use depends on what type of data they want to predict.

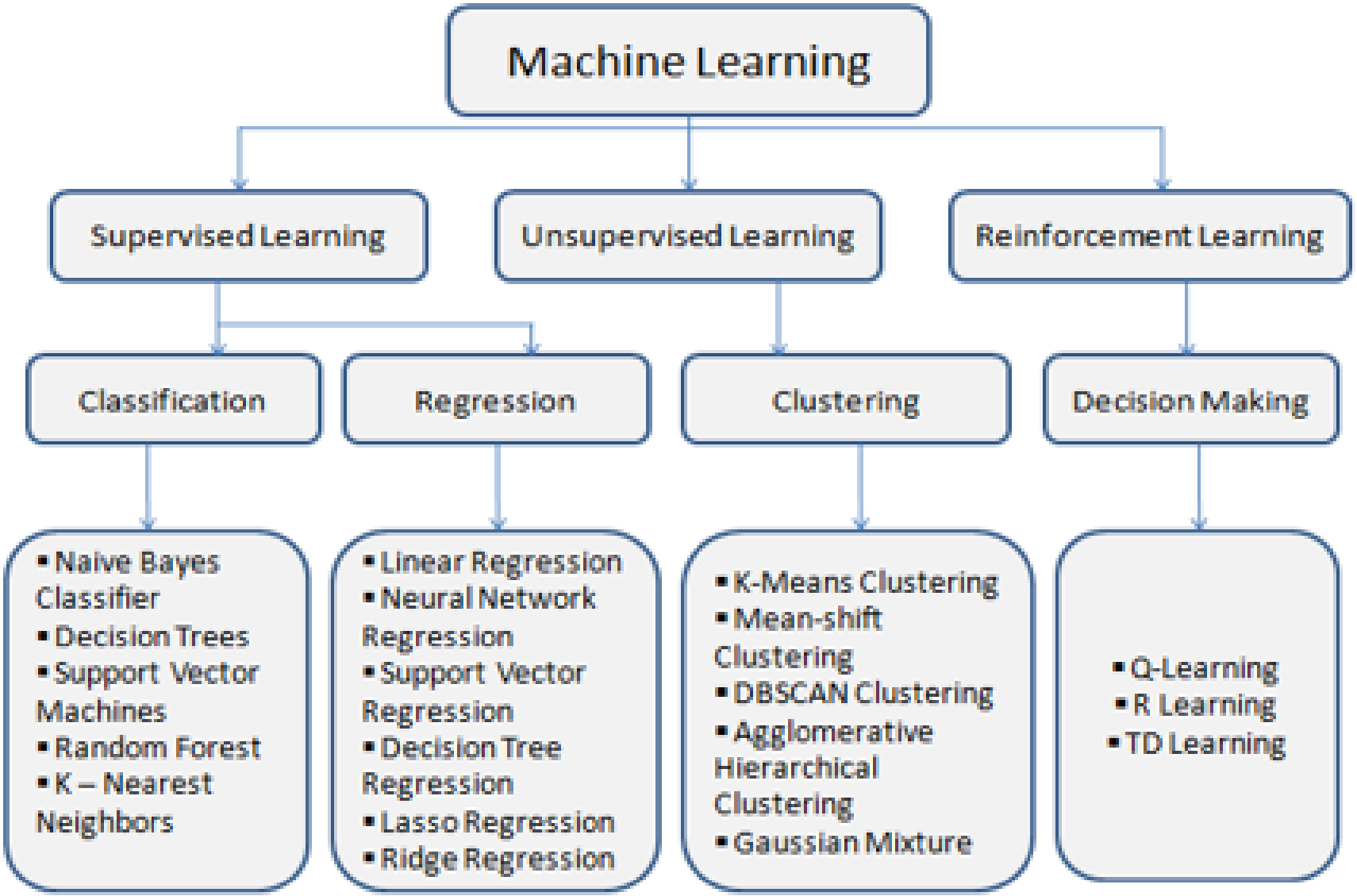
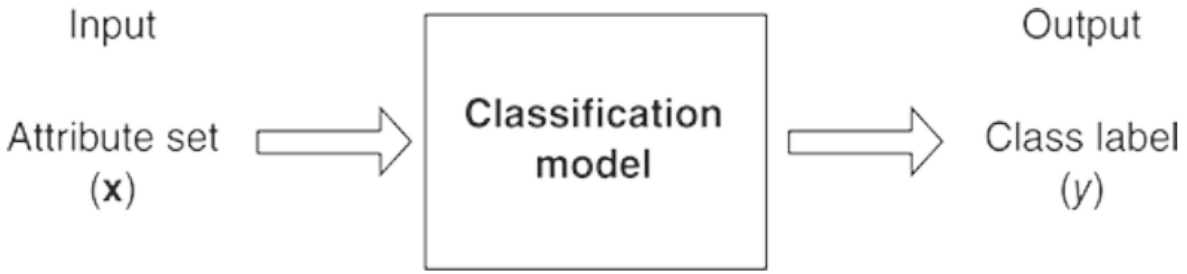


Fig: 2 Classification of Machine Learning algorithms

Supervised Machine Learning

supervised learning

- is the learning of the model where with input variable (say, x) and an output variable (say, Y) and an algorithm to map the input to the output, i. e., $Y = f(x)$
- the basic aim is to approximate the mapping function so well that when there is a new input data (x) then the corresponding output variable can be predicted
- why supervised learning?
  — it is called supervised learning because the process of an learning (from the training dataset) can be thought of as a teacher who is supervising the entire learning process



Examples of supervised learning

Task	Attribute set	Class label
Spam filtering	Features extracted from email message header and content	spam or non-spam
Tumor identification	Features extracted from magnetic resonance imaging (MRI) scans	malignant or benign
Galaxy classification	Features extracted from telescope images	elliptical, spiral, or irregular-shaped

Supervised Machine Learning

- Supervised learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms to classify data or predict outcomes accurately.
- The machine learning program is both given the input data and the corresponding labelling. This means that the learn data has to be labelled by a human being beforehand.
- Supervised learning helps organizations solve a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.
- Some methods used in supervised learning include
  - naïve bayes
  - neural networks
  - decision trees
  - random forest
  - support vector machine (SVM)
  - boosting and bagging
  - ...

Unsupervised Machine Learning

unsupervised learning

- no labels are provided to the learning algorithm. The algorithm has to figure out the a clustering of the input data.
- the main aim of unsupervised learning is to model the distribution in the data in order to learn more about the data
- it is called so, because there is no correct answer and there is no such teacher (unlike supervised learning)
- algorithms are left to their own devises to discover and present the interesting structure in the data
- examples of unsupervised learning
  - cluster analysis
  - association mining
  - ...

Semi-supervised learning

- is a machine learning technique that uses a small portion of labeled data and lots of unlabeled data to train a predictive model
- during training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set

Reinforcement Machine Learning

- Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action.
- The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it.
- The goal of an agent is to get the most reward points, and hence, it improves its performance.
- Reinforcement learning solves a specific type of problem where decision-making is sequential and the goal is long-term, such as game-playing, robotics, etc.

Key terminologies used in ML

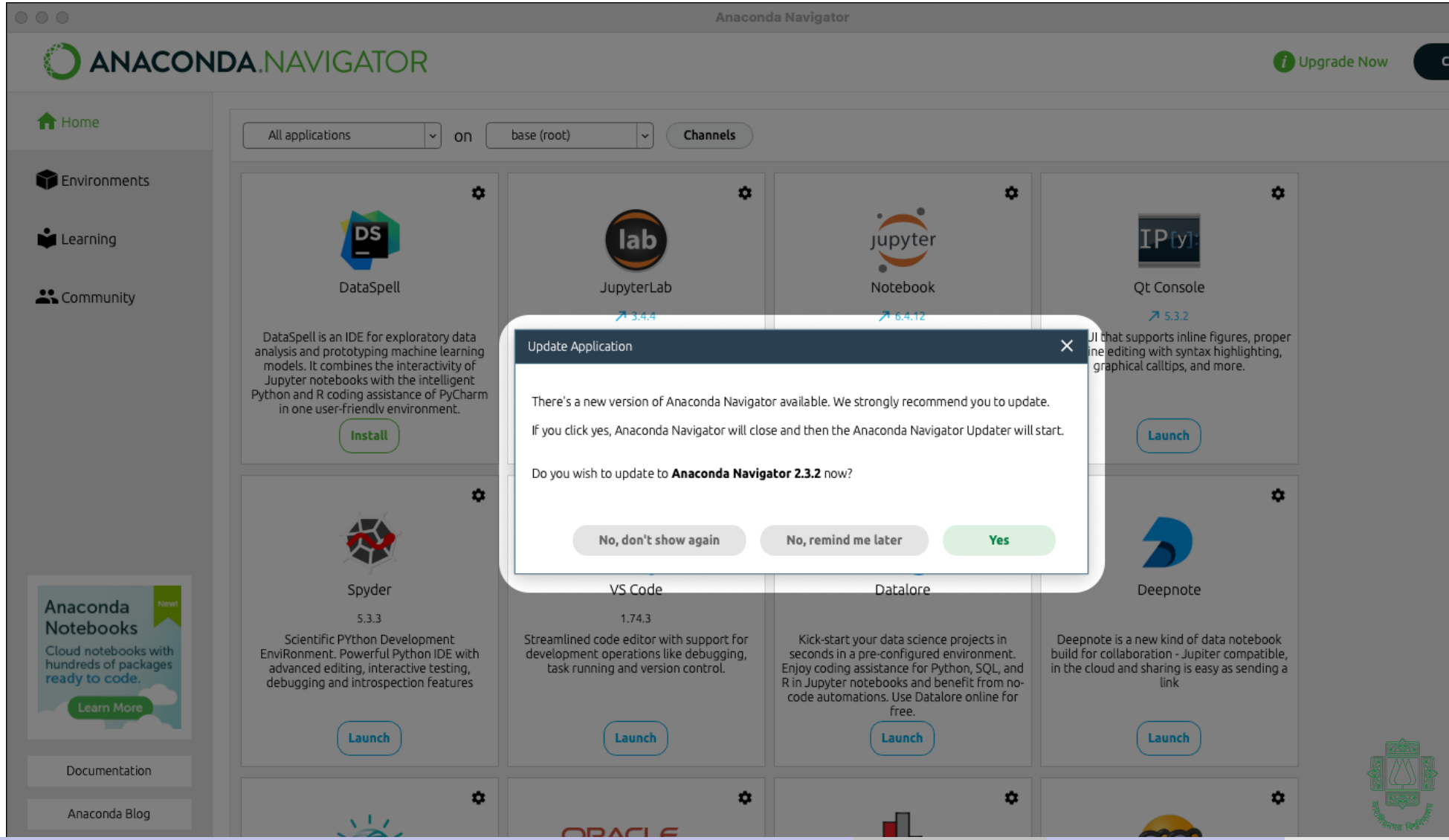
in machine learning, there are so many important terminologies that you should have a look at, some of which are as follows:

- Model
- Training
- Feature
- Target
- Prediction

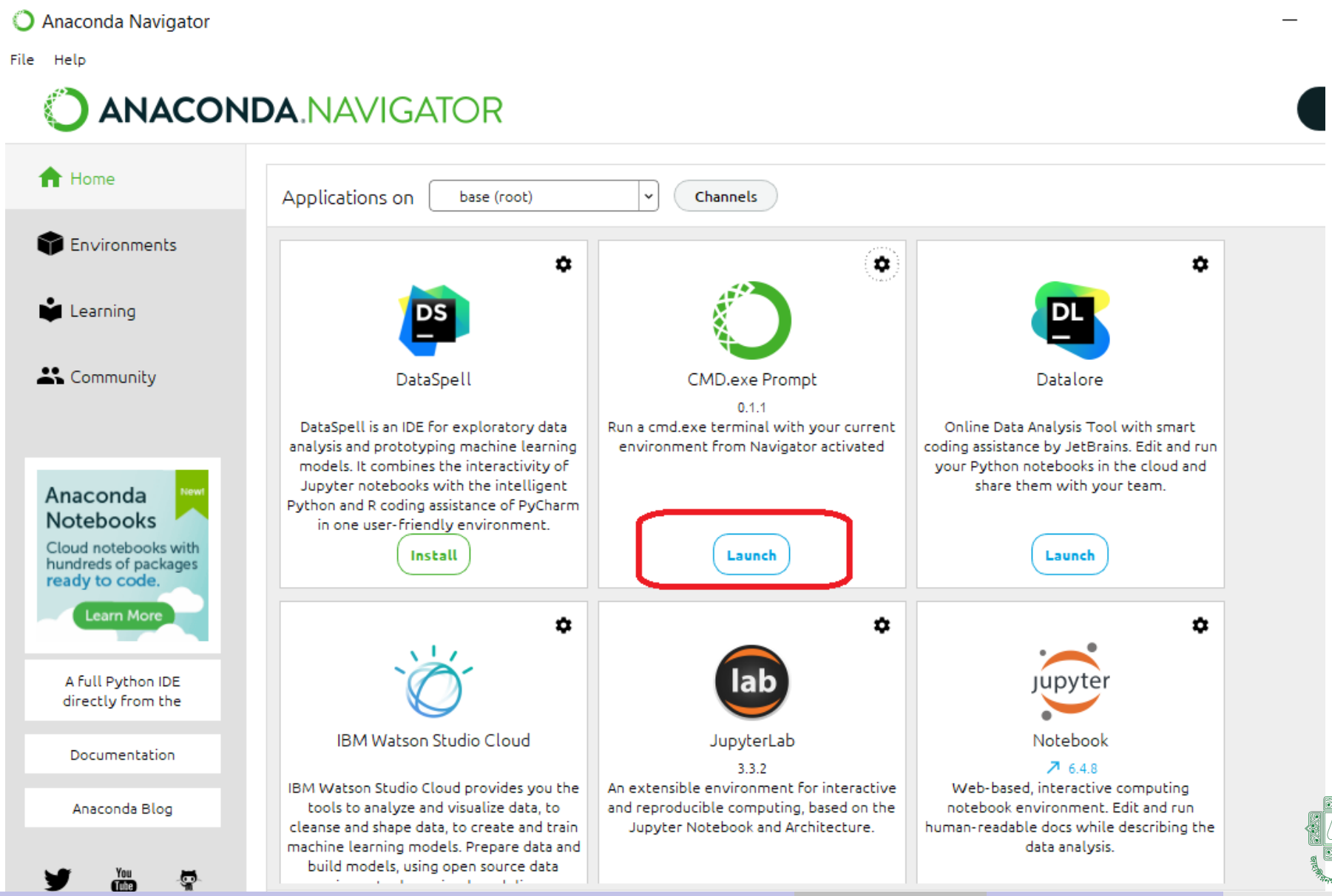
Challenges of machine learning

- technological singularity
- AI impact on jobs
- privacy
- bias and discrimination
- accountability

Practical Session



Manual update



6

Manual update

you can also update Anaconda Navigator manually through a command line interface (CLI). Open the Anaconda Prompt on Windows (or terminal on Linux or macOS) and use the following commands:

```
conda deactivate
conda update anaconda-navigator

# or
conda update --all
```

Chapter 2: Python Overview and Data Exploration

2 Chapter 2: Python Overview and Data Exploration

2.1 What is Python?

2.2 Why should you learn Python?

2.3 Getting Python!

2.4 Jupyter Notebook

2.5 Pandas: Series

2.6 Multiple indexes

2.7 Pandas: Filtering

2.8 Pandas: Functions

2.9 Pandas: Series to dataframe

2.10 Pandas: DataFrame

2.11 Data Processing

Python!

- is
  - a high-level programming language
  - an interpreted
  - an object-oriented
- is *fundamental* to data science and machine learning, as well as an everexpanding list of areas including cyber-security, and web programming
- is a language geared toward scientists and engineers who may not have formal software development training
- created in 1991 by Guido van Rossum
  — Named for Monty Python
- useful as a scripting language or glue language
- Filename extensions: .py,.pyi,.pyc,.pyd,.pyo (prior to 3.5),.pyw,.pyz (since 3.5)

Guido van Rossum is a Dutch programmer best known as the creator of the Python programming language.

- Why should you learn Python?
  - beginner friendly, it has simple syntax similar to the English language
  - write fewer lines than some other programming languages
  - Python works on different platforms (Windows, Mac, Linus, etc.)
  - used by many popular organization such as
    - google, yahoo!, youtube, NASA, etc.
    - many Linux distributions
    - games and apps (e.g. Eve Online)
- What is Python used for?
  - web development
  - desktop applications with PyQt, Gtk, wxWidgets and many more
  - mobile applications using kivy or BeeWare
  - Data Science and Visualization using *Numpy*, *Pandas* and *Matplotlib*
  - Machine learning with Tensorflow and Scikit-learn
  - artificial intelligence

Getting Python!

1. Basic Python 3.11.2 download (i.e., IDE then e.g. PyChanrm)
   `https://www.python.org/downloads/`
2. Python 3.7 from the "Anaconda Distribution"–Spyder, jupyter python
   `https://www.anaconda.com/download/`
3. Online Python (the best one!)
   `https://colab.research.google.com/notebooks/intro.ipynb`

CAUTION: the "basic" Python installation from https://www.python.org will NOT be sufficient for all the data-analysis!

How to Run Python Scripts From an IDE or a Text Editor?

- when developing larger and more complex applications, it is recommended that you use an integrated development environment (IDE) or an advanced text editor
- to execute your script, choose the Run or Build command, which is usually available from the tool bar or from the main menu
- Python's standard distribution includes IDLE as the default IDE, and you can use it to write, debug, modify, and run your modules and scripts
- other IDEs such as jupyter, Spyder, Eclipse-PyDev, PyCharm, Eric, and NetBeans also allow you to run Python scripts from inside the environment
- advanced text editors like Sublime Text and Visual Studio Code also allow you to run your scripts

Jupyter Notebook

two modes in Jupyter Notebook

1. command mode
2. edit mode

```
# See installed package
!pip list

# See the directiory
%pwd

# list of the folder in the directory
%ls

# See availabe line magics command
%lsmagic
```

Pandas Module

Pandas Module

- pandas stands for Python Data Analysis Library
- two of the main data structures in pandas are Series and DataFrame
- the series data structure is a glorified list with batteries included, and the DataFrame is a glorified table with extra, extra batteries included
- Pandas can present data in a way that is suitable for data analysis via its Series and DataFrame data structures
- it contains multiple methods for convenient data filtering
- it has a variety of utilities to perform Input/Output operations in a seamless manner
- it can read data from a variety of formats such as CSV, TSV, MS Excel, etc.

Pandas: Series

1. a series is similar to a one-dimensional array
2. it can store data of any type
3. the values of a Pandas `Series` are mutable but the size of a `Series` is immutable and cannot be changed
4. the first element in the `series` is assigned the index 0, while the last element is at index $N-1$, where N is the total number of elements in the series
5. to create a Pandas `Series`, we must first import the Pandas package via the Python's import command
6. How to create a simple series like $s = (1, 2, 3, 4)$?

Pandas: Series

1. how to create a simple series like $s = (1, 2, 3, 4)$?
2. to create the Series, we invoke the pd.Series() method and pass an array

```
import pandas as pd
s = pd.Series([1,2,3,4])
print(s)
```

```
In [1]: import pandas as pd

        s = pd.Series([1, 2, 3, 4])
        s

Out[1]: 0    1
        1    2
        2    3
        3    4
        dtype: int64
```

3. with every element in the list is a corresponding index

Pandas: Series

1. the index may not seem important at first glance, but it is very important later when you need to filter or slice the series item
2. if you do not specify the index for each element, a sequential and numeric one is created automatically (starting from zero)
3. How do you create a series with the index specified?

```
s = pd.Series([1, 2, 3, 4], index=[10, 11, 12, 13])
s
```

```
In [2]: s = pd.Series([1, 2, 3, 4], index=[10, 11, 12, 13])
        s

Out[2]: 10    1
        11    2
        12    3
        13    4
        dtype: int64
```

Pandas: Series

1. the index can also be string type

```
s = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
s
```

```
In [3]: s = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
        s

Out[3]: a    1
        b    2
        c    3
        d    4
        dtype: int64
```

Pandas: Multiple indexes

- a dataframe can have a multiple indices using MultiIndex

```
row_index = pd.MultiIndex.from_tuples([
('left', 'soccer'),
('left', 'baseball'),
('right', 'soccer'),
('right', 'baseball')])
col_index = pd.MultiIndex.from_tuples([
('amateur', 'male'),
('amateur', 'female'),
('Professional', 'male'),
('Professional', 'female')])
data = [
[5, 10, 15, 20],
[6, 11, 20, 25],
[10, 11, 25, 30],
[15, 17, 30, 35]
]
df = pd.DataFrame(data, index=row_index, columns=col_index)
```

Pandas: Multiple indexes

Out[6]:

		amateur		Professional	
		male	female	male	female
left	soccer	5	10	15	20
	baseball	6	11	20	25
right	soccer	10	11	25	30
	baseball	15	17	30	35

Filtering columns

---What about subsetting or slicing multi-index columns?

In [7]:
```
df.loc[:, df.columns.get_level_values(0).isin(['amateur'])]
```

Out[7]:

		amateur	
		male	female
left	soccer	5	10

Pandas: Filtering

1. Filtering is important. Let's see how to filter based on values.

```
s = pd.Series([1, 2, 3, 4])
s1 = s[s > 2]
s1
```

```
In [16]: s = pd.Series([1, 2, 3, 4])
         s1 = s[s > 2]
         s1

Out[16]: 2    3
         3    4
         dtype: int64
```

Pandas: Filtering

1. How do we filter based on index?

```
[11]: s2 = s[s.index < 2]
      s2
```

```
[11]: 0    1
      1    2
      dtype: int64
```

```
[12]: s2 = s[s.index.isin([0, 1])]
      s2
```

```
[12]: 0    1
      1    2
      dtype: int64
```

Pandas Series

1. You can access the `index` and `values` of a series individually

```
[4]: s.index
```

```
[4]: Index(['a', 'b', 'c', 'd'], dtype='object')
```

```
[5]: s.values
```

```
[5]: array([1, 2, 3, 4])
```

2. we can represent this data as a series

```
[6]: s = pd.Series([10, 15], index=['left', 'right'])
     s
```

```
[6]: left     10
     right    15
     dtype: int64
```

Pandas Functions

1. There are just too many useful functions attached to a series. Here's a few that are interesting.

```
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c',
    'd', 'e'])
x_bar = s.mean()
x_min = s.min()
x_max = s.max()
x_std = s.std()
print(x_bar);print(x_min);print(x_max),print(x_std)
```

```
In [15]: s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
         x_bar = s.mean()
         x_min = s.min()
         x_max = s.max()
         x_std = s.std()
         print(x_bar);print(x_min);print(x_max),print(x_std)

3.0
1
5
1.5811388300841898
```

Pandas: Functions

1. the `value_counts()` function can quickly give you the frequencies of unique values

```
s = pd.Series([1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 4])
s.value_counts()
```

```
[9]: s = pd.Series([1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 4])
     s.value_counts()

[9]: 4    4
     2    3
     3    2
     1    2
     dtype: int64
```

Pandas: Series to dataframe

1. We will talk about Pandas dataframes next, but for now, it's quite easy to convert a series to a dataframe using .to_frame(). Note that we have to supply the name so that the column name is meaningful.

```
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c',
    'd', 'e'], name='number')
s
```

Pandas: Series to dataframe

In [17]:
```
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'], name='number')
s
```

Out[17]:
```
a    1
b    2
c    3
d    4
e    5
Name: number, dtype: int64
```

In [18]:
```
s.to_frame()
```

Out[18]:

	number
a	1
b	2
c	3
d	4
e	5

Pandas: DataFrame

1. Dataframes are the most powerful data structure in pandas
2. as a start, it's helpful to simply think of a dataframe as a table
3. Below, we create a dataframe from a list of tuples, where each tuple is a pair of numbers. Notice how the columns are not specified and so they default to 0 and 1, and the index is not specified so a numeric, sequential one is created.

```
df = pd.DataFrame([(1, 2), (3, 4), (5, 6)])
df
```

In [20]:
```
df = pd.DataFrame([(1, 2), (3, 4), (5, 6)])
df
```

Out[20]:

	0	1
0	1	2
1	3	4
2	5	6

Pandas: DataFrame

1. If we wanted to specify the column names, supply a list of column names.

```
df = pd.DataFrame([(1, 2), (3, 4), (5, 6)],
    columns=['x', 'y'])
df
```

In [21]:
```
df = pd.DataFrame([(1, 2), (3, 4), (5, 6)], columns=['x', 'y'])
df
```

Out[21]:

	x	y
0	1	2
1	3	4
2	5	6

Pandas: DataFrame

1. If we need to specify the index, supply a list of index for each row.

```
df = pd.DataFrame([(1, 2), (3, 4), (5, 6)],
    columns=['x', 'y'], index=['a', 'b', 'c'])
df
```

```
In [22]: df = pd.DataFrame([(1, 2), (3, 4), (5, 6)], columns=['x', 'y'], index=[
         df
```

Out[22]:

	x	y
a	1	2
b	3	4
c	5	6

Pandas: DataFrame

1. There's other ways to create a dataframe. Let's use a list of dictionaries. Notice how we do not need to supply a list of column names?
2. it organizes data into rows and columns, making it a two-dimensional data structure
3. to create a DataFrame, you can choose to start from scratch or convert other data structures like Numpy arrays into a DataFrame
4. to create a DataFrame, we must invoke the pd.DataFrame() method

```
import pandas as pd
df = pd.DataFrame({
  'x': [1, 3, 4],
  'y': [2, 4, 5]
}, index=['a', 'b', 'c'])
print(df)
```

In [23]:
```
df = pd.DataFrame({
    'x': [1, 3, 4],
    'y': [2, 4, 5]
}, index=['a', 'b', 'c'])
df
```

Out[23]:

	x	y
a	1	2
b	3	4
c	4	5

In [24]:
```
df = pd.DataFrame([
    {'x': 1, 'y': 2},
    {'x': 3, 'y': 4},
    {'x': 5, 'y': 6}
], index=['a', 'b', 'c'])
df
```

Out[24]:

	x	y
a	1	2
b	3	4
c	5	6

Data Merging

```
import pandas as pd
d={'subject_id': ['1','2','3','4','5'], 'student_name':
    ['Rezaul', 'Sagor','Khairul', 'Gony', 'Shafi']}
Room435=pd.DataFrame(d,columns=['subject_id','student_name'])
print(Room435)


data = {'subject_id': ['4', '5', '6', '7', '8'],
  'student_name': ['Shafi', 'Rezaul', 'Zia', 'Gony',
      'Azhar']}
Room302 = pd.DataFrame(data, columns=['subject_id',
    'student_name'])
print(Room302)
pd.merge(Room435, Room302, on='subject_id')
```

In [32]:
```
d={'subject_id': [1,2,3,4,5], 'student_name': ['Rezaul', 'Sagor','Khairul', 'Gony', 'Shafi']}
```

In [62]:
```
import pandas as pd
d={'subject_id': ['1','2','3','4','5'], 'student_name': ['Rezaul', 'Sagor','Khairul', 'Gony', 'Shafi
Room435=pd.DataFrame(d,columns=['subject_id','student_name'])
print(Room435)
```

```
  subject_id student_name
0          1       Rezaul
1          2        Sagor
2          3      Khairul
3          4         Gony
4          5        Shafi
```

In [64]:
```
data = {'subject_id': ['4', '5', '6', '7', '8'],
'student_name': ['Shafi', 'Rezaul', 'Zia', 'Gony', 'Azhar']}
Room302 = pd.DataFrame(data, columns=['subject_id', 'student_name'])
print(Room302)
```

```
  subject_id student_name
0          4        Shafi
1          5       Rezaul
2          6          Zia
3          7         Gony
4          8        Azhar
```

In [65]:
```
pd.merge(Room435, Room302, on='subject_id')
```

Out[65]:

	subject_id	student_name_x	student_name_y
0	4	Gony	Shafi
1	5	Shafi	Rezaul

Data Concatenation

1. Concatenation of data, which basically means to add one set of data to another, can be done by calling the `concat()` function.

```
print(pd.concat([df1, df2]))
```

```
In [70]: print(pd.concat([Room435, Room302]))
```

	subject_id	student_name
0	1	Rezaul
1	2	Sagor
2	3	Khairul
3	4	Gony
4	5	Shafi
0	4	Shafi
1	5	Rezaul
2	6	Zia
3	7	Gony
4	8	Azhar

Data Grouping

```
# import pandas library
import pandas as pd

raw = {
  'Name': ['John', 'John', 'Grace', 'Grace', 'Benjamin',
      'Benjamin', 'Benjamin',
  'Benjamin', 'John', 'Alex', 'Alex', 'Alex'],
  'Position': [2, 1, 1, 4, 2, 4, 3, 1, 3, 2, 4, 3],
  'Year': [2009, 2010, 2009, 2010, 2010, 2010, 2011, 2012,
      2011, 2013, 2013, 2012],
  'Marks':[408, 398, 422, 376, 401, 380, 396, 388, 356, 402,
      368, 378]
}
df = pd.DataFrame(raw)

group = df.groupby('Year')
print(group.get_group(2010))
```

In [35]:
```
# import pandas library
import pandas as pd
raw = {'Name': ['John', 'John', 'Grace', 'Grace', 'Benjamin', 'Benjamin', 'Benjami
'Benjamin', 'John', 'Alex', 'Alex', 'Alex'],
'Position': [2, 1, 1, 4, 2, 4, 3, 1, 3, 2, 4, 3],
'Year': [2009, 2010, 2009, 2010, 2010, 2010, 2011, 2012, 2011, 2013, 2013, 2012],
'Marks':[408, 398, 422, 376, 401, 380, 396, 388, 356, 402, 368, 378]}
df = pd.DataFrame(raw)
df
```

Out[35]:

	Name	Position	Year	Marks
0	John	2	2009	408
1	John	1	2010	398
2	Grace	1	2009	422
3	Grace	4	2010	376
4	Benjamin	2	2010	401
5	Benjamin	4	2010	380
6	Benjamin	3	2011	396
7	Benjamin	1	2012	388
8	John	3	2011	356
9	Alex	2	2013	402
10	Alex	4	2013	368

In [46]:
```
group = df.groupby('Year')
print(group.get_group(2010))
```

```
       Name  Position  Year  Marks
1      John         1  2010    398
3     Grace         4  2010    376
4  Benjamin         2  2010    401
5  Benjamin         4  2010    380
```

In [48]:
```
print(group.get_group(2011))
```

```
       Name  Position  Year  Marks
6  Benjamin         3  2011    396
8      John         3  2011    356
```

Properties

- There's a lot of neat and useful properties and functions attached to a dataframe. The dtypes property gives you the data types of your fields.

```
df = pd.DataFrame({
  'x': [1, 3, 4],
  'y': [2, 4, 5]
}, index=['a', 'b', 'c'])
df.dtypes
```

- If you need to know the dimensions of your dataframe, use the shape property.

```
df.shape
```

Columns or field

- You can access each column by bracket [] or dot notation . . To access a column by name, specify the column name inside the brackets.
- Multiple columns may be selecting by passing a list of column names.
- To access the columns by dot notation, do the following. Be careful when accessing columns/fields using dot notation; if you have a field that is named the same as a function attached to the dataframe (e.g. mean), the function has precedence over the field (e.g. df['mean'] is preferred over df.mean).

```
df['Year']
#or,
df.Year
# df[['Year']] with column name
# Multiple columns may be selecting by passing a list of
    column names.
df[['Name', 'Year']]
```

Column Name or Field

In [28]:
```
# import pandas library
import pandas as pd
raw = {'Name': ['John', 'John', 'Grace', 'Grace', 'Benjamin', 'Benjamin', 'Benjamin',
'Benjamin', 'John', 'Alex', 'Alex', 'Alex'],
'Position': [2, 1, 1, 4, 2, 4, 3, 1, 3, 2, 4, 3],
'Year': [2009, 2010, 2009, 2010, 2010, 2010, 2011, 2012, 2011, 2013, 2013, 2012],
'Marks':[408, 398, 422, 376, 401, 380, 396, 388, 356, 402, 368, 378]}
df = pd.DataFrame(raw)
df
df['Year']
```

Out[28]:
```
0     2009
1     2010
2     2009
3     2010
4     2010
5     2010
6     2011
7     2012
8     2011
9     2013
10    2013
11    2012
Name: Year, dtype: int64
```

Pandas: Series to dataframe

In [17]:
```
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'], name='numb
s
```

Out[17]:
```
a    1
b    2
c    3
d    4
e    5
Name: number, dtype: int64
```

In [18]:
```
s.to_frame()
```

Out[18]:

	number
a	1
b	2
c	3
d	4
e	5

Rows or records

- To access the rows or records, use `iloc` and specify the numeric location value or `loc` and specify the index value. Below, `.iloc[0]` refers to the first record.

```
df.iloc[0]
# or
df.loc['a']
```

- Multiple rows may be selected by numeric index through slicing.

```
df.iloc[0:2]
```

Methods

- Use `describe()` to get summary statistics over your fields.

```
[41]: df.describe()
```

[41]:

	x	y
count	3.000000	3.000000
mean	2.666667	3.666667
std	1.527525	1.527525
min	1.000000	2.000000
25%	2.000000	3.000000
50%	3.000000	4.000000
75%	3.500000	4.500000
max	4.000000	5.000000

Methods

- The `info()` method gives other types of field profiling information.
- The cummulative sum is retrieved by calling `cumsum()`.

[42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, a to c
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x       3 non-null      int64
 1   y       3 non-null      int64
dtypes: int64(2)
memory usage: 152.0+ bytes
```

The cummulative sum is retrieved by calling `cumsum()`.

[43]:
```
df.cumsum()
```

[43]:

	x	y
a	1	2
b	4	6

Methods

- You can mathematically operate on a dataframe. Below, we turn the integer values into percentages.

```
[44]: df / df.sum()
```

[44]:

	x	y
a	0.125	0.181818
b	0.375	0.363636
c	0.500	0.454545

Transposing is easy.

```
[45]: df.transpose()
```

[45]:

	a	b	c
x	1	3	4
y	2	4	5

You can also transpose a dataframe as follows.

```
[46]: df.T
```

[46]:

	a	b	c

Importing CSV Data

- you can copy the data and paste in a text editor like Notepad, and then save it with the name mydata.csv in the same directory as your Python scripts

```
import pandas as pd
data = pd.read_csv('mydata.csv')
print(data)
```

- if you have saved your file in a different path, ensure you pass the correct path as the argument to the method
- this can either be a relative path, like "../mydata.csv", or an absolute path like "/Users/MRK/data/mydata.csv"

```
import pandas as pd
data = pd.read_csv(r"C:\Users\Admin\OneDrive\Lecture
    Slides\Matchine Learning\data\data.csv")
df = pd.DataFrame(data)
df.head()
```

Importing Excel Data

- Pandas also has the read_excel function that can be used for reading Excel data into a Pandas DataFrame
- we will use an Excel file named workers.xlsx with details of workers in a company

```
import pandas as pd
data = pd.read_excel('workers.xlsx')
print (data)
print (data.loc[[1,4,7],['Name','Salary']])
```

- the loc() method is a nice utility that helps us read only certain rows of a specific column in the dataset
- Pandas also allows us to read from two Excel sheets simultaneously. Suppose our previous data is in Sheet1, and we have some other data in Sheet2 of the same Excel file

Importing Excel Data

- Pandas also allows us to read from two Excel sheets simultaneously. Suppose our previous data is in Sheet1, and we have some other data in Sheet2 of the same Excel file
- the following code shows how we can read from the two sheets simultaneously:

```
import pandas as pd
with pd.ExcelFile('workers.xlsx') as x:
s1 = pd.read_excel(x, 'Sheet1')
s2 = pd.read_excel(x, 'Sheet2')

print("Sheet 1:")
print (s1)
print("")
print("Sheet 2:")
print (s2)
```

Important Methods Pandas Packages

Data Importing	Data Cleaning	Data Statistic
pd.read_csv()	pd.fillna()	pd.head()
pd.read_table()	pd.dropna()	pd.tail()
pd.read_excel()	pd.sort_values()	pd.describe()
pd.read_sql()	pd.apply()	pd.info()
pd.read_json()	pd.groupby()	pd.mean()
pd.read_html()	pd.append()	pd.median()
pd.DataFrame()	pd.join()	pd.count()
pd.concat()	pd.rename()	pd.std()
pd.series()	pd.to_csv()	pd.max()
pd.date_range()	pd.set_index()	pd.min()

Sorting

- The function sort_values can sort the records by index. Below, we sort by gender. If not specified, the sort will always be ascendingly.
- pandas.DataFrame.isin() is used to filter data frames. isin() method helps in selecting rows with having a particular(or Multiple) value in a particular column

```
df = pd.DataFrame({
  'Name': ['john doe', 'jack smith', 'jason demming', 'joe
      turing'],
  'Handedness': [0, 1, 0, 1],
  'Score': [0.95, 0.85, 0.75, 0.65],
  'Age': [25, 18, 22, 20]
})
df
# sorting based on single variable
df.sort_values(['Handedness'])
# sorting based on multiple variable
df.sort_values(['Score','Handedness'])
```

Out[3]:

	Name	Handedness	Score	Age
0	john doe	0	0.95	25
1	jack smith	1	0.85	18
2	jason demming	0	0.75	22
3	joe turing	1	0.65	20

In [4]:
```
df.sort_values(['Handedness'])
```

Out[4]:

	Name	Handedness	Score	Age
0	john doe	0	0.95	25
2	jason demming	0	0.75	22
1	jack smith	1	0.85	18
3	joe turing	1	0.65	20

In [5]:
```
df.sort_values(['Score','Handedness'])
```

Out[5]:

	Name	Handedness	Score	Age
3	joe turing	1	0.65	20
2	jason demming	0	0.75	22
1	jack smith	1	0.85	18
0	john doe	0	0.95	25

Python 3 Beginner's Reference Cheat Sheet

Main data types

boolean = *True / False*
integer = 10
float = 10.01
string = "123abc"
list = [value1, value2, ...]
dictionary = { key1:value1, key2:value2, ...}

Numeric operators

+	addition
-	subtraction
*	multiplication
/	division
**	exponent
%	modulus
//	floor division

Comparison operators

==	equal
!=	different
>	higher
<	lower
>=	higher or equal
<=	lower or equal

Boolean operators

and	logical AND
or	logical OR
not	logical NOT

Special characters

#	coment
\n	new line
\<char>	scape char

String operations

string[i]	retrieves character at position i
string[-1]	retrieves last character
string[i:j]	retrieves characters in range i to j

List operations

list = []	defines an empty list
list[i] = x	stores x with index i
list[i]	retrieves the item with index I
list[-1]	retrieves last item
list[i:j]	retrieves items in the range i to j
del list[i]	removes the item with index i

Dictionary operations

dict = {}	defines an empty dictionary
dict[k] = x	stores x associated to key k
dict[k]	retrieves the item with key k
del dict[k]	removes the item with key k

String methods

string.upper()	converts to uppercase
string.lower()	converts to lowercase
string.count(x)	counts how many times x appears
string.find(x)	position of the x first occurrence
string.replace(x,y)	replaces x for y
string.strip(x)	returns a list of values delimited by x
string.join(L)	returns a string with L values joined by string
string.format(x)	returns a string that includes formatted x

List methods

list.append(x)	adds x to the end of the list
list.extend(L)	appends L to the end of the list
list.insert(i,x)	inserts x at i position
list.remove(x)	removes the first list item whose value is x
list.pop(i)	removes the item at position i and returns its value
list.clear()	removes all items from the list
list.index(x)	returns a list of values delimited by x
list.count(x)	returns a string with list values joined by S
list.sort()	sorts list items
list.reverse()	reverses list elements
list.copy()	returns a copy of the list

Dictionary methods

dict.keys()	returns a list of keys
dict.values()	returns a list of values
dict.items()	returns a list of pairs (key,value)
dict.get(k)	returns the value associtated to the key k
dict.pop()	removes the item associated to the key and returns its value
dict.update(D)	adds keys-values (D) to dictionary
dict.clear()	removes all keys-values from the dictionary
dict.copy()	returns a copy of the dictionary

Legend: **x,y** stand for any kind of data values, **s** for a string, **n** for a number, **L** for a list where **i,j** are list indexes, **D** stands for a dictionary and **k** is a dictionary key.

Python 3 Beginner's Reference Cheat Sheet

Built-in functions

print(x, sep='y')	prints x objects separated by y
input(s)	prints s and waits for an input that will be returned
len(x)	returns the length of x (s, L or D)
min(L)	returns the minimum value in L
max(L)	returns the maximum value in L
sum(L)	returns the sum of the values in L
range(n1,n2,n)	returns a sequence of numbers from n1 to n2 in steps of n
abs(n)	returns the absolute value of n
round(n1,n)	returns the n1 number rounded to n digits
type(x)	returns the type of x (string, float, list, dict ...)
str(x)	converts x to string
list(x)	converts x to a list
int(x)	converts x to a integer number
float(x)	converts x to a float number
help(s)	prints help about x
map(function, L)	Applies function to values in L

Conditional statements

```
if <condition>:
    <code>
else if <condition>:
    <code>
...
else:
    <code>

if <value> in <list>:
```

Data validation

```
try:
    <code>
except <error>:
    <code>
else:
    <code>
```

Working with files and folders

```
import os
os.getcwd()
os.makedirs(<path>)
os.chdir(<path>)
os.listdir(<path>)
```

Loops

```
while <condition>:
    <code>

for <variable> in <list>:
    <code>

for <variable> in range(start,stop,step):
    <code>

for key, value in dict.items():
    <code>
```

Loop control statements

break	finishes loop execution
continue	jumps to next iteration
pass	does nothing

Running external programs

```
import os
os.system(<command>)
```

Functions

```
def function(<params>):
    <code>
    return <data>
```

Modules

```
import module
module.function()

from module import *
function()
```

Reading and writing files

```
f = open(<path>,'r')
f.read(<size>)
f.readline(<size>)
f.close()

f = open(<path>,'r')
for line in f:
    <code>
f.close()

f = open(<path>,'w')
f.write(<str>)
f.close()
```

Legend: **x,y** stand for any kind of data values, **s** for a string, **n** for a number, **L** for a list where **i,j** are list indexes, **D** stands for a dictionary and **k** is a dictionary key.

Classroom Practices

- Read the Automobile mileage dataset, available at this URL: `https://github.com/TrainingByPackt/Big-Data-Analysis-with-Python/blob/master/Lesson01/imports-85.data`. Convert it to `.csv`.
- Read this data file and select select asubset of the first 10 rows of the DataFrame
- Draw pairwise plot of each variables by the group.

Classroom Practices

- Read the iris dataset, available at this URL: `https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv`.
- Read this data file and select select asubset of the first 10 rows of the DataFrame
- Draw pairwise plot of each variables by the species.
- draw histogram and boxplot by the species

Covariance Analysis

```
# generate related variables
from numpy import mean
from numpy import std
from numpy.random import randn
from numpy.random import seed
# seed random number generator
seed(1)
# prepare data
data1 = 20 * randn(1000) + 100
data2 = data1 + (10 * randn(1000) + 50)
from numpy import cov
covariance = cov(data1, data2)
print(covariance)
```

Correlation Analysis With Heatmap

```
    # calculate Pearson's correlation
    from scipy.stats import pearsonr
    corr, _ = pearsonr(data1, data2)
# Note:  corr, _ provies only value of r
# whereas corr proives r with p-value

    print('Pearsons correlation: %.3f' % corr)


    # calculate the spearmans's rank correlation between two
        variables
    from scipy.stats import spearmanr
    corr, _ = spearmanr(data1, data2)
    print('Spearmans correlation: %.3f' % corr)
```

Classroom Practices

- Read the Boston Housing dataset dataset, available at this URL: `https://www.kaggle.com/c/boston-housing`.
- Read this data file and find the correlation with heatmap

Housing Dataset: Correlation Analysis With Heatmap

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_table('/Users/apple/Downloads/housing.data',
    header=None, sep='\s+')
df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM',
'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B',
'LSTAT', 'MEDV']
# Correlation between different variables
corr = df.corr()
# Set up the matplotlib plot configuration
f, ax = plt.subplots(figsize=(12, 10))
# Generate a mask for upper traingle
mask = np.triu(np.ones_like(corr, dtype=bool))
# Configure a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)
# Draw the heatmap
sns.heatmap(corr, annot=True, mask = mask, cmap=cmap)
```

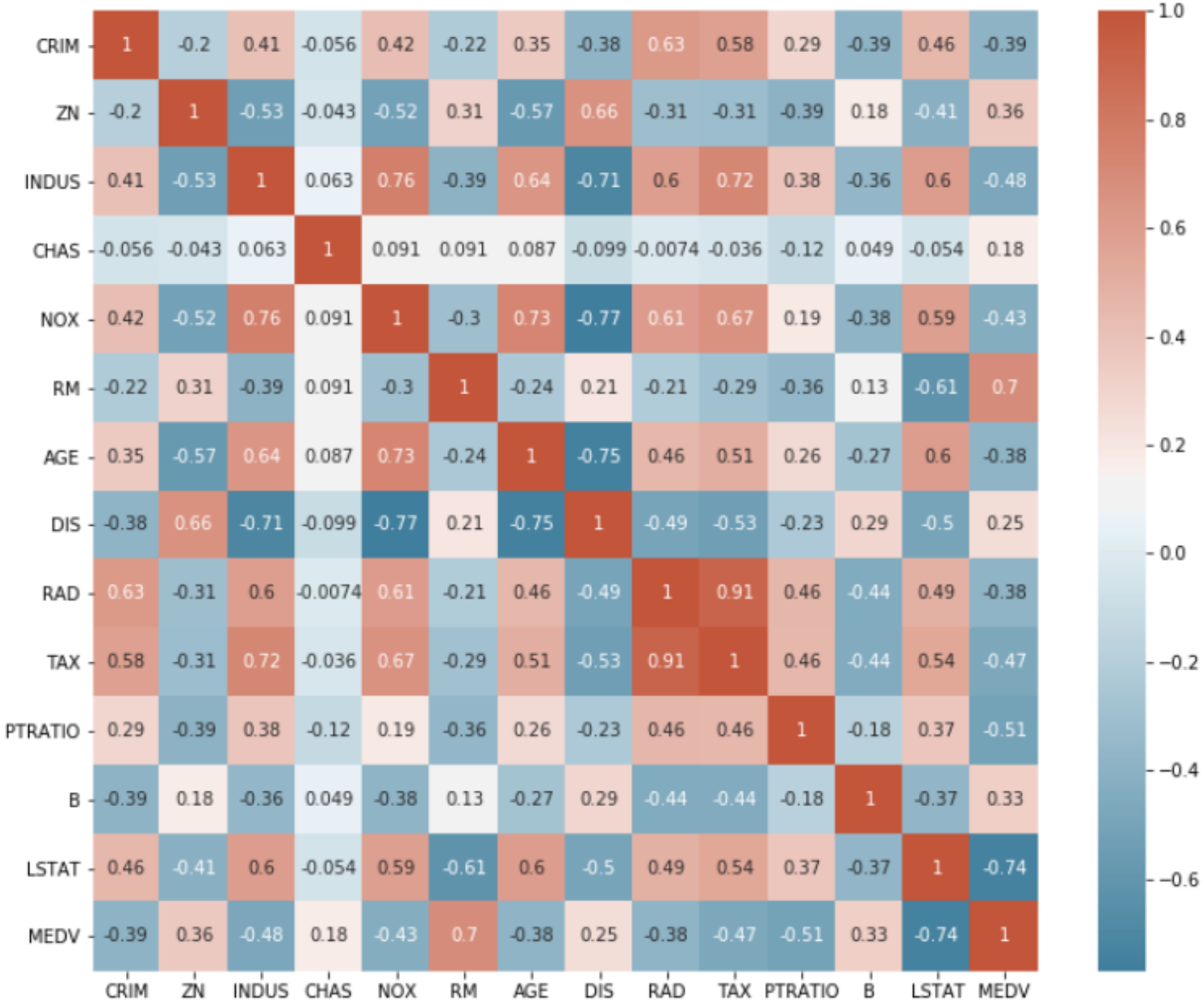


Fig 3. Correlation Heatmap for Housing Dataset

Linear Regression Analysis

```
import numpy as np
x = np.arange(10, 20)
y = np.array([2, 1, 4, 5, 8, 12, 18, 25, 96, 48])
z = np.array([5, 3, 2, 1, 0, -2, -8, -11, -15, -16])
xyz = np.array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
[2, 1, 4, 5, 8, 12, 18, 25, 96, 48],
[5, 3, 2, 1, 0, -2, -8, -11, -15, -16]])


# fit a linear regression model to the data
import statsmodels.api as sm
X = sm.add_constant(x.ravel())
results = sm.OLS(y,X).fit()
results.summary()
```

```
# Alternatively, using matchine learning tools
import scipy.stats
import scipy.stats slope, intercept, r, p, stderr =
    scipy.stats.linregress(x, y)
print(slope, intercept, r, p, stderr)

line = f'Regression line: y={intercept:.2f}+{slope:.2f}x,
    r={r:.2f}'
line

import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(x, y, linewidth=0, marker='s', label='Data points')
ax.plot(x, intercept + slope * x, label=line)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend(facecolor='white')
plt.show()
```

Polynomial Regression

```
## Polynomial Regression
import numpy
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

plt.scatter(x, y)
plt.show()

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))
myline = numpy.linspace(1, 22, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()

print(r2_score(y, mymodel(x))) # print R^2

# Predict future value
speed = mymodel(17)
print(speed)
```

Example 1: Multiple Regression

```
import pandas
# df = pandas.read_csv("cars.csv")
df=pandas.read_csv("C:\\Users\\Dr. M R
    Karim\\Dropbox\\a_Lecture Slides\\Python\R code and
    Data\\cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

# Multiple regession Model,
import statsmodels.api as sm
X = sm.add_constant(X)
# X = sm.add_constant(X.ravel())
results = sm.OLS(y,X).fit()
results.summary()
```

```
# Alternatively,
from sklearn import linear_model # sklearn is a matchine
    learing package
regr = linear_model.LinearRegression()
regr.fit(X, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

#predict the CO2 emission of a car where the weight is
    2300kg, and the volume is 1300ccm:
predictedCO2 = regr.predict([[2300, 1300]])
print(predictedCO2)
```

Example 2: Multiple Regression

```
import pandas as pd
from sklearn import linear_model
import tkinter as tk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import
    FigureCanvasTkAgg

Stock_Market = {'Year':
    [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,201
  'Month': [12,
      11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
  'Interest_Rate':
      [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1
  'Unemployment_Rate':
      [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9
  'Stock_Index_Price':
      [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,
}
df =
    pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate
```

N.B. the details can be found in
https://www.statsmodels.org/devel/generated/statsmodels.regression.linear_model.OLS.html

Example 2: Multiple Regression

```
X = df[['Interest_Rate','Unemployment_Rate']].astype(float)
    # here we have 2 input variables for multiple
    regression. If you just want to use one variable for
    simple linear regression, then use X =
    df['Interest_Rate'] for example.Alternatively, you may
    add additional variables within the brackets
Y = df['Stock_Index_Price'].astype(float) # output variable
    (what we are trying to predict)

# Multiple regession Model,
import statsmodels.api as sm
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
results.summary()
# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
```

Problems & Motivation

Let us explore the dataset. We will be using the **Boston House Prices Dataset**, with 506 rows and 13 attributes with a target column. Let's take a quick look at the dataset.

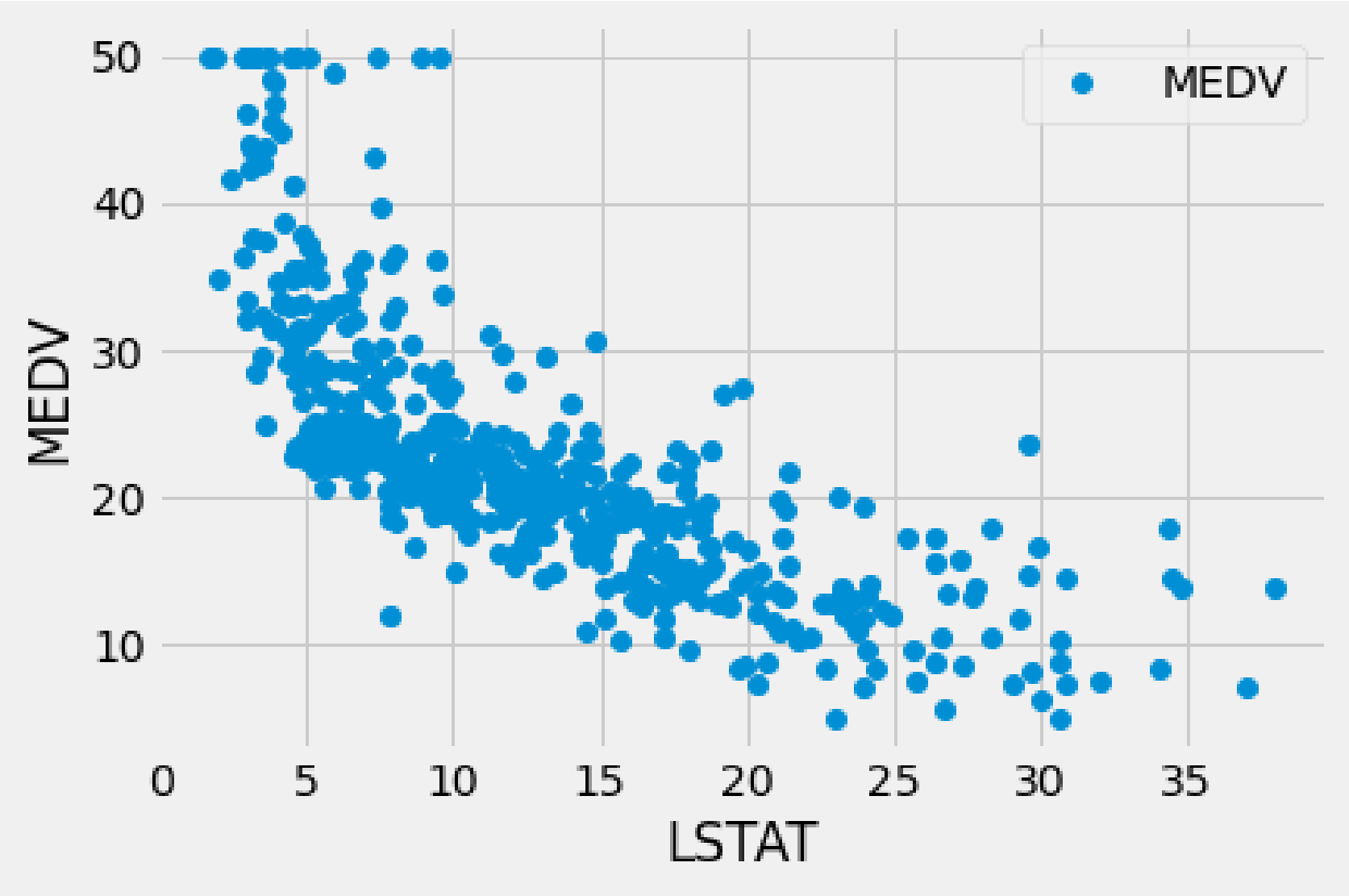
```
import io
boston = pd.read_csv(io.BytesIO(uploaded['boston.csv']))
# Dataset is now stored in a Pandas Dataframe
boston.head(10)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

- **Crim**: Per capita crime rate by town
- **Zn**: Proportion of residential land zoned for lots over 25,000 sq. ft.
- **Indus**: Proportion of non-retail business acres per town
- **Chas**: Charles River dummy variable (= 1 if tract bounds river; 0, otherwise)
- **Nox**: Nitrogen oxides concentration (parts per 10 million)
- **Rm**: Average number of rooms per dwelling
- **Age**: Proportion of owner-occupied units built before 1940
- **Dis**: Weighted mean of distances to five Boston employment centers
- **Rad**: Index of accessibility to radial highways
- **Tax**: Full-value property tax rate per \$10,000
- **Ptratio**: Pupil–Teacher ratio by town
- **B**: $1000(Bk - 0.63)^2$, where Bk is the proportion of Blacks by town
- **Lstat**: Lower status of the population (percent)
- **Medv**: Median Price value of owner-occupied homes in \$1000s

Classroom Exercise

- We want to predict the median price value of owner-occupied homes. How do you predict the price?


MEDV
50
40
30
20
10
0
5
10
15
20
25
30
35
LSTAT
MEDV

Python Code

```
from google.colab import files
uploaded = files.upload()
import io
boston = pd.read_csv(io.BytesIO(uploaded['boston.csv']))
# Dataset is now stored in a Pandas Dataframe
boston.head()
import matplotlib.pyplot as plt
boston.plot(x='LSTAT',y='MEDV',style='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.show()
```

Python Code: OLS for Linear Regression Model

```
# Simple Linear Regression Model
import statsmodels.api as sm
model =sm.OLS(boston.MEDV, sm.add_constant(boston.LSTAT))
result = model.fit()
print(result.summary())
```

```python
import statsmodels.api as sm
model =sm.OLS(boston.MEDV, sm.add_constant(boston.LSTAT))
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.544
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     601.6
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           5.08e-88
Time:                        03:26:52   Log-Likelihood:                -1641.5
No. Observations:                 506   AIC:                             3287.
Df Residuals:                     504   BIC:                             3295.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.5538      0.563     61.415      0.000      33.448      35.659
LSTAT         -0.9500      0.039    -24.528      0.000      -1.026      -0.874
==============================================================================
Omnibus:                      137.043   Durbin-Watson:                   0.892
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              291.373
Skew:                           1.453   Prob(JB):                     5.36e-64
Kurtosis:                       5.319   Cond. No.                         29.7
==============================================================================
```

Python Code: OLS for Multiple Linear Regression Model

```
# Multiple Linear Regression Model
import statsmodels.api as sm
X = boston[['LSTAT', 'CRIM']]
model =sm.OLS(boston.MEDV, sm.add_constant(X))
result = model.fit()
print(result.summary())

## Alternatively
import statsmodels.formula.api as smf
# formula: response ~ predictor + predictor
est = smf.ols(formula='MEDV ~ LSTAT + CRIM',
    data=boston).fit()
print(est.summary())

# In GLM framework
Gaussian_model = sm.GLM(boston.MEDV, sm.add_constant(X),
    family=sm.families.Gaussian()).fit()
print(Gaussian_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.548
Model:                            OLS   Adj. R-squared:                  0.546
Method:                 Least Squares   F-statistic:                     304.4
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           2.33e-87
Time:                        03:37:58   Log-Likelihood:                -1639.6
No. Observations:                 506   AIC:                             3285.
Df Residuals:                     503   BIC:                             3298.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.3192      0.574     59.816      0.000      33.192      35.446
LSTAT         -0.9114      0.043    -21.004      0.000      -0.997      -0.826
CRIM          -0.0704      0.036     -1.956      0.051      -0.141       0.000
==============================================================================
Omnibus:                      146.159   Durbin-Watson:                   0.872
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              329.172
Skew:                           1.517   Prob(JB):                     3.32e-72
Kurtosis:                       5.531   Cond. No.                         32.7
==============================================================================
```

Chapter 3: Naïve Bayes Classification

3 Chapter 3: Naïve Bayes Classification

3.1 Problem & Motivation

3.2 Bayes Theorem

3.3 Conditional Probability

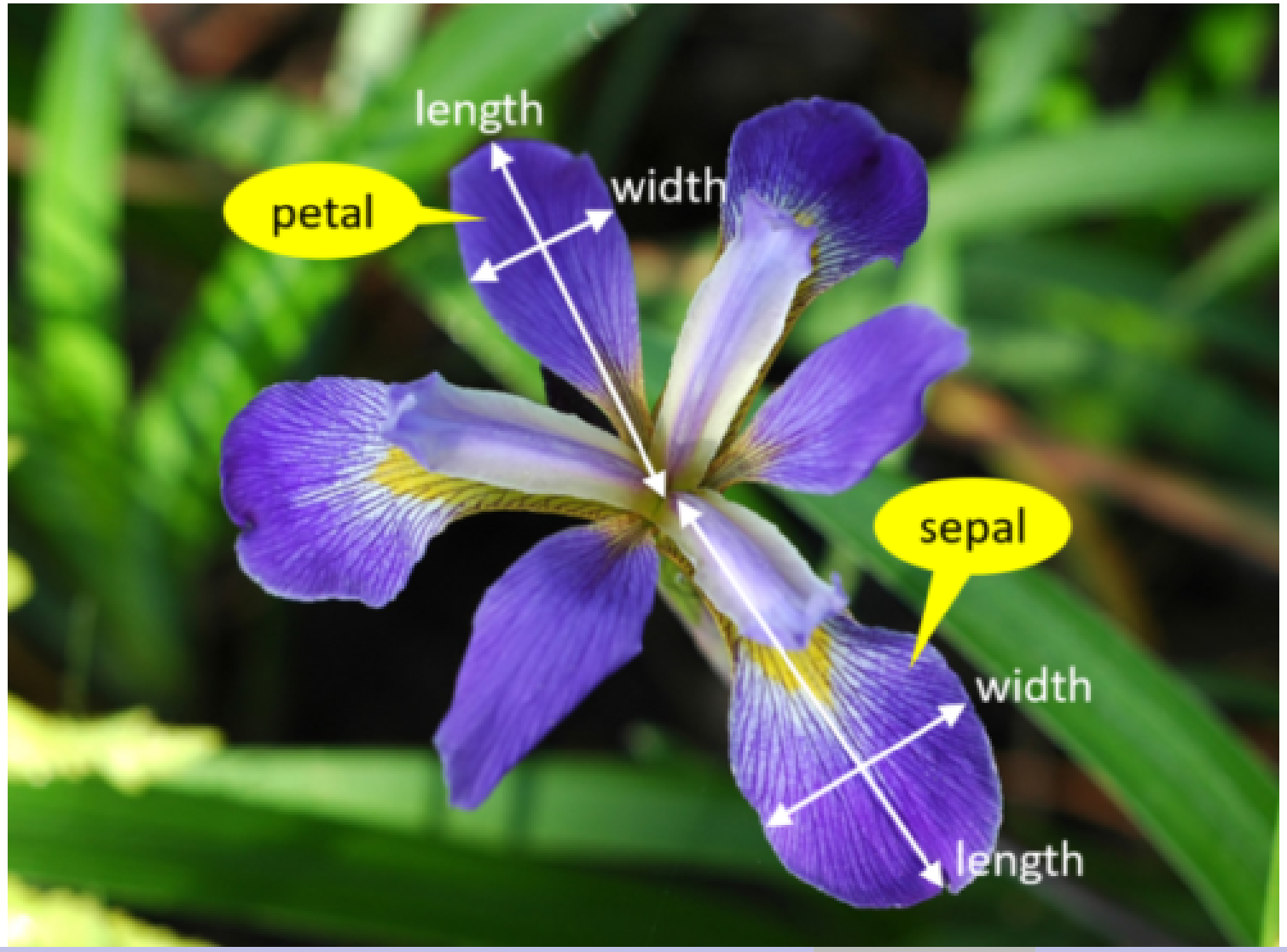
3.4 Naive Bayes Classification

3.5 Practical Example

3.6 Practical Session: Naive Bayes Classification in Python and R

Problem & Motivation

- **Flower characteristics (variables)**



Name of the species of the Iris Flower

Data Matrix for iris data

	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

Problem & Motivation: new observations

1. suppose you have following characteristics of six flowers

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	6.70	3.00	5.20	2.30
2	6.30	2.50	5.00	1.90
3	6.50	3.00	5.20	2.00
4	6.20	3.40	5.40	2.30
5	5.90	3.00	5.10	1.80
6	1.00	2.00	1.00	0.40

2. what are the species name of these flower?
3. how do you classify these flower to Versicolor, Setosa or Virginica species?
4. how the variables: Sepal.Length, Sepal.Width, Petal.Length and Petal.Width are used to predict the Species of flowers?

Naive Bayes Algorithm

1. naive Bayes is a common machine learning technique used in the field of data science, for example, cancer detection problem
2. assessing fraud transaction in the business
3. it is a family of simple "probabilistic classifiers" based on applying Bayes' theorem
4. naive Bayes the algorithm makes a very strong assumption that the value of a particular *feature* is independent of the value of any other feature, given the class variable. It simplifies the computation and makes the algorithm computationally efficient, but it might not hold true in all real-world scenarios. A Bayes classifier without the Naive assumption could account for such dependencies between features and provide more accurate predictions.
5. for example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter
   — a naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple

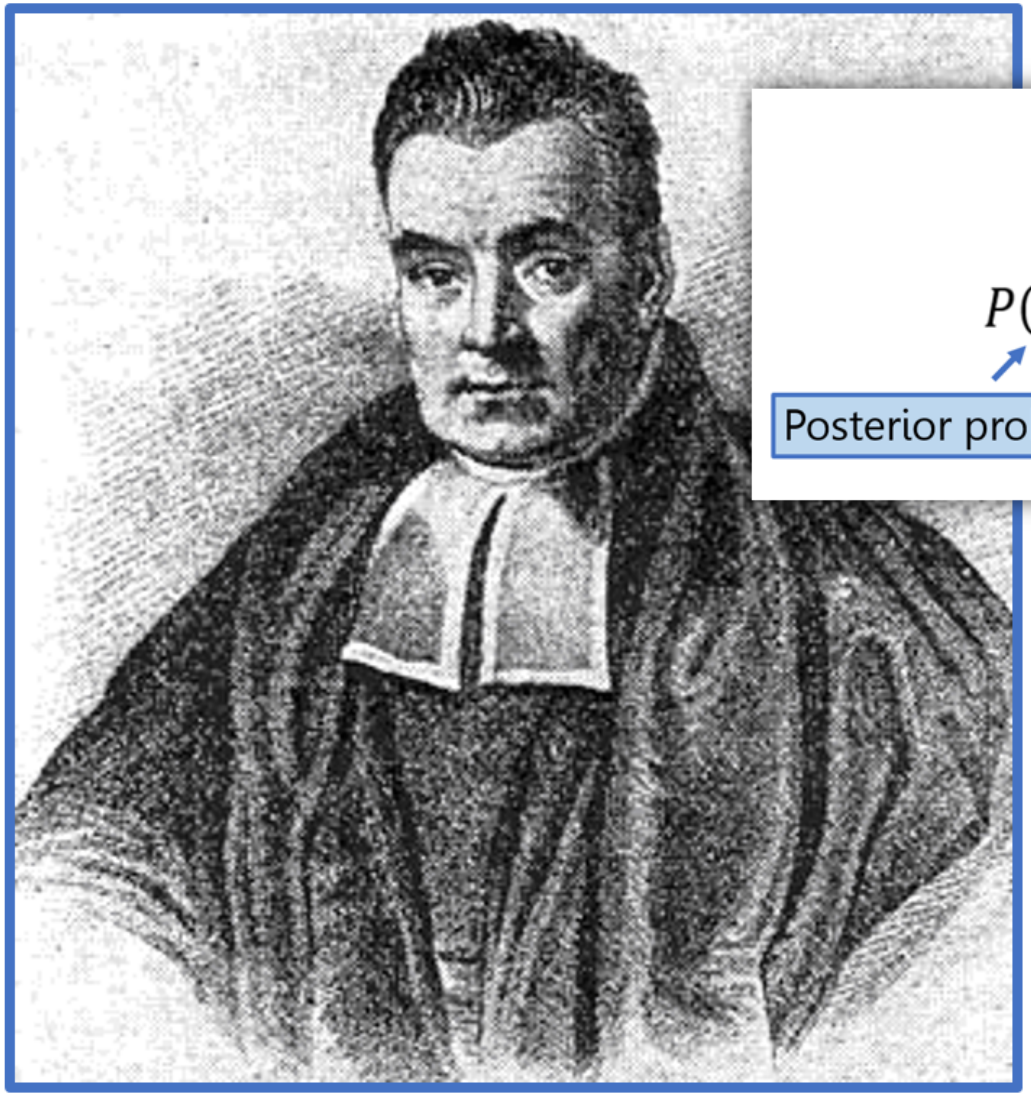
Bayes Theorem

1. Bayes' theorem is stated mathematically as the following equation:

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

where A and B are events and $P(B) \neq 0$

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true $\Rightarrow$ posterior probability
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true
- $P(A)$ and $P(B)$ are the probabilities of observing A and B respectively; they are known as the marginal probability.



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

Likelihood

Prior probability

Posterior probability

Evidence

- the posterior is proportional to the prior times the likelihood

(Q). What do you mean by conditional probability?

Marginal, Joint and Conditional Probabilities

- **Example**

Let

$$M = \text{event an officer is a man}$$
$$W = \text{event an officer is a woman}$$
$$A = \text{event an officer is promoted}$$
$$A^c = \text{event an officer is not promoted}$$

Dividing the data values in Table 4.4 by the total of 1200 officers enables us to summarize the available information with the following probability values.

TABLE 4.4 Promotion Status of Police Officers Over the Past Two Years

	Men	Women	Total
Promoted	288	36	324
Not Promoted	672	204	876
Total	960	240	1200

- Find the joint and marginal probabilities table.

JOINT PROBABILITY TABLE FOR PROMOTIONS

Joint probabilities appear in the body of the table.

	Men (M)	Women (W)	Total
Promoted (A)	.24	.03	.27
Not Promoted (A^c)	.56	.17	.73
Total	.80	.20	1.00

Marginal probabilities appear in the margins of the table.

$P(M \cap A) = 288/1200 = .24$ probability that a randomly selected officer is a man *and* is promoted

$P(M \cap A^c) = 672/1200 = .56$ probability that a randomly selected officer is a man *and* is not promoted

$P(W \cap A) = 36/1200 = .03$ probability that a randomly selected officer is a woman *and* is promoted

$P(W \cap A^c) = 204/1200 = .17$ probability that a randomly selected officer is a woman *and* is not promoted

Independent Events and Conditional Probability

- *independent events*
  if two events, A and B are independent then the joint probability is
  $$P(A \text{ and } B) = P(A \cap B) = P(A)P(B)$$
  — for example, if two coins are flipped, then the chance of both being heads is
  $$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$
- *conditional probability*
  - is the probability of a particular event occurring, given that another event has occurred, i. e., the conditional probability is
  $$P(A|B) = \frac{P(A \cap B)}{P(B)}$$
- when events are *independent* then the occurrence of one event does not affect the occurrence of another, then
  $$\Pr(A|B) = \Pr(A)$$

Continuous Assessment:

1. what the conditional probability when two events A and B are mutually exclusive?
2. what the conditional probability that an event B is contained within an event A, that is $B \subset A$

Back to the Previous Example

JOINT PROBABILITY TABLE FOR PROMOTIONS

Joint probabilities appear in the body of the table.

	Men (M)	Women (W)	Total
Promoted (A)	.24	.03	.27
Not Promoted (A^c)	.56	.17	.73
Total	.80	.20	1.00

Marginal probabilities appear in the margins of the table.

(a). What is the probability that an officer is promoted if he is a man?

(b). What is the probability that an officer is promoted if she is a women?

Solution of (a).

Let us begin the conditional probability analysis by computing the probability that an officer is promoted given that the officer is a man.

We now see that the conditional probability $P(A \mid M)$ can be computed as .24/.80. Refer to the joint probability table (Table 4.5). Note in particular that .24 is the joint probability of A and M; that is, $P(A \cap M) = .24$. Also note that .80 is the marginal probability that a randomly selected officer is a man; that is, $P(M) = .80$. Thus, the conditional probability $P(A \mid M)$ can be computed as the ratio of the joint probability $P(A \cap M)$ to the marginal probability $P(M)$.

$$P(A \mid M) = \frac{P(A \cap M)}{P(M)} = \frac{.24}{.80} = .30$$

Continuous Assessment:

(b). What is the probability that an officer is promoted if she is a women?

General Form: Bayes' Theorem

BAYES' THEOREM (TWO-EVENT CASE)

$$P(A_1 \mid B) = \frac{P(A_1)P(B \mid A_1)}{P(A_1)P(B \mid A_1) + P(A_2)P(B \mid A_2)} \tag{4.17}$$

$$P(A_2 \mid B) = \frac{P(A_2)P(B \mid A_2)}{P(A_1)P(B \mid A_1) + P(A_2)P(B \mid A_2)} \tag{4.18}$$

BAYES' THEOREM

$$P(A_i \mid B) = \frac{P(A_i)P(B \mid A_i)}{P(A_1)P(B \mid A_1) + P(A_2)P(B \mid A_2) + \cdots + P(A_n)P(B \mid A_n)}$$

Example: Bayes' Theorem

FIGURE 4.9 Probability Revision Using Bayes' Theorem

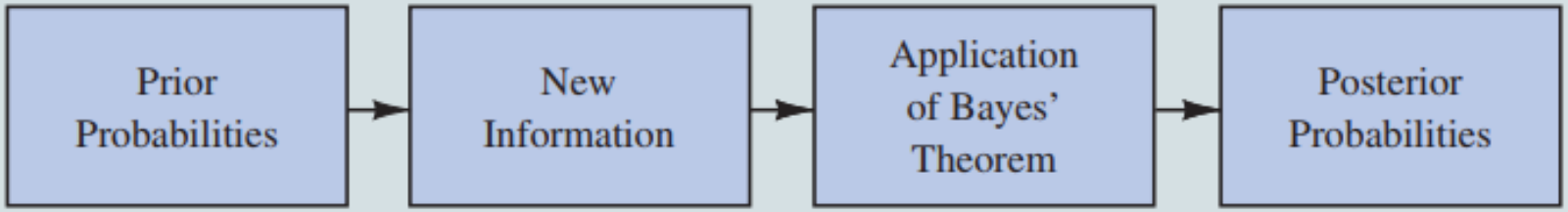


TABLE 4.6 Historical Quality Levels of Two Suppliers

	Percentage Good Parts	Percentage Bad Parts
Supplier 1	98	2
Supplier 2	95	5

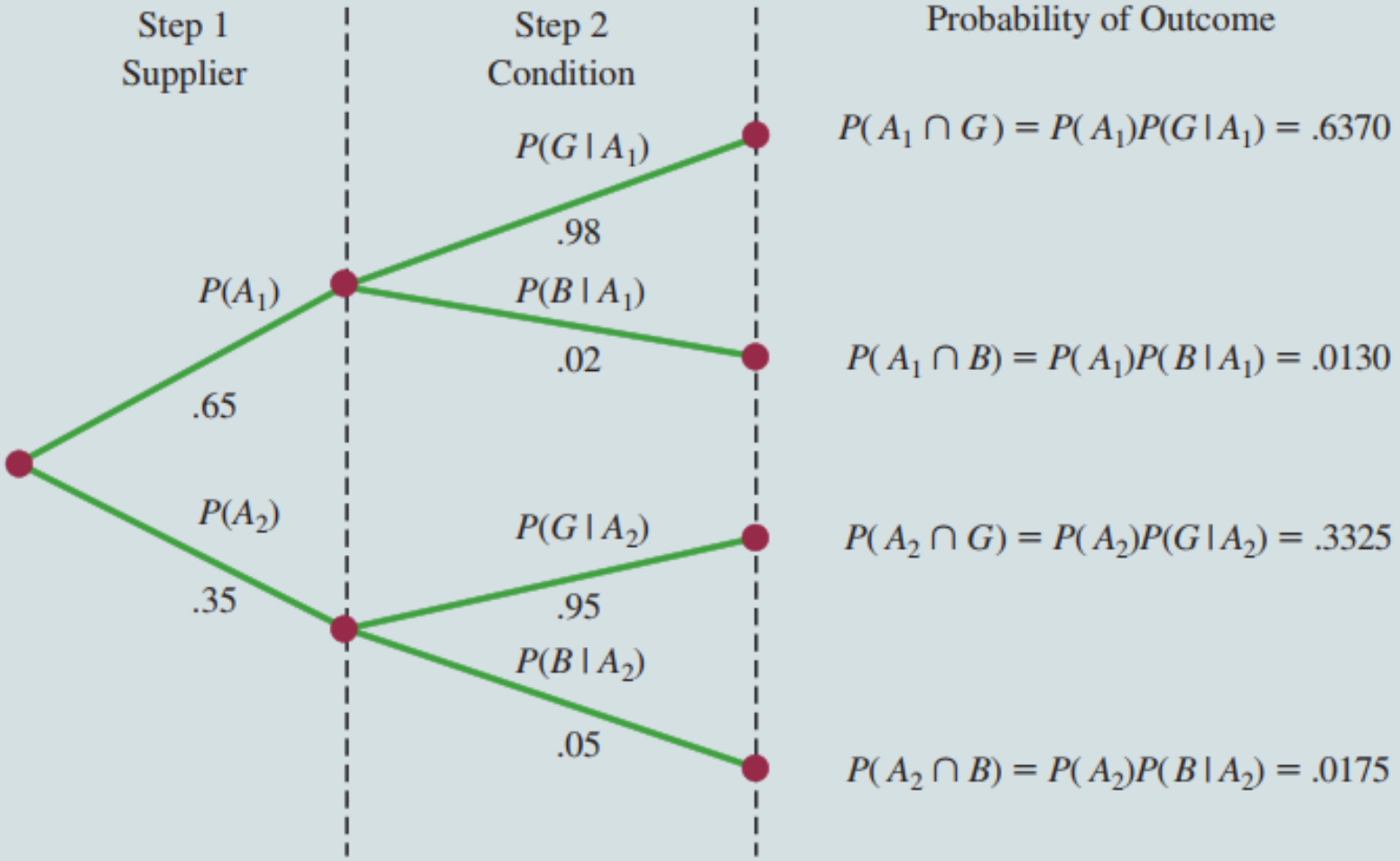
As an application of Bayes' theorem, consider a manufacturing firm that receives shipments of parts from two different suppliers. Let A_1 denote the event that a part is from supplier 1 and A_2 denote the event that a part is from supplier 2. Currently, 65% of the parts purchased by the company are from supplier 1 and the remaining 35% are from supplier 2. Hence, if a part is selected at random, we would assign the prior probabilities $P(A_1) = .65$ and $P(A_2) = .35$.

The quality of the purchased parts varies with the source of supply. Historical data suggest that the quality ratings of the two suppliers are as shown in Table 4.6. If we let G denote the event that a part is good and B denote the event that a part is bad, the information in Table 4.6 provides the following conditional probability values.

$$P(G \mid A_1) = .98 \quad P(B \mid A_1) = .02$$
$$P(G \mid A_2) = .95 \quad P(B \mid A_2) = .05$$

The tree diagram in Figure 4.10 depicts the process of the firm receiving a part from one of the two suppliers and then discovering that the part is good or bad as a two-step experiment. We see that four experimental outcomes are possible; two correspond to the part being good and two correspond to the part being bad.

FIGURE 4.11 Probability Tree for Two-Supplier Example



Examples: Bayes' Theorem

Using equation (4.17) and the probability values provided in the example, we have

$$P(A_1 \mid B) = \frac{P(A_1)P(B \mid A_1)}{P(A_1)P(B \mid A_1) + P(A_2)P(B \mid A_2)}$$

$$= \frac{(.65)(.02)}{(.65)(.02) + (.35)(.05)} = \frac{.0130}{.0130 + .0175}$$

$$= \frac{.0130}{.0305} = .4262$$

In addition, using equation (4.18), we find $P(A_2 \mid B)$.

$$P(A_2 \mid B) = \frac{(.35)(.05)}{(.65)(.02) + (.35)(.05)}$$

$$= \frac{.0175}{.0130 + .0175} = \frac{.0175}{.0305} = .5738$$

Note that in this application we started with a probability of .65 that a part selected at random was from supplier 1. However, given information that the part is bad, the probability that the part is from supplier 1 drops to .4262. In fact, if the part is bad, it has better than a 50–50 chance that it came from supplier 2; that is, $P(A_2 \mid B) = .5738$.

Naive Bayes Classification

1. posterior probability: $P(h_j|x_i) \quad ; \quad j = 1, \ldots, m$
2. prior probability: $P(h_j) \quad ; \quad j = 1, \ldots, m$
3. Bayes theorem:

$$P(h_j|x_i) = \frac{P(x_i|h_j)P(h_j)}{\sum_{j=1}^{m} P(x_i|h_j)P(h_j)}$$

4. assign probabilities of hypothesis give a data value
5. assign new data x_k in the class whose have the highest posterior probabilities

Example 1: given in Dunham, M. H. (2003)

Table: Training Data

ID	Income	Credit status	Class	x_i
1	4	Excellent	h_1	x_4
2	3	Good	h_1	x_7
3	2	Excellent	h_1	x_2
4	3	Good	h_1	x_7
5	4	Good	h_1	x_8
6	2	Excellent	h_1	x_2
7	3	Bad	h_2	x_{11}
8	2	Bad	h_2	x_{10}
9	3	Bad	h_3	x_{11}
10	1	Bad	h_4	x_9

Example 1: given in Dunham, M. H. (2003)

- credit authorizations (hypotheses):
  - h_1 = authorize purchase,
  - h_2 = authorize after further identification,
  - h_3 = do not authorize,
  - h_4 = do not authorize but contact police
- assign twelve data values for all combinations of credit and income:

Income →	1	2	3	4
Excellent	x_1	x_2	x_3	x_4
Good	x_5	x_6	x_7	x_8
Bad	x_9	x_{10}	x_{11}	x_{12}

- from training data:
  $P(h_1) = 60\%; P(h_2) = 20\%; P(h_3) = 10\%; P(h_4) = 10\%$
- calculate $P(x_i|h_j)$ and $P(x_i)$

- ex:

$$P(x_7|h_1) = 2/6;$$

$$P(x_4|h_1) = 1/6;$$

$$P(x_2|h_1) = 2/6;$$

$$P(x_8|h_1) = 1/6;$$

  $P(x_i|h_1) = 0$ for all other x_i

- predict the class for $x_4 = \{\text{Income,Credit}\} = \{4, \text{Excellent}\}$:
  - calculate $P(h_j|x_4)$ for all h_j
  - place x_4 in class with largest value

$$P(h_1|x_4) = \frac{P(x_4|h_1)(P(h_1)}{P(x_4)} = \frac{(1/6)(0.6)}{0.1} = 1$$

  so, x_4 in class h_1

Why is it called Naïve Bayes?

- **Naïve:** It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes:** It is called Bayes because it depends on the principle of Bayes' Theorem.

Types of Naïve Bayes Model

There are three types of Naive Bayes Model, which are given below:

- **Gaussian:** The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial:** The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli:** The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Advantages & Disadvantages of Naive Bayes Classifier

Advantages of Naive Bayes

- this algorithm works very fast and can easily predict the class of a test dataset
- very useful to solve multi-class prediction problems
- Naive Bayes classifier performs better than other models with less training data if the assumption of independence of features holds
- for categorical input variables, the Naive Bayes algorithm performs exceptionally well in comparison to numerical variables

Disadvantages of Naive Bayes

- if your test data set has a categorical variable of a category that wasn't present in the training data set, the Naive Bayes model will assign it zero probability and won't be able to make any predictions in this regard
  — this phenomenon is called 'Zero Frequency,' and you'll have to use a smoothing technique to solve this problem
- it assumes that all the features are independent

Laplace Smoothing

assume binary attribute A, direct estimate:

$$P(A|c_k) = \frac{n_{1k}}{n_{0k} + n_{1k}}$$

Laplace estimate:

$$P(A|c_k) = \frac{n_{1k} + 1}{n_{0k} + n_{1k} + 2}$$

equivalent to prior observation of one example of class k where $A = 0$ and one where $A = 1$

generalized Laplace estimate:

$$P(A_i = v_j|c_k) = \frac{n_{ijk} + 1}{n_k + s_i}$$

- n_{ijk}: number of examples in c_k where $A_i = v_j$
- n_k: number of examples in c_k
- s_i: number of possible values for A_i

Example 2: Python code for Naive Bayes Classification

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
# Data Loading
url='https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw
/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv'
df = pd.read_csv(url)
df.head()
## Confusion Matrix
x = df[['sepal_length', 'sepal_width', 'petal_length',
    'petal_width']]
y = df['species']
x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size=0.25, random_state =3)
nb=GaussianNB()
nb.fit(x_train,y_train)
GaussianNB(priors=None, var_smoothing=1e-09)
predictionsNB = nb.predict(x_test)
print('Classification
    Report:\n',classification_report(y_test, predictionsNB))
print('Confusion Matrix:\n',confusion_matrix(y_test,
    predictionsNB))
print('Accuracy Score:',accuracy_score(y_test,
    predictionsNB))
```

Example 2: R code for Naive Bayes Classification

```
## Example 2 with metric predictors:
data(iris)
m <- naiveBayes(Species ~ ., data = iris)
## alternatively:
m <- naiveBayes(iris[,-5], iris[,5])
table(predict(m, iris), iris[,5])
NewObservation<-c(1,2,1,.4)
NewIris<-rbind(iris[,1:4],NewObservation)
tail(NewIris)
predict(m, NewIris[151,])
```

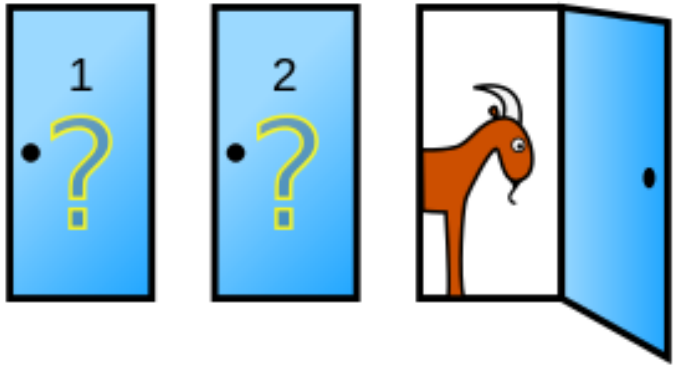
Example 3: From library(mlbench) Package

```
# Example 3: (USA Voting Records 1984)
## Categorical data only:
data(HouseVotes84, package = "mlbench")
model <- naiveBayes(Class ~ ., data = HouseVotes84)
predict(model, HouseVotes84[1:10,])
predict(model, HouseVotes84[1:10,], type = "raw")
pred <- predict(model, HouseVotes84)
table(pred, HouseVotes84$Class)
## using laplace smoothing:
model <- naiveBayes(Class ~ ., data = HouseVotes84, laplace
    = 3)
pred <- predict(model, HouseVotes84)
table(pred, HouseVotes84$Class)
```

Note: It is problematic when a frequency-based probability is zero, because it will wipe out all the information in the other probabilities, and a solution would be Laplace smoothing, which is a technique for smoothing categorical data

Practical Example

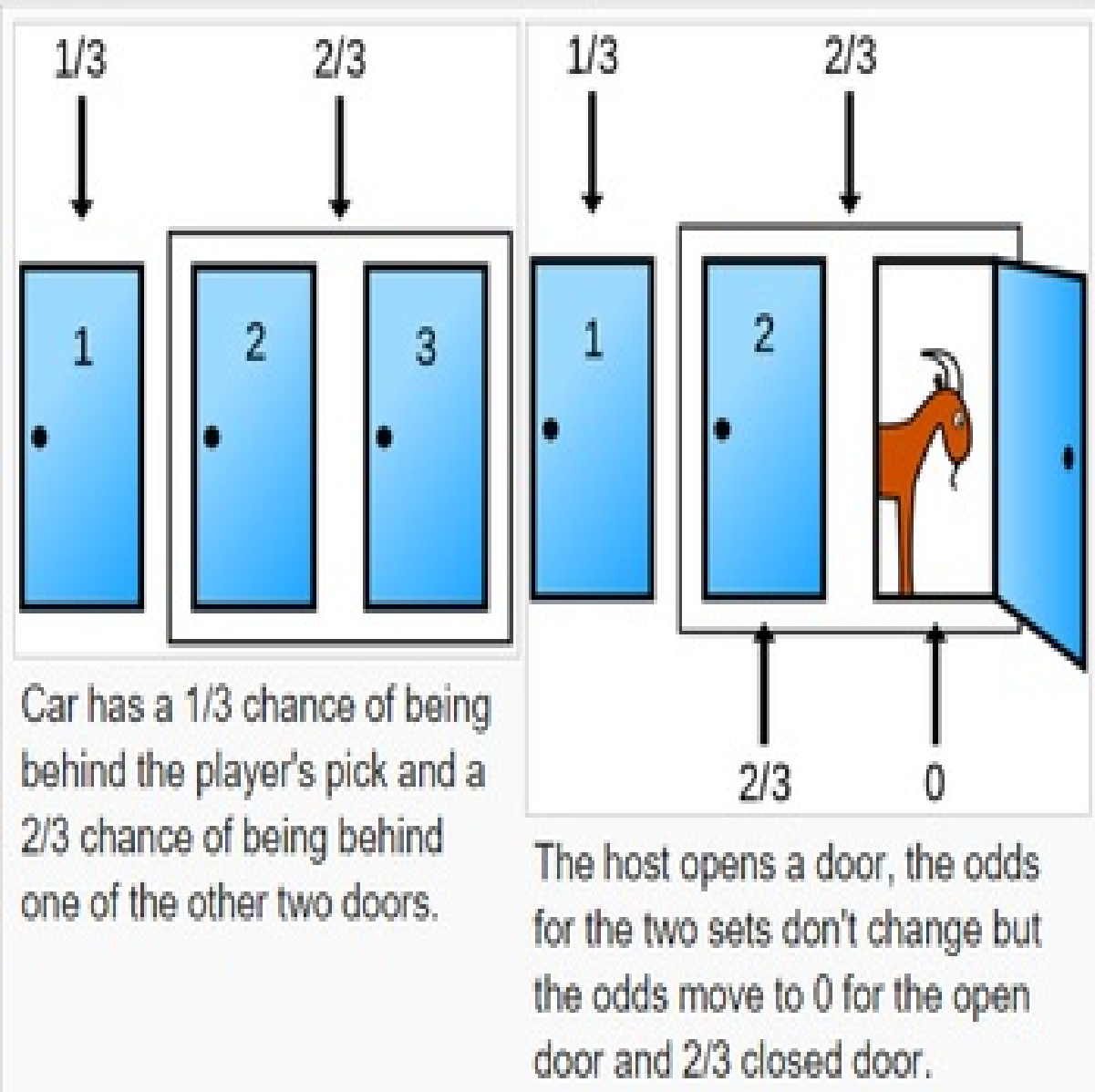
1. *suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?* (From Parade magazine's Ask Marilyn column)



(a). what is the chances of winning the car if you switch your choice No. 1 to No. 2?

(b). what is the chances of winning the car if we stay with your original choice No. 1?

Fun


1/3
2/3
1
2
3
Car has a 1/3 chance of being behind the player's pick and a 2/3 chance of being behind one of the other two doors.
1/3
2/3
1
2
2/3
0
The host opens a door, the odds for the two sets don't change but the odds move to 0 for the open door and 2/3 closed door.

1. $$\Pr(\text{Car in Door 1}) = \frac{1}{3}, \Pr(\text{Car in Door 2}) = \frac{1}{3}, \Pr(\text{Car in Door 3}) = \frac{1}{3}$$

2. $$\Pr(\text{Open Door III}|\text{Car in Door 1}) = \frac{1}{2}, \Pr(\text{Open Door III}|\text{Car in Door 2}) = 1$$
$$\Pr(\text{Open Door III}|\text{Car in Door 3}) = 0$$

3. Application of Bayes Rule:

$$\Pr(A_k|B) = \frac{\Pr(B|A_k)\Pr(A_k)}{\sum_i \Pr(B|A_i)\Pr(A_i)}$$

$$\begin{aligned}\Pr(2|III) &= \frac{\Pr(III|2)\Pr(2)}{\Pr(III|1)\Pr(1) + \Pr(III|2)\Pr(2) + \Pr(III|3)\Pr(3)} \\ &= \frac{(1)(1/3)}{(1/2)(1/3) + (1)(1/3) + (0)(1/3)} = \frac{2}{3}\end{aligned}$$

$$\begin{aligned}\Pr(1|III) &= \frac{\Pr(III|1)(\Pr(1))}{\Pr(III|1)\Pr(1) + \Pr(III|2)\Pr(2) + \Pr(III|3)\Pr(3)} \\ &= \frac{(1/2)(1/3)}{(1/2)(1/3) + (1)(1/3) + (0)(1/3)} = \frac{1}{3}\end{aligned}$$

Therefore, you are better off switching!

Chapter 4: Decision Tree, CART Algorithm and Random Forest

4 Chapter 4: Decision Tree, CART Algorithm and Random Forest

4.1 Problem & Motivation

4.2 Decision Trees

4.3 Important Terminology related to Decision trees

4.4 Advantages of ID3 Algorithm

4.5 Limitations of ID3 Algorithm

4.6 C4.5 and 5.0 Algorithms

4.7 C5.0 Algorithm

4.8 Classification and regression trees (CART) Algorithm

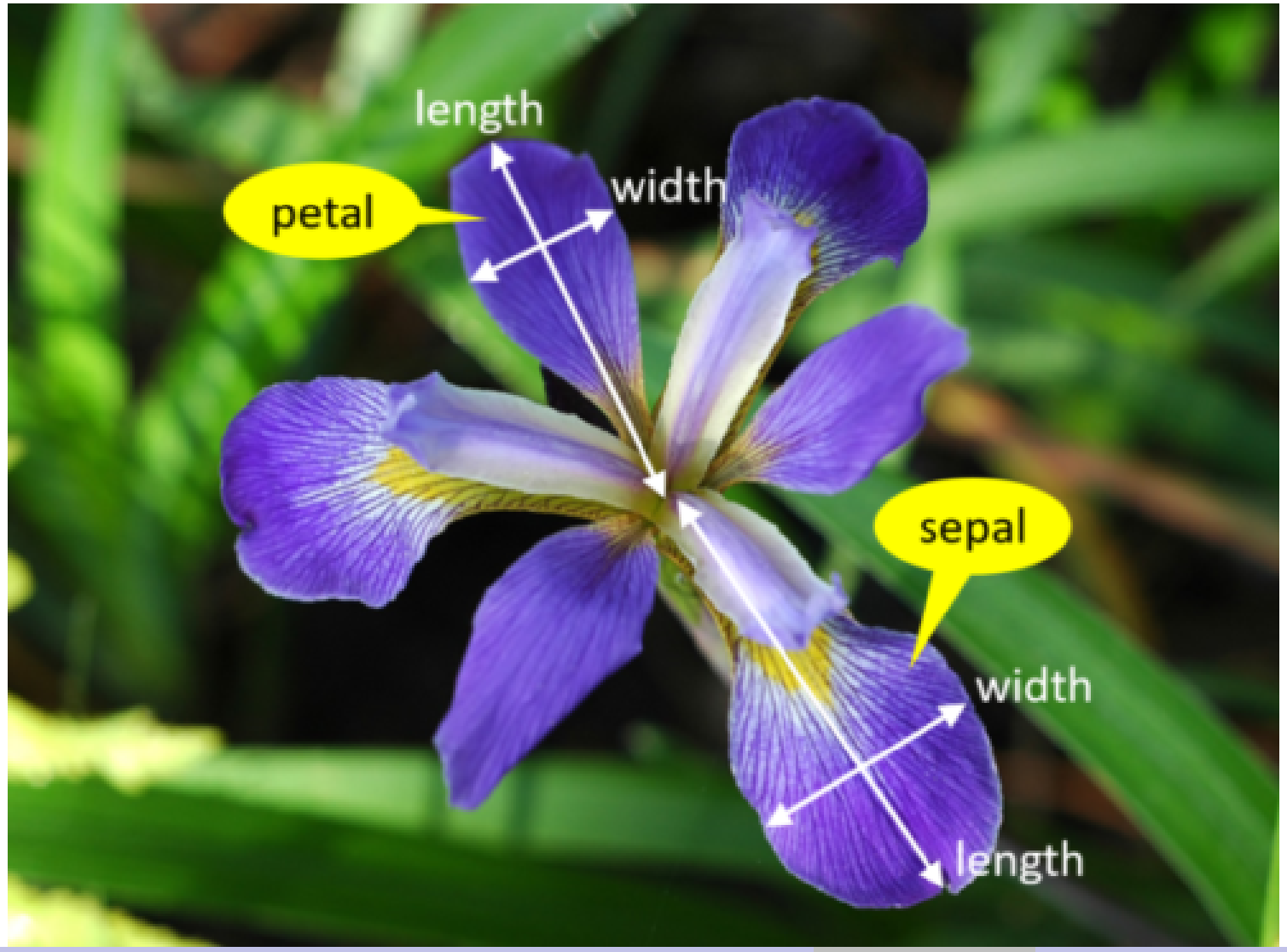
4.9 Random Forest

4.10 Python code: Decision Tree, CART Analysis and Random Forest

4.11 R code: Decision Trees

Problem & Motivation

- **Flower characteristics (variables)**



Name of the species of the Iris Flower

Data Matrix for iris data

	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

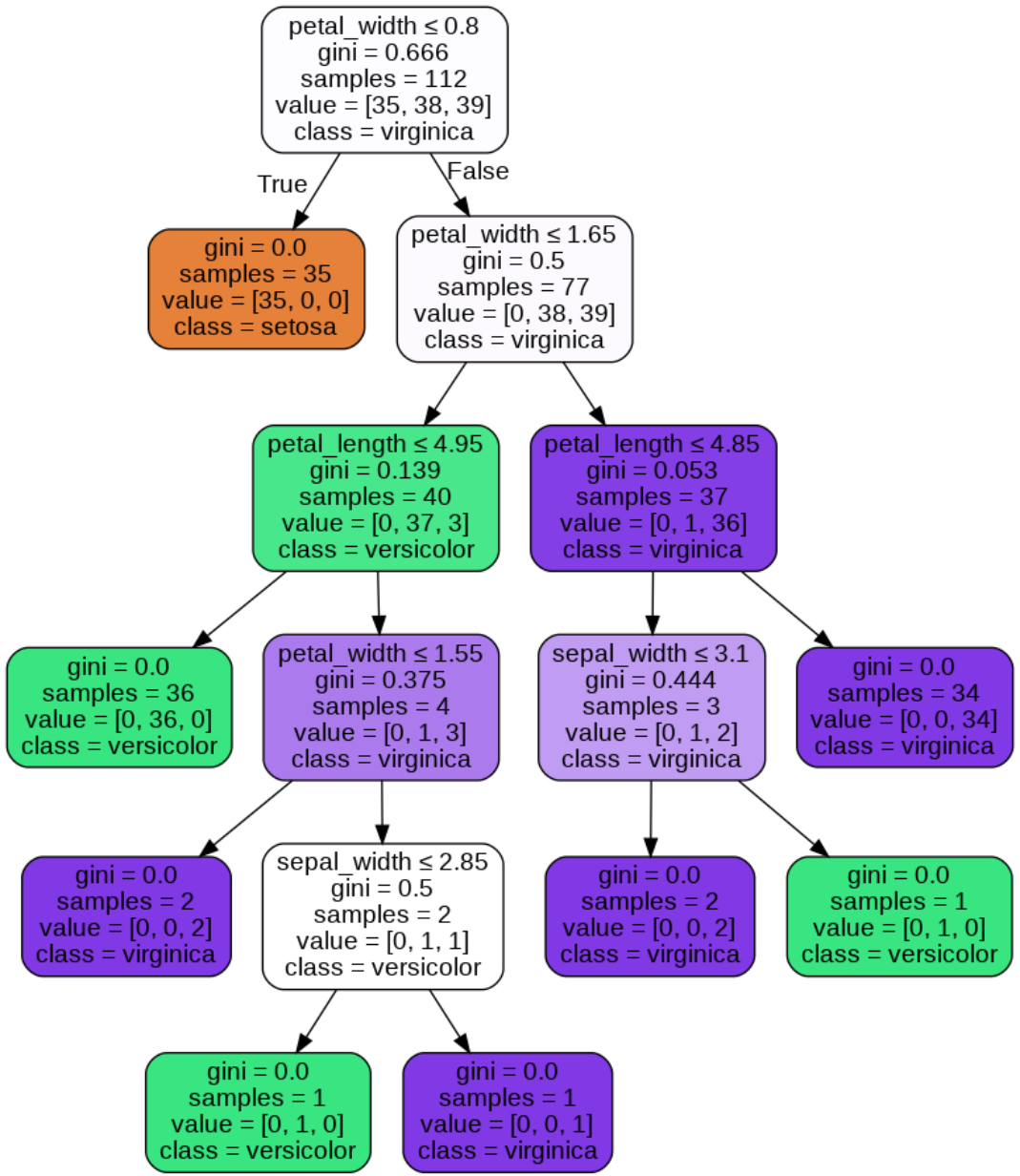
Problem & Motivation: new observations

1. suppose you have following characteristics of six flowers

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	6.70	3.00	5.20	2.30
2	6.30	2.50	5.00	1.90
3	6.50	3.00	5.20	2.00
4	6.20	3.40	5.40	2.30
5	5.90	3.00	5.10	1.80
6	1.00	2.00	1.00	0.40

2. what are the species name of these flower?
3. how do you classify these flower to Versicolor, Setosa or Virginica species?
4. how the variables:Sepal.Length, Sepal.Width, Petal.Length and Petal.Width are used to predict the Species of flowers?

Solution: CART Algorithm (in Python)

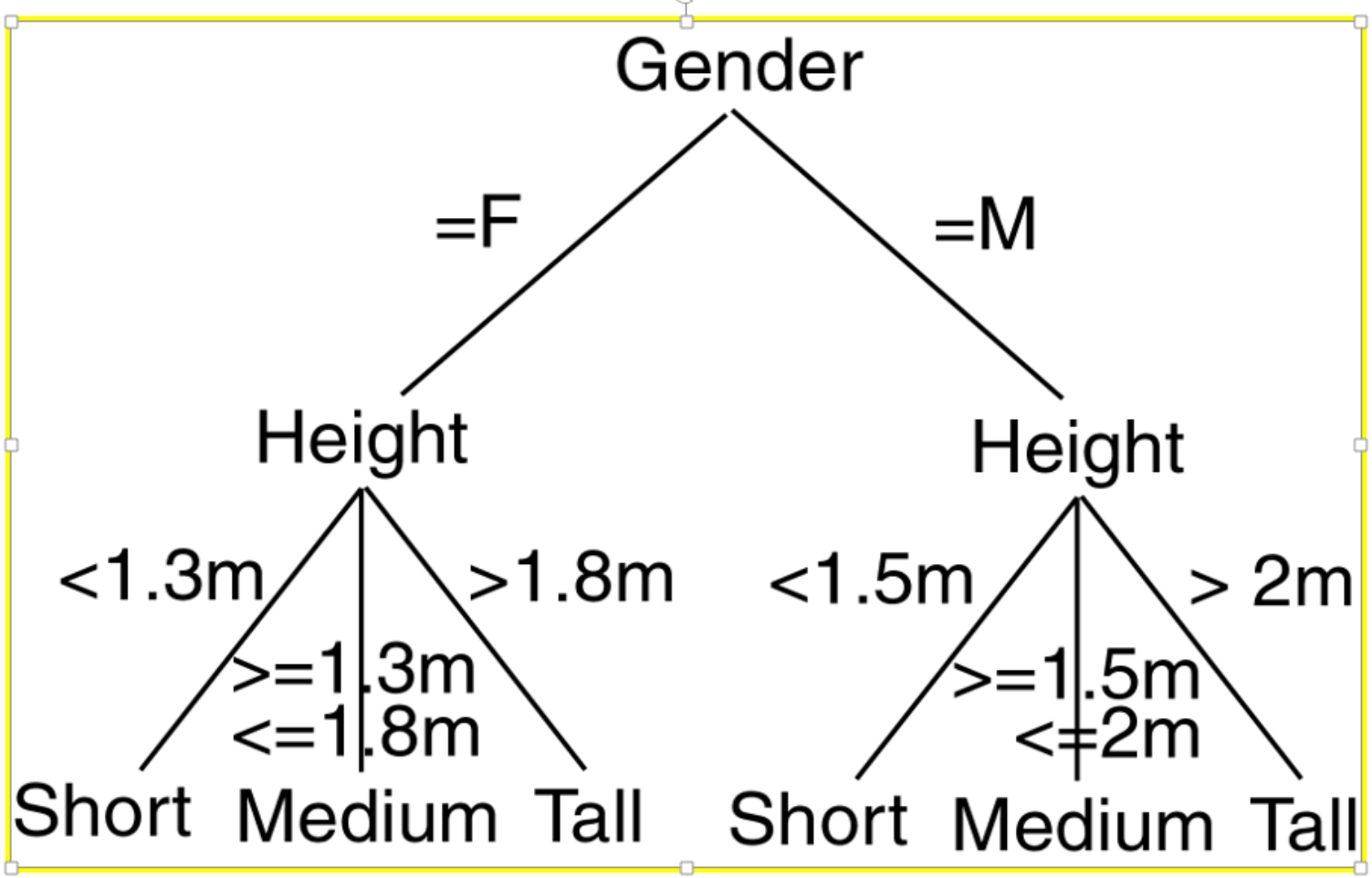


Example: given in Dunham, M. H. (2003)

TABLE 4.1: Data for Height Classification

Name	Gender	Height	Output1	Output2
Kristina	F	1.6 m	Short	Medium
Jim	M	2 m	Tall	Medium
Maggie	F	1.9 m	Medium	Tall
Martha	F	1.88 m	Medium	Tall
Stephanie	F	1.7 m	Short	Medium
Bob	M	1.85 m	Medium	Medium
Kathy	F	1.6 m	Short	Medium
Dave	M	1.7 m	Short	Medium
Worth	M	2.2 m	Tall	Tall
Steven	M	2.1 m	Tall	Tall
Debbie	F	1.8 m	Medium	Medium
Todd	M	1.95 m	Medium	Medium
Kim	F	1.9 m	Medium	Tall
Amy	F	1.8 m	Medium	Medium
Wynette	F	1.75 m	Medium	Medium

Decision Tree Example



Decision Trees

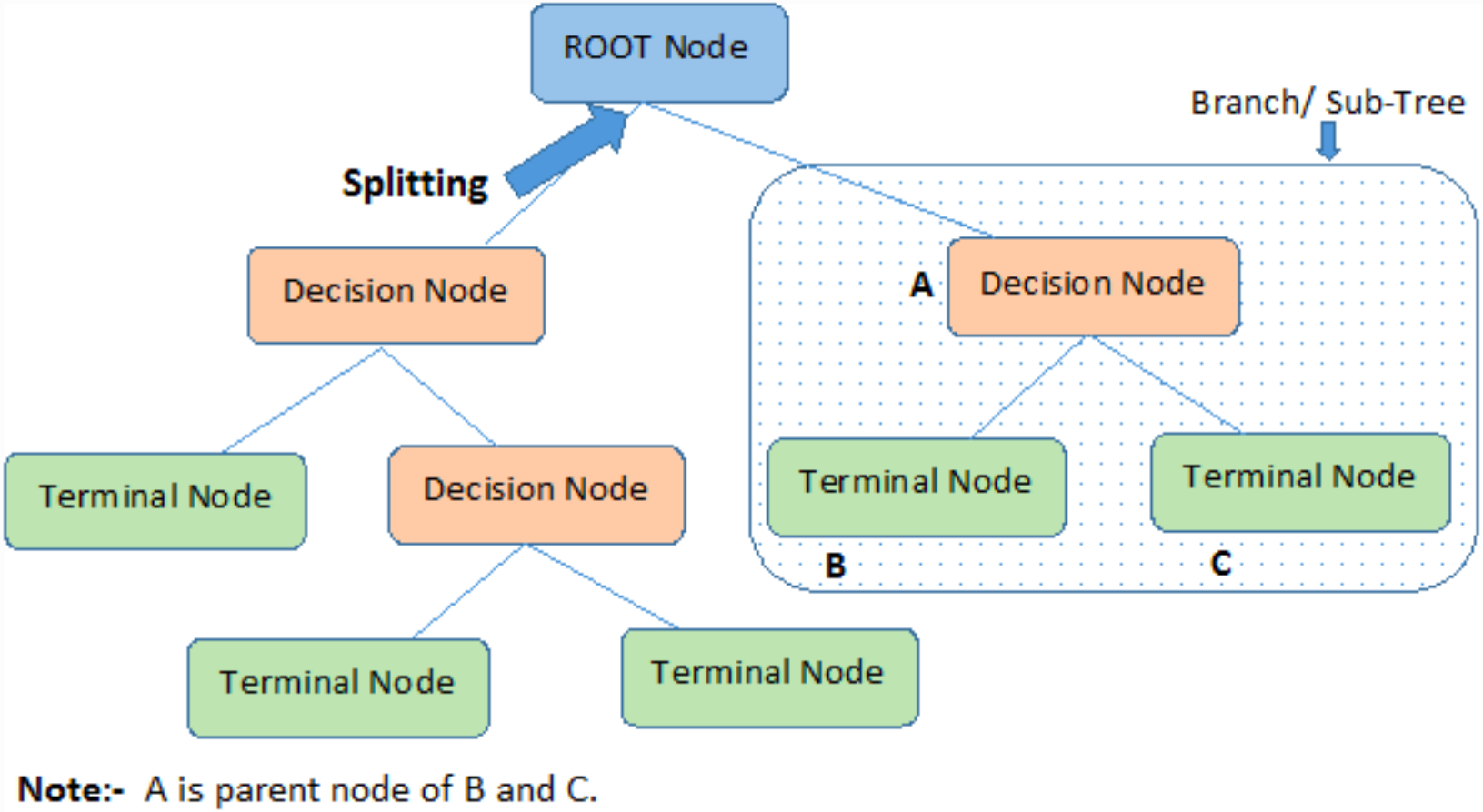
- decision tree is the most useful algorithm belongs to the family of supervised learning algorithms
- it is the non-parametric supervised learning approach, and can be applied to both regression and classification problems.
- unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too
- Various Decision Tree Algorithms
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 Algorithm
  - C5.0 Algorithms
  - CART (Classification and Regression trees) Algorithms
  - Random Forest
  - CHAID (Chi-square Automatic Interaction Detector) Algorithms

Decision Trees

- the goal of using a decision tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data)
- continuous and categorical decision tree
  - categorical variable decision tree: decision tree which has a categorical target variable ⇒ ID3 proposed by Ross Quinlan
  - continuous variable decision tree: decision tree has a continuous target variable ⇒ C4.5 (and also C5.0) improved by Ross Quinlan

Important Terminology related to Decision trees

1. Root Node: it represents the entire population or sample and this further gets divided into two or more homogeneous sets
2. Splitting: it is a process of dividing a node into two or more sub-nodes
3. Decision Node: when a sub-node splits into further sub-nodes, then it is called the decision node
4. Leaf / Terminal Node: nodes do not split is called Leaf or terminal node
5. Pruning: when we remove sub-nodes of a decision node, this process is called pruning — you can say the opposite process of splitting
6. Branch / Sub-Tree: a subsection of the entire tree is called branch or sub-tree
7. Parent and Child Node: a node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node



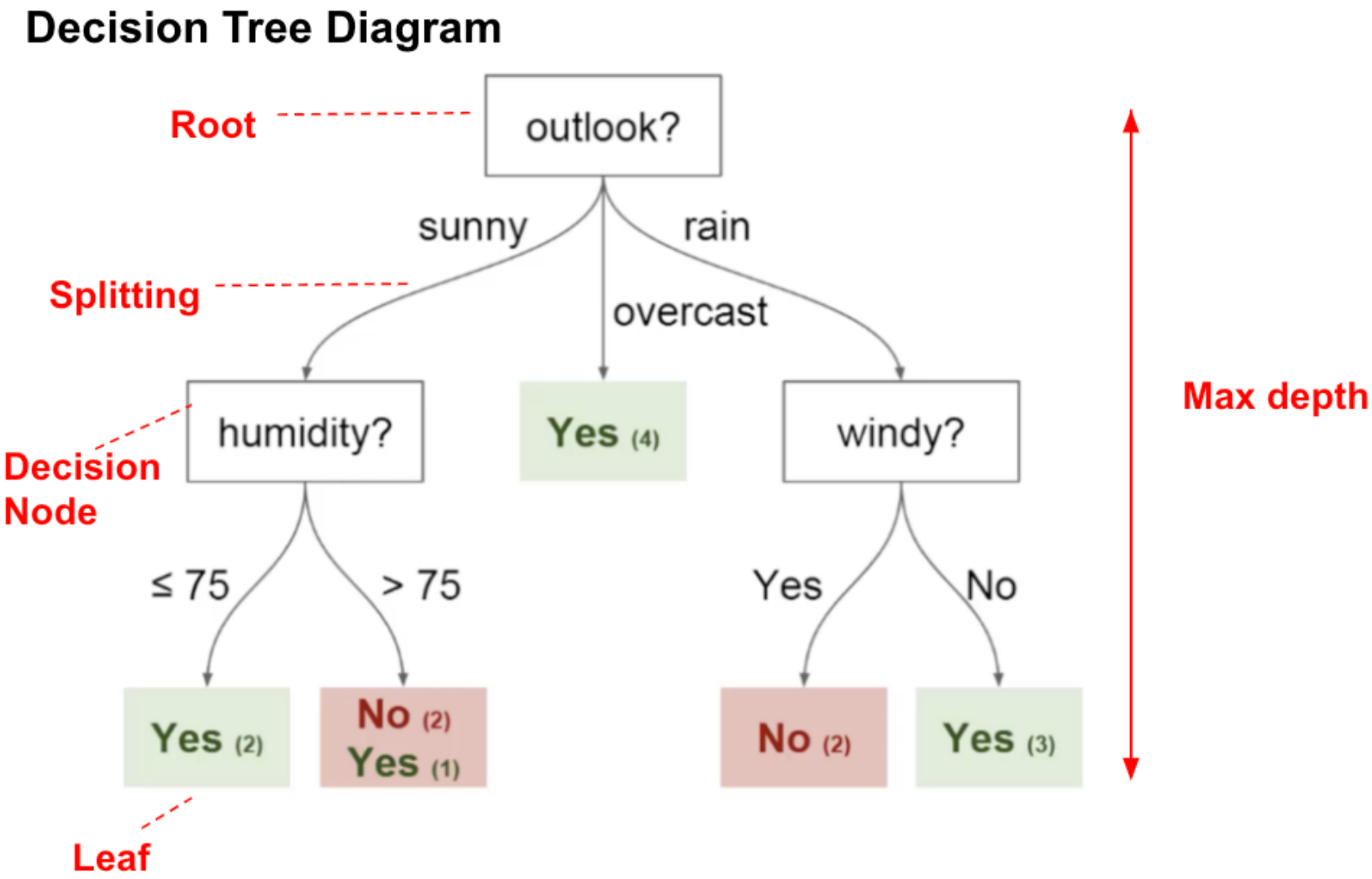
It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Decision tree with an Example

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

How do you predict class label for the following example...?

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	Normal	True	?



The maximum depth of a decision tree is simply the largest possible length between the root to a leaf.

Attribute Selection Measures: how to choose the best attribute in Decision Tree?

If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy. For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criteria like :

- Entropy and Information gain
  —ID3 chooses split attribute with the highest information gain
- Gain (used in ID3)
- Gain Ratio (used in C4.5)
- Gini Index (used in CART Algorithm)
- Reduction in Variance
- Chi-Square, etc.

Information and Entropy

1. given probabilities $p_1, p_2, ..., p_s$ whose sum is 1, Entropy is defined as:

$$H(p_1, p_2, ..., p_s) = \sum_{i=1}^{s} (p_i \log(1/p_i))$$

2. Entropy measures the amount of randomness or surprise or uncertainty
3. certainly, when all data in a set belong to a single class, there is no uncertainty and in this case the entropy is zero
4. goal in classification
   - no surprise
   - entropy = 0

ID3 (Iterative Dichotomiser 3) Algorithm

ID3 (Iterative Dichotomiser 3) Algorithm

- ID3 stands for Iterative Dichotomiser 3 . This algorithm iteratively divides features into two or more groups at each step. Here "Iterative" means continuous/repeated and "dichotomiser" means dividing.
- ID3 follows a top-down approach i.e., the tree is built from the top and at every step greedy approach is applied. The greedy approach means that at each iteration we select the best feature at the present moment to create a node and this node is again split after applying some statistical methods.
- ID3 generally is *not a very ideal algorithm as it overfits when the data is continuous*.

- the ID3 algorithm begins with the original set S as the root node
- the objective of decision tree classification is to iteratively partition the given data set into subsets where all elements in each final subset belong to the same class
- on each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy $\mathrm{H}(S)$ or the information gain $\mathrm{Gain}(D, S)$ of that attribute
- ID3 chooses the attribute with the highest information gain (which has the smallest entropy):

$$\mathrm{Gain}(D, S) = H(D) - \sum_{i=1}^{s} \Pr(D_i) H(D_i)$$

- the value which maximizes the gain would be the threshold

Example: given in Dunham, M. H. (2003)

TABLE 4.1: Data for Height Classification

Name	Gender	Height	Output1	Output2
Kristina	F	1.6 m	Short	Medium
Jim	M	2 m	Tall	Medium
Maggie	F	1.9 m	Medium	Tall
Martha	F	1.88 m	Medium	Tall
Stephanie	F	1.7 m	Short	Medium
Bob	M	1.85 m	Medium	Medium
Kathy	F	1.6 m	Short	Medium
Dave	M	1.7 m	Short	Medium
Worth	M	2.2 m	Tall	Tall
Steven	M	2.1 m	Tall	Tall
Debbie	F	1.8 m	Medium	Medium
Todd	M	1.95 m	Medium	Medium
Kim	F	1.9 m	Medium	Tall
Amy	F	1.8 m	Medium	Medium
Wynette	F	1.75 m	Medium	Medium

Example: given in Dunham, M. H. (2003)

1. the beginning state of the training data in Table 4.1 (in the last slide) with the Output1 classification is that (4/ 15) are short, (8/15) are medium, and (3/15) are tall
2. thus, the entropy of the starting set is

$$\frac{4}{15}\log\left(\frac{15}{4}\right)+\frac{8}{15}\log\left(\frac{15}{8}\right)+\frac{3}{15}\log\left(\frac{15}{3}\right)=0.4384$$

3. Information Gain using gender:
   - Female: 3/9 log(9/3)+6/9 log(9/6)=0.2764
   - Male: 1/6 (log 6/1) + 2/6 log(6/2) + 3/6 log(6/3) = 0.4392
   - Weighted sum: (9/15)(0.2764) + (6/15)(0.4392) = 0.34152
   - Information Gain: 0.4384 – 0.34152 = 0.09688

4. Information Gain using height:
   - in a height attribute, we have two tuples that are 1.6, two are 1.7, one is 1.75, two are 1.8, one is 1.85, one is 1.88, two are 1.9, one is 1.95, one is 2, one is 2.1, and one is 2.2
   - determining the split values for height is not easy
   - even though the training data set has these 11 values, we know that there will be many more. Just as with continuous data, we divide into ranges:

$$(0, 1.6], (1.6, 1.7], (1.7, 1.8], (1.8, 1.9], (1.9, 2.0], (2.0, \infty)$$

   - there are
     - 2 tuples in the first division with entropy $(2/2(0) + 0 + 0) = 0$,
     - 2 in (1.6, 1.7] with entropy $(2/2(0) + 0 + 0) = 0$,
     - 3 in (1 .7, 1.8] with entropy $(0 + 3/3(0) + 0) = 0$,
     - 4 in (1.8, 1.9] with entropy $(0+4/4(0) +0) = 0$,
     - 2 in (1 .9, 2.0] with entropy $(0+ 1/2\ (0.301)+ 1\ /2(0.301)) = 0.301$, and
     - 2 in the last with entropy $(0 + 0 + 2/2(0)) = 0$
   - all of these states are completely ordered and thus an entropy of 0 except for the (1.9, 2.0] state

5. Information Gain using height: $0.4384 - (2/15)(0.301) = 0.3983$
6. choose height as first splitting attribute due to the highest information gain

Assumptions While Creating a Decision Tree

- all the training data is passed to the root node. Training data is the data that we use to train the machine model and the model learns using this data and makes predictions on the test data
- the features (or) the attributes in the dataset are preferred to be categorical (or) discrete
- if the features are continuous in the data set, they are discretized before data modeling
- recursive distribution of the records is done on the values of the feature/attribute
- the order of having a root node followed by internal nodes followed by leaf nodes is done using statistical methods

Advantages of ID3 Algorithm

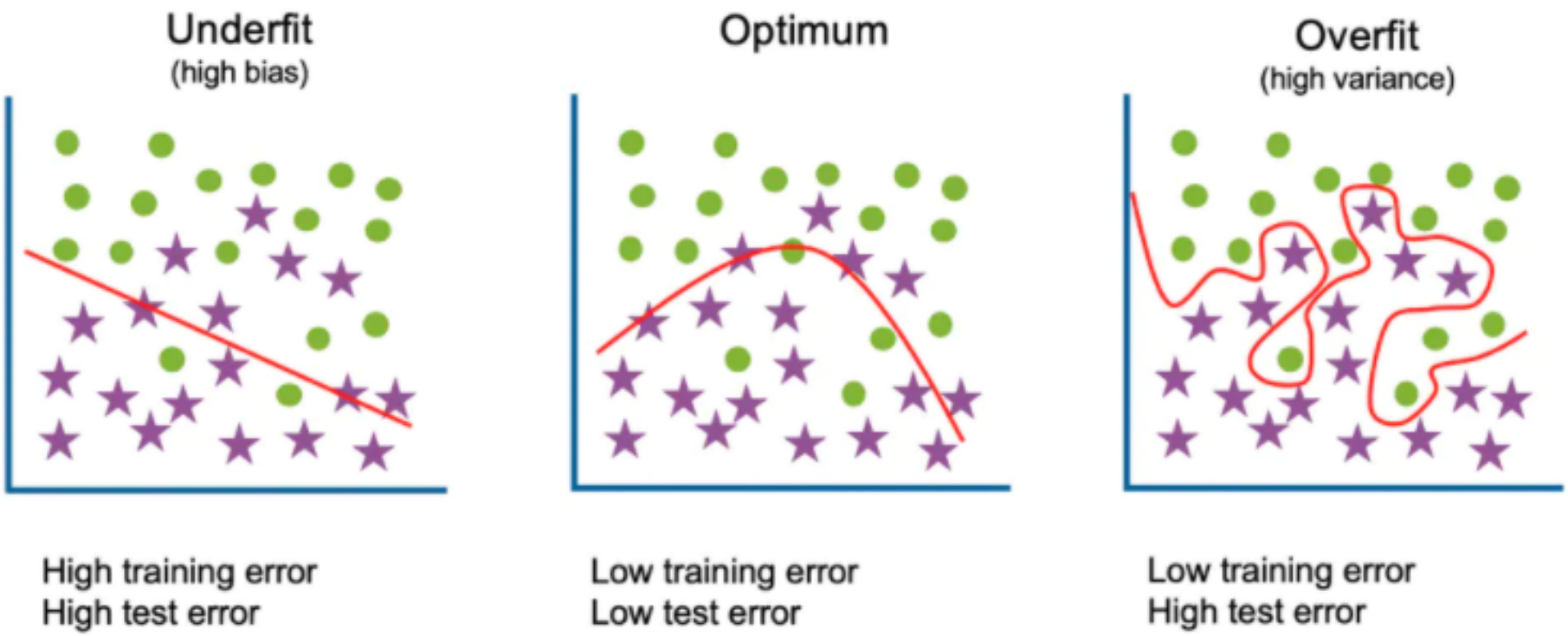
- simple to understand and interpret.
- able to handle both numerical and categorical data
- requires little data preparation
- uses a white box or open-box model
- possible to validate a model using statistical tests
- performs well with large datasets
- robust against co-linearity, particularly boosting
- decision trees can approximate any Boolean function
- ...

Limitations of ID3 Algorithm

- ID3 does not guarantee an optimal solution
- trees can be very non-robust
  — a small change in the training data can result in a large change in the tree and consequently the final predictions
- ID3 is harder to use on continuous data
- does not handle numeric attributes and missing values
- only one attribute at a time is tested for making a decision
- correlations among attributes in the database are ignored by the decision tree process
- Decision tree learners create underfit trees if some classes are imbalanced
- since the decision tree is constructed from the training data, ID3 can overfit the training data

what do you mean by overfitting problem?

Overfitting



C4.5 Algorithm

C4.5 Algorithm

Ross Quinlan, inventor of ID3, made some improvements for some of these limitations and created a new algorithm named C4.5 algorithm

- the algorithm can create a more generalized models including continuous data and could handle missing data
- additionally, some resources such as Weka named this algorithm as J48
- C4.5 handles overfitting by the method of "pruning" i.e., it removes the branches/subpart of the tree that *"does not hold much importance (or) is redundant*.
- to be specific, C4.5 follows post pruning i.e removing branches after the tree is created.

- Splitting
  - the ID3 approach favors attributes with many divisions and thus may lead to overfitting
  - an improvement can be made by taking into account the cardinality of each division
  - this approach uses the GainRatio

$$GainRatio(D, S) = \frac{\text{Gain}(D, S)}{H\left(\frac{|D_1|}{|D|}, \ldots, \frac{|D_s|}{|D|}\right)}$$

  - for splitting purposes, C4.5 uses the largest GainRatio that ensures a larger than average information gain
  - the value which maximizes the largest GainRatio would be the threshold.

Example: given in Dunham, M. H. (2003)

Example 4.8 shows the calculation of GainRatio for the first split in Example 4.7

EXAMPLE 4.8

To calculate the GainRatio for the gender split, we first find the entropy associated with the split ignoring classes

$$H\left(\frac{9}{15}, \frac{6}{15}\right) = \frac{9}{15}\log\left(\frac{15}{9}\right) + \frac{6}{15}\log\left(\frac{15}{6}\right) = 0.292 \tag{4.24}$$

This gives the GainRatio value for the gender attribute as

$$\frac{0.09688}{0.292} = 0.332 \tag{4.25}$$

The entropy for the split on height (ignoring classes) is

$$H\left(\frac{2}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{2}{15}\right) \tag{4.26}$$

Disadvantages of C4.5 Algorithm

- C4.5 Algorithm is time consuming and it is not practical for large datasets
- C4.5 constructs empty branches with zero values
- over fitting happens when algorithm model picks up data with uncommon characteristics, especially when data is noisy

C5.0 Algorithm

C5.0 algorithm offers a number of improvements on C4.5 algorithm

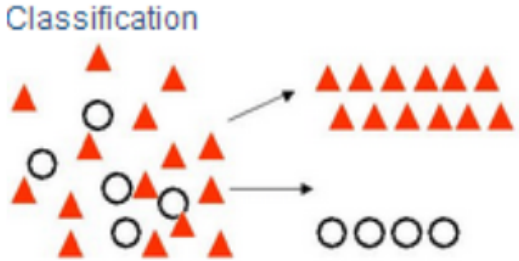
- Speed - C5.0 is significantly faster than C4.5 (several orders of magnitude)
- Memory usage - C5.0 is more memory efficient than C4.5
- Smaller decision trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees
- Support for boosting - boosting improves the trees and gives them more accuracy
- Weighting - C5.0 allows you to weight different cases and misclassification types
- Winnowing - a C5.0 option automatically winnows the attributes to remove those that may be unhelpful

Classification and Regression Trees (CART) Algorithm

Classification and regression trees (CART)

Classification and Regression Trees or CART for short is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification and regression predictive modeling problems

- **Classification Trees**: where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.



- **Regression Trees**: where the target variable is continuous and tree is used to predict it's value.

Classification and regression trees (CART)

- ID3 uses *information gain* whereas C4.5 uses *gain ratio* for splitting
- classically, this algorithm is referred to as "decision trees", but on some statistical platforms like R they are referred to by the more modern term CART
- CART uses *Gini index* in the process of splitting the dataset into a decision tree
- *Gini index* is a metric to measure how often a randomly chosen element would be incorrectly identified
- a gini index is used to find the best split

$$\text{gini} = 1 - \sum p_j^2$$

  where p_j is the frequency of class C_j in database D
- the CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees

Steps to pick a decision Node

- Calculate Gini Index for the each attribute by using

$$\text{gini index} = 1 - \sum p_j^2$$

  where p_j is the frequency of label class C_j in database D
- Calculate weighted sum of gini indexes for that feature
- Pick the attribute with the lowest gini index value
- Repeat above steps until a generalized tree has been created

CART Algorithm Step by Step with an Example

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

How do you predict class label for the following example...?

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	Normal	True	?

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = $1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$

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

Gini(Outlook=Rain) = $1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$

Then, we will calculate weighted sum of gini indexes for outlook feature.

Gini(Outlook) = $(5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 = 0.171 + 0 + 0.171 = 0.342$

Temperature

Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

Gini(Temp=Hot) = $1 - (2/4)^2 - (2/4)^2 = 0.5$

Gini(Temp=Cool) = $1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$

Gini(Temp=Mild) = $1 - (4/6)^2 - (2/6)^2 = 1 - 0.444 - 0.111 = 0.445$

We'll calculate weighted sum of gini index for temperature feature

Gini(Temp) = $(4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 = 0.142 + 0.107 + 0.190 = 0.439$

Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = $1 - (3/7)^2 - (4/7)^2 = 1 - 0.183 - 0.326 = 0.489$

Gini(Humidity=Normal) = $1 - (6/7)^2 - (1/7)^2 = 1 - 0.734 - 0.02 = 0.244$

Weighted sum for humidity feature will be calculated next

Gini(Humidity) = $(7/14) \times 0.489 + (7/14) \times 0.244 = 0.367$

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

Gini(Wind=Weak) = $1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$

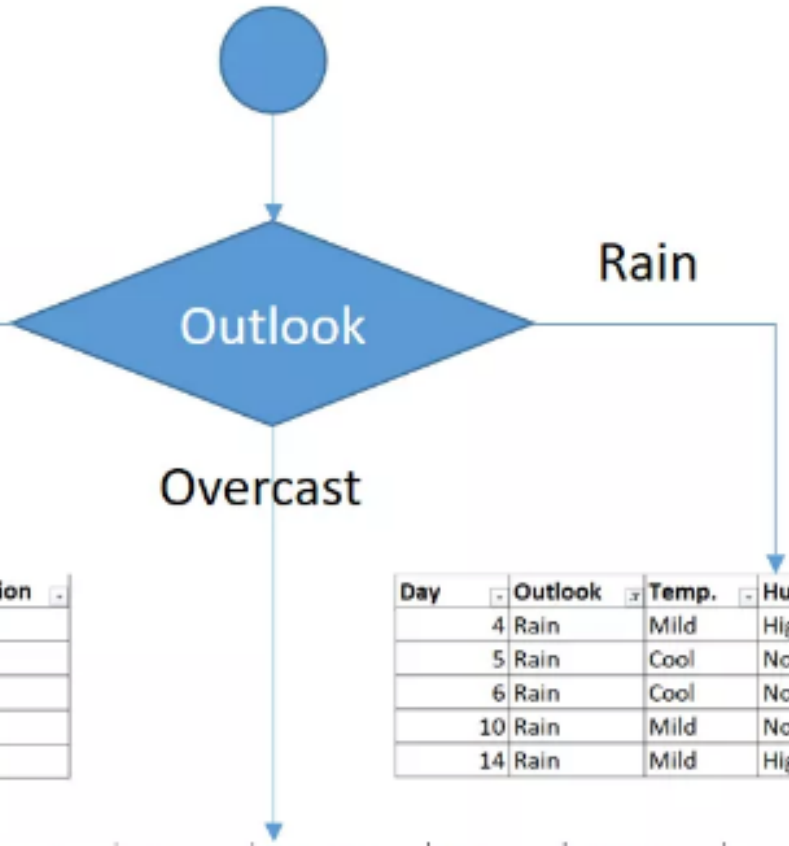
Gini(Wind=Strong) = $1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Wind) = $(8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$

Time to decide

We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

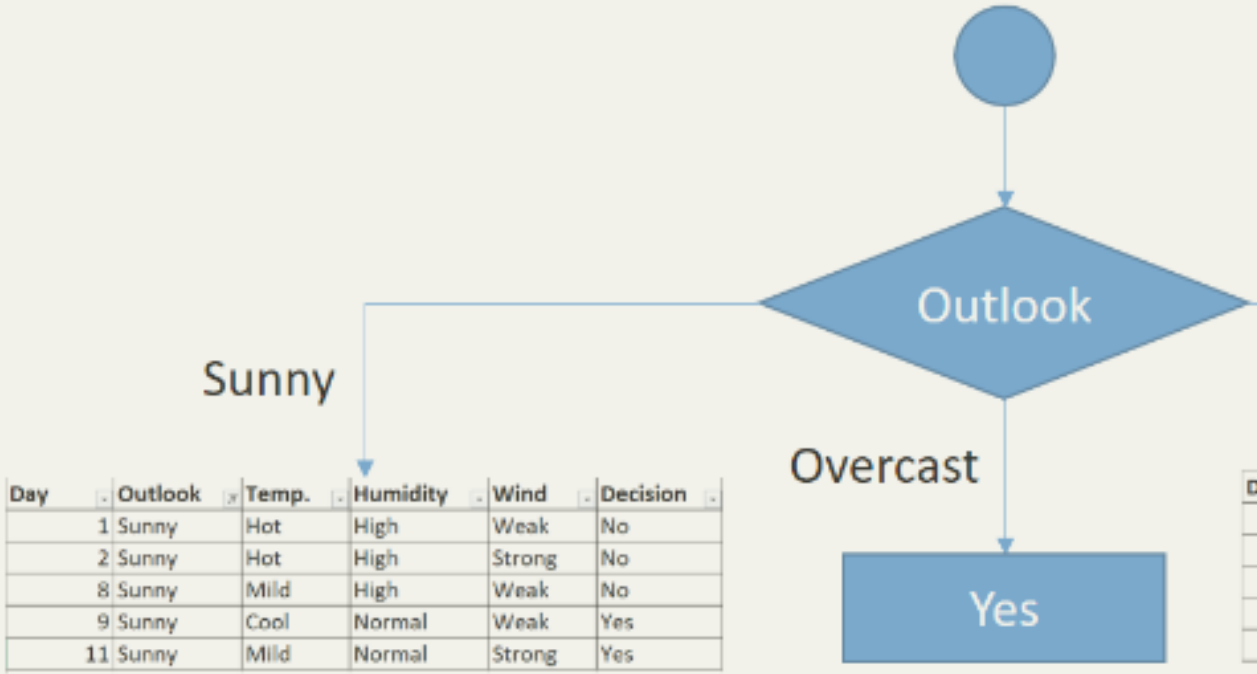


Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

You might realize that sub dataset in the overcast leaf has only yes decisions. This means t
overcast leaf is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	De
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Tree is over for overcast outlook leaf

We will apply same principles to those sub datasets in the following steps.

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$

Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$

Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.) = $(2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2$

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$

Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$

Gini(Outlook=Sunny and Humidity) = $(3/5)x0 + (2/5)x0 = 0$

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak) = $1 - (1/3)^2 - (2/3)^2 = 0.266$

Gini(Outlook=Sunny and Wind=Strong) = $1- (1/2)^2 - (1/2)^2 = 0.2$

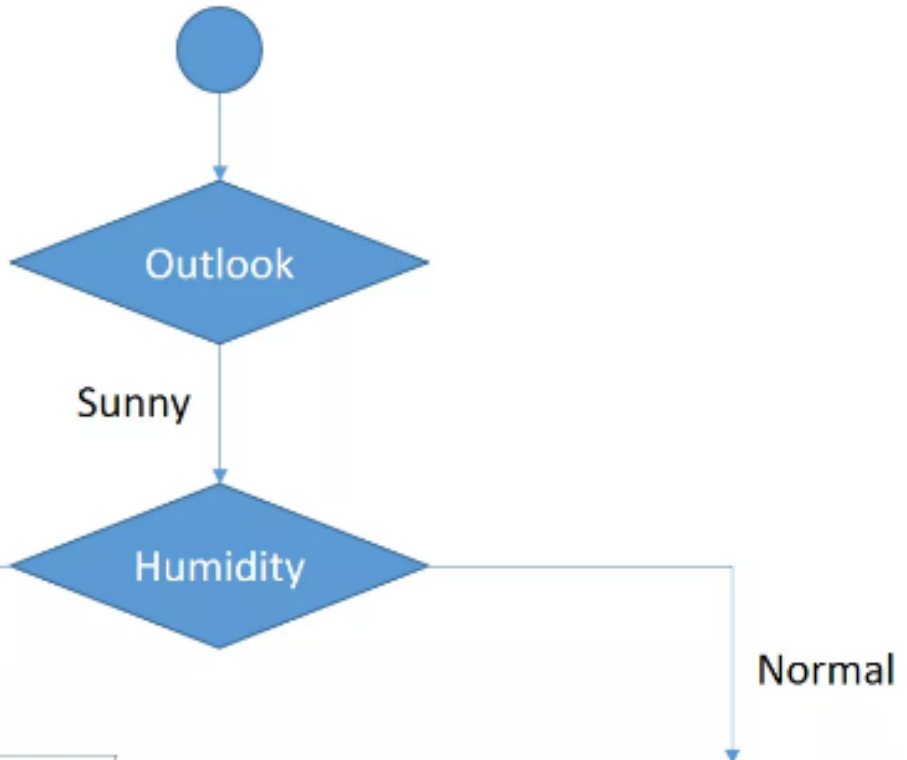
Gini(Outlook=Sunny and Wind) = $(3/5)\times0.266 + (2/5)\times0.2 = 0.466$

Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

We'll put humidity check at the extension of sunny outlook.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

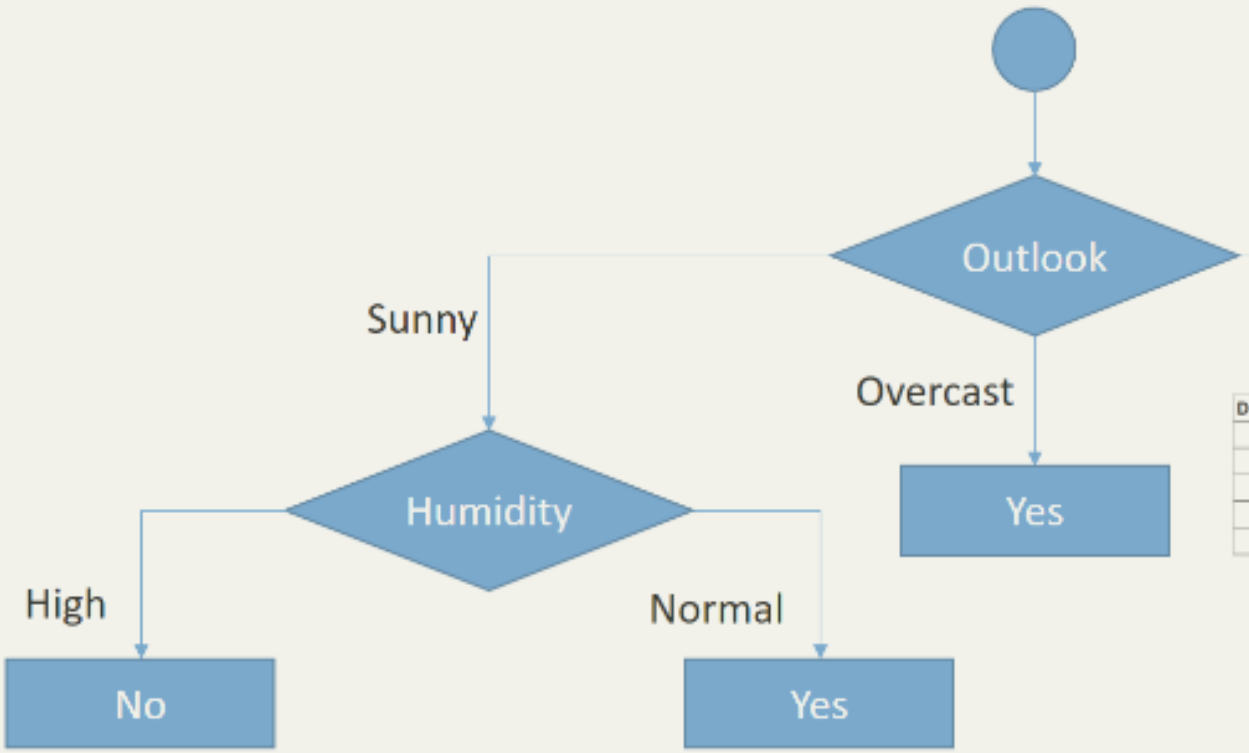
Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

We'll put humidity check at the extension of sunny outlook.

As seen, decision is always no for high humidity and sunny outlook. On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Now, we need to focus on rain outlook.

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We'll calculate gini index scores for temperature, humidity and wind features when outlook is rain.

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$

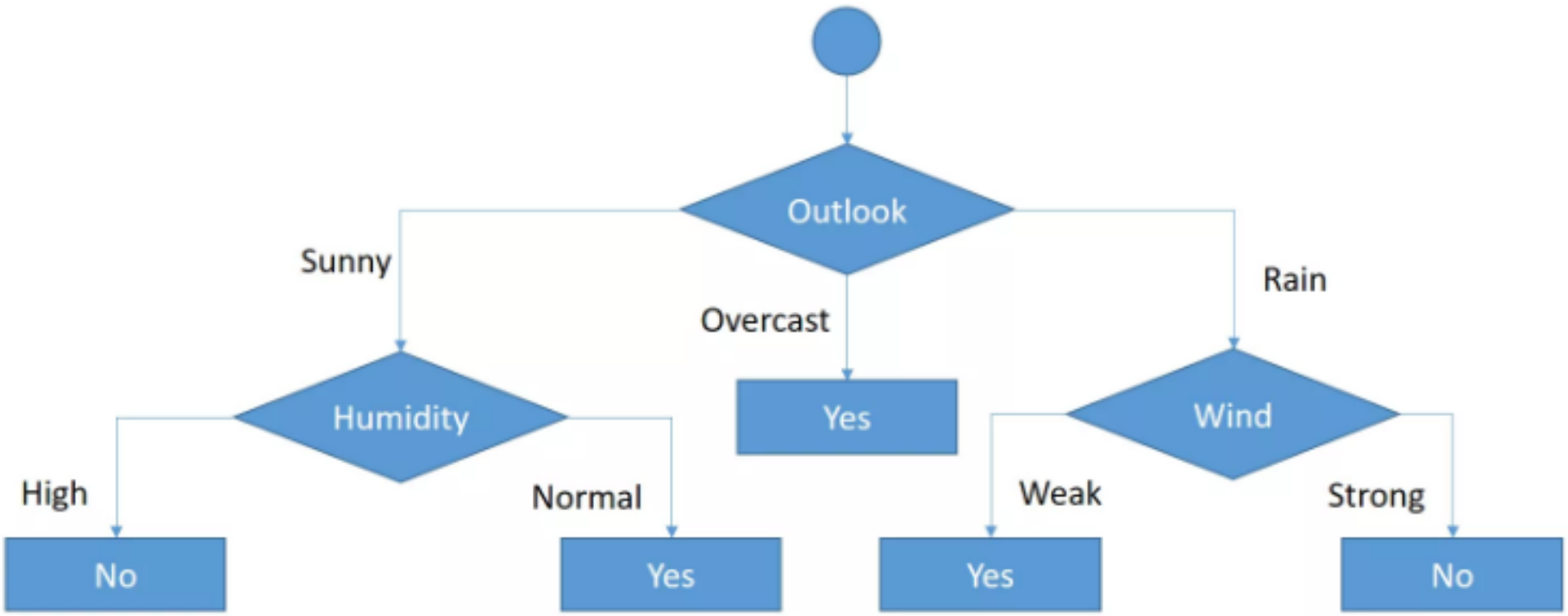
Gini(Outlook=Rain and Wind) = $(3/5)\times 0 + (2/5)\times 0 = 0$

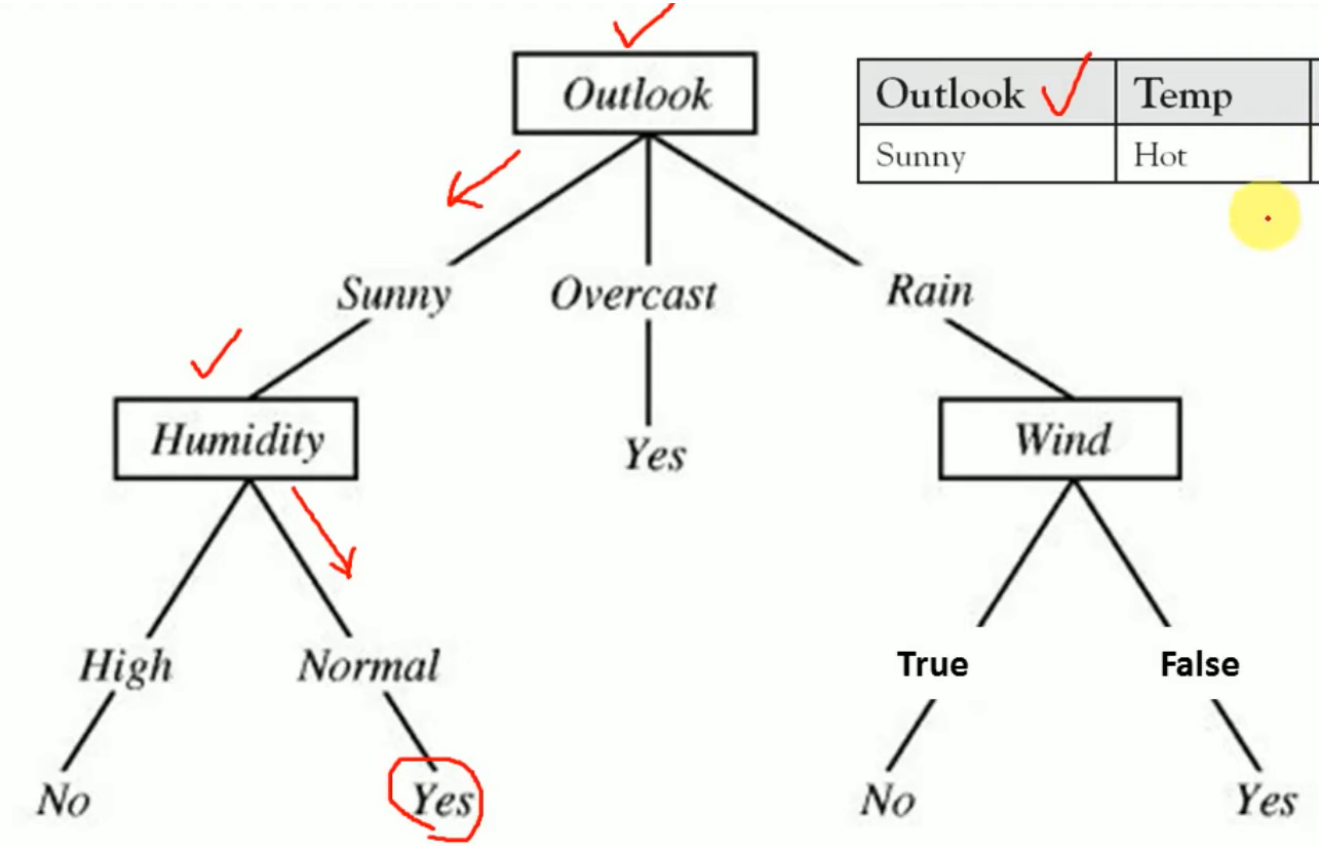
Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Put the wind feature for rain outlook branch and monitor the new sub data sets.


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Weak
Strong
No
Yes
Yes
No



Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	Normal	True	? Yes

Python code: CART

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

dataset=pd.read_csv('temp.csv')
dataset=pd.DataFrame(data=dataset.iloc[:,1:6].values,columns=["outlook",
filter = dataset["outlook"]=="Rainy"
dataset.where(filter).count()
dataset_encoded=dataset.iloc[:,0:5]
le=LabelEncoder()

for i in dataset_encoded:
dataset_encoded[i]=le.fit_transform(dataset_encoded[i])

print(dataset_encoded)
print(dataset)
```

Python code: CART

```
#Feature Set
X=dataset_encoded.iloc[:,0:4].values
#Label Set
y=dataset_encoded.iloc[:,4].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1,rando

model=DecisionTreeClassifier(criterion='gini')
model.fit(X_train,y_train)


if model.predict([[0,1,0,1]])==1:
print("yes you can play")
else:
print("no you cant")
```

Entropy vs Gini Index (ID3 vs. CART Algorithm)

- The Entropy and Information Gain method focuses on purity and impurity in a node. The Gini Index or Gini Impurity measures the probability for a random instance being misclassified when chosen randomly. The lower the Gini Index, the better the lower the likelihood of misclassification.

Classification and regression trees (CART)

- CART is a technique that generates a binary decision tree
- each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric)
- the leaf nodes of the tree contain an output variable (y) which is used to make a prediction
- the selection of which input variable to use and the specific split or cut-point is chosen using a Twoing criterion
- the formula of Twoing criterion to choose split point, s, for node t:

$$\text{Twoing criterion: } \Phi(s/t) = 2P_L P_R \sum_{j=1}^{m} |\Pr(C_j|t_L) - \Pr(C_j|t_R)|$$

P_L, P_R probability that a tuple in the training set will be on the left or right side of the tree.

Example: given in Dunham, M. H. (2003)

TABLE 4.1: Data for Height Classification

Name	Gender	Height	Output1	Output2
Kristina	F	1.6 m	Short	Medium
Jim	M	2 m	Tall	Medium
Maggie	F	1.9 m	Medium	Tall
Martha	F	1.88 m	Medium	Tall
Stephanie	F	1.7 m	Short	Medium
Bob	M	1.85 m	Medium	Medium
Kathy	F	1.6 m	Short	Medium
Dave	M	1.7 m	Short	Medium
Worth	M	2.2 m	Tall	Tall
Steven	M	2.1 m	Tall	Tall
Debbie	F	1.8 m	Medium	Medium
Todd	M	1.95 m	Medium	Medium
Kim	F	1.9 m	Medium	Tall
Amy	F	1.8 m	Medium	Medium
Wynette	F	1.75 m	Medium	Medium

- at the start, there are six choices for split point:

$$\Phi(\text{Gender}) = 2(6/15)(9/15)(2/15 + 4/15 + 3/15) = 0.224$$
$$\Phi(1.6) = 0$$
$$\Phi(1.7) = 2(2/15)(13/15)(0 + 8/15 + 3/15) = 0.1690$$
$$\Phi(1.8) = 2(5/15)(10/15)(4/15 + 6/15 + 3/15) = 0.385$$
$$\Phi(1.9) = 2(9/15)(6/15)(4/15 + 2/15 + 3/15) = 0.256$$
$$\Phi(2.0) = 2(12/15)(3/15)(4/15 + 8/15 + 3/15) = 0.32$$

split at 1.8 due to the highest value

- CART handles missing data by simply ignoring that record in calculating the goodness of a split on that attribute

Scalable Decision Tree Techniques

- the SPRINT (Scalable PaRallelizable INduction of decision Trees) algorithm addresses the scalability issue by ensuring that the CART technique can be applied regardless of availability of main memory
- with SPRINT, a gini index is used to find the best split

$$\text{gini} = 1 - \sum p_j^2$$

  where p_j is the frequency of class C_j in database D
- the split with the best gini value is chosen
- when the target attribute is binary, the gini index and twoing criteria are equivalent
- with continuous data, the split point is chosen to be the midpoint of every pair of consecutive values from the training set

	Splitting Criteria	Attribute type	Missing values	Pruning Strategy	Outlier Detection
ID3	Information Gain	Handles only Categorical value	Do not handle missing values.	No pruning is done	Susceptible to outliers
CART	Towing Criteria	Handles both Categorical & Numeric value	Handle missing values.	Cost-Complexity pruning is used	Can handle Outliers
C4.5	Gain Ratio	Handles both Categorical & Numeric value	Handle missing values.	Error Based pruning is used	Susceptible to outliers

How to avoid/counter Overfitting in Decision Trees?

- the common problem with Decision trees, especially having a table full of columns, they fit a lot.
- if there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worse case it will end up making 1 leaf for each observation
- thus this affects the accuracy when predicting samples that are not part of the training set and this is a phenomenon known as overfitting
- here are two ways to remove overfitting:
  1. Pruning Decision Trees
  2. Random Forest

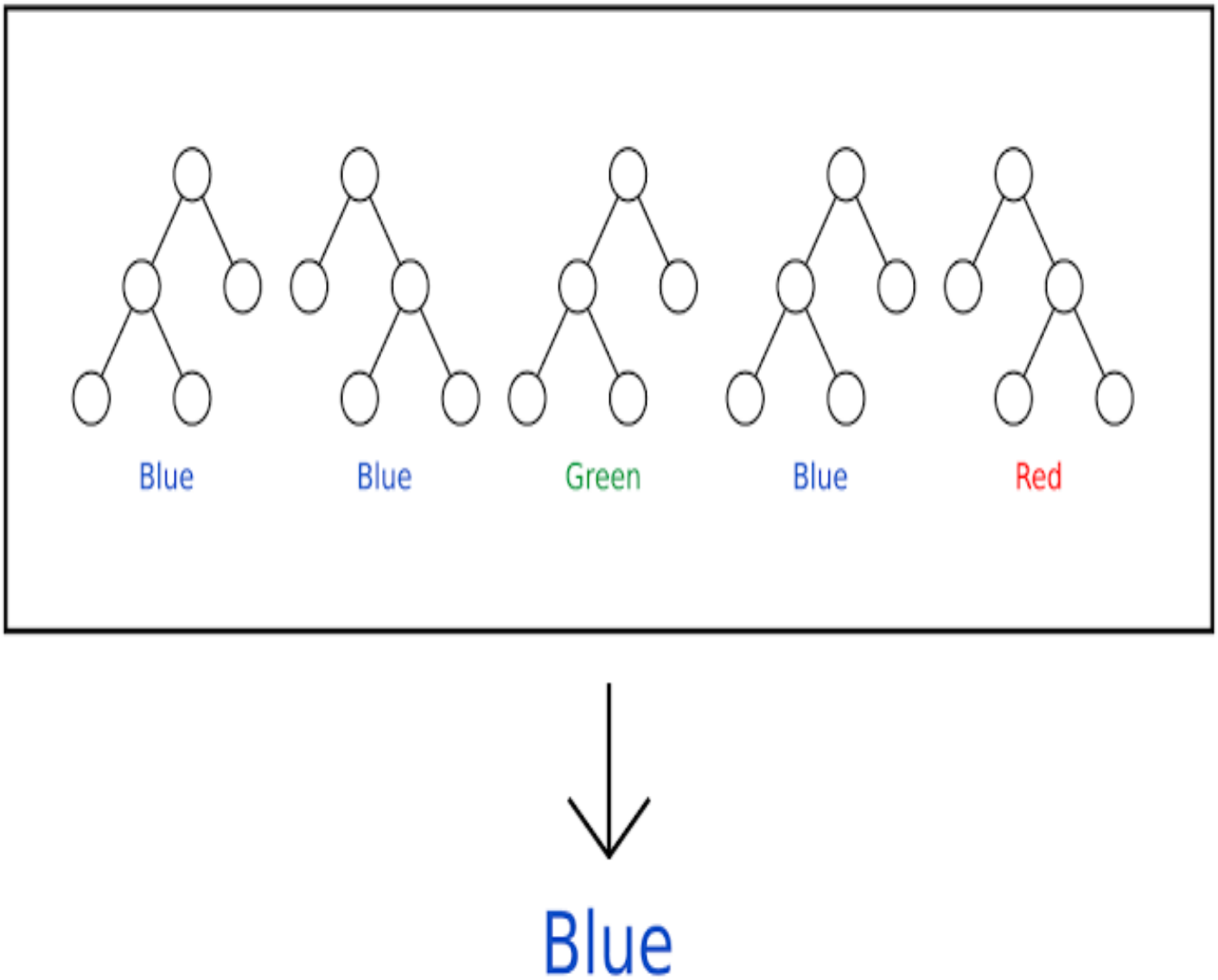
Pruning Decision Trees

- Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.
- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.
- There are mainly two types of tree pruning technology used:
  - Cost Complexity Pruning
  - Reduced Error Pruning.

Random Forest

random forest is an example of ensemble learning, in which we combine multiple machine learning algorithms to obtain better predictive performance

- why the name "Random"?
  - a random sampling of training data set when building trees
  - random subsets of features considered when splitting nodes
- a technique known as bagging is used to create an ensemble of trees where multiple training sets are generated with replacement
- the random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting
- it reduces the overfitting by averaging the result


Blue
Blue
Green
Blue
Red
Blue

CHAID (Chi-square Automatic Interaction Detector)

CHAID (Chi-square Automatic Interaction Detector)

- CHAID stands for Chi-square Automatic Interaction Detector
- it is known to be the oldest of all algorithms in history and is used very less these days
- in CHAID chi-square is the attribute selection measure to split the nodes when it's a classification-based use case and uses F-test as an attribute selection measure when it is a regression-based use case
- higher the chi-square value higher is the preference given to that feature
- the major difference between CHAID and CART is, CART splits one node into two nodes whereas CHAID splits one node into 2 or more nodes

Exercise

- Which is Better Linear or Tree-based Models?

Python code: Plot for the Decision Tree

```
import numpy as np
import pandas as pd
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
# Data Loading
url='https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c
df = pd.read_csv(url)
df.head()
x = df[['sepal_length', 'sepal_width', 'petal_length',
    'petal_width']]
y = df['species']
x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size=0.25, random_state =3)
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(x_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Python code: Decision Tree

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

x = df[['sepal_length', 'sepal_width', 'petal_length',
    'petal_width']]
y = df['species']
feature_cols = ['sepal_length', 'sepal_width',
    'petal_length', 'petal_width']
class_names=['setosa', 'versicolor', 'virginica']
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
filled=True, rounded=True,
special_characters=True,feature_names =
    feature_cols,class_names=class_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Python code: Random Forest

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
#Train the model using the training sets
    y_pred=clf.predict(X_test)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
clf.predict([[3, 5, 4, 2]])

from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets
    y_pred=clf.predict(X_test)
clf.fit(x_train,y_train)
import pandas as pd
feature_imp =
    pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=False)
feature_imp
```

Python code: Random Forest

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

Python code: Classification and Regression Tree (CART)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# load Boston Housing dataset
from sklearn.datasets import load_boston
boston_dataset = load_boston()

boston = pd.DataFrame(boston_dataset.data,
    columns=boston_dataset.feature_names)

boston['MEDV'] = boston_dataset.target
names = boston_dataset.feature_names

print(boston.head())
print(names)
print(boston.shape)
array = boston.values

X = array[:,0:13]
Y = array[:,13]
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=0.3, random_state=1234)
```

Python code: Classification and Regression Tree (CART)

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(max_leaf_nodes = 20)
rt = model.fit(X_train, Y_train)
rt
import pandas as pd
import numpy as np
import random as rnd

rnd.seed(123458)
X_new = X[rnd.randrange(X.shape[0])]
X_new = X_new.reshape(1,13)
YHat = model.predict(X_new)


df = pd.DataFrame(X_new, columns = names)
df["Predicted Price"] = YHat
df.head(1)
from sklearn.metrics import r2_score
YHat = model.predict(X_test)

r2 = r2_score(Y_test, YHat)
print("R-Squared = ", r2)
```

Python code: Classification and Regression Tree (CART)

```
from IPython.display import Image
from sklearn import tree
import matplotlib.image as mpimg
import pydotplus
import io
import matplotlib.pyplot as plt

import sys
str1 = "/usr/local/Cellar/graphviz/2.40.1_1/bin/"
sys.path.append(str1)

dot_data = io.StringIO()
tree.export_graphviz(rt, out_file=dot_data, filled=True,
    feature_names = names, class_names = 'MEDV')

# Draw graph
pydotplus.graph_from_dot_data(dot_data.getvalue()).write_png('dt.png')

plt.figure(figsize=(100, 100))
img = mpimg.imread('dt.png')
imgplot = plt.imshow(img)

plt.show()
```

Decision Trees in R (Regression & Classification)

- this page shows how to build a decision tree with R

```
# rm(list = ls())
## Example 1
library("party")
### classification
irisct <- ctree(Species ~ .,data = iris)
irisct
plot(irisct)
table(predict(irisct), iris$Species)
### estimated class probabilities, a list
tr <- treeresponse(irisct, newdata = iris[1:10,])
tr

### Example 2
### regression
airq <- subset(airquality, !is.na(Ozone))
airct <- ctree(Ozone ~ ., data = airq,
controls = ctree_control(maxsurrogate = 3))
airct
plot(airct)
mean((airq$Ozone - predict(airct))^2)
### extract terminal node ID, two ways
all.equal(predict(airct, type = "node"), where(airct))
pred<-predict(airct, type = "node")
pred
```

Decision Trees (Ordinal Regression & Survival Analysis)

- this page shows how to build a decision tree with R

```
### Exaple 3
### ordinal regression
data("mammoexp", package = "TH.data")
mammoct <- ctree(ME ~ ., data = mammoexp)
plot(mammoct)
### estimated class probabilities
treeresponse(mammoct, newdata = mammoexp[1:10,])
pred<-predict(mammoct, type = "node")
pred

### Exaple 4
### survival analysis
if (require("TH.data") && require("survival")) {
  data("GBSG2", package = "TH.data")
  GBSG2ct <- ctree(Surv(time, cens) ~ .,data = GBSG2)
  plot(GBSG2ct)
  treeresponse(GBSG2ct, newdata = GBSG2[1:2,])
}
```

Random Forest

```
### Example 5
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
library(randomForest)
rf <- randomForest(Species ~ ., data=trainData, ntree=100,
    proximity=TRUE)
table(predict(rf), trainData$Species)

# ?randomForest
print(rf)
plot(rf)
# Find the important variable
importance(rf)
varImpPlot(rf)
# Predict testData
irisPred <- predict(rf, newdata=testData)
table(irisPred, testData$Sp)
# Margin of Predictions
plot(margin(rf, testData$Species))
# The margin of a data point is as the proportion of votes
    for the correct class minus maximum proportion of votes
    for other classes.
# Generally speaking, positive margin means correct
    classification.
```

Chapter 5: K-Nearest Neighbors (KNN) Algorithm (and Quiz #1)

5 Chapter 5: K-Nearest Neighbors (KNN) Algorithm (and Quiz #1)

5.1 K-Nearest Neighbor (KNN) Algorithm

5.2 How does KNN work?

5.3 Distance Metrics Used in KNN Algorithm

5.4 Example

5.5 Advantages and Disadvantages of KNN Algorithm

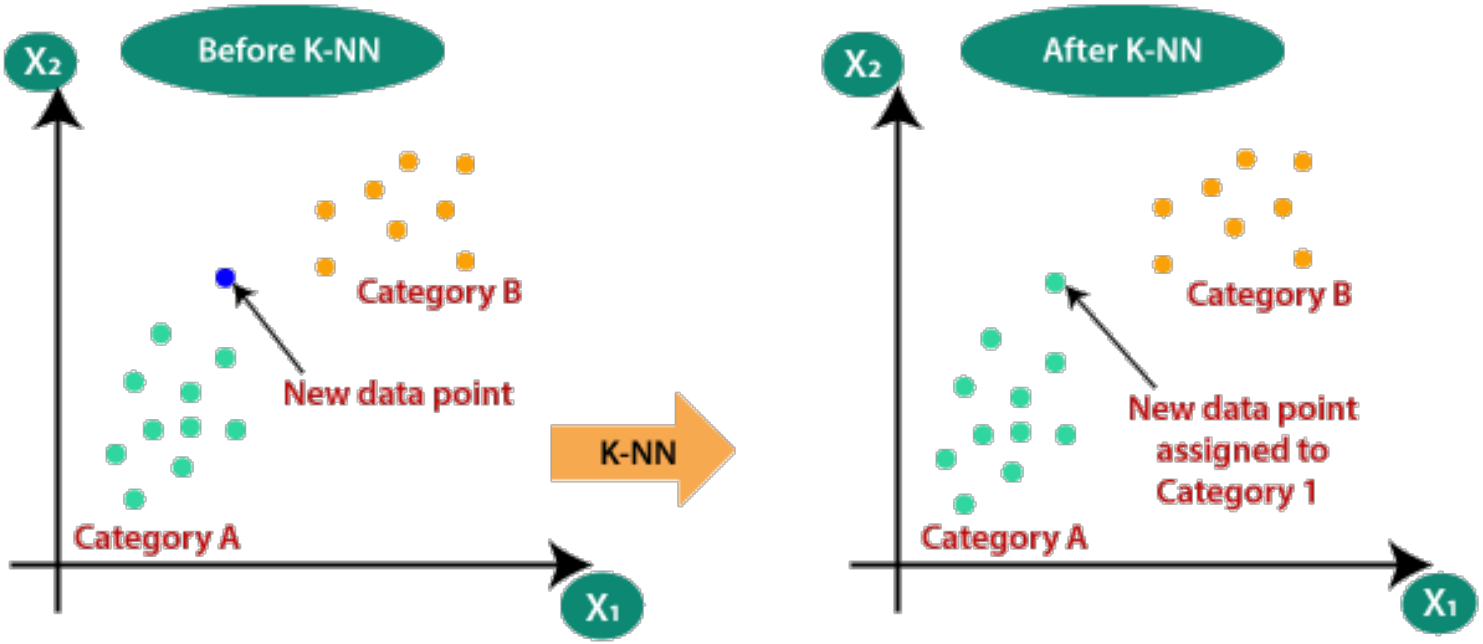
5.6 Python Code: KNN algorithm

K-Nearest Neighbor (KNN) Algorithm

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique
- while it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data
- it is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset

Why do we need a K-NN Algorithm?

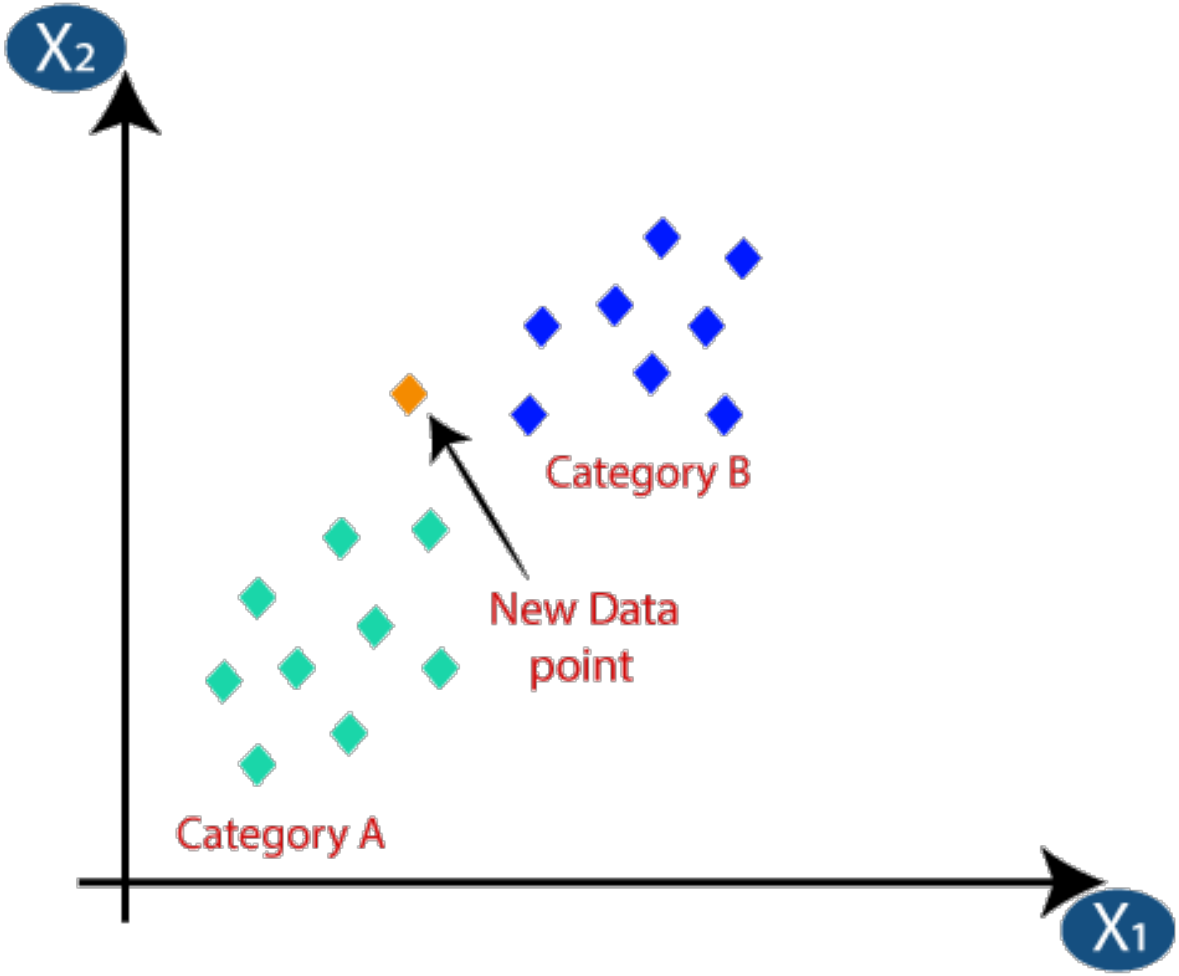
- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x_1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



How does KNN work?

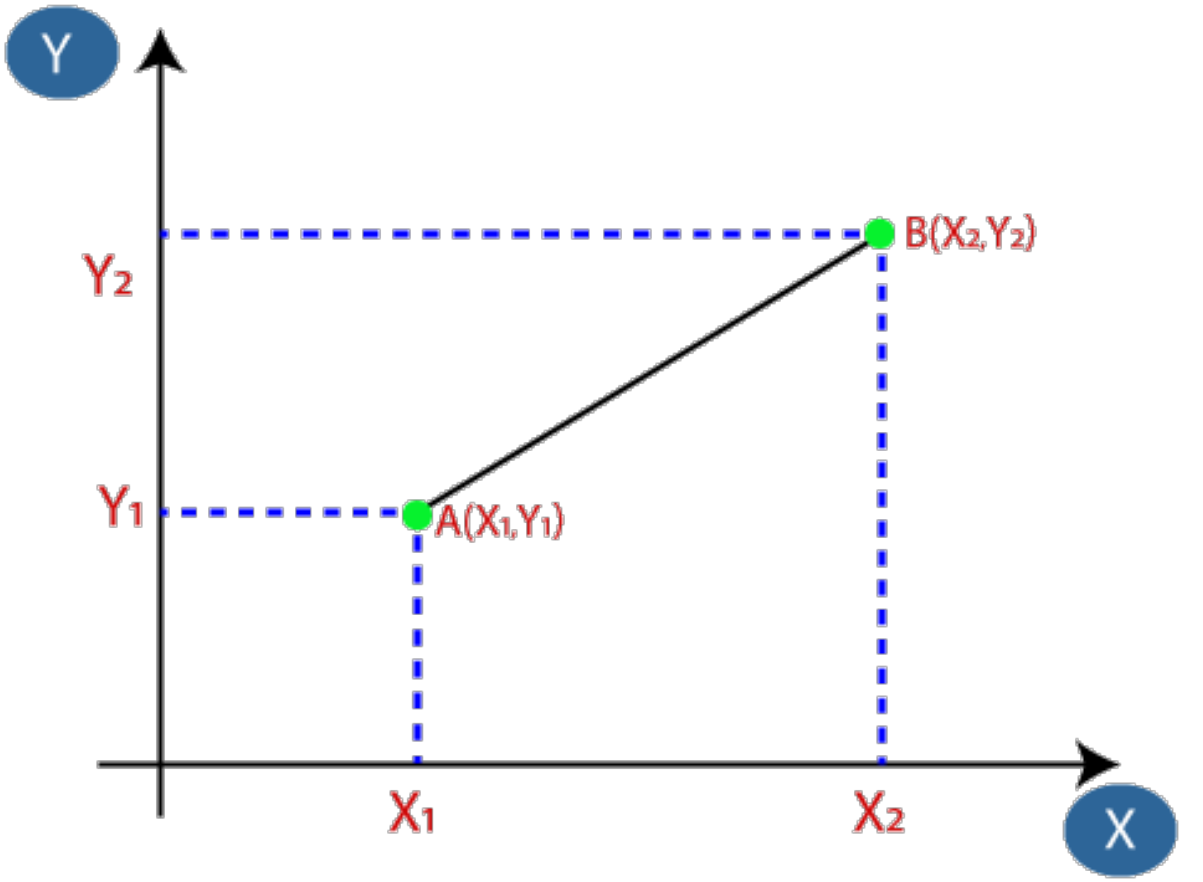
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of K number of neighbors
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance (or, you can use any distance measure.).
- **Step-4:** Among these K neighbors, count the number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

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



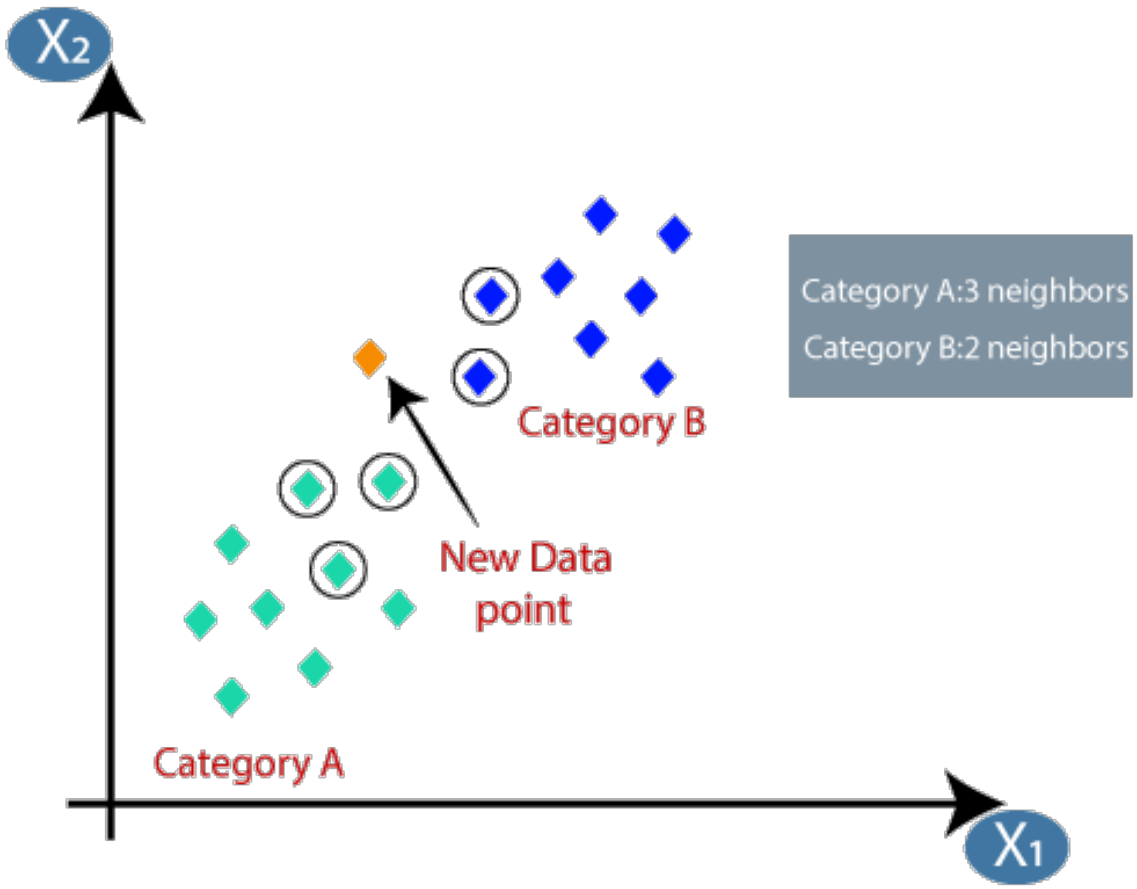
- Firstly, we will choose the number of neighbors, so we will choose the $K = 5$.

- Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



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

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



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

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

- Euclidean Distance
- Mahalanobis Distance
- Manhattan Distance
- Minkowski Distance
- ...

Distance Measures

alternatively, distance measure measure how unlike or dissimilar objects $t_i = \langle t_{i1}, \ldots, t_{ip} \rangle$ and $t_j = \langle t_{j1}, \ldots, t_{jp} \rangle$ are

1. Euclidean (straight-line) distance:

$$dis(t_i, t_j) = \sqrt{\sum_{h=1}^{p} (t_{ih} - t_{jh})^2} = \sqrt{(\mathbf{t}_i - \mathbf{t}_j)^T (\mathbf{t}_i - \mathbf{t}_j)}$$

   where $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{ip}]^T$ and $\mathbf{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jp}]^T$

2. Mahalanobis or Statistical distance:

$$dis(t_i, t_j) = \sqrt{(\mathbf{t}_i - \mathbf{t}_j)^T \mathbf{S}^{-1} (\mathbf{t}_i - \mathbf{t}_j)}$$

   where $\mathbf{S}$ contains the sample variances and covariances

3. Manhattan or city-block distance:

$$dis(t_i, t_j) = \sum_{h=1}^{p} |t_{ih} - t_{jh}|$$

Distance Measures

the dissimilar objects $t_i = \langle t_{i1}, \ldots, t_{ip} \rangle$ and $t_j = \langle t_{j1}, \ldots, t_{jp} \rangle$ are

4. Minkowski metric:

$$dis(t_i, t_j) = \left[\sum_{h=1}^{p} |t_{ih} - t_{jh}|^m \right]^{\frac{1}{m}}$$

   - for $m = 1$, $dis(t_i, t_j)$ measures the "Manhattan or City-block" distance between two points in p dimensions
   - for $m = 2$, $dis(t_i, t_j)$ becomes the Euclidean distance+

5. Canberra metric:

$$dis(t_i, t_j) = \sum_{i=1}^{p} \frac{|t_{ih} - t_{jh}|}{(t_{ih} + t_{jh})}$$

6. Czekanowski coefficient:

$$dis(t_i, t_j) = 1 - \frac{2 \sum_{i=1}^{p} \min(t_{ih}, t_{jh})}{\sum_{i=1}^{p} (t_{ih} + t_{jh})}$$

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

Sepal Length	Sepal Width	Species
5.2	3.1	?

Distance formula:

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

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

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

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

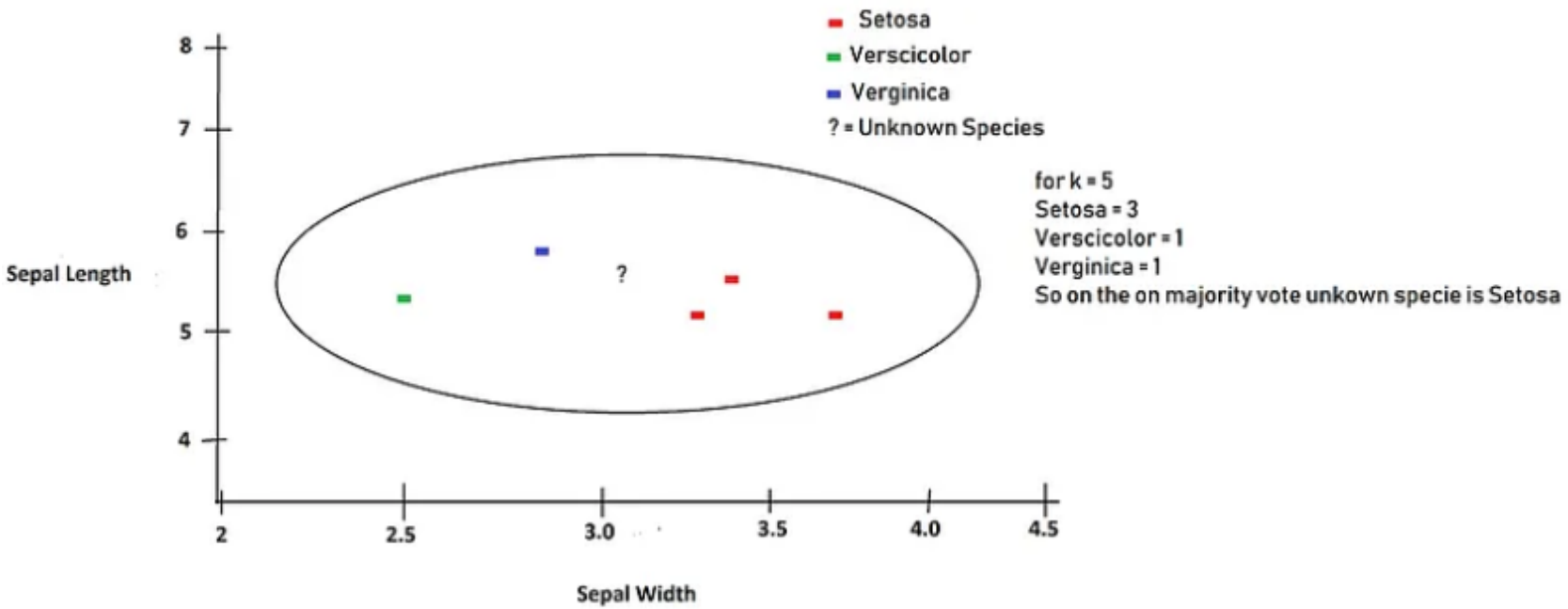
Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608
5.1	3.8	Setosa	0.707
7.2	3.0	Virginica	2.002
5.4	3.4	Setosa	0.36
5.1	3.3	Setosa	0.22
5.4	3.9	Setosa	0.82
7.4	2.8	Virginica	2.22
6.1	2.8	Verscicolor	0.94
7.3	2.9	Virginica	2.1
6.0	2.7	Verscicolor	0.89
5.8	2.8	Virginica	0.67
6.3	2.3	Verscicolor	1.36
5.1	2.5	Verscicolor	0.60
6.3	2.5	Verscicolor	1.25
5.5	2.4	Verscicolor	0.75

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

Sepal Length	Sepal Width	Species	Distance	Rank
5.1	3.3	Setosa	0.22	1
5.4	3.4	Setosa	0.36	2
5.1	3.7	Setosa	0.608	3
5.1	2.5	Verscicolor	0.6	4
5.8	2.8	Virginica	0.67	5

Nearest Neighbors for 5

Sepal Length	Sepal Width	Species
5.2	3.1	?



Feature Species is Setosa because a majority vote for setosa=3 and virginica=1 and virginica =1 so on the basis of highest vote KNN for K=5 is Setosa.

Sepal Length	Sepal Width	Species
5.2	3.1	Setosa

So in this way k-Nearest Neighbors algorithm work.

How do you find the optimal value of K?

- There is no particular way to determine the best value for K, so we need to try some values to find the best out of them. The most preferred value for K is 5.
- Large values for K are good, but it may find some difficulties
- A very low value for K such as $K = 1$ or $K = 2$, can be noisy and lead to the effects of outliers in the model
- the rule-of-thumb, K should be the square root of n (number of data points in the training dataset
- K should be chosen as the odd so that there are no ties
- If the square root is even, then add or subtract 1 to it.
- alternatively,
  1. Grid Search with cross-validation (details are given in the Python Code)
  2. Elbow Method
  3. ...

1. Grid Search with Cross-Validation:
   - Perform a grid search over a range of k values (e.g., $k = 1, 3, 5, 7, 9$) and use cross-validation to assess the model's performance for each k. Choose the k that results in the best cross-validated performance (e.g., using k-fold cross-validation).
2. Elbow Method:
   - Plot the accuracy or another relevant performance metric of the KNN model against different k values. The point at which the performance starts to level off (resembling an "elbow" in the plot) may be a good choice for k.

Advantages of KNN Algorithm

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

Disadvantages of KNN Algorithm

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

Python Code: KNN algorithm

```
import numpy as np
import pandas as pd
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score

# Data Loading
url='https://gist.githubusercontent.com/curran/a08a1080b88344b0c
df = pd.read_csv(url)
df.head()
```

Python Code: KNN algorithm

```
plt.figure()
sns.pairplot(df, hue = "species", markers=["o", "s", "D"])
plt.show()
```

Python Code: KNN algorithm

```
x = df[['sepal_length', 'sepal_width', 'petal_length',
    'petal_width']]
y = df['species']
x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size=0.25, random_state =3)

#Fitting K-NN classifier to the training set
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=5, metric='minkowski',
    p=2 )
knn.fit(x_train, y_train)

predictionsKNN = knn.predict(x_test)
print('Classification
    Report:\n',classification_report(y_test, predictionsKNN))
print('Confusion Matrix:\n',confusion_matrix(y_test,
    predictionsKNN))
print('Accuracy Score:',accuracy_score(y_test,
    predictionsKNN))
```

Python Code: KNN algorithm

```
# creating list of K for KNN
from sklearn.model_selection import cross_val_score
k_list = list(range(1,50,2))
# creating list of cv scores
cv_scores = []

# perform 10-fold cross validation
for k in k_list:
knn = KNeighborsClassifier(n_neighbors=k)
scores = cross_val_score(knn, x_train, y_train, cv=10,
    scoring='accuracy')
cv_scores.append(scores.mean())
```

Python Code: KNN algorithm

```
# changing to misclassification error
MSE = [1 - x for x in cv_scores]

plt.figure()
plt.figure(figsize=(15,10))
plt.title('The optimal number of neighbors', fontsize=20,
    fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=15)
plt.ylabel('Misclassification Error', fontsize=15)
sns.set_style("whitegrid")
plt.plot(k_list, MSE)

plt.show()
# finding best k
best_k = k_list[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d." % best_k)
```

Chapter 6: Evaluating a Machine Learning Model

6 Chapter 6: Evaluating a Machine Learning Model

6.1 Machine Learning Model Evaluation

6.2 Validation (or Holdout) Approach

6.3 ROC Curve and AUC Curve for classification

6.4 Cross Validation

6.5 Stratified k-fold cross-validation

6.6 Python code for k fold cross-validation.

Evaluating a Machine Learning Model

In this chapter we will examine strategies for evaluating the quality of models created through our learning algorithms. Models are only as useful as the quality of their predictions, and thus, fundamentally, our goal is not to create models (which is easy) but to create high-quality models (which is hard). Evaluating machine learning models is a critical step in the model development process to ensure that the model's performance meets the desired criteria.

- At first consideration, evaluating supervised-learning models might appear straightforward: train a model and then calculate how well it did using some performance metric (accuracy, squared errors, etc.). However, this approach is fundamentally flawed. If we train a model using our data,and then evaluate how well it did on that data, we are not achieving our desired goal.
- Our goal is not to evaluate how well the model does on our training data, but how well it does on data it has never seen before (e.g., a new customer, a new crime, a new image). For this reason, our method of evaluation should help us understand *how well models are able to make predictions* from data they have never seen before.

Validation (or holdout) Approach

- One strategy might be to hold off a slice of data for testing. This is called validation (or holdout). In validation, our observations (features and targets) are split into two sets, traditionally called the training set and the test set.
- There is no fixed rule while selecting the size of the training set and testing set. Its all about trial and error, so try out different ratios 80-20, 70-30, 65-35 and pick one that gives the best performance result.
- For splitting the dataset, we can use the `train_test_split` function of `scikit-learn`. The bellow line of code can be used to split dataset:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,
    y, test_size= 0.2, random_state=1)
```

Machine Learning Model Evaluation

Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. The evaluation also helps to analyze a model's key weaknesses. Some important indicators are:

1. Classification Problems:
   - Accuracy
   - Precision
   - Recall
   - F_1-Score
   - Receiver Operating Characteristic (ROC) curve
   - Area Under Curve (AUC)
   - ...
2. Regression Problems:
   - Mean Absolute Error (MAE): The average of the absolute errors.
   - Mean Squared Error (MSE): The average of the squared errors.
   - Root Mean Squared Error (RMSE): The square root of the MSE.
   - R-squared (R^2): Measures the proportion of variance in the dependent variable that's predictable.
   - ...

3. Clustering Problems:
   - Silhouette Score: Measures how similar an object is to its own cluster compared to other clusters.
   - Davies-Bouldin Index: Measures the average similarity ratio of each cluster with the cluster that is most similar to it.

Accuracy is a measure for the closeness of the measurements to a specific value, while precision is the closeness of the measurements to each other, i.e. not necessarily to a specific value.

Confusion Matrix and Classification Report

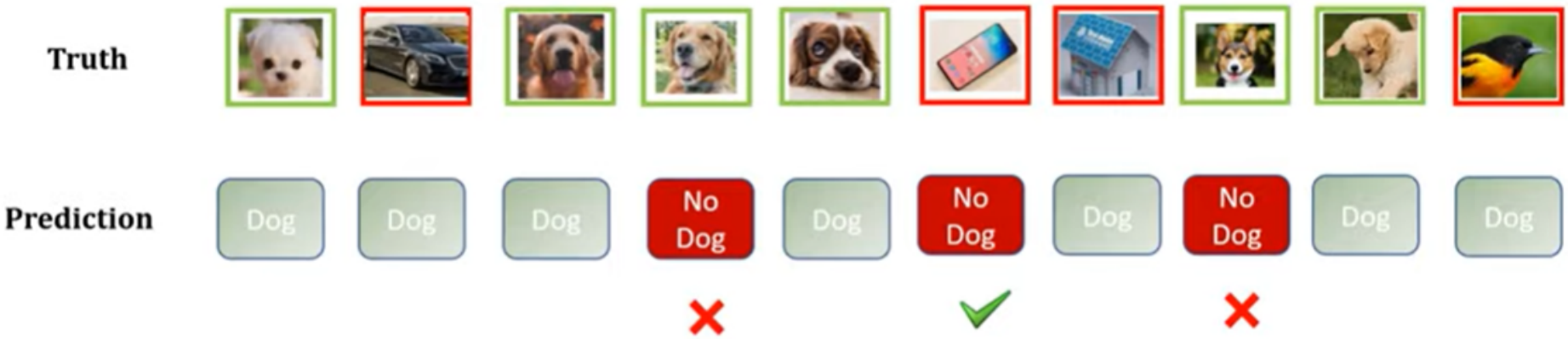
		Actual Class		
		Positive	**Negative**	
Predicted Class	**Positive**	Correct Classification True Positive (TP)	False Positive (FP) (Type II error)	**Precision** $= \frac{TP}{(TP+FP)}$
	Negative	False Negative (FN) (Type I error)	Correct Classification True Negative (TN)	
		Sensitivity (recall) $= \frac{TP}{(TP+FN)}$	**Specificity** $= \frac{TN}{(TN+FP)}$	

True Negative = 1 **False Negative = 2**

Classification Report in Python

True Positive = 4

False Positive = 3



True Negative = 1 **False Negative = 2**

How many we got right? → 5

Accuracy → 5/10 → **0.5**

the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly

Recall is out of all dog truth how many you got it right?

Total Dog truth samples = 6

True Positive = 4

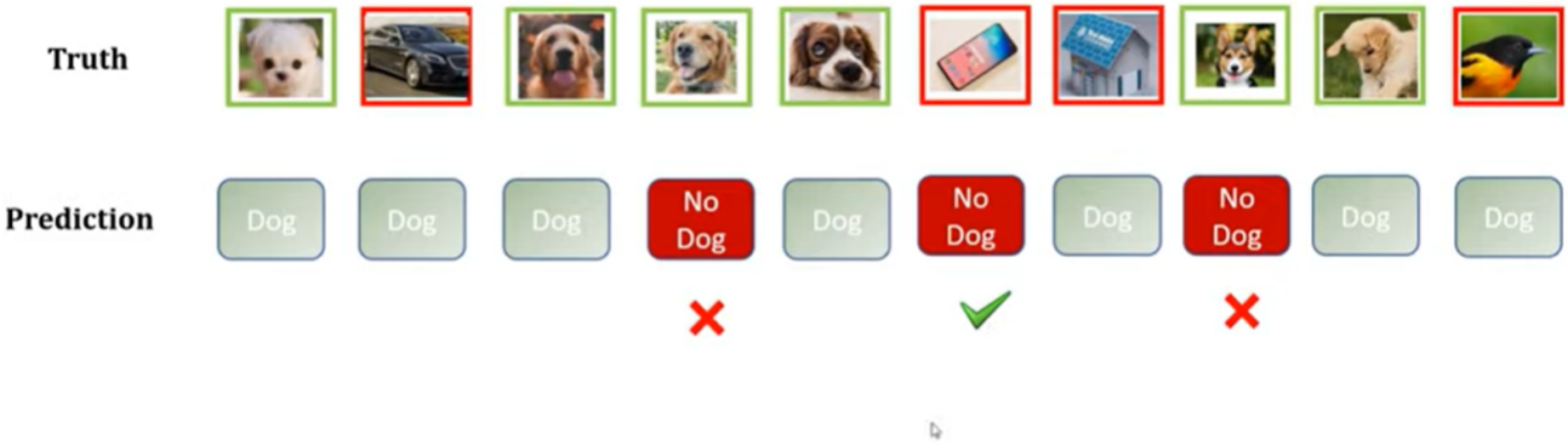
Recall = 4 / 6 = 0.67

Recall = TP / (TP + FN)

the recall is the number of true positive results divided by the number of all positive samples that should have been identified as positive

Confusion Matrix and Classification Report

		Actual Class		
		Positive	**Negative**	
Predicted Class	**Positive**	Correct Classification True Positive (TP) **(4)**	False Positive (FP) (Type II error) **(3)**	**Precision** $= \frac{TP}{(TP+FP)}$ $= \frac{4}{(4+3)}$
	Negative	False Negative (FN) (Type I error) **(2)**	Correct Classification True Negative (TN) **(1)**	
		Sensitivity (Recall) $= \frac{TP}{(TP+FN)}$ $= \frac{4}{(4+2)}$	**Specificity** $= \frac{TN}{(TN+FP)}$ $= \frac{1}{(1+3)}$	



Precision = 1 / 3 = 0.33

Recall = 1 / 4 = 0.25

The traditional F-measure or balanced F-score (**F_1 score**) is the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

F_β [edit]

A more general F score, F_β, that uses a positive real factor β, where β is chosen such that recall is considered β times as important as precision, is:

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}.$$

In terms of Type I and type II errors this becomes:

$$F_\beta = \frac{(1 + \beta^2) \cdot \mathrm{true\ positive}}{(1 + \beta^2) \cdot \mathrm{true\ positive} + \beta^2 \cdot \mathrm{false\ negative} + \mathrm{false\ positive}}.$$

Two commonly used values for β are 2, which weighs recall higher than precision, and 0.5, which weighs recall lower than precision.

Accuracy:

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

The accuracy is not always an adequate performance measure. Let us assume we have 1000 samples. 995 of these are negative and 5 are positive cases. Let us further assume we have a classifier, which classifies whatever it will be presented as negative. The accuracy will be a surprising 99.5%, even though the classifier could not recognize any positive samples.

Recall aka. True Positive Rate:

(it is also called Sensitivity)

$$recall = \frac{TP}{FN + TP}$$ **= Sensitivity**

False Positive Rate

$$FPR = \frac{FP}{TN + FP}$$ **=1-Specificity**

Precision:

$$precision : \frac{TP}{FP + TP}$$

		predicted		
actual		dog	cat	snake
	dog	6	2	0
	cat	1	6	0
	snake	1	1	8

$$Precision_i = \frac{M_{ii}}{\sum_j M_{ji}}$$

$$Recall_i = \frac{M_{ii}}{\sum_j M_{ij}}$$

The precision for our animals can be calculated as

$$precision_{dogs} = 6/(6+1+1) = 3/4 = 0.75$$

$$precision_{cats} = 6/(2+6+1) = 6/9 = 0.67$$

$$precision_{snakes} = 8/(0+0+8) = 1$$

The recall is calculated like this:

$$recall_{dogs} = 6/(6+2+0) = 3/4 = 0.75$$

$$recall_{cats} = 6/(1+6+0) = 6/7 = 0.86$$

$$recall_{snakes} = 8/(1+1+8) = 4/5 = 0.8$$

ROC curve

- An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:
  - True Positive Rate (Sensitivity)
  - False Positive Rate (1-Specificity)
- ROC curves are used for selecting the most appropriate classification models based on their performance with respect to the false positive rate (FPR) and true positive rate (TPR).

True Positive Rate (TPR)

- True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$
$$= \text{Sensitivity}$$

- False Positive Rate (FPR) is defined as follows:

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$
$$= \text{1-Specificity}$$

ROC Curve (Receiver Operating Characteristic curve

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

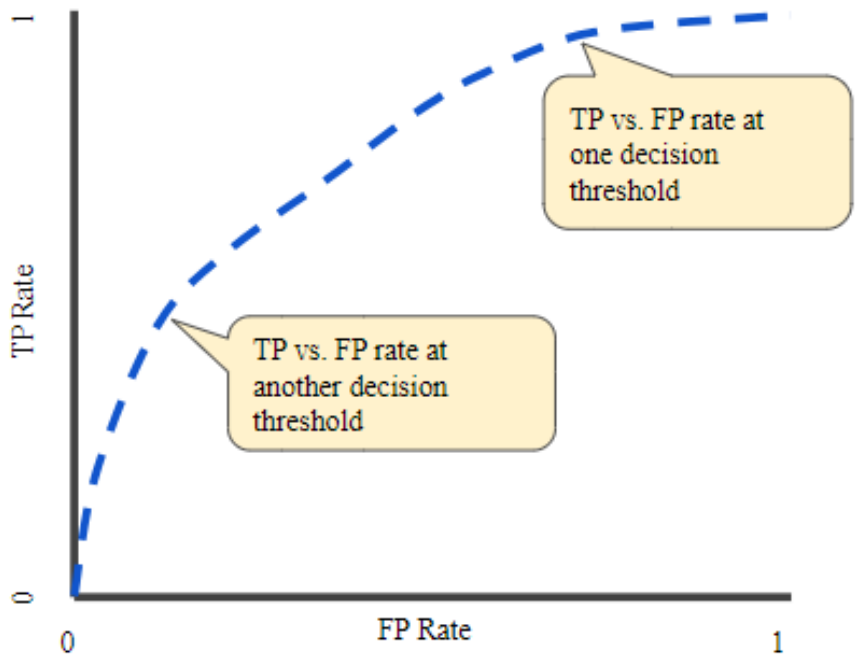
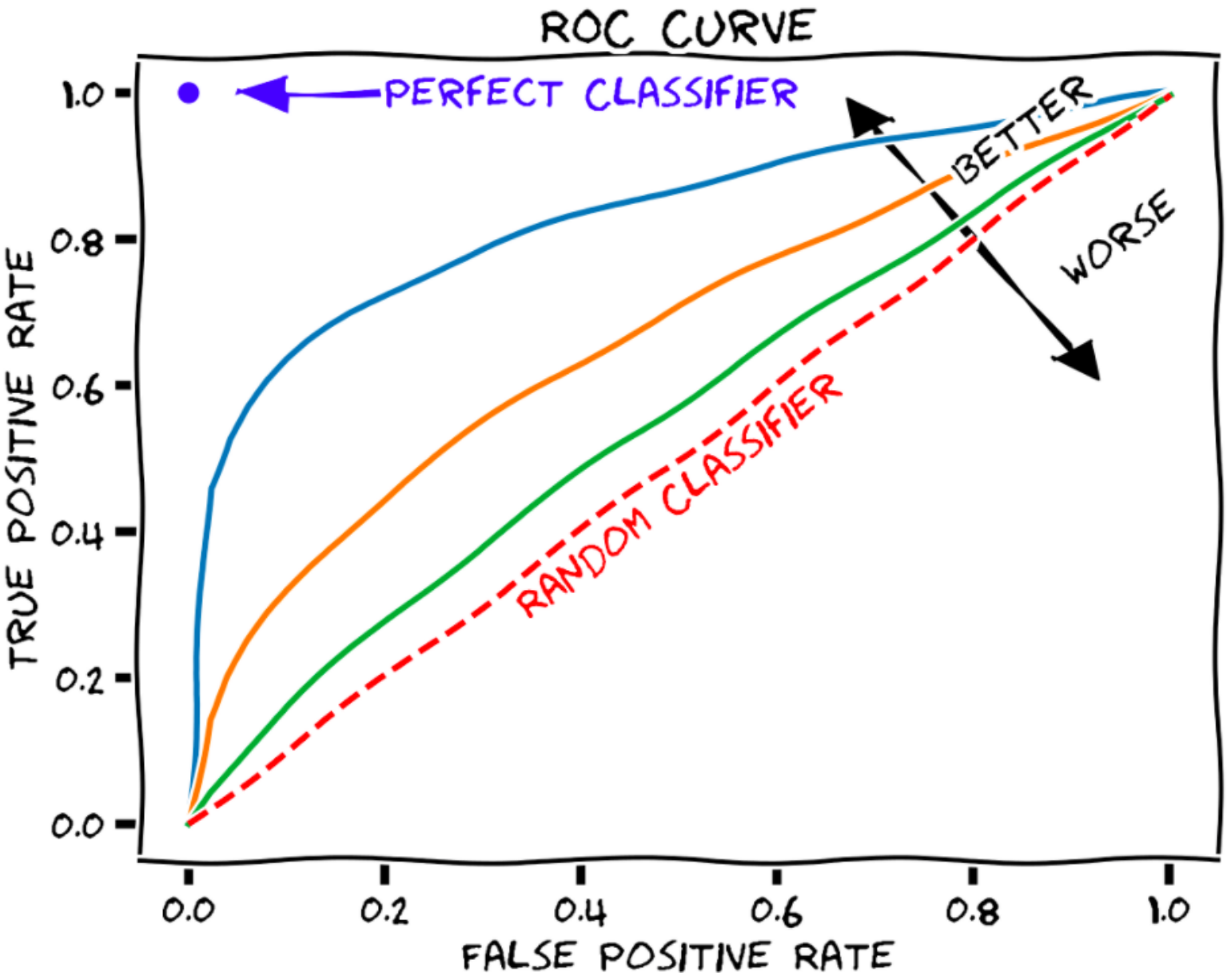


Figure:. TP vs. FP rate at different classification thresholds.


ROC CURVE
PERFECT CLASSIFIER
BETTER
WORSE
RANDOM CLASSIFIER
TRUE POSITIVE RATE
FALSE POSITIVE RATE
0.0
0.2
0.4
0.6
0.8
1.0

- higher value of TPR would mean that the value of false negative is very low which would mean almost all positives are predicted correctly.
- lower value of FPR would mean that the value of false positive is very low which means almost all negatives are predicted correctly.
- going by the above, the decision threshold near top left of ROC curve would result in the model having optimal performance. In the above diagram, the point is represented using "Perfect Classifier"

Drawback of Validation (Training/Test) Approach

the validation approach has two major weaknesses.

- First, the performance of the model can be highly dependent on which few observations were selected for the test set.
- Second, the model is not being trained using all the available data, and it's not being evaluated on all the available data.

Cross Validation

Cross Validation is a statistical method of evaluating generalization performance that is more stable and thorough than using a split into a training and a test set. In cross validation, the data is *instead split repeatedly* and multiple models are trained. The common methods are

- Validation Set Approach
- Leave-P-out cross-validation
- Leave one out cross-validation (LOOCV)
- k-fold cross-validation
- Stratified k-fold cross-validation

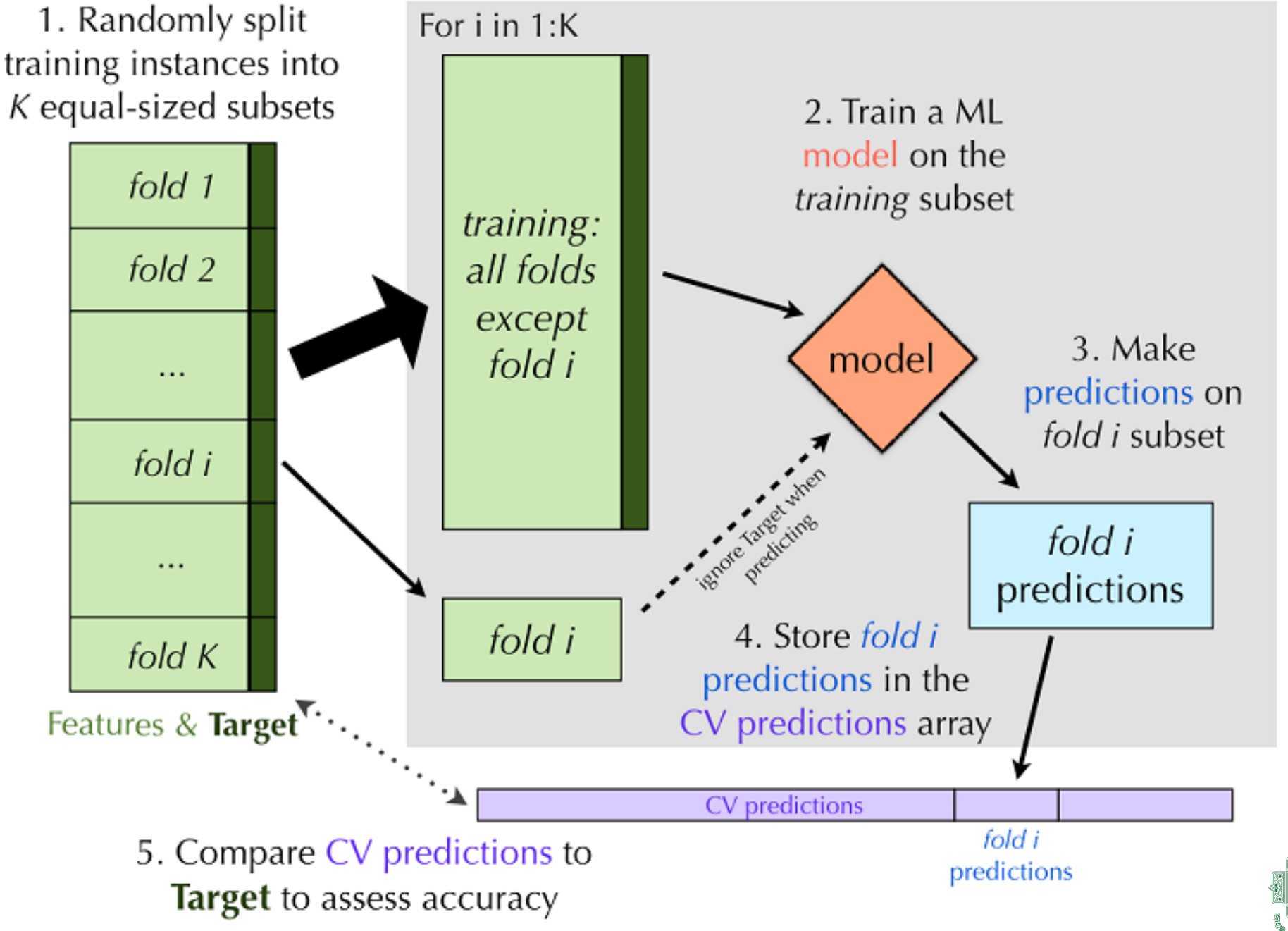
k-Fold Cross-Validation

the most commonly used version of cross-validation is k-fold cross-validation, where k is a user-specified number, usually 5 or 10.

- the steps for k-fold cross-validation are:
- for each group:
  - take one group as the reserve or test data set
  - use remaining groups as the training dataset
  - fit the model on the training set and evaluate the performance of the model using the test set
- the performance on the model for each of the k iterations is then averaged to produce an overall measurement
- it is always suggested that the value of k should be 10 as the lower value of k is takes towards validation and higher value of k leads to LOOCV method.

Example: 5-Fold Cross-Validation

Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.


1. Randomly split training instances into K equal-sized subsets
fold 1
fold 2
...
fold i
...
fold K
Features & Target
For i in 1:K
training: all folds except fold i
fold i
2. Train a ML model on the training subset
model
ignore Target when predicting
3. Make predictions on fold i subset
fold i predictions
4. Store fold i predictions in the CV predictions array
CV predictions
fold i predictions
5. Compare CV predictions to Target to assess accuracy

Stratified k-fold cross-validation

Splitting the dataset into k-folds by starting with the first one-k-th part of the data, as described in the previous section, might not always be a good idea. For example, let'shave a look at the iris dataset:

In[7]:

```
from sklearn.datasets import load_iris
iris = load_iris()
print("Iris labels:\n{}".format(iris.target))
```

Out[7]:

```
Iris labels:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

As you can see, the first third of the data is the class 0, the second third is the class 1, and the last third is the class 2. Imagine doing three-fold cross-validation on this dataset. The first fold would be only class 0, so in the first split of the data, the test set would be only class 0, and the training set would be only classes 1 and 2. As the classes in training and test sets would be different for all three splits, the three-fold cross-validation accuracy would be zero on this dataset. That is not very helpful, as we can do much better than 0% accuracy on `iris`.

Stratified k-fold cross-validation

- In stratified cross-validation, we split the data such that the proportions between classes are the same in each fold as they are in the whole dataset.
- For example, if 90% of your samples belong to class A and 10% of your samples belong to class B, then stratified cross-validation ensures that in each fold, 90% of samples belong to class A and 10% of samples belong to class B.
- It is usually a good idea to use stratified k-fold cross-validation instead of k-fold cross-validation to evaluate a classifier, because it results in more reliable estimates of generalization performance.

Comparison of Cross-validation to train/test split in Machine Learning

- Train/test split: The input data is divided into two parts, that are training set and test set on a ratio of **70:30**, **75:25**, **80:20**, etc. It provides a high variance, which is one of the biggest disadvantages.
  - **Training Data:** The training data is used to train the model, and the dependent variable is known.
  - **Test Data:** The test data is used to make the predictions from the model that is already trained on the training data. This has the same features as training data but not the part of that.
- Cross-Validation dataset:
  - It is used to overcome the disadvantage of train/test split by splitting the dataset into groups of train/test splits, and averaging the result. It can be used if we want to optimize our model that has been trained on the training dataset for the best performance. It is more efficient as compared to train/test split as every observation is used for the training and testing both.

Advantages of Cross Validation

- Overcoming Overfitting: Cross validation helps to prevent overfitting by providing a more robust estimate of the model's performance on unseen data.
- Model Selection: Cross validation can be used to compare different models and select the one that performs the best on average.
- Hyperparameter tuning: Cross validation can be used to optimize the hyperparameters of a model, such as the regularization parameter, by selecting the values that result in the best performance on the validation set.
- Data Efficient: Cross validation allows the use of all the available data for both training and validation, making it a more data-efficient method compared to traditional validation techniques.

Disadvantages of Cross Validation:

There are some limitations of the cross-validation technique, which are given below:

- For the ideal conditions, it provides the optimum output. But for the inconsistent data, it may produce a drastic result. So, it is one of the big disadvantages of cross-validation, as there is no certainty of the type of data in machine learning.
- In predictive modeling, the data evolves over a period, due to which, it may face the differences between the training set and validation sets. Such as if we create a model for the prediction of stock market values, and the data is trained on the previous 5 years stock values, but the realistic future values for the next 5 years may drastically different, so it is difficult to expect the correct output for such situations.
- The choice of the number of folds in cross validation can impact the bias-variance tradeoff, i.e., too few folds may result in high variance, while too many folds may result in high bias.
- Computationally Expensive and Time-Consuming

Applications of Cross-Validation

- This technique can be used to compare the performance of different predictive modeling methods.
- It has great scope in the medical research field.
- It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.

Python code for k fold cross-validation.

```
# importing cross-validation from sklearn package.
from sklearn import cross_validation

# value of K is 10.
data = cross_validation.KFold(len(train_set), n_folds=10,
    indices=False)
```

k fold cross-validation for Logistic Classification

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
iris = load_iris()
logreg = LogisticRegression()
scores = cross_val_score(logreg, iris.data, iris.target)
print("Cross-validation scores: {}".format(scores))
```

```
scores = cross_val_score(logreg, iris.data, iris.target,
    cv=5)
print("Cross-validation scores: {}".format(scores))
```

A common way to summarize the cross-validation accuracy is to compute the mean:

```
print("Average cross-validation score:
    {:.2f}".format(scores.mean()))
```

Chapter 7: Regression Analysis for Prediction

7 Chapter 7: Regression Analysis for Prediction

7.1 Problems & Motivation

7.2 Problem & Motivation

7.3 Scatter Diagram

7.4 Correlation Coefficient

7.5 Regression Analysis

7.6 Concept of Regression Analysis

7.7 Types of parametric regression analysis

7.8 Simple Linear Regression Model

7.9 A Probabilistic View of Linear Regression

7.10 Distributional Assumptions Underlying Regression

7.11 Simple Linear Regression Model

Problems & Motivation

Let us explore the dataset. We will be using the **Boston House Prices Dataset**, with 506 rows and 13 attributes with a target column. Let's take a quick look at the dataset.

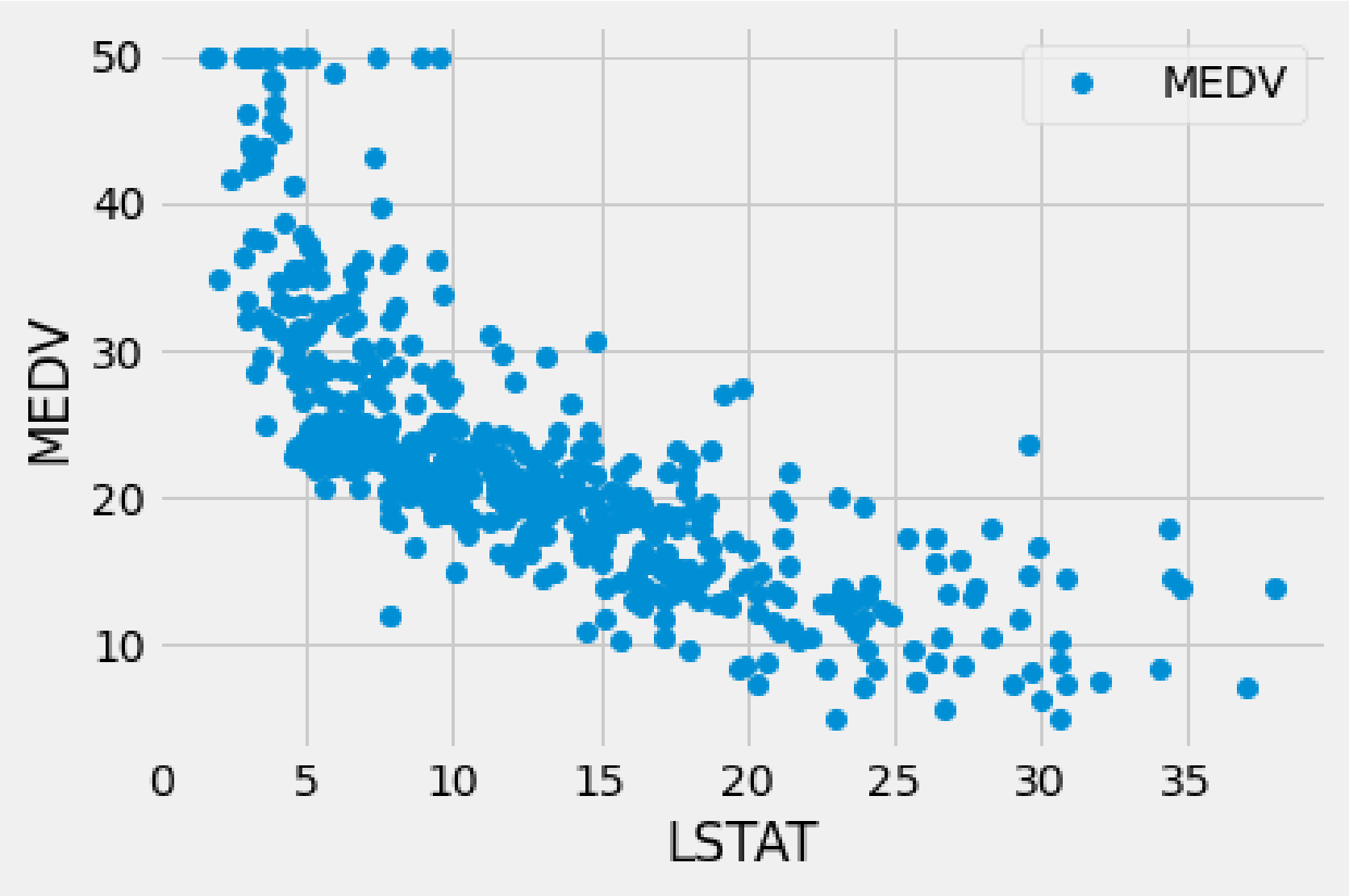
```
import io
boston = pd.read_csv(io.BytesIO(uploaded['boston.csv']))
# Dataset is now stored in a Pandas Dataframe
boston.head(10)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

- **Crim**: Per capita crime rate by town
- **Zn**: Proportion of residential land zoned for lots over 25,000 sq. ft.
- **Indus**: Proportion of non-retail business acres per town
- **Chas**: Charles River dummy variable (= 1 if tract bounds river; 0, otherwise)
- **Nox**: Nitrogen oxides concentration (parts per 10 million)
- **Rm**: Average number of rooms per dwelling
- **Age**: Proportion of owner-occupied units built before 1940
- **Dis**: Weighted mean of distances to five Boston employment centers
- **Rad**: Index of accessibility to radial highways
- **Tax**: Full-value property tax rate per \$10,000
- **Ptratio**: Pupil–Teacher ratio by town
- **B**: $1000(Bk - 0.63)^2$, where Bk is the proportion of Blacks by town
- **Lstat**: Lower status of the population (percent)
- **Medv**: Median Price value of owner-occupied homes in \$1000s

Problems & Motivation

- We want to predict the median price value of owner-occupied homes. How do you predict the price?


MEDV
50
40
30
20
10
0
5
10
15
20
25
30
35
LSTAT
MEDV

Python Code

```
from google.colab import files
uploaded = files.upload()
import io
boston = pd.read_csv(io.BytesIO(uploaded['boston.csv']))
# Dataset is now stored in a Pandas Dataframe
boston.head()
import matplotlib.pyplot as plt
boston.plot(x='LSTAT',y='MEDV',style='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.show()
```

```python
import statsmodels.api as sm
model =sm.OLS(boston.MEDV, sm.add_constant(boston.LSTAT))
result = model.fit()
print(result.summary())
```

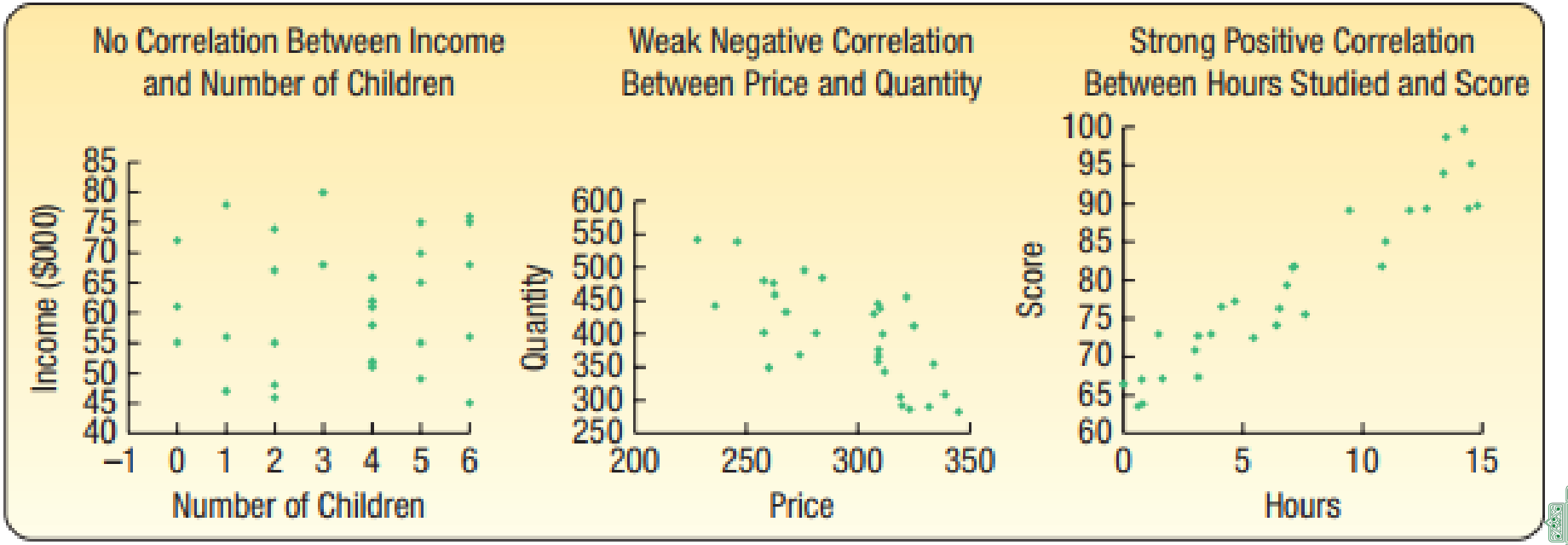
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.544
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     601.6
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           5.08e-88
Time:                        03:26:52   Log-Likelihood:                -1641.5
No. Observations:                 506   AIC:                             3287.
Df Residuals:                     504   BIC:                             3295.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.5538      0.563     61.415      0.000      33.448      35.659
LSTAT         -0.9500      0.039    -24.528      0.000      -1.026      -0.874
==============================================================================
Omnibus:                      137.043   Durbin-Watson:                   0.892
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              291.373
Skew:                           1.453   Prob(JB):                     5.36e-64
Kurtosis:                       5.319   Cond. No.                         29.7
==============================================================================
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.548
Model:                            OLS   Adj. R-squared:                  0.546
Method:                 Least Squares   F-statistic:                     304.4
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           2.33e-87
Time:                        03:37:58   Log-Likelihood:                -1639.6
No. Observations:                 506   AIC:                             3285.
Df Residuals:                     503   BIC:                             3298.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.3192      0.574     59.816      0.000      33.192      35.446
LSTAT         -0.9114      0.043    -21.004      0.000      -0.997      -0.826
CRIM          -0.0704      0.036     -1.956      0.051      -0.141       0.000
==============================================================================
Omnibus:                      146.159   Durbin-Watson:                   0.872
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              329.172
Skew:                           1.517   Prob(JB):                     3.32e-72
Kurtosis:                       5.531   Cond. No.                         32.7
==============================================================================
```

Scatter Diagram

a scatter diagram is a graphic tool used to portray the relationship between two variables

- the dependent variable is scaled on the Y-axis and is the variable being estimated
- the independent variable is scaled on the X-axis and is the variable used as the predictor



What is Correlation Analysis?

Correlation analysis

▸▸a group of techniques to measure the relationship between two variables

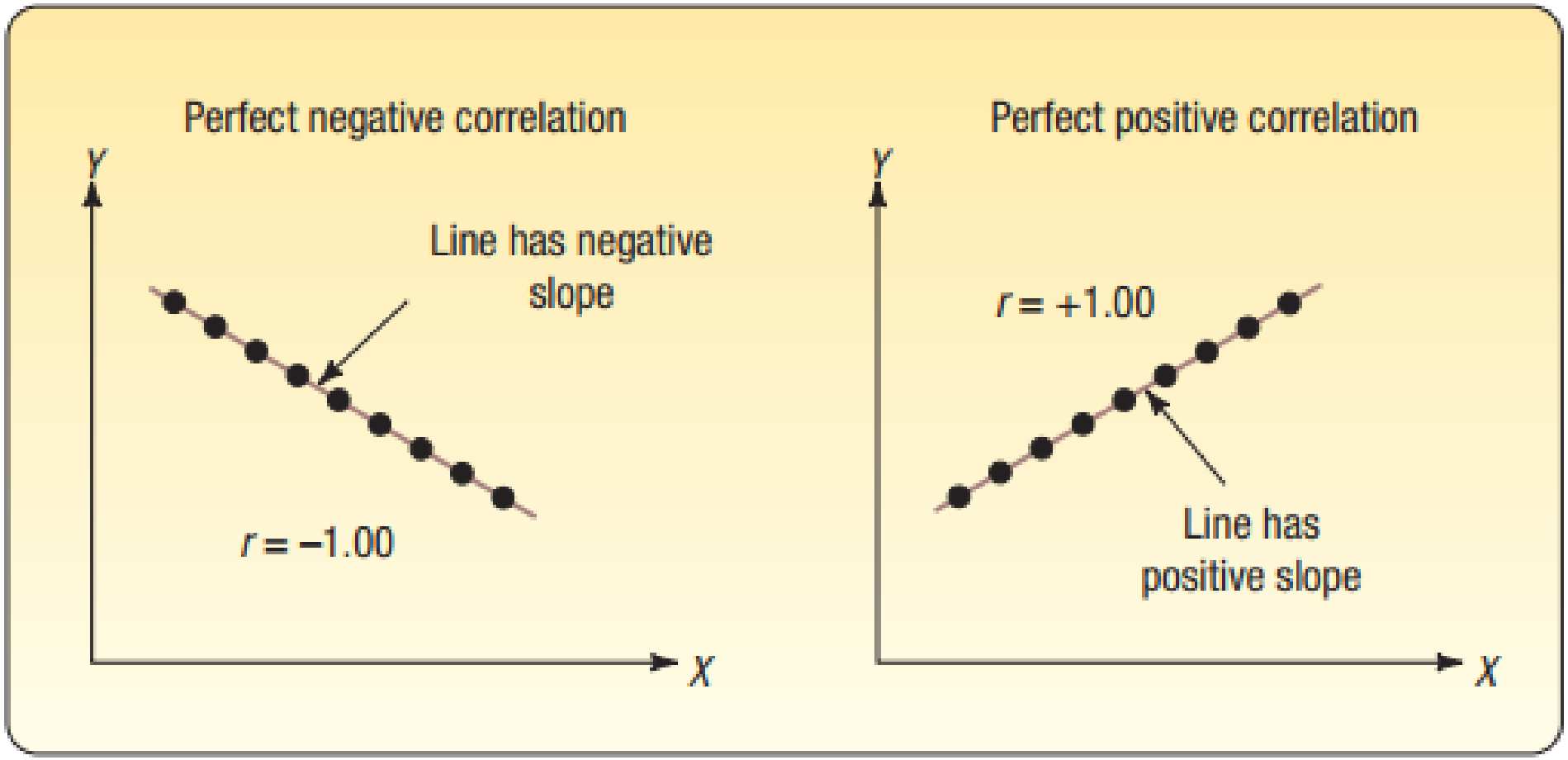
Pearson's Correlation coefficient

- is a measure of the strength of the linear relationship between two variables
- is denoted by r and defined as

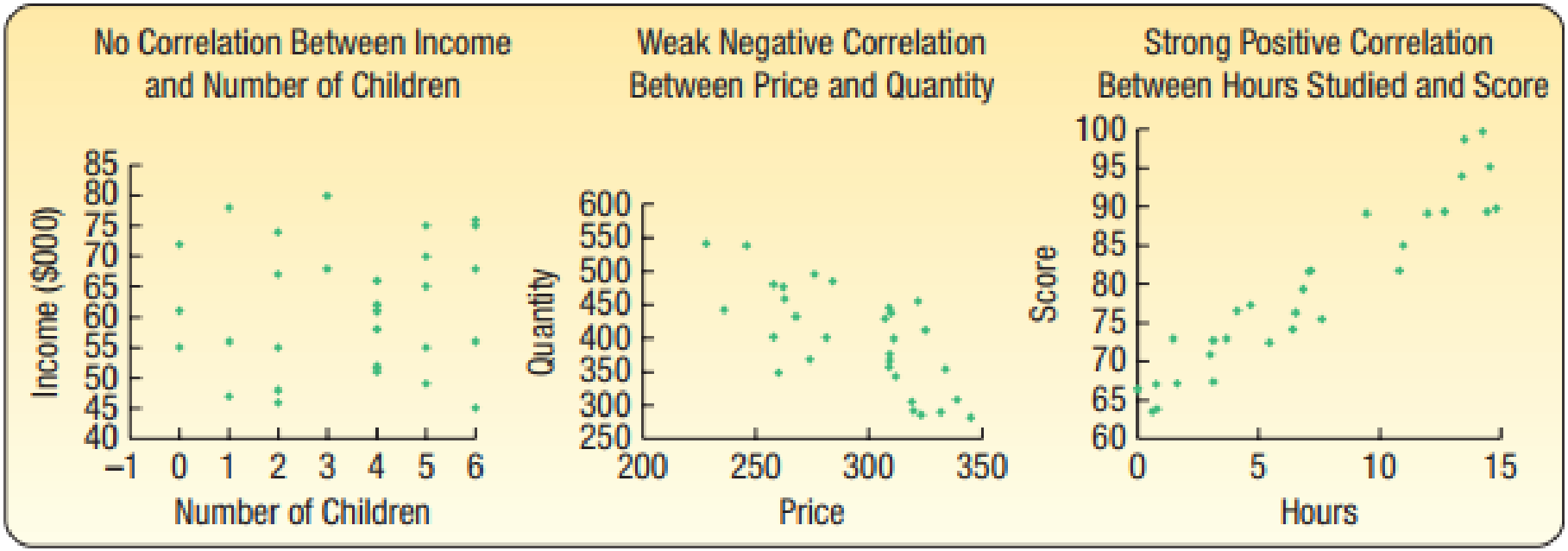
$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}\sqrt{\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right)}}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

Correlation Coefficient

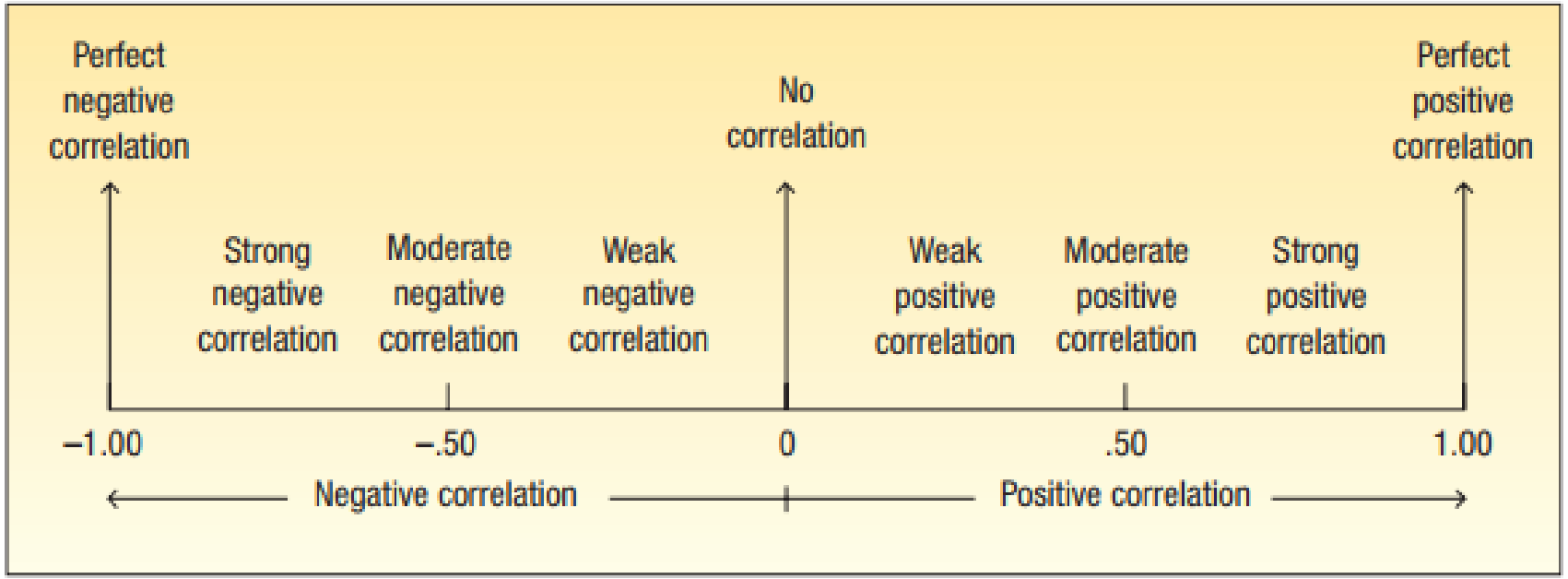


Correlation Coefficient



Correlation Coefficient

the following drawing summarizes the strength and direction of the correlation coefficient



Characteristics of correlation coefficient

1. the sample correlation coefficient is identified by the lowercase letter r
2. it shows the direction and strength of the linear relationship between two interval or ratio-scale variables
3. it ranges from -1 up to and including $+1$
4. a value near 0 indicates there is little linear relationship between the variables
5. a value near 1 indicates a direct or positive linear relationship between the variables
6. a value near -1 indicates an inverse or negative linear relationship between the variables

Regression Analysis

- correlation analysis *does not tell us why* and *how behind the relationship* but it just says the relationship exists
- in regression analysis, we find the relationship among variables and we also estimate or predict one variable based on another variable
- regression analysis *describes how a set of independent variables is numerically related to the dependent variable*
- the variable being predicted or estimated is called the is the dependent variable
- the variable or variables being used to predict the value of the dependent variable are called the independent variables
- both the independent and the dependent variables must be interval or ratio scale

Regression Analysis

- regression is to build a function of independent variables (also known as predictors) to predict a dependent variable (also called response) for example,

$$y = f(x_1, x_2, \ldots, x_p) + \epsilon$$

- for example, banks assess the risk of home-loan applicants based on their age, income, expenses, occupation, number of dependents, total credit limit, etc.

Types of parametric regression analysis

- for continuous response variable
  - Simple Linear Regression
  - Multiple Regression
  - Polynomial Regression
  - Multivariate Regression
- for categorical response variable
  - Logistic Regression
    - binomial (also called ordinary) logistic regression
    - multinomial logistic regression
    - ordinal logistic regression
    - alternating logistic regressions
- for discrete response variable
  - Poisson Regression
- for survival-time (time-to-event) outcomes
  - Cox Regression (or proportional hazards regression)

Regression Models

another types of regression analysis:

- Ridge Regression
- Lasso Regression
- Bayesian regression
- Nonparametric regression

other types of regression model in machine learning

- Generalized linear model (GLM)
- Additive model
- Generalized additive model (GAM)
- Generalized additive model for location, scale and shape (GAMLSS)
- Random effect model
- Linear mixed model (LMM)
- Generalized linear mixed model (GLMM)
- Generalized estimating equations (GEE)

A Probabilistic View of Regression Analysis

- regression analysis: to find out the relationship between a response variable Y and a set of covariates $\mathbf{X}$.
- response variable: Y and a set of covariates: $\mathbf{X} = (X_1, X_2, \ldots, X_p)^T$
- the conditional density:

$$\underbrace{f_{Y|\mathbf{X}}(y|\mathbf{x})}_{\text{parametric}} \leftarrow \underbrace{\left\{ \begin{array}{l} \mu(\mathbf{x}) \\ \sigma^2(\mathbf{x}) \end{array} \right.}_{\text{parametric/nonparametric}}$$

- conditional Mean of Y given $\mathbf{X} = \mathbf{x}$:

$$\mathbb{E}(Y|\mathbf{X} = \mathbf{x}) = \mu(\mathbf{X})$$

aims:

  - estimation of unknown function $\mathbb{E}(Y|\mathbf{X}) = \mu(\mathbf{X})$

Distributional Assumptions Underlying Classical Regression

- if
  - $f_{Y|\mathbf{X}}(y|\mathbf{x})$ is a normal density (Remarks: Y must be a normal random variate)
  - $\mu(\mathbf{X})$ is a linear function (i.e., $\mu(X_i) = \beta_0 + \beta_1 X_i$) and
  - $\sigma^2(\mathbf{x})$ is a constant i.e., $\sigma^2(\mathbf{x}) = \sigma^2$ (say)

  then we use a *classical linear regression model* to estimate $\mathbb{E}(Y|\mathbf{X})$
- in this case, we assume

$$\mathbb{E}(Y|\mathbf{X} = \mathbf{x}) = \beta_0 + \beta_1 X_i + \mathbb{E}(\epsilon|\mathbf{X})$$

  where β measures the marginal change in the mean of Y due to a marginal change in $\mathbf{x}$.
- the *classical linear regression model* is

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad ; \quad i = 1, 2, \ldots, n \tag{1}$$

  assumption: $\Rightarrow \epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$, that is $Y_i \overset{iid}{\sim} N(\beta_0 + \beta_1 X_i, \sigma^2)$
- therefore, it is also called normal regression

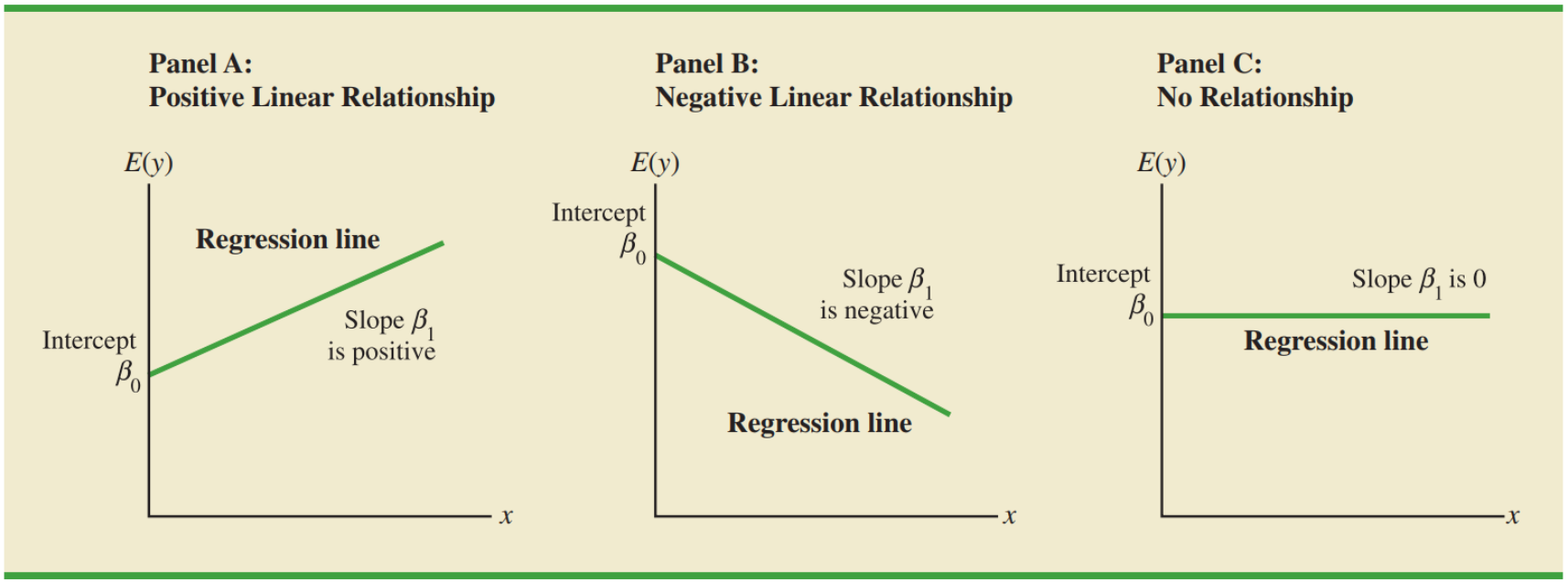
Simple Linear Regression Model

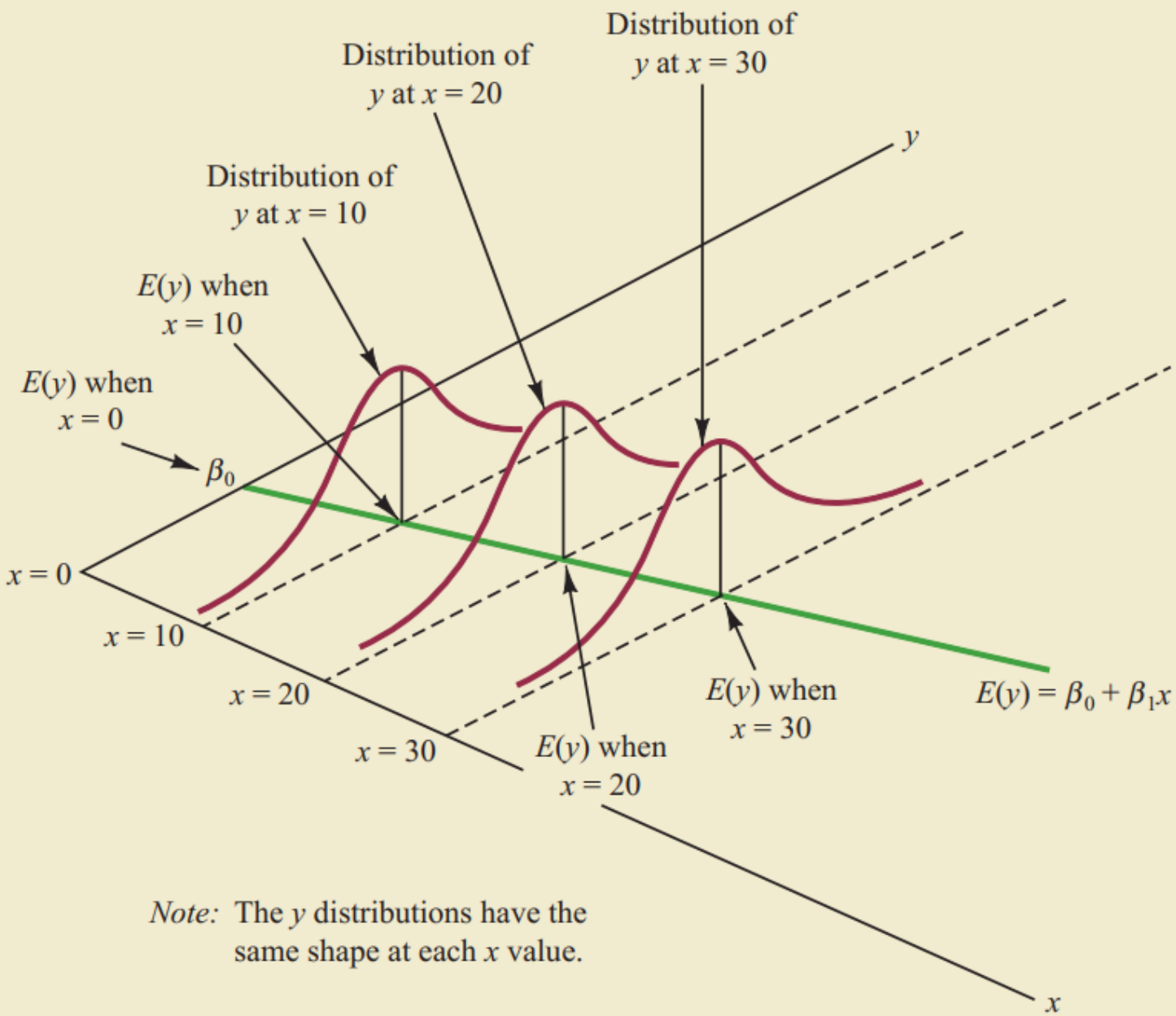
- suppose the linear mean regression model is

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad ; \quad i = 1, 2, \ldots, n \tag{2}$$

where β_0 is the constant or intercept and β_1 is the slope coefficient

FIGURE 14.1 POSSIBLE REGRESSION LINES IN SIMPLE LINEAR REGRESSION





Note: The y distributions have the same shape at each x value.

- the β_0 and β_1 are also called regression coefficients
- usually, β_0 and β_1 are unknown
- we can estimate β_0 and β_1 by following methods
  - ordinary least square (OLS) method
  - maximum likelihood estimation
  - method-of-moments (or GMM)
  - ...
- the estimated mean regression model (or equation) is

$$\widehat{Y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_i \tag{3}$$

where $\widehat{\beta}_0$ and $\widehat{\beta}_1$ are respectively, the estimator of β_0 and β_1

Least Square Estimation

1. minimizing function:

$$Q = \min_{\beta_0,\beta_0} \sum_{i=1}^{n} \epsilon_i^2 = \min_{\beta_0,\beta_1} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$
$$= \min_{\beta_0,\beta_1} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$

2. the estimators for β_0 and β_1 can be found by solving the following normalized equation

$$\frac{\partial Q}{\partial \beta_0} = \frac{\partial}{\partial \beta_0} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 = 0 \tag{4}$$

and

$$\frac{\partial Q}{\partial \beta_1} = \frac{\partial}{\partial \beta_1} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 = 0 \tag{5}$$

OLS estimatior

from (4), we have

$$\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)(-1) = 0$$

$$\Rightarrow \sum_{i=1}^{n} Y_i - n\beta_0 - \beta_1 \sum_{i=1}^{n} X_i = 0$$

or

$$\beta_0 = \bar{Y} - \beta_1 \bar{X}$$

from (5), we have

$$\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)(-X_i) = 0$$

$$\Rightarrow \sum_{i=1}^{n} X_i Y_i - \beta_0 \sum_{i=1}^{n} X_i - \beta_1 \sum_{i=1}^{n} X_i^2 = 0 \qquad (6)$$

putting the value of β_0 in Equation (6), we have

$$\sum_{i=1}^{n} X_i Y_i - \left(\bar{Y} - \beta_1 \bar{X}\right) \sum_{i=1}^{n} X_i - \beta_1 \sum_{i=1}^{n} X_i^2 = 0$$

$$\Rightarrow \sum_{i=1}^{n} X_i Y_i - \bar{Y} \sum_{i=1}^{n} X_i + \beta_1 \bar{X} \sum_{i=1}^{n} X_i - \beta_1 \sum_{i=1}^{n} X_i^2$$

$$\Rightarrow \sum_{i=1}^{n} X_i Y_i - n\bar{Y}\bar{X} + n\beta_1 \bar{X}^2 - \beta_1 \sum_{i=1}^{n} X_i^2 = 0$$

$$\Rightarrow \sum_{i=1}^{n} X_i Y_i - n\bar{Y}\bar{X} - \beta_1 \left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2 \right) = 0$$

$$\Rightarrow \beta_1 = \frac{\sum_{i=1}^{n} X_i Y_i - n\bar{X}\bar{Y}}{\sum_{i=1}^{n} X_i^2 - n\bar{X}^2}$$

- hence, the least square estimator of β_0 and β_1 are

$$\widehat{\beta}_0 = \bar{Y} - \widehat{\beta}_1 \bar{X}$$

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} X_i Y_i - n\bar{X}\bar{Y}}{\sum_{i=1}^{n} X_i^2 - n\bar{X}^2}$$

- the expression of $\widehat{\beta}_1$ can also be written as

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (X_i - \bar{X})^2} = r\left(\frac{s_y}{s_x}\right)$$

because

$$\sum_{i=1}^{n}(X_i - \bar{X})^2 = \sum_{i=1}^{n} X_i^2 - n\bar{X}^2 \quad \text{and} \quad \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) = \sum_{i=1}^{n} X_i Y_i - n\bar{X}\bar{Y}$$

Interpretation of Slope Coefficient

$\widehat{\beta}_1$ is the slope of the fitted (estimated) line

- it shows the amount of change in $\widehat{y}$ for a change of one unit in x
- a positive value for $\widehat{\beta}_1$ indicates a direct relationship between the two variables and a negative value indicates an inverse relationship
- the sign of $\widehat{\beta}_1$ and the sign of r, the correlation coefficient, are always the same

Parameter Estimates

The point estimates of the three unknown parameters in a simple linear regression model are

$$\hat{\beta}_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{S_{XY}}{S_{XX}}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\sigma}^2 = \frac{\text{SSE}}{n-2} = \frac{\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2}{n-2}$$

$$= \frac{\sum_{i=1}^{n} y_i^2 - \hat{\beta}_0 \sum_{i=1}^{n} y_i - \hat{\beta}_1 \sum_{i=1}^{n} x_i y_i}{n-2}$$

Example

The manager of a car plant wishes to investigate how the plant's electricity usage depends upon the plant's production. The data set shown in Figure 12.5 is compiled and provides the plant's production and electrical usage for each month of the previous year. The electrical usage is in units of a million kilowatt-hours, and the production is measured as the value in million-dollar units of the cars produced in that month.

	Production ($ million)	Electricity usage (million kWh)
January	4.51	2.48
February	3.58	2.26
March	4.31	2.47
April	5.06	2.77
May	5.64	2.99
June	4.99	3.05
July	5.29	3.18
August	5.83	3.46
September	4.70	3.03
October	5.61	3.26
November	4.90	2.67
December	4.20	2.53

FIGURE 12.5

(i). Show the relationship between the plant's production and the plant's electricity usage graphically. Make a comment on your diagram.

(ii). Write down the regression model and estimated regression equation.

(ii). Find the least square estimates and write the estimated regression equation. Interpret the results.

(iv). If a production level of \$5.5 million worth of cars is planned for next month, then predict the electricity usage.

(v). Estimate the error variance.

(vi). Draw the regression line.

Solution of Example

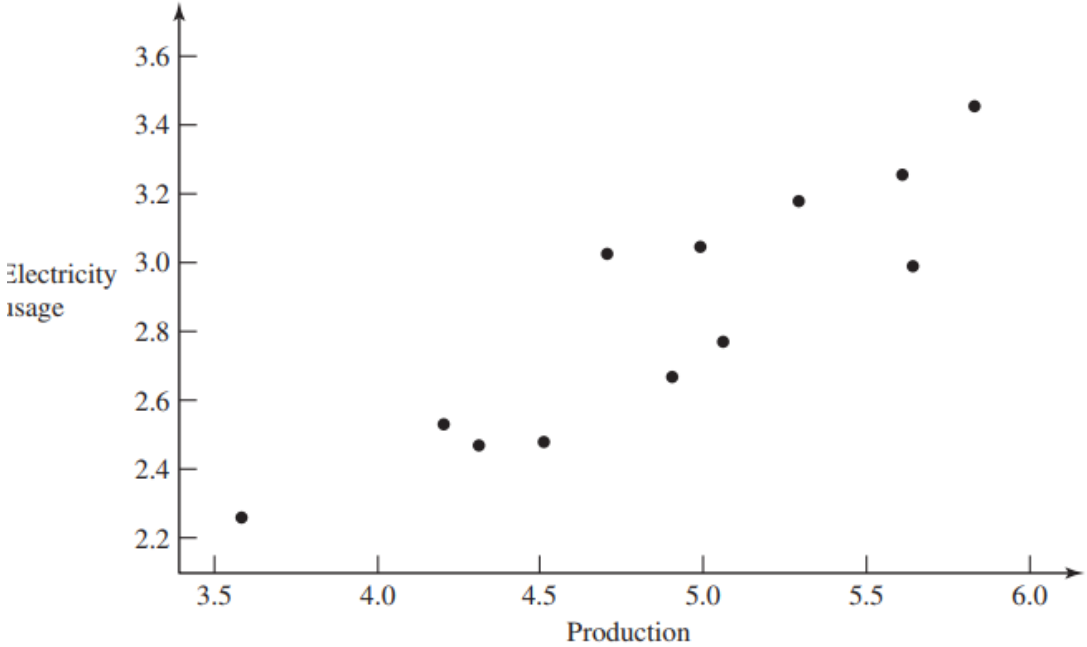


Figure 12.6 shows a graph of the data set, and the manager concludes that it is sensible to fit a straight line to the data points. The electricity usage is taken to be the dependent variable y, and the production is taken to be the explanatory variable x. The linear model

$$y = \beta_0 + \beta_1 x + \epsilon$$

Example 67

Car Plant Electricity Usage

For this example $n = 12$ and

$$\sum_{i=1}^{12} x_i = 4.51 + \cdots + 4.20 = 58.62$$

$$\sum_{i=1}^{12} y_i = 2.48 + \cdots + 2.53 = 34.15$$

$$\sum_{i=1}^{12} x_i^2 = 4.51^2 + \cdots + 4.20^2 = 291.2310$$

$$\sum_{i=1}^{12} y_i^2 = 2.48^2 + \cdots + 2.53^2 = 98.6967$$

$$\sum_{i=1}^{12} x_i y_i = (4.51 \times 2.48) + \cdots + (4.20 \times 2.53) = 169.2532$$

The estimate of the slope parameter is therefore

$$\hat{\beta}_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$$= \frac{(12 \times 169.2532) - (58.62 \times 34.15)}{(12 \times 291.2310) - 58.62^2} = 0.49883$$

and the estimate of the intercept parameter is

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = \frac{34.15}{12} - \left(0.49883 \times \frac{58.62}{12}\right) = 0.4090$$

The fitted regression line is thus

$$y = \hat{\beta}_0 + \hat{\beta}_1 x = 0.409 + 0.499x$$

which is shown together with the data points in Figure 12.16. If a production level of \$5.5 million worth of cars is planned for next month, then the plant manager can predict that the electricity usage will be

$$\hat{y}|_{5.5} = 0.409 + (0.499 \times 5.5) = 3.1535$$

With $\hat{\beta}_1 = 0.499$, this model predicts that the electricity usage increases by about half a million kilowatt-hours for every additional \$1 million of production.

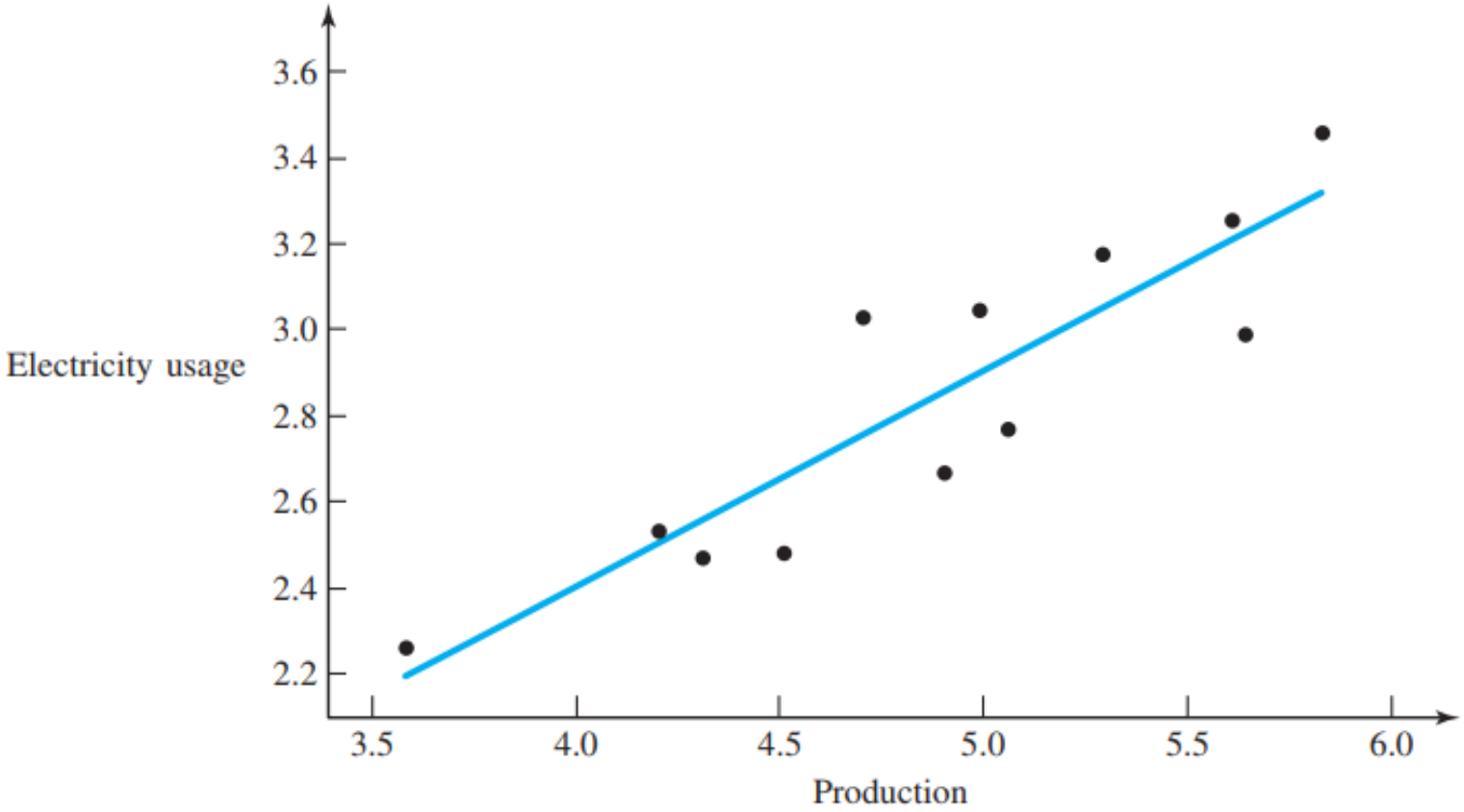
For this problem the error variance is estimated as

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^n y_i^2 - \hat{\beta}_0 \sum_{i=1}^n y_i - \hat{\beta}_1 \sum_{i=1}^n x_i y_i}{n-2}$$

$$= \frac{98.6967 - (0.4090 \times 34.15) - (0.49883 \times 169.2532)}{10} = 0.0299$$

so that

$$\hat{\sigma} = \sqrt{0.0299} = 0.1729$$



Fitted line plot for the car plant electricity usage data set.

Classroom Practice

1. The following sample of observations were randomly selected.

x	4	5	3	6	10
y	4	6	5	7	7

(a). Determine the regression equation.
(b). Write the estimated regression equation (or line).
(c). Determine the estimated value of y when x is 7.

Solution of the Exercises

x	y	x^2	y^2	xy
4	4	16	16	16
5	6	25	36	30
3	5	9	25	15
6	7	36	49	42
10	7	100	49	70
$\sum x_i = 28$	$\sum y_i = 29$	$\sum x_i^2 = 186$	$\sum y_i^2 = 175$	$\sum x_i y_i = 173$
$\bar{x} = 5.6$	$\bar{y} = 5.8$			

Solution of the Exercises

(a). the least square estimator of β_0 and β_1 are

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} = 0.3630 \quad \text{and} \quad \widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x} = 3.7671$$

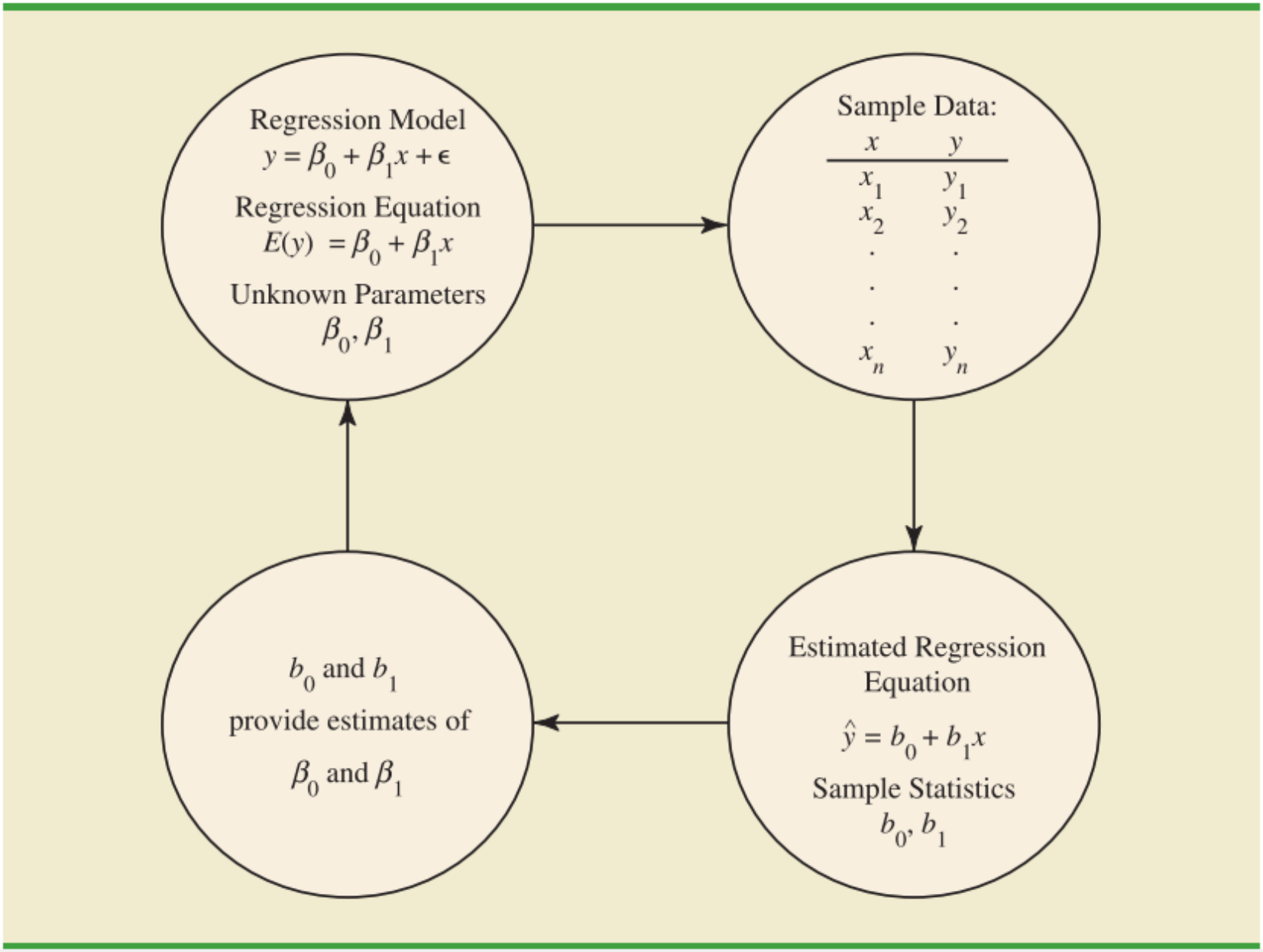
(b). hence the estimated equation (or line) is

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x = 3.7671 + 0.3630x$$

(c). the estimated value of y is

$$\widehat{y} = 3.7671 + 0.3630 \times 7 = 6.3082$$

when x is 7.


Regression Model
$y = \beta_0 + \beta_1 x + \epsilon$
Regression Equation
$E(y) = \beta_0 + \beta_1 x$
Unknown Parameters
β_0, β_1
Sample Data:
x y
x_1 y_1
x_2 y_2
x_n y_n
Estimated Regression Equation
$\hat{y} = b_0 + b_1 x$
Sample Statistics
b_0, b_1
b_0 and b_1
provide estimates of
β_0 and β_1

F TEST FOR SIGNIFICANCE IN SIMPLE LINEAR REGRESSION

$$H_0: \beta_1 = 0$$
$$H_a: \beta_1 \neq 0$$

TEST STATISTIC

$$F = \frac{\text{MSR}}{\text{MSE}} \qquad \textbf{(14.21)}$$

REJECTION RULE

p-value approach:	Reject H_0 if p-value α
Critical value approach:	Reject H_0 if $F \geq F_\alpha$

where F_α is based on an F distribution with 1 degree of freedom in the numerator and $n - 2$ degrees of freedom in the denominator.

$$H_0: \beta_1 = 0$$
$$H_a: \beta_1 \neq 0$$

TEST STATISTIC

$$t = \frac{b_1}{s_{b_1}}$$

REJECTION RULE

p-value approach:	Reject H_0 if p-value α
Critical value approach:	Reject H_0 if $t \leq -t_{\alpha/2}$ or if $t \geq t_{\alpha/2}$

where $t_{\alpha/2}$ is based on a t distribution with $n - 2$ degrees of freedom.

Multiple Linear Regression Model

- consider a bivariate random vector (X, Y), and iid observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ from (X, Y).
- the general interest is in a (multiple) linear regression model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

  where
  - $\boldsymbol{\beta}$ is the vector of regression coefficients, say of dimension $p \times 1$,
  - $\boldsymbol{X}$ is the design matrix, $\boldsymbol{Y}$ is the column vector of all Y -observations, and
  - $\boldsymbol{\varepsilon}$ is the column vector of the error terms $\varepsilon_1, \ldots, \varepsilon_n$, for which we assume that they are independent of each other and have the same distribution as the random variable ε
- under some assumptions regarding the random variable ε:

$$E(\varepsilon|\boldsymbol{X} = \boldsymbol{x}) = \boldsymbol{0} \text{ and } \mathrm{Var}(\varepsilon|\boldsymbol{X} = \boldsymbol{x}) = \sigma^2 \boldsymbol{I}$$

  with $0 < \sigma^2 < \infty$, the OLS estimator of $\boldsymbol{\beta}$ is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

Estimate of σ^2

- the mean square error (MSE):

$$\widehat{\sigma}^2 = MSE = \frac{SSE}{\mathrm{df}_e} = \frac{SSE}{(n-2)} \equiv s^2$$

- the sum of squares due to regression, is denoted SSR and defined by

$$SSR = \sum_i (\widehat{Y}_i - \bar{Y})^2$$

- it is known that

$$SST = SSR + SSE$$

  where
  - SST = total sum of square
  - SSR = sum of square due to regression
  - SSE = sum of square due to error

Coefficient of Determination

- the total sum of squares (proportional to the variance of the data): $SST = \sum_i (y_i - \bar{y})^2$,
- The sum of squares of residuals, also called the residual sum of squares: $SSE = \sum_i (y_i - \widehat{y}_i)^2 = \sum_i \varepsilon_i^2$
- the *coefficient of determination is the proportion of the total variation in the dependent variable Y that is explained, or accounted for, by the variation in the independent variable X*
- it is denoted by R^2 and defined by

$$R^2 = 1 - \frac{SSE}{SST}$$

- Suppose $R^2 = 0.49$. This implies that 49% of the variability of the dependent variable has been accounted for, and the remaining 51% of the variability is still unaccounted for

Coefficient of Determination (Conti...)

- R^2 is a statistic that will give some information about the goodness of fit of a model
- in regression, the R^2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points
- an R^2 of 1 indicates that the regression predictions perfectly fit the data
- R^2 increases as we increase the number of variables in the model (R^2 is monotone increasing with the number of variables included i.e., it will never decrease)
- an adjusted R^2 is a modification of R^2 that adjusts for the number of explanatory terms in a model (p) relative to the number of data points (n)

Classroom Practice

1. The following sample of observations were randomly selected.

No. of TV Commercials (x)	2	5	1	3	4	1	5	3	4	2
Total Sales(y)	50	57	41	54	54	38	63	48	59	46

(a). Find the linear relationship between number of TV Commercials and total sales.
(b). Fit a model y on x.
(c). Find the coefficient of determination. Interpret your findings.
(d). Predict (or forecast) total sales when $x = 5$.

Adjusted R^2

- the adjusted R^2 is defined as

$$\bar{R}^2 = 1 - (1 - R^2)\left(\frac{n-1}{n-p-1}\right) \quad ; \quad p \geq 1$$

where p is the total number of explanatory variables in the model (not including the constant term), and n is the sample size

- it can also be written as:

$$\bar{R}^2 = 1 - \frac{SSE/\mathrm{df}_e}{SST/\mathrm{df}_t}$$

where df_t is the degrees of freedom $n-1$ of the estimate of the population variance of the dependent variable, and df_e is the degrees of freedom $n-p-1$ of the estimate of the underlying population error variance

Adjusted R^2

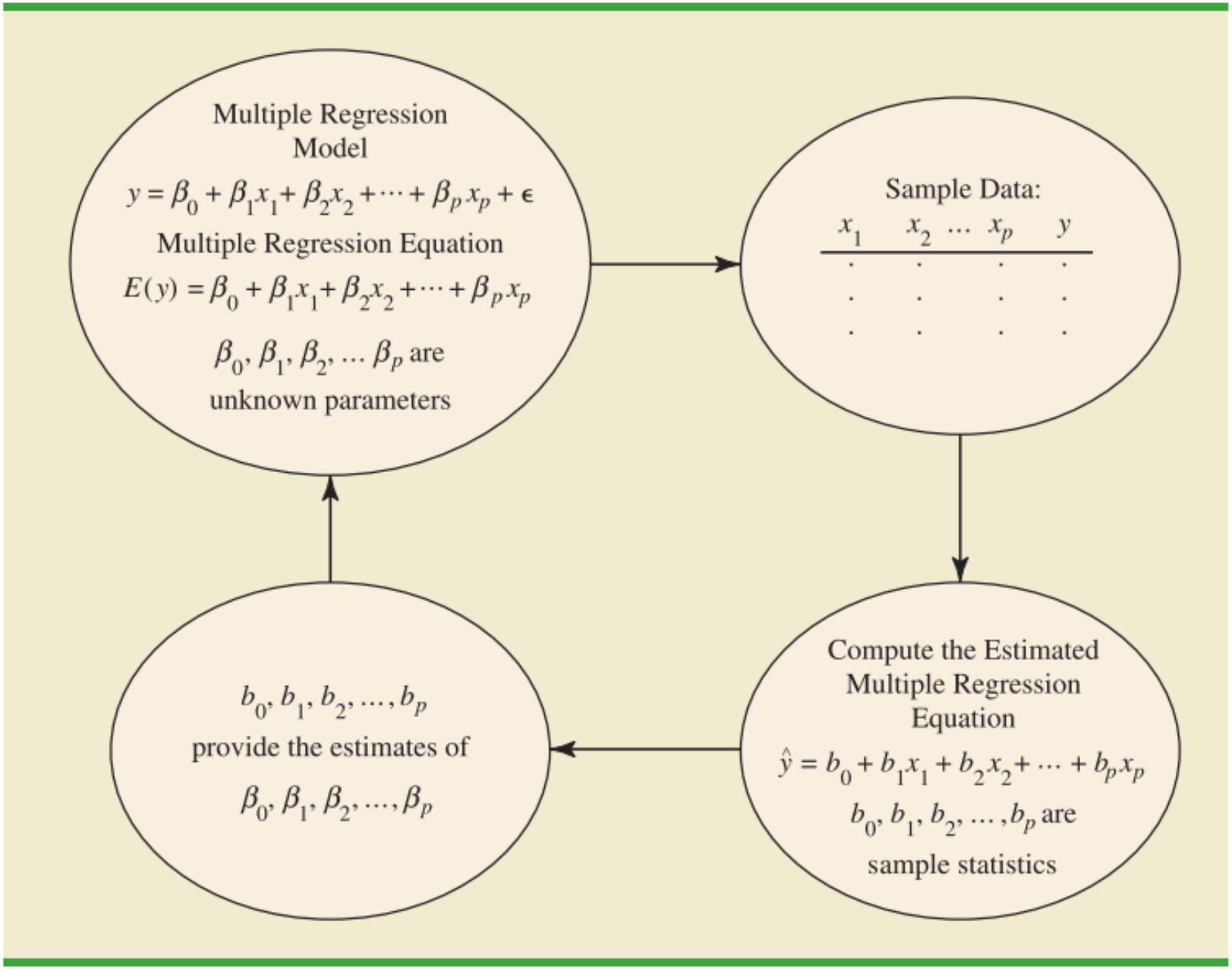
- the explanation of this statistic is almost the same as R^2 but it penalizes the statistic as extra variables are included in the model
- the term $\frac{n-1}{n-p-1}$ is called the penalty of using the more regressors in a model
- when the number of regressors p, increases, $(1 - R^2)$ will decrease, but $\frac{n-1}{n-p-1}$ will increase
- whether more regressors improve the explanatory power of a model depends on the trade-of between R^2 and the penalty
- R^2 and $\bar{R}^2$ are also used for measuring goodness-of-fit of the fitted model.

Homework

1. The following sample of observations were randomly selected.

No. of TV Commercials (x)	2	5	1	3	4	1	5	3	4	2
Total Sales(y)	50	57	41	54	54	38	63	48	59	46

(a). Find the linear relationship between number of TV Commercials and total sales.
(b). Fit a model y on x.
(c). Find the coefficient of determination. Interpret your findings.
(d). Find the adjusted R^2. Interpret your findings.
(e). Predict (or forecast) total sales when $x = 5$.


Multiple Regression Model
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$
Multiple Regression Equation
$E(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$
$\beta_0, \beta_1, \beta_2, \ldots \beta_p$ are unknown parameters
Sample Data:
x_1 x_2 … x_p y
Compute the Estimated Multiple Regression Equation
$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$
$b_0, b_1, b_2, \ldots, b_p$ are sample statistics
$b_0, b_1, b_2, \ldots, b_p$ provide the estimates of
$\beta_0, \beta_1, \beta_2, \ldots, \beta_p$

F TEST FOR OVERALL SIGNIFICANCE

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

H_a: One or more of the parameters is not equal to zero

TEST STATISTIC

$$F = \frac{\text{MSR}}{\text{MSE}} \tag{15.14}$$

REJECTION RULE

p-value approach:	Reject H_0 if p-value $\leq \alpha$
Critical value approach:	Reject H_0 if $F \geq F_\alpha$

where F_α is based on an F distribution with p degrees of freedom in the numerator and $n - p - 1$ degrees of freedom in the denominator.

Are all independent variables (feathers) significant?

Source	Sum of Squares	Degrees of Freedom	Mean Square	F
Regression	SSR	p	$\text{MSR} = \dfrac{\text{SSR}}{p}$	$F = \dfrac{\text{MSR}}{\text{MSE}}$
Error	SSE	$n - p - 1$	$\text{MSE} = \dfrac{\text{SSE}}{n - p - 1}$	
Total	SST	$n - 1$		

If the p-value of F-statistic is less than α, it indicates that all independent variables are not insignificant at $100\alpha\%$ level of significance. Now, try to check individually whether the independent variable is significant or not. Check the p-value of the t-statistic.

t TEST FOR INDIVIDUAL SIGNIFICANCE

For any parameter β_i

$$H_0: \beta_i = 0$$
$$H_a: \beta_i \neq 0$$

TEST STATISTIC

$$t = \frac{b_i}{s_{b_i}}$$

REJECTION RULE

p-value approach:	Reject H_0 if p-value $\leq \alpha$
Critical value approach:	Reject H_0 if $t \leq -t_{\alpha/2}$ or if $t \geq t_{\alpha/2}$

where $t_{\alpha/2}$ is based on a t distribution with $n - p - 1$ degrees of freedom.

Consider the following multiple regression model involving q independent variables, where $q < p$.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q + \epsilon \tag{16.11}$$

If we add variables $x_{q+1}, x_{q+2}, \ldots, x_p$ to this model, we obtain a model involving p independent variables.

$$\begin{aligned} y = \beta_0 &+ \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q \\ &+ \beta_{q+1} x_{q+1} + \beta_{q+2} x_{q+2} + \cdots + \beta_p x_p + \epsilon \end{aligned} \tag{16.12}$$

To test whether the addition of $x_{q+1}, x_{q+2}, \ldots, x_p$ is statistically significant, the null and alternative hypotheses can be stated as follows.

H_0: $\beta_{q+1} = \beta_{q+2} = \cdots = \beta_p = 0$
H_a: One or more of the parameters is not equal to zero

$$H_0: \beta_{q+1} = \beta_{q+2} = \cdots = \beta_p = 0$$
$$H_a: \text{One or more of the parameters is not equal to zero}$$

The following F statistic provides the basis for testing whether the additional independent variables are statistically significant.

$$F = \frac{\dfrac{\text{SSE}(x_1, x_2, \ldots, x_q) - \text{SSE}(x_1, x_2, \ldots, x_q, x_{q+1}, \ldots, x_p)}{p - q}}{\dfrac{\text{SSE}(x_1, x_2, \ldots, x_q, x_{q+1}, \ldots, x_p)}{n - p - 1}} \tag{16.13}$$

This computed F value is then compared with F_α, the table value with $p - q$ numerator degrees of freedom and $n - p - 1$ denominator degrees of freedom. If $F > F_\alpha$, we reject H_0 and conclude that the set of additional independent variables is statistically significant. Note that for the special case where $q = 1$ and $p = 2$, equation (16.13) reduces to equation (16.10).

Python Code: OLS for Linear Regression Model

```
# Simple Linear Regression Model
import statsmodels.api as sm
model =sm.OLS(boston.MEDV, sm.add_constant(boston.LSTAT))
result = model.fit()
print(result.summary())
```

```python
import statsmodels.api as sm
model =sm.OLS(boston.MEDV, sm.add_constant(boston.LSTAT))
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.544
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     601.6
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           5.08e-88
Time:                        03:26:52   Log-Likelihood:                -1641.5
No. Observations:                 506   AIC:                             3287.
Df Residuals:                     504   BIC:                             3295.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.5538      0.563     61.415      0.000      33.448      35.659
LSTAT         -0.9500      0.039    -24.528      0.000      -1.026      -0.874
==============================================================================
Omnibus:                      137.043   Durbin-Watson:                   0.892
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              291.373
Skew:                           1.453   Prob(JB):                     5.36e-64
Kurtosis:                       5.319   Cond. No.                         29.7
==============================================================================
```

Python Code: OLS for Multiple Linear Regression Model

```
# Multiple Linear Regression Model
import statsmodels.api as sm
X = boston[['LSTAT', 'CRIM']]
model =sm.OLS(boston.MEDV, sm.add_constant(X))
result = model.fit()
print(result.summary())

## Alternatively
import statsmodels.formula.api as smf
# formula: response ~ predictor + predictor
est = smf.ols(formula='MEDV ~ LSTAT + CRIM',
    data=boston).fit()
print(est.summary())

# In GLM framework
Gaussian_model = sm.GLM(boston.MEDV, sm.add_constant(X),
    family=sm.families.Gaussian()).fit()
print(Gaussian_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.548
Model:                            OLS   Adj. R-squared:                  0.546
Method:                 Least Squares   F-statistic:                     304.4
Date:                Fri, 27 Jan 2023   Prob (F-statistic):           2.33e-87
Time:                        03:37:58   Log-Likelihood:                -1639.6
No. Observations:                 506   AIC:                             3285.
Df Residuals:                     503   BIC:                             3298.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.3192      0.574     59.816      0.000      33.192      35.446
LSTAT         -0.9114      0.043    -21.004      0.000      -0.997      -0.826
CRIM          -0.0704      0.036     -1.956      0.051      -0.141       0.000
==============================================================================
Omnibus:                      146.159   Durbin-Watson:                   0.872
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              329.172
Skew:                           1.517   Prob(JB):                     3.32e-72
Kurtosis:                       5.531   Cond. No.                         32.7
==============================================================================
```

Chapter 8: Logistic Regression Classifier

8 Chapter 8: Logistic Regression Classifier

8.1 Problem & Motivation

8.2 Scatter Diagram

8.3 Correlation Coefficient

8.4 Regression Analysis

8.5 Types of parametric regression analysis

8.6 Logistic Regression

8.7 Interpreting the Logistic Regression Equation

8.8 Multinomial Logistic Regression

8.9 Multinomial Logistic Regression Classifier

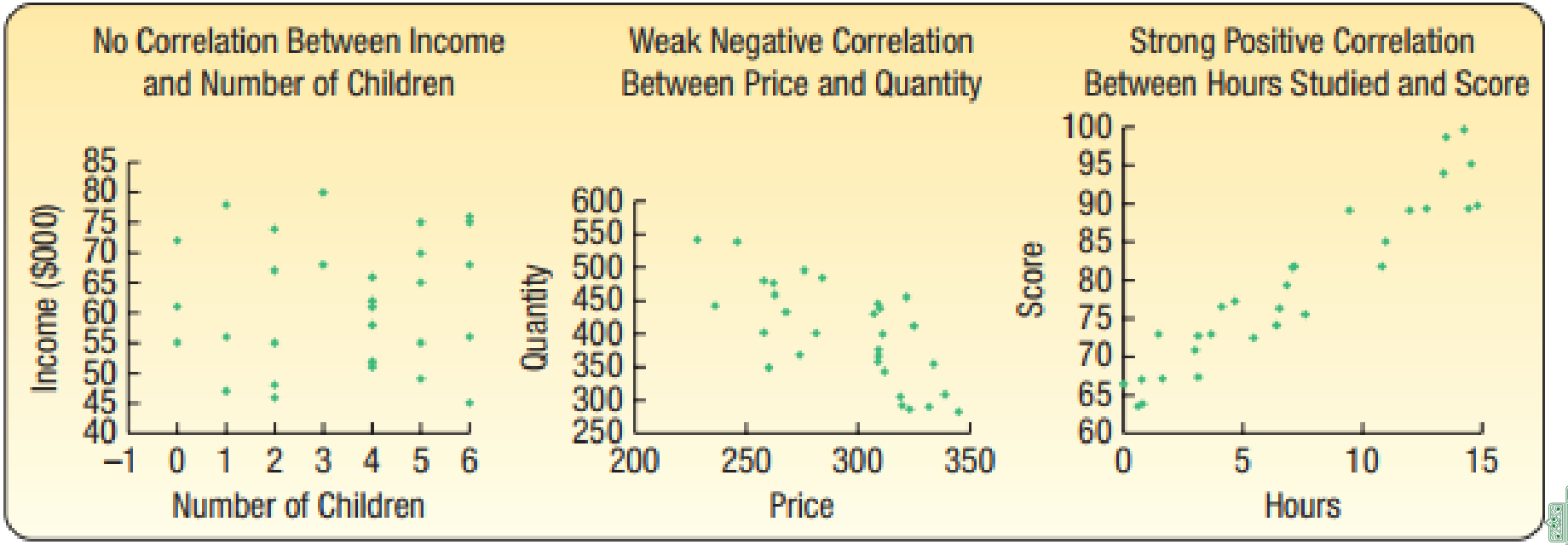
8.10 Geometric Intuition of Logistic Classifiers

8.11 Python code: Logistic Regression Classifier

Scatter Diagram

a scatter diagram is a graphic tool used to portray the relationship between two variables

- the dependent variable is scaled on the Y-axis and is the variable being estimated
- the independent variable is scaled on the X-axis and is the variable used as the predictor



What is Correlation Analysis?

Correlation analysis

▸▸a group of techniques to measure the relationship between two variables

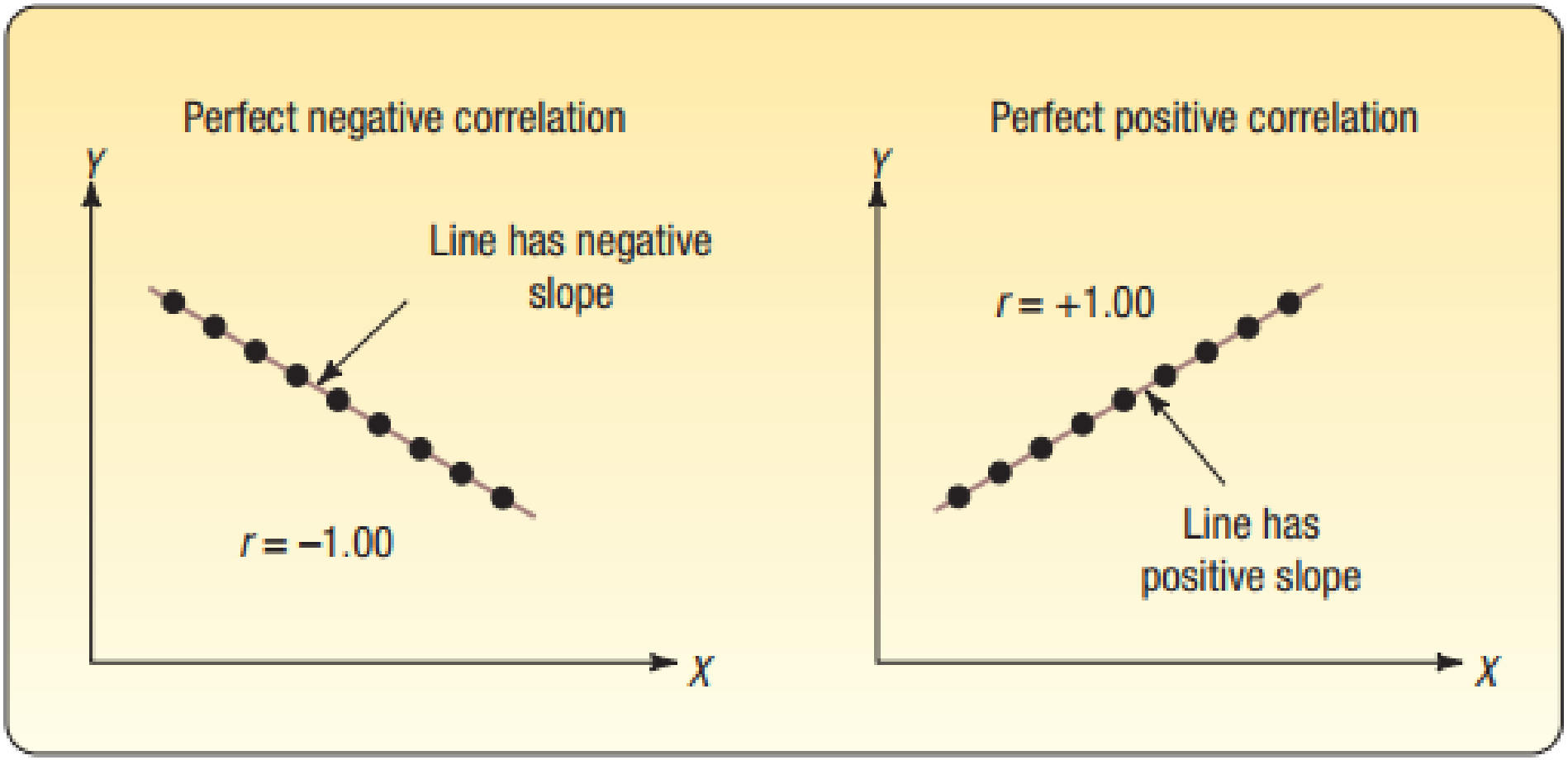
Pearson's Correlation coefficient

- is a measure of the strength of the linear relationship between two variables
- is denoted by r and defined as

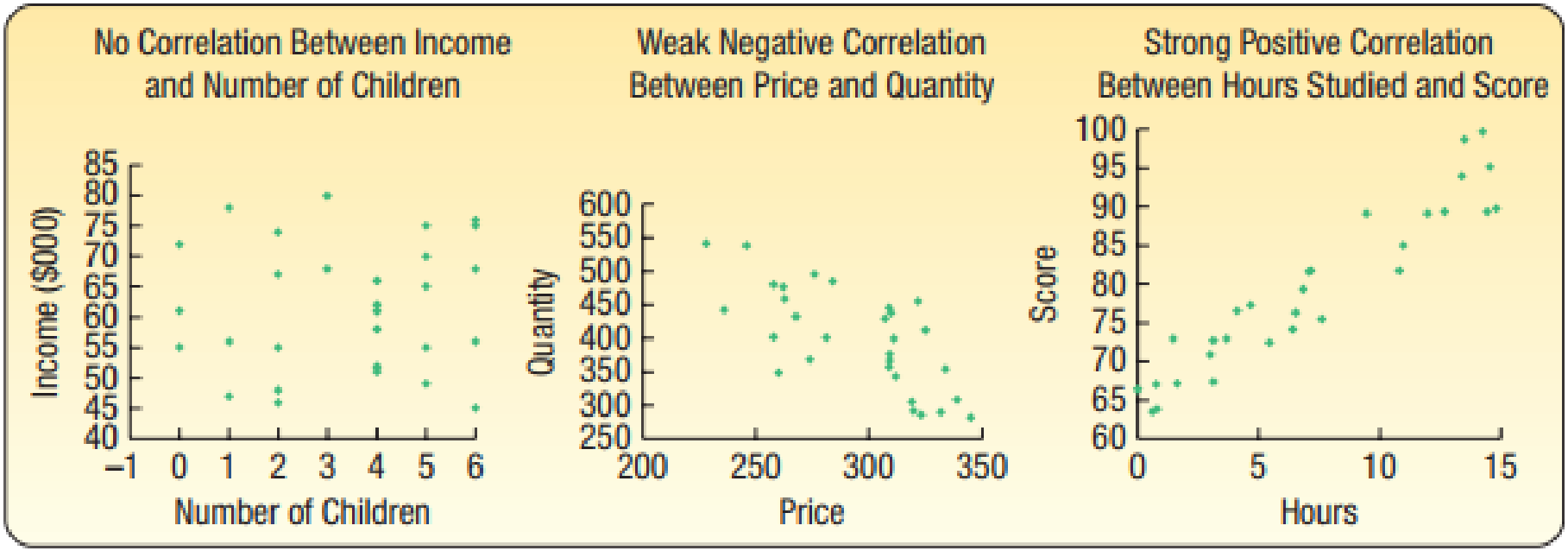
$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}\sqrt{\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right)}}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

Correlation Coefficient

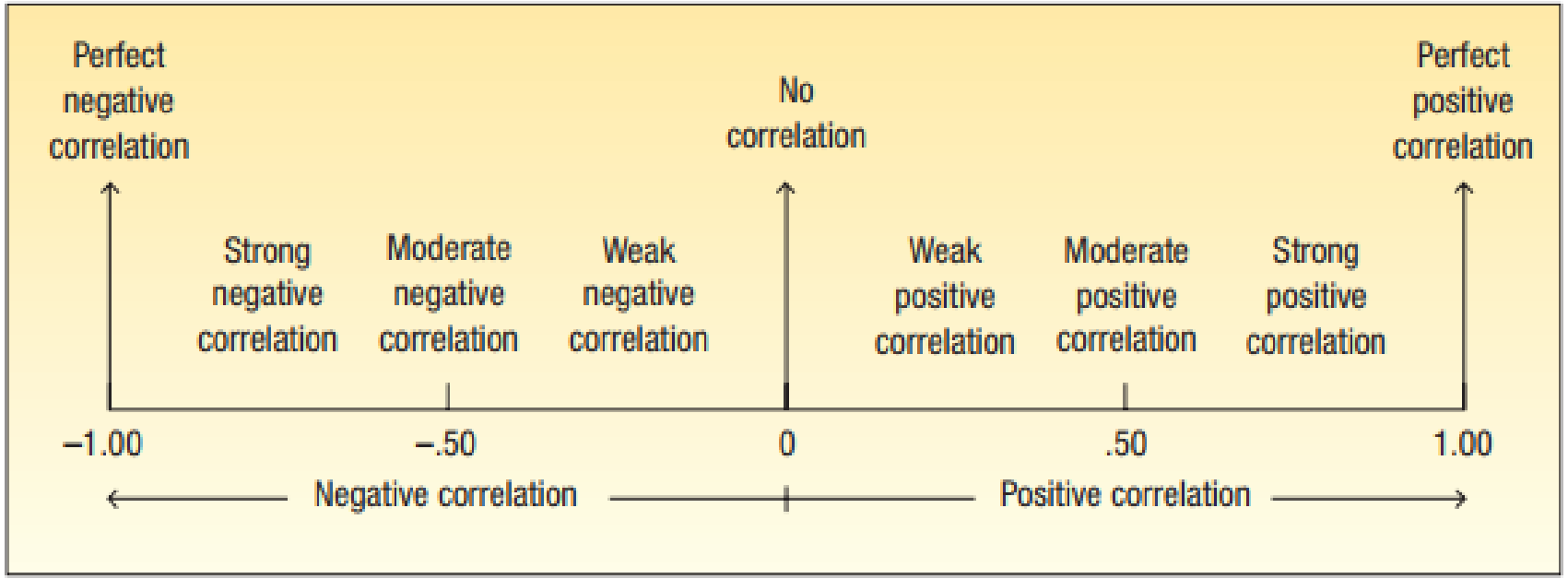


Correlation Coefficient



Correlation Coefficient

the following drawing summarizes the strength and direction of the correlation coefficient



Characteristics of correlation coefficient

1. the sample correlation coefficient is identified by the lowercase letter r
2. it shows the direction and strength of the linear relationship between two interval or ratio-scale variables
3. it ranges from -1 up to and including $+1$
4. a value near 0 indicates there is little linear relationship between the variables
5. a value near 1 indicates a direct or positive linear relationship between the variables
6. a value near -1 indicates an inverse or negative linear relationship between the variables

Regression Analysis

- correlation analysis *does not tell us why* and *how behind the relationship* but it just says the relationship exists
- regression is to build a function of independent variables (also known as predictors) to predict a dependent variable (also called response) for example,

$$y = f(x_1, x_2, \ldots, x_p) + \epsilon$$

- for example, banks assess the risk of home-loan applicants based on their age, income, expenses, occupation, number of dependents, total credit limit, etc.

Types of parametric regression analysis

- for continuous response variable
  - Simple Linear Regression
  - Multiple Regression
  - Polynomial Regression
  - Multivariate Regression
- for categorical response variable
  - Logistic Regression
    - binomial (also called ordinary) logistic regression
    - multinomial logistic regression
    - ordinal logistic regression
    - alternating logistic regressions
- for discrete response variable
  - Poisson Regression
- for survival-time (time-to-event) outcomes
  - Cox Regression (or proportional hazards regression)

Regression Models

another types of regression analysis:

- Ridge Regression
- Lasso Regression
- Bayesian regression
- Nonparametric regression

other types of regression model in machine learning

- Generalized linear model (GLM)
- Additive model
- Generalized additive model (GAM)
- Generalized additive model for location, scale and shape (GAMLSS)
- Random effect model
- Linear mixed model (LMM)
- Generalized linear mixed model (GLMM)
- Generalized estimating equations (GEE)

Logistic Regression

- logistic regression is used to predict a categorical response variable (nominal & ordinal level of measurement data) given one or more independent variables
- the main idea of logistic regression is to estimate the probability that the *dependent variable belongs to a category* given certain features
- a binomial logistic regression (often referred to simply as logistic regression), predicts the probability that an observation falls into one of two categories of a dichotomous dependent variable (nominal level-of-measurement) based on one or more independent variables that can be either continuous or categorical

Binary Logistic Regression

- consider a model with p predictors, $x_1, ..., x_p$, and one binary (Bernoulli) response variable y, which we denote $p \equiv \Pr(y = 1)$
- if the two values of the dependent variable y are coded as 0 or 1, the value of $E(y)$ provides the probability that $y = 1$ given a particular set of values for the independent variables $x_1, x_2, \ldots, x_p$, that is

$$E(y) = \Pr(Y = 1 | x_1, \ldots, x_p)$$

- the value of $E(y)$ ranges from 0 to 1, with the value of $E(y)$ gradually approaching 1 as the value of x becomes larger and the value of $E(y)$ approaching 0 as the value of x becomes smaller
- we assume a linear relationship between the predictor variables and the log-odds (also called logit) of the event that $Y = 1$

Bernoulli distribution: Example

It is a discrete probability distribution of a random binary variable:

$$f(y_i|p) = p^{y_i}(1-p)^{1-y_i}$$

$$f(y_i|p) = \exp\left(y_i \log\frac{p}{1-p} + \log(1-p)\right)$$

$$\boxed{f_{\theta_i}(y_i) = \exp\left(\frac{y_i\,\theta_i - b(\theta_i)}{\phi_i} + c(y_i, \phi_i)\right)}$$

$$\theta_i = \log\frac{p}{1-p} \longrightarrow f(y_i|\theta_i) = \exp\left(y_i\theta_i - \log(1+e_i^{\theta})\right)$$

$$b(\theta_i) = \log(1+e^{\theta_i}) \longrightarrow \mathbb{E}[y_i] = \frac{e^{\theta_i}}{(1+e^{\theta_i})} = p$$

$$b(\theta_i) = \log(1+e^{\theta_i}) \longrightarrow \mathrm{var}[y_i] = \frac{e^{\theta_i}}{(1+e^{\theta_i})^2} = p(1-p)$$

- this linear relationship can be written in the following mathematical form (where θ is the log-odds, e is the base of the logarithm, and β_i are parameters of the model):

$$\theta = \log_e\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

- we can recover the odds by exponentiating the log-odds:

$$\frac{p}{1-p} = e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}$$

- by simple algebraic manipulation (and dividing numerator and denominator by $e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}$), the probability that $Y = 1$ is

$$p = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}} = \frac{e^s}{1+e^s} \text{ where } s = \sum_{j=0}^{p} \beta_j x_j$$

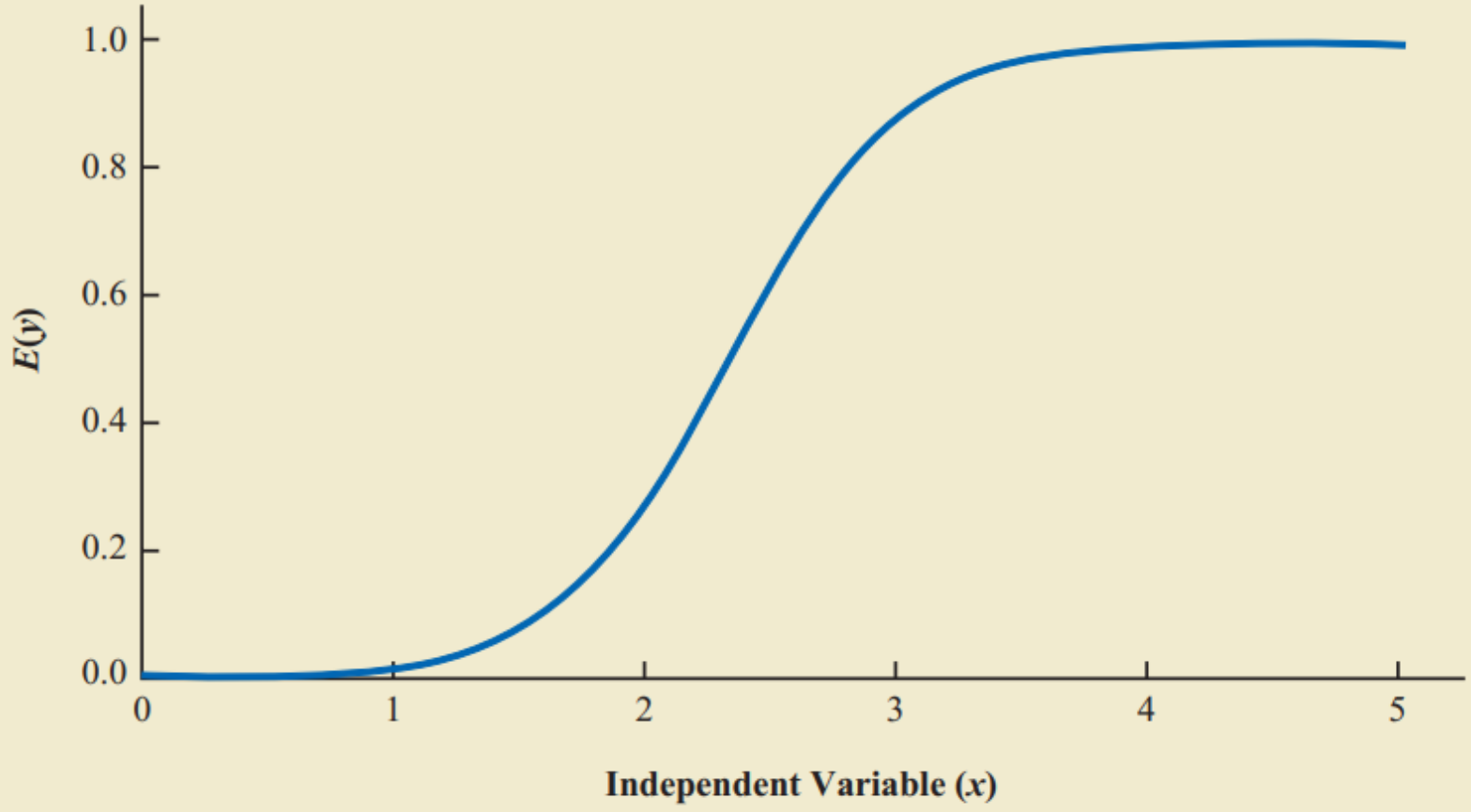
  which is the sigmoid function

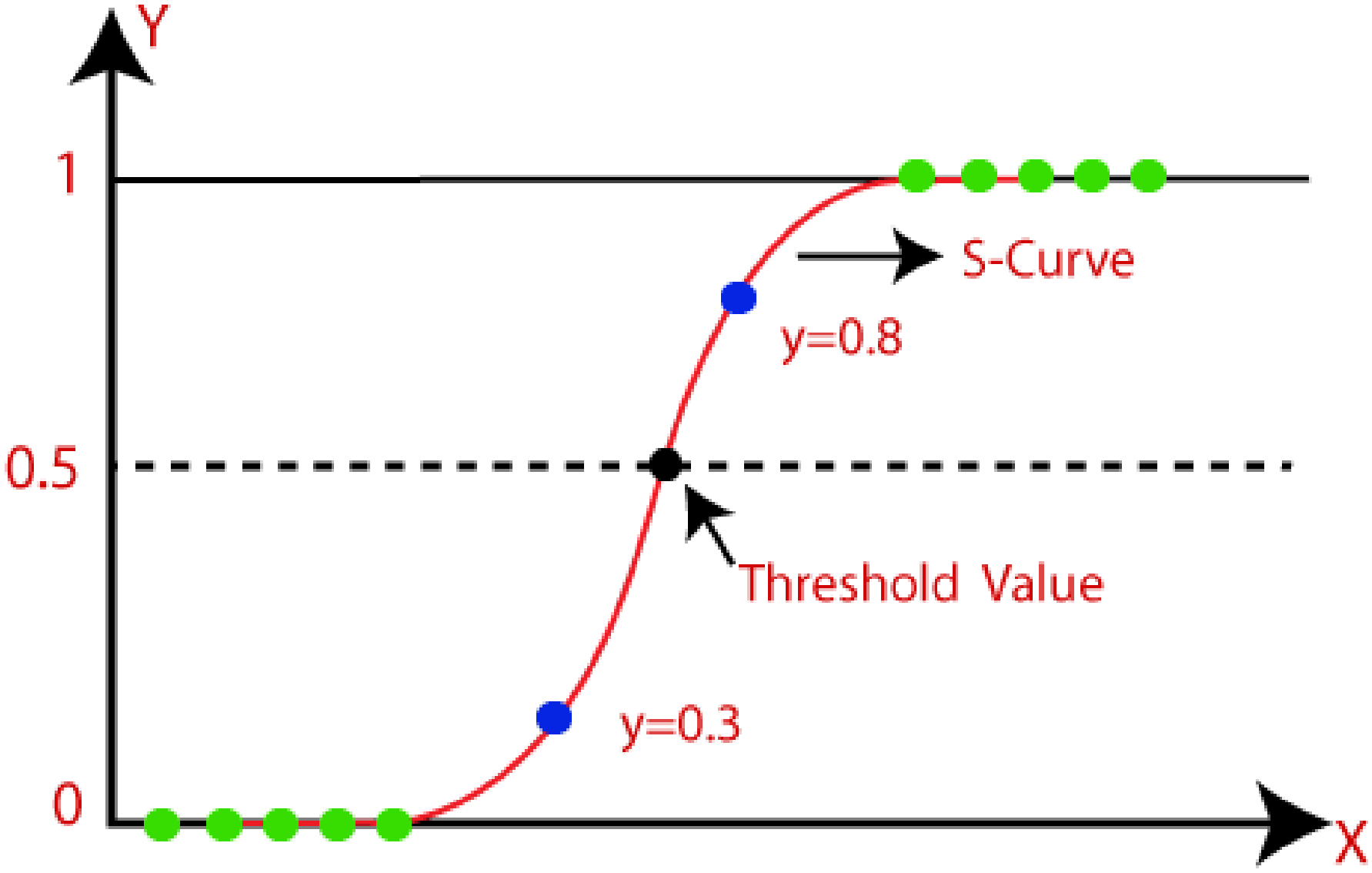
- logistic regression equation

$$E(y) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

For two variables

FIGURE 15.12 LOGISTIC REGRESSION EQUATION FOR $\beta_0 = -7$ AND $\beta_1 = 3$




Y
1
0.5
0
X
S-Curve
y=0.8
y=0.3
Threshold Value

Example

An experiment is done to test the effect of a toxic substance on insects.

- The data originate from the textbook, Applied Linear Statistical Models by Kutner, Nachtsheim, Neter, & Li.
- At each of six dose levels, 250 insects are exposed to the substance and the number of insects that die is counted.
- We can use statistical software to calculate the observed probabilities as the number of observed deaths out of 250 for each dose level.

Dose	Sample Size (n_i)	Deaths (d_i)
1	250	28
2	250	53
3	250	93
4	250	126
5	250	172
6	250	197

Example

- A binary logistic regression model is used to describe the connection between the observed probabilities of death as a function of dose level. The data is in event/trial format, which has to be taken into account by the statistical software used to conduct the analysis.

Example:

1. Step 1: Data Preparation

Age (X1)	Smoking (X2)	Heart Attack (Y)
40	0	0
55	1	1
48	1	0
60	0	1
35	0	0
50	1	1
45	0	0
58	1	1
42	0	0

2. **Step 2: Model Training**
   Using logistic regression:

$$\text{logit}(p) = \beta_0 + \beta_1 \times \text{Age} + \beta_2 \times \text{Smoking}$$

   where

   - β_0 is the intercept,
   - β_1 is the coefficient for Age,
   - β_2 is the coefficient for Smoking.

3. **Step 3: Model Coefficients**
   - Intercept (β_0): Value from model
   - Coefficient for Age (β_1): Value from model
   - Coefficient for Smoking (β_2): Value from model

4. Step 4: Making Predictions
   - For a 45-year-old non-smoker (Age=45, Smoking=0):
   - Probability of Heart Attack: Value from model prediction

Python code: Logistic Regression

```
#Step 1: Data Preparation
import pandas as pd

# Creating a DataFrame from the given data
data = pd.DataFrame({
  'Age': [40, 55, 48, 60, 35, 50, 45, 58, 42],
  'Smoking': [0, 1, 1, 0, 0, 1, 0, 1, 0],
  'HeartAttack': [0, 1, 0, 1, 0, 1, 0, 1, 0]
})

# Step 2: Model Training python
from sklearn.linear_model import LogisticRegression

# Extracting features (X) and target variable (y)
X = data[['Age', 'Smoking']]
y = data['HeartAttack']
```

```python
# Creating and fitting the logistic regression model
model = LogisticRegression()
model.fit(X, y)
# Intercept and coefficients
intercept = model.intercept_[0]
coeff_age = model.coef_[0][0]
coeff_smoking = model.coef_[0][1]

print(f'Intercept (beta0): {intercept}')
print(f'Coefficient for Age (beta1): {coeff_age}')
print(f'Coefficient for Smoking (beta2): {coeff_smoking}')
```

Note: `sklearn` does not provide a built-in function to show the summary of a regression model. It does however provide functions to extract the model coefficients and intercept:

```python
import pandas as pd
import statsmodels.api as sm

# Step 1: Data Preparation
data = pd.DataFrame({
 'Age': [40, 55, 48, 60, 35, 50, 45, 58, 42],
 'Smoking': [0, 1, 1, 0, 0, 1, 0, 1, 0],
 'HeartAttack': [0, 1, 0, 1, 0, 1, 0, 1, 0]
})

# Step 2: Model Training
X = sm.add_constant(data[['Age', 'Smoking']]) # add a constant
    term for the intercept
y = data['HeartAttack']
model = sm.Logit(y, X)
result = model.fit()

# Step 3: Model Summary
print(result.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            HeartAttack   No. Observations:                    9
Model:                          Logit   Df Residuals:                        6
Method:                           MLE   Df Model:                            2
Date:                Sat, 18 Nov 2023   Pseudo R-squ.:                   1.000
Time:                        07:06:46   Log-Likelihood:            -2.6995e-05
converged:                       False   LL-Null:                       -6.1827
Covariance Type:            nonrobust   LLR p-value:                  0.002065
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       -599.0874   4.87e+04     -0.012      0.990    -9.6e+04    9.48e+04
Age           12.8833   1022.665      0.013      0.990   -1991.503    2017.270
Smoking      -29.8408   1.61e+04     -0.002      0.999   -3.16e+04    3.16e+04
==============================================================================
```

Example: Partial sample data for the Simmons stores example

TABLE 15.11 PARTIAL SAMPLE DATA FOR THE SIMMONS STORES EXAMPLE

Customer	Annual Spending ($1000)	Simmons Card	Coupon
1	2.291	1	0
2	3.215	1	0
3	2.135	1	0
4	3.924	0	0
5	2.528	1	0
6	2.473	0	1
7	2.384	0	0
8	7.076	0	0
9	1.182	1	1
10	3.345	0	0

Let us now return to the Simmons Stores example. The variables in the study are defined as follows:

$$y = \begin{cases} 0 \text{ if the customer did not use the coupon} \\ 1 \text{ if the customer used the coupon} \end{cases}$$

$$x_1 = \text{annual spending at Simmons Stores (\$1000s)}$$

$$x_2 = \begin{cases} 0 \text{ if the customer does not have a Simmons credit card} \\ 1 \text{ if the customer has a Simmons credit card} \end{cases}$$

Thus, we choose a logistic regression equation with two independent variables.

$$E(y) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2}} \tag{15.31}$$

Summary Results

FIGURE 15.13 PARTIAL LOGISTIC REGRESSION OUTPUT FOR THE SIMMONS STORES EXAMPLE

```
Logistic Regression Table
                                                      Odds     95%   CI
Predictor        Coef    SE Coef       Z       p    Ratio   Lower   Upper
Constant     -2.14637   0.577245   -3.72   0.000
Spending     0.341643   0.128672    2.66   0.008     1.41    1.09    1.81
Card          1.09873   0.444696    2.47   0.013     3.00    1.25    7.17

Log-Likelihood = -60.487
Test that all slopes are zero: G = 13.628, DF = 2, P-Value = 0.001
```

- the estimated logistic regression equation

$$\widehat{p} \equiv \widehat{y} = \frac{e^{b_0+b_1x_1+b_2x_2}}{1+e^{b_0+b_1x_1+b_2x_2}}$$

- for this example, the estimated logistic regression equation

$$\widehat{y} = \frac{e^{-2.14637+0.341643x_1+1.09873x_2}}{1+e^{-2.14637+0.341643x_1+1.09873x_2}}$$

- For example, to estimate the probability of using the coupon for customers who spend \$2000 annually and do not have a Simmons credit card, we substitute for $x_1 = 2$ and $x_2 = 0$,

$$\widehat{y} = \frac{e^{-2.14637+0.341643(2)+1.09873(0)}}{1+e^{-2.14637+0.341643(2)+1.09873(0)}} = 0.1880$$

thus, an estimate of the probability of using the coupon for this particular group of customers is approximately 0.19.

- similarly, to estimate the probability of using the coupon for customers who spend \$2000 annually and have a Simmons credit card, we substitute for $x_1 = 2$ and $x_2 = 1$

$$\widehat{y} = \frac{e^{-2.14637+0.341643(2)+1.09873(1)}}{1 + e^{-2.14637+0.341643(2)+1.09873(1)}} = 0.4099$$

- thus, for this group of customers, the probability of using the coupon is approximately 0.41
- it appears that the probability of using the coupon is much higher for customers with a Simmons credit card. Before reaching any conclusions, however, we need to assess the statistical significance of our model

Interpreting the Logistic Regression Equation

- The odds in favor of an event occurring is defined as the probability the event will occur divided by the probability the event will not occur
- in logistic regression the event of interest is always $y = 1$
- given a particular set of values for the independent variables, the odds in favor of $y = 1$ can be calculated as follows:

$$\text{odds} = \frac{\Pr(y = 1|x_1, \ldots, x_p)}{\Pr(y = 0|x_1, \ldots, x_p)} = \frac{\Pr(y = 1|x_1, \ldots, x_p)}{1 - \Pr(y = 1|x_1, \ldots, x_p)}$$

- the odds ratio measures the impact on the odds of a one-unit increase in only one of the independent variables
- the odds ratio is the odds that $y = 1$ given that one of the independent variables has been increased by one unit (odds1) divided by the odds that $y = 1$ given no change in the values for the independent variables (odds_0)
- odds ratio is

$$\text{Odds Ratio} = \frac{\text{odds}_1}{\text{odds}_0}$$

- for example, suppose we want to compare the odds of using the coupon for customers who spend \$2000 annually and have a Simmons credit card ($x_1 = 2$ and $x2 = 1$) to the odds of using the coupon for customers who spend \$2000 annually and do not have a Simmons credit card ($x_1 = 2$ and $x2 = 0$)

$$\text{odds}_1 = \frac{\Pr(y = 1|x_1 = 2, x_2 = 1)}{1 - \Pr(y = 1|x_1 = 2, x_2 = 1)} \Rightarrow \widehat{\text{odds}}_1 = \frac{0.4099}{1 - 0.4099} = 0.6946$$

and

$$\text{odds}_0 = \frac{\Pr(y = 1|x_1 = 2, x_2 = 0)}{1 - \Pr(y = 1|x_1 = 2, x_2 = 0)} \Rightarrow \widehat{\text{odds}}_0 = \frac{0.1880}{1 - 0.1880} = 0.2315$$

- the estimated odds ratio is

$$\text{estimated odds ratio} = \frac{\widehat{\text{odds}}_1}{\widehat{\text{odds}}_1} = \frac{0.6946}{0.2315} = 3.00$$

- thus, we can conclude that the estimated odds in favor of using the coupon for customers who spent \$2000 last year and have a Simmons credit card are 3 times greater than the estimated odds in favor of using the coupon for customers who spent \$2000 last year and do not have a Simmons credit card

Note:

- the odds ratio for each independent variable is computed while holding all the other independent variables constant
- but it does not matter what constant values are used for the other independent variables

In general: Odds Ratio

- a unique relationship exists between the odds ratio for a variable and its corresponding regression coefficient
- For each independent variable in a logistic regression equation it can be shown that

$$\text{Odds Ratio} = e^{\beta_j} \quad ; \quad j = 1, \ldots, p$$

- To illustrate this relationship, consider the independent variable x_1 in the Simmons example
  - the estimated odds ratio for x_1 is

$$\text{Estimated odds ratio} = e^{b_1} = e^{0.341643} = 1.41$$

  - the estimated odds ratio for x_2 is

$$\text{Estimated odds ratio} = e^{b_2} = e^{1.09873} = 3.00$$

Now See the Summary Results and Interpret the Results

FIGURE 15.13 PARTIAL LOGISTIC REGRESSION OUTPUT FOR THE SIMMONS STORES EXAMPLE

Logistic Regression Table

Predictor	Coef	SE Coef	Z	p	Odds Ratio	95% CI Lower	95% CI Upper
Constant	-2.14637	0.577245	-3.72	0.000			
Spending	0.341643	0.128672	2.66	0.008	1.41	1.09	1.81
Card	1.09873	0.444696	2.47	0.013	3.00	1.25	7.17

Log-Likelihood = -60.487

Test that all slopes are zero: G = 13.628, DF = 2, P-Value = 0.001

Multinomial Logistic Regression

- is used when the dependent variable in question is nominal (equivalently categorical, meaning that it falls into any one of a set of categories that cannot be ordered in any meaningful way) and for which there are more than two categories
- for example:
  - which blood type does a person have, given the results of various diagnostic tests?
  - which candidate will a person vote for, given particular demographic characteristics?
  - if the respondents in the survey had three choices:
    (i). I feel optimistic,
    (ii). I don't feel optimistic,
    (iii). I am not sure,

Multinomial Logistic Regression

- is used when the dependent variable in question is nominal (equivalently categorical, meaning that it falls into any one of a set of categories that cannot be ordered in any meaningful way) and for which there are more than two categories
- for example:
  - which blood type does a person have, given the results of various diagnostic tests?
  - which candidate will a person vote for, given particular demographic characteristics?
  - if the respondents in the survey had three choices:
    (i). I feel optimistic,
    (ii). I don't feel optimistic,
    (iii). I am not sure,

- to arrive at the multinomial logit model, one can imagine, for K possible outcomes, running $K-1$ independent binary logistic regression models, in which one outcome is chosen as a "pivot" and then the other $K-1$ outcomes are separately regressed against the pivot outcome
- this would proceed as follows, if outcome K (the last outcome) is chosen as the pivot:

$$\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_1 \cdot \mathbf{X}_i$$

$$\ln \frac{\Pr(Y_i = 2)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_2 \cdot \mathbf{X}_i$$

$$\cdots\cdots$$

$$\ln \frac{\Pr(Y_i = K-1)}{\Pr(Y_i = K)} = \boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i$$

We can use this to find the other probabilities:

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{x}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{x}_i}}$$

$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{x}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{x}_i}}$$

$$\ldots\ldots$$

$$\Pr(Y_i = K-1) = \frac{e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{x}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{x}_i}}$$

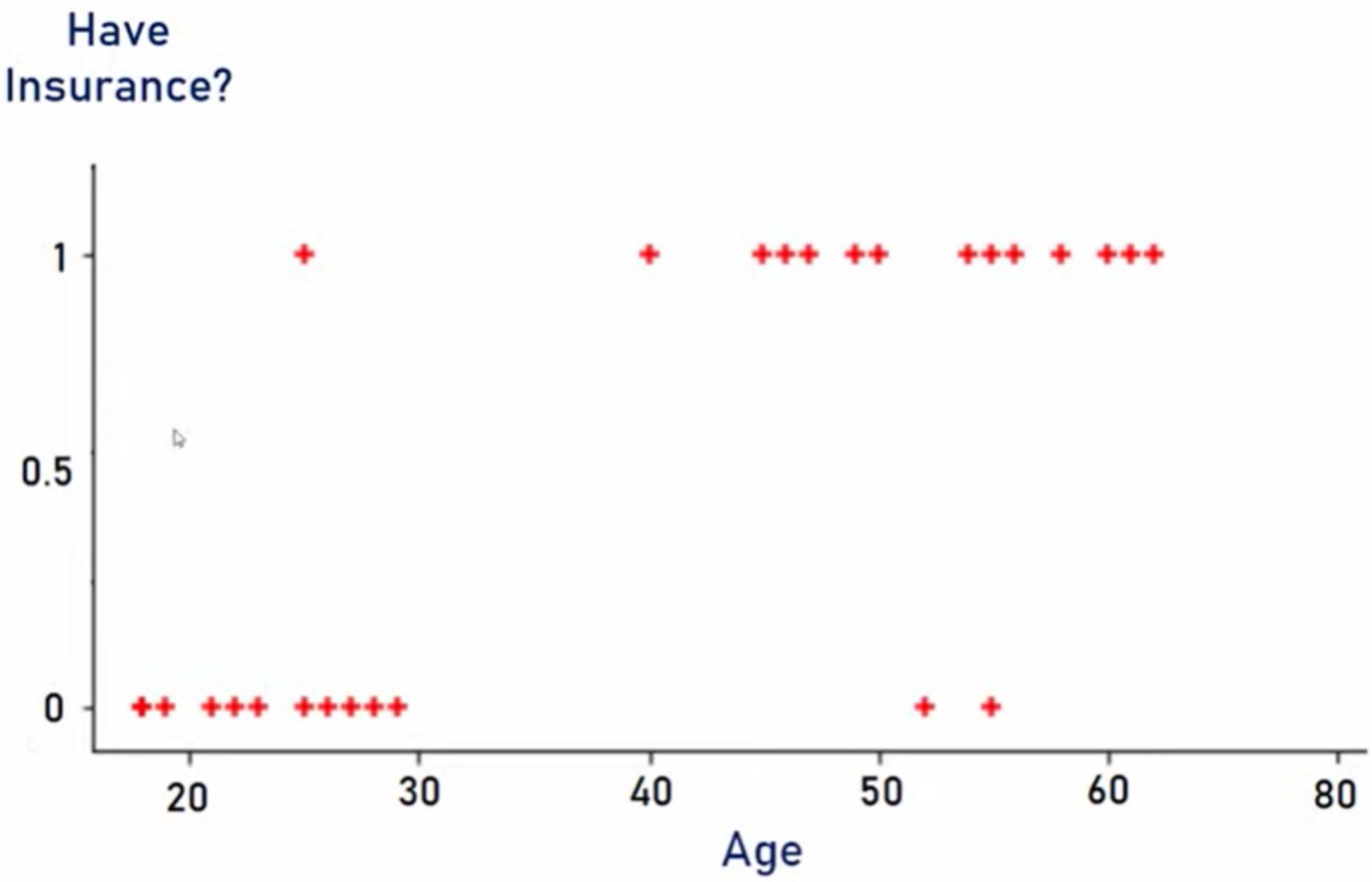
or generally: $\Pr(Y_i = c) = \dfrac{e^{\boldsymbol{\beta}_c \cdot \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{x}_i}}$

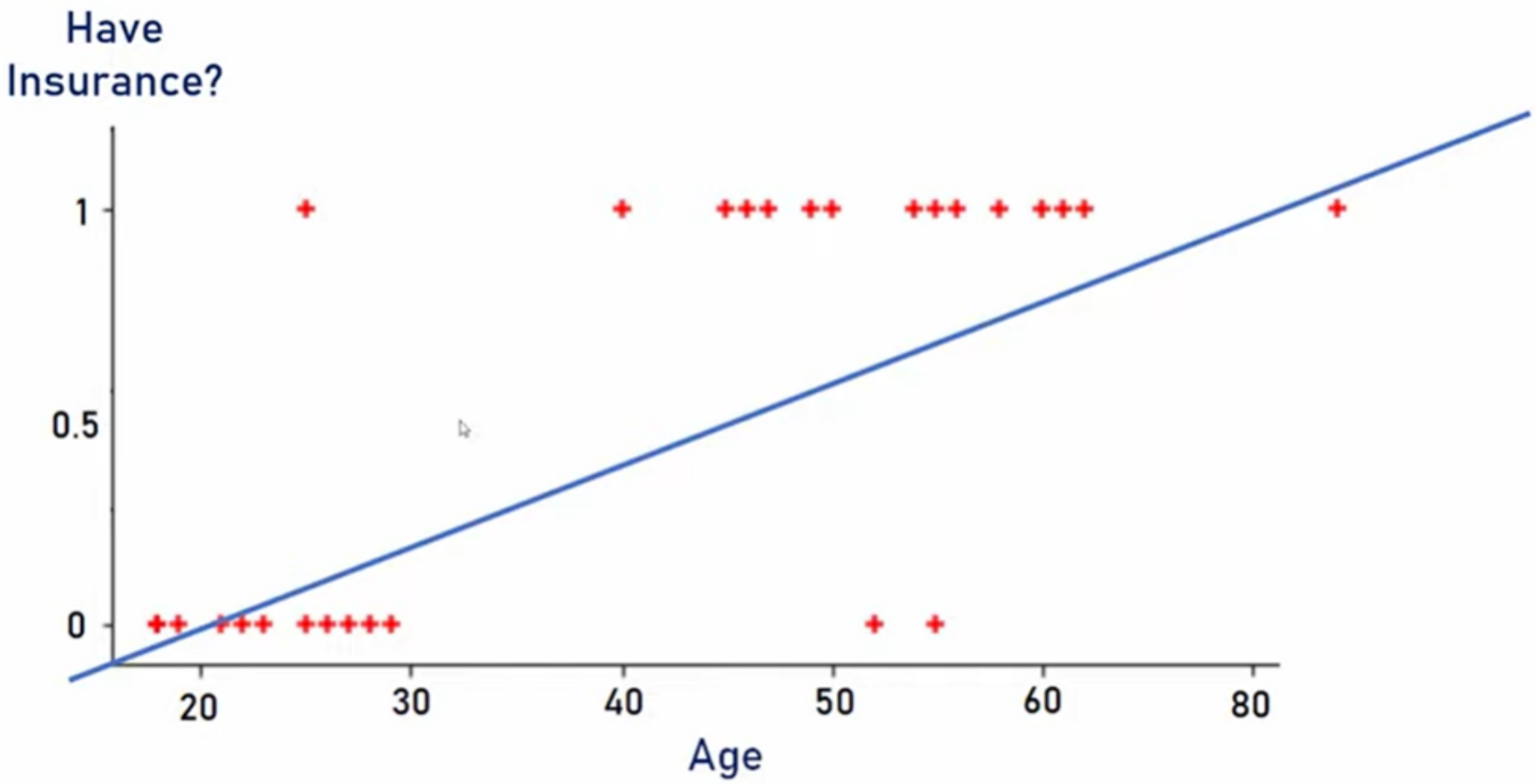
Example: Geometric Intuition of Logistic Classifiers

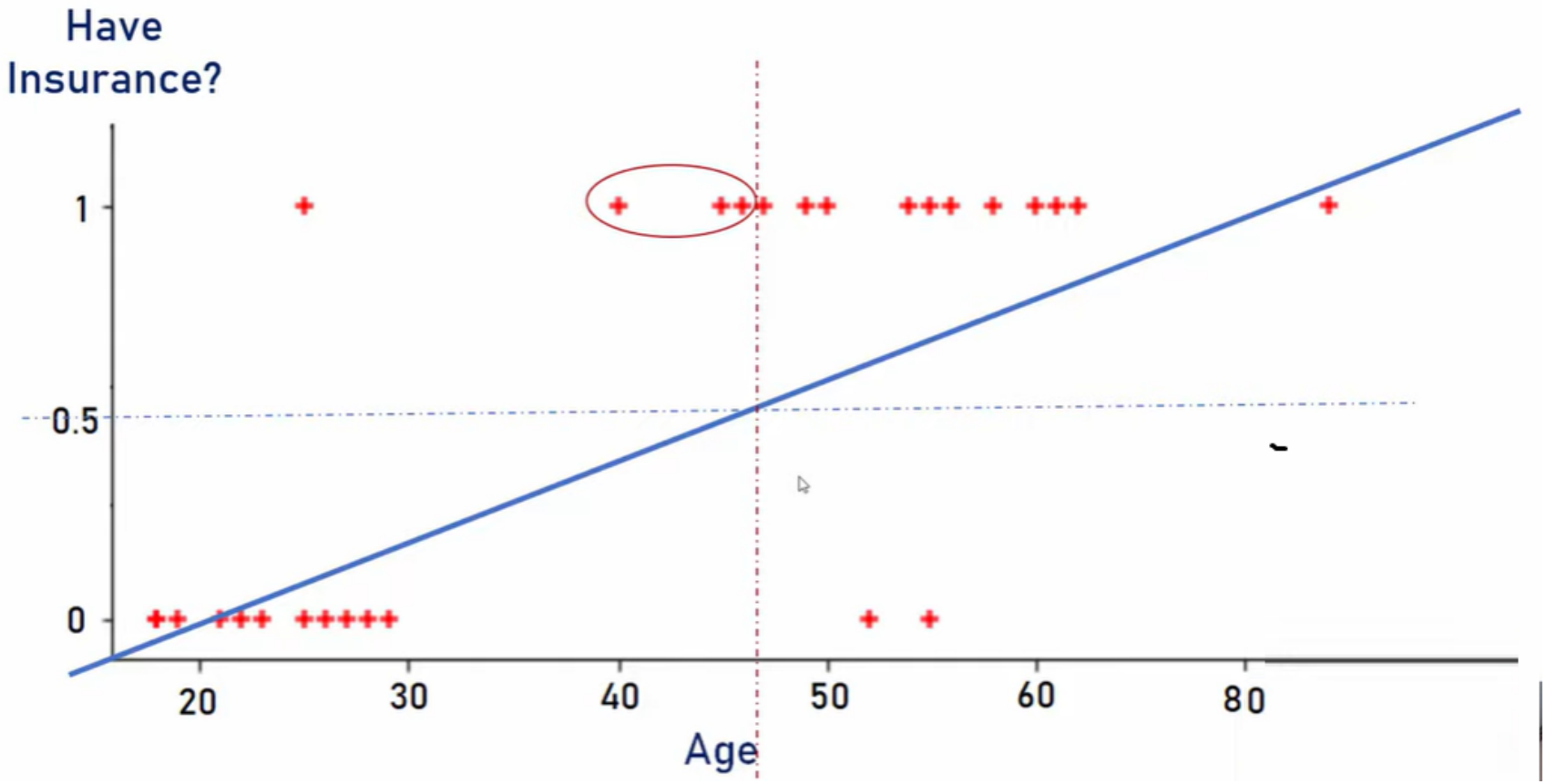
age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

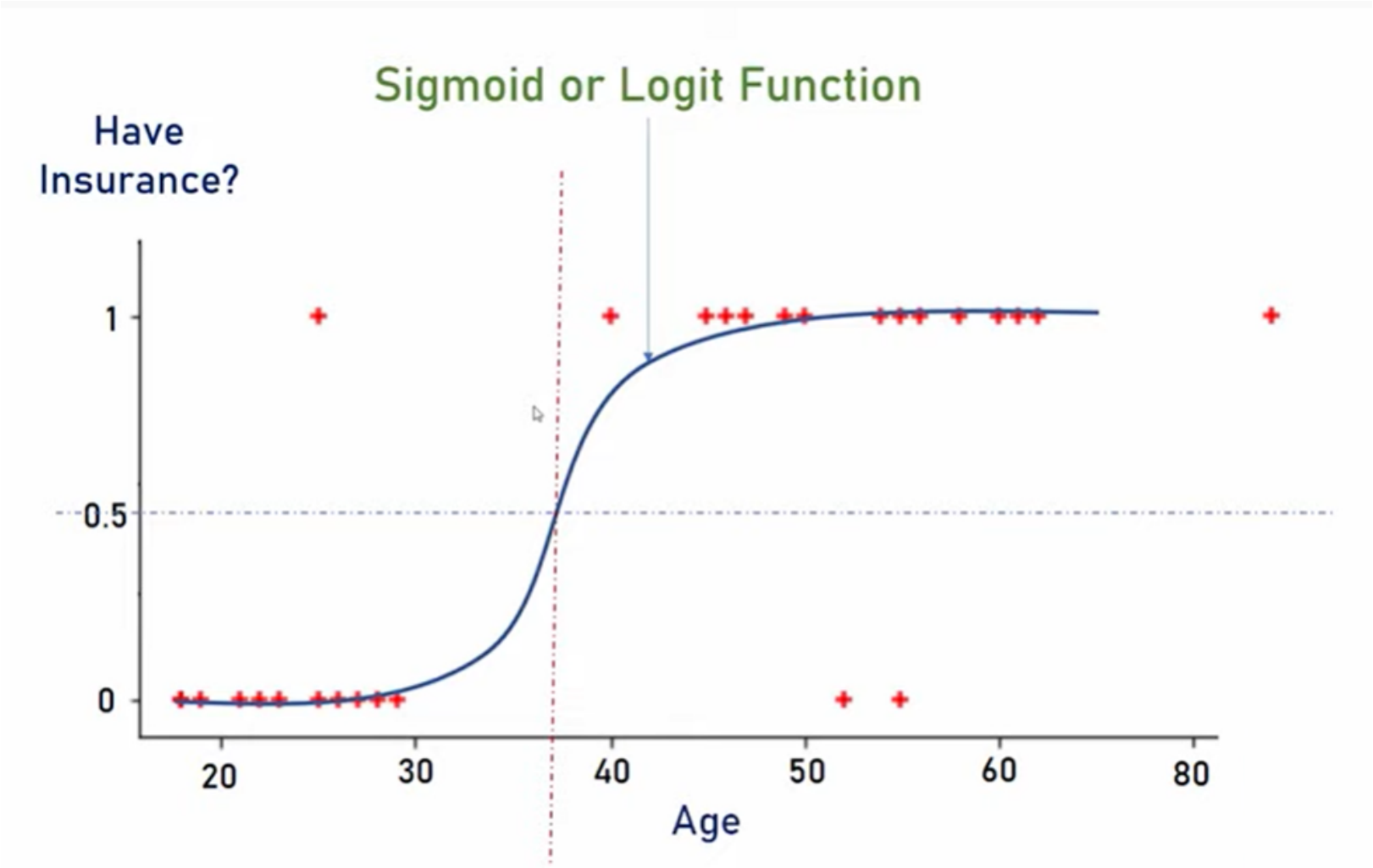
Binary Classification

Given an age of a person, come up with a **function** that can predict if person will buy insurance or not


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Sigmoid or Logit Function
Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

e = Euler's number ~ 2.71828

$sigmoid(200) = \frac{1}{1+2.71^{-200}}$ = almost close to 1

$sigmoid(-200) = \frac{1}{1+2.71^{200}}$ = almost close to 0

Sigmoid function converts input into range 0 to 1

Step 1

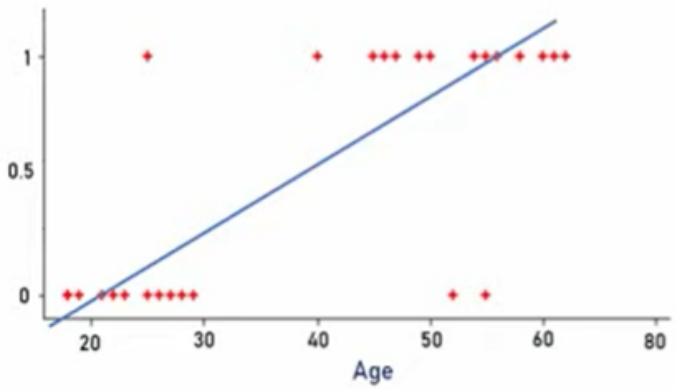
$$y = m * x + b$$

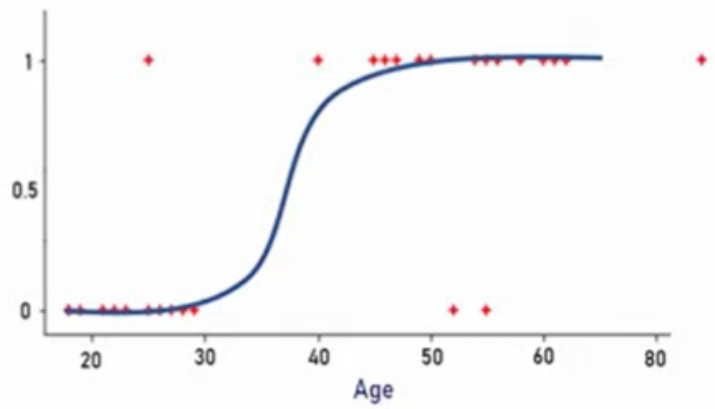
Age

Step 2

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

If person will buy insurance





value < 0.5 = person will not buy insurance

value >= 0.5 = person **will** buy insurance

Age = 35 → $y = 0.042 * x - 1.53$ | $z = \frac{1}{1 + e^{-y}}$ → 0.48

value < 0.5 = person will not buy insurance

value >= 0.5 = person **will** buy insurance

Age = 43 → $y = 0.042 * x - 1.53$ | $z = \frac{1}{1 + e^{-y}}$ → 0.57

$$y = 0.042 * x - 1.53$$

Age

$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 - 1.53$$

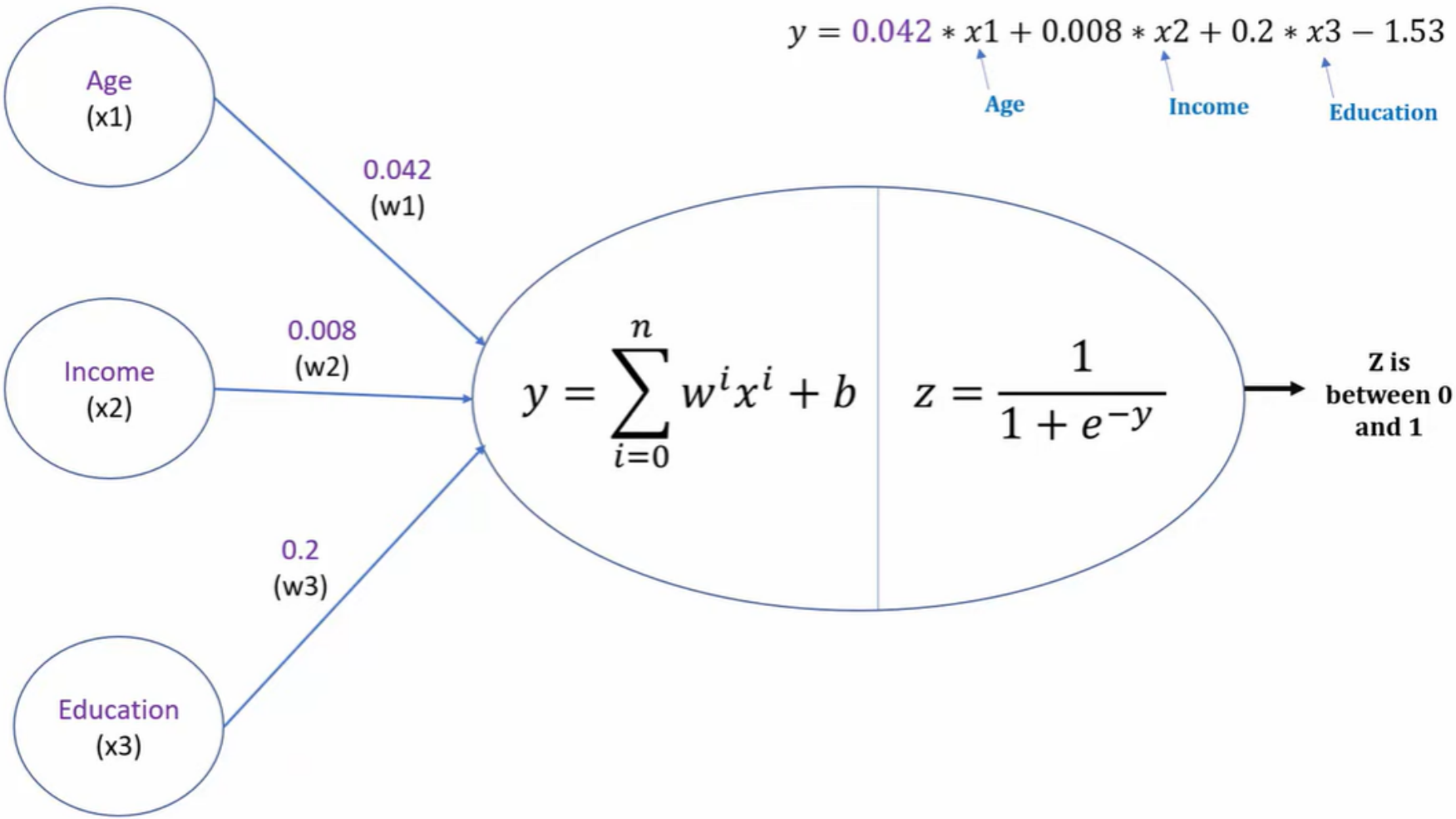
Age Income Education

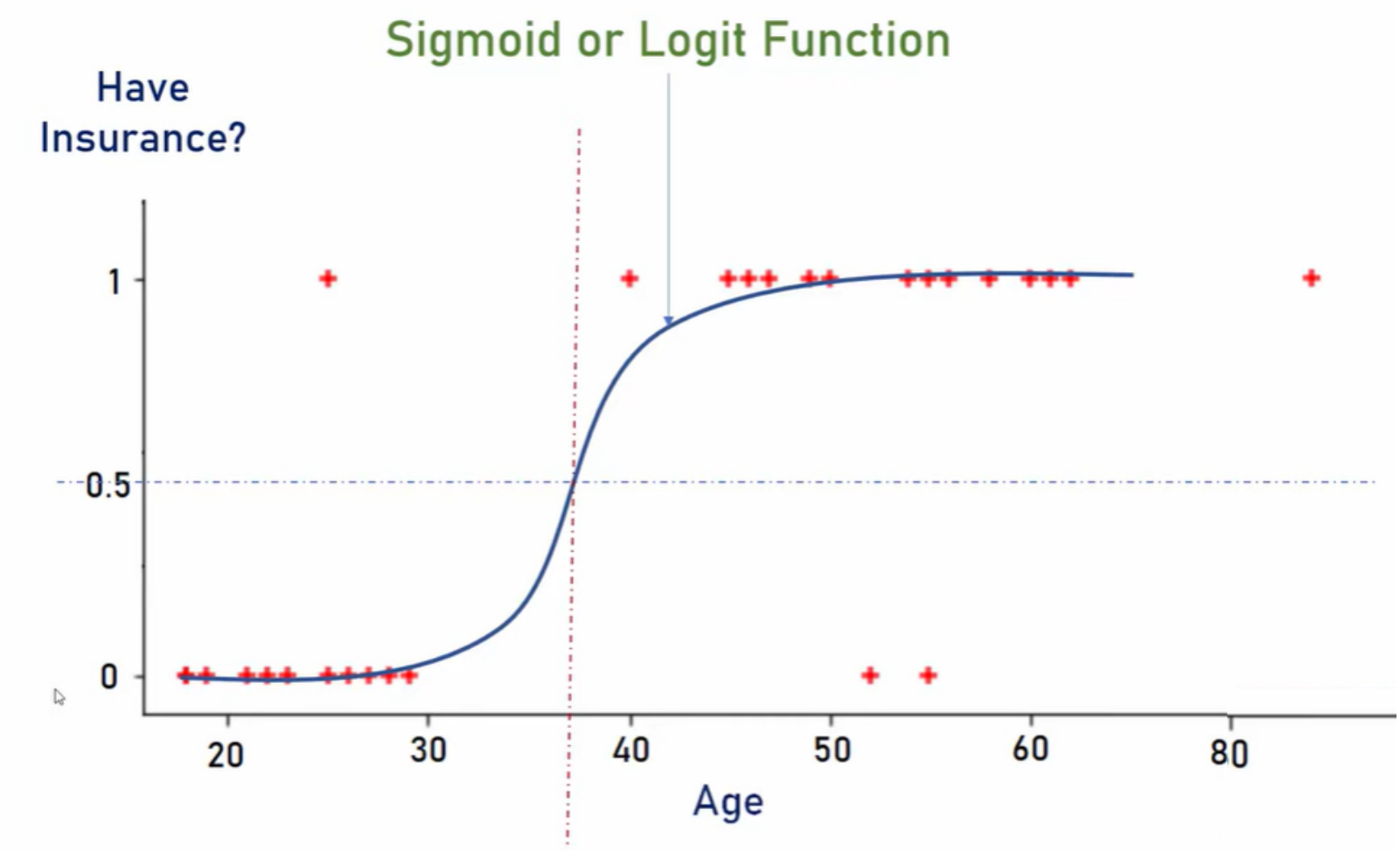
$$y = w1 * x1 + w2 * x2 + w3 * x3 + \boldsymbol{b}$$

$$y = \sum_{i=0}^{n} w^i x^i + b$$

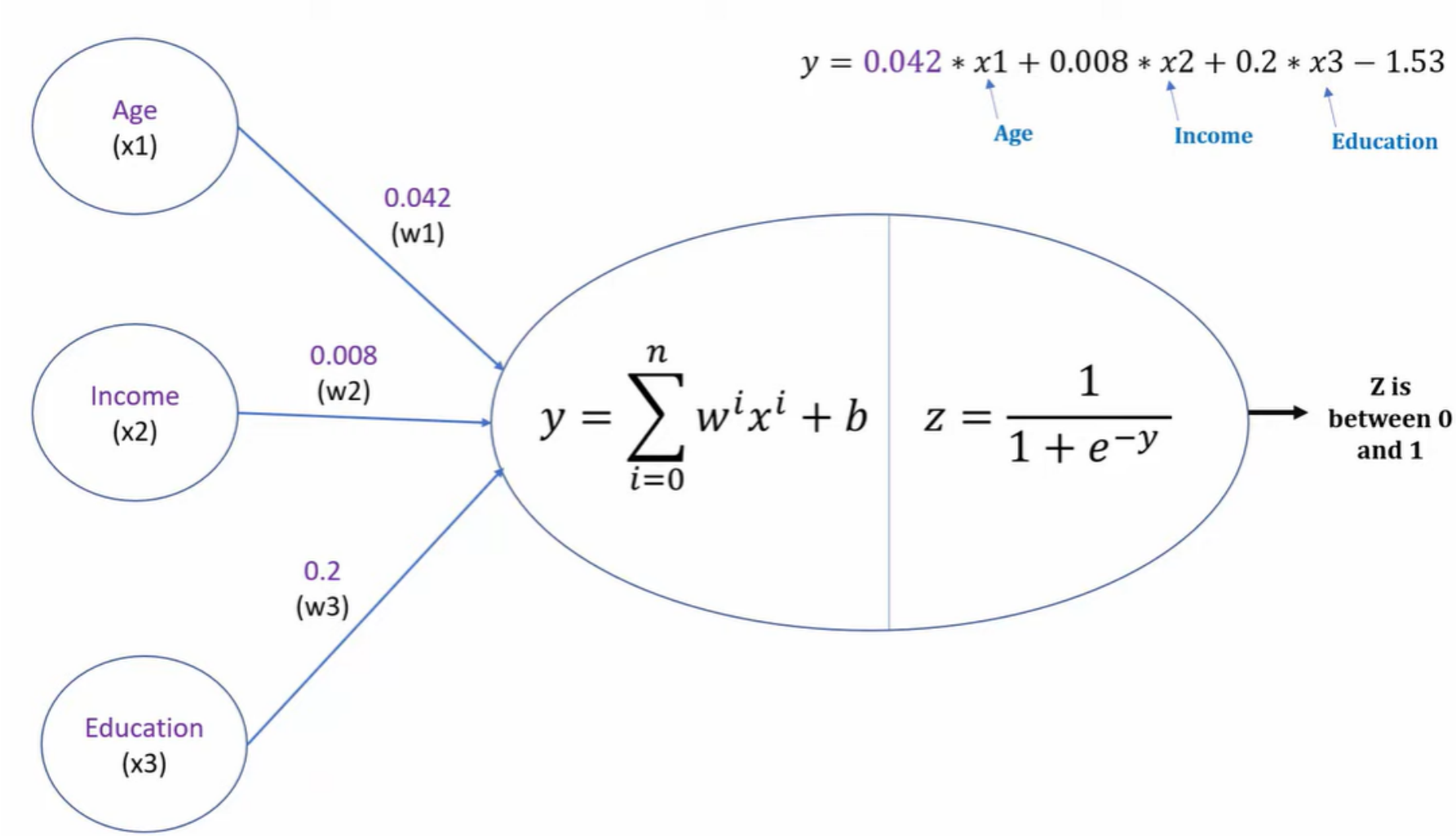
$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 - 1.53$$

Age Income Education




Sigmoid or Logit Function
Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

Neural Network vs. Logistic Classifiers



Assumption of Logistic Classifiers

- classes are almost/perfectly linearly separable
- independence of errors
- linearity in the logit for continuous variables
- absence of multicollinearity, and lack of strongly influential outliers

Python code: Logistic Regression Classifier

```
# data load
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()

# Convert data set to pandas data frame
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()

# See the categories of target variable
iris.target_names

# take a general notation
X = iris.data[:, :4] # we only take the first four features.
Y = iris.target
```

Python code: Logistic Regression Classifier

```
# Split dataset into training and test dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=0.3, random_state=1234)

# Create an instance of Logistic Regression Classifier and
    fit the data.
logreg = LogisticRegression(C=1e5)
logreg.fit(X, Y)

# See the accuracy rate
logreg.score(X_test,Y_test)
```

Python code: Logistic Regression Classifier

```
# predict for the test dataset and generate classification
    report
predictionsModel = logreg.predict(X_test)
print('Classification
    Report:\n',classification_report(Y_test,
    predictionsModel))
print('Confusion Matrix:\n',confusion_matrix(Y_test,
    predictionsModel))
print('Accuracy Score:',accuracy_score(Y_test,
    predictionsModel))
```

R Code: Binary Logistic Regression

```
## Binary Logistic Regression
iris2 <- iris
iris2$virginica <- iris$Species == "virginica"
fit_logisReg <- glm(virginica ~ Petal.Width + Petal.Length +
    Sepal.Length + Sepal.Width, data=iris2, family=binomial)
plot(fit_logisReg)
summary(fit_logisReg)
```

R Code: Multinational Logistic Regression

```
### Multinomial Logistic Regression
# install.packages("VGAM")
library(VGAM) #Load package VGAM
fit.MLR <- vglm( Species ~ Sepal.Length + Sepal.Width +
    Petal.Length + Petal.Width, family=multinomial,data=iris)
summary(fit.MLR)
```

R Code: Multinational Logistic Regression classification

```
#Perform classification
probabilities.MLR <- predict(fit.MLR, iris[,1:4],
    type="response")
predictions <- apply(probabilities.MLR, 1, which.max)
mtab=table(iris$Species, predictions)
iris$predictions[predictions==1] <- "setosa"
iris$predictions[predictions==2] <- "versicolor"
iris$predictions[predictions==3] <- "virginica"
mtab<-table(iris$predictions,iris$Species)
library(caret )
confusionMatrix(mtab)
```

Chapter 9: Support Vector Machine (SMV)

9 Chapter 9: Support Vector Machine (SMV)

9.1 Problem & Motivation

9.2 Basic Algebra

9.3 SVM: Basic Concept

9.4 The Support Vector Classifier

9.5 Optimal Separating Hyperplanes

9.6 Computing the Support Vector Classifier

9.7 Support Vector Machines and Kernels

9.8 Applications of SVM

9.9 Python Code: Support Vector Machine

9.10 R Code: Support Vector Machine

Problem & Motivation

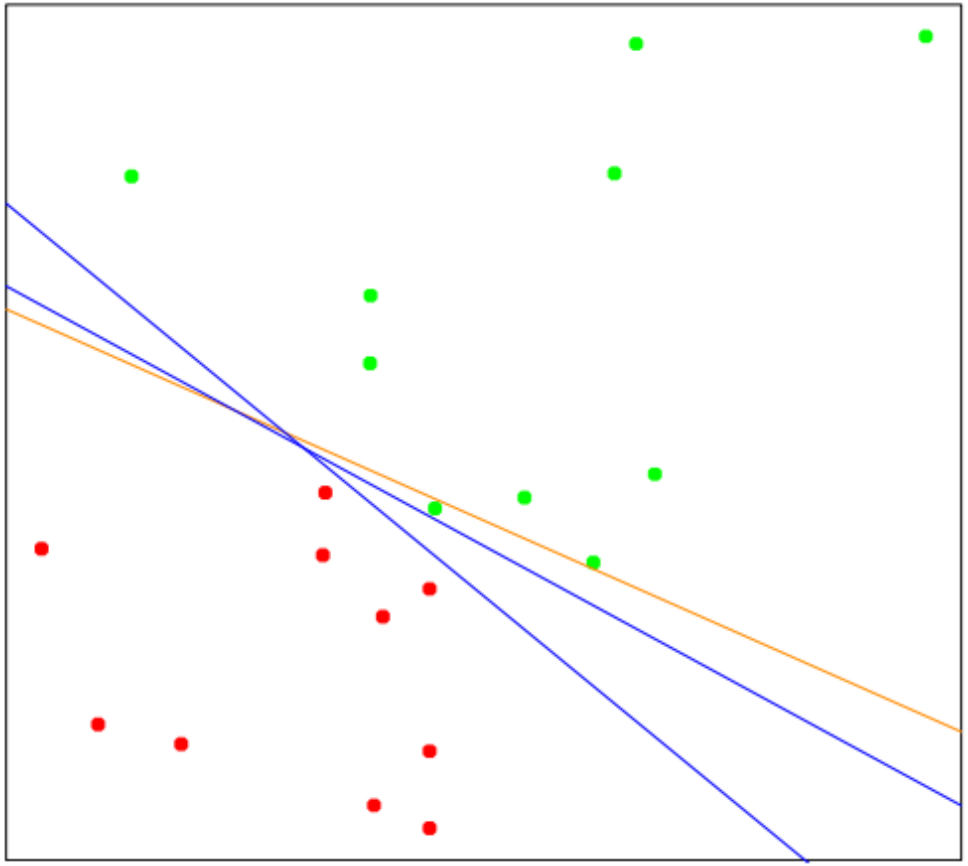
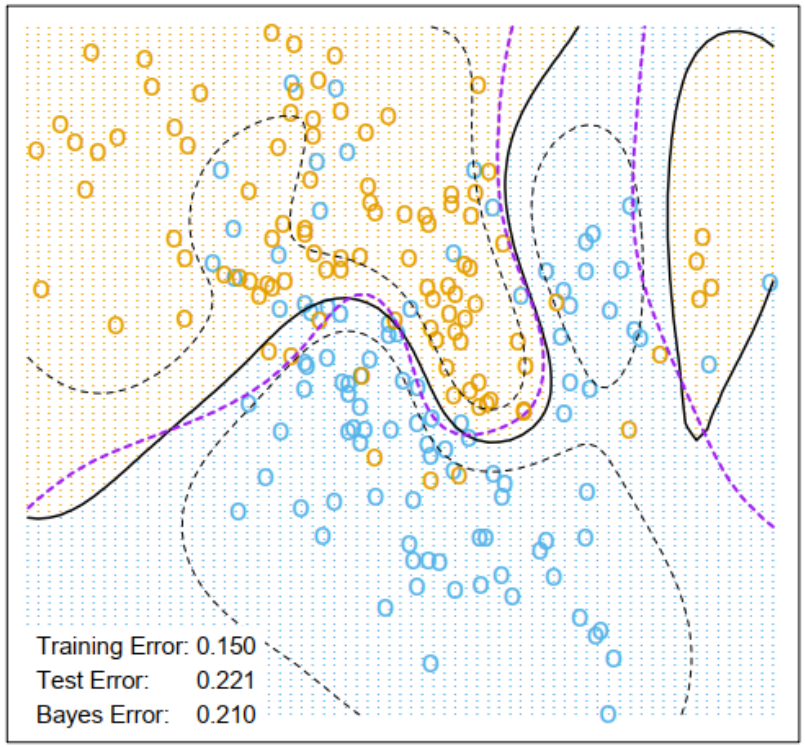


Figure: The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the perceptron learning algorithm with different random starts.

Problem & Motivation



- suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in
- goal: find an optimal separating hyperplane between two perfectly separated classes

Problem & Motivation

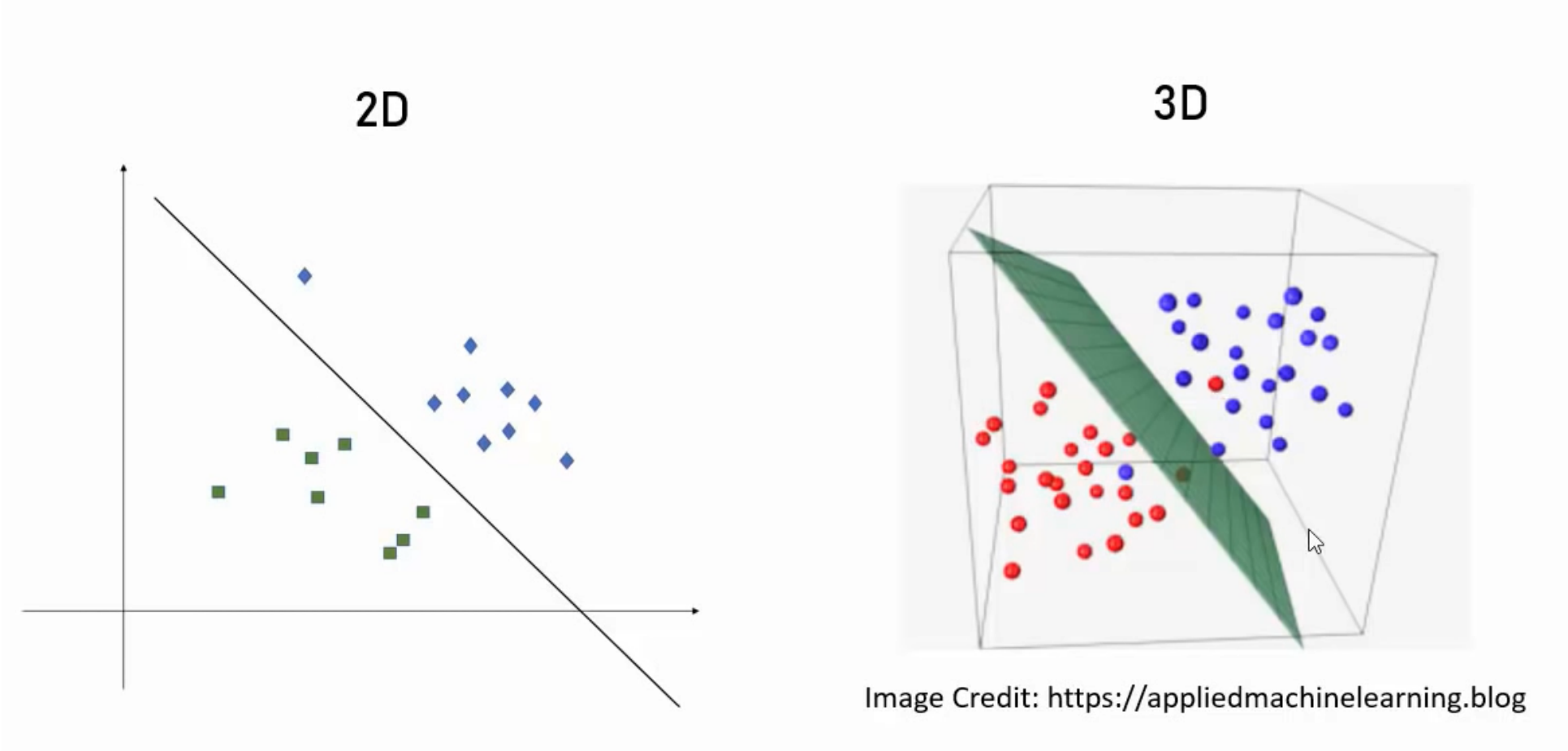


Image Credit: https://appliedmachinelearning.blog

Basic Algebra: Hyperplane & Separating Hyperplane

- a hyperplane in $\mathbb{R}^p$, for some constants $\beta_0, \ldots, \beta_p$ is the set of points $x^T = (x_1, \ldots, x_p)$ such that

$$\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = b$$

  where b is an arbitrary constant
- if $p = 2$ then it is a line and if $p = 3$ then it is a plane
- a hyperplane $\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = b$ is said to be a separating hyperplane just in case

$$\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} > b \quad \text{if} \quad y_i = 1$$

$$\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} < b \quad \text{if} \quad y_i = -1$$

  in other words,

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) > b \quad \text{for each} \quad i = 1, \ldots, n$$

Basic Algebra: Hyperplane & Separating Hyperplane

- it can be shown that the perpendicular distance between x_i and the separating hyperplane is

$$\frac{1}{\|\beta\|} y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$$

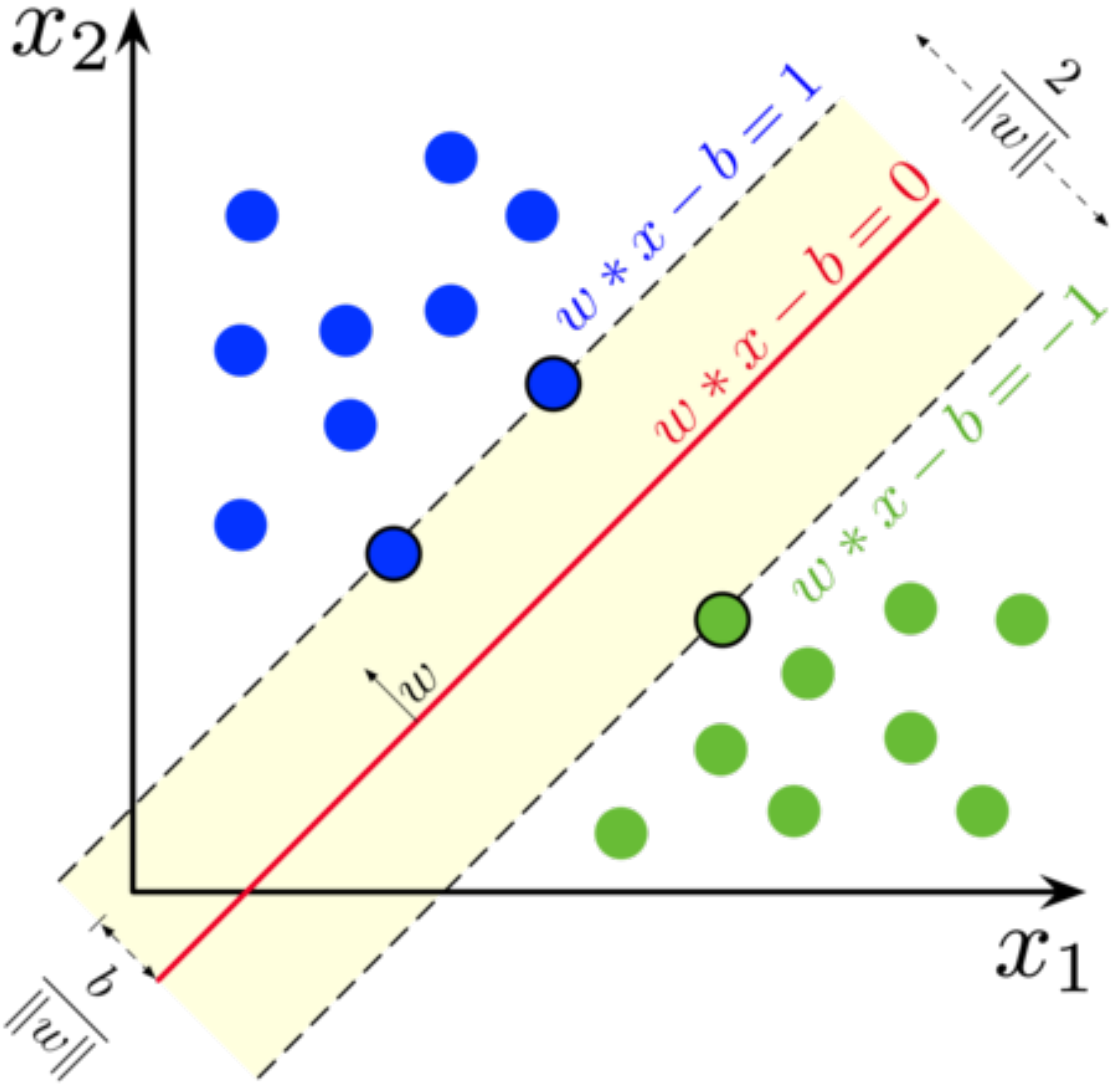
where $\beta = (\beta_1, \ldots, \beta_p)^T$

- for a fixed hyperplane $\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = b$,

$$\min_{i \in \{1,\ldots,n\}} \left\{ \frac{1}{\|\beta\|} y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \right\}$$

is called a margin

SVC: Support Vectors



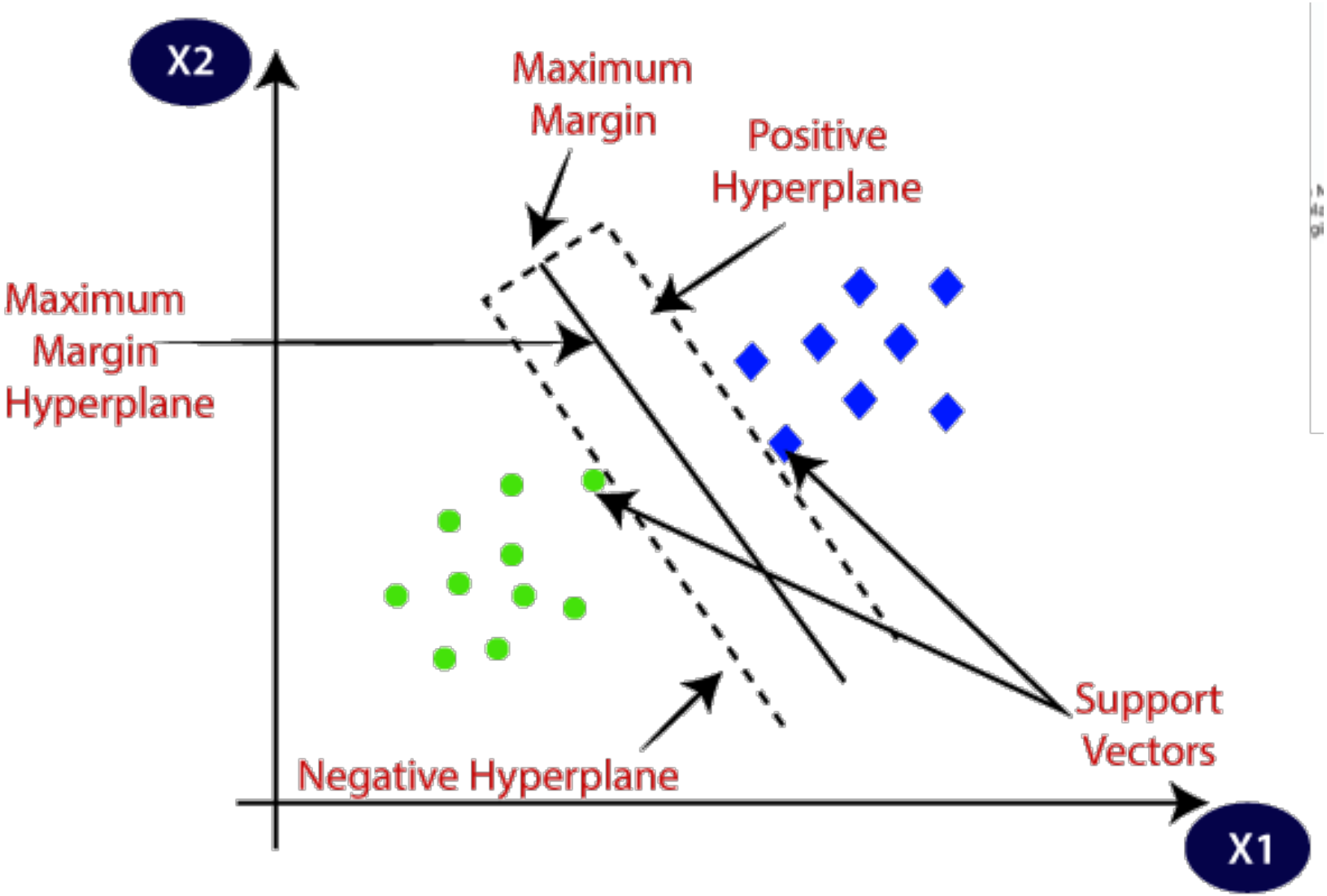
SVM: Basic Concept

- support-vector machines (also called support-vector networks, support-vector classifiers) are supervised learning models
- with associated learning algorithms that analyze data used for classification and regression analysis
- it presents one of the most robust prediction methods, based on the statistical learning framework
- an SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible
- new data are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall

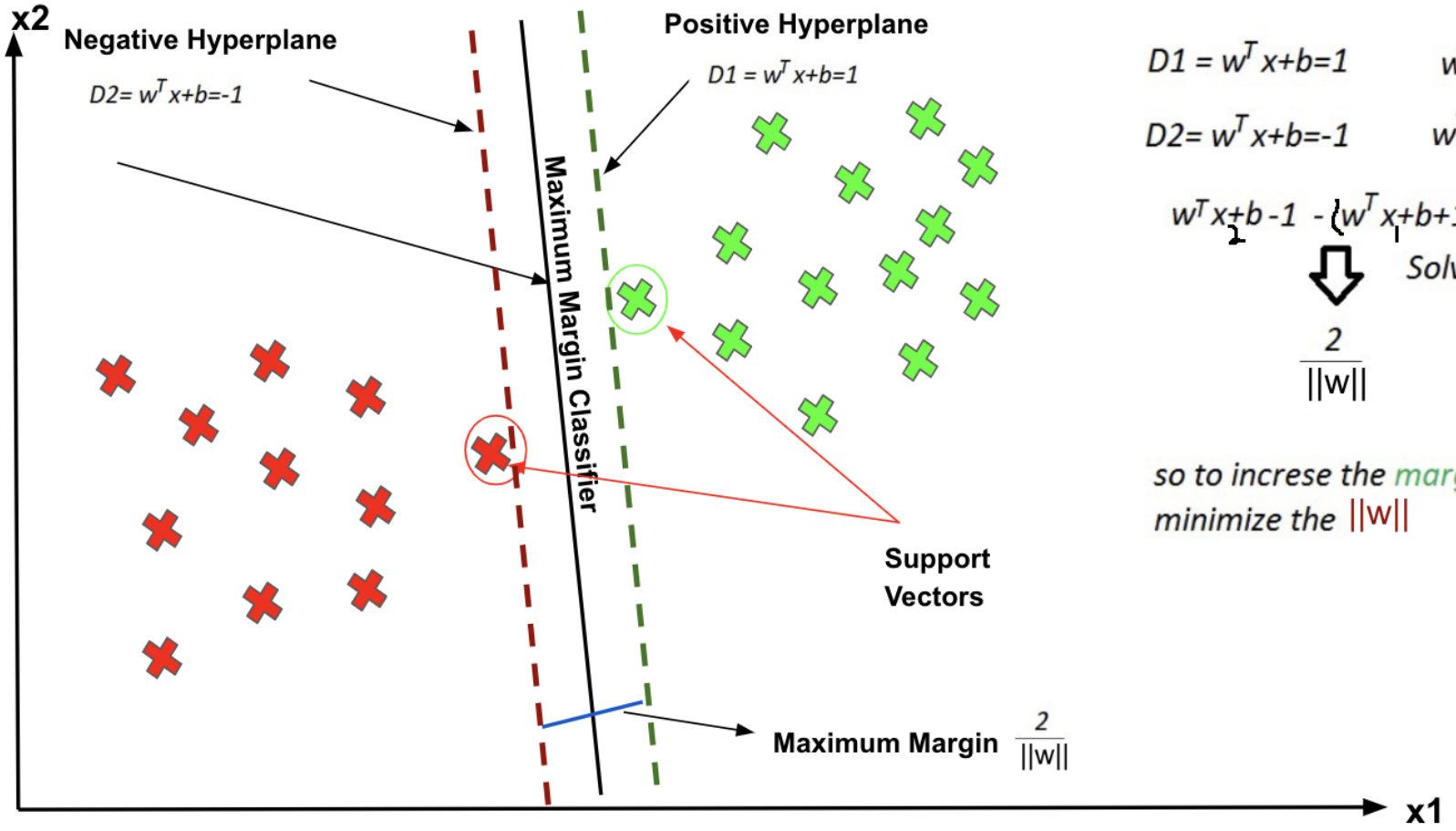
SVM: Basic Concept

- a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane
- this is called a linear classifier
- linear classification: kernel trick, implicitly mapping their inputs into high-dimensional feature spaces
- maximum-margin hyperplane: choose the separating hyperplane so that the distance from it to the nearest data point on each side is maximized
- maximum-margin classifier: the linear classifier
- a SVM constructs a separating hyperplane or set of separating hyperplanes in a high- or infinite-dimensional space

SVM: Basic Components



Main Concept of SVM



$D1 = w^T x+b=1$ $\quad$ $w^T x+b-1=0$

$D2= w^T x+b=-1$ $\quad$ $w^T x+b+1=0$

$w^T x_2+b-1 - (w^T x_1+b+1)$

$\Downarrow$ Solve algebraically

$\frac{2}{||w||}$

so to increse the margin, minimize the $||w||$ $\quad$ $\frac{1}{2}||w||^2$

Optimal Separating Hyperplanes

- a linearly separable
- optimal separating hyperplane provide a unique solution to the separating hyperplane problem by maximizing the margin between the two classes on the training data, i.e.,

$$(\widehat{\beta_0}, \widehat{\beta}) = \arg \max_{(\beta_0, \beta)} \min_{i \in \{1, \ldots, n\}} \left\{ \frac{1}{\|\beta\|} y_i (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \right\}$$

solving this optimization problem is difficult, in particular due to the scaling by the inverse of $\|\beta\|$, which makes the problem nonconvex optimization problem

Optimal Separating Hyperplanes

- a linearly separable
- a possible alternative would be canonical hyperplane by fixing the numerator and in this case, the optimization problem can be modified:

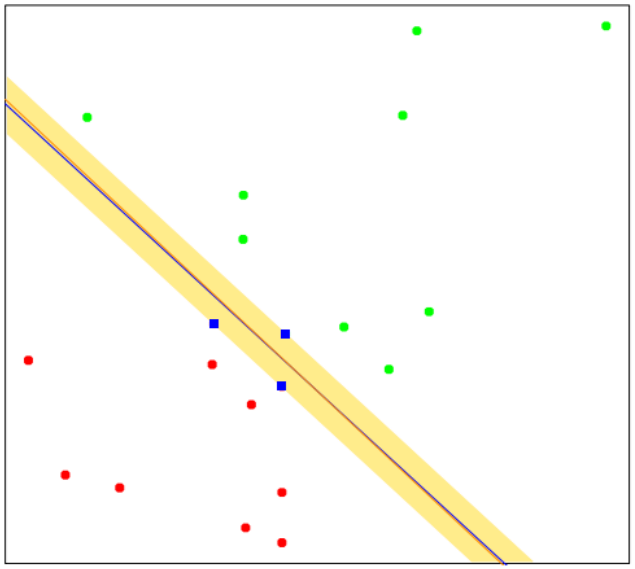
$$\min_{(\beta_0,\beta)} \|\beta\|$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, i = 1, \ldots, n$$

this is a convex optimization problem (quadratic criterion, linear inequality constraints)

Optimal Separating Hyperplanes

- optimal separating hyperplane provide a unique solution to the separating hyperplane problem by maximizing the margin between the two classes on the training data

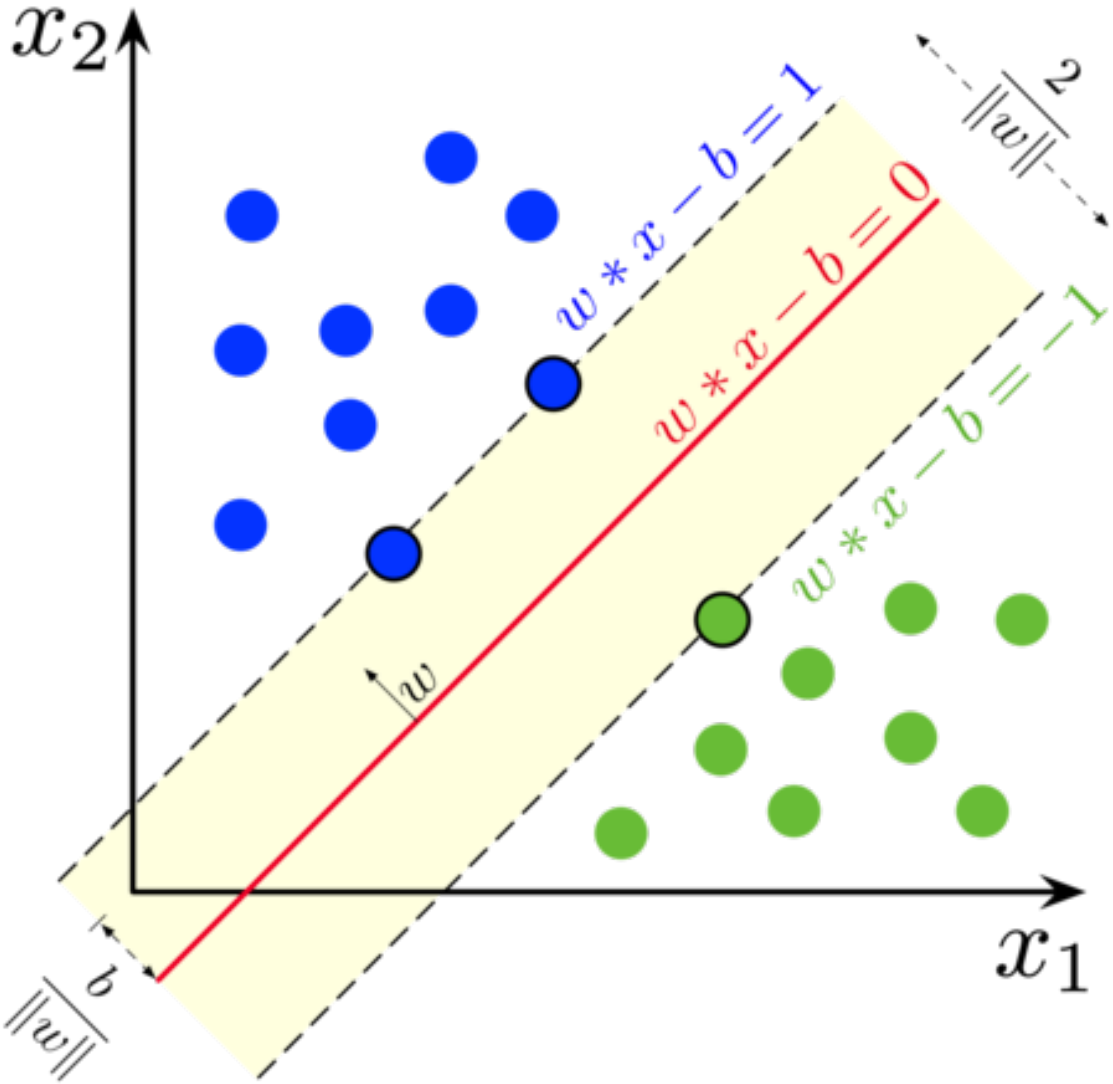


- the shaded region delineates the maximum margin separating the two classes
- there are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab

The Support Vector Classifier (SVC)

- optimal separating hyperplanes for the case when two classes are linearly separable
- support vector machine is a generalization of optimal separating hyperplanes
  - which is an extensions to the nonseparable case, where the classes overlap
  - where the classes may not be separable by a linear boundary
- support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane
  - using these support vectors, we maximize the margin of the classifier
  - deleting the support vectors will change the position of the hyperplane
  - these are the points that help us build our SVM

SVC: Support Vectors



SVC: Nonseparable Case

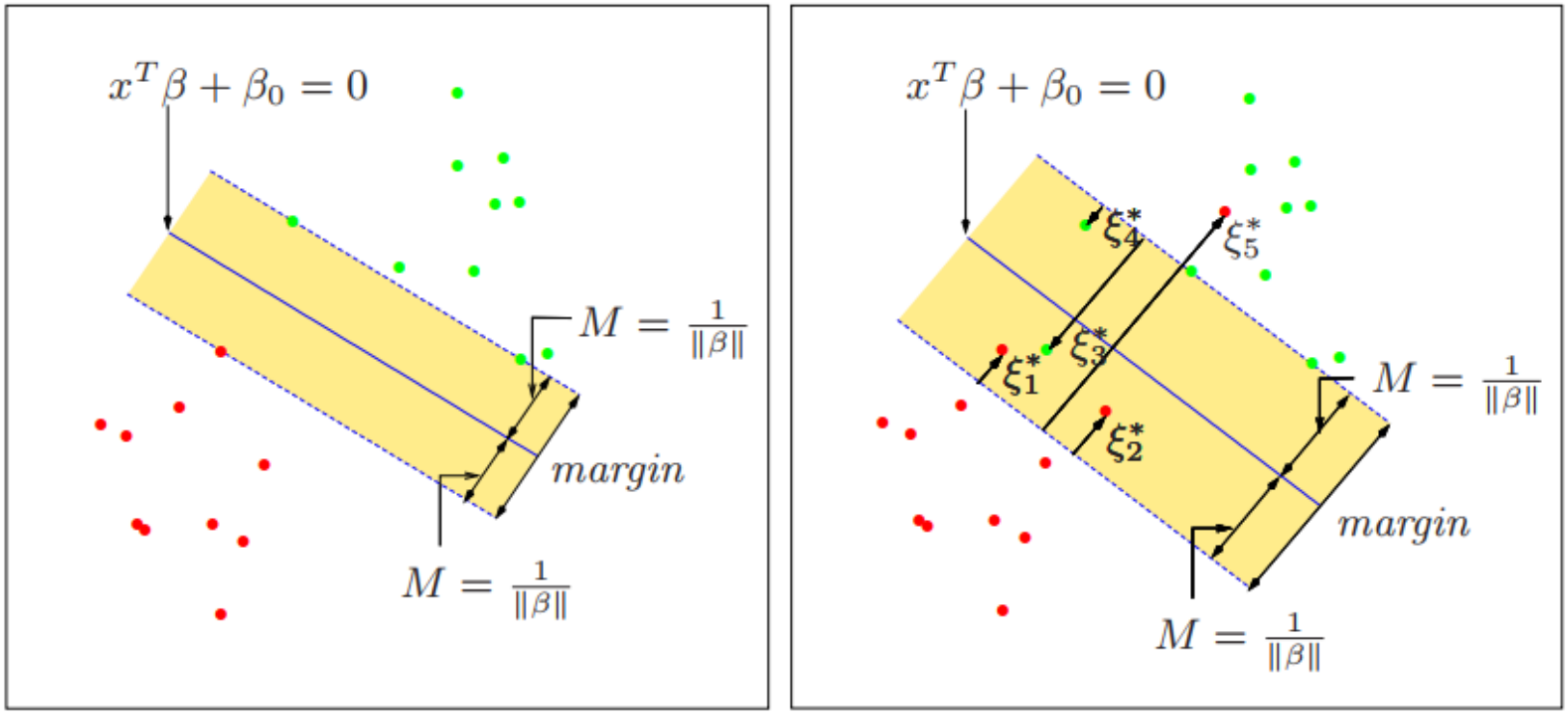
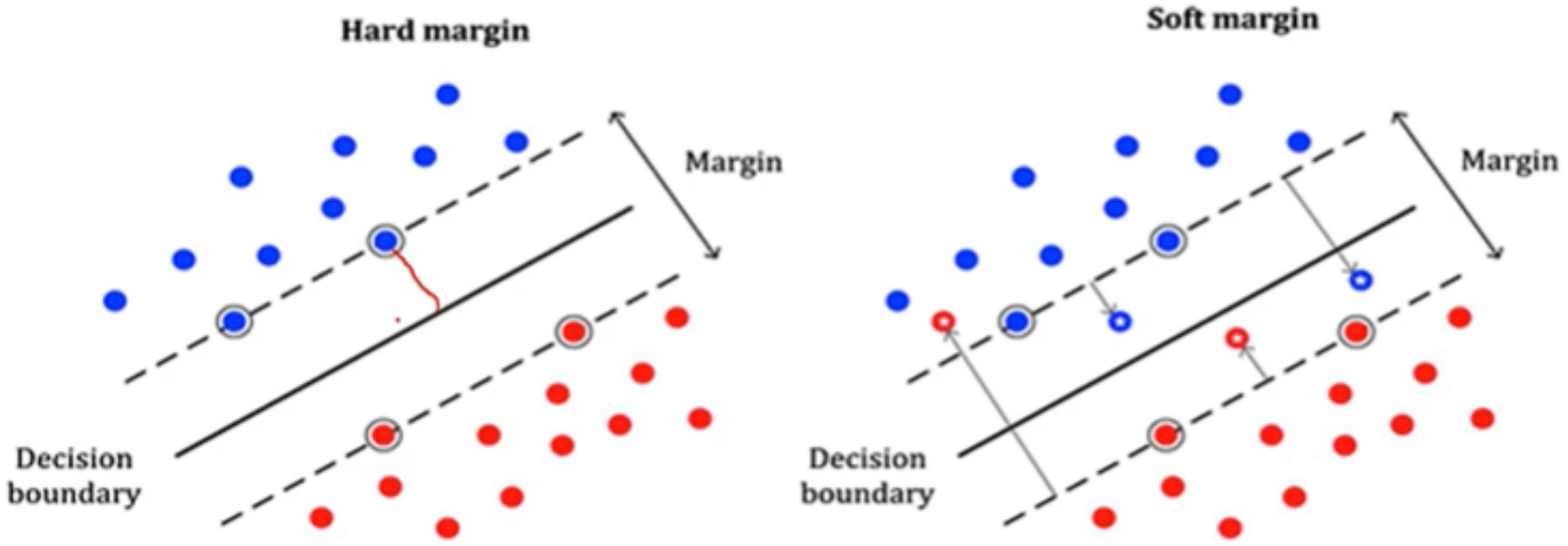


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Hard margin SVM aims to find a hyperplane that perfectly separates the data into two classes without any misclassification, while soft margin SVM allows some misclassification by introducing slack variables. Geometrically, the hard margin SVM looks for the widest margin possible, while the soft margin SVM finds the hyperplane that maximizes the margin while minimizing the penalty for misclassification.

Hard Margin vs Soft Margin



SVC: Nonseparable Case

- suppose now that the classes overlap in feature space
- one way to deal with the overlap is to still maximize $M = \frac{1}{\|\beta\|}$, but allow for some points to be on the wrong side of the margin
  - define slack variable
$$\xi = (\xi_1, \ldots, \xi_n)$$
  - the optimization problem:
$$\max_{\beta_0, \beta, \|\beta\|=1} M \text{ subject to } \begin{cases} y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i); i = 1, \ldots, n \\ \forall i, \xi_i \geq 0, \sum_{i=1}^n \xi_i \leq \text{constant} \end{cases} \quad (7)$$
  it is convex which leads to the "standard" support vector classifier
- miss-classifications occur when $\xi_i > 1$

SVC: Nonseparable Case

- since $M = \frac{1}{\|\beta\|}$ and we can drop the norm constraint on β then problem (7) can be written as
- the optimization problem:

$$\min_{\beta_0,\beta} \|\beta\| \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq (1 - \xi_i); i = 1, \ldots, n \\ \forall i, \xi_i \geq 0, \sum_{i=1}^n \xi_i \leq \text{constant} \end{cases}$$

  this problem is quadratic with linear inequality constraints, hence it is a convex optimization problem
- given the solutions $\widehat{\beta}_0$ and $\widehat{\beta}$, the decision function can be written as

$$\widehat{G}(x) = \text{sign}[x^T\widehat{\beta} + \widehat{\beta}_0]$$

Computing the SVC (see, Hastie et al. (2009))

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \tag{12.8}$$
$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i,$$

where the "cost" parameter C replaces the constant in (12.7); the separable case corresponds to $C = \infty$.

The Lagrange (primal) function is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i, \tag{12.9}$$

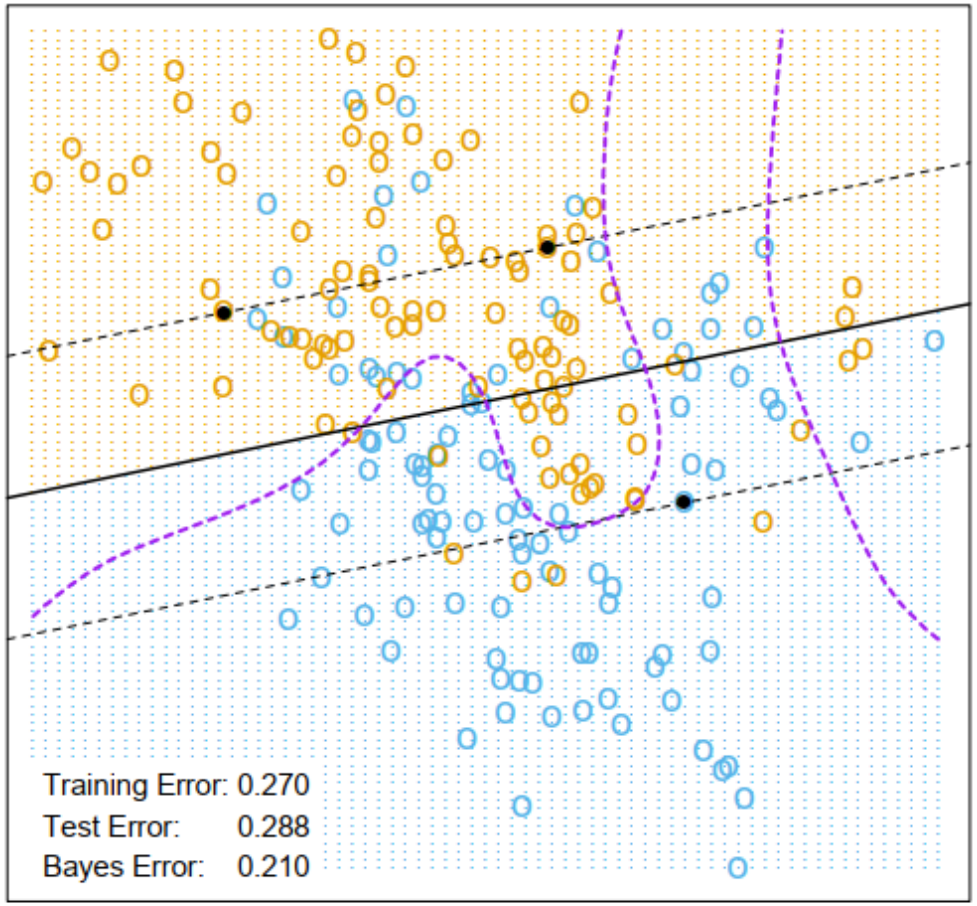
which we minimize w.r.t β, β_0 and ξ_i. Setting the respective derivatives to zero, we get

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i, \tag{12.10}$$
$$0 = \sum_{i=1}^{N}\alpha_i y_i, \tag{12.11}$$
$$\alpha_i = C - \mu_i,\ \forall i, \tag{12.12}$$

as well as the positivity constraints $\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i$. By substituting (12.10)–(12.12) into (12.9), we obtain the Lagrangian (Wolfe) dual objective function

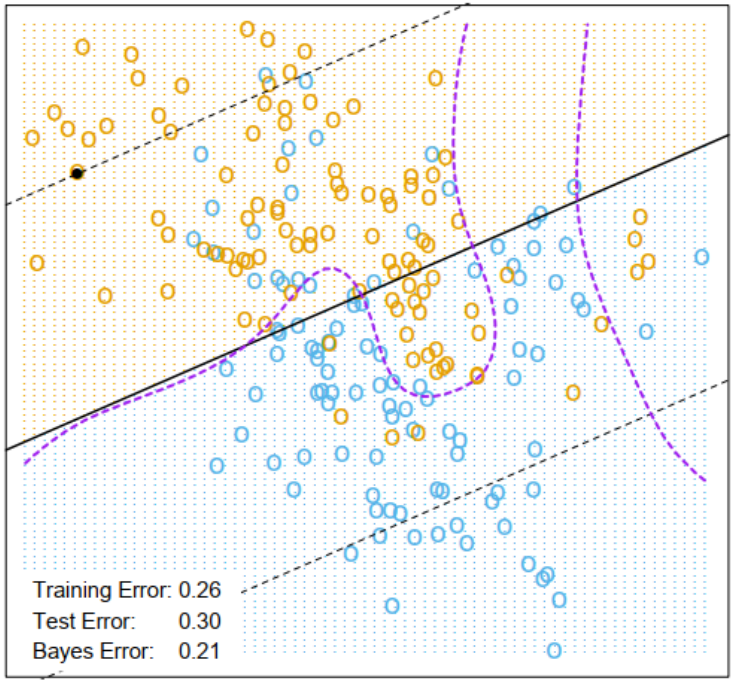
$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_i y_{i'}x_i^T x_{i'}, \tag{12.13}$$

SVC: Mixture Example(see, Hastie et al. (2009))



$C = 10000$

SVC: Mixture Example (see, Hastie et al. (2009))



$C = 0.01$

FIGURE 12.2. *The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C. The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0$, $\alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.*

Support Vector Machines and Smoothing Function

- the support vector classifier described so far finds linear boundaries in the input feature space
- other linear method such as polynomials or smoothing splines methods make the more flexible method by enlarging the feature space
- idea SVM: enlarge feature space using basis smoothing function expansions

$$h_m(x), m = 1, \ldots, M$$

- translates to nonlinear boundaries in the original space
- fit SVC using input features

$$h(x_i) = (h_1(x_i), \ldots, h_M(x_i)); i = 1, \ldots, n$$

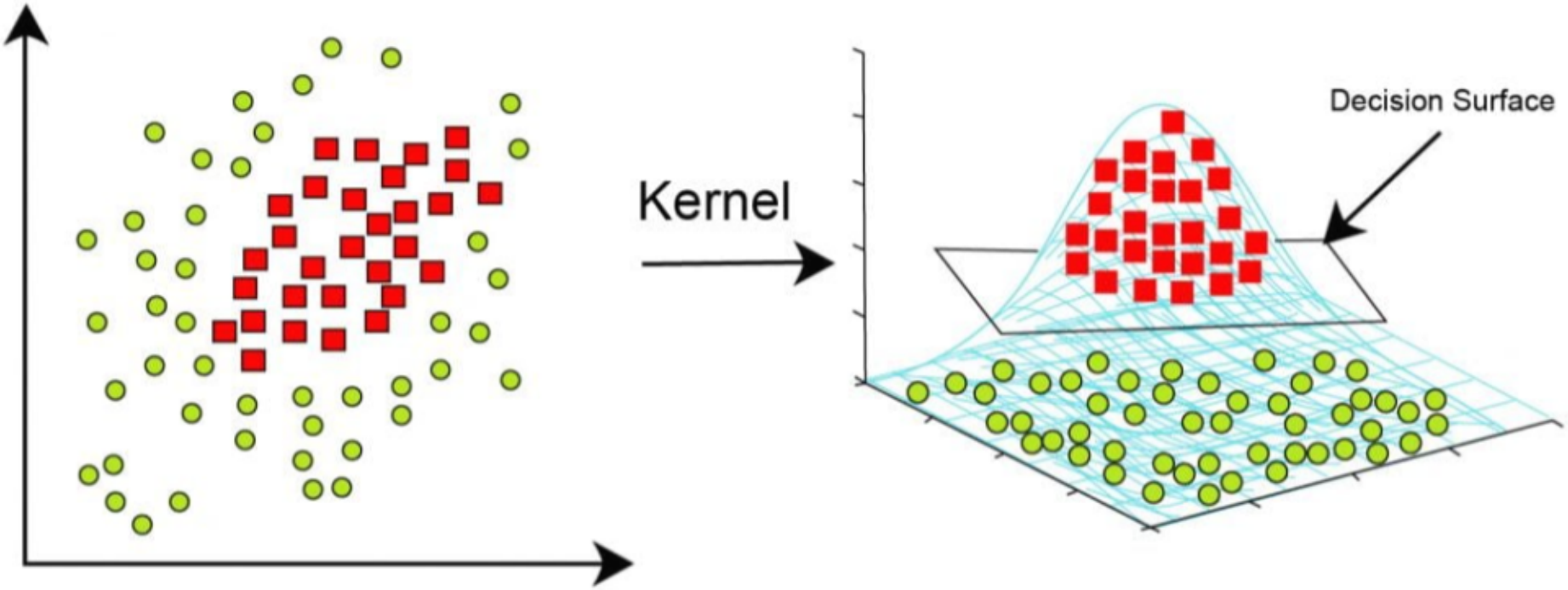
producing the (nonlinear) function

$$\widehat{f}(x) = h(x)^T \widehat{\beta} + \widehat{\beta}_0$$

and classifier

$$\widehat{G}(x) = \text{sign}(\widehat{f}(x))$$

Support Vector Machines and Kernels



Support Vector Machines and Kernels

- support vector machine SVM: M very large
- solution can be written as

$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i y_i K(x, x_i) + \widehat{\beta}_0$$

with $\alpha_i = \gamma - \mu_i$ (μ_i certain Lagrange multipliers) and kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \text{ where } \langle \cdot, \cdot \rangle \text{ the inner product}$$

- kernel choices
  - dth degree polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$
  - radial basis: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$
  - neural net work: $K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

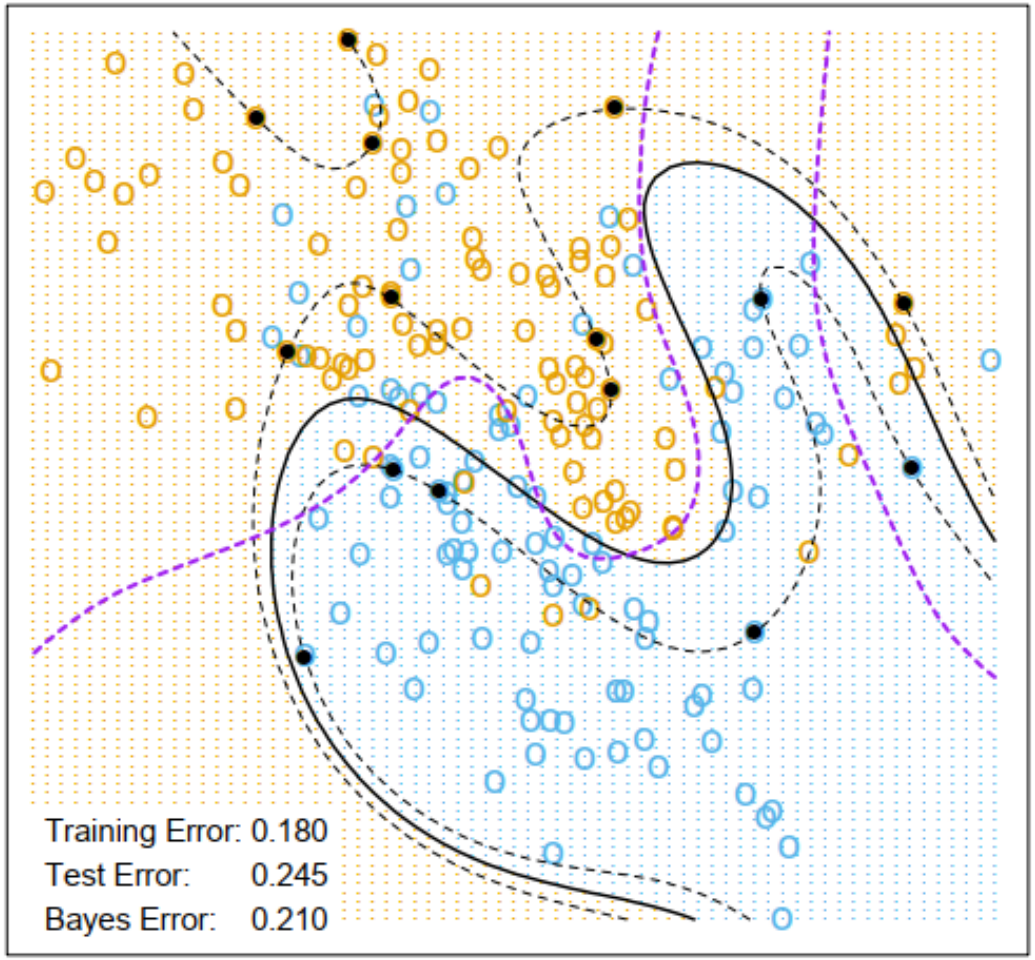
small value of $\gamma \to$ smoother boundary

for more information:

`https://data-flair.training/blogs/svm-kernel-functions/`

Nonlinear SVMs - Example

Figure: SVM - Degree-4 Polynomial in Feature Space



Nonlinear SVMs - Example

Figure: SVM - Radial Kernel in Feature Space

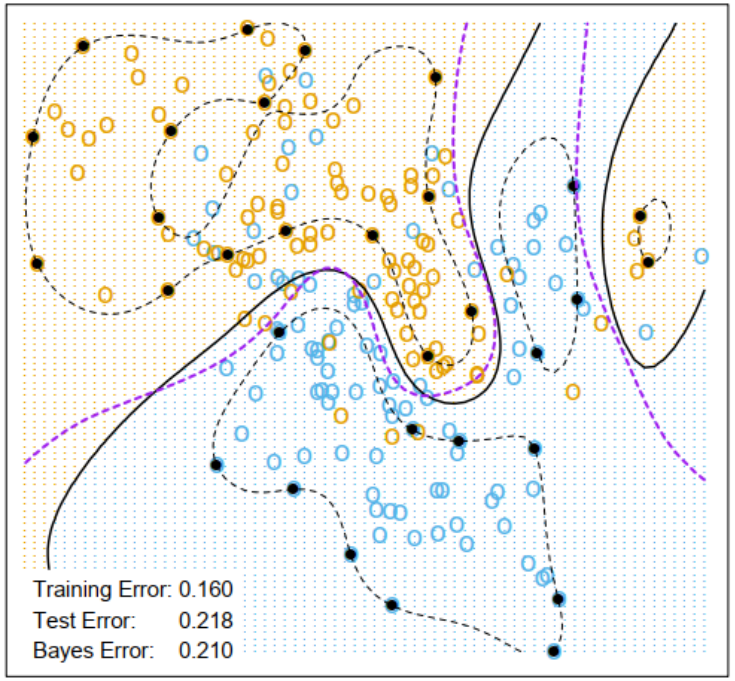


FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*

Important Parameters

value of C (C-classification and ν-classification):

- C is the penalty value that penalizes the algorithm when it tries to maximize the margins and causes misclassification— thus, this value manages the trade-off between maximization of margin and misclassification
  — a lower value of C means that a small margin is created, thus minimizing the error, while a large value of C means a large margin is created that can invite misclassification
  — and here medium C giving is the better model
  — C ranges from 0 to ∞ and can be a bit hard to estimate and use
  — a modification to this was the introduction of ν which operates between $0-1$ and represents the lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane

Important Parameters

value of γ:

- a small gamma value means that only a small number of data points close to the plausible margin line are considered making the model underfit
- on the contrary, a large value of gamma causes a large number of even distant data points to be involved in calculating the separation line that can cause the model to overfit
- γ is used when we use the Gaussian RBF kernel. if you use linear or polynomial kernel then you do not need gamma only you need C hypermeter
  — somewhere it is also used as σ because σ and γ are related

Advantages of Support Vector Machines

Advantages

- High Dimensionality: SVM is an effective tool in high-dimensional spaces, which is particularly applicable to document classification and sentiment analysis where the dimensionality can be extremely large — with the help of Kernel Trick
- Memory Efficiency: uses a subset of training points in the decision function (called support vectors)
- Versatility
  — SVM can be used for linearly separable as well as non-linearly separable data

Disadvantages

- Kernel Parameters Selection
- Non-Probabilistic: SVM doesn't directly provide probability estimates
- SVM is incapable of handling text structures
- choice of the kernel is perhaps the biggest limitation of the support vector machine

Applications of SVM

- Face Detection
  — SVMs are capable of classifying images of persons in an environment by creating a square box that separates the face from the rest
- Text and hypertext categorization
  — SVMs can be used for document classification in the sense that it performs the text and hypertext categorization
- Bioinformatics
  — in the field of bioinformatics, SVMs are used for protein and genomic classification
  — they can classify the genetic structure of the patients based on their biological problems
- Handwriting recognition
- ...

Python Code: SVM

```
# data load
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()
# See the feature of the data set
iris.feature_names
# Convert data set to pandas data frame
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
# See the target variable
df['target']=iris.target
df.head()
# See the categories of target variable
iris.target_names
# take a general notation
X = iris.data[:, :4] # we only take the first four features.
Y = iris.target
# Split dataset into training and test dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=0.3, random_state=1234)
```

Python Code: SVM

```
# write function for Support vector machine
from sklearn.svm import SVC
model=SVC()
# model=SVC(C=10)
# model fitting
model.fit(X_train,Y_train)
# See the accuracy rate
model.score(X_test,Y_test)
# predict for the test dataset and generate classification
    report
predictionsModel = model.predict(X_test)
print('Classification
    Report:\n',classification_report(Y_test,
    predictionsModel))
print('Confusion Matrix:\n',confusion_matrix(Y_test,
    predictionsModel))
print('Accuracy Score:',accuracy_score(Y_test,
    predictionsModel))
```

Exercise

Train SVM classifier using skearn digits dataset (i.e. `from sklearn.datasets import load_digits`) and then

1. Measure accuracy of your model using different kernels such sas RBF and linear.
2. Tune your model further using regularization and gamma parameter and try to come up with highest accuracy score
3. Use 80% of samples as training data size

R Code: Support Vector Machine (SVM)

```
# rm(list=ls());
library(kernlab)    # SVM methodology
library(e1071)      # SVM methodology
data(iris)
attach(iris)
## classification mode
# default with factor response:
model <- svm(Species ~ ., data = iris)
# alternatively the traditional interface:
x <- subset(iris, select = -Species)
y <- Species
model <- svm(x, y)
print(model)
summary(model)
# test with train data
pred <- predict(model, x)
# (same as:)
pred <- fitted(model)
# Check accuracy:
mtab=table(pred, y)
mtab
# install.packages('caret', dependencies=TRUE)
library(caret )
confusionMatrix(mtab)
```

R Code: Support Vector Machine

```
# transform iris data to binary problem, and scale data
setosa <- as.factor(iris$Species == "setosa")
iris2 = scale(iris[,-5])

# fit binary C-classification model
m <- svm(setosa ~ Petal.Width + Petal.Length,
data = iris2, kernel = "linear")

# plot data and separating hyperplane
plot(Petal.Length ~ Petal.Width, data = iris2, col = setosa)
(cf <- coef(m))
abline(-cf[1]/cf[3], -cf[2]/cf[3], col = "red")

# plot margin and mark support vectors
abline(-(cf[1] + 1)/cf[3], -cf[2]/cf[3], col = "blue")
abline(-(cf[1] - 1)/cf[3], -cf[2]/cf[3], col = "blue")
points(m$SV, pch = 5, cex = 2)
```

R Code: Support Vector Machine

```
## simple example using the spam data set
data(spam) # available in library(kernlab)
## create test and training set
## Make training and test data sets.
index <- sample(2, nrow(spam), replace=TRUE, prob=c(0.7,
    0.3))
spamtrain <- spam[index==1,]
spamtest <- spam[index==2,]
## train a support vector machine
filter <- ksvm(type~.,data=spamtrain,kernel="rbfdot",
kpar=list(sigma=0.05),C=5,cross=3)
filter
## predict mail type on the test set
mailtype <- predict(filter,spamtest[,-58])
## Check results
mtab=table(mailtype,spamtest[,58])
mtab
confusionMatrix(mtab)
```

R Code: Support Vector Machine

```
data(iris)
## Create a kernel function using the build in rbfdot
    function
rbf <- rbfdot(sigma=0.1)
rbf
## train a bound constraint support vector machine
irismodel <- ksvm(Species~.,data=iris,type="C-bsvc",
kernel=rbf,C=10,prob.model=TRUE)
irismodel
## get fitted values
fitted(irismodel)
## Test on the training set with probabilities as output
predict(irismodel, iris[,-5], type="probabilities")
#### you can Use custom kernel
k <- function(x,y) {(sum(x*y) +1)*exp(-0.001*sum((x-y)^2))}
class(k) <- "kernel"
```

R Code: Support Vector Machine

```
#### you can Use custom kernel

k <- function(x,y) {(sum(x*y) +1)*exp(-0.001*sum((x-y)^2))}
class(k) <- "kernel"
iris2 = iris[,-5]
## train svm using custom kernel
model <- ksvm(Species ~ ., data = iris2,kernel=k,
C=5,cross=5)

# test with train data
pred <- predict(model, iris2)
# (same as:)
pred <- fitted(model)

# Check accuracy:
mtab=table(pred, iris[,5])
mtab
# install.packages('caret', dependencies=TRUE)
library(caret )
confusionMatrix(mtab)
```

Chapter 10: Feature Selection Techniques

10 Chapter 10: Feature Selection Techniques

10.1 Filter Methods

10.2 Wrapper Methods

10.3 Embedded Methods

10.4 Feature Selection Statistics

10.5 Boruta Feature Selection Algorithm

Feature Selection Method For Machine Learning

When building a machine learning model, we know that having too many features brings issues such as the curse of dimensionality, besides the need for more memory, processing time, and power. On our Feature Engineering pipelines we employ feature selection techniques to try to remove less useful features from our datasets. This raises a problem: how can we determine which features are useful?

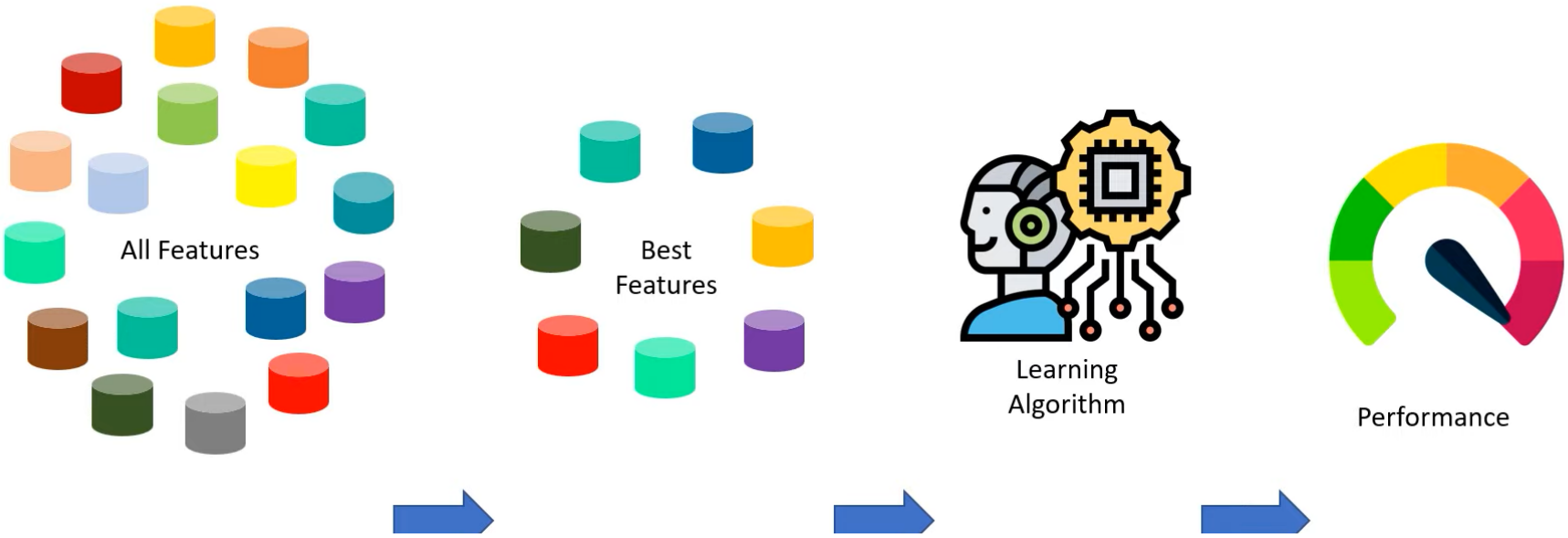
Feature Selection Techniques

1. Supervised Feature Selection
   - Filter methods
   - Embedded methods
   - Wrapper methods
   - Hybrid methods
2. Unsupervised Feature Selection

Filter Methods

1. Basic Filter Methods
   - Variance Threshod (Remove the Constant Feature and Quasi-Constant Features)
   - Mean Absolute Difference (MAD)
   - Remove Duplicate Features
2. Correlation & Ranking Filter Methods
   - Pearson's correlation coefficient (heatmap)
   - Spearman's rank coefficient
   - Kendall's rank coefficient
3. Statistical Methods
   - ANOVA or F-test
   - Mutual Information
   - Chi-Square
   - Information Gain
   - Fisher's Score

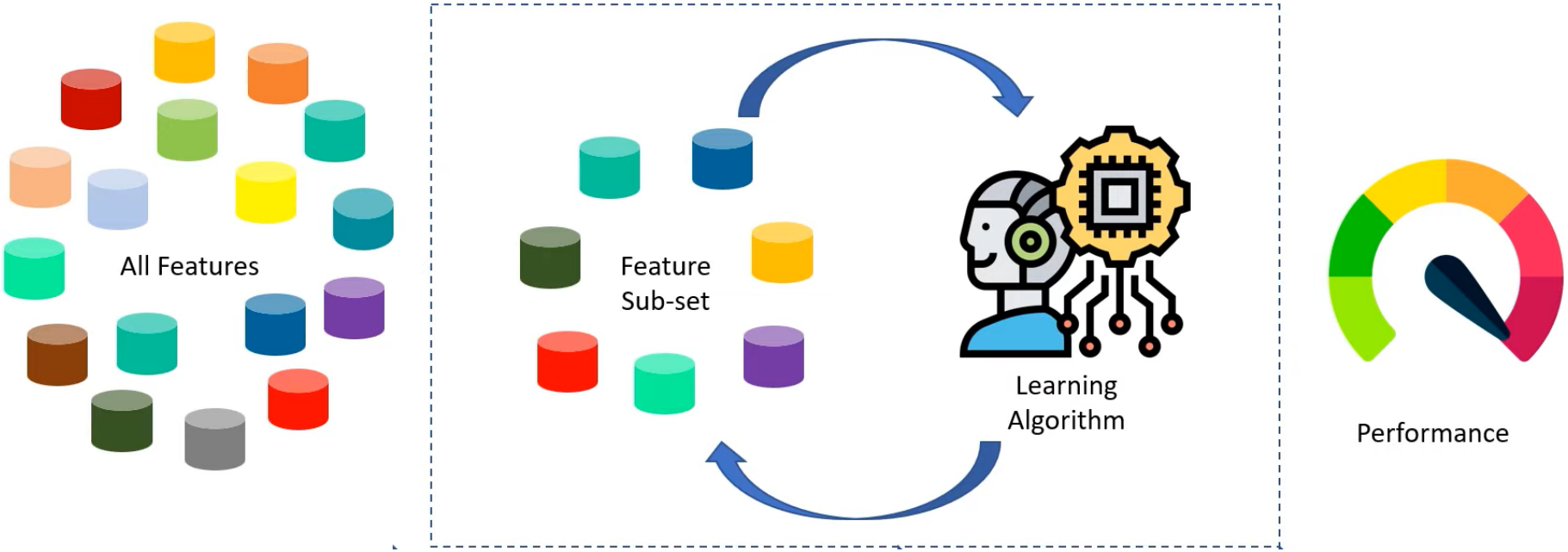
Feature Selection Techniques: Filter Method



Wrapper Methods

1. Forward Feature Selection
2. Backward Feature Elimination
3. Exhaustive Feature Selection
4. Recursive Feature Elimination
5. Boruta Feature Selection Algorithm

Feature Selection Techniques: Wrapper Method



Wrapper Method Techniques

Difference between Filter and Wrapper methods

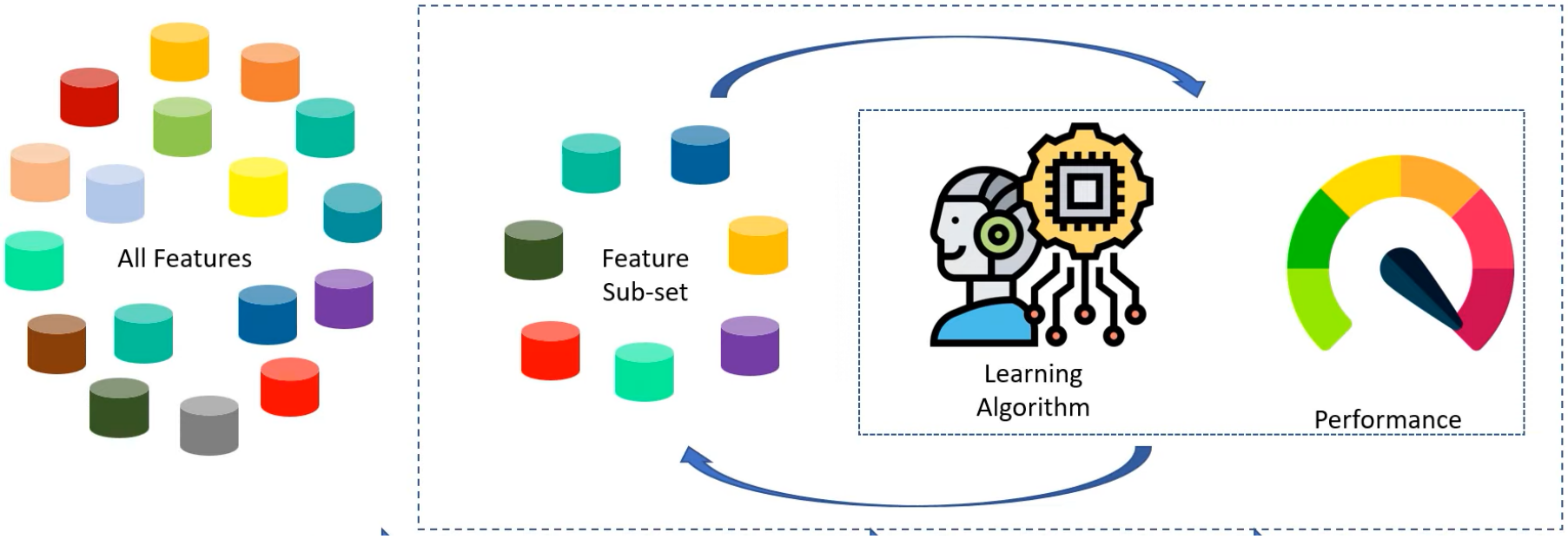
- Filter methods measure the relevance of features by their correlation with dependent variable while wrapper methods measure the usefulness of a subset of feature by actually training a model on it.
- Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally very expensive as well. Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.
- Filter methods might fail to find the best subset of features in many occasions but wrapper methods can always provide the best subset of features.
- Using the subset of features from the wrapper methods make the model more prone to overfitting as compared to using subset of features from the filter methods.

Embedded Methods

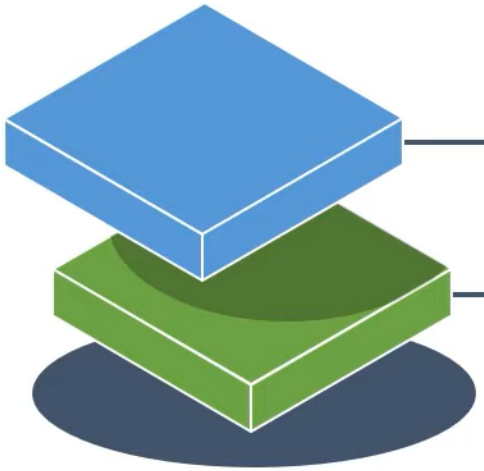
- LASSO Regularization (L_1)
- Random Forest Importance
- LASSO Regularization (L_2)

https://scikit-learn.org/stable/modules/feature_selection.html
https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/

Feature Selection Techniques: Embedded Method



Embedded Method Techniques



- Regularization Techniques like L1, L2 or Elastic Nets
- Random Forest, Tress based methods for feature selection

Advantages of Feature Selection

Disadvantages of Feature Selection

Filter Method

- Ignores interaction with classifier
- Ignores feature dependencies
- Vagueness in filter method

Wrapper Method

- Computationally intensive
- High risk of overfitting
- Lower discriminative power

Feature Selection Statistics

Output Variable	Categorical	Numerical
Numerical	ANOVA or Kendall's rank coefficient	Pearson's Correlation or Spearman's rank Correlation
Categorical	Chi-Squared Test or Mutual Information	ANOVA or Kendall's rank coefficient

Boruta Feature Selection Algorithm

Boruta is a wrapper method of the Feature selection built around the Random Forest Classifier algorithm. It relies in two principles:

- shadow features
- binomial distributions

How does the Boruta algorithm work?

1. Create shadow features
   — random features and shuffle values in columns
2. Train Random Forest/XGBoost and calculate feature importance via mean decrease accuracy or mean decreases Gini impurity
   — the higher the score, the better or more important
3. Check if real features have higher importance compared to shadow features
4. Repeat this for every iteration
5. If original feature performed better, then mark it as important

Toy Example

1. It all starts with X and y

In order to see Boruta in action, let's build a toy dataset with 3 features (*age*, *height* and *weight*), a target variable (*income*) and 5 observations.

```
import pandas as pd

### make X and y
X = pd.DataFrame({'age': [25, 32, 47, 51, 62],
                  'height': [182, 176, 174, 168, 181],
                  'weight': [75, 71, 78, 72, 86]})
y = pd.Series([20, 32, 45, 55, 61], name = 'income')
```

	age	height	weight
0	25	182	75
1	32	176	71
2	47	174	78
3	51	168	72
4	62	181	86

	income
0	20
1	32
2	45
3	55
4	61

Toy Example

```
import numpy as np

### make X_shadow by randomly permuting each column of X
np.random.seed(42)
X_shadow = X.apply(np.random.permutation)
X_shadow.columns = ['shadow_' + feat for feat in X.columns]

### make X_boruta by appending X_shadow to X
X_boruta = pd.concat([X, X_shadow], axis = 1)
```

This is how *X_boruta* would look like in our case:

	age	height	weight	shadow_age	shadow_height	shadow_weight
0	25	182	75	51	176	75
1	32	176	71	32	182	71
2	47	174	78	47	168	78
3	51	168	72	25	181	72
4	62	181	86	62	174	86

X_boruta

Toy Example

```
from sklearn.ensemble import RandomForestRegressor

### fit a random forest (suggested max_depth between 3 and 7)
forest = RandomForestRegressor(max_depth = 5, random_state = 42)
forest.fit(X_boruta, y)

### store feature importances
feat_imp_X = forest.feature_importances_[:len(X.columns)]
feat_imp_shadow = forest.feature_importances_[len(X.columns):]

### compute hits
hits = feat_imp_X > feat_imp_shadow.max()
```

For our toy dataset, the outcome is:

	age	height	weight	shadow_age	shadow_height	shadow_weight
feature importance %	39	19	8	11	14	9
hits	1	1	0	-	-	-

Outcome of one run

Toy Example

```
### initialize hits counter
hits = np.zeros((len(X.columns)))

### repeat 20 times
for iter_ in range(20):

   ### make X_shadow by randomly permuting each column of X
   np.random.seed(iter_)
   X_shadow = X.apply(np.random.permutation)
   X_boruta = pd.concat([X, X_shadow], axis = 1)

   ### fit a random forest (suggested max_depth between 3 and 7)
   forest = RandomForestRegressor(max_depth = 5, random_state = 42)
   forest.fit(X_boruta, y)

   ### store feature importance
   feat_imp_X = forest.feature_importances_[:len(X.columns)]
   feat_imp_shadow = forest.feature_importances_[len(X.columns):]

   ### compute hits for this trial and add to counter
   hits += (feat_imp_X > feat_imp_shadow.max())
```

This is the outcome:

	age	height	weight
hits (in 20 trials)	20	4	0

Toy Example

Now, how do we set a decision criterion? This is the second brilliant idea contained in Boruta.

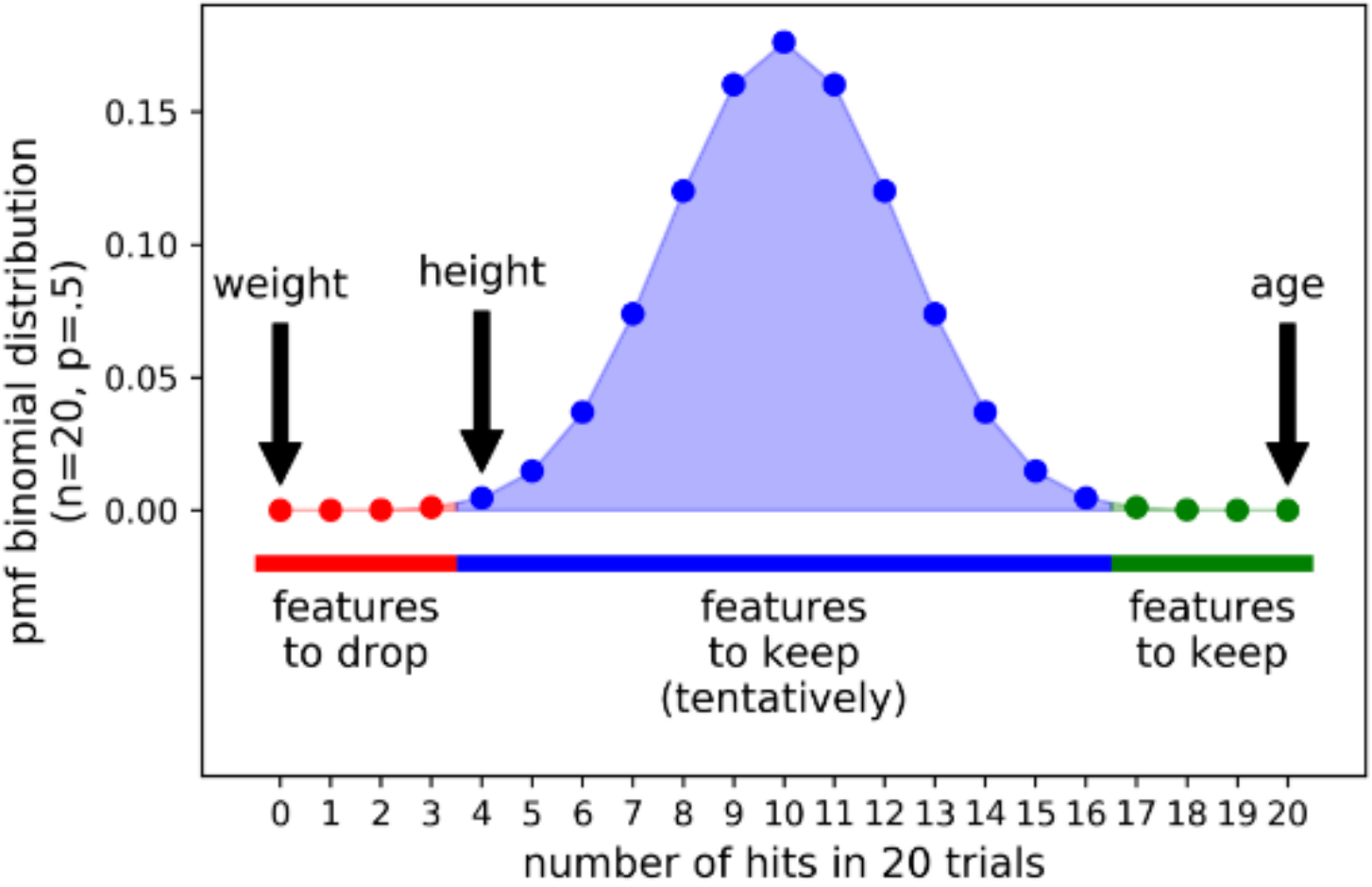
Let's take a feature (no matter if *age*, *height* or *weight*) and say we have absolutely no clue if it's useful or not. What is the probability that we shall keep it? **The maximum level of uncertainty about the feature is expressed by a probability of 50%, like tossing a coin.** Since each independent experiment can give a binary outcome (hit or no hit), **a series of *n* trials follows a <u>binomial distribution</u>.**

In Python, the probability mass function of a binomial distibution can be computed as follows:

```
import scipy as sp

trials = 20
pmf = [sp.stats.binom.pmf(x, trials, .5) for x in range(trials + 1)]
```

Toy Example

These values look like a bell:



Toy Example

In Boruta, there is not a hard threshold between a refusal and an acceptance area. Instead, there are 3 areas:

- an **area of refusal** (the red area): the features that end up here are considered as noise, so they are dropped;
- an **area of irresolution** (the blue area): Boruta is indecisive about the features that are in this area;
- an **area of acceptance** (the green area): the features that are here are considered as predictive, so they are kept.

The areas are defined by selecting the two most extreme portions of the distribution called **tails of the distribution** (in our example each tail accounts for 0.5% of the distribution).

Toy Example

So, we did 20 iterations on our toy data and ended up with some statistically grounded conclusions:

- in order to predict the *income* of a person, *age* is predictive and should be kept, *weight* is just noise and should be dropped,
- Boruta was indecisive about *height*: the choice is up to us, but in a conservative frame, it is advisable to keep it.

In this paragraph, we have implemented the necessary code, but a great (optimized) library for Boruta in Python exists.

Python code: Boruta Algorithm

```
!pip install boruta
from boruta import BorutaPy
from sklearn.ensemble import RandomForestRegressor
import numpy as np
###initialize Boruta
forest = RandomForestRegressor(
n_jobs = -1,
max_depth = 5
)
boruta = BorutaPy(
estimator = forest,
n_estimators = 'auto',
max_iter = 100 # number of trials to perform
)
### fit Boruta (it accepts np.array, not pd.DataFrame)
boruta.fit(np.array(X), np.array(y))
### print results
green_area = X.columns[boruta.support_].to_list()
blue_area = X.columns[boruta.support_weak_].to_list()
print('features in the green area:', green_area)
print('features in the blue area:', blue_area)
```

Chapter 11: Neural Network (NN) Modelling

11 Chapter 11: Neural Network (NN) Modelling

11.1 Problem & Motivation

11.2 Basic Structure of ANN

11.3 What is a neuron?

11.4 Activation Functions in NN

11.5 Perceptron

11.6 Multi-Layered Perceptron Model

11.7 Feed Forward Neural Networks

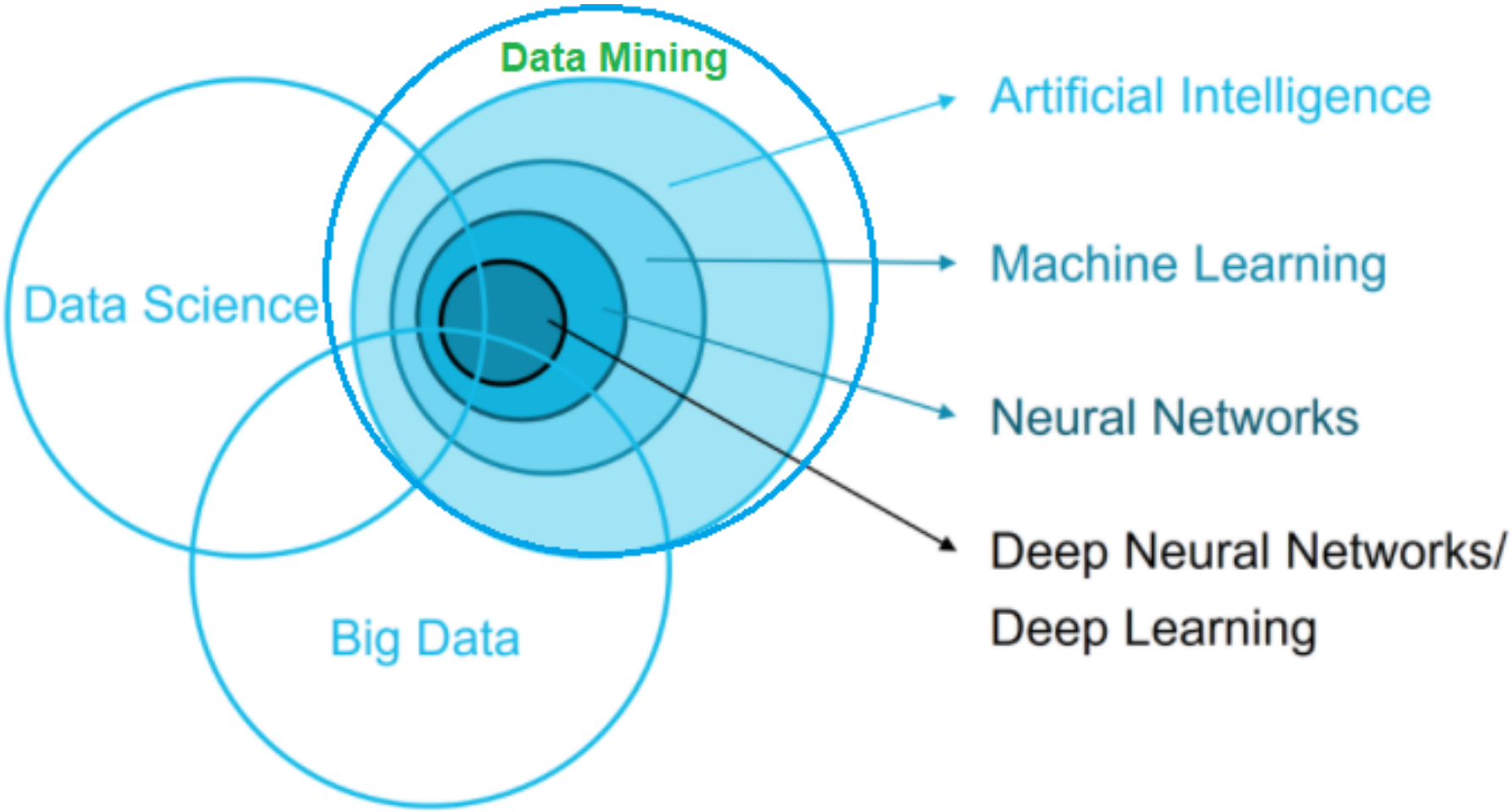
11.8 Backpropagation Algorithm

11.9 Overfitting vs. underfitting

11.10 Disadvantages of ANN

11.11 Vanishing Gradient Problem

Deep Neural Network

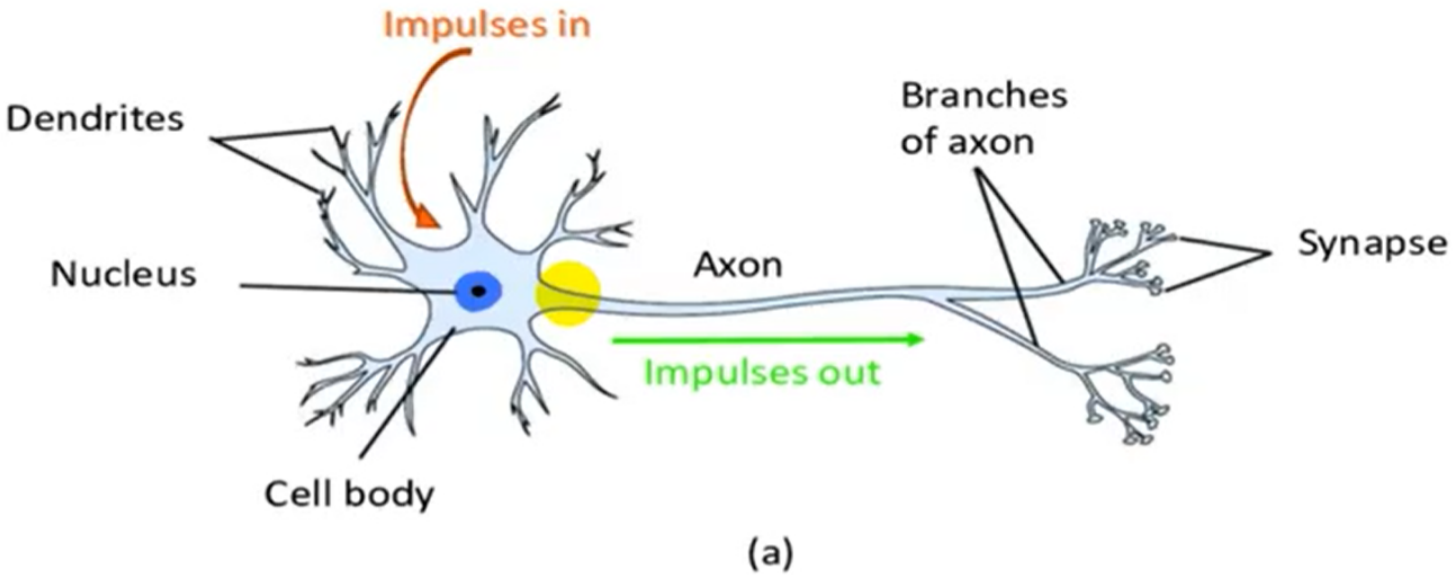


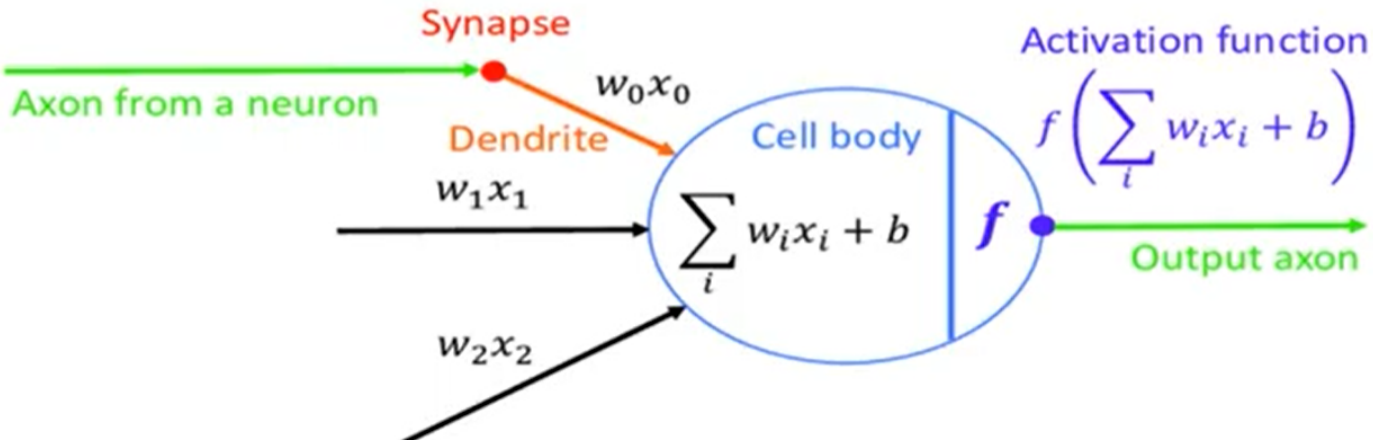
What is a neural network?

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a special type of machine learning process, called *deep learning*, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

- is a supervised learning (why?)
- neural networks (NN), often referred to as artificial neural networks (ANN) to distinguish them from biological neural networks
- the NN is structured as
  - a directed graph with many nodes (processing elements)
  - arcs (interconnections) between them
- the nodes in the graph are like individual neurons, while the arcs are their interconnections

How Artificial Neural Networks Work?



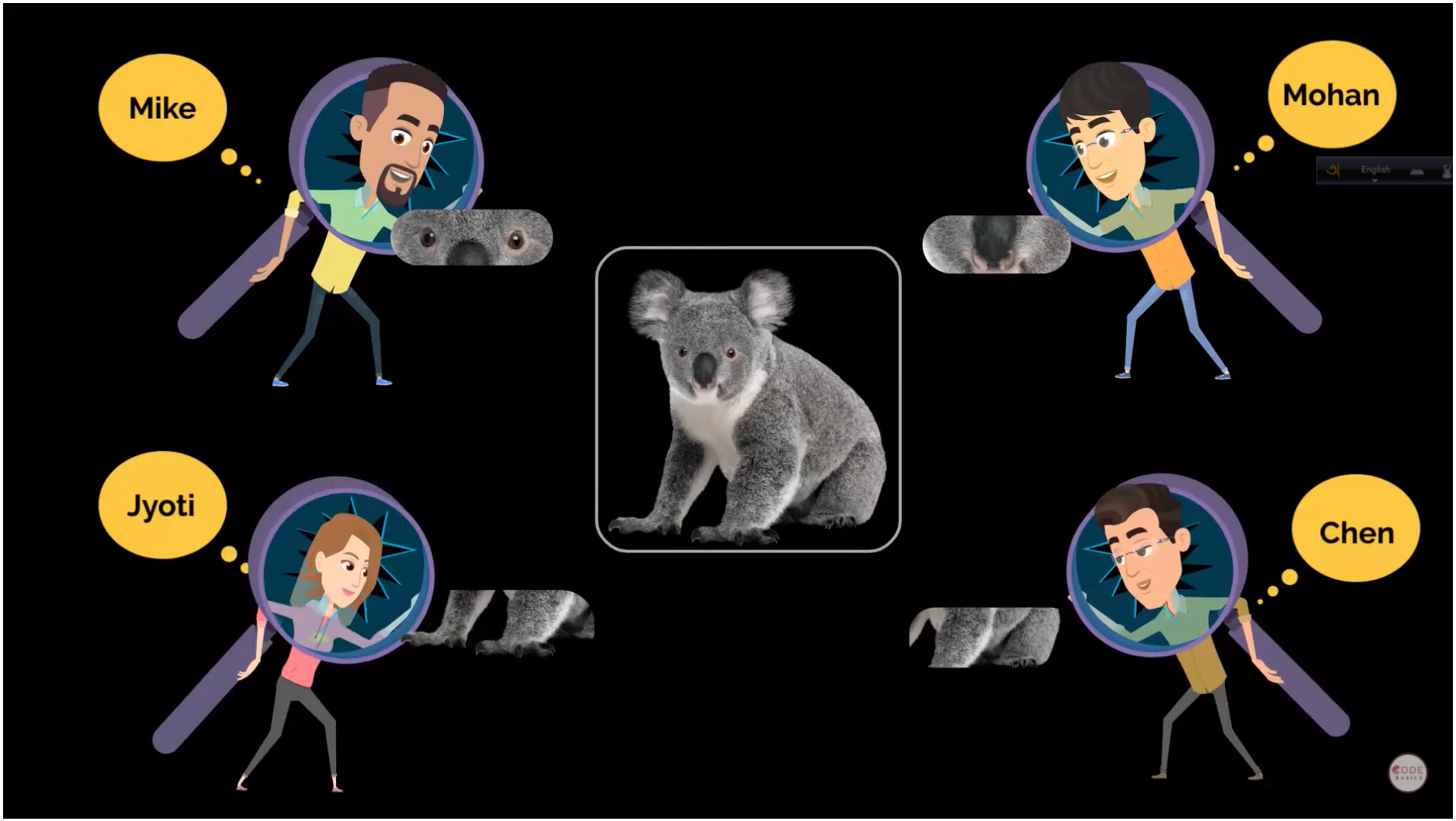


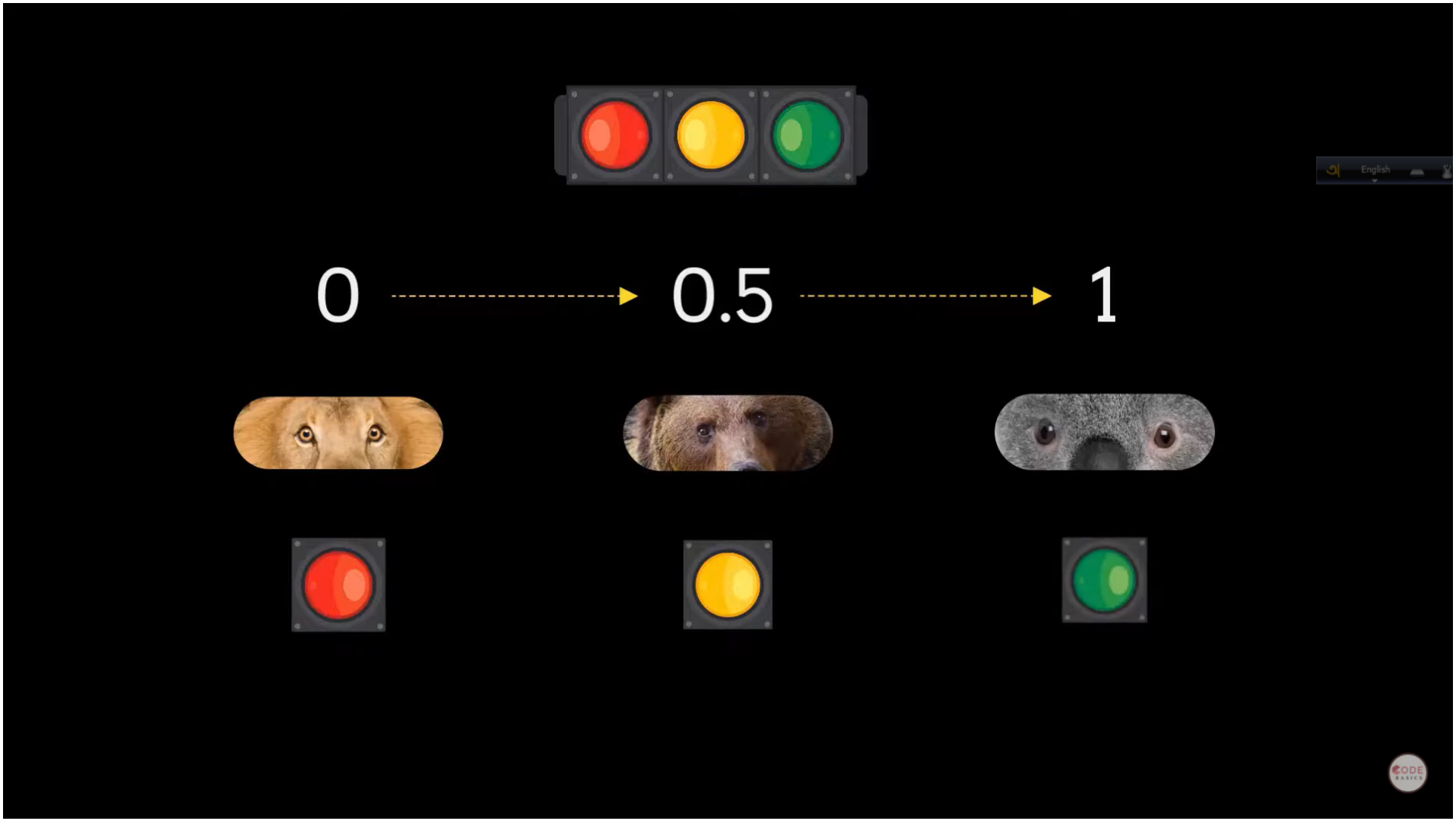
Types of Neural Networks

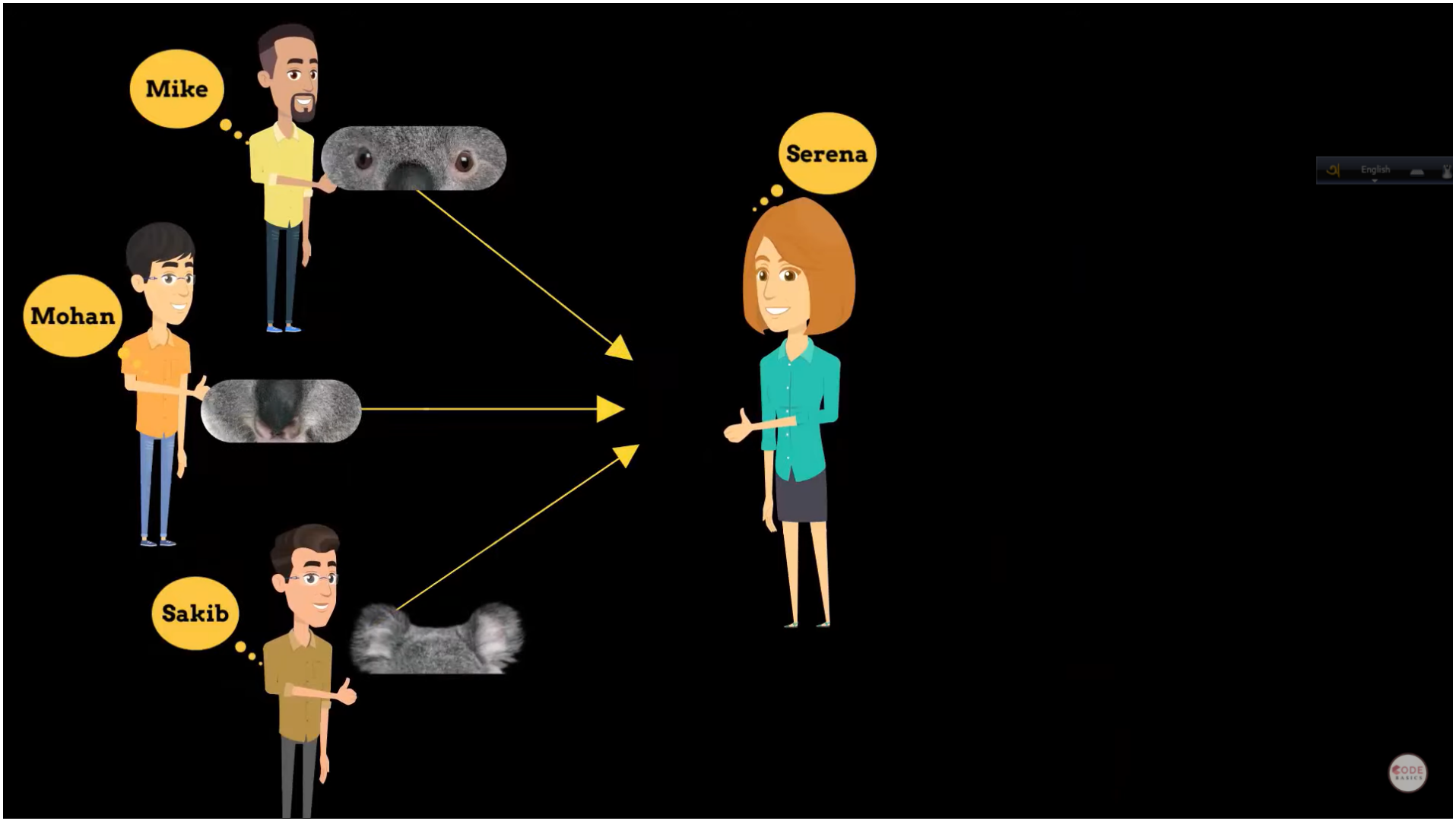
the three most important types of neural networks are

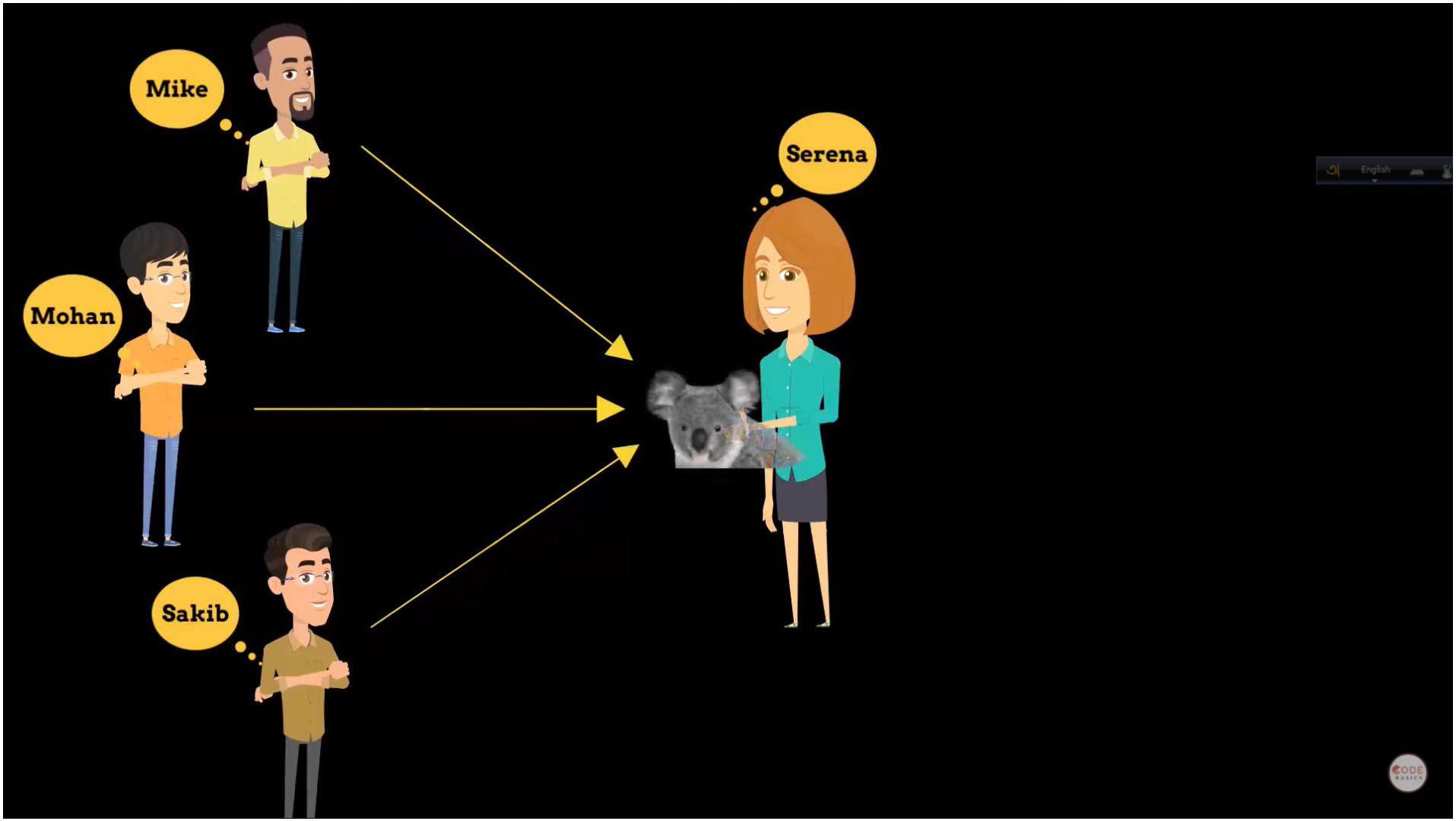
1. Artificial Neural Networks (ANN)
   - Perceptron
   - Feed Forward Networks
   - Multi-Layer Perceptron
   - Radial Based Networks
2. Convolutional Neural Networks (CNN)
   — mainly used for image processing
3. Recurrent Neural Networks (RNN)
   — mainly used for Natural Language Processing (NLP)

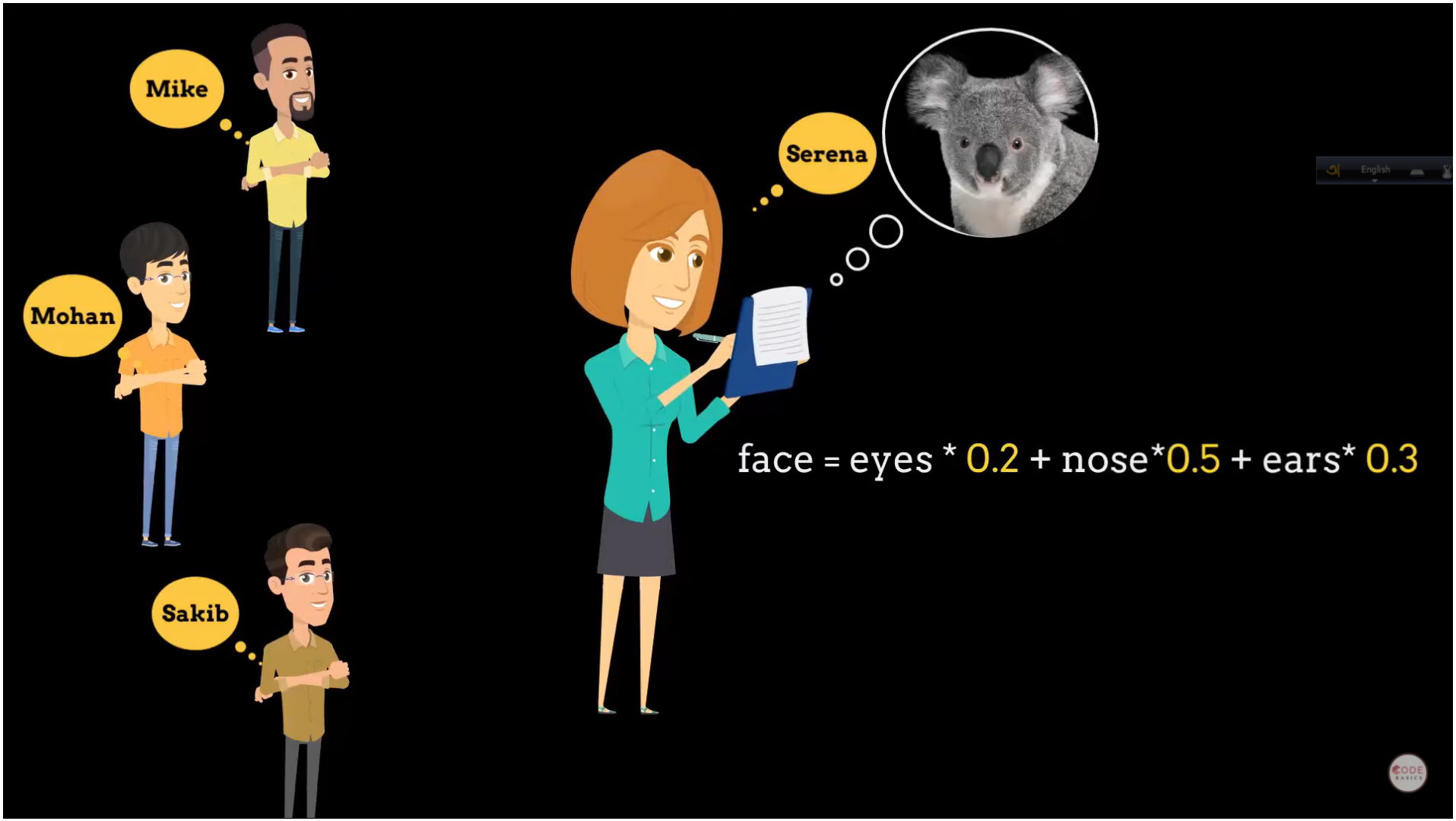
Is this Koala?
Yes
No

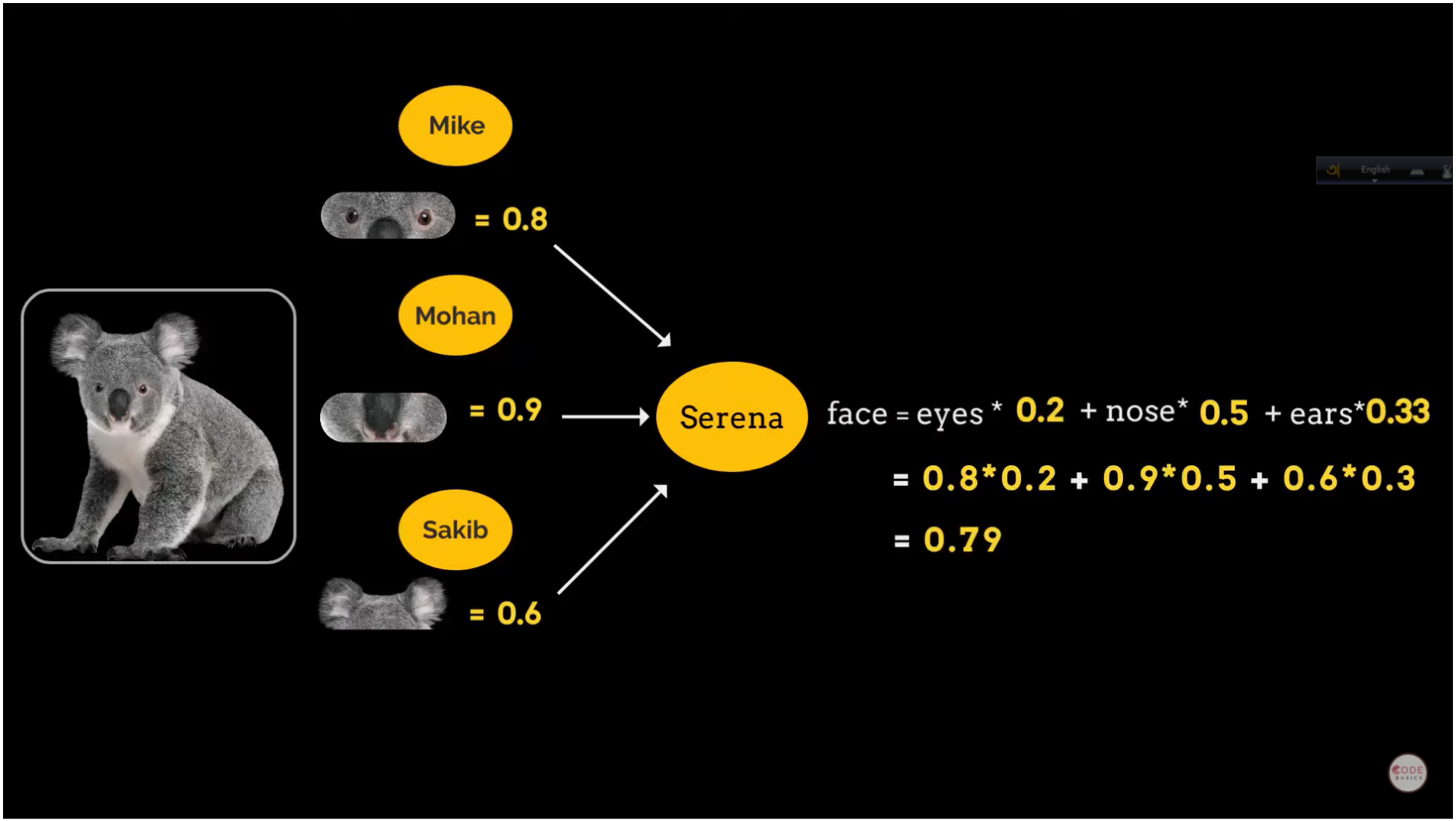

Mike
Mohan
Jyoti
Chen

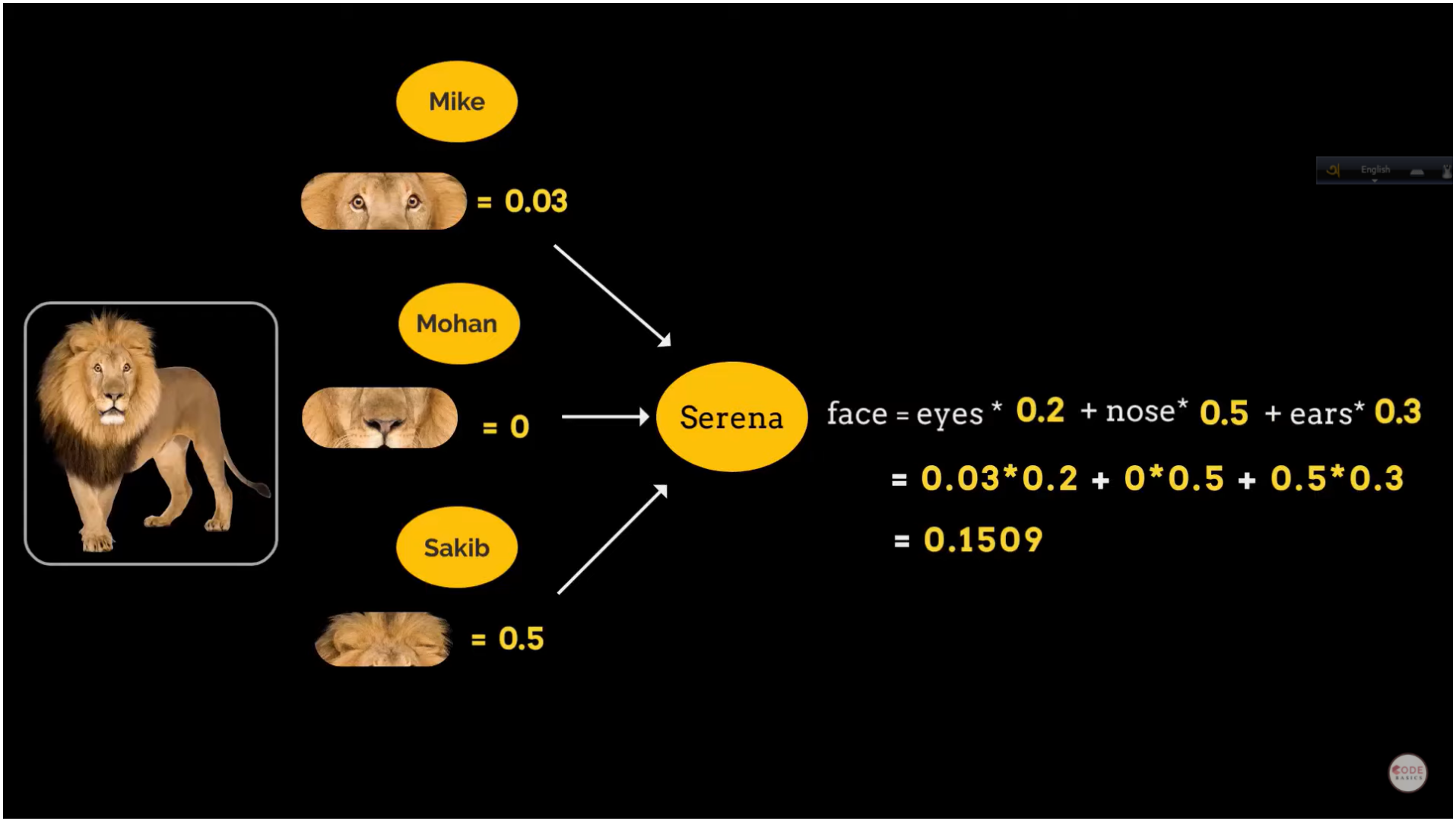

0
0.5
1

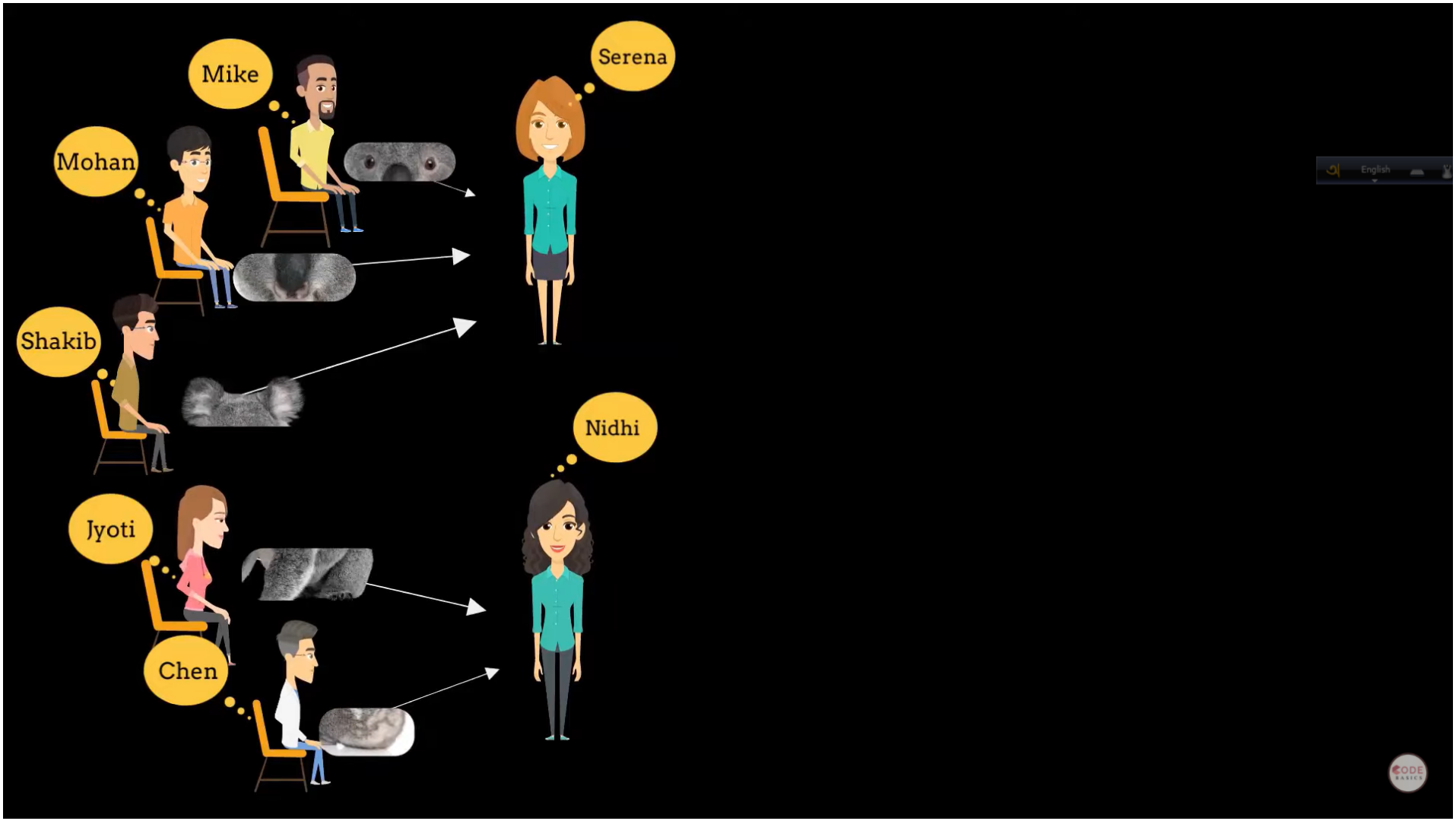

Mike
Mohan
Sakib
Serena

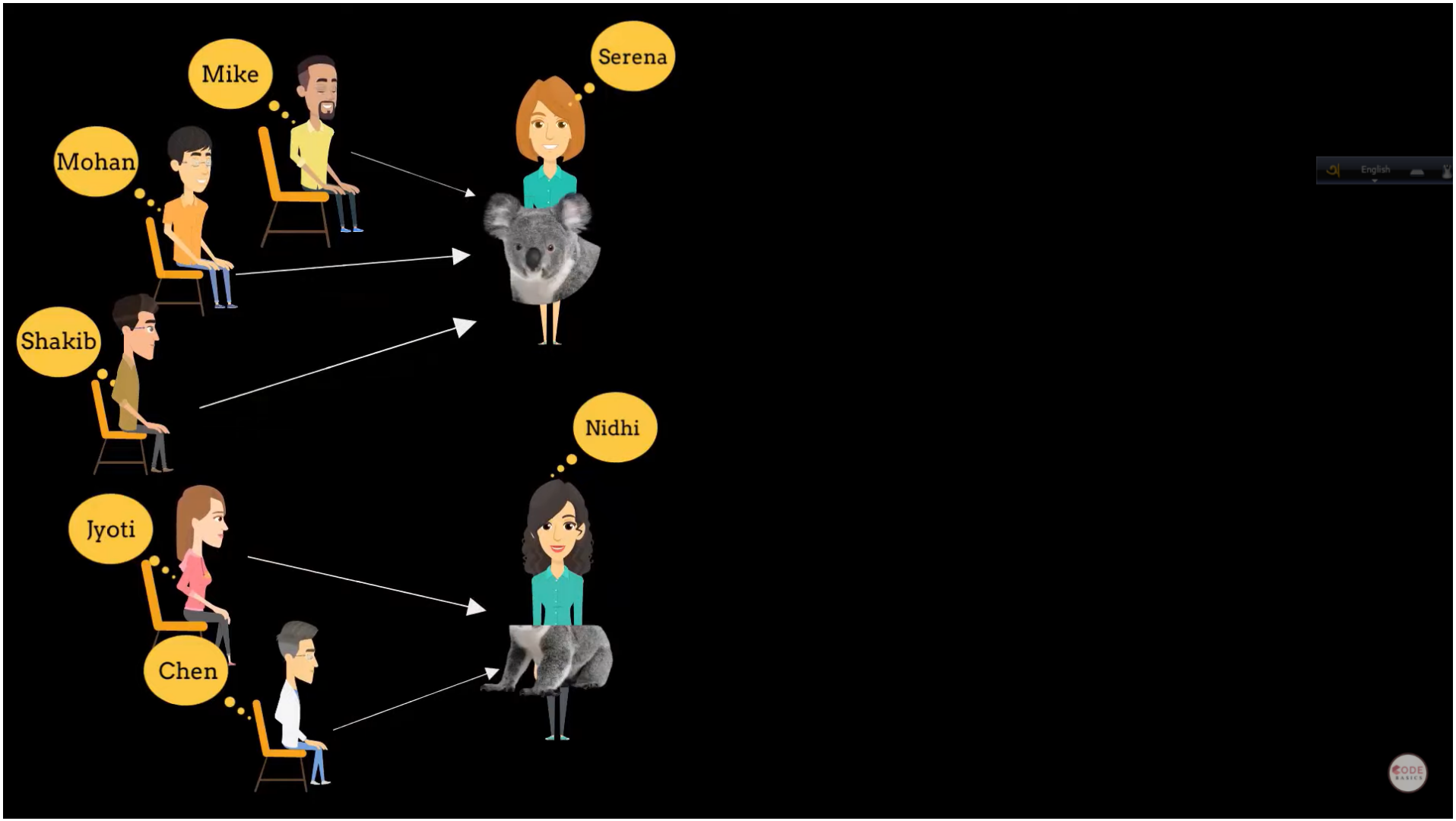

Mike
Mohan
Sakib
Serena

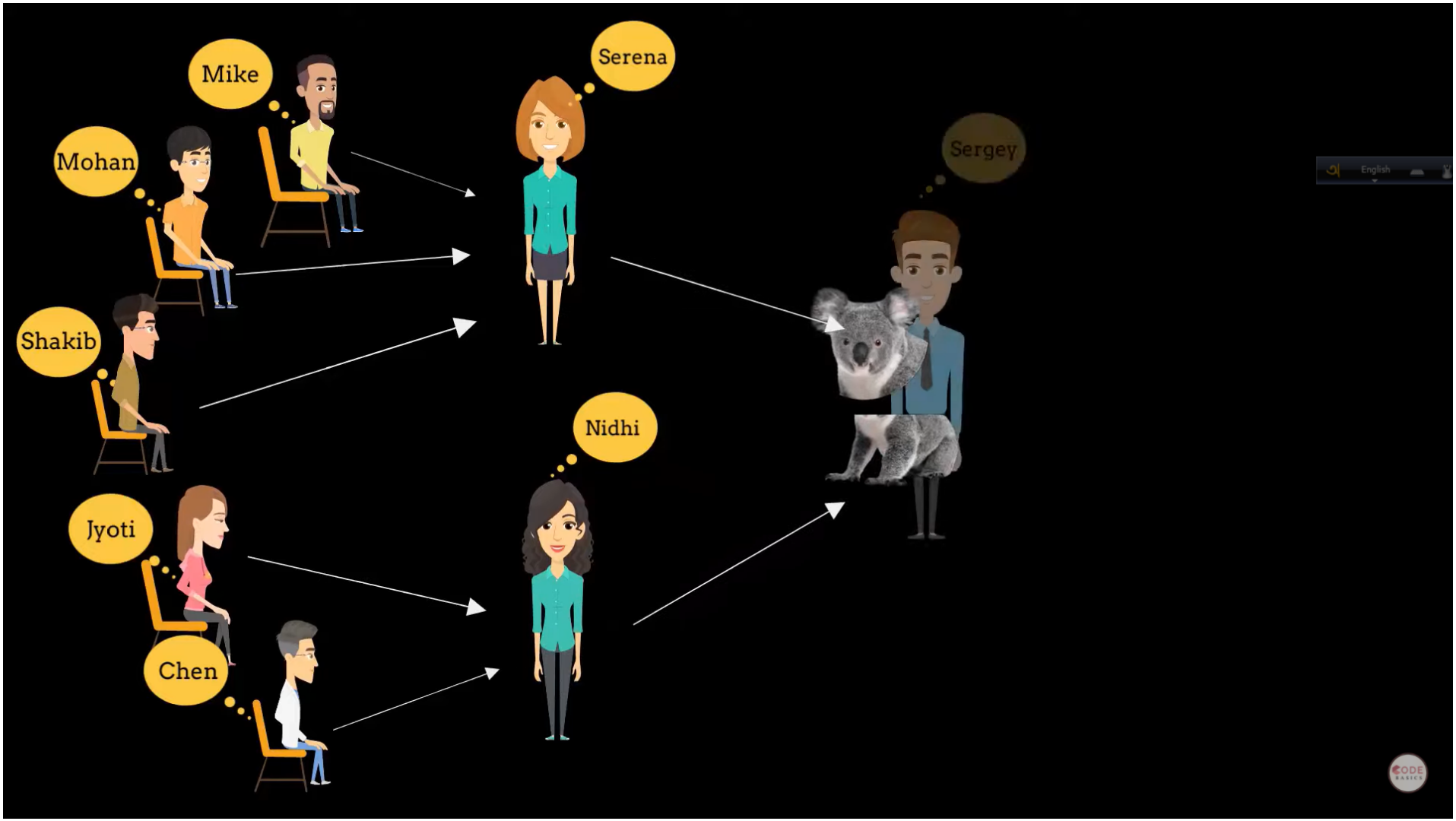

Mike
Mohan
Sakib
Serena
face = eyes * 0.2 + nose*0.5 + ears* 0.3

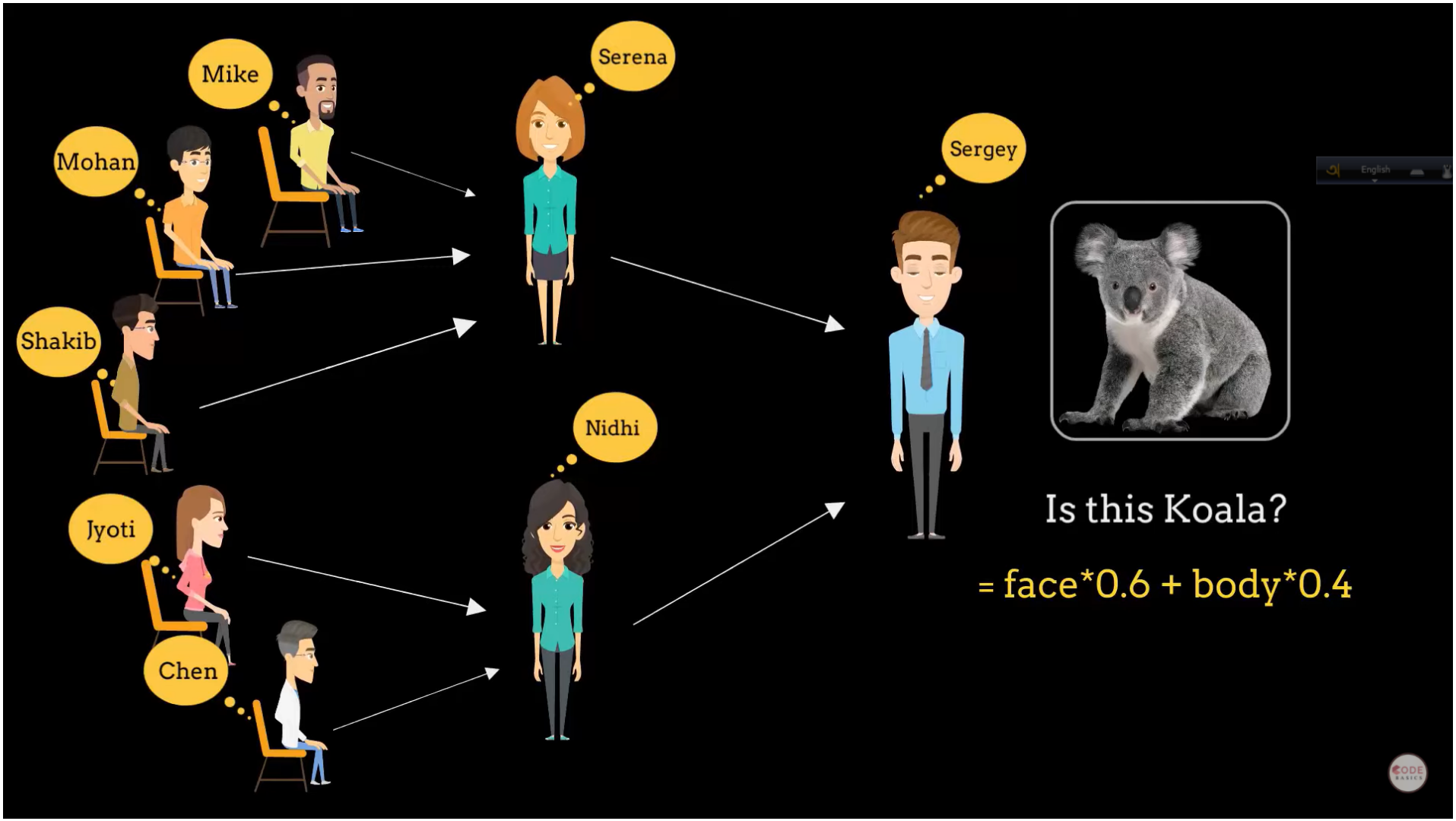

Mike
= 0.8
Mohan
= 0.9
Sakib
= 0.6
Serena
face = eyes * 0.2 + nose* 0.5 + ears*0.33
= 0.8*0.2 + 0.9*0.5 + 0.6*0.3
= 0.79

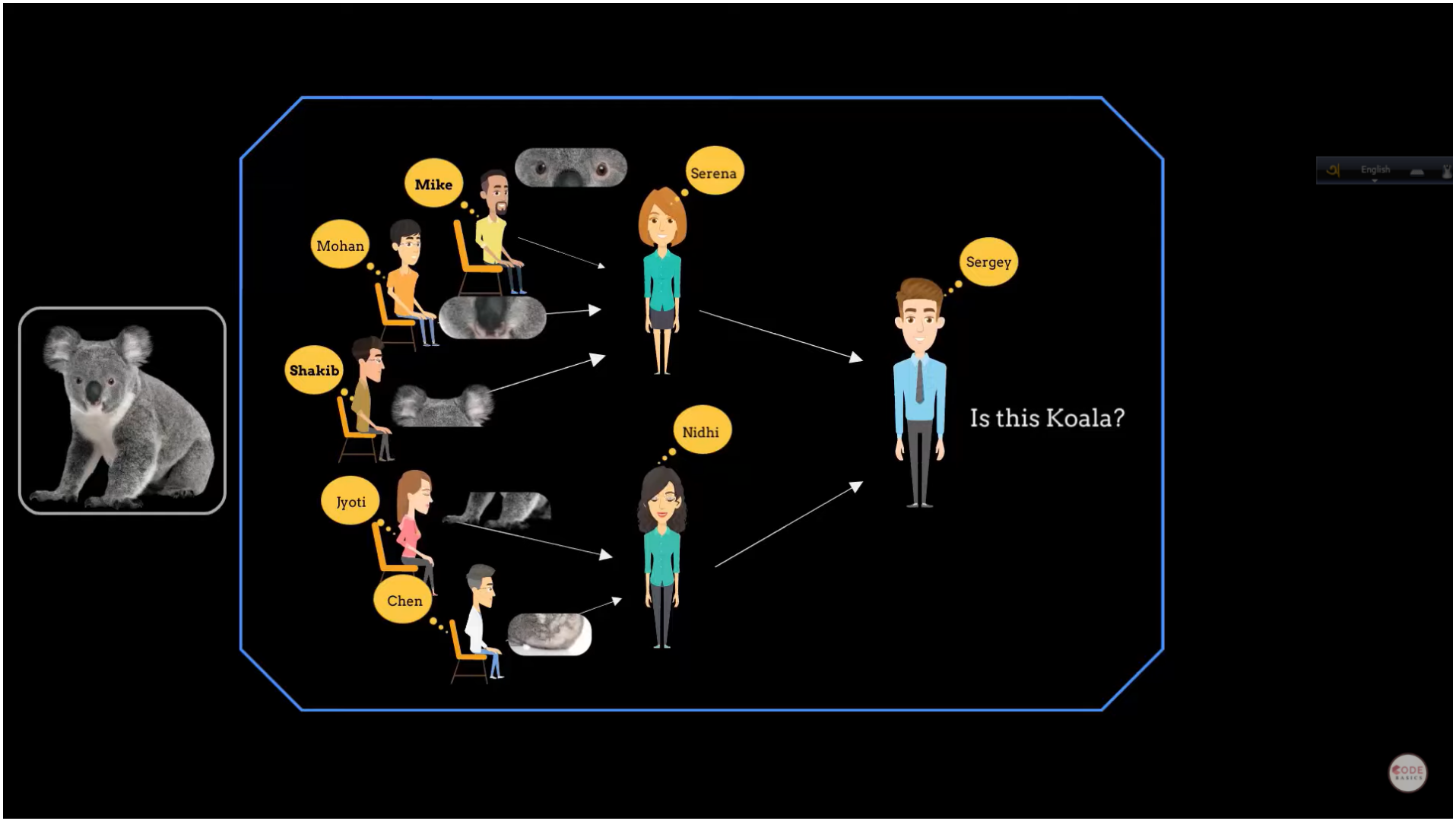

Mike
= 0.03
Mohan
= 0
Sakib
= 0.5
Serena
face = eyes * 0.2 + nose* 0.5 + ears* 0.3
= 0.03*0.2 + 0*0.5 + 0.5*0.3
= 0.1509

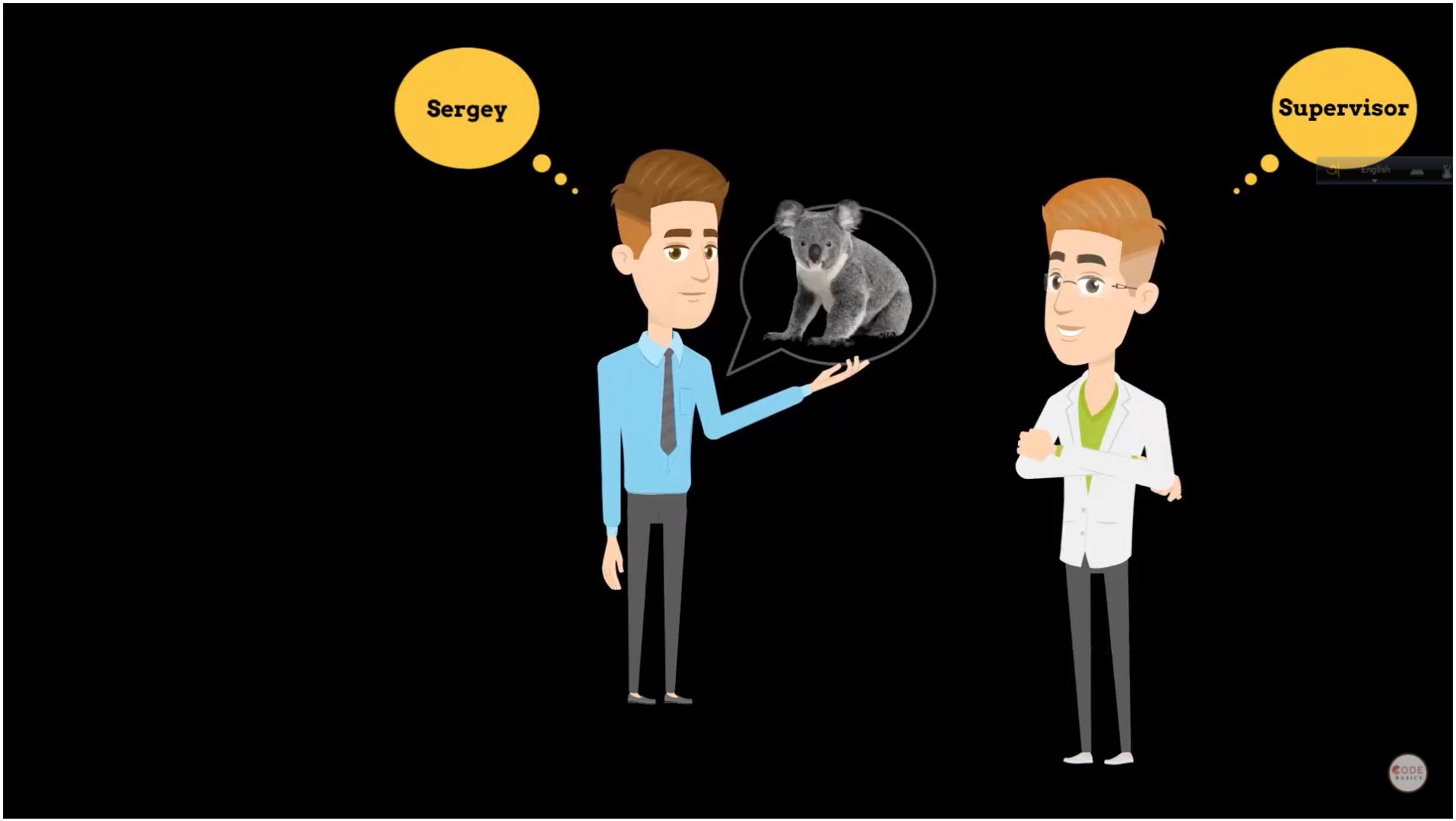

Mike
Serena
Mohan
Shakib
Nidhi
Jyoti
Chen

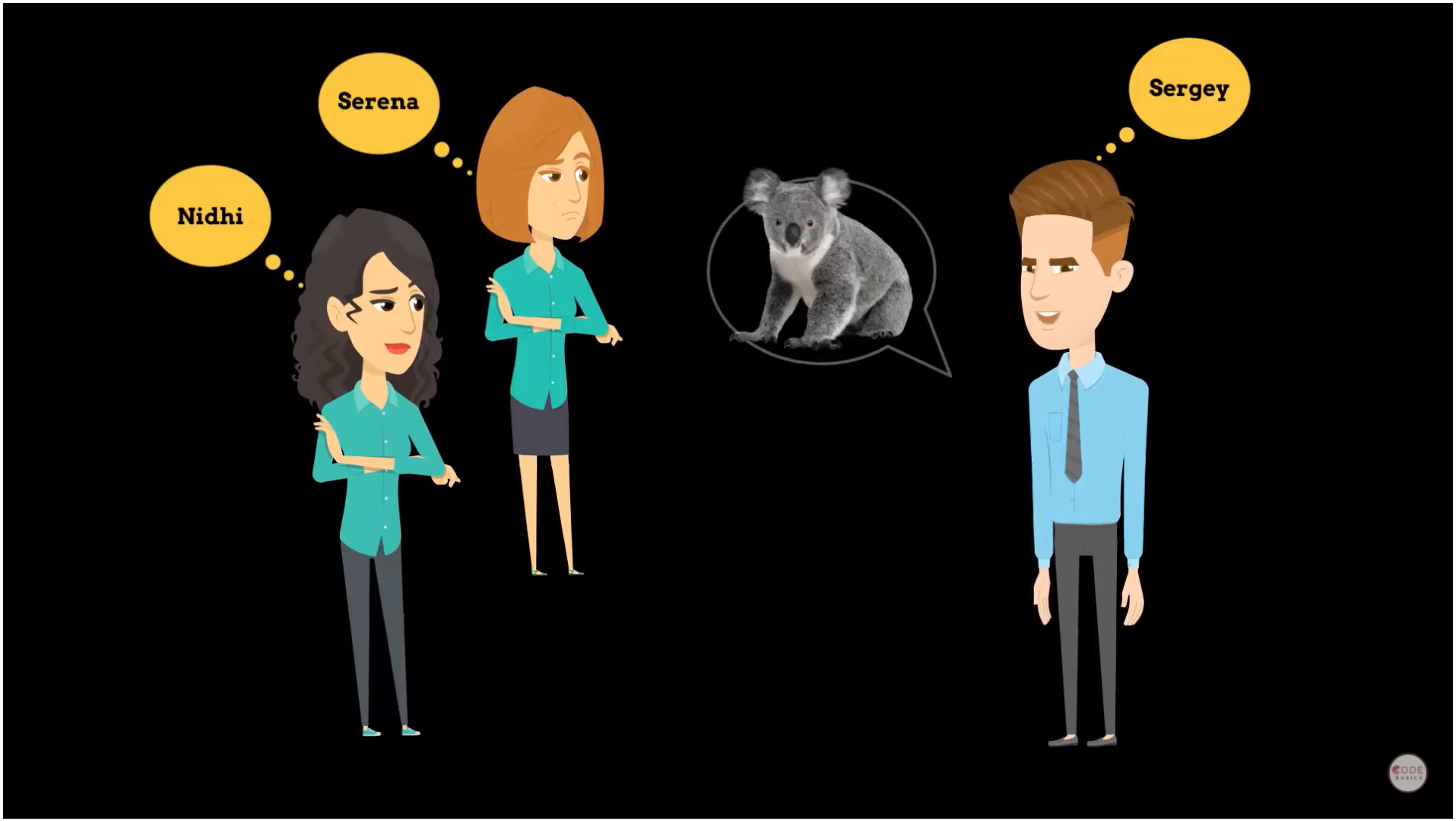

Mike
Serena
Mohan
Shakib
Nidhi
Jyoti
Chen

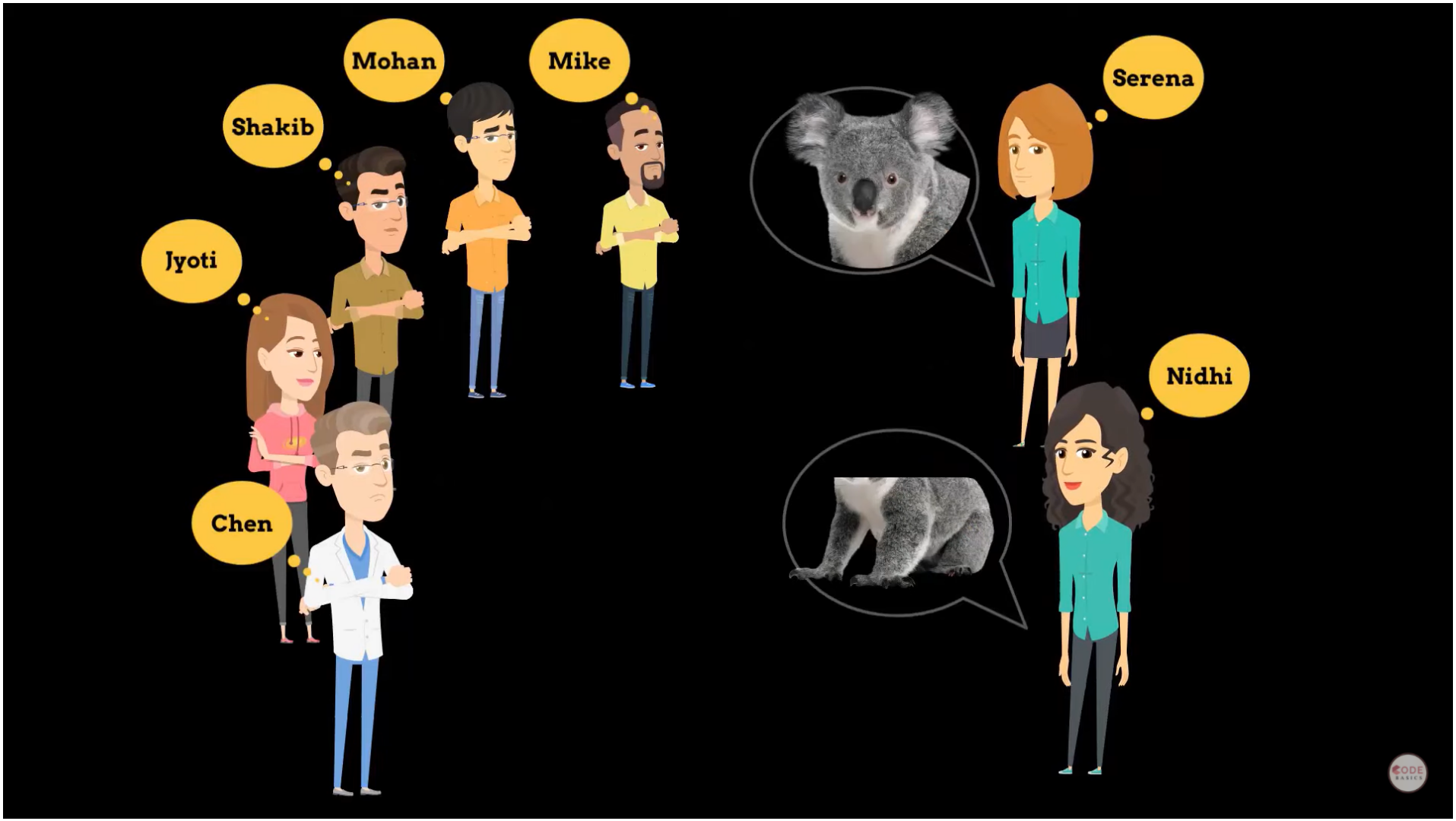

Mike
Serena
Mohan
Sergey
Shakib
Nidhi
Jyoti
Chen

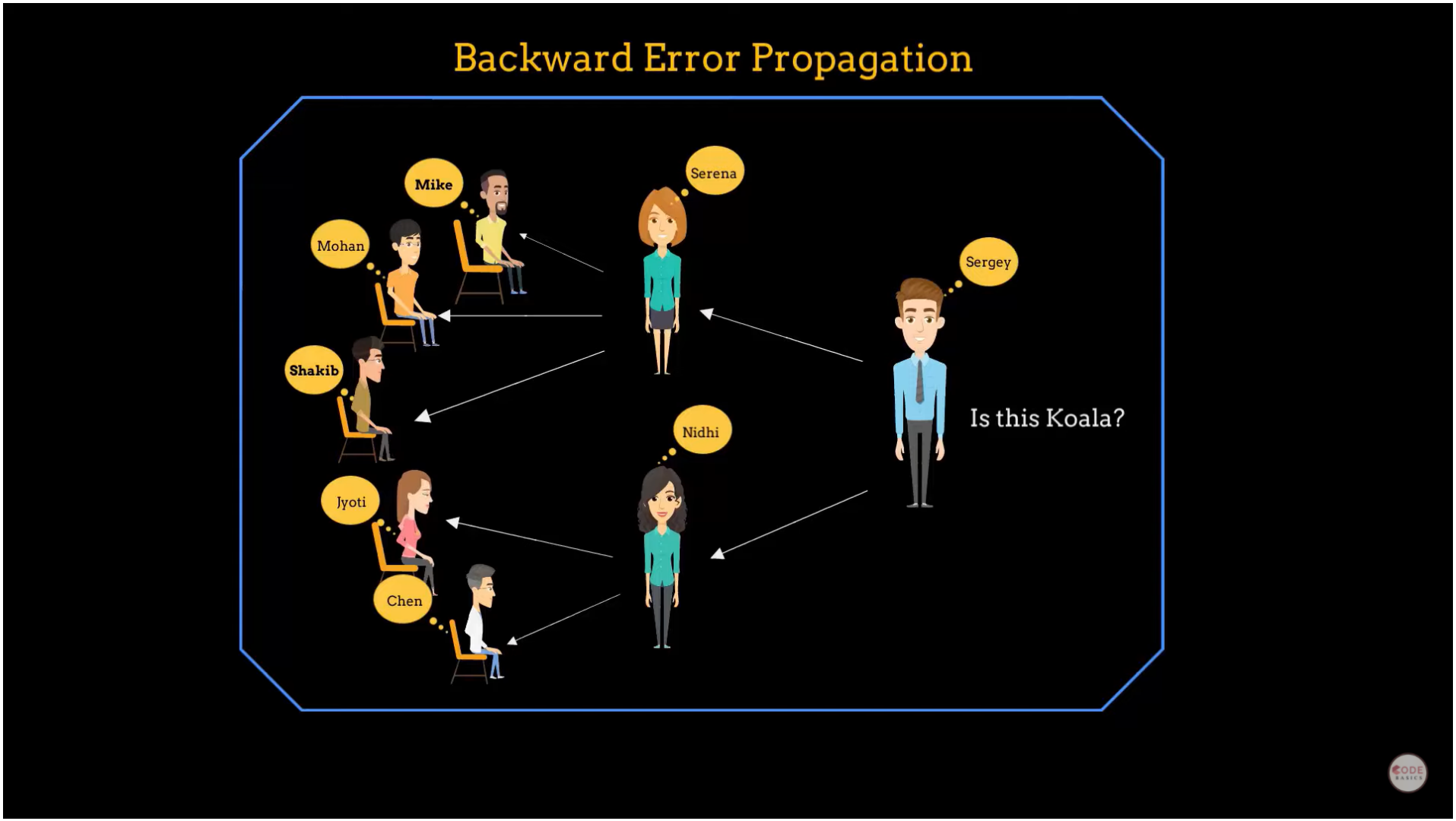

Mike
Mohan
Shakib
Jyoti
Chen
Serena
Nidhi
Sergey
Is this Koala?
= face*0.6 + body*0.4

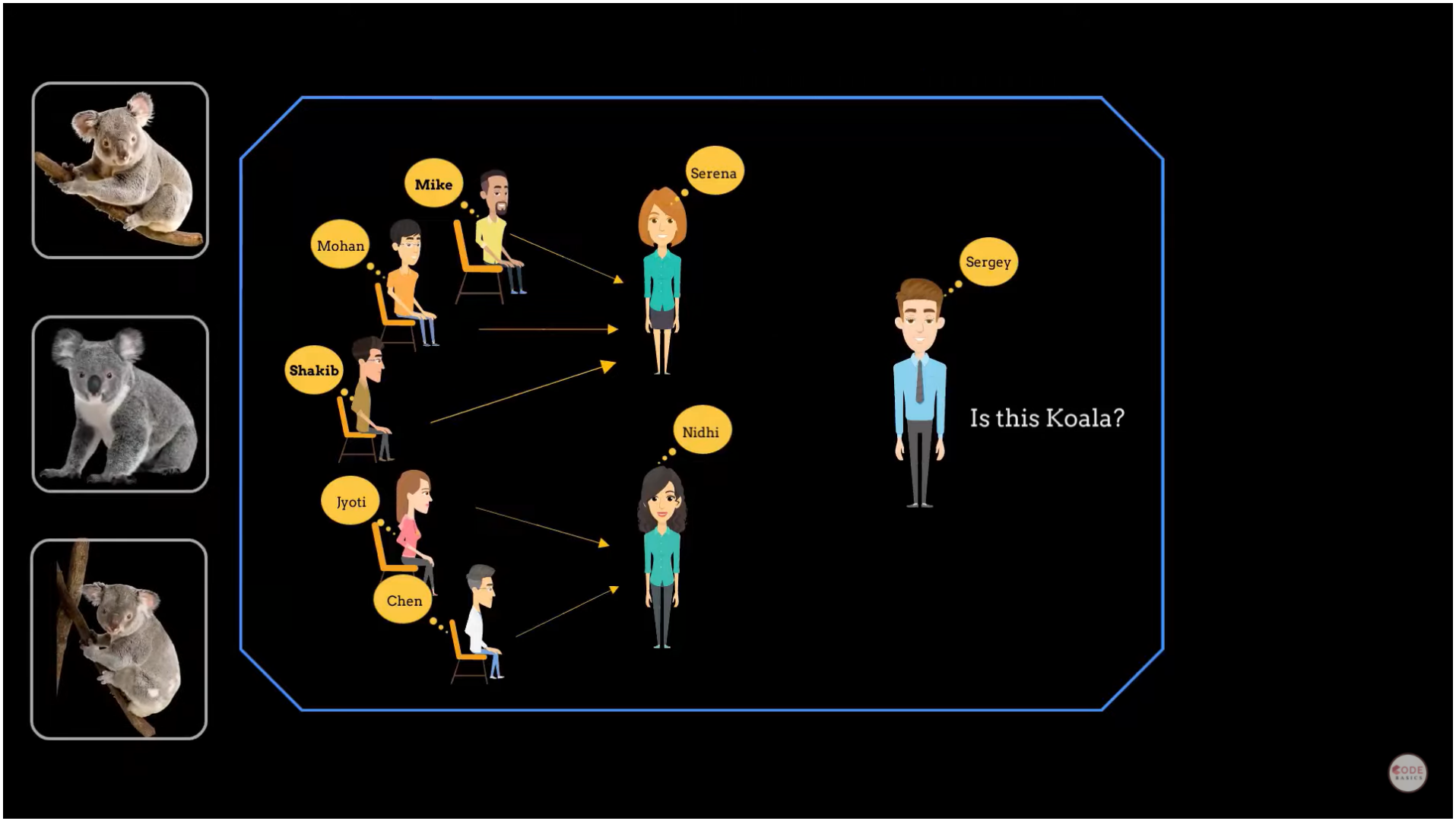

Mike
Serena
Mohan
Sergey
Shakib
Is this Koala?
Nidhi
Jyoti
Chen

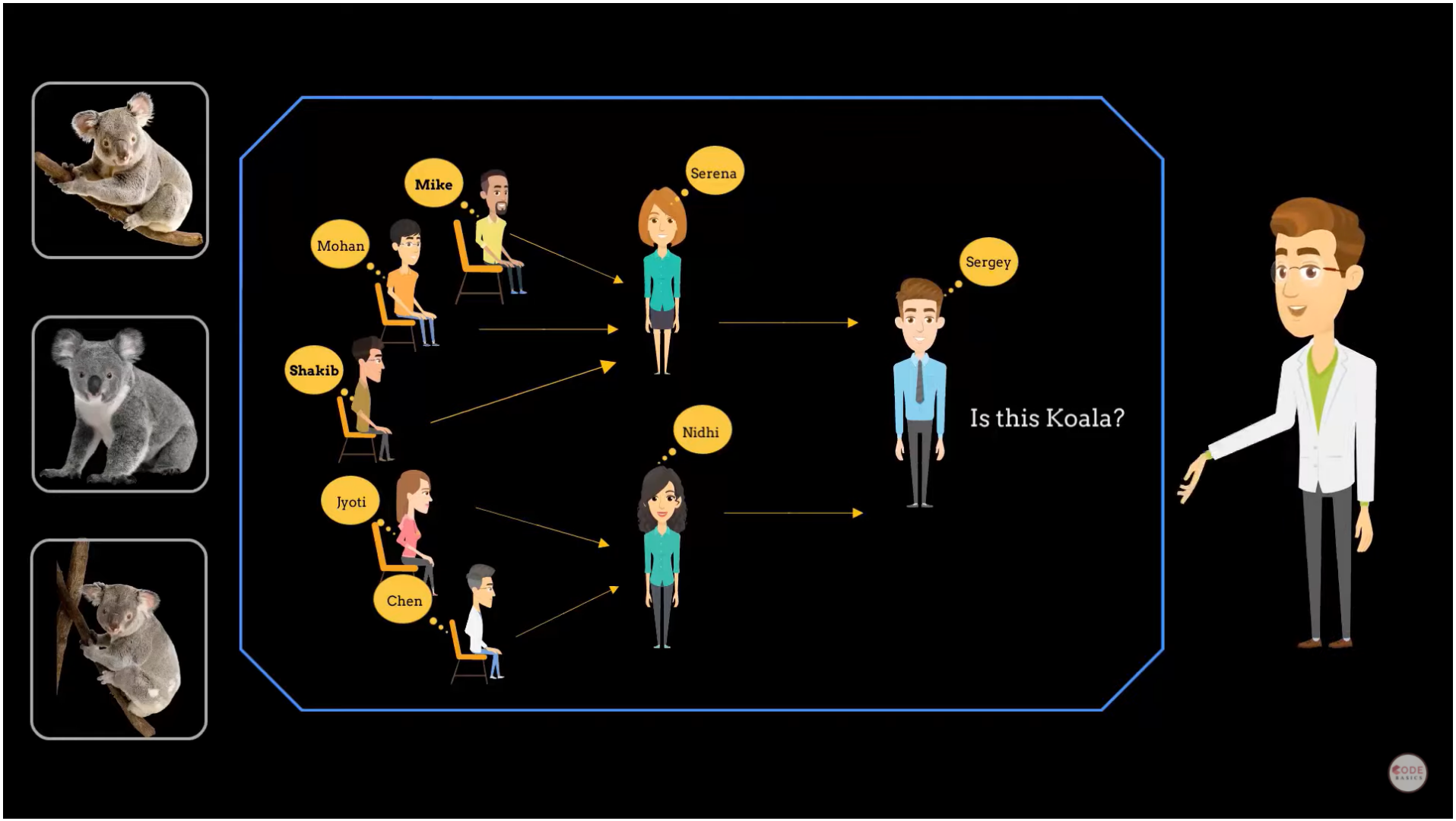

Sergey
Supervisor

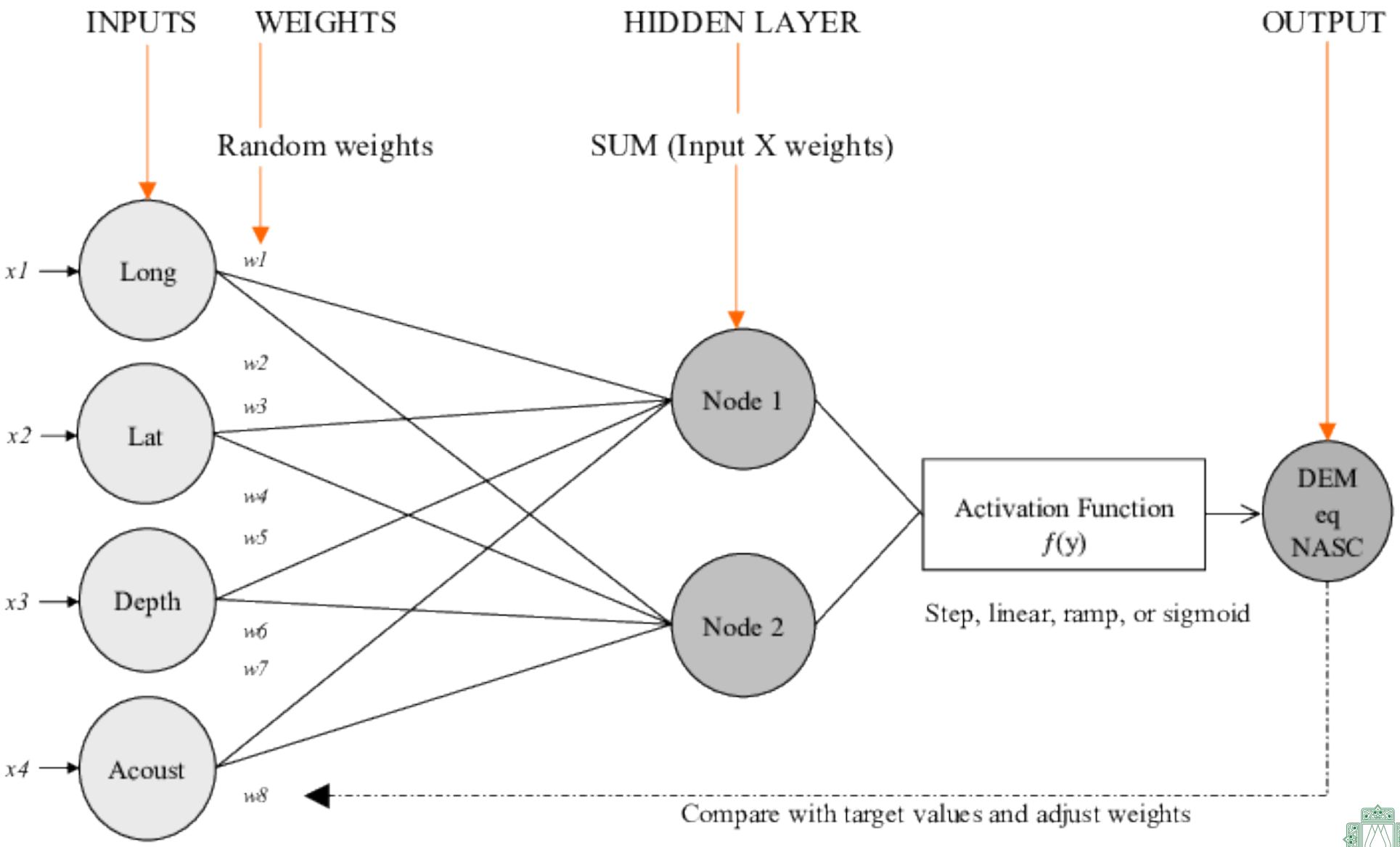

Nidhi
Serena
Sergey


Mohan
Mike
Shakib
Jyoti
Chen
Serena
Nidhi

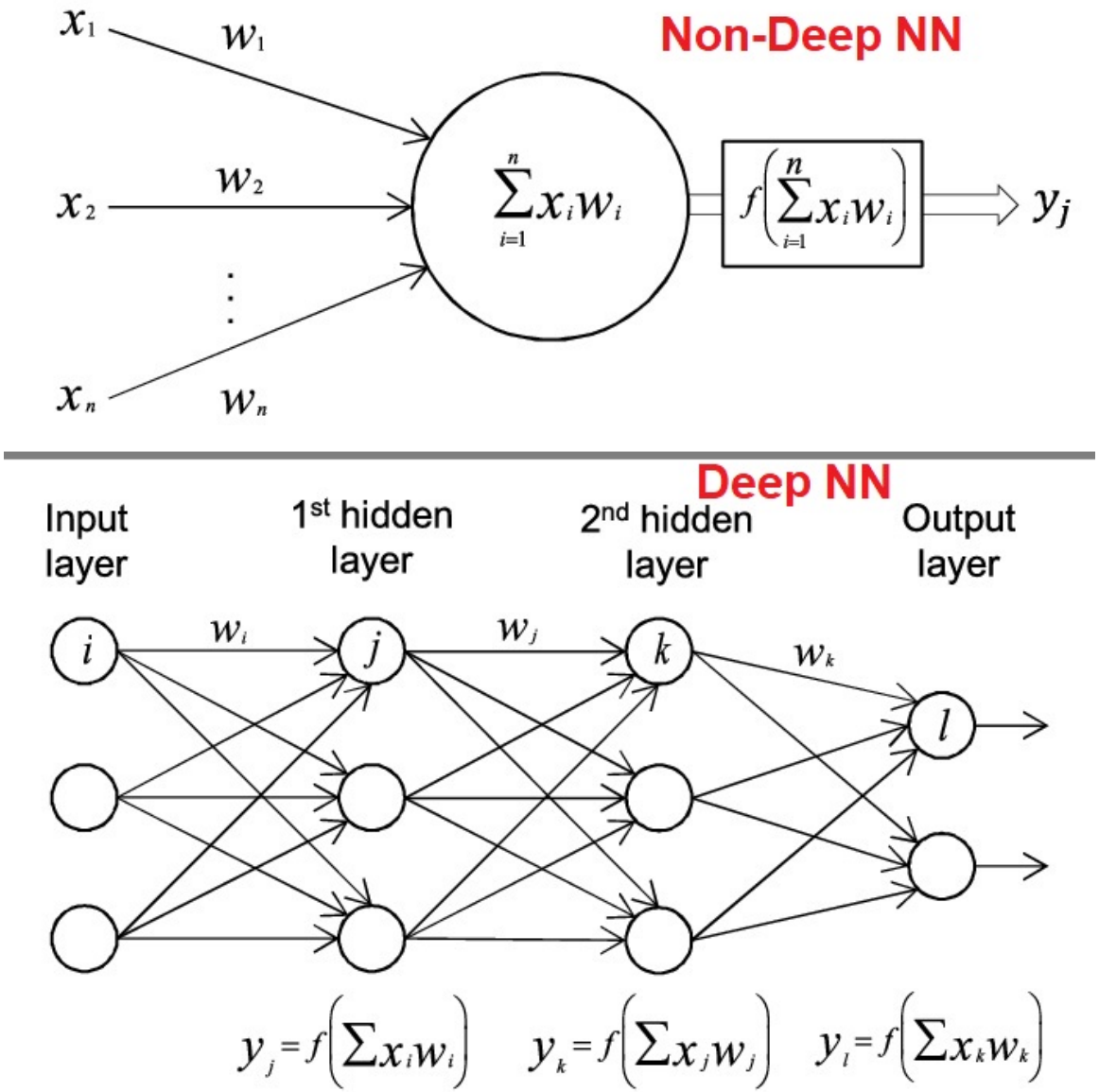

Backward Error Propagation
Mike
Serena
Mohan
Sergey
Shakib
Is this Koala?
Nidhi
Jyoti
Chen


Mike
Serena
Mohan
Sergey
Shakib
Is this Koala?
Nidhi
Jyoti
Chen


Mike
Serena
Mohan
Sergey
Shakib
Is this Koala?
Nidhi
Jyoti
Chen

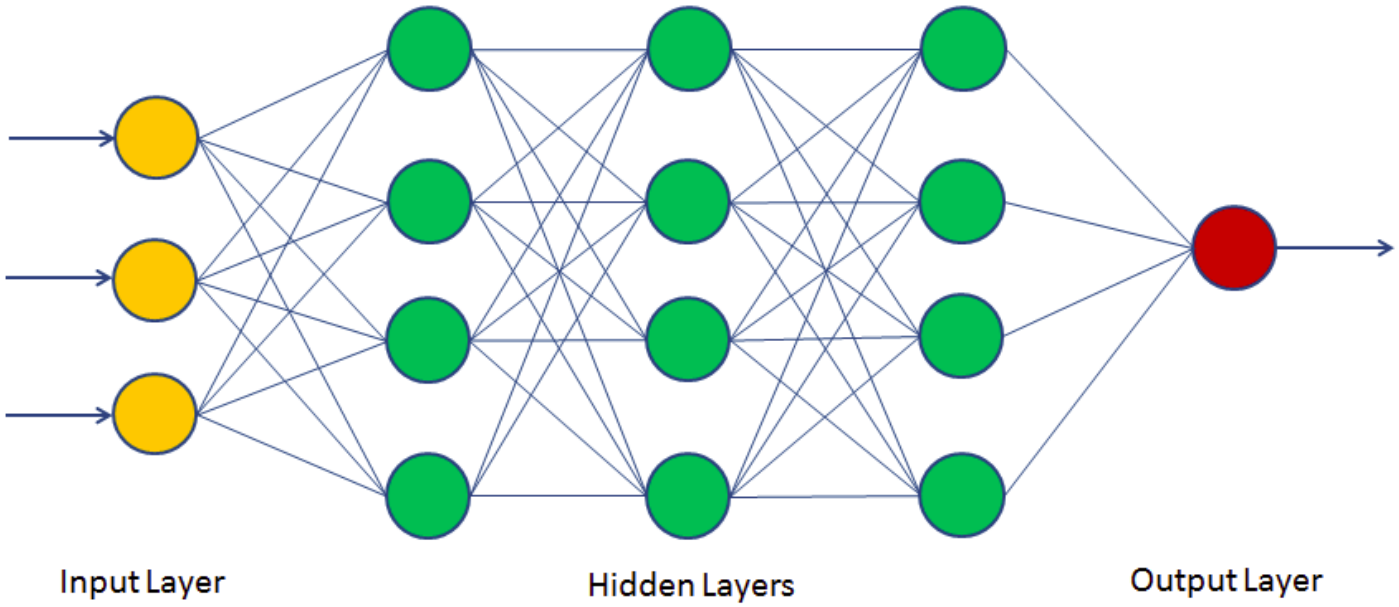

INPUTS
WEIGHTS
HIDDEN LAYER
OUTPUT
Random weights
SUM (Input X weights)
x1
x2
x3
x4
Long
Lat
Depth
Acoust
w1
w2
w3
w4
w5
w6
w7
w8
Node 1
Node 2
Activation Function
f(y)
Step, linear, ramp, or sigmoid
DEM
eq
NASC
Compare with target values and adjust weights

Basic Structure of ANN

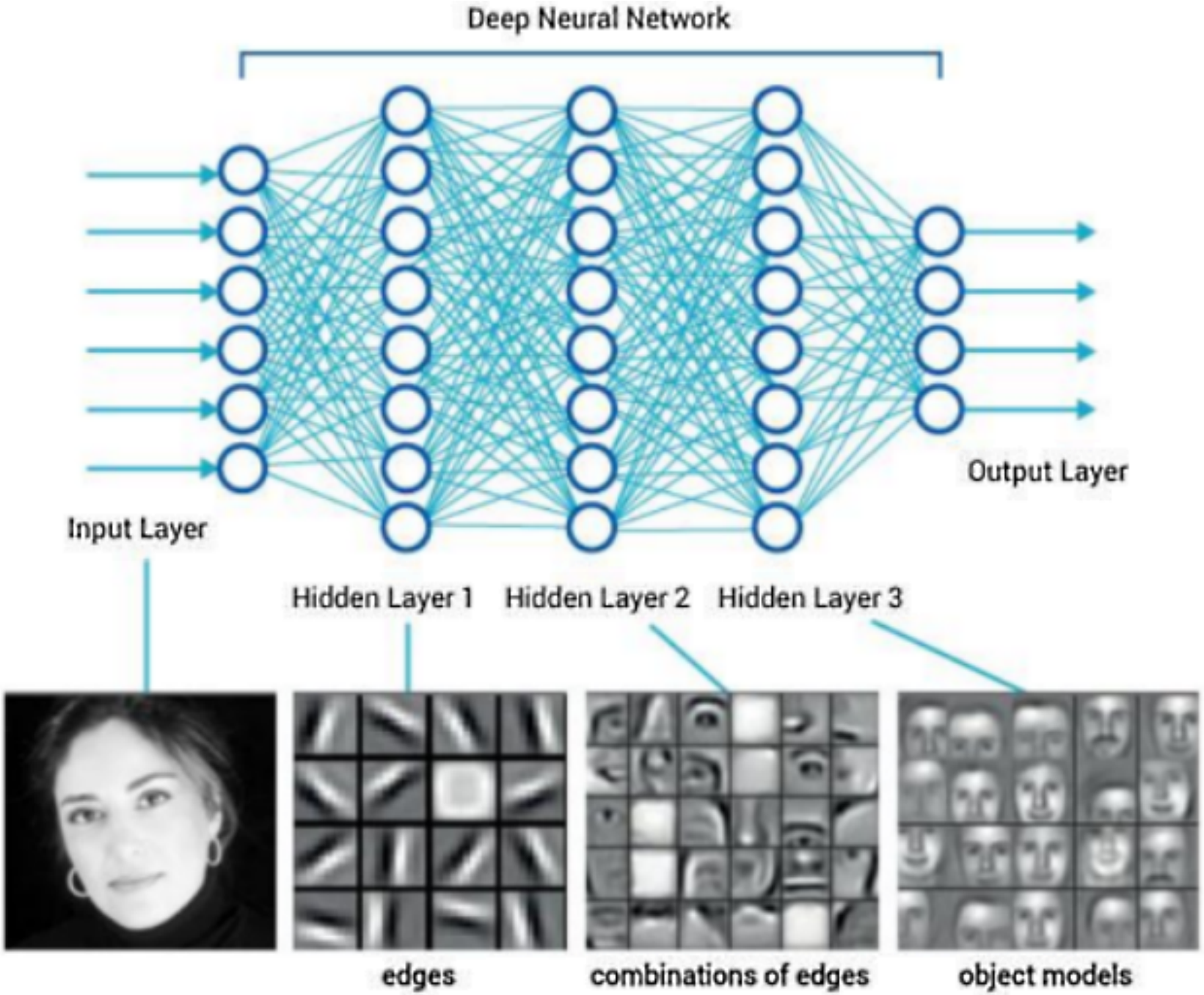


Basic Structure of ANN

- typical ANN structure for classification:
  - one output node per class
  - output value is class membership function value

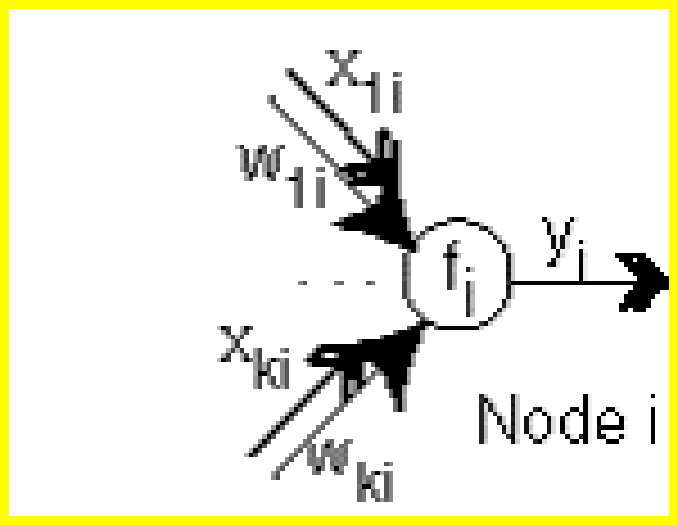


- the input nodes exist in a input layer, while the output nodes exist in an output layer
- the hidden nodes exist over one or more hidden layers


Deep Neural Network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
edges
combinations of edges
object models

ANN Node

- f_i is the activation function (also called a firing rule, processing element function or a squashing function)



- NNs often have an extra input called a bias $\Rightarrow \theta$

$$S = \sum_{i=1}^{k} w_{ji} x_{ji} - \theta$$

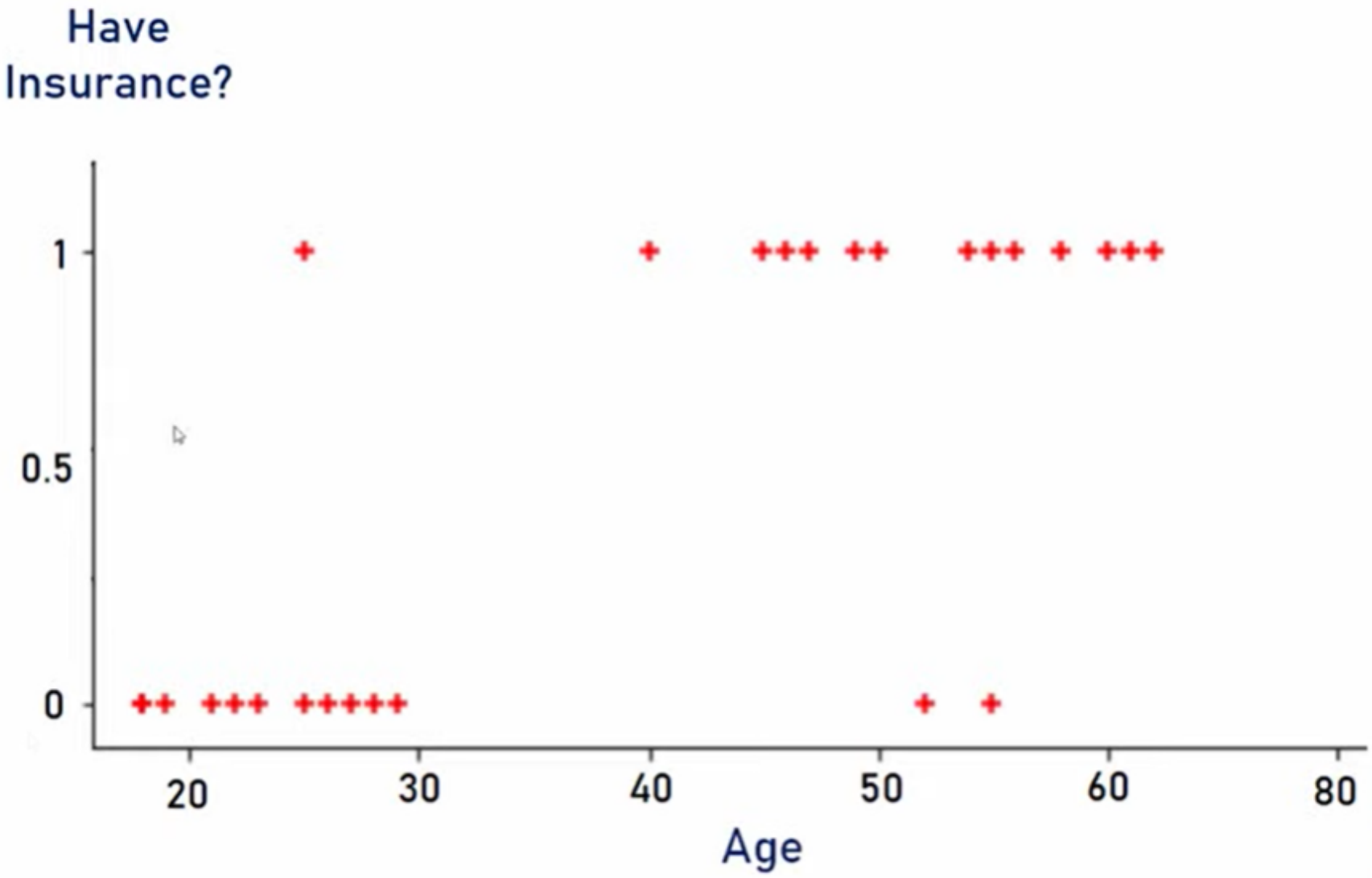
What is a neuron?

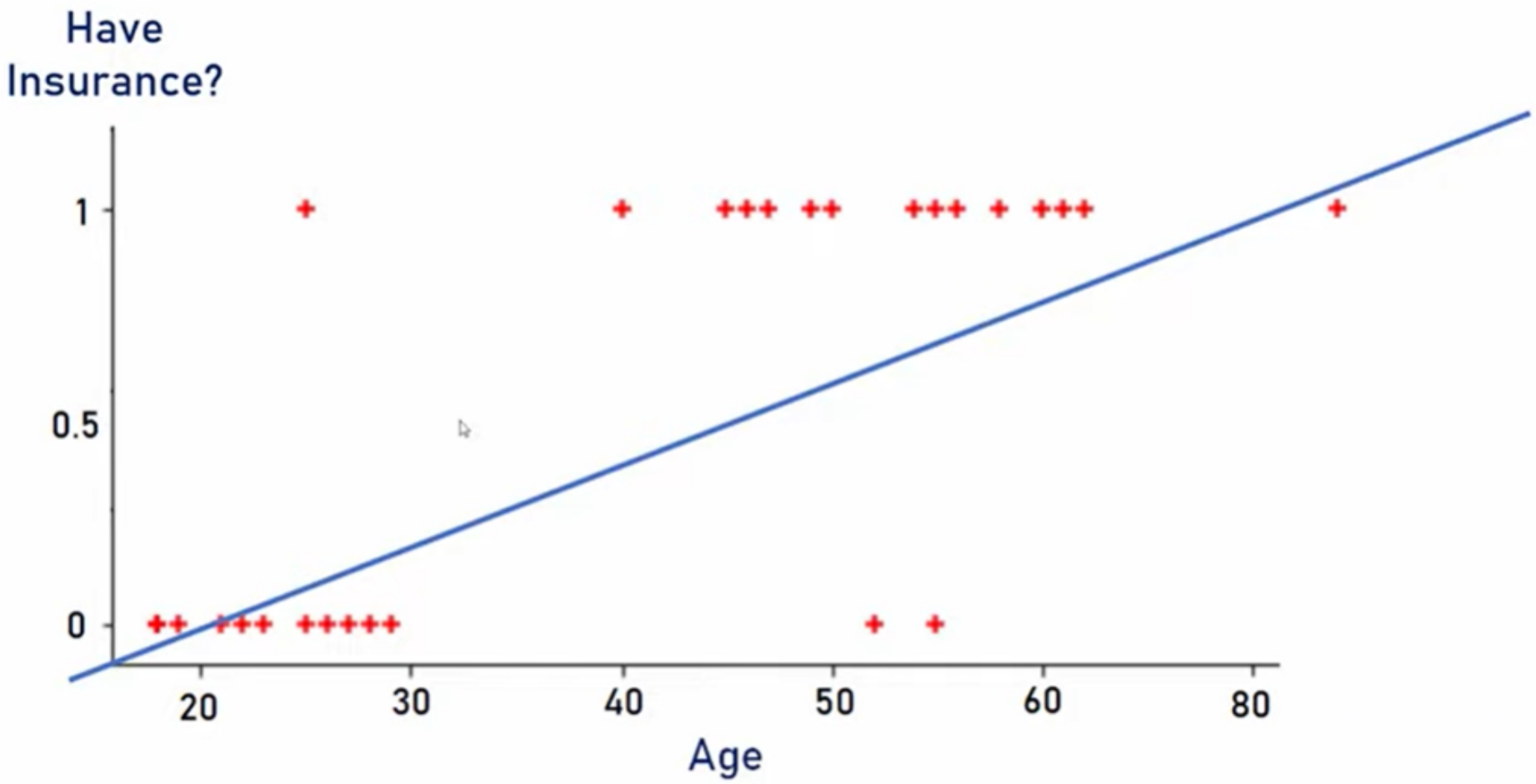
age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

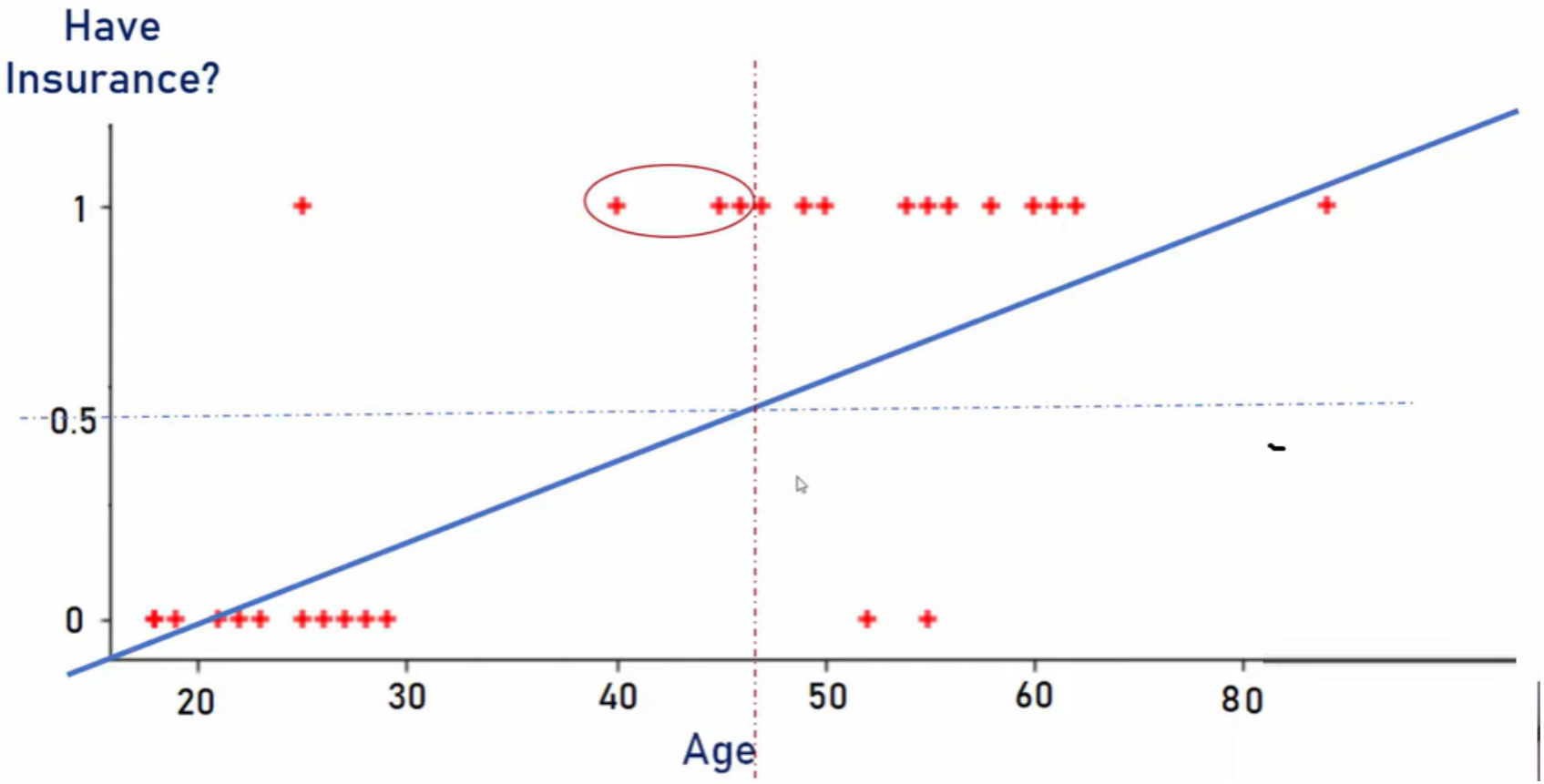
Binary Classification

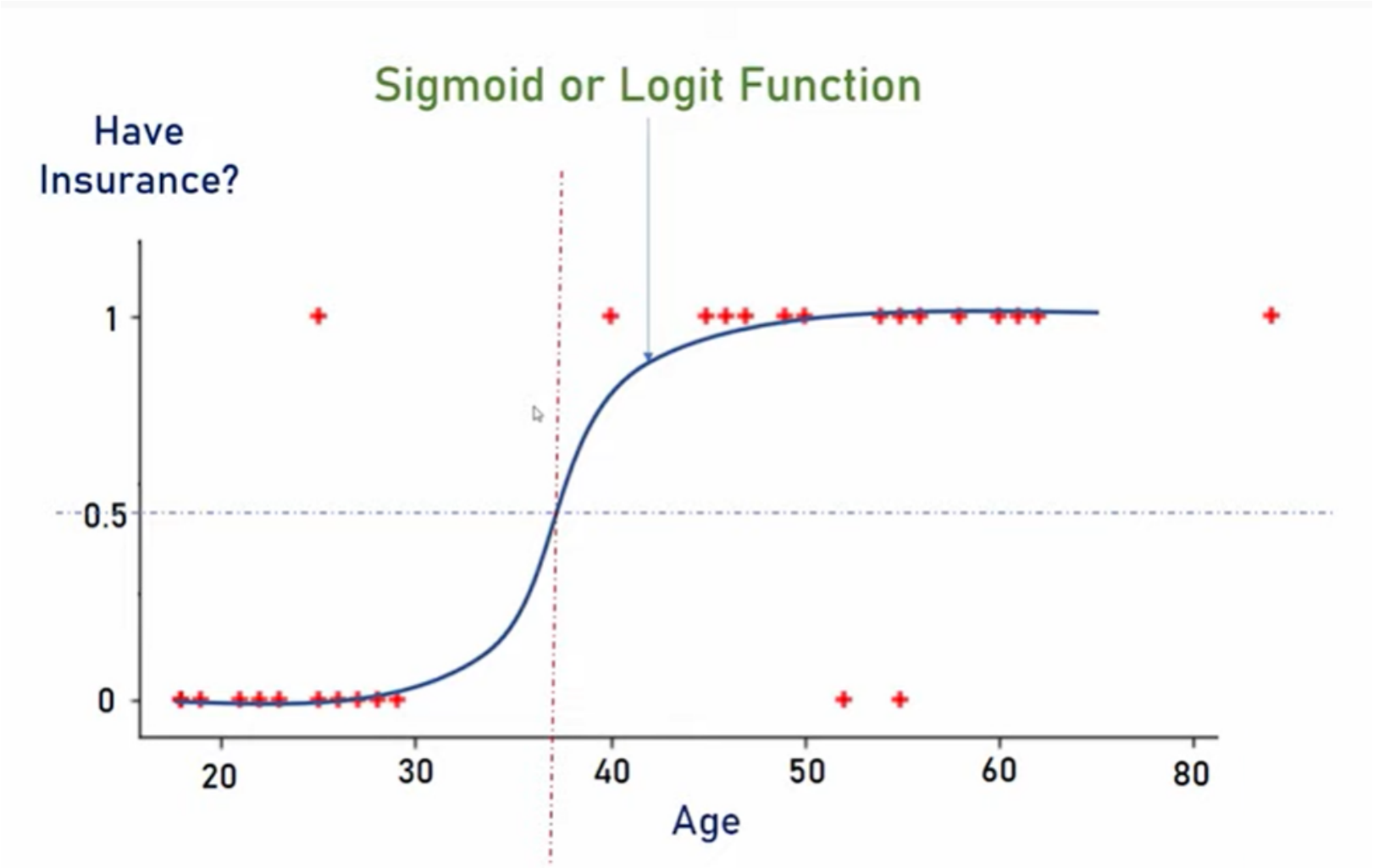
Given an age of a person, come up with a **function** that can predict if person will buy insurance or not

What is a neuron?




Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Sigmoid or Logit Function
Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

e = Euler's number ~ 2.71828

$sigmoid(200) = \frac{1}{1+2.71^{-200}}$ = almost close to 1

$sigmoid(-200) = \frac{1}{1+2.71^{200}}$ = almost close to 0

Sigmoid function converts input into range 0 to 1

Why do we need nonlinear function (activation function)?

Step 1

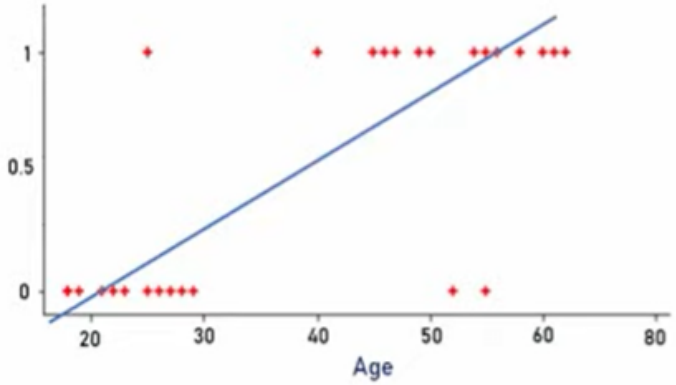
$$y = m * x + b$$

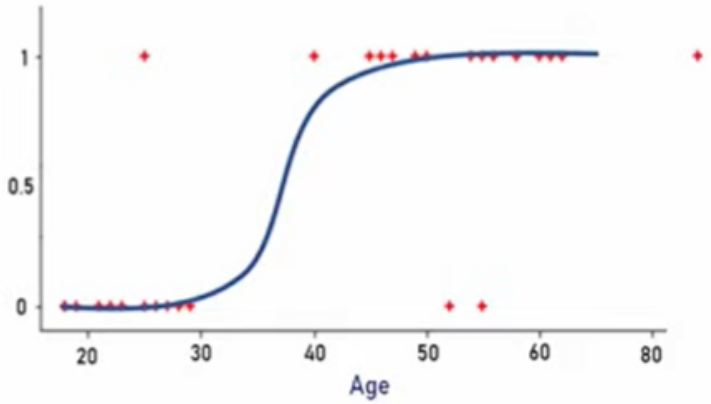
Age

Step 2

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

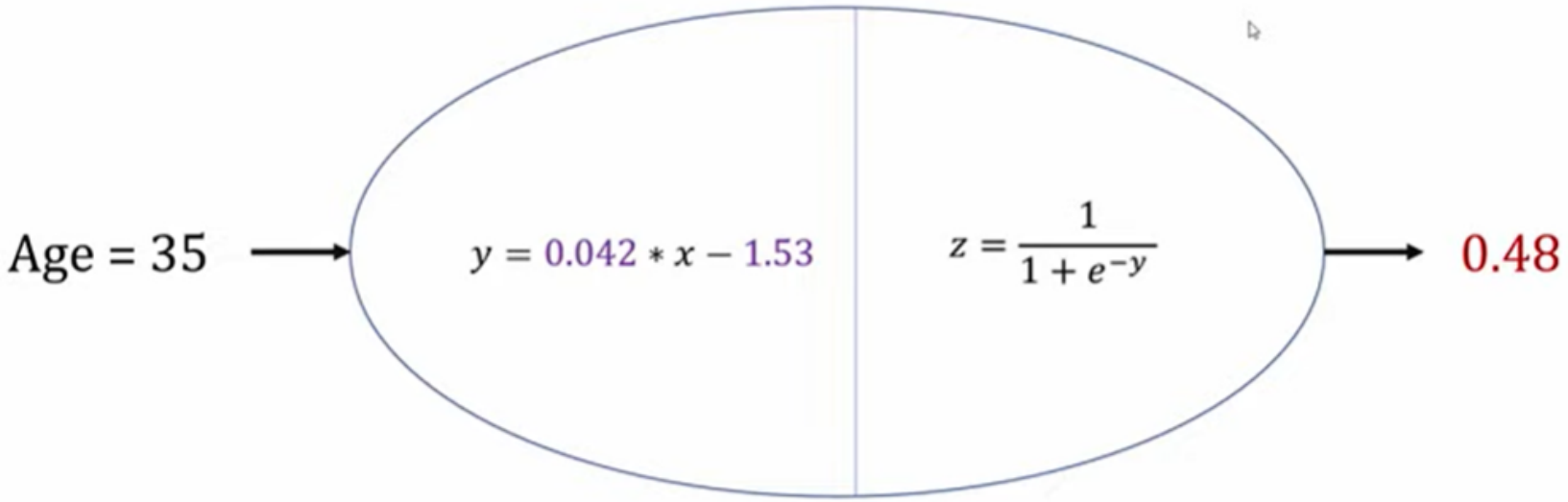
If person will buy insurance





value < 0.5 = person will not buy insurance

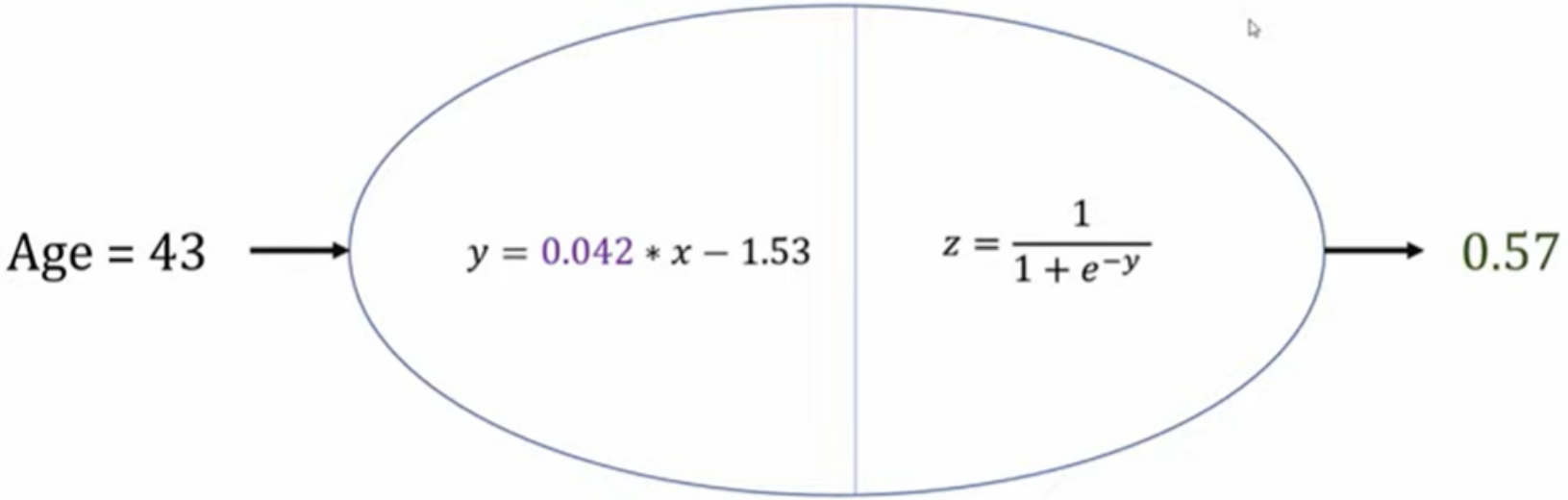
value >= 0.5 = person **will** buy insurance



What is a neuron?

value < 0.5 = person will not buy insurance

value >= 0.5 = person **will** buy insurance



$$y = 0.042 * x - 1.53$$

Age

$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 - 1.53$$

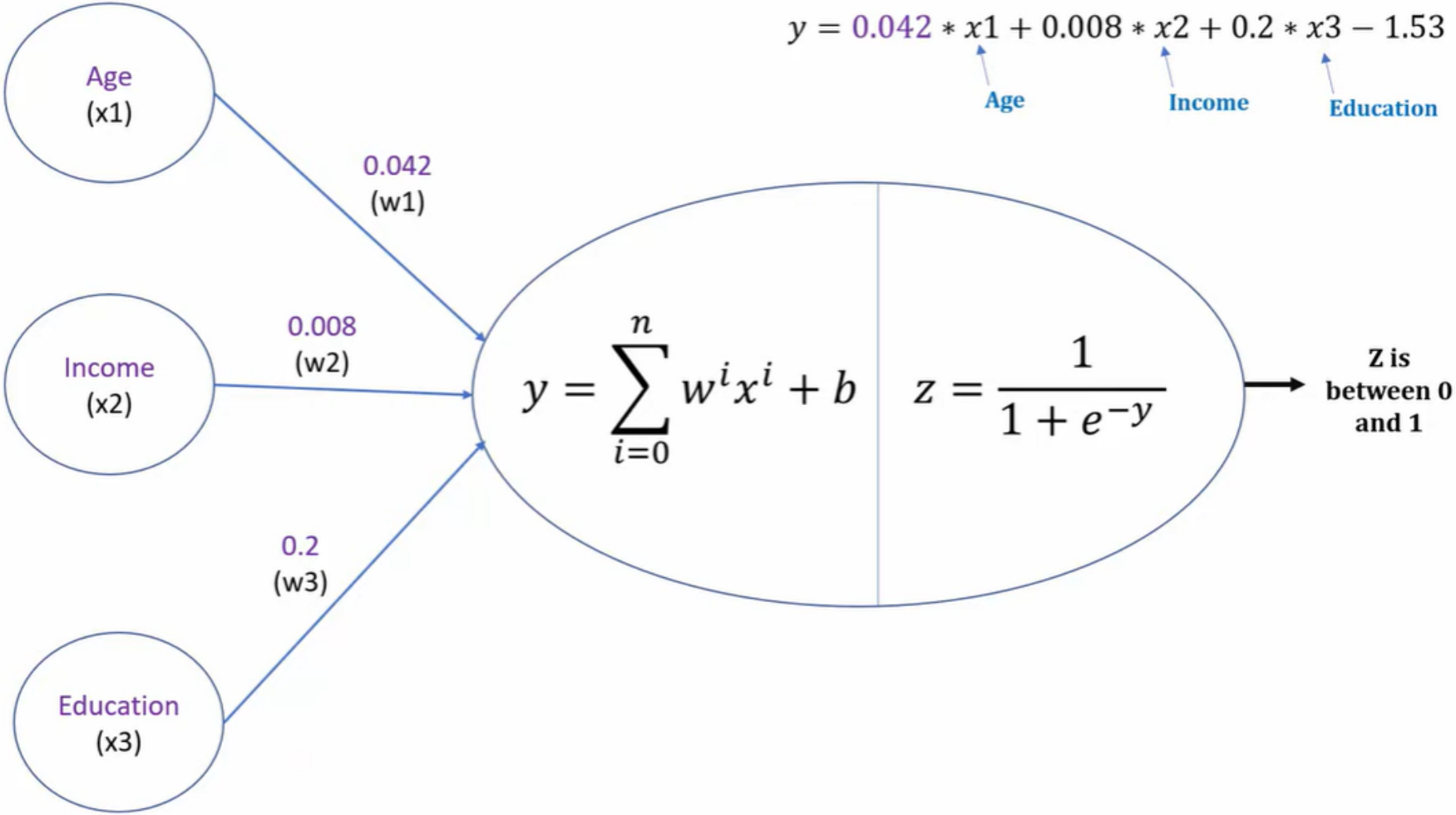
Age Income Education

$$y = w1 * x1 + w2 * x2 + w3 * x3 + b$$

$$y = \sum_{i=0}^{n} w^i x^i + b$$

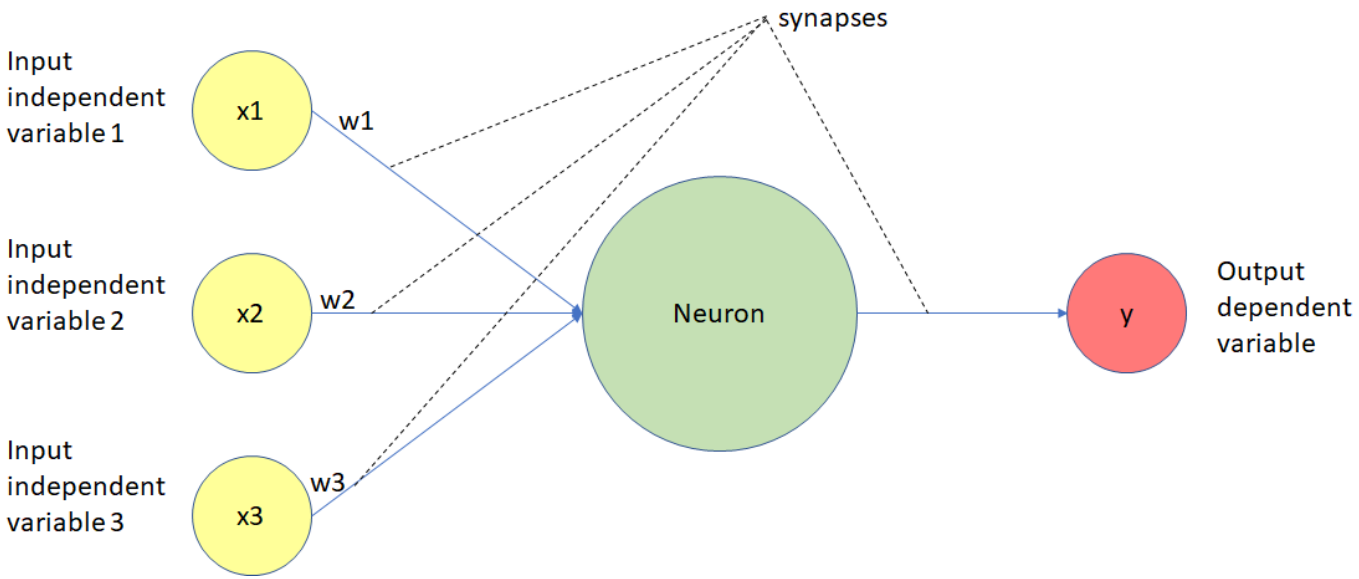
$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 - 1.53$$

Age Income Education



What is a neuron?

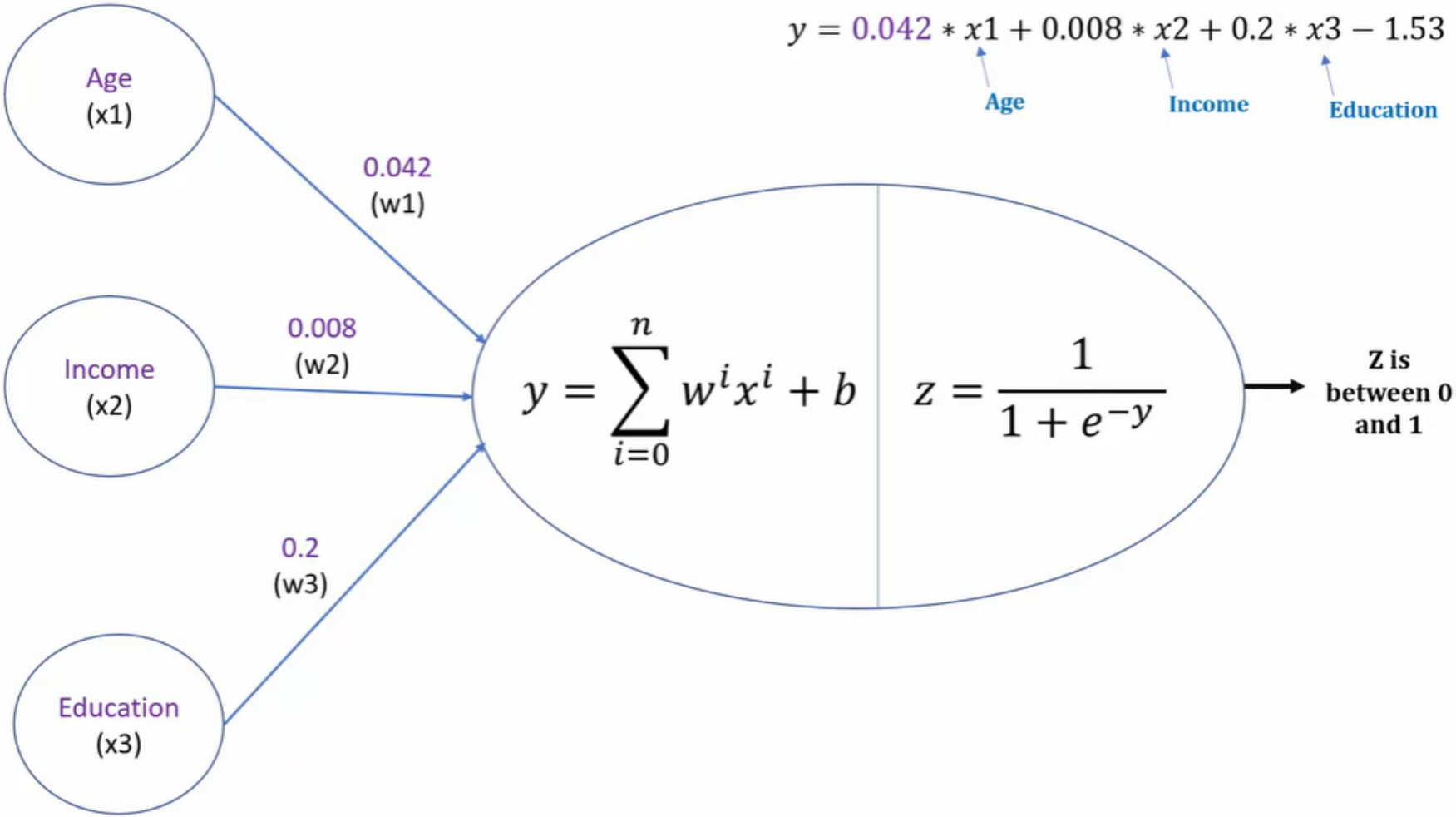
- neurons in deep learning models are nodes through which data and computations flow
- neurons work like this:
  - they receive one or more input signals
    — these input signals can come from either the raw data set or from neurons positioned at a previous layer of the neural net
  - they perform some calculations
  - they send some output signals to neurons deeper in the neural net through a synapse



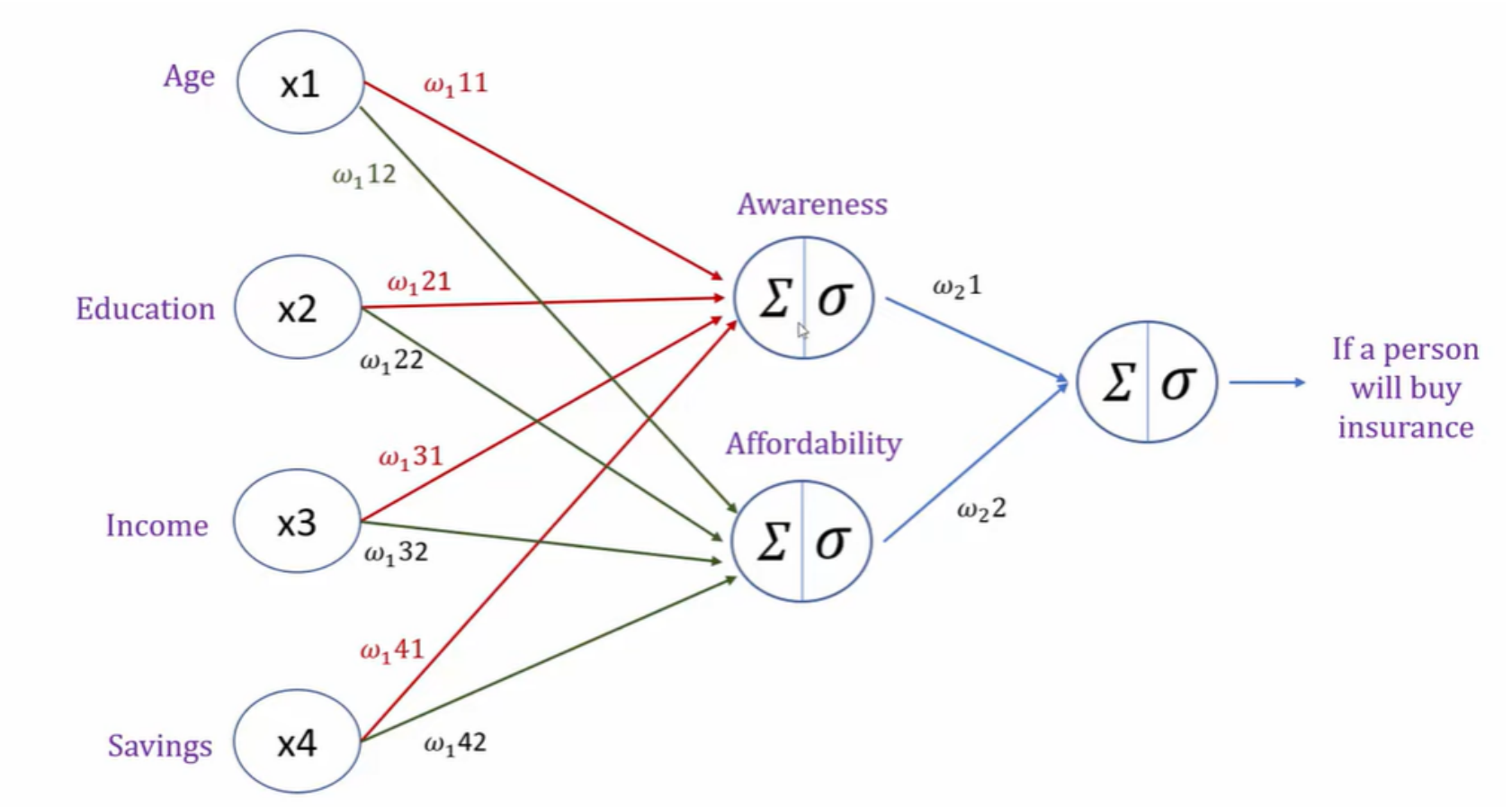
What is the role of activation functions in a neural network?

- to help the network learn complex patterns in the data
- the purpose of the activation function is to introduce non-linearity into the output of a neuron
- they decide whether a neuron should be activated or not and it is a non-linear transformation that can be done on the input before sending it to the next layer of neurons or finalizing the output

What is an Activation Function in NN?

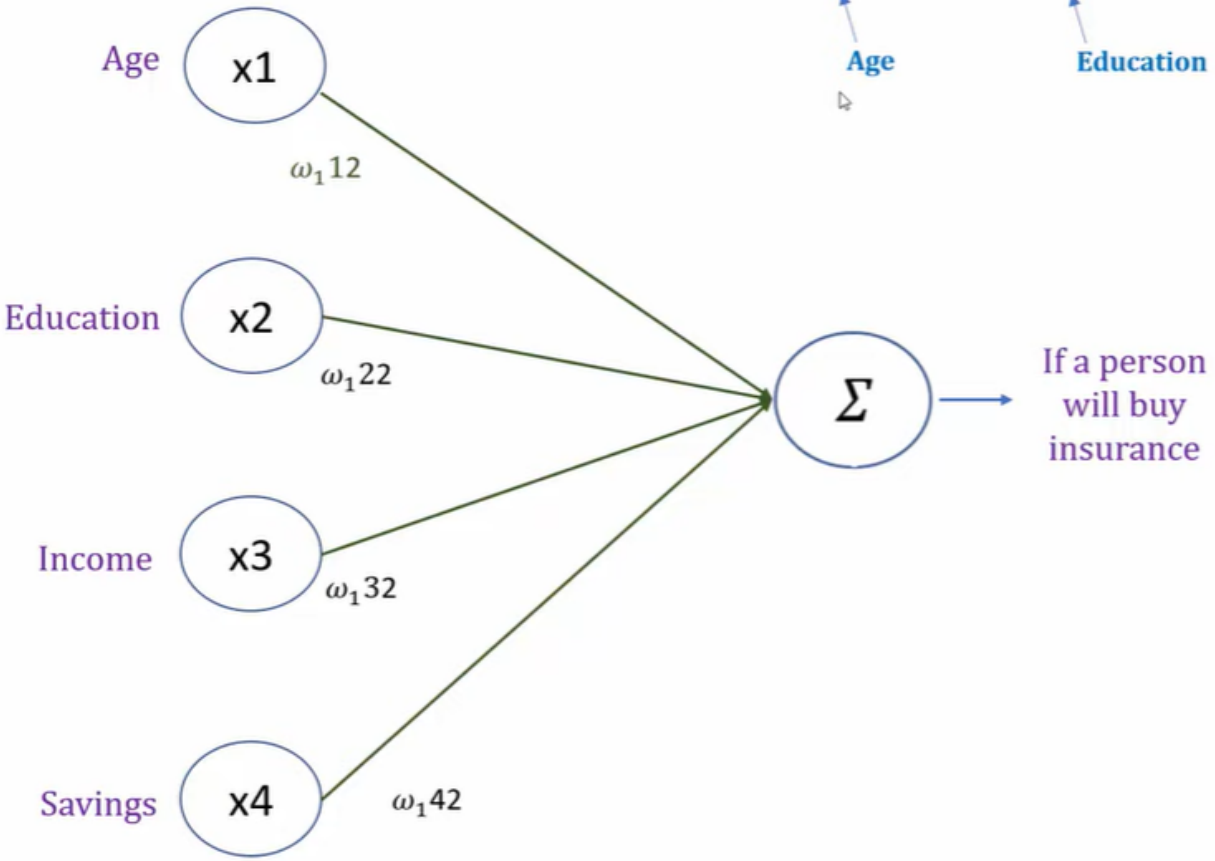


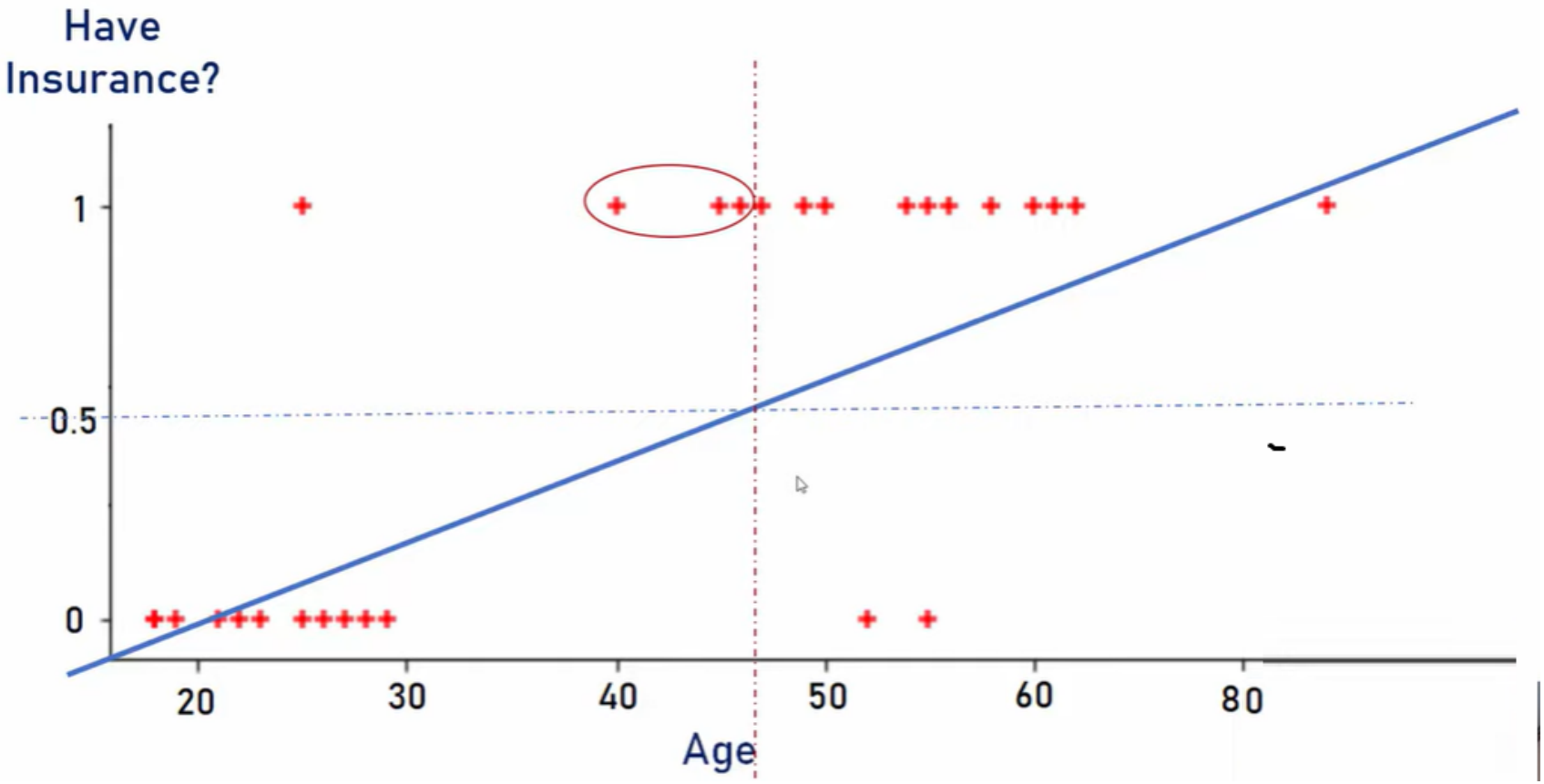
A Neural Network is combinations of basic Neurons — also called perceptrons

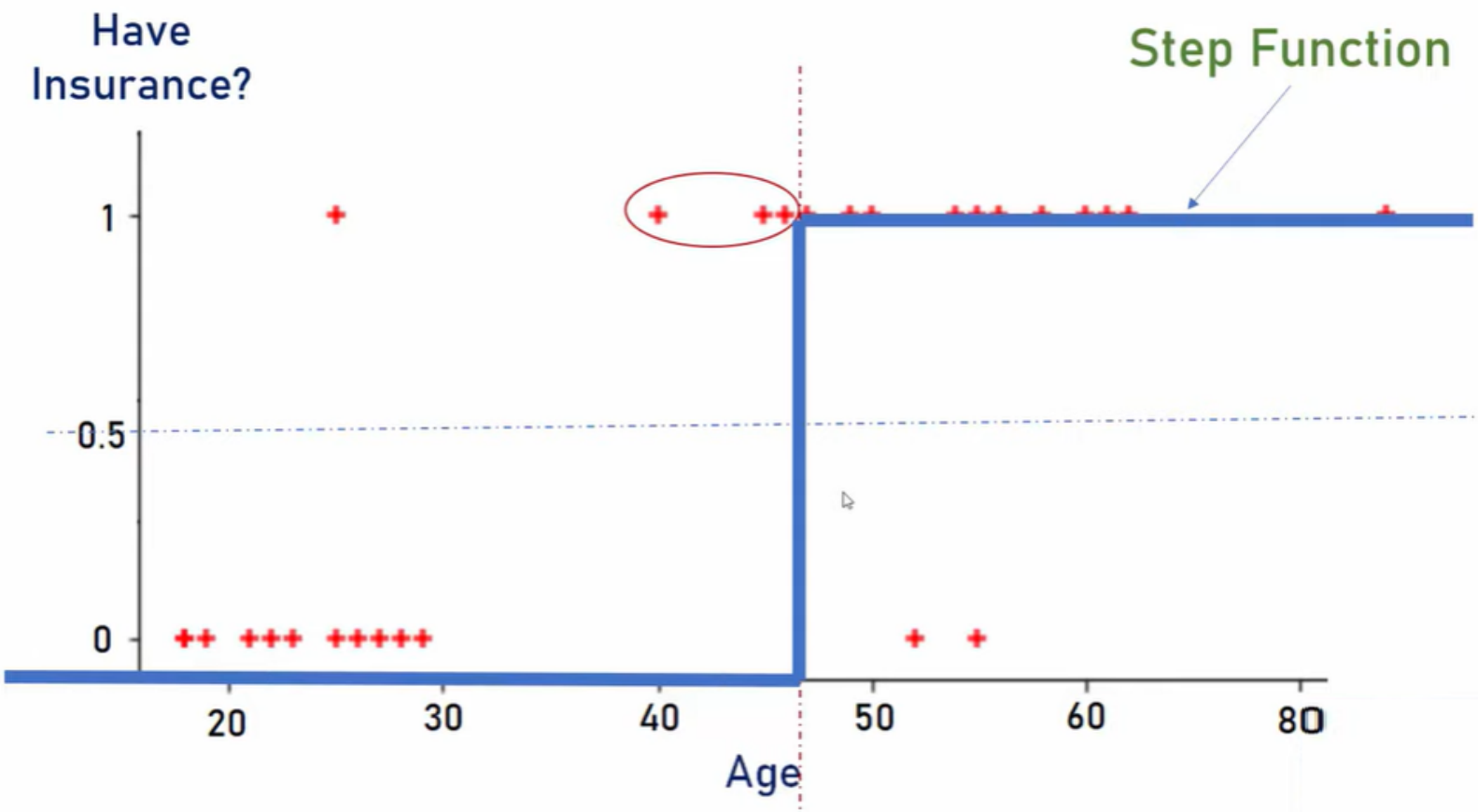

Age
x1
Education
x2
Income
x3
Savings
x4
$\omega_1 11$
$\omega_1 12$
$\omega_1 21$
$\omega_1 22$
$\omega_1 31$
$\omega_1 32$
$\omega_1 41$
$\omega_1 42$
Awareness
Affordability
Σ σ
$\omega_2 1$
$\omega_2 2$
If a person will buy insurance

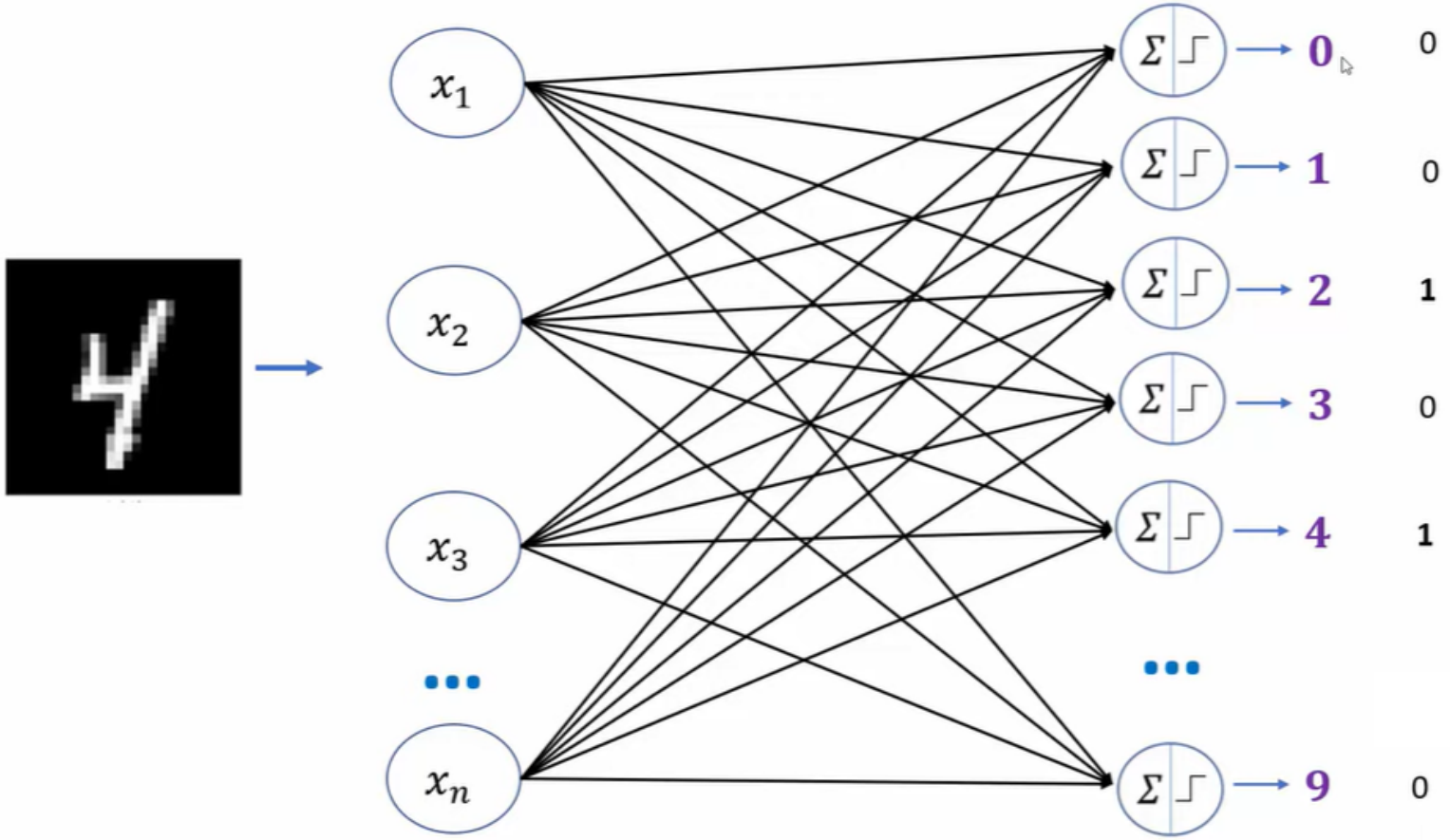
$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 + 0.02 * x4 - 1.53$$

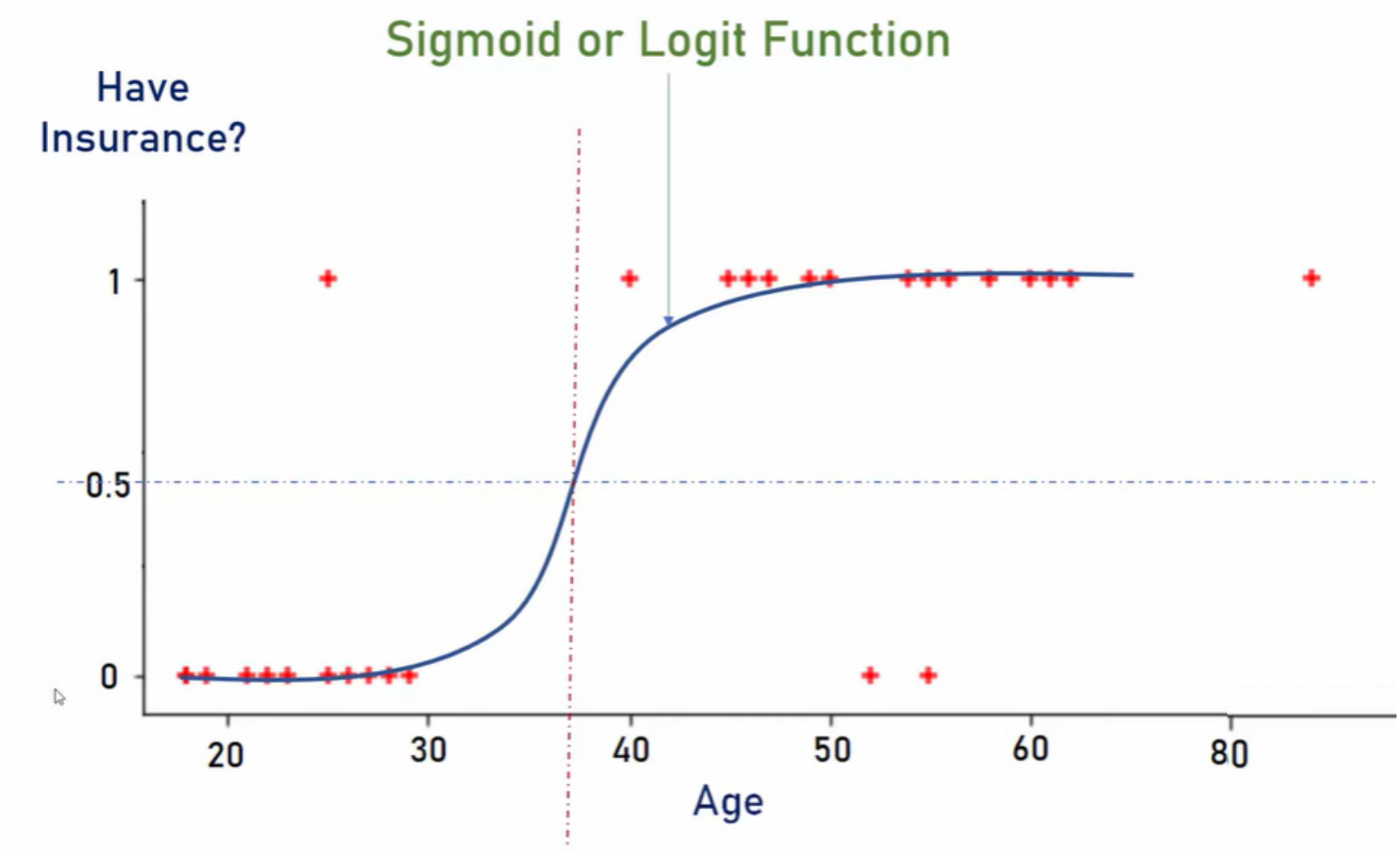
Age Education Income Savings

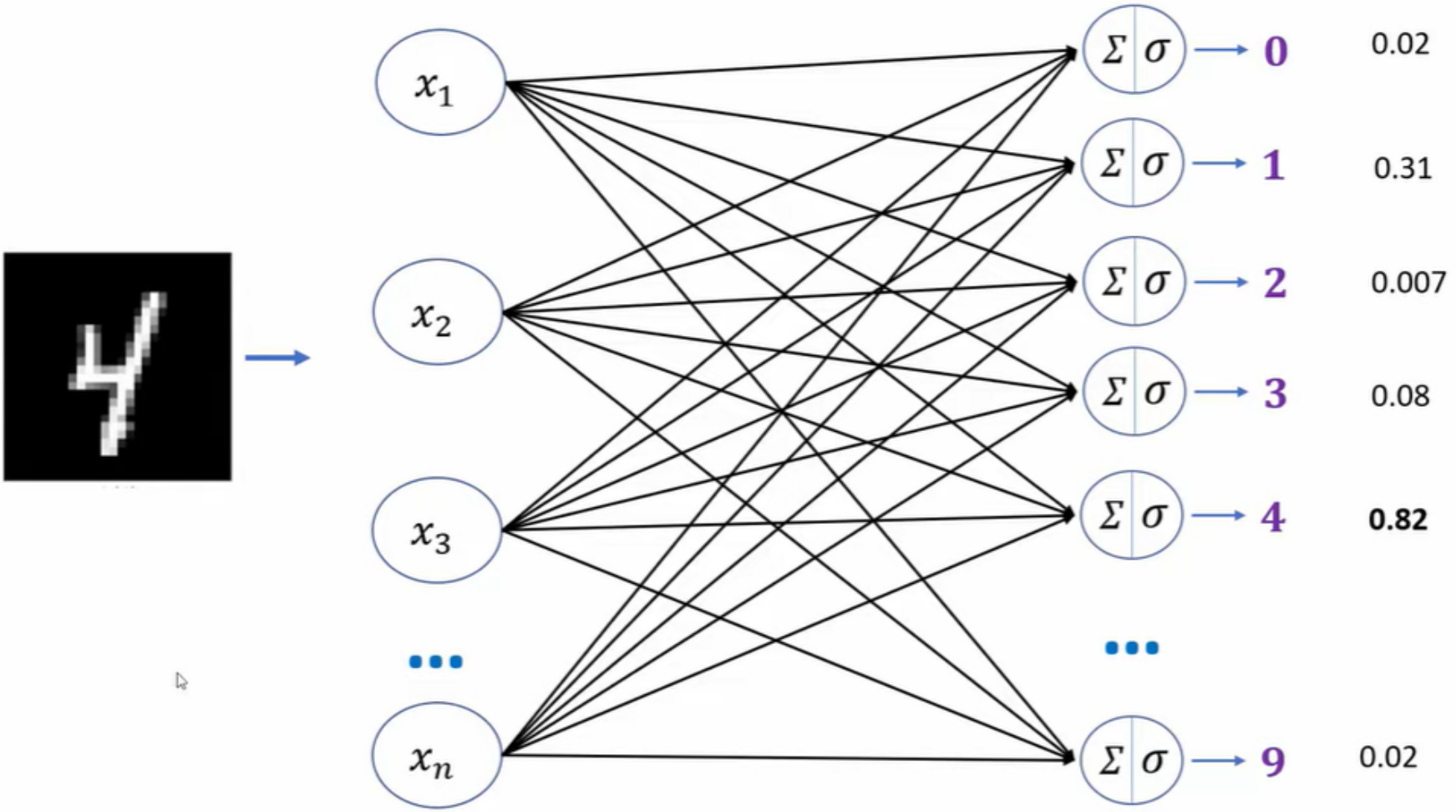



Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

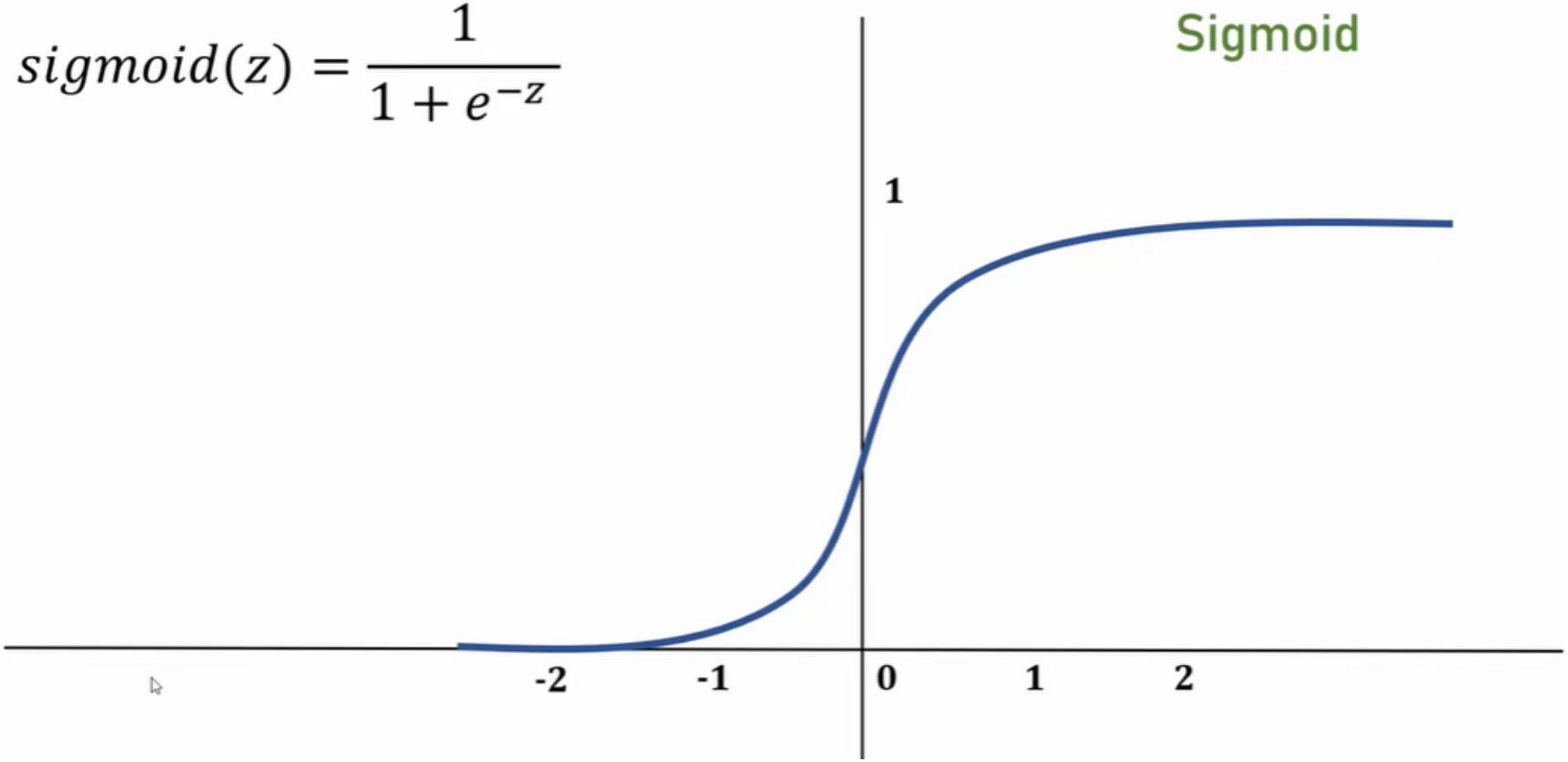

Have Insurance?
Step Function
1
0.5
0
20
30
40
50
60
80
Age


x_1
x_2
x_3
x_n
Σ
0
1
2
3
4
9
0
0
1
0
1
0

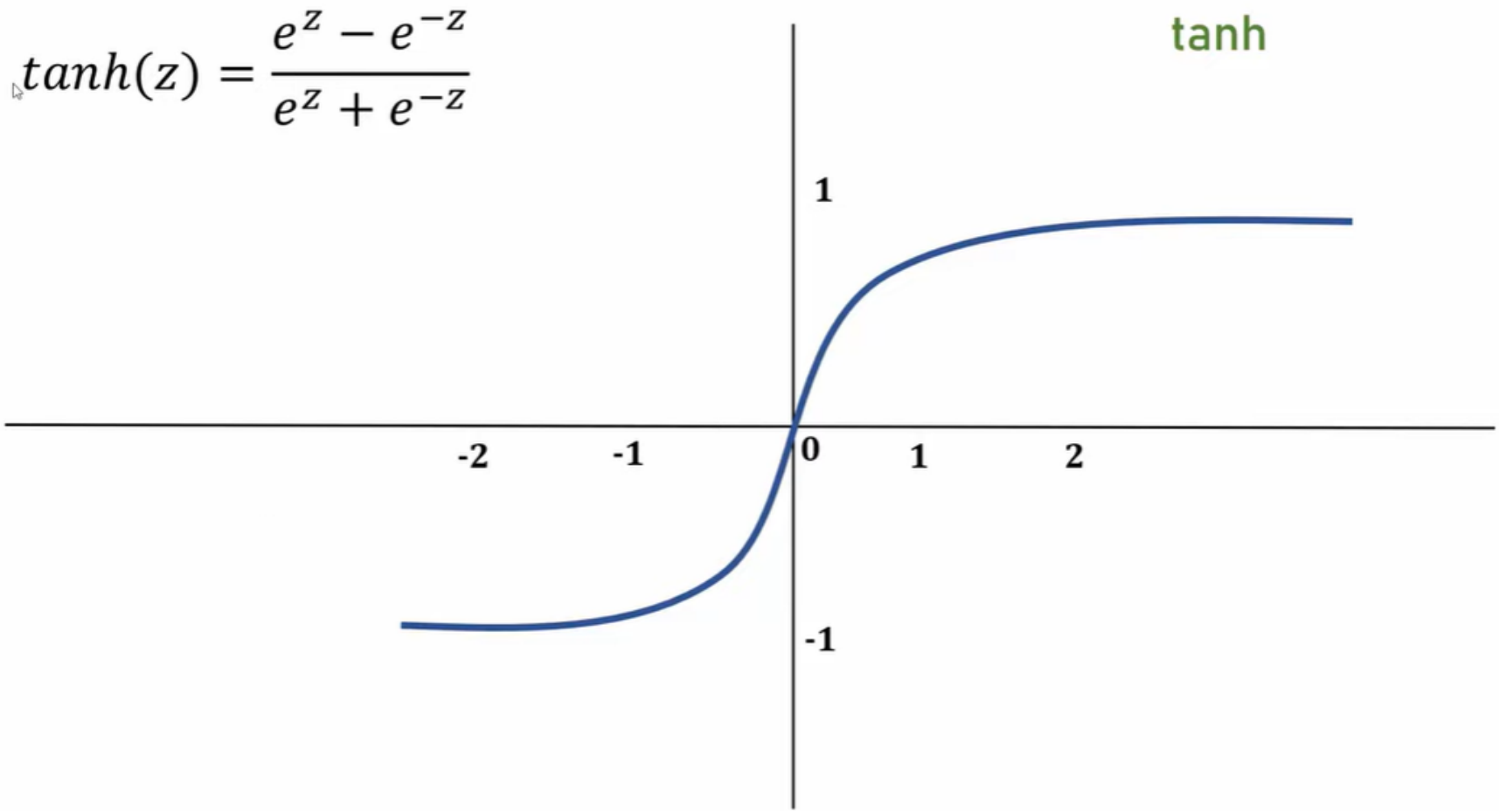

Sigmoid or Logit Function
Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


x_1
x_2
x_3
x_n
Σ σ
0
1
2
3
4
9
0.02
0.31
0.007
0.08
0.82
0.02

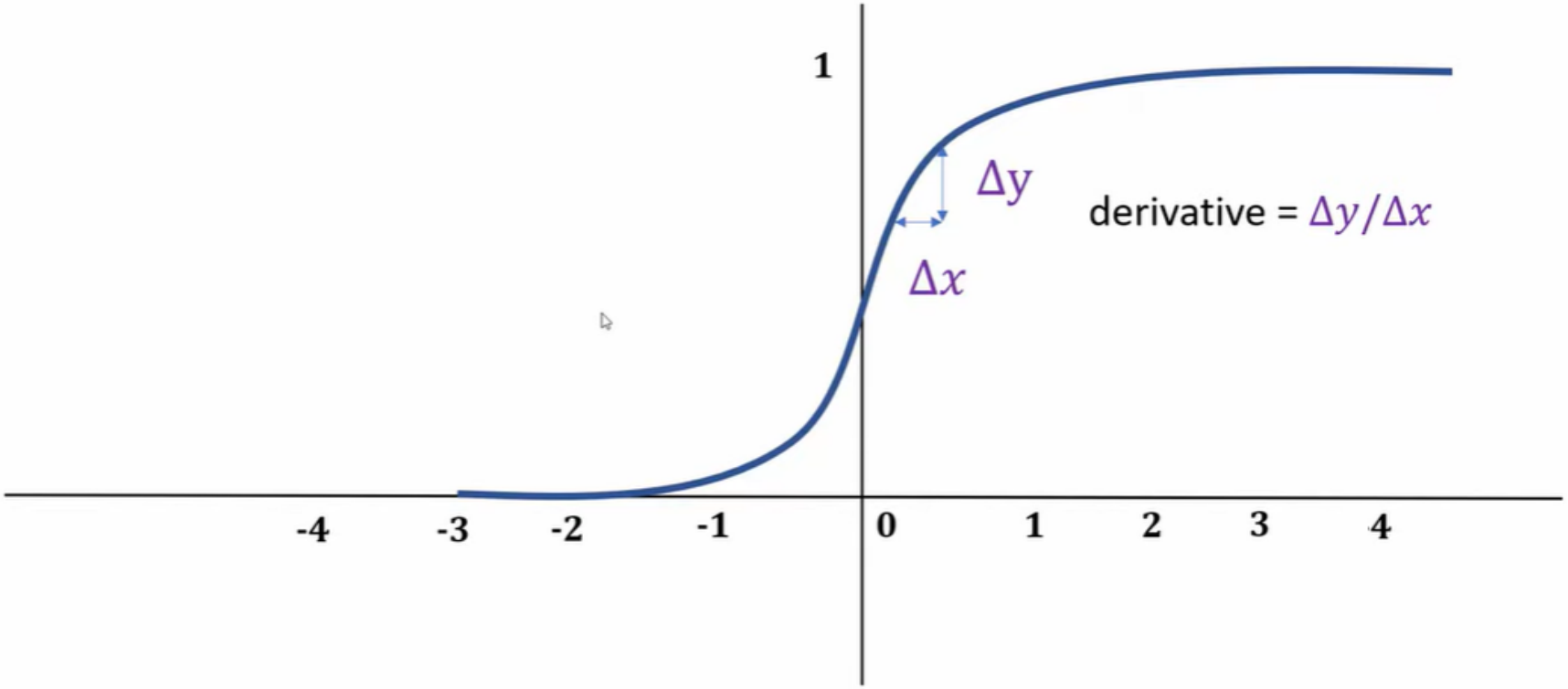
$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$



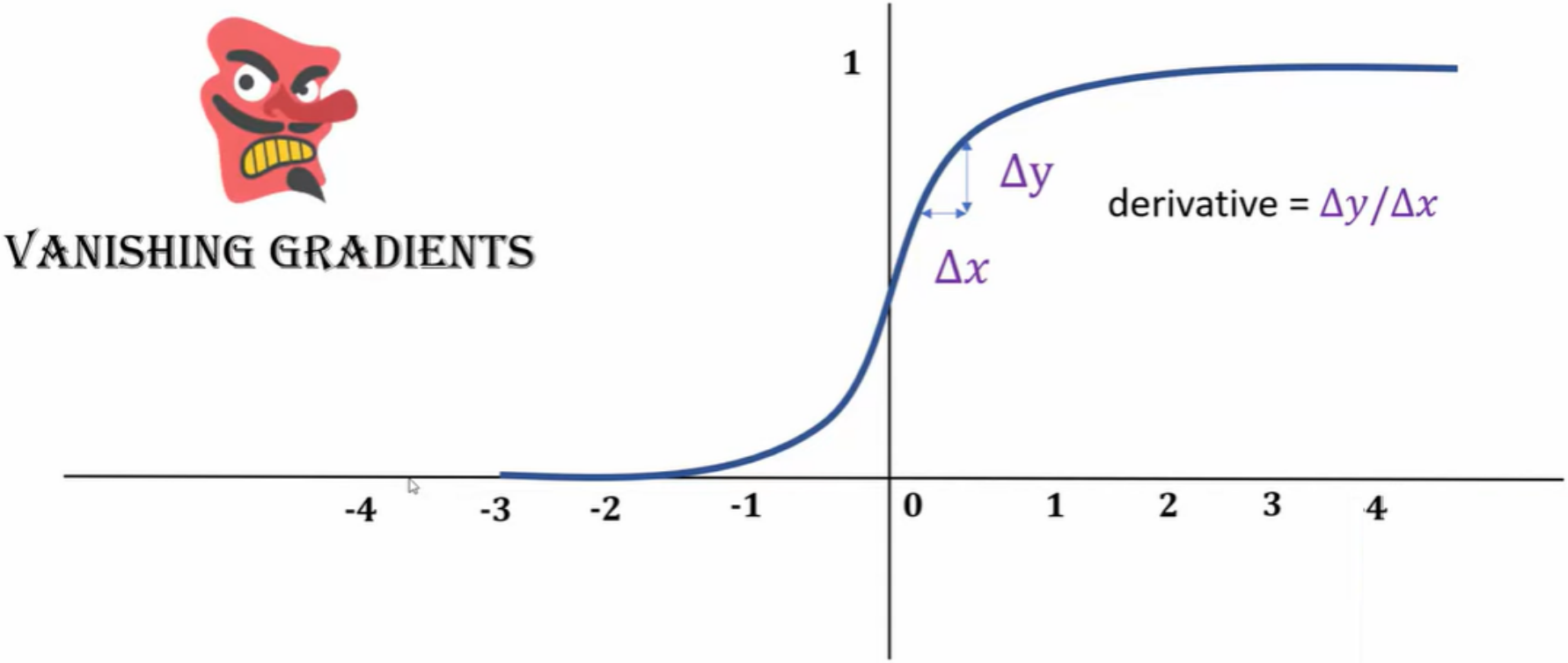
$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



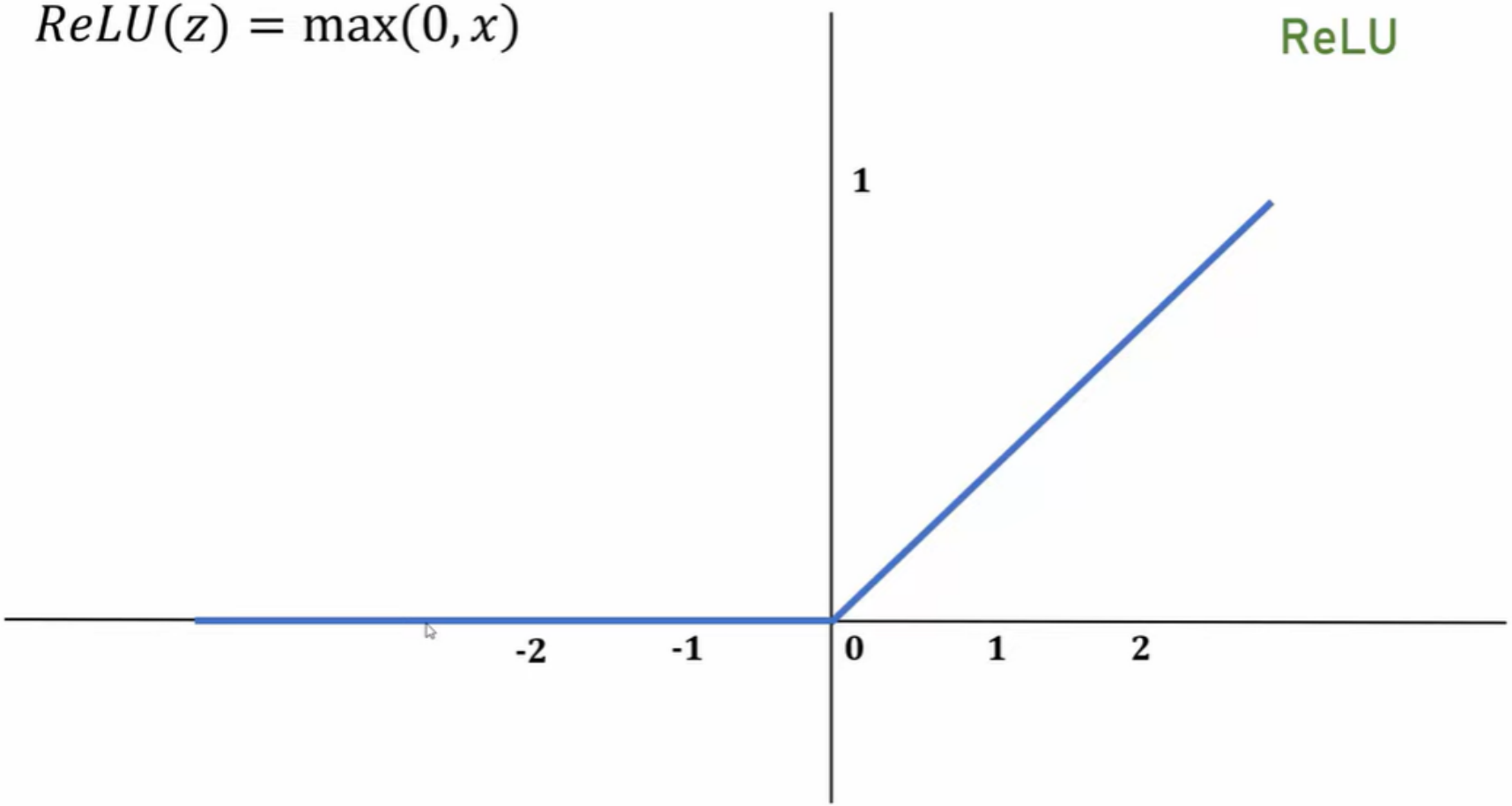
Issues with sigmoid and tanh



Issues with sigmoid and tanh

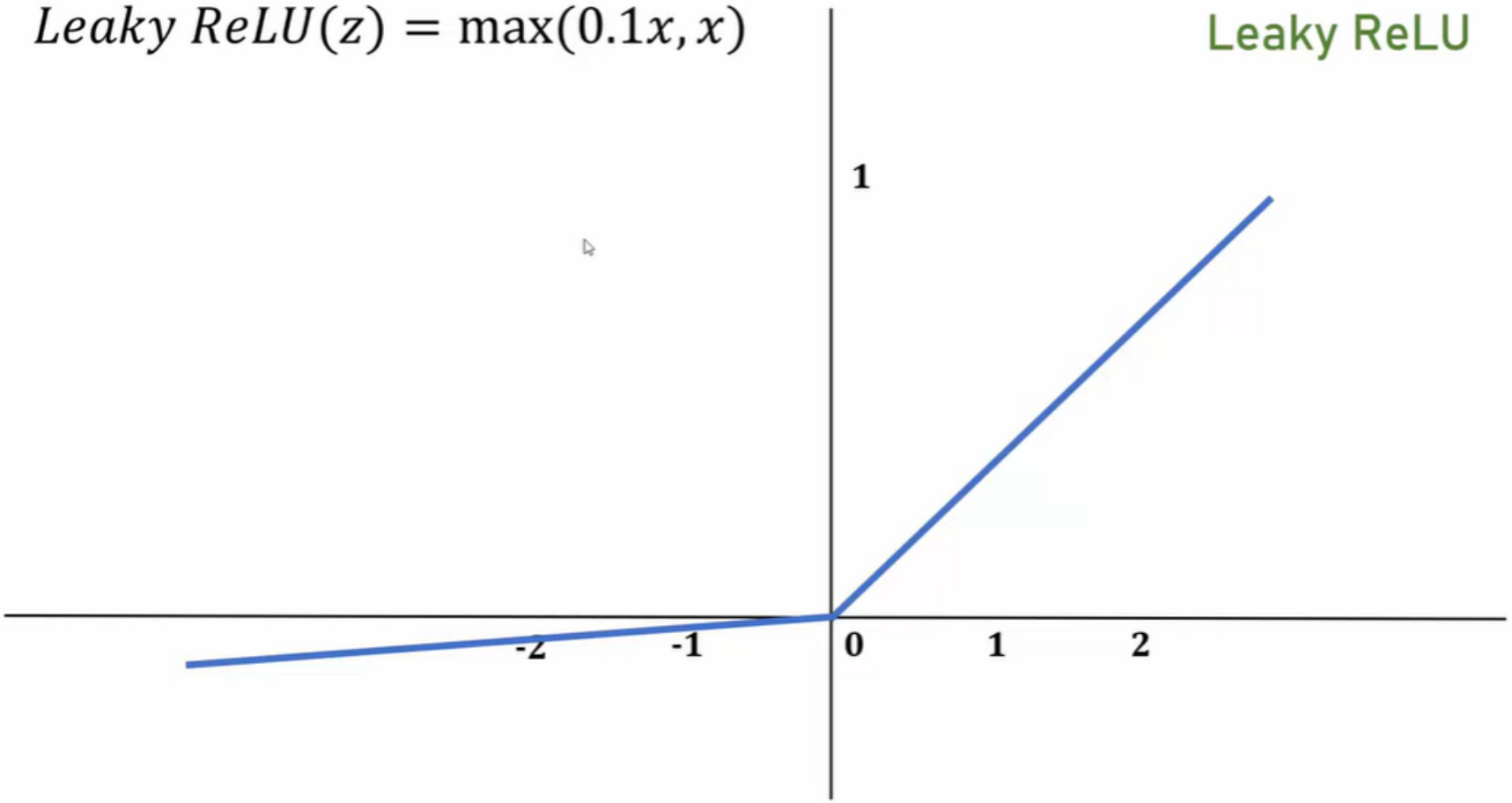


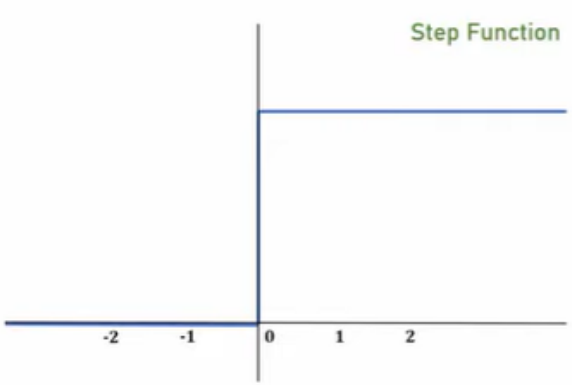
$$ReLU(z) = \max(0, x)$$

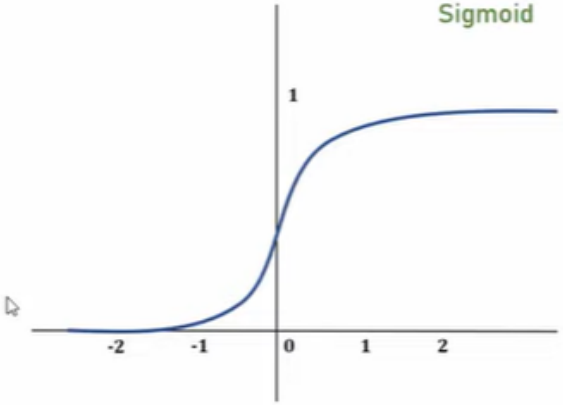


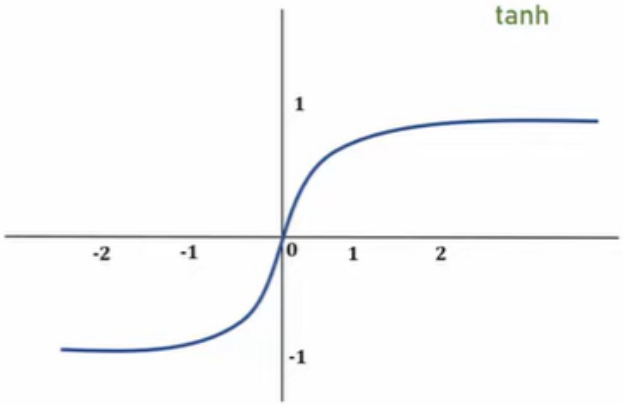
For hidden layers if you are not sure which activation function to use, just use ReLU as your default choice

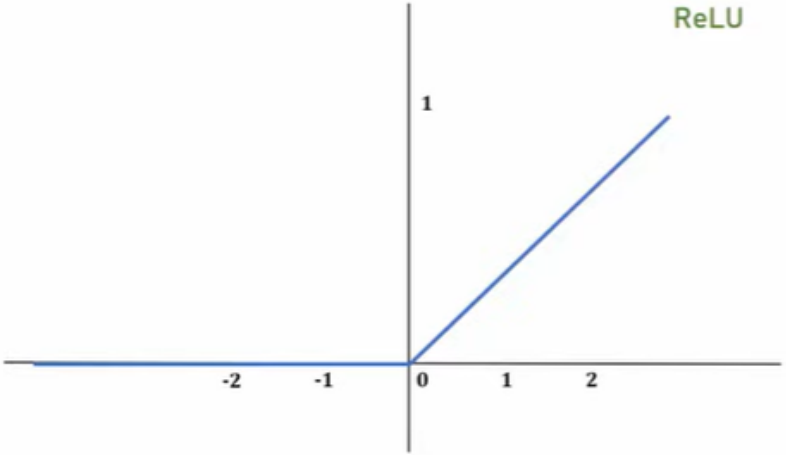
$$Leaky\ ReLU(z) = \max(0.1x, x)$$

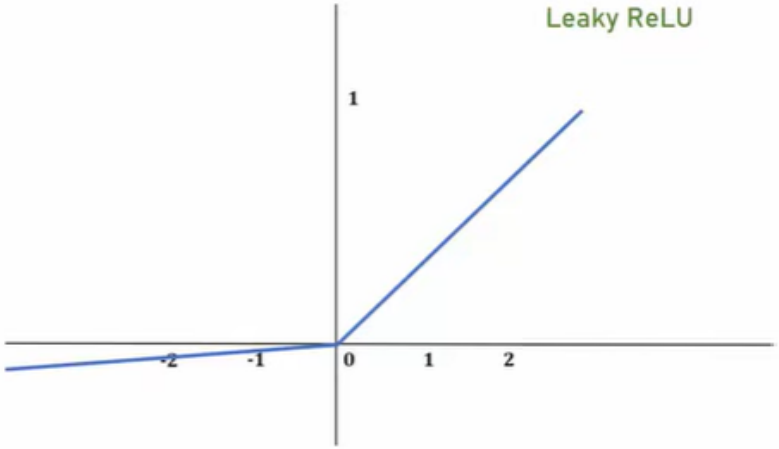



Step Function
-2
-1
0
1
2


Sigmoid
1
-2
-1
0
1
2


tanh
1
-2
-1
0
1
2
-1


ReLU
1
-2
-1
0
1
2


Leaky ReLU
1
-2
-1
0
1
2

Activation Functions

Linear:

$$f_i(S) = c\,S$$

Threshold or Step:

$$f_i(S) = \left\{ \begin{array}{ll} 1 & if S > T \\ 0 & otherwise \end{array} \right\}$$

Ramp:

$$f_i(S) = \left\{ \begin{array}{ll} 1 & if S > T_2 \\ \frac{S-T_1}{T_2-T_1} & if T_1 \leq S \leq T_2 \\ 0 & if S < T_1 \end{array} \right\}$$

Sigmoid:

$$f_i(S) = \frac{1}{(1+e^{-c\,S})}$$

Hyperbolic Tangent:

$$f_i(S) = \frac{(1-e^{-S})}{(1+e^{-c\,S})}$$

Gaussian:

$$f_i(S) = e^{\frac{-S^2}{v}}$$

DNN Activation Functions in Python

Function Type	*Equation*	*Derivative*
Linear	$f(x) = ax + c$	$f'(x) = a$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)\,(1 - f(x))$
TanH	$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
Parametric ReLU	$f(x) = \begin{cases} \alpha x \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
ELU	$f(x) = \begin{cases} \alpha(e^{x} - 1) \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$

Use sigmoid in output layer. All other places try to use tanh

```
In [1]: import math

        def sigmoid(x):
          return 1 / (1 + math.exp(-x))

In [2]: sigmoid(100)

Out[2]: 1.0

In [3]: sigmoid(1)

Out[3]: 0.7310585786300049

In [4]: sigmoid(-56)

Out[4]: 4.780892883885469e-25

In [5]: sigmoid(0.5)

Out[5]: 0.6224593312018546
```

```
In [6]: def tanh(x):
          return (math.exp(x) - math.exp(-x)) / (math.exp(x) + math.exp(-x))

In [7]: tanh(-56)
Out[7]: -1.0

In [8]: tanh(50)
Out[8]: 1.0

In [9]: tanh(1)
Out[9]: 0.7615941559557649

In [10]: def relu(x):
             return max(0,x)

In [11]: relu(-7)
Out[11]: 0

In [12]: relu(1)
Out[12]: 1
```

```
In [10]: def relu(x):
             return max(0,x)

In [15]: relu(-100)
Out[15]: 0

In [14]: relu(8)
Out[14]: 8

In [16]: def leaky_relu(x):
             return max(0.1*x,x)

In [17]: leaky_relu(-100)
Out[17]: -10.0

In [18]: leaky_relu(8)
Out[18]: 8
```

Softmax function

Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. Thus sigmoid is widely used for binary classification problems.

The softmax function can be used for multiclass classification problems. This function returns the probability for a datapoint belonging to each individual class. Here is the mathematical expression of the same-

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

While building a network for a multiclass problem, the output layer would have as many neurons as the number of classes in the target. For instance if you have three classes, there would be three neurons in the output layer. Suppose you got the output from the neurons as [1.2 , 0.9 , 0.75].

Categorical Cross Entropy Loss

$$\text{Cross Entropy} = -\sum_{i=1}^{n}\sum_{i=c}^{C} t_{ic} \cdot log(y_{ic})$$

Here,

- T_{ic} = Target Output
- y_{ic} = Soft Max Output
- C= = Number of Classes
- N = Number of Data Samples

$$\text{Soft Max o/p} = y_c = P(t = c|z) = \frac{e^{z_c}}{\sum_{d=1}^{C} e^{z_d}} \quad \text{for } c = 1 \cdots C$$

Here,

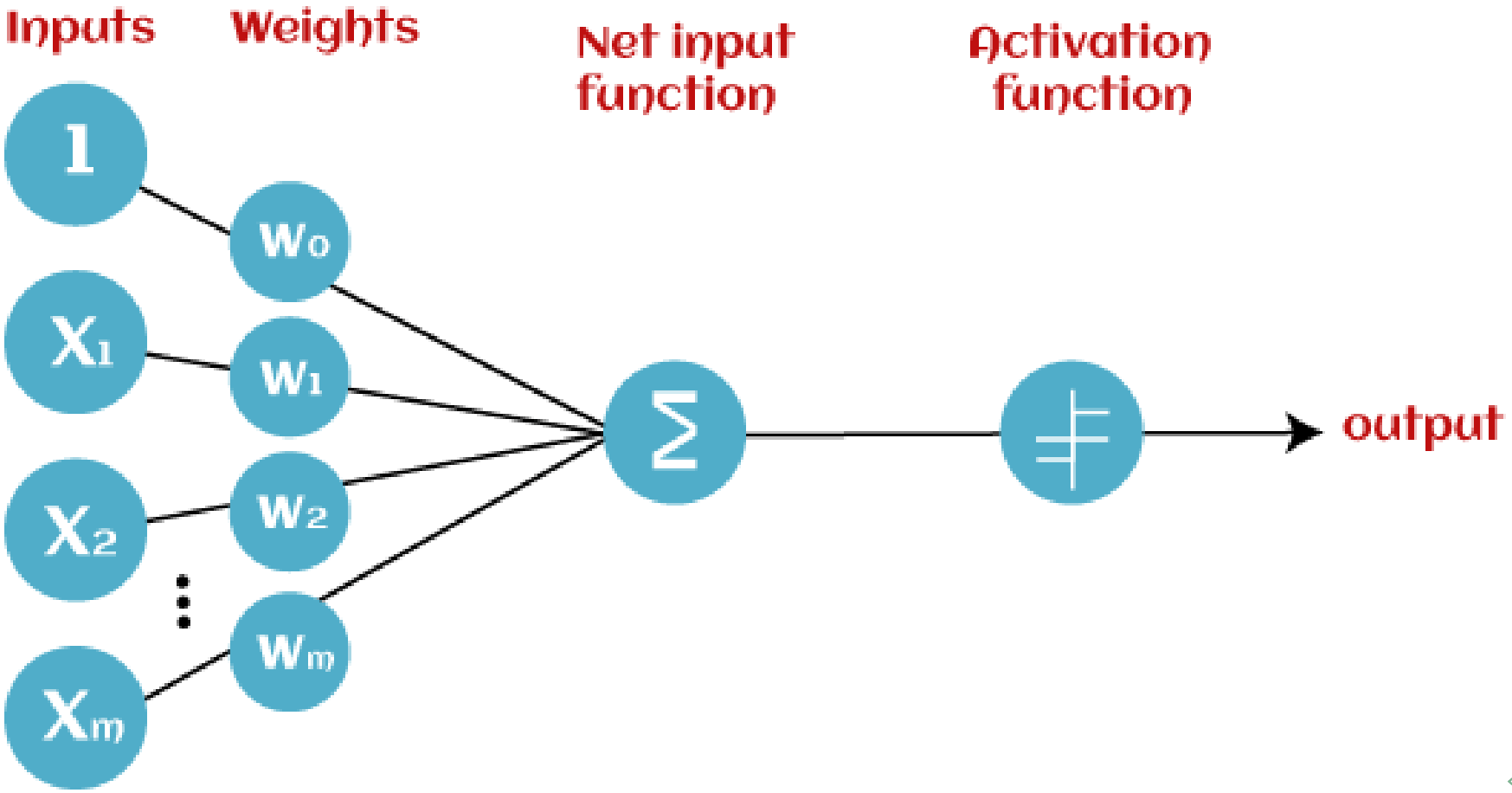
- y_c = probability of current output belonging to class c
- Numerator = exponential of weighted sum o/p of class c
- Denominator = sum of exponential of weighted sum o/p of classes 1 to C

Perceptron

- The Perceptron is the most basic and oldest form of neural networks.
- It consists of just 1 neuron which takes the input and applies activation function on it to produce a binary output.
- It doesn't contain any hidden layers and can only be used for binary classification tasks.
- Disadvantages of Perceptron
  - Perceptrons can only learn linearly separable problems such as boolean AND problem.
  - For non-linear problems such as the boolean XOR problem, it does not work.

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

- Single-layer Perceptron Model
- Multi-layer Perceptron model

Single-layer Perceptron Model

- This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model.
- The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.
- In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.
- If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

Multi-Layered Perceptron Model

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers. The multi-layer perceptron model is also known as the *Backpropagation algorithm*, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

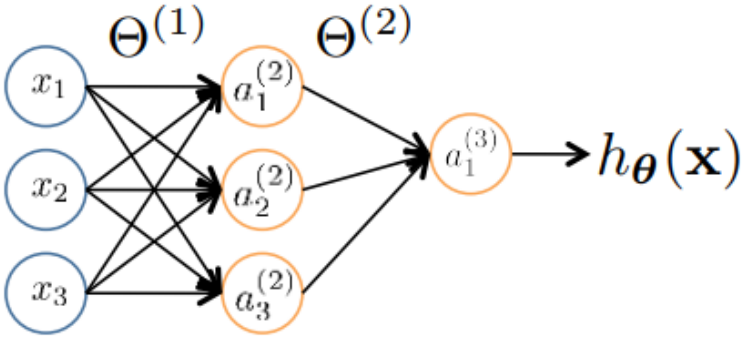
Advantages of Multi-Layer Perceptron

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.

Feed Forward Neural Networks



$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = weight matrix stores parameters from layer j to layer $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

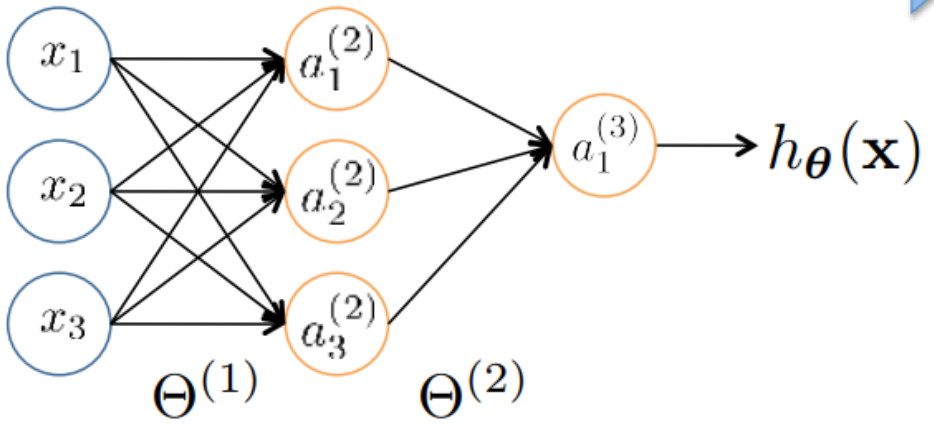
Vectorization

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

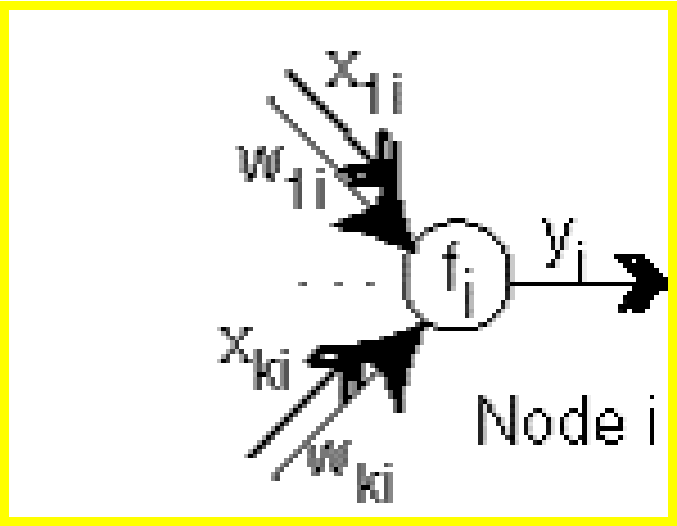
$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

How to determine the weight?

the weight in the ANN may be determined in two ways

- the more common approach is to have them determined via a learning process



$$y_i = f_i(\sum_{j=1}^{k} w_{ji}\ x_{ji}) = f_i([w_{1i} \ldots w_{ki}] \begin{bmatrix} x_{1i} \\ \ldots \\ x_{ki} \end{bmatrix})$$

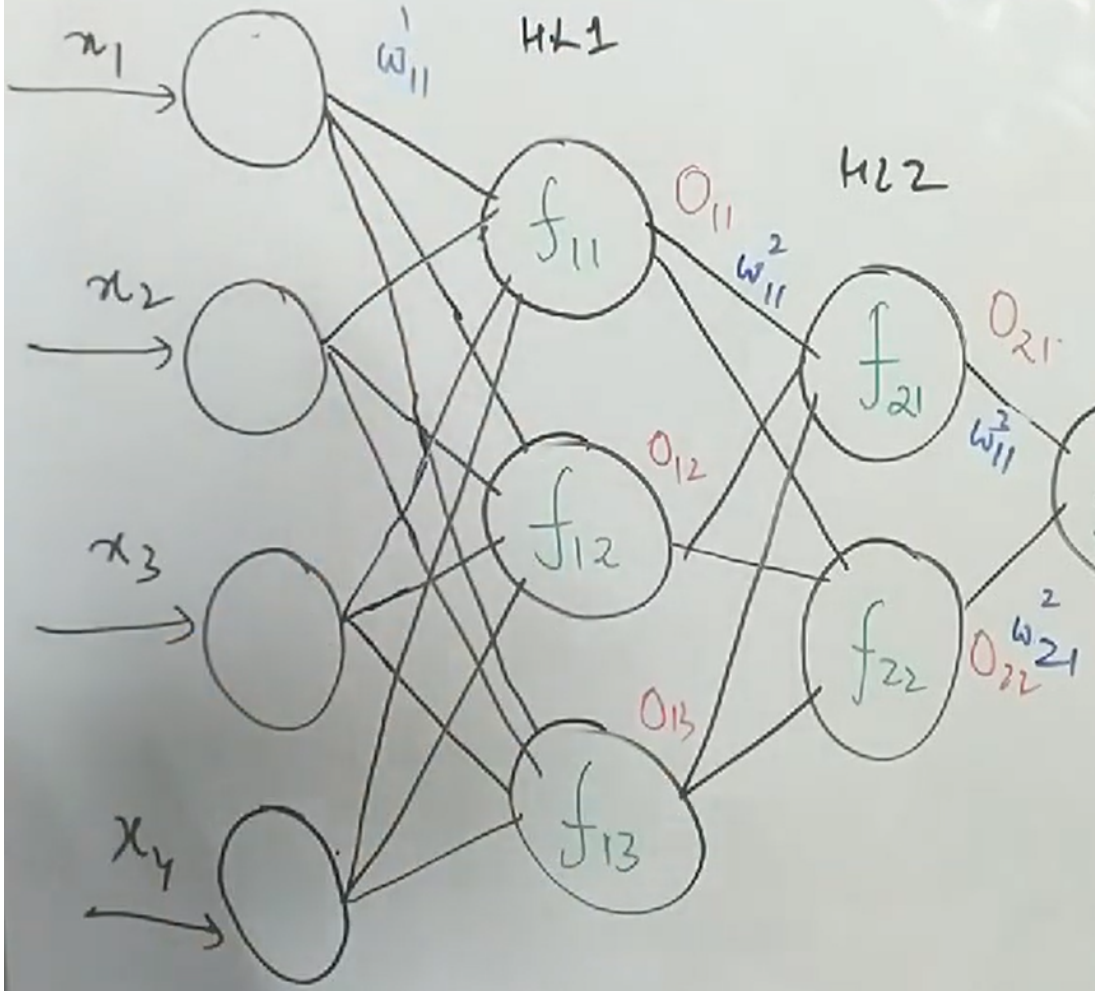
Backpropagation Algorithm

- Backpropagation is a learning technique that adjusts weights in the ANN by propagating weight changes backward from the sink to the source nodes
- weight is changing by using delta method

$$\Delta w_{ij} = \eta x_{ij}(\widehat{y}_j - y_j)$$

where value y_j is the output value and $\widehat{y}_j$ is the predicted value for output at node j and η is called the learning rate (i.e., $\eta = 1/|\#\text{of entries in training set}|$)

CHAIN RULE IN BACKPROPAGATION



$$w^3_{11\,new} = w^3_{11\,old} - \eta \frac{\partial L}{\partial w^3_{11}}$$

learning

$$\frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w^3_{11}}$$

$$Loss = \sum_{i=1}^{n} (y - \hat{y})^2$$

Applications of DNN

NNs have been used in

- pattern recognition
- speech recognition and synthesis
- medical applications (diagnosis, drug design)
- fault detection
- problem diagnosis
- robot control and computer vision
- business: DNNs have been used to “advise" booking of airline seats to increase profitability
- ...

Issues of ANN Algorithm

- the activation functions typically are sigmoidal for the output layer and relu for the hidden layer
- sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- the output value that is generated indicates the probability that the corresponding input tuple belongs to that class
- the tuple will then be assigned to the class with the highest probability of membership
- for each tuple in the training set, propagate it through the network and evaluate the output prediction to the actual result
  — if the prediction is accurate, adjust labels to ensure that this prediction has a higher output weight the next time
  — if the prediction is not correct, adjust the weights to provide a lower output value for this class

Issues of ANN Algorithm

there are many issues to be examined:

1. number of hidden layers: in the simplest case, there is only one hidden layer
2. number of hidden nodes: choosing the best number of hidden nodes per hidden layer is one of the most difficult problems when using ANN
   — rules of thumb are often given that are based on the size of the training set
   — the number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer
3. training data: with too much training data the ANN may suffer from overfitting, while too little and it may not be able to classify accurately enough
4. number of sinks: usually assumed that the number of output nodes is the same as the number of classes, this is not always the case

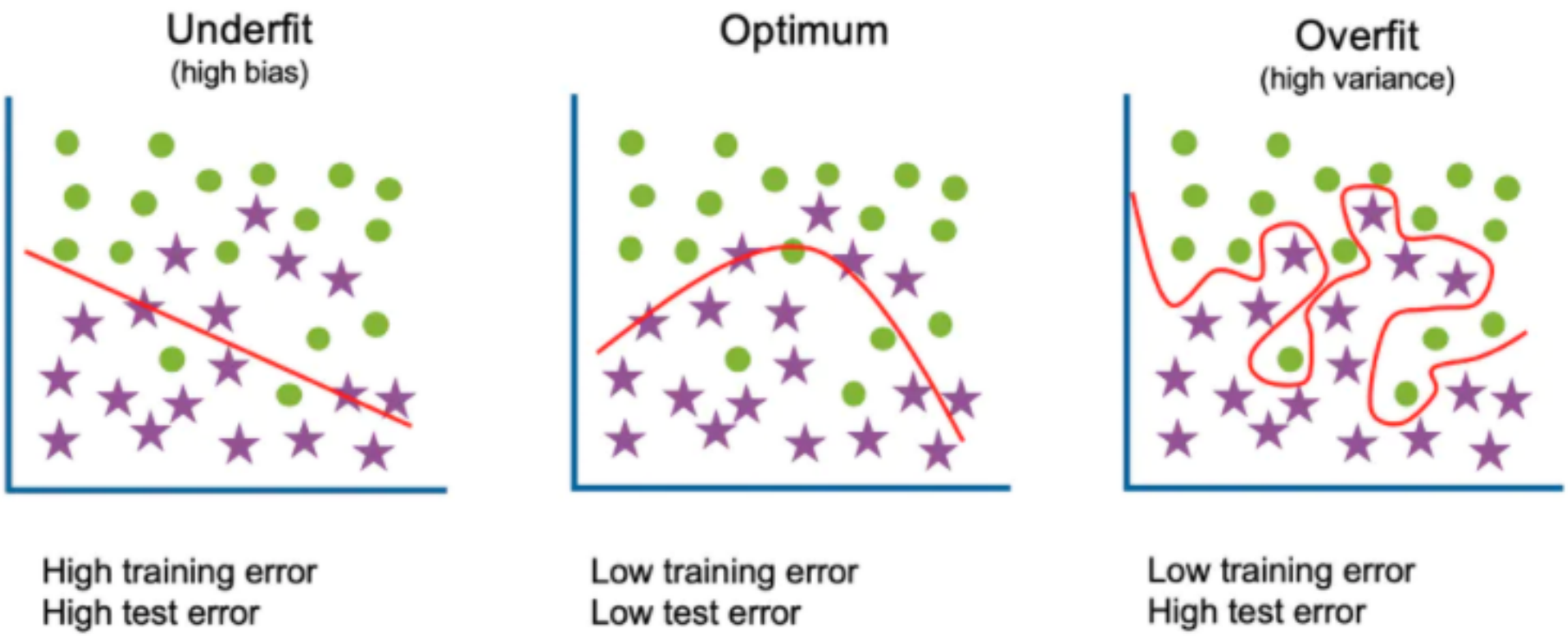
Issues of ANN Algorithm

1. weights: initial weights are usually assumed to be small positive numbers and are assigned randomly
2. learning technique: the technique for adjusting the weights is called the learning technique
   — algorithms: Propagation, Backpropagation, Gradient Descent are used for weight estimation
   — the most common approach is some form of backpropagation

Advantages to the use of ANNs for classification

- ANNs are more robust than DTs because of the weights
- the ANN improves its performance by learning
  — this may continue even after the training set has been applied
- ANNs may contain many processing elements and thus can be used in massively parallel systems
- the use of ANNs can be parallelized for better performance
- there is a low error rate and thus a high degree of accuracy once the appropriate training has been performed
- ANNs are more robust than DTs in noisy environments

Overfitting



Overfitting

overfitting occurs when the ANN is trained to fit one set of data almost exactly

- the error that occurs with the given training data is quite small; however, when new data are examined, the error is very large
- to avoid overfitting, smaller ANNs are advisable
- another approach that can be used to avoid overfitting is to stop the learning process early
- using a larger training set also helps

Disadvantages of ANN

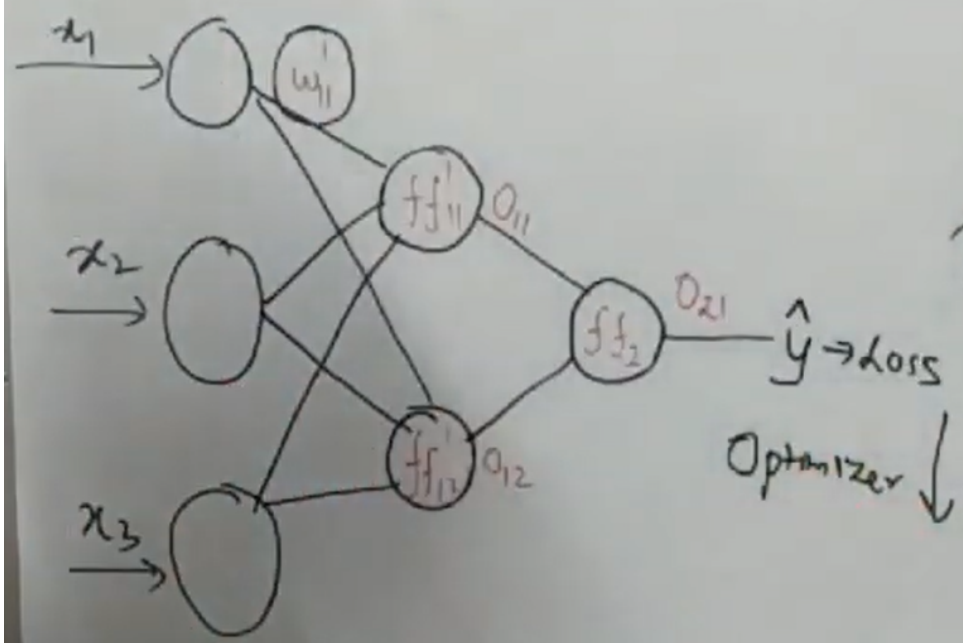
- a major drawback to the use of ANNs is the fact that they are difficult to explain to end users (unlike decision trees, which are easy to understand)
- unlike decision trees, ANNs usually work only with numeric data
- they have a long training time (time during which the learning takes place) and thus are not appropriate for real-time applications
- may suffer from overfitting

Note: If anyone is interested in learning about the neural network modeling in-depth, you can read this book. https://livebook.manning.com/book/grokking-deep-learning/table-of-contents/

As number of hidden layers grow, gradient becomes very small and weights will hardly change . This will hamper the learning process.

Vanishing Gradients

VANISHING GRADIENT PROBLEM



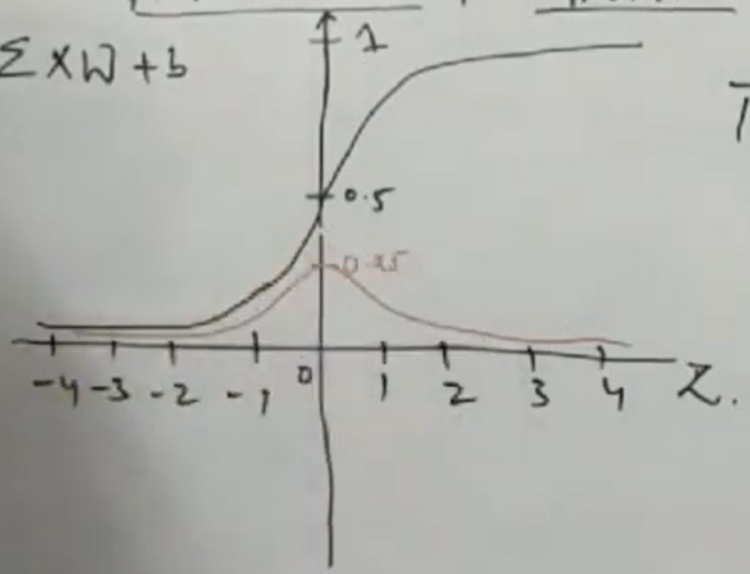
Weight updation

$$W'_{11\,new} = W'_{11\,old} - \eta \boxed{\frac{\partial L}{\partial W'_{11\,old}}}$$

$$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}} \rightarrow \text{Chain Rule}$$

VANISHING GRADIENT PROBLEM

$z = \Sigma X W + b$



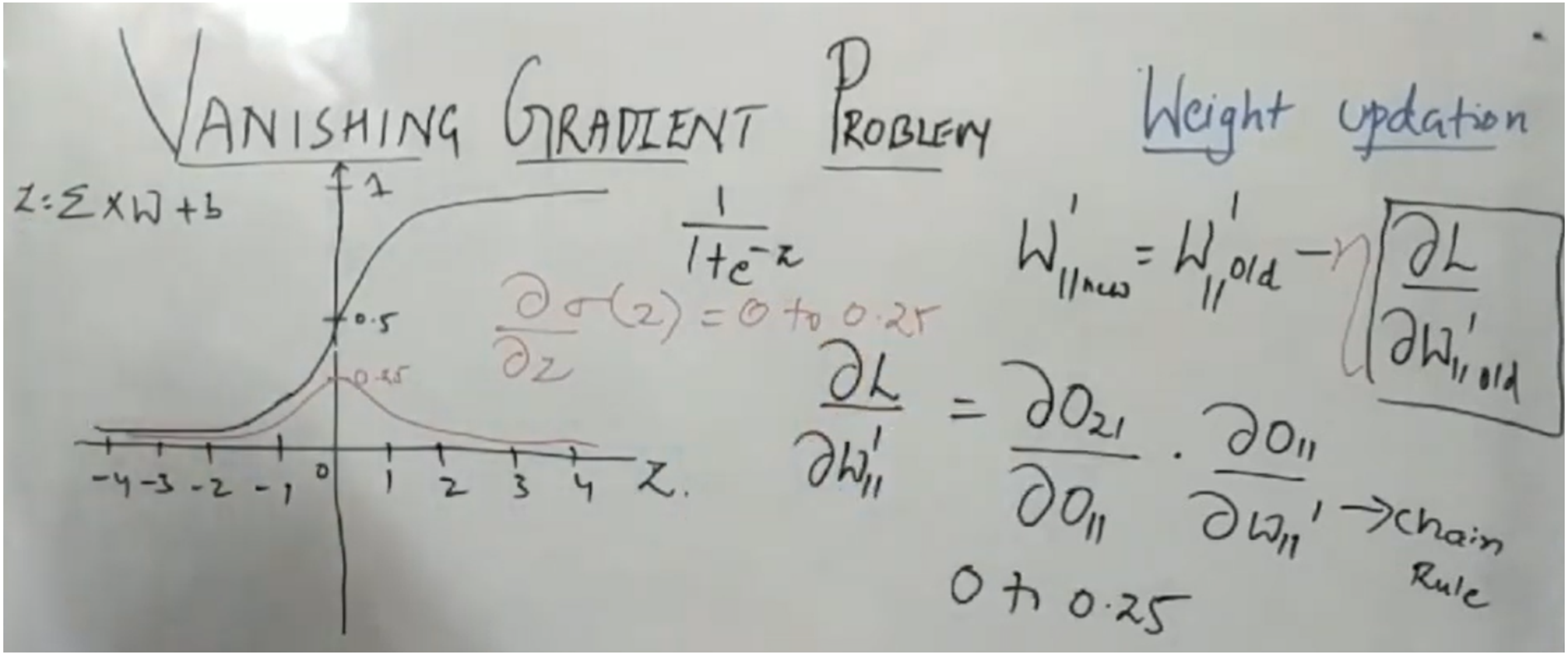
$$\frac{1}{1+e^{-z}}$$

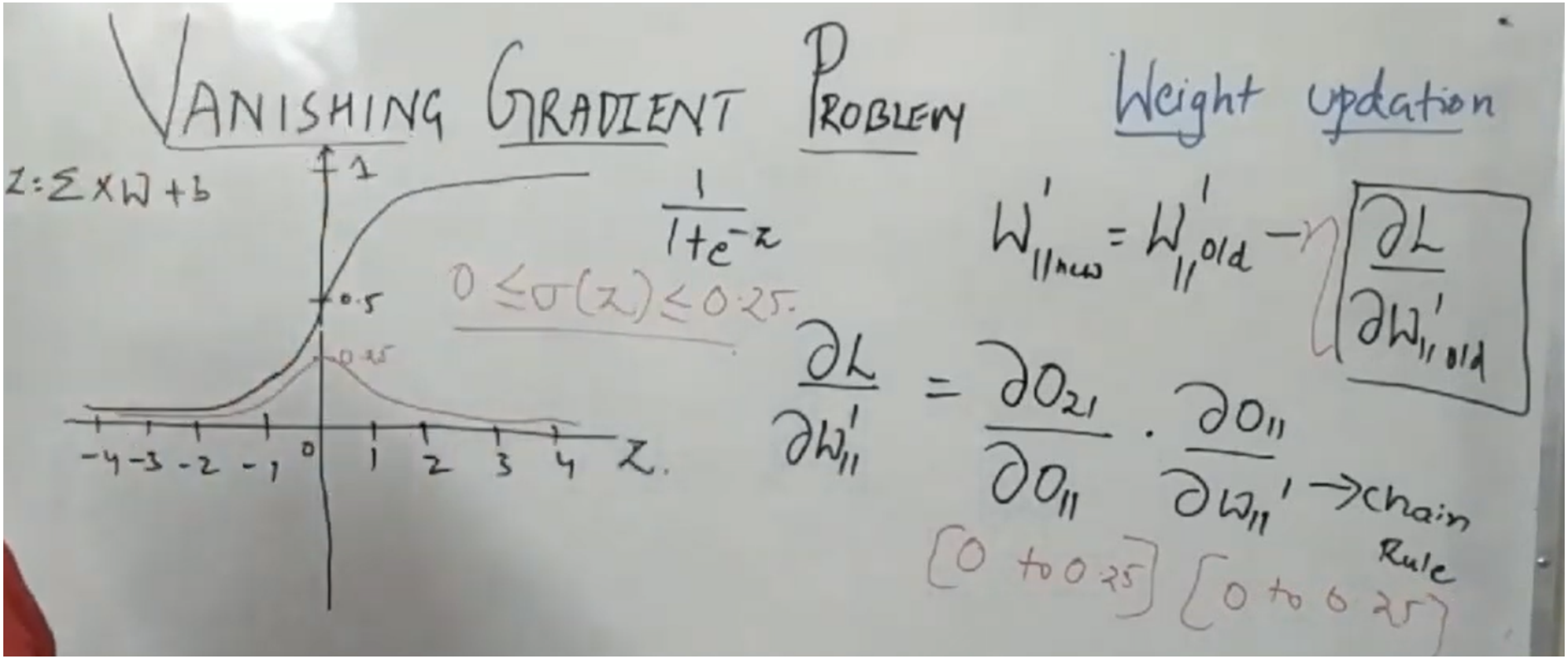
Weight updation

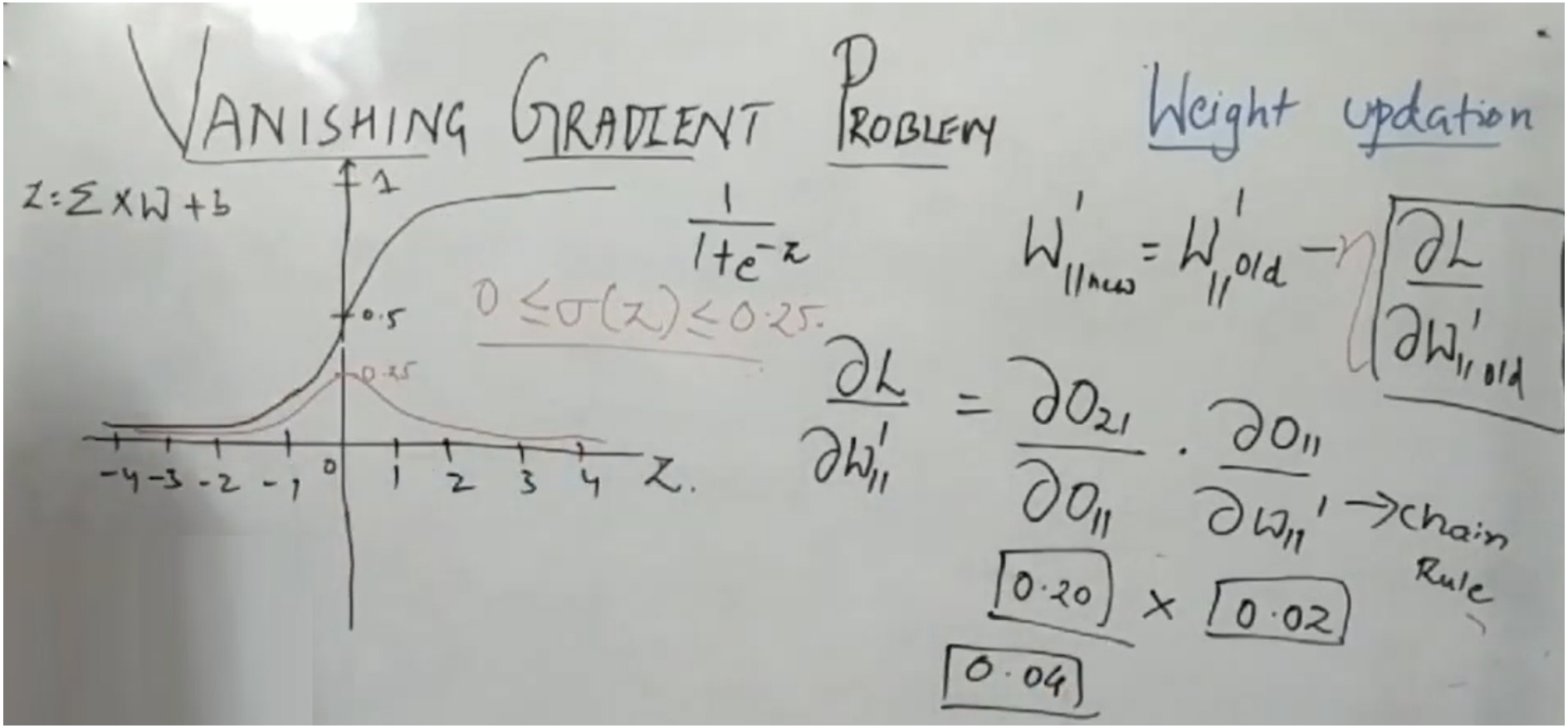
$$W'_{11\,new} = W'_{11\,old} - \eta \boxed{\frac{\partial L}{\partial W'_{11\,old}}}$$

$$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W_{11}'} \rightarrow \text{chain Rule}$$

0 to 0.25

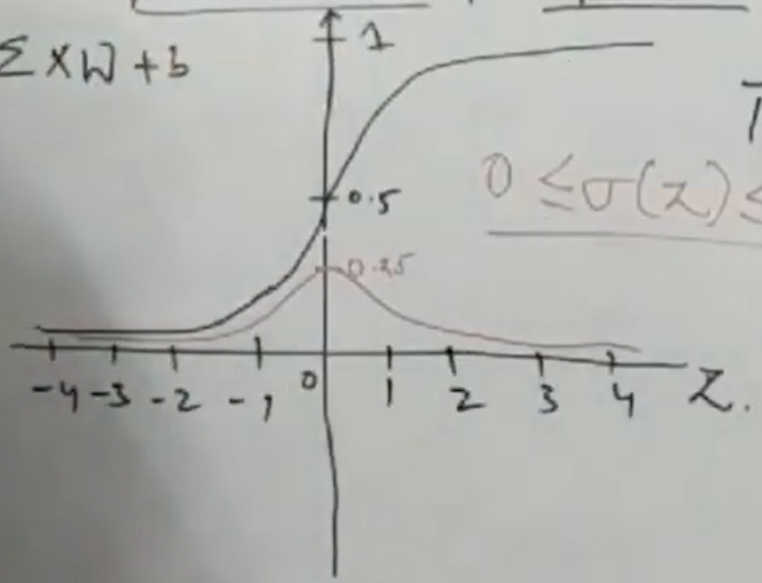

VANISHING GRADIENT PROBLEM
Weight updation
Z = Σ X W + b
1
0.5
0.25
-4 -3 -2 -1 0 1 2 3 4 Z.
1/(1+e^-z)
∂σ(z)/∂z = 0 to 0.25
W'11new = W'11old − η ∂L/∂W'11old
∂L/∂W'11 = ∂O21/∂O11 . ∂O11/∂W'11 → Chain Rule
0 to 0.25


VANISHING GRADIENT PROBLEM
Weight updation
$z = \Sigma x W + b$
$\frac{1}{1+e^{-z}}$
$0 \le \sigma(z) \le 0.25$
$W'_{11\,new} = W'_{11\,old} - \eta \frac{\partial L}{\partial W'_{11\,old}}$
$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}}$
→ chain Rule
[0 to 0.25]
[0 to 0.25]
1
0.5
0.25
-4 -3 -2 -1 0 1 2 3 4 z.


VANISHING GRADIENT PROBLEM
Weight updation
$z = \Sigma X W + b$
$\frac{1}{1+e^{-z}}$
$0 \le \sigma(z) \le 0.25$
$W'_{11new} = W'_{11old} - \eta \frac{\partial L}{\partial W'_{11old}}$
$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}}$
chain Rule
0.20 × 0.02
0.04
1
0.5
0.25
-4 -3 -2 -1 0 1 2 3 4
z.

Vanishing Gradient Problem

$z = \Sigma x W + b$



$$\frac{1}{1+e^{-z}}$$

$$0 \le \sigma(z) \le 0.25.$$

Weight updation

$$W'_{11\,new} = W'_{11\,old} - \eta \boxed{\frac{\partial L}{\partial W'_{11\,old}}}$$

$W'_{11\,old} = 2.5$, $\eta = 1$

$$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}} \rightarrow \text{chain Rule}$$

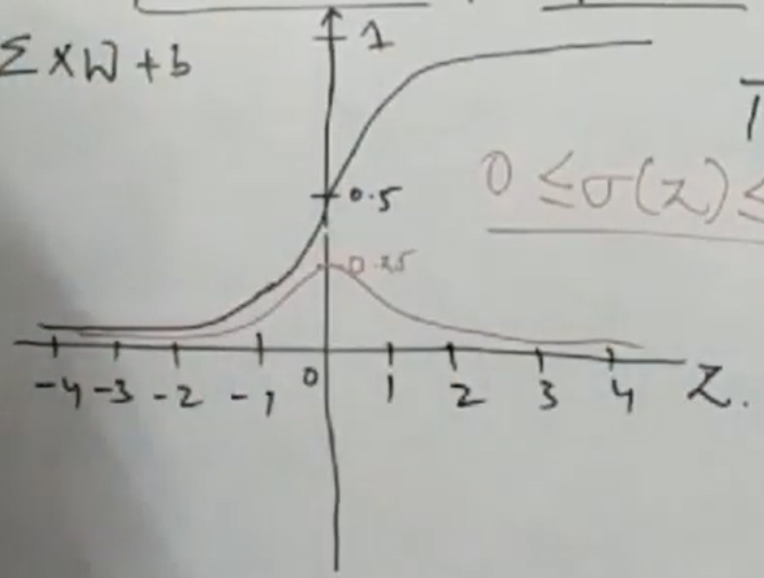
$\frac{\partial O_{21}}{\partial O_{11}} = 0.04$

$$= 2.5 - 1 \times 0.04$$

$$= 2.46$$

Vanishing Gradient Problem

$z = \sum x W + b$



$$\frac{1}{1+e^{-z}}$$

$0 \le \sigma(z) \le 0.25$

$$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}} \rightarrow \text{chain Rule}$$

$0.20 \times 0.1 \times 0.05 \times 0.04$

10^{-4}

Weight updation

$$W'_{11new} = W'_{11old} - \eta \boxed{\frac{\partial L}{\partial W'_{11old}}}$$

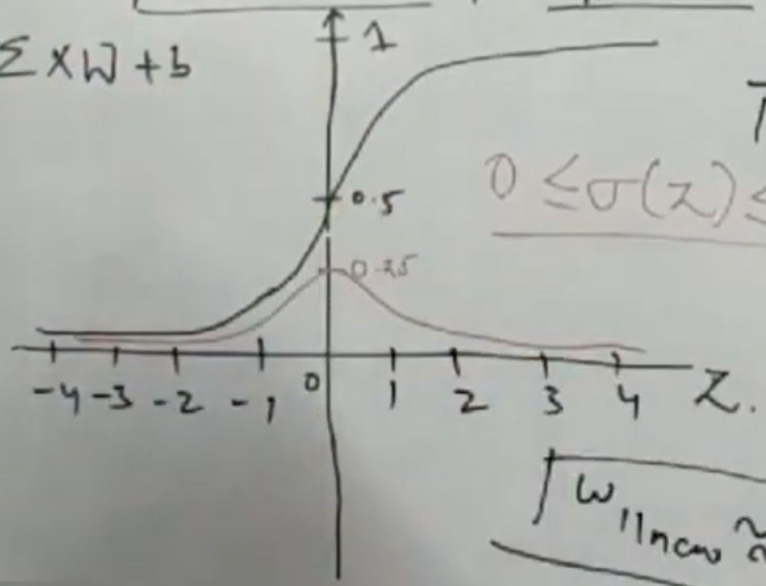
$2.5 \qquad 1$

$= 2.5 - 1 \times 0.04$

$= 2.46 \approx 2.499999$

VANISHING GRADIENT PROBLEM

$z = \Sigma x W + b$



$$\frac{1}{1+e^{-z}}$$

$$0 \le \sigma(z) \le 0.25.$$

$$\boxed{w_{11new} \approx w_{11old}}$$

Weight updation

$$W'_{11new} = W'_{11old} - \eta \boxed{\frac{\partial L}{\partial W'_{11old}}}$$

2.5 $\quad 1$

$$\frac{\partial L}{\partial W'_{11}} = \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial W'_{11}} \rightarrow \text{Chain Rule}$$

4 layer

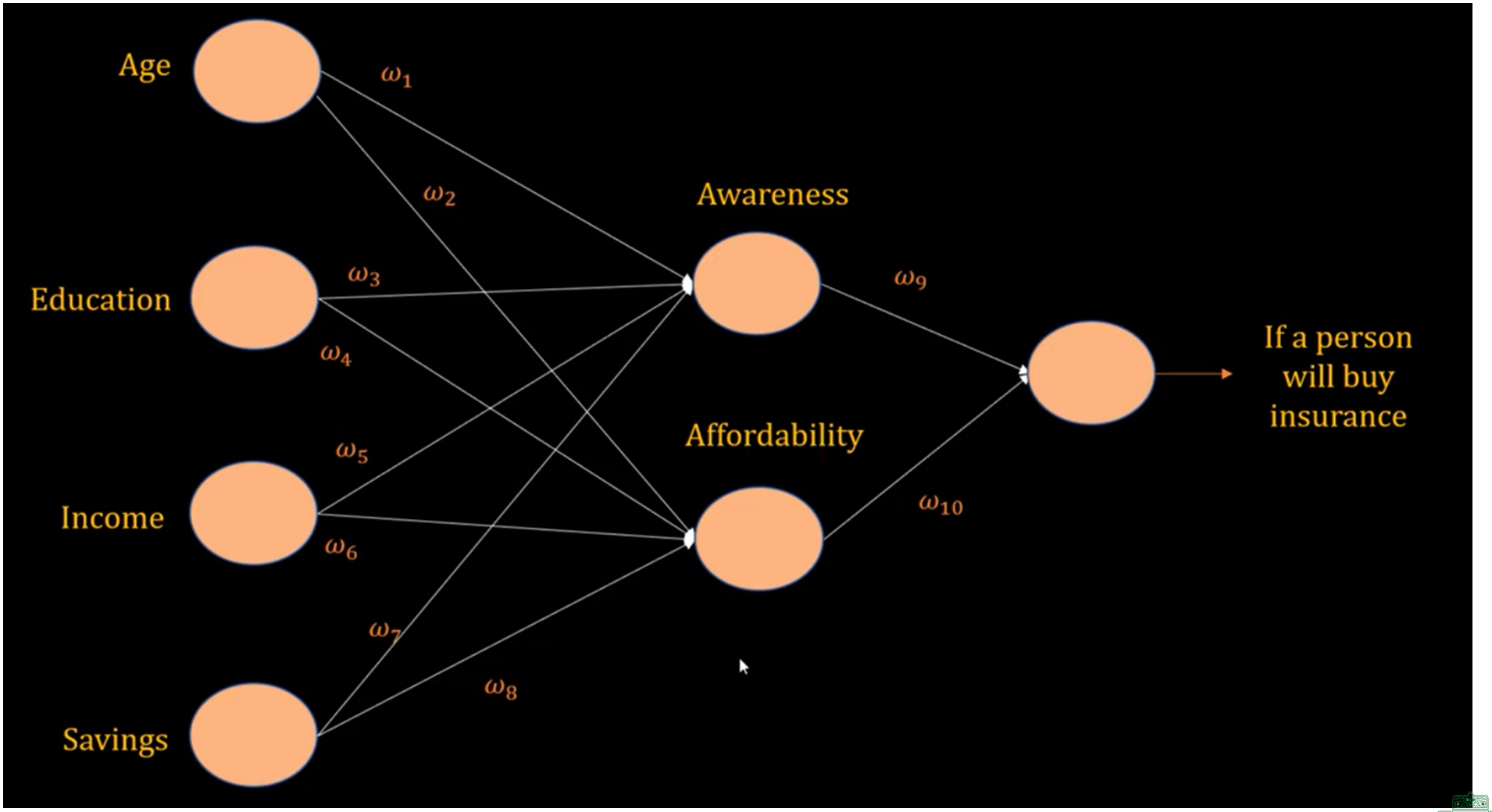
$$= 0.20 \times 0.1 \times 0.05 \times 0.0$$

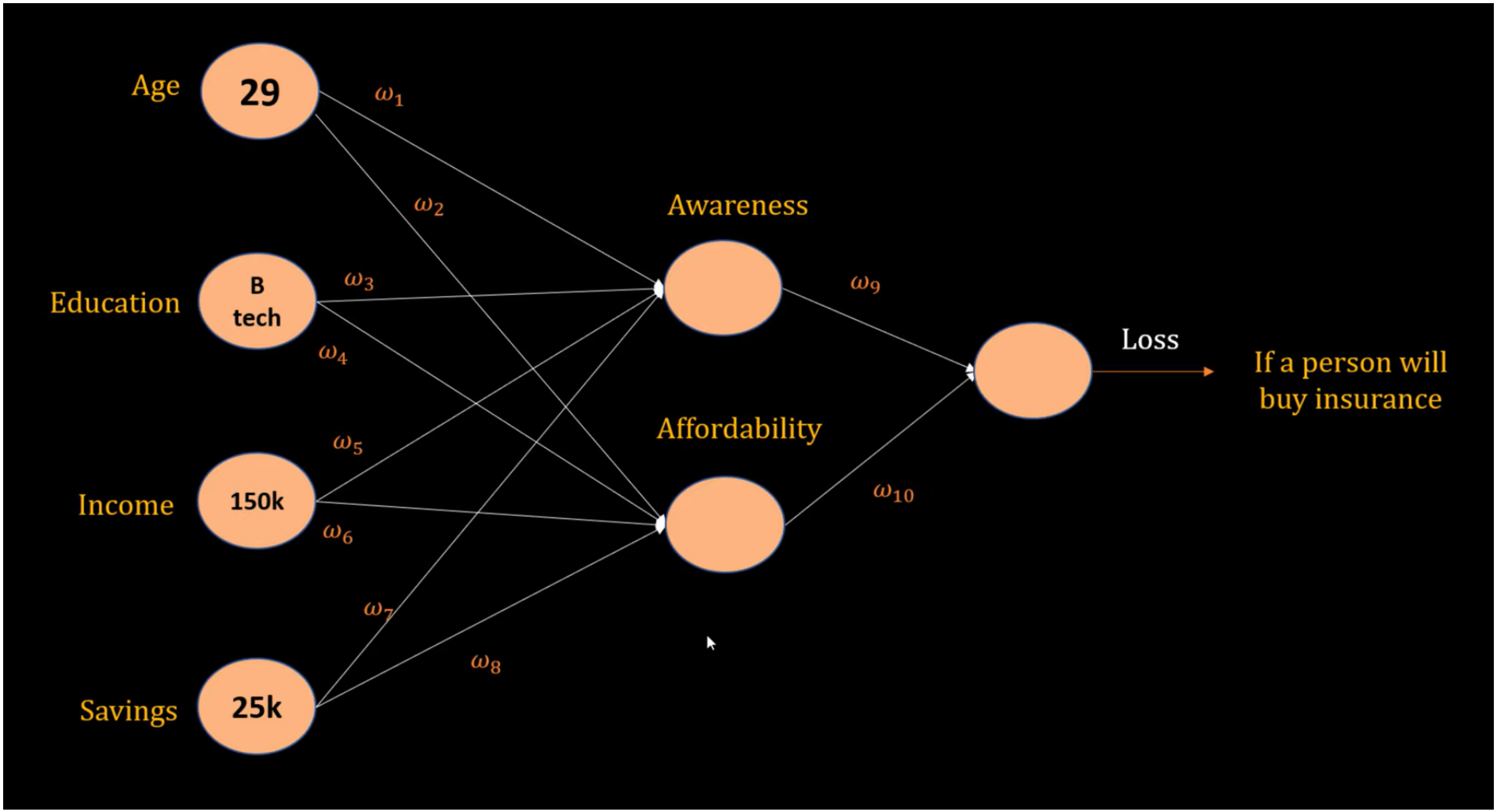
$$= 2.5 - 1 \times 0.04$$

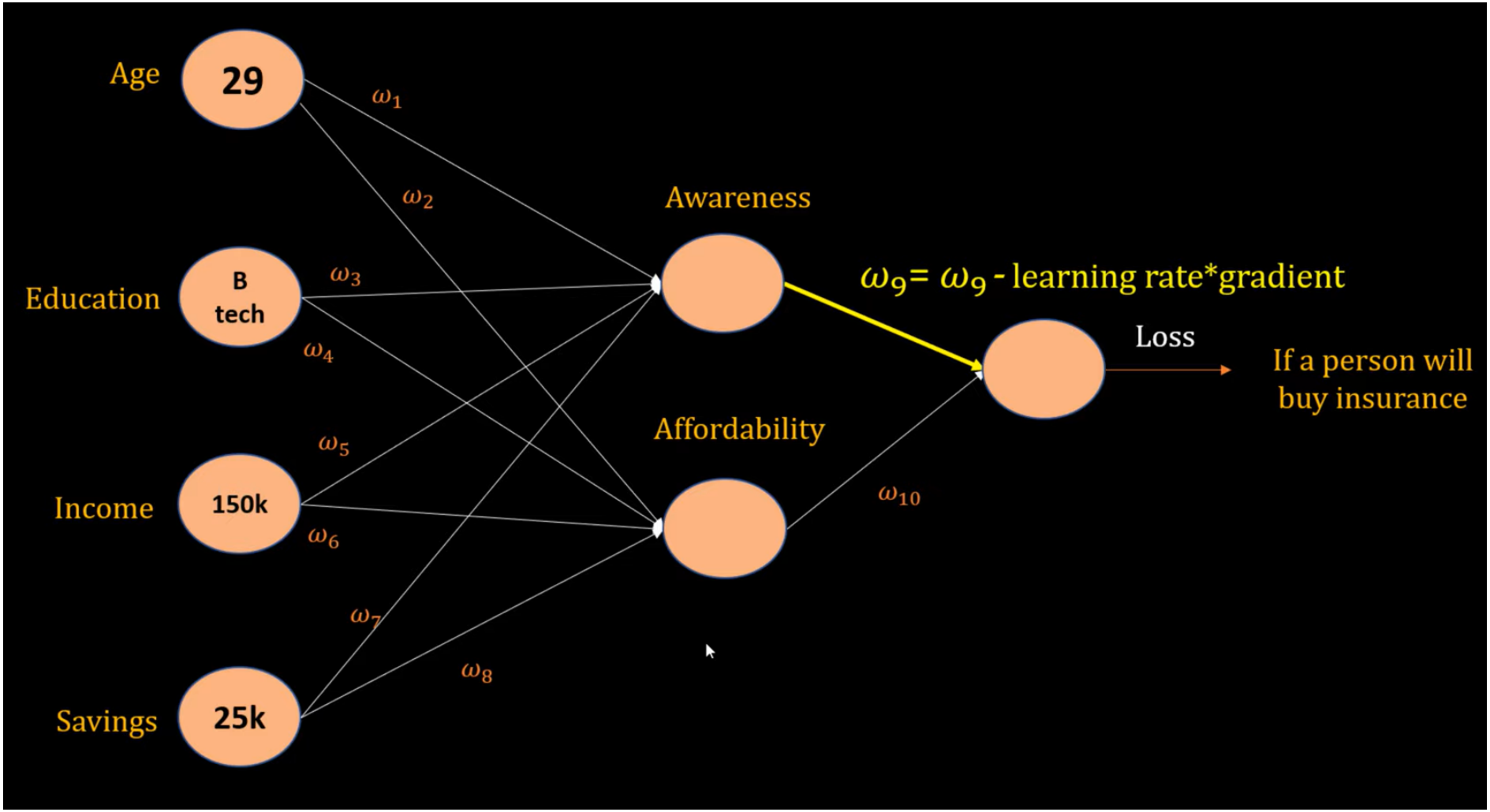
$$= 2.46 \approx 2.499999$$

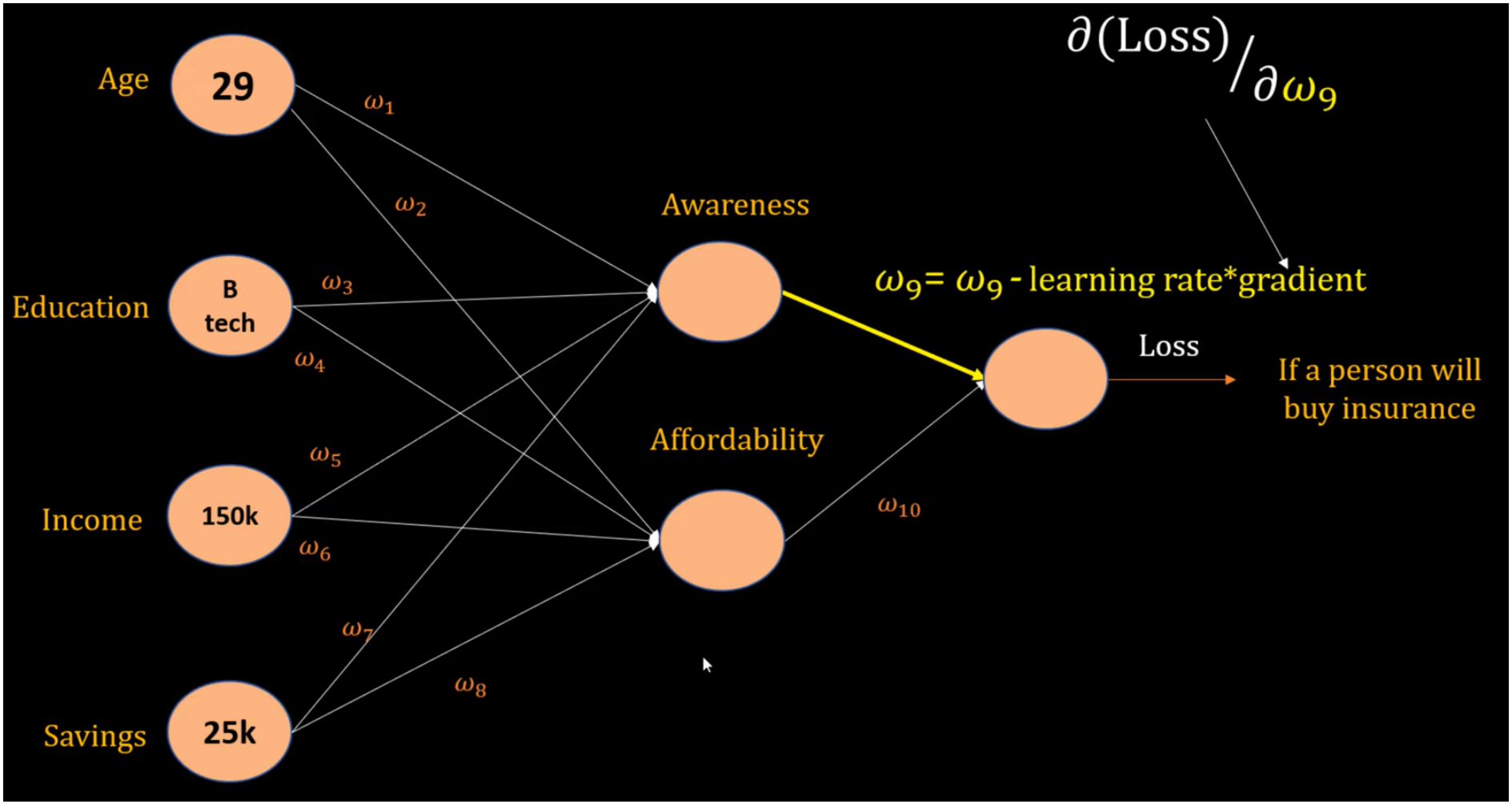
10^{-4}

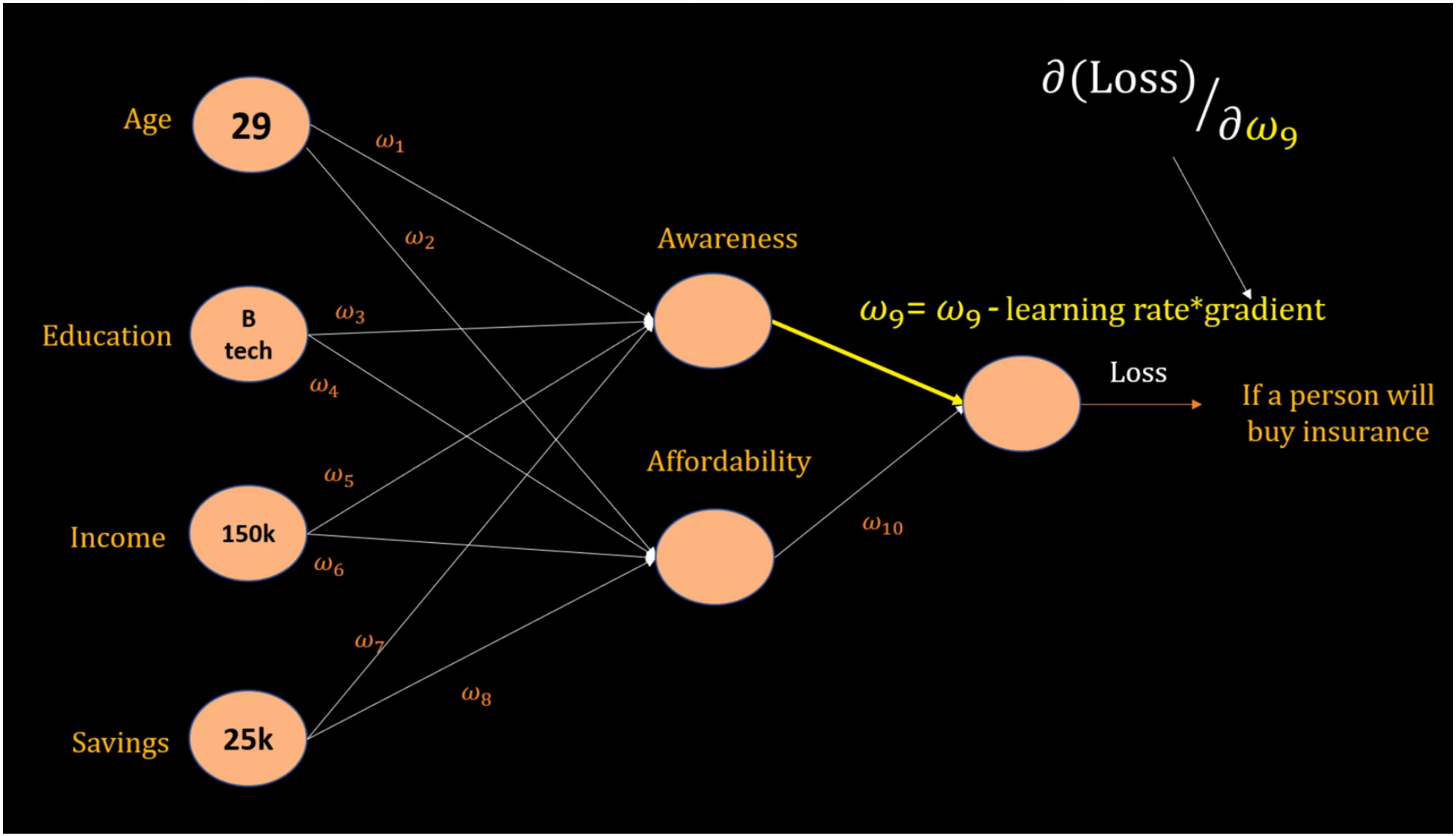
Example: Insurance Data

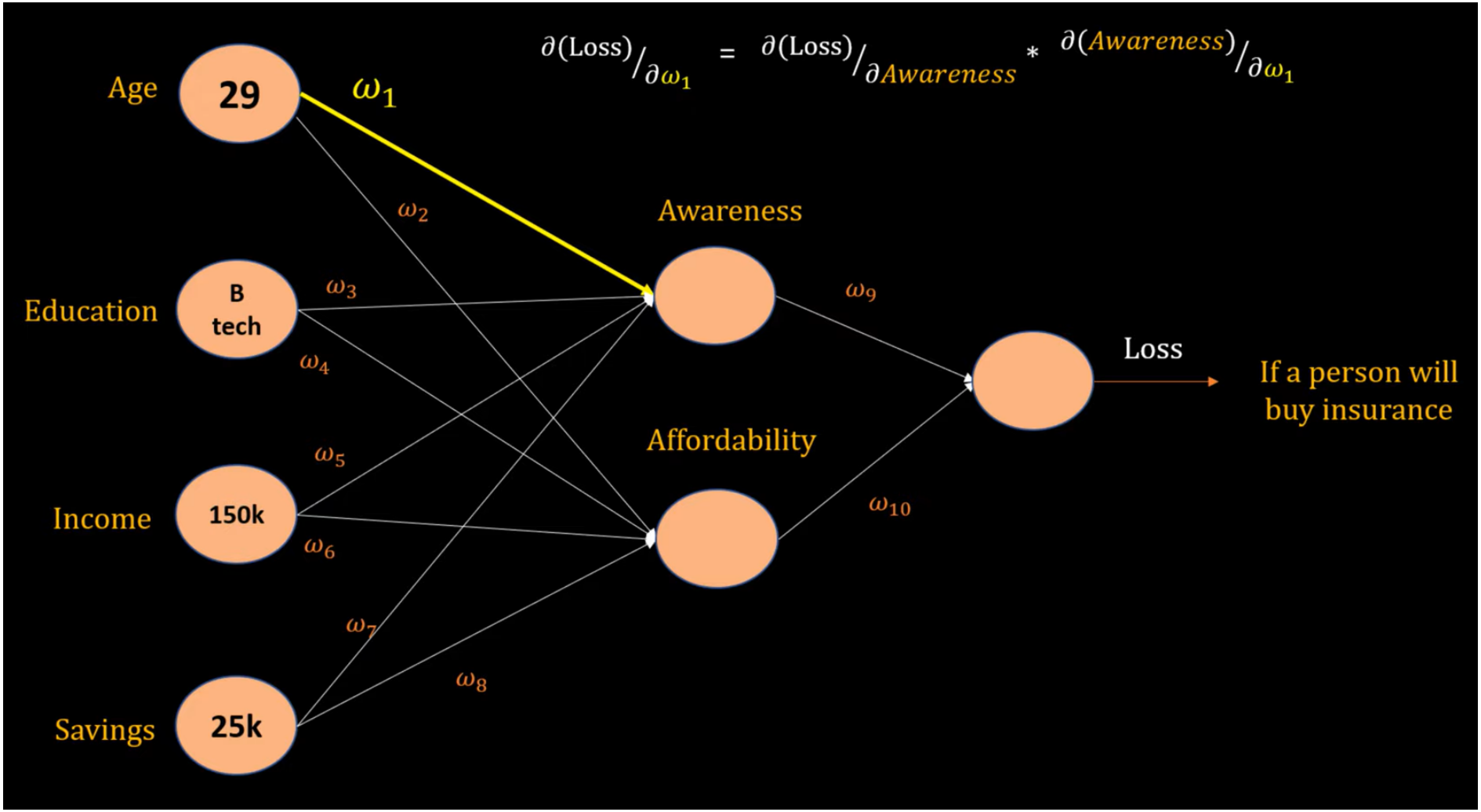



Age
29
Education
B tech
Income
150k
Savings
25k
ω_1
ω_2
ω_3
ω_4
ω_5
ω_6
ω_7
ω_8
ω_9
ω_{10}
Awareness
Affordability
Loss
If a person will buy insurance


Age
29
Education
B tech
Income
150k
Savings
25k
ω_1
ω_2
ω_3
ω_4
ω_5
ω_6
ω_7
ω_8
Awareness
Affordability
ω_9= ω_9 - learning rate*gradient
ω_{10}
Loss
If a person will buy insurance


$\partial(\text{Loss})/\partial\omega_9$
Age
29
Education
B tech
Income
150k
Savings
25k
ω_1
ω_2
ω_3
ω_4
ω_5
ω_6
ω_7
ω_8
ω_{10}
Awareness
Affordability
$\omega_9 = \omega_9$ - learning rate*gradient
Loss
If a person will buy insurance


Age
29
Education
B tech
Income
150k
Savings
25k
ω_1
ω_2
ω_3
ω_4
ω_5
ω_6
ω_7
ω_8
ω_{10}
Awareness
Affordability
$\partial(\text{Loss})/\partial\omega_9$
$\omega_9 = \omega_9$ - learning rate*gradient
Loss
If a person will buy insurance

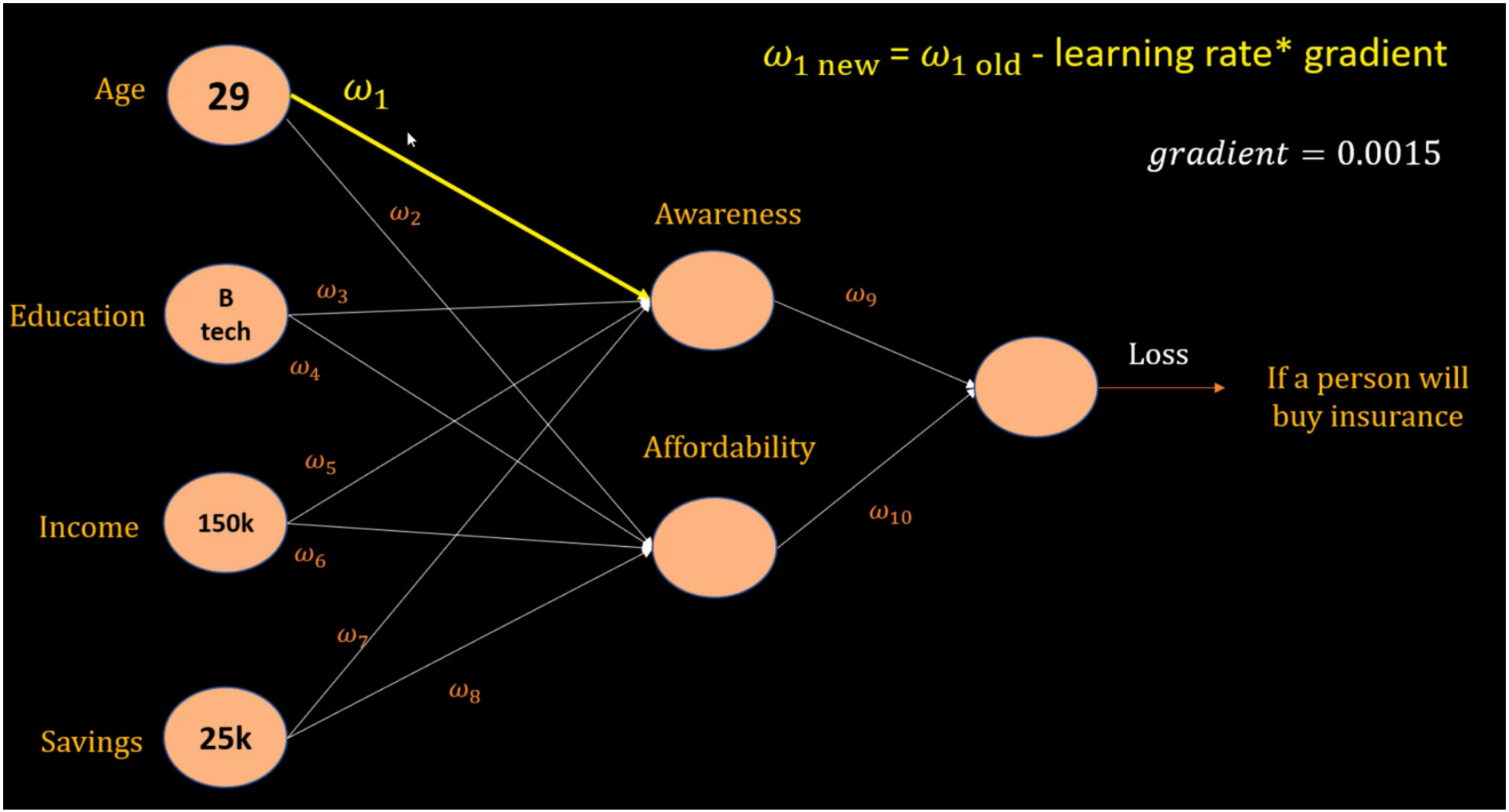

$\partial(\text{Loss})/\partial\omega_1 = \partial(\text{Loss})/\partial Awareness * \partial(Awareness)/\partial\omega_1$
Age
29
ω_1
ω_2
Awareness
Education
B tech
ω_3
ω_4
ω_9
Loss
If a person will buy insurance
Affordability
ω_5
Income
150k
ω_6
ω_{10}
ω_7
ω_8
Savings
25k

$$\partial(\text{Loss})/\partial\omega_1 = \partial(\text{Loss})/\partial Awareness * \partial(Awareness)/\partial\omega_1$$

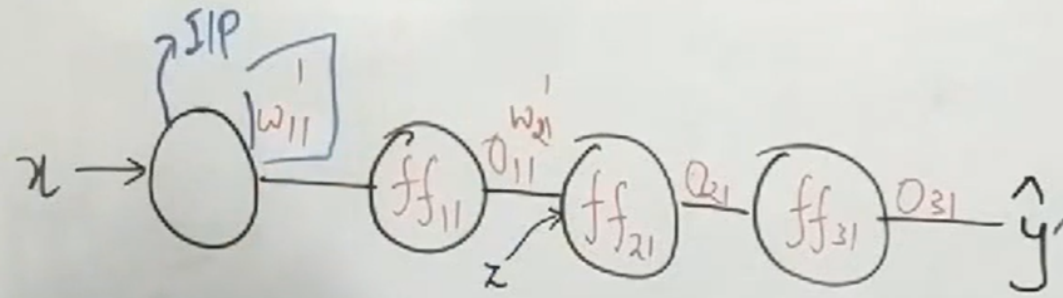
$$gradient = d1 * d2$$

$$gradient = 0.03 * 0.05$$

$$gradient = 0.0015$$


$\omega_{1\ new} = \omega_{1\ old}$ - learning rate* gradient
$gradient = 0.0015$
Age
29
ω_1
ω_2
Education
B tech
ω_3
ω_4
Income
150k
ω_5
ω_6
Savings
25k
ω_7
ω_8
Awareness
Affordability
ω_9
ω_{10}
Loss
If a person will buy insurance

EXPLODING GRADIENT PROBLEM



$$w'_{11_{new}} = w'_{11_{old}} - \eta \frac{\partial L}{\partial w'_{11}}$$

Learning Rate

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial O_{31}}{\partial O_{21}} \cdot \boxed{\frac{\partial O_{21}}{\partial O_{11}}} \cdot \frac{\partial O_{11}}{\partial w'_{11}}$$

$$O_{21} = \phi(z)$$

$$z = w_{21} \cdot O_{11} + b_2.$$

$$\partial(\text{Loss})/\partial\omega_1 = \partial(\text{Loss})/\partial Awareness * \partial(Awareness)/\partial\omega_1$$

$$gradient = d1 * d2$$

$$gradient = 100 * 500$$

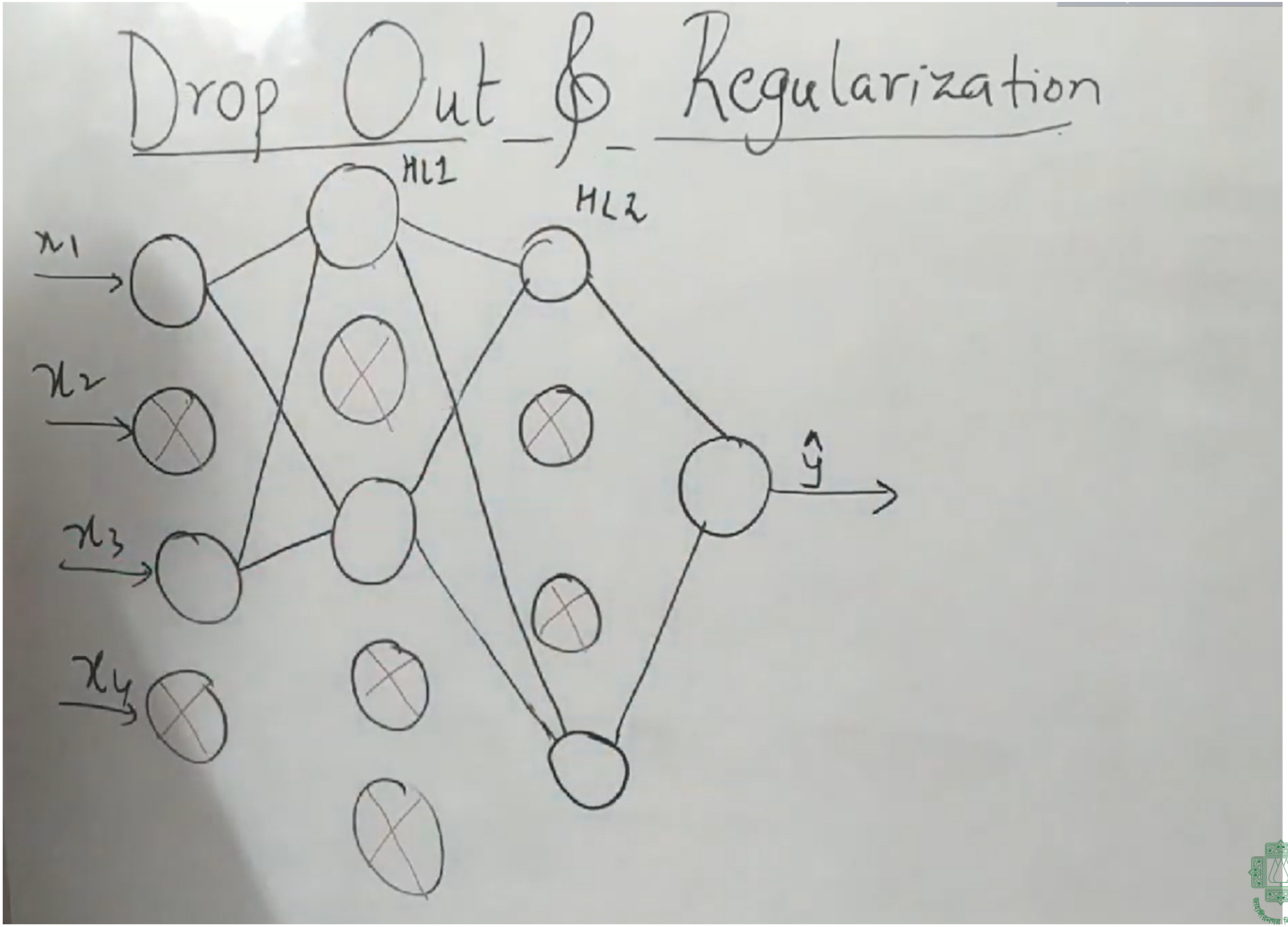
$$gradient = 50000$$

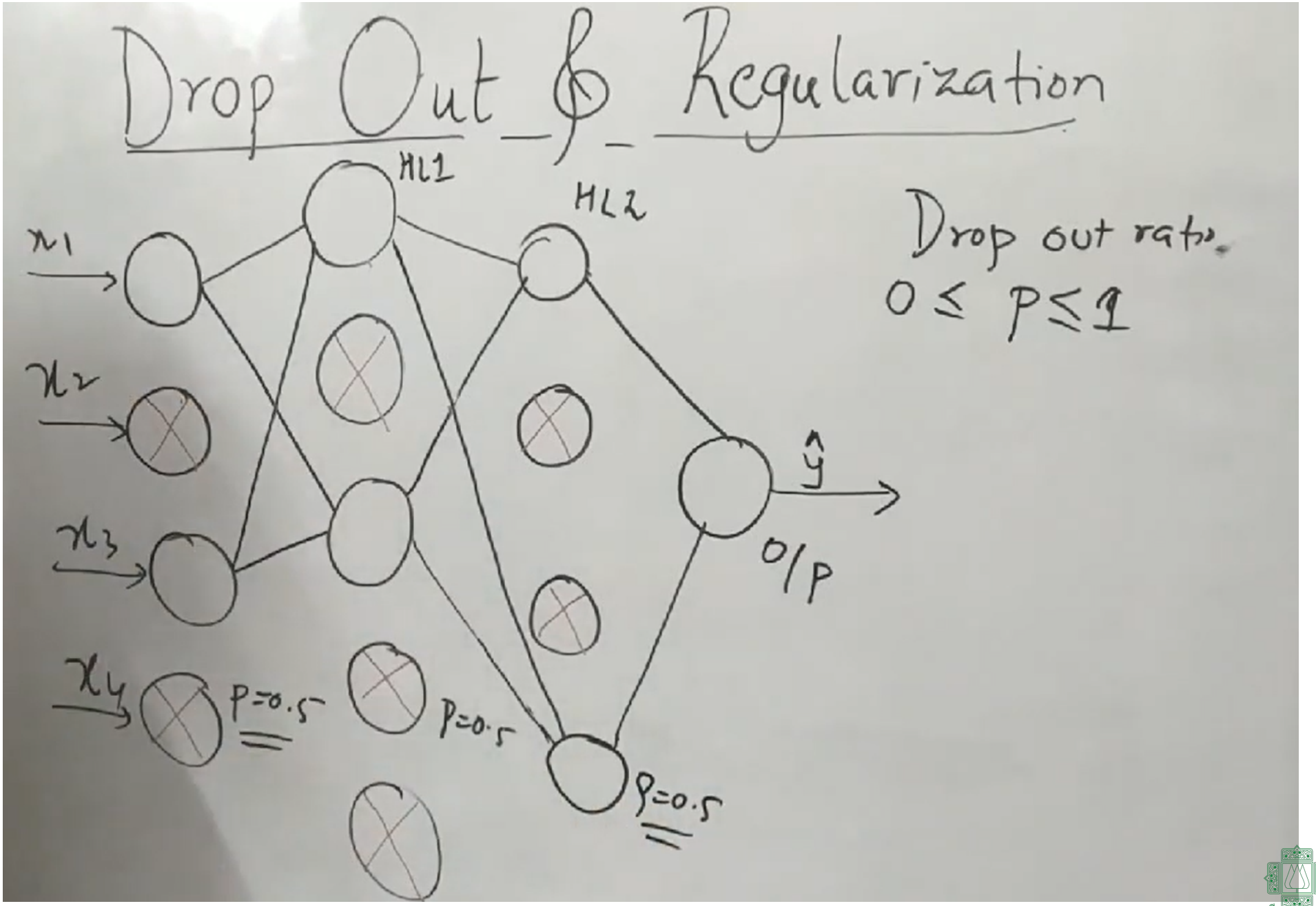
When individual derivatives are large, the final derivate will also become huge and weights would change drastically.

Exploding Gradients

$$gradient = d1 * d2 * d3 * d4 * \ldots * dn$$

Vanishing gradient problem is more prominent in very deep neural networks.


Drop Out of Regularization
HL1
HL2
x_1
x_2
x_3
x_4
$\hat{y}$


Drop Out as Regularization
HL1
HL2
x_1
x_2
x_3
x_4
$\hat{y}$
O/P
p=0.5
p=0.5
p=0.5
Drop out rate
$0 \leq p \leq 1$

Python code: for iris dataset

```
# See the categories of target variable
iris.target_names
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=0.3, random_state=1234)
mlp = MLPClassifier(hidden_layer_sizes=100,
    activation='relu', solver='adam', max_iter=5000)
mlp.fit(X_train,Y_train)
predictionsLDA= mlp.predict(X_test)
#print('Classification
    Matrix:\n',metrics.confusion_matrix(Y_test,
    predictionsLDA,labels=["setosa", "versicolor",
    "virginica"]))
print('Classification
    Report:\n',classification_report(Y_test, predictionsLDA))
print('Accuracy Score:',accuracy_score(Y_test,
    predictionsLDA))
accuNN=accuracy_score(Y_test, predictionsLDA)
```

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

Please refrain from accessing these files from automated scripts with high frequency. Make copies!`

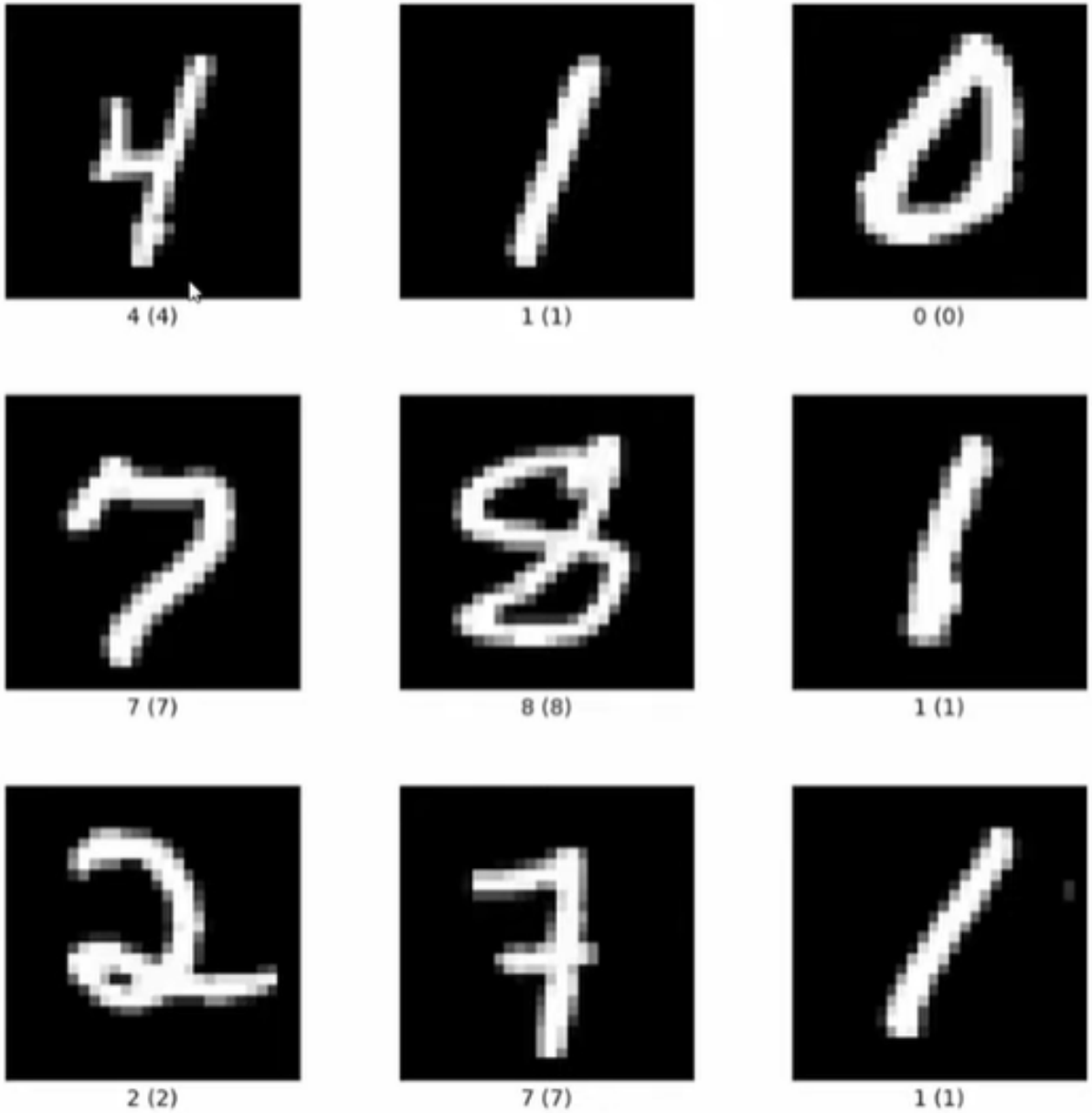
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

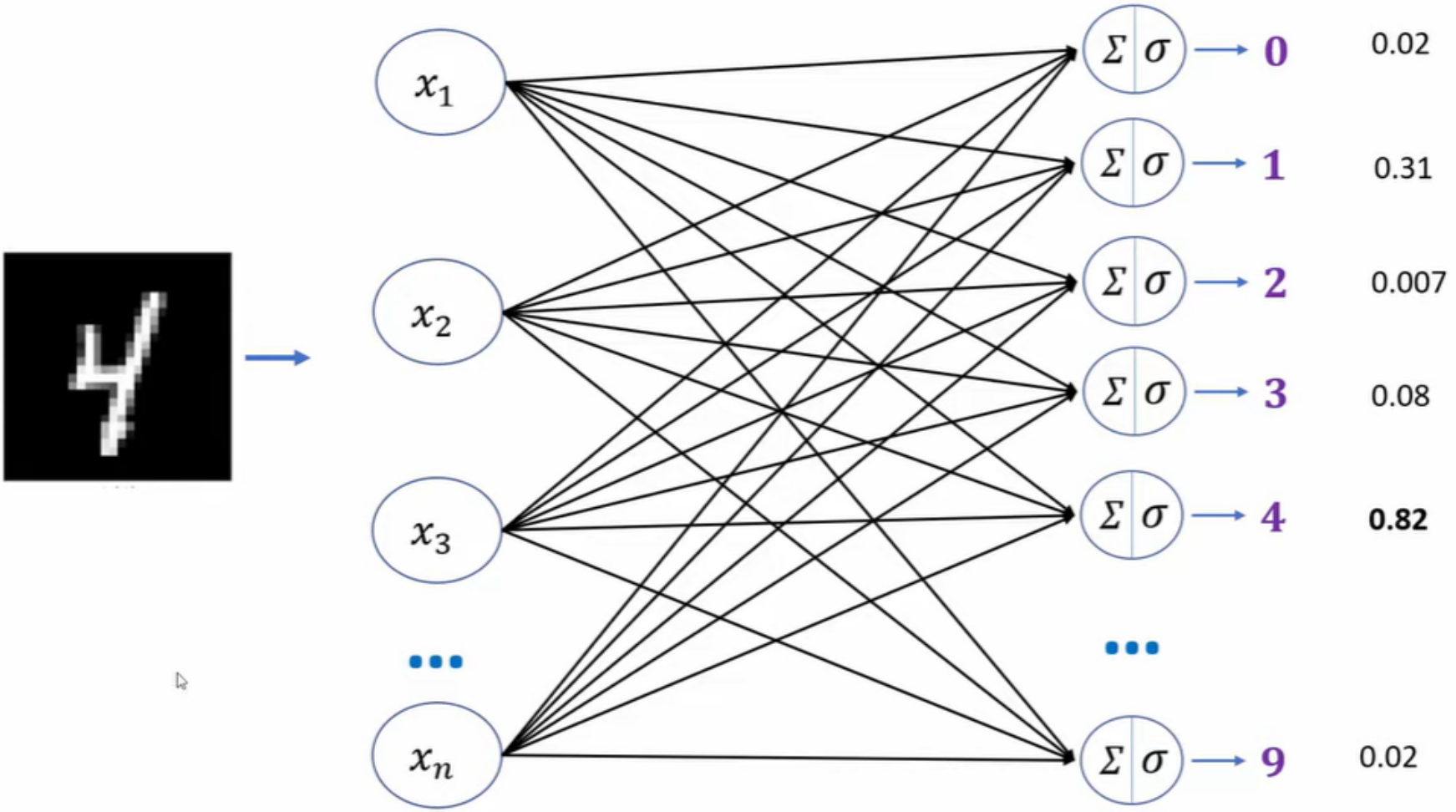
It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

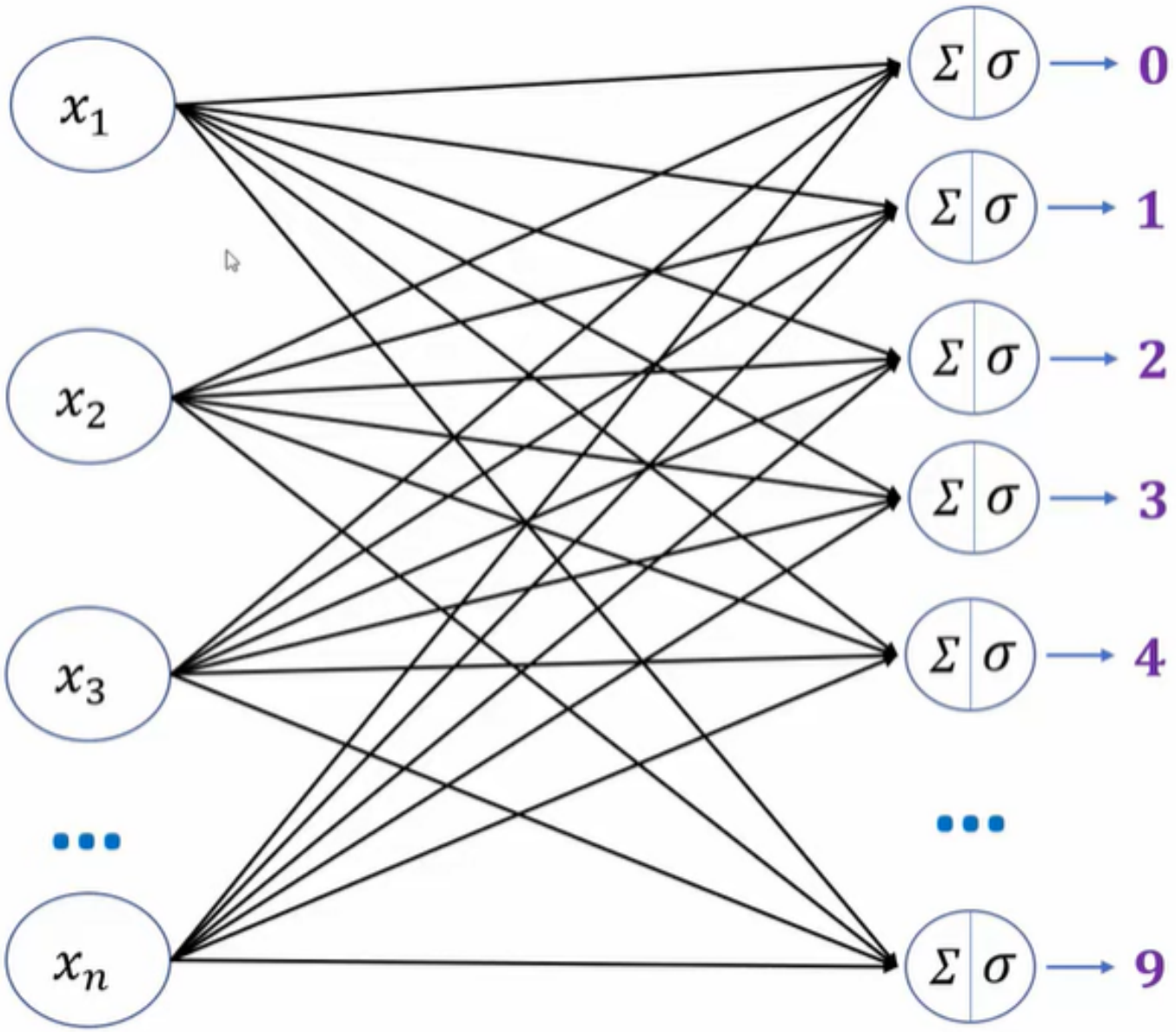
Four files are available on this site:

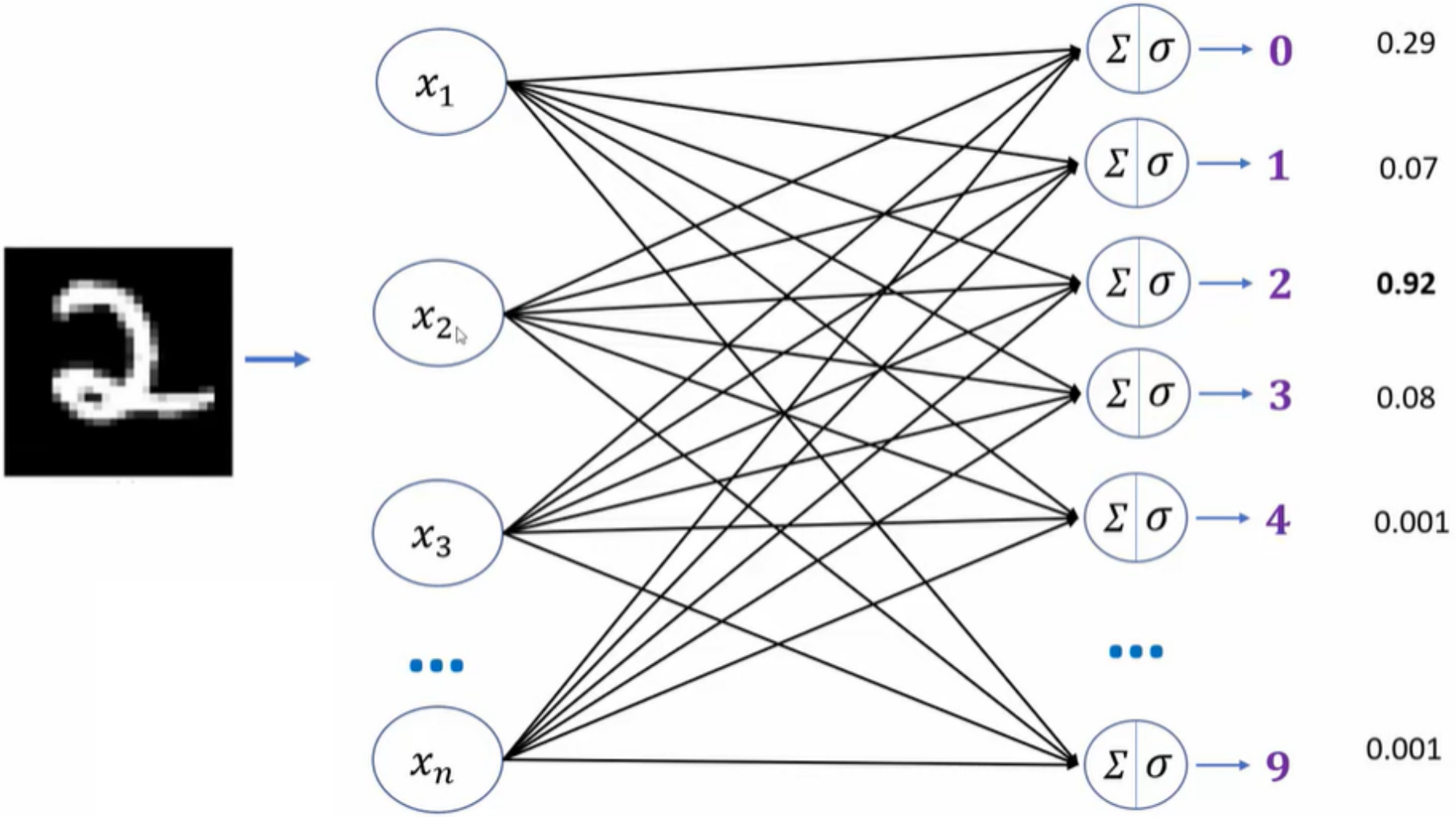
```
train-images-idx3-ubyte.gz:  training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:  training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:   test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:   test set labels (4542 bytes)
```

http://yann.lecun.com/exdb/mnist/


4 (4)
1 (1)
0 (0)
7 (7)
8 (8)
1 (1)
2 (2)
7 (7)
1 (1)

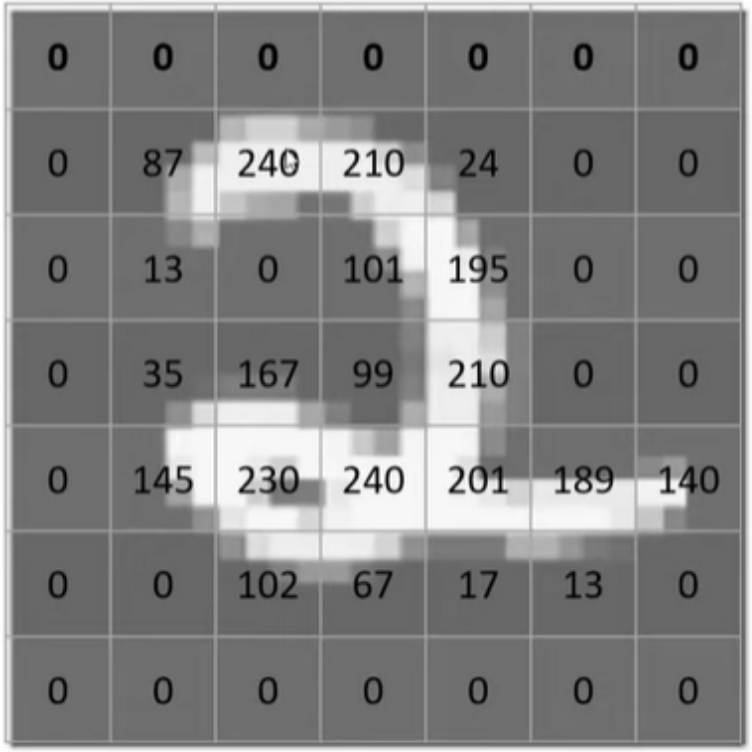

x_1
x_2
x_3
x_n
Σ σ
0
1
2
3
4
9
0.02
0.31
0.007
0.08
0.82
0.02


x_1
x_2
x_3
x_n
Σ σ
0
1
2
3
4
9

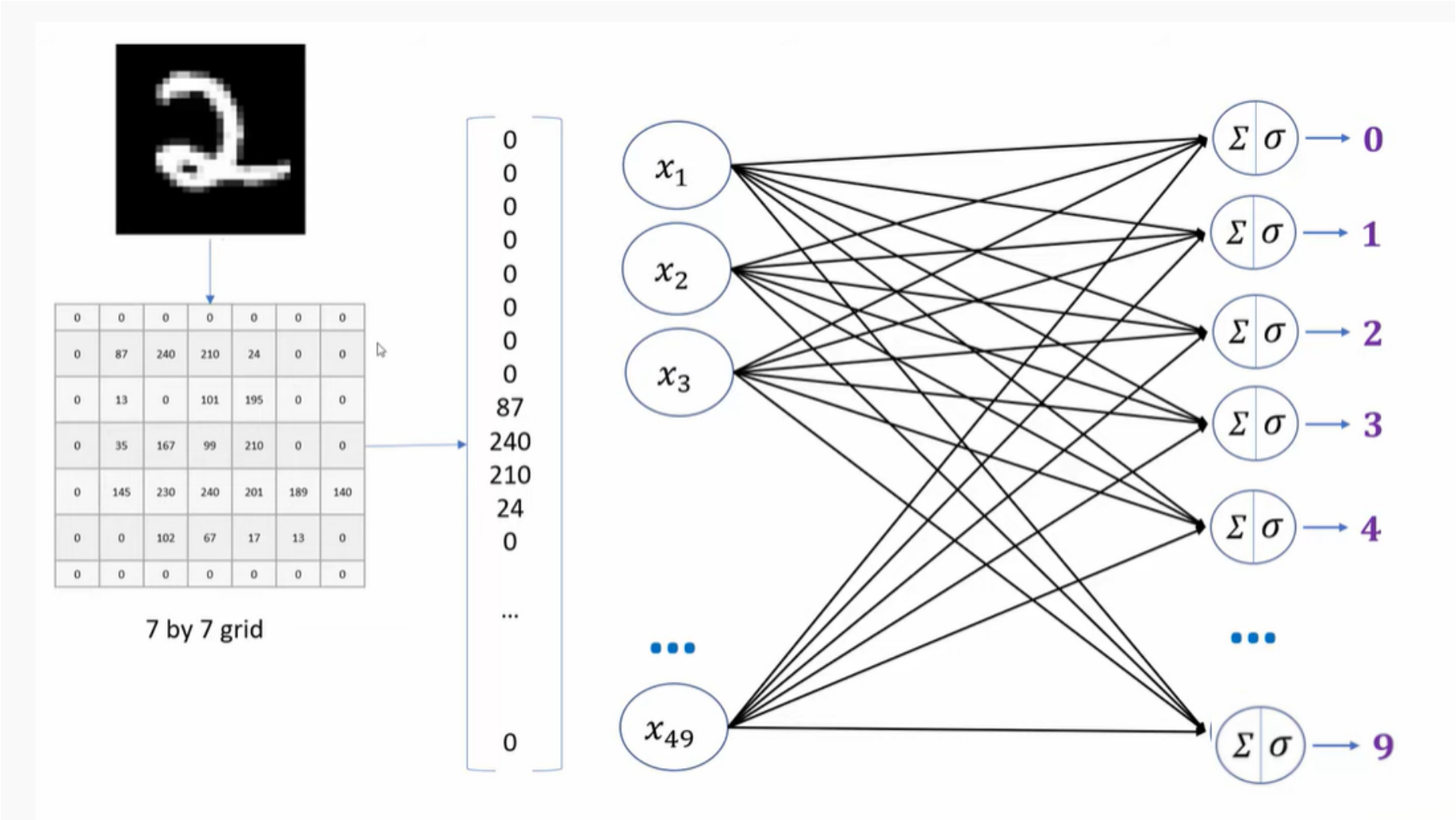

x_1
x_2
x_3
x_n
Σ σ
0
1
2
3
4
9
0.29
0.07
0.92
0.08
0.001
0.001

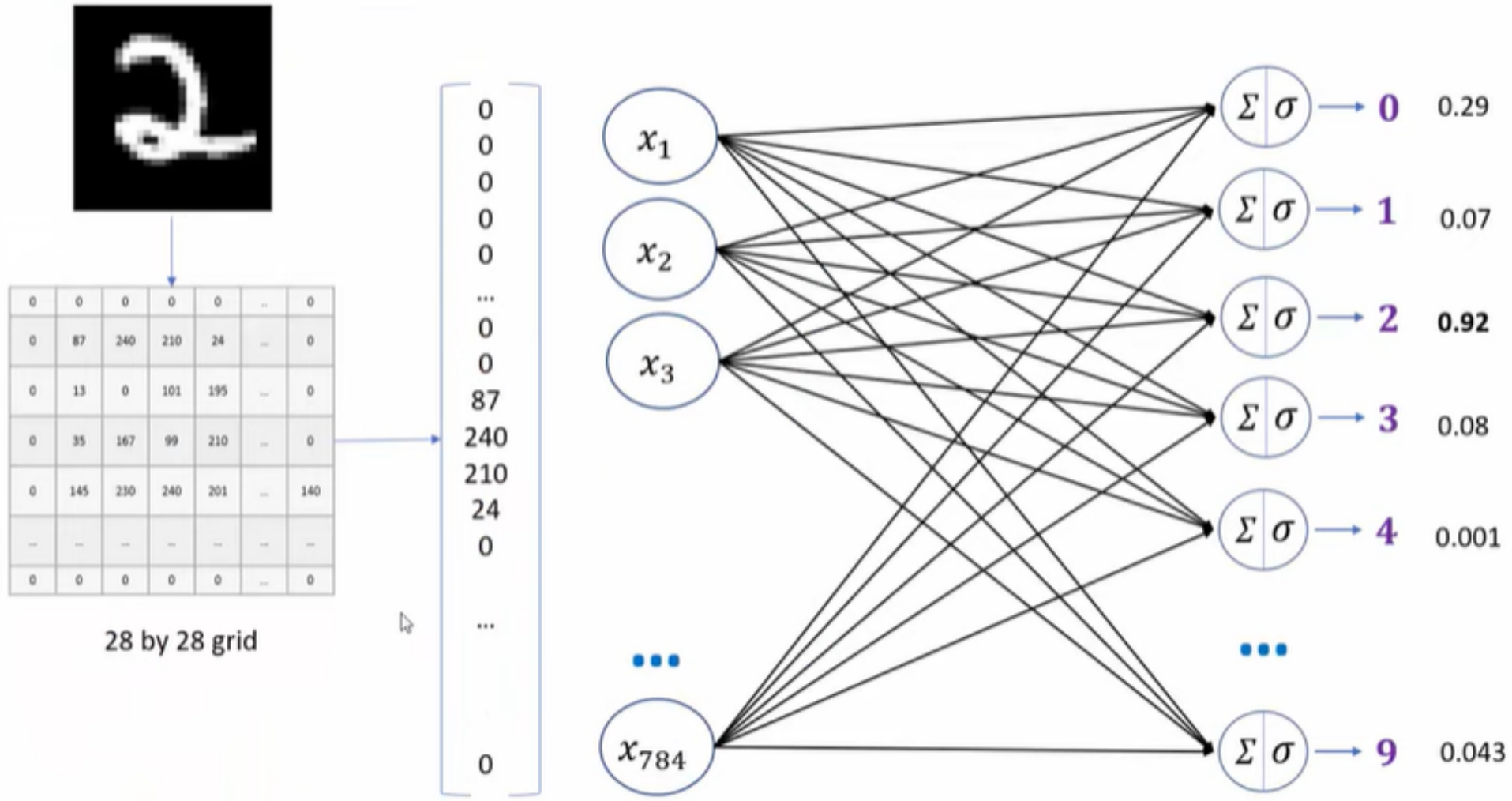
0 – Black, 255 - White

0 – Black, 255 - White



0	0	0	0	0	0	0
0	87	240	210	24	0	0
0	13	0	101	195	0	0
0	35	167	99	210	0	0
0	145	230	240	201	189	140
0	0	102	67	17	13	0
0	0	0	0	0	0	0


0 0 0 0 0 0 0
0 87 240 210 24 0 0
0 13 0 101 195 0 0
0 35 167 99 210 0 0
0 145 230 240 201 189 140
0 0 102 67 17 13 0
0 0 0 0 0 0 0
7 by 7 grid
0
0
0
0
0
0
0
0
87
240
210
24
0
...
0
x_1
x_2
x_3
x_{49}
Σ σ
0
1
2
3
4
9



for different types of loss function in tensorflow, see the documentation https://www.tensorflow.org/api_docs/python/tf/keras/losses

```
import keras
keras.backend.backend()  'tensorflow'

from keras.models import Sequential
from keras.layers import Flatten, Dense, Activation

model = Sequential()
model.add(Flatten(input_shape=[28, 28]))
model.add(Dense(100, activation="relu"))
model.add(Dense(10, activation="softmax"))
```

for different types of optimization method in tensorflow, see the documentation
https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Python code: Deep Neural Network

```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
import numpy as np                    # advanced math library
import matplotlib.pyplot as plt       # MATLAB like plotting
    routines
import random                         # for generating random
    numbers

from keras.datasets import mnist # MNIST dataset is included
    in Keras
from keras.models import Sequential # Model type to be used

from keras.layers.core import Dense, Dropout, Activation #
    Types of layers to be used in our model
from keras.utils import np_utils                    # NumPy
    related tools
# The MNIST data is split between 60,000 28 x 28 pixel
    training images and 10,000 28 x 28 pixel images
```

Python code: Deep Neural Network

```
# See length of test data set
len(X_test)
# See the shape of the training data set
X_train.shape
# See data at first position
X_train[0]
# See Train data at 3rd position
plt.matshow(X_train[2])
# See target data at 3rd position
y_train[2]
# See target data at first 5th position
y_train[:5]

# Scalling
X_train=X_train/255
X_test=X_test/255

# Reshape data
```

Python code: Deep Neural Network

```
# Build and compile one- hidden layer ANN model
model = Sequential()
model.add(Dense(10,input_shape=(784,),activation='sigmoid'))
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy']
)
model.fit(X_train_flattened,y_train,epochs=5)
```

Python code: Deep Neural Network

```
# Evaluate the model
model.evaluate(X_test_flattened,y_test)

# Model prediction
model.predict(X_test_flattened)

# See the actual test data
plt.matshow(X_test[0])

# See the predicted test data
y_predicted=model.predict(X_test_flattened)
# See the score for predicted value

# See the predicted value
np.argmax(y_predicted[0])
```

Python code: Deep Neural Network

```
# See predicted value
y_predicted_labels=[np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]

# See actual value
y_test[:5]

# Confusion Matrix
cm=tf.math.confusion_matrix(labels=y_test,predictions=y_predicte
cm
```

```
# See the heat plot
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
y_predict=y_predicted_labels[:len(y_test)]
print('Classification
    Report:\n',classification_report(y_test,y_predict))
print('Confusion Matrix:\n',confusion_matrix(y_test,
    y_predict))
print('Accuracy Score:',accuracy_score(y_test,y_predict))
```

Python code: Deep Neural Network

```
# Now add hidden layer
# The first hidden layer is a set of 512 nodes (artificial
    neurons).
# Each node will receive an element from each input vector
    and apply some weight and bias to it.
model=Sequential([Dense(100,input_shape=(784,),activation='relu'
model.compile(
optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy']
)
model.fit(X_train_flattened,y_train,epochs=5)
```

Python code: Deep Neural Network

```
# See the Confusiton Matrix
y_predicted=model.predict(X_test_flattened)
y_predicted_labels=[np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]
cm=tf.math.confusion_matrix(labels=y_test,
predictions=y_predicted_labels)
cm
# See the plot
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Python code: Deep Neural Network

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
y_predict=y_predicted_labels[:len(y_test)]
print('Classification
    Report:\n',classification_report(y_test,y_predict))
print('Confusion Matrix:\n',confusion_matrix(y_test,
    y_predict))
print('Accuracy Score:',accuracy_score(y_test,y_predict))
```

Python code: Deep Neural Network

```
# Alternatively use the Flatten function
model=keras.Sequential([
keras.layers.Flatten(input_shape=(28,28)),
keras.layers.Dense(100,input_shape=(784,),activation='relu'),
keras.layers.Dense(10,input_shape=(784,),activation='sigmoid')
])
model.compile(
optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy']
)
model.fit(X_train_flattened,y_train,epochs=5)
```

Python code: for iris dataset

Python code: for iris dataset

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
# Importing the dependancies
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import StandardScaler
# import some data to play with
iris = datasets.load_iris()
print(iris)

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features.
```

Exam Questions

1. What is meant by ANN? What do you understand by Backpropagation?
2. What is a Neural Network? What ss the role of activation functions in a Neural Network? What ss Data Normalization, and Why do we need It?
3. What is overfitting? How do you solve overfitting problem?
4. What is dropout? Why do we use dropout?
5. What Do You Understand by Backpropagation? What Are the Softmax and ReLU Functions?
6. How are weights initialized in a network? How do you update weight in ANN?
7. What do you understand by Leaky ReLU activation function?
8. Explain two ways to deal with the vanishing gradient problem in a deep neural network.

Chapter 12: Clustering

12 Chapter 12: Clustering

12.1 Problem & Motivation

12.2 Hierarchical Clustering

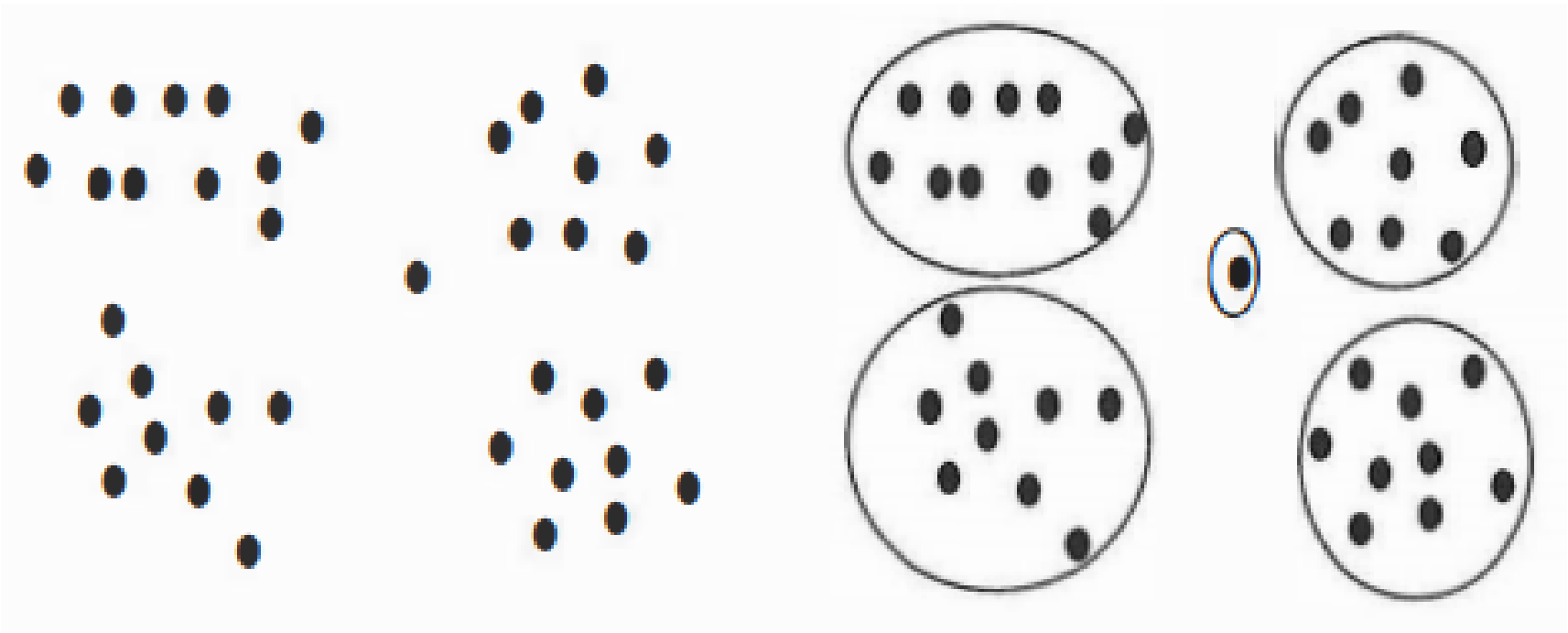
12.3 The k-Means Clustering:Example (step by step)

12.4 Python Code: Clustering

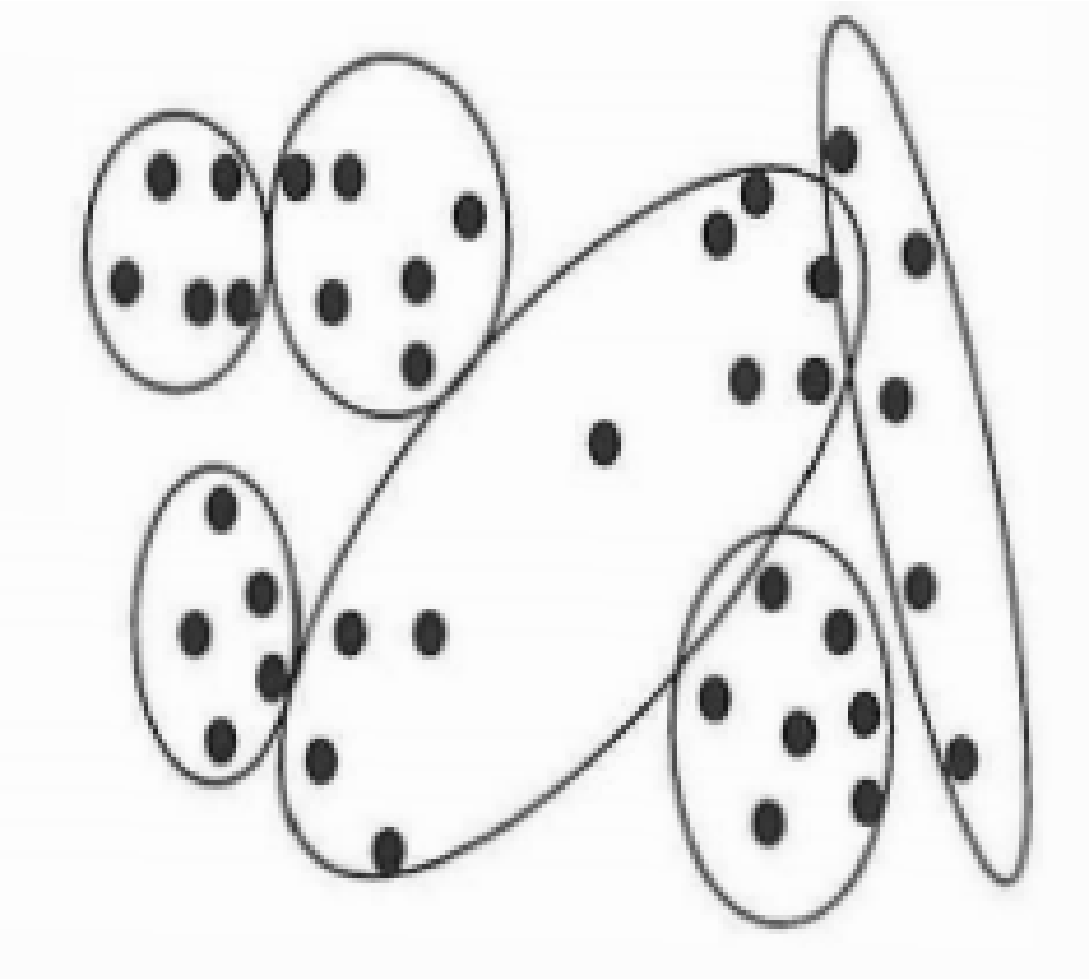
12.5 R Code: Clustering

- cluster analysis: we search for patterns in a data set by grouping the (multivariate) observations into clusters
- you don't know your target variable or class label
- the goal is to find an optimal grouping for which the observations or objects within each cluster are similar, but the clusters are dissimilar to each other
- Classification vs. Clustering
  - in classification analysis, we allocate the observations to a known number of predefined groups or populations
  - in clustering analysis, no prior knowledge about the
    - number of clusters
    - member of themselves in advance
    - meaning of clusters
  - a unsupervised learning

Clustering



Clustering



Dissimilarity or Distance Measures for Pairs of Items

alternatively, distance measure measure how unlike or dissimilar objects $t_i = \langle t_{i1}, \ldots, t_{ip} \rangle$ and $t_j = \langle t_{j1}, \ldots, t_{jp} \rangle$ are

1. Euclidean (straight-line) distance:

$$dis(t_i, t_j) = \sqrt{\sum_{h=1}^{p} (t_{ih} - t_{jh})^2} = \sqrt{(\mathbf{t}_i - \mathbf{t}_j)^T (\mathbf{t}_i - \mathbf{t}_j)}$$

   where $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{ip}]^T$ and $\mathbf{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jp}]^T$

2. Mahalanobis or Statistical distance:

$$dis(t_i, t_j) = \sqrt{(\mathbf{t}_i - \mathbf{t}_j)^T \mathbf{S}^{-1} (\mathbf{t}_i - \mathbf{t}_j)}$$

   where $\mathbf{S}$ contains the sample variances and covariances

3. Manhattan or city-block distance:

$$dis(t_i, t_j) = \sum_{h=1}^{p} |t_{ih} - t_{jh}|$$

Distance Measures

the dissimilar objects $t_i = \langle t_{i1}, \ldots, t_{ip} \rangle$ and $t_j = \langle t_{j1}, \ldots, t_{jp} \rangle$ are

4. Minkowski metric:

$$dis(t_i, t_j) = \left[\sum_{h=1}^{p} |t_{ih} - t_{jh}|^m \right]^{\frac{1}{m}}$$

   - for $m = 1$, $dis(t_i, t_j)$ measures the "Manhattan or City-block" distance between two points in p dimensions
   - for $m = 2$, $dis(t_i, t_j)$ becomes the Euclidean distance+

5. Canberra metric:

$$dis(t_i, t_j) = \sum_{i=1}^{p} \frac{|t_{ih} - t_{jh}|}{(t_{ih} + t_{jh})}$$

6. Czekanowski coefficient:

$$dis(t_i, t_j) = 1 - \frac{2 \sum_{i=1}^{p} \min(t_{ih}, t_{jh})}{\sum_{i=1}^{p} (t_{ih} + t_{jh})}$$

Similarity or Proximity Measure for Pairs of Items

- the use of similarity measures is well known to anyone who has performed internet searches using a search engine
- similarity of two objects $t_i = \langle t_{i1}, \ldots, t_{ip} \rangle$ and $t_j = \langle t_{j1}, \ldots, t_{jp} \rangle$ are
  1. Dice:
  $$sim(t_i, t_j) = \frac{2 \sum_{h=1}^{p} t_{ih} t_{jh}}{\sum_{h=1}^{p} t_{ih}^2 + \sum_{h=1}^{p} t_{jh}^2}$$
  2. Jaccard:
  $$sim(t_i, t_j) = \frac{\sum_{h=1}^{p} t_{ih} t_{jh}}{\sum_{h=1}^{p} t_{ih}^2 + \sum_{h=1}^{p} t_{jh}^2 - \sum_{h=1}^{p} t_{ih} t_{jh}}$$
  3. Cosine:
  $$sim(t_i, t_j) = \frac{\sum_{h=1}^{p} t_{ih} t_{jh}}{\sum_{h=1}^{p} t_{ih}^2 \sum_{h=1}^{p} t_{jh}^2}$$
  4. Overlap:
  $$sim(t_i, t_j) = \frac{\sum_{h=1}^{p} t_{ih} t_{jh}}{\min\left(\sum_{h=1}^{p} t_{ih}^2, \sum_{h=1}^{p} t_{jh}^2\right)}$$

Similarity or Proximity Measure for Pairs of Items of Binary Variables

1. when items are often compared on the basis of the presence or absence of certain characteristics, then the presence or absence of a characteristic can be described mathematically by introducing a binary variable, which assumes the value 1 if the characteristic is present and the value 0 if the characteristic is absent
2. for $p = 5$ binary variables, for instance, the "scores" for two items i and k might be arranged as follows:

	variable				
	1	2	3	4	5
Item i	1	0	0	1	1
Item k	1	1	0	1	0

3. in this case, there are two 1-1 matches, one 0-0 match, and two mismatches

Similarity or Proximity Measure for Pairs of Items of Binary Variables

Let x_{ij} be the score (1 or 0) of the jth binary variable on the ith item and x_{kj} be the score (again, 1 or 0) of the jth variable on the kth item, $j = 1, 2, \ldots, p$. Consequently,

$$(x_{ij} - x_{kj})^2 = \begin{cases} 0 & \text{if } x_{ij} = x_{kj} = 1 \quad \text{or} \quad x_{ij} = x_{kj} = 0 \\ 1 & \text{if } x_{ij} \neq x_{kj} \end{cases} \tag{12-6}$$

and the squared Euclidean distance, $\sum_{j=1}^{p} (x_{ij} - x_{kj})^2$, provides a count of the number of mismatches. A large distance corresponds to many mismatches—that is, dissimilar items. From the preceding display, the square of the distance between items i and k would be

$$\begin{aligned} \sum_{j=1}^{5} (x_{ij} - x_{kj})^2 &= (1-1)^2 + (0-1)^2 + (0-0)^2 + (1-1)^2 + (1-0)^2 \\ &= 2 \end{aligned}$$

Similarity or Proximity Measure for Pairs of Items of Binary Variables

- in general, we can made a table

		Item k		
		1	0	Total
item i	1	a	b	$a+b$
	0	c	d	$c+d$
Total		$a+c$	$b+d$	$p=a+b+c+d$

- in this table, a represents the frequency of $1-1$ matches, b is the frequency of $1-0$ mismatches, and so forth

Table 12.1 Similarity Coefficients for Clustering Items*

Coefficient	Rationale
1. $\frac{a+d}{p}$	Equal weights for 1–1 matches and 0–0 matches.
2. $\frac{2(a+d)}{2(a+d)+b+c}$	Double weight for 1–1 matches and 0–0 matches.
3. $\frac{a+d}{a+d+2(b+c)}$	Double weight for unmatched pairs.
4. $\frac{a}{p}$	No 0–0 matches in numerator.
5. $\frac{a}{a+b+c}$	No 0–0 matches in numerator or denominator. (The 0–0 matches are treated as irrelevant.)
6. $\frac{2a}{2a+b+c}$	No 0–0 matches in numerator or denominator. Double weight for 1–1 matches.
7. $\frac{a}{a+2(b+c)}$	No 0–0 matches in numerator or denominator. Double weight for unmatched pairs.
8. $\frac{a}{b+c}$	Ratio of matches to mismatches with 0–0 matches excluded.

*[p binary variables; see (12-7).]

Similarity for Binary Variables

Example 12.1 (Calculating the values of a similarity coefficient) Suppose five individuals possess the following characteristics:

	Height	Weight	Eye color	Hair color	Handedness	Gender
Individual 1	68 in	140 lb	green	blond	right	female
Individual 2	73 in	185 lb	brown	brown	right	male
Individual 3	67 in	165 lb	blue	blond	right	male
Individual 4	64 in	120 lb	brown	brown	right	female
Individual 5	76 in	210 lb	brown	brown	left	male

Define six binary variables $X_1, X_2, X_3, X_4, X_5, X_6$ as

$$X_1 = \begin{cases} 1 & \text{height} \geq 72 \text{ in.} \\ 0 & \text{height} < 72 \text{ in.} \end{cases} \qquad X_4 = \begin{cases} 1 & \text{blond hair} \\ 0 & \text{not blond hair} \end{cases}$$

$$X_2 = \begin{cases} 1 & \text{weight} \geq 150 \text{ lb} \\ 0 & \text{weight} < 150 \text{ lb} \end{cases} \qquad X_5 = \begin{cases} 1 & \text{right handed} \\ 0 & \text{left handed} \end{cases}$$

$$X_3 = \begin{cases} 1 & \text{brown eyes} \\ 0 & \text{otherwise} \end{cases} \qquad X_6 = \begin{cases} 1 & \text{female} \\ 0 & \text{male} \end{cases}$$

Similarity for Binary Variables

The scores for individuals 1 and 2 on the $p = 6$ binary variables are

		X_1	X_2	X_3	X_4	X_5	X_6
Individual	1	0	0	0	1	1	1
	2	1	1	1	0	1	0

and the number of matches and mismatches are indicated in the two-way array

		Individual 2		
		1	0	Total
Individual 1	1	1	2	3
	0	3	0	3
	Totals	4	2	6

Similarity for Binary Variables

Employing similarity coefficient 1, which gives equal weight to matches, we compute

$$\frac{a+d}{p} = \frac{1+0}{6} = \frac{1}{6}$$

Continuing with similarity coefficient 1, we calculate the remaining similarity numbers for pairs of individuals. These are displayed in the 5×5 symmetric matrix

$$\begin{array}{cc} & \text{Individual} \\ & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \text{Individual} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 1 & & & & \\ \frac{1}{6} & 1 & & & \\ \frac{4}{6} & \frac{3}{6} & 1 & & \\ \frac{4}{6} & \frac{3}{6} & \frac{2}{6} & 1 & \\ 0 & \frac{5}{6} & \frac{2}{6} & \frac{2}{6} & 1 \end{bmatrix} \end{array}$$

Based on the magnitudes of the similarity coefficient, we should conclude that individuals 2 and 5 are most similar and individuals 1 and 5 are least similar. Other pairs fall between these extremes. If we were to divide the individuals into two relatively homogeneous subgroups on the basis of the similarity numbers, we might form the subgroups (1 3 4) and (2 5).

Clustering Techniques

- Hierarchical Clustering
  — proceed by either a series of successive mergers or a series of successive divisions
  - agglomerative clustering method
  - divisive clustering method
- Nonhierarchical Clustering
  — k-means clustering

Hierarchical Clustering

1. Hierarchical agglomerative clustering (HAC) $\Rightarrow$ bottom - up
   - in each step, an observation or a cluster of observations is merged into another cluster
   - in this process, the number of clusters shrinks and the clusters themselves grow larger
   - in this method, the two closest clusters are merged into a single new cluster
   - start with n clusters (individual items) and end with one single cluster containing the entire data set
2. Hierarchical divisive clustering (HDC) $\Rightarrow$ top - down
   - starts with a single cluster containing all n items and partitions a cluster into two clusters at each step
   - the end result of the divisive approach is n clusters of one item each

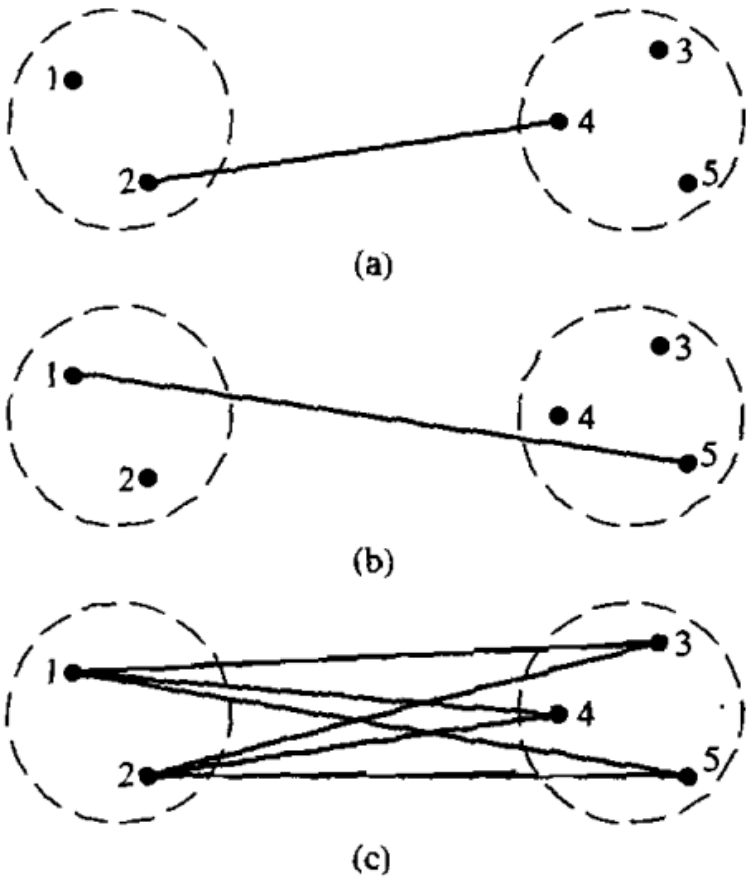
agglomerative methods are more commonly used than divisive methods

Linkage Methods

- since an agglomerative hierarchical procedure combines the two closest clusters at each step, we must consider the question of measuring the similarity or dissimilarity of two clusters

  linkage methods
  - Single Link (Nearest Neighbor): smallest distance between points
  - Complete Link (Farthest Neighbor): largest distance between points
  - Average Link: average distance between points
  - Centroid: distance between centroids ($centroid = C_m = \frac{\sum_{i=1}^{N} t_{mi}}{N}$)
- **dendrogram**: the results of a hierarchical clustering procedure can be displayed graphically using a tree diagram, which shows all the steps in the hierarchical procedure, including the distances at which clusters are merged

Linkage Methods



Cluster distance

d_{24}

d_{15}

$$\frac{d_{13} + d_{14} + d_{15} + d_{23} + d_{24} + d_{25}}{6}$$

Figure 12.2 Intercluster distance (dissimilarity) for (a) single linkage, (b) complete linkage, and (c) average linkage.

HAC: Example of crime rates

Example 14.3.2(a). Hartigan (1975a, p. 28) compared the crime rates per 100,000 population for various cities. The data are in Table 14.1 (taken from the 1970 U.S.

Table 14.1. City Crime Rates per 100,000 Population

City	Murder	Rape	Robbery	Assault	Burglary	Larceny	Auto Theft
Atlanta	16.5	24.8	106	147	1112	905	494
Boston	4.2	13.3	122	90	982	669	954
Chicago	11.6	24.7	340	242	808	609	645
Dallas	18.1	34.2	184	293	1668	901	602
Denver	6.9	41.5	173	191	1534	1368	780
Detroit	13.0	35.7	477	220	1566	1183	788
Hartford	2.5	8.8	68	103	1017	724	468
Honolulu	3.6	12.7	42	28	1457	1102	637
Houston	16.8	26.6	289	186	1509	787	697
Kansas City	10.8	43.2	255	226	1494	955	765
Los Angeles	9.7	51.8	286	355	1902	1386	862
New Orleans	10.3	39.7	266	283	1056	1036	776
New York	9.4	19.4	522	267	1674	1392	848
Portland	5.0	23.0	157	144	1530	1281	488
Tucson	5.1	22.9	85	148	1206	756	483
Washington	12.5	27.6	524	217	1496	1003	793

Example: HAC-Single Linkage (step by step solution)

- use single linkage method
- the distance matrix for the first 6 row points/items is

City	Distance between Cities					
Atlanta	0	536.6	516.4	590.2	693.6	716.2
Boston	536.6	0	447.4	833.1	915.0	881.1
Chicago	516.4	447.4	0	924.0	1073.4	971.5
Dallas	590.2	833.1	924.0	0	527.7	464.5
Denver	693.6	915.0	1073.4	527.7	0	358.7
Detroit	716.2	881.1	971.5	464.5	358.7	0

- the smallest distance is 358.7 between Denver and Detroit, and therefore these two cities are joined at the first step to form

$$C_1 = \{Denver, Detroit\}$$

Example: HAC-Single Linkage (step by step solution)

- in the next step, the distance matrix is calculated for Atlanta, Boston, Chicago, Dallas, and C_1

Atlanta	0	536.6	516.4	590.2	693.6
Boston	536.6	0	447.4	833.1	881.1
Chicago	516.4	447.4	0	924.0	971.5
Dallas	590.2	833.1	924.0	0	464.5
C_1	693.6	881.1	971.5	464.5	0

where the update the distance matrix *Atlanta* and C_1 is

$$\begin{aligned} dist(Atlanta, C_1) &= dist(Atlanta, \{Denver, Detroit\}) \\ &= \min[dist(Atlanta, Denver), dist(Atlanta, Detroit)] \\ &= \min[693.6, 716.2] = 693.6 \end{aligned}$$

- the smallest distance is 447.4 between Boston and Chicago and hence,

$$C_2 = \{Boston, Chicago\}$$

Example: HAC-Single Linkage (step by step solution)

- at the next step, the distance matrix is calculated for Atlanta, Dallas, C_1, and C_2:

Atlanta	0	516.4	590.2	693.6
C_2	516.4	0	833.1	881.1
Dallas	590.2	833.1	0	464.5
C_1	693.6	881.1	464.5	0

- the smallest distance is 464.5 between Dallas and C_1, so that

$$C_3 = \{Dallas, C_1\}$$

Example: HAC-Single Linkage (step by step solution)

- the distance matrix is

Atlanta	0	516.4	590.2
C_2	516.4	0	833.1
C_3	590.2	833.1	0

- the smallest distance is 516.4, which defines

$$C_4 = \{Atlanta, C_2\}$$

- the distance matrix for C_3 and C_4 is

	C_3	C_4
C_3	0	590.2
C_4	590.2	0

- the last cluster is given by

$$C_5 = \{C_3, C_4\}$$

Example: HAC-Single Linkage (step by step solution)

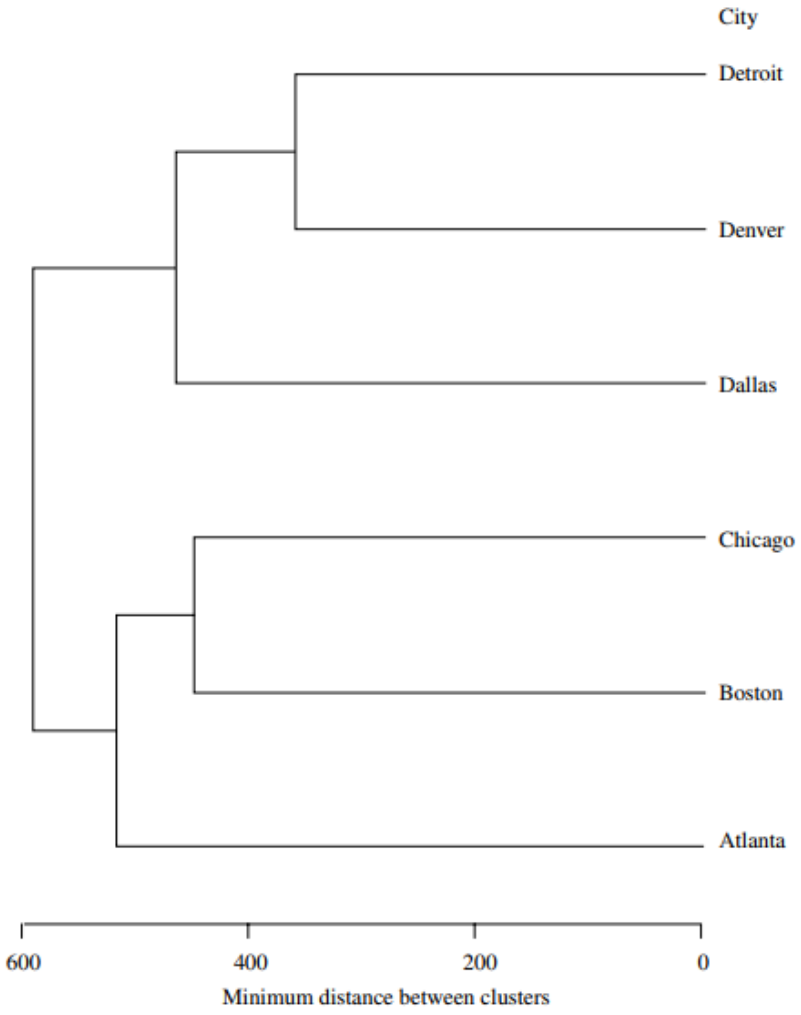


Figure: Dendrogram for single linkage of the first six observations in the city crime data in Table 14.1

Example: HAC-Single Linkage (step by step solution)

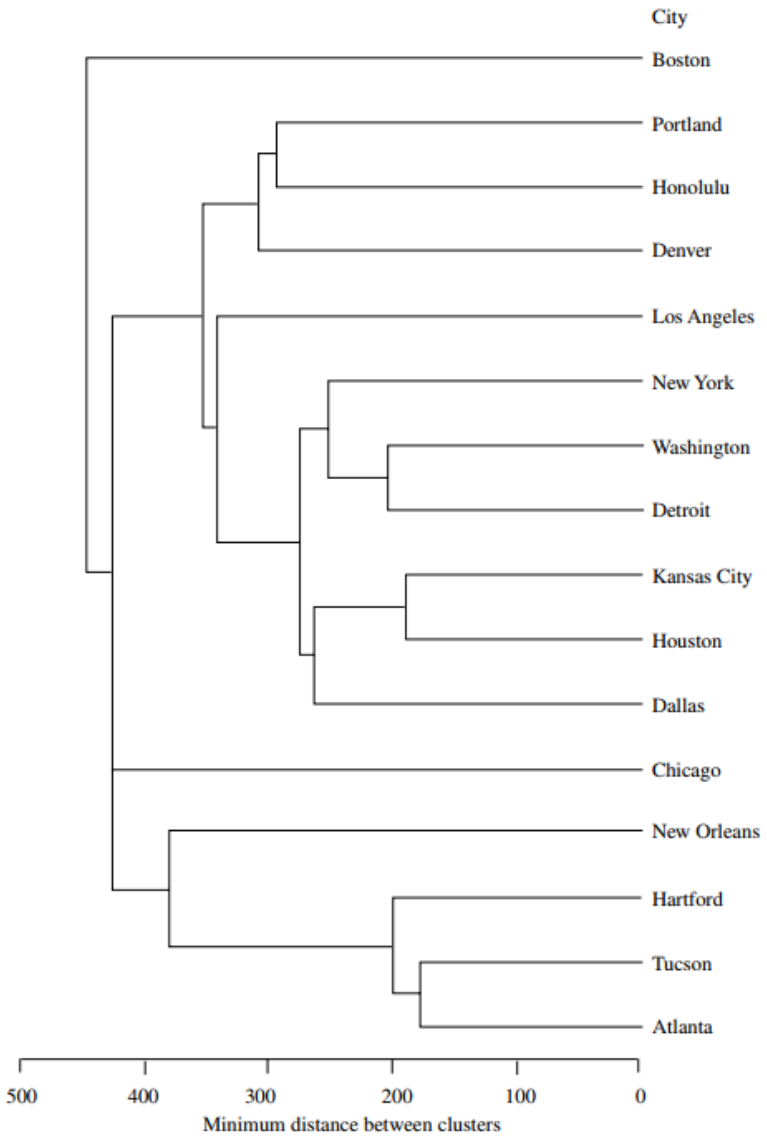


Figure 14.3. Dendrogram for single linkage of the complete city crime data from Table 14.1

Example: HAC-Complete Linkage (step by step solution)

- use complete linkage method
- the The distance matrix for the first 6 row points/items is

City	Distance between Cities					
Atlanta	0	536.6	516.4	590.2	693.6	716.2
Boston	536.6	0	447.4	833.1	915.0	881.1
Chicago	516.4	447.4	0	924.0	1073.4	971.5
Dallas	590.2	833.1	924.0	0	527.7	464.5
Denver	693.6	915.0	1073.4	527.7	0	358.7
Detroit	716.2	881.1	971.5	464.5	358.7	0

- the smallest distance is 358.7 between Denver and Detroit, and therefore these two cities are joined at the first step to form

$$C_1 = \{Denver, Detroit\}$$

- since the first cluster is based on the initial distance matrix, it will be the same regardless of which hierarchical clustering method is used

Example: HAC-Complete Linkage (step by step solution)

- in the next step, the distance matrix is calculated for Atlanta, Boston, Chicago, Dallas, and C_1

Atlanta	0	536.6	516.4	590.2	716.2
Boston	536.6	0	447.4	833.1	915.0
Chicago	516.4	447.4	0	924.0	1073.4
Dallas	590.2	833.1	924.0	0	527.7
C_1	716.2	915.0	1073.4	527.7	0

where the update the distance matrix *Atlanta* and C_1 is

$$\begin{aligned} dist(Atlanta, C_1) &= dist(Atlanta, \{Denver, Detroit\}) \\ &= \max[dist(Atlanta, Denver), dist(Atlanta, Detroit)] \\ &= \max[693.6, 716.2] = 716.2 \end{aligned}$$

- the smallest distance is 447.4 between Boston and Chicago and hence,

$$C_2 = \{Boston, Chicago\}$$

Example: HAC-Complete Linkage (step by step solution)

- at the next step, the distance matrix is calculated for Atlanta, Dallas, C_1, and C_2:

Atlanta	0	536.6	590.2	716.2
C_2	536.6	0	924.0	833.1
Dallas	590.2	924.0	0	527.7
C_1	693.6	881.1	527.7	0

- the smallest distance is 527.7 between Dallas and C_1, so that

$$C_3 = \{Dallas, C_1\}$$

Example: HAC-Complete Linkage (step by step solution)

- the distance matrix is

Atlanta	0	536.6	716.2
C_2	536.6	0	1073.4
C_3	590.2	1073.4	0

- the smallest distance is 536.6, which defines

$$C_4 = \{Atlanta, C_2\}$$

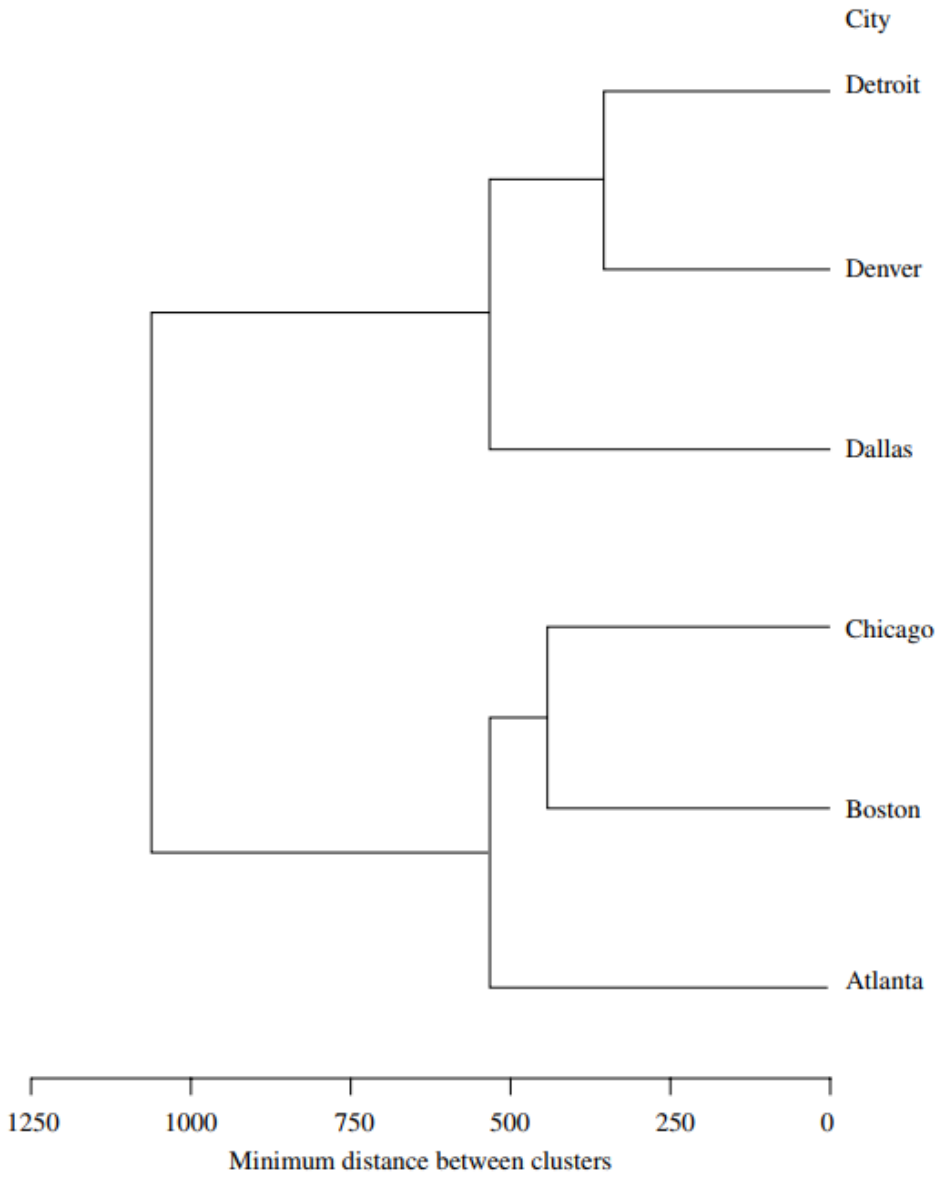
- the distance matrix for C_3 and C_4 is

	C_3	C_4
C_3	0	1073.4
C_4	1073.4	0

- the last cluster is given by

$$C5 = \{C_3, C_4\}$$

Example: HAC-Complete Linkage (step by step solution)



Example: HAC-Complete Linkage (step by step solution)

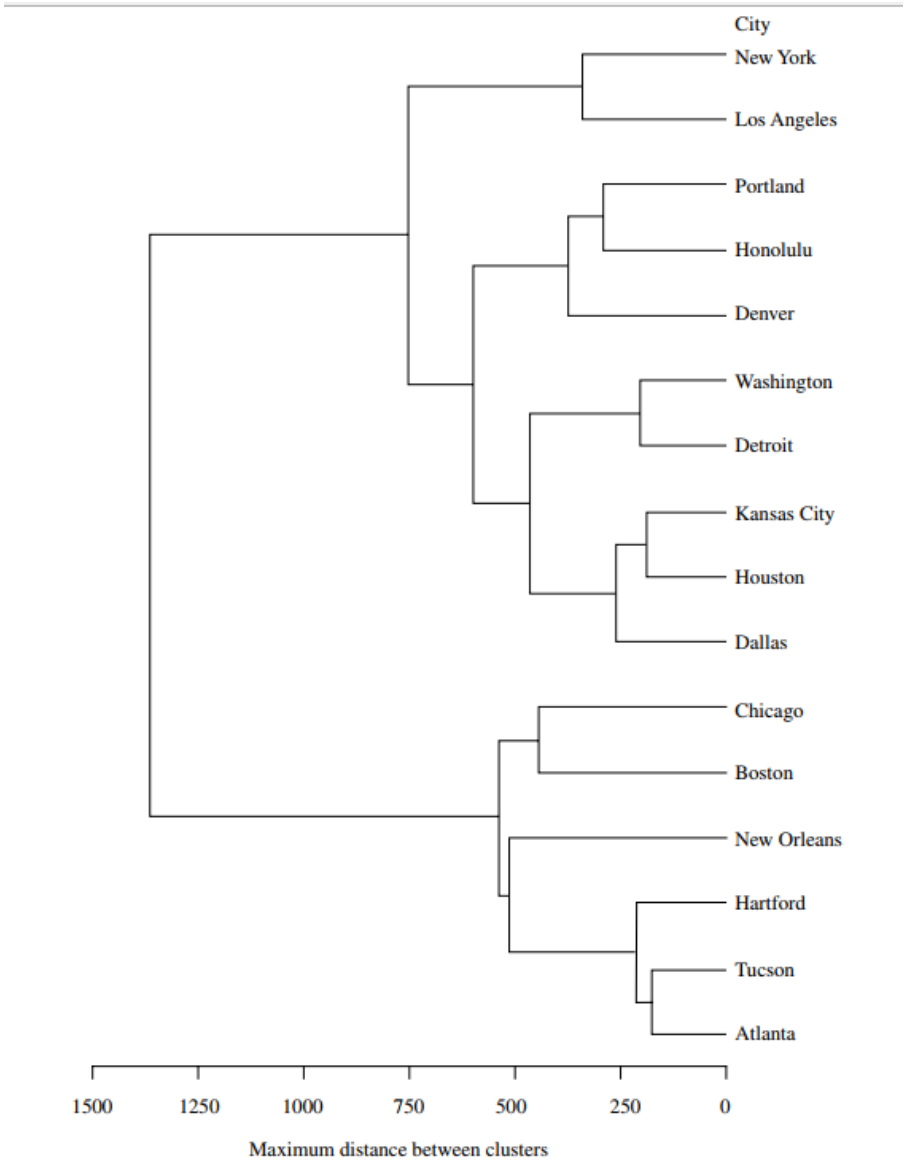


Figure 14.5. Dendrogram for complete linkage of the complete city crime data of Table 14.1

The k-Means Clustering

- the basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized
- there are several k-means algorithms available
- the standard algorithm is the Hartigan-Wong algorithm (Hartigan and Wong 1979), which defines the total within-cluster variation as the sum of squared distances Euclidean distances between items and the corresponding centroid:

$$W(C_k) = \sum_{t_i \in C_k} (t_i - m_k)^2$$

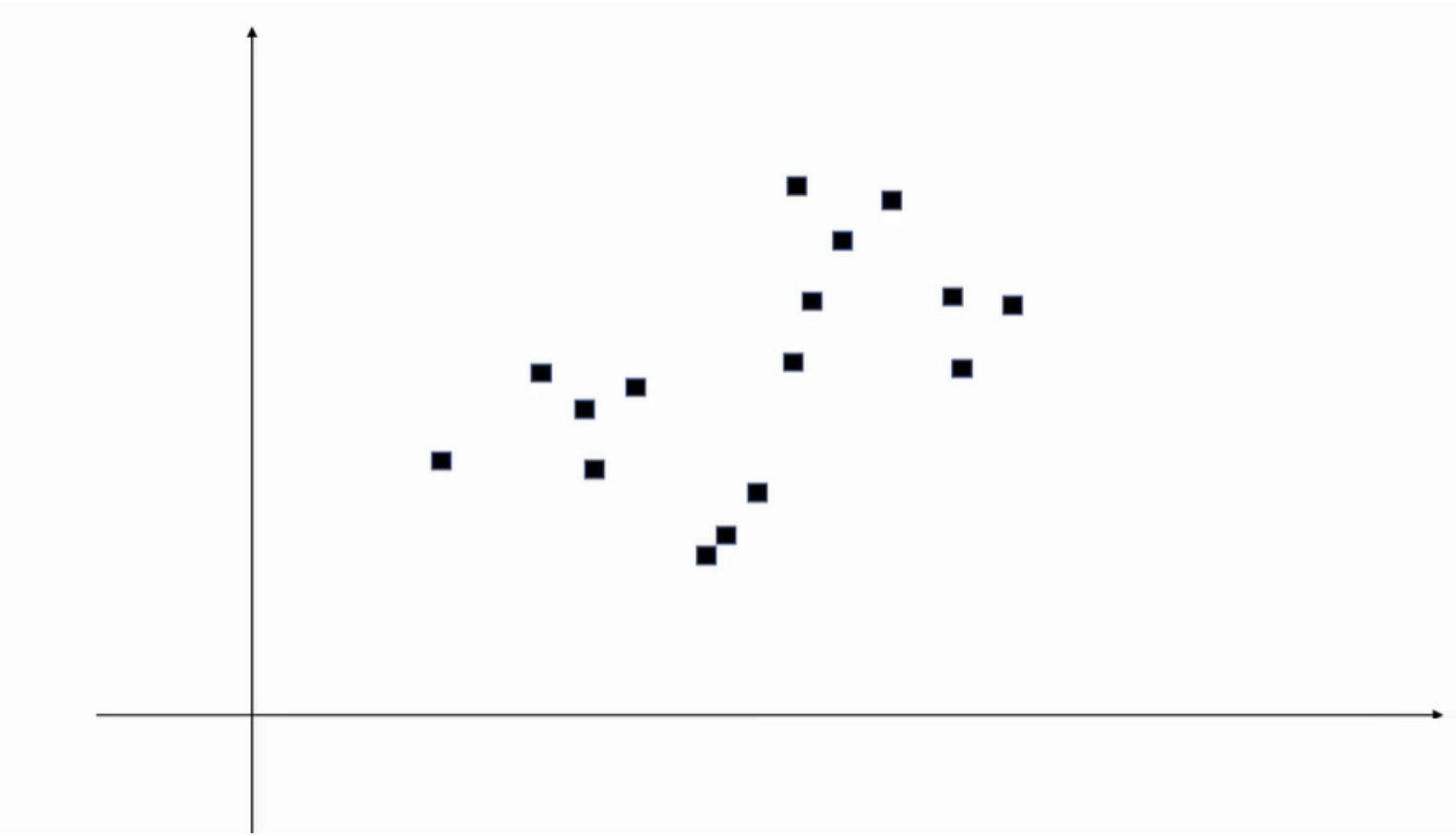
  - t_i design a data point belonging to the cluster C_k
  - m_k is the mean value of the points assigned to the cluster C_k

The k-Means Clustering

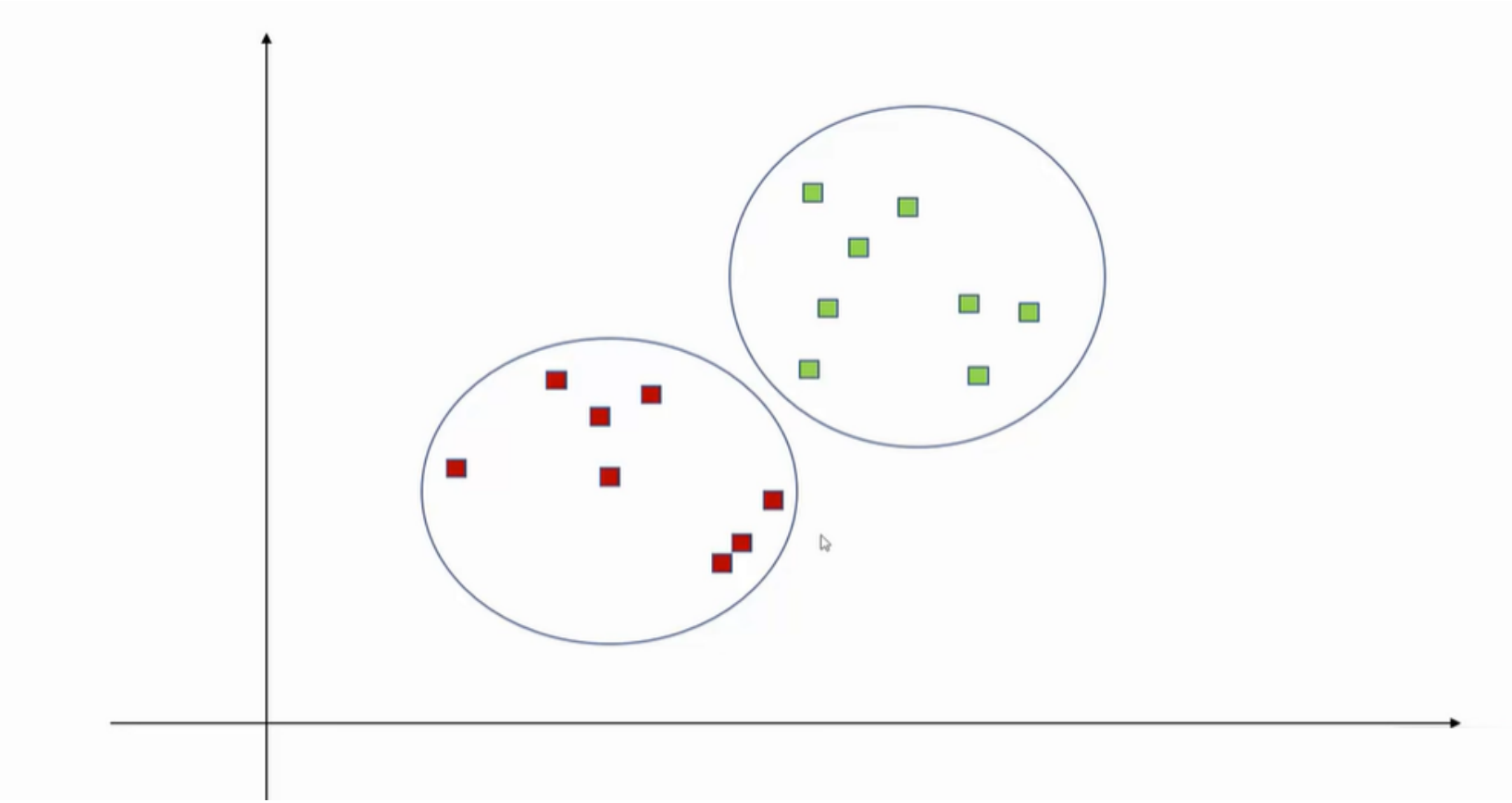
- each observation t_i is assigned to a given cluster such that the sum of squares (SS) distance of the observation to their assigned cluster centers m_k is a minimum.
- initial set of clusters randomly chosen
- the basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized.
- initial set of clusters randomly chosen
- iteratively, items are moved among sets of clusters until the desired set is reached
- high degree of similarity among elements in a cluster is obtained
- given a cluster $K_i = \{t_{i1}, t_{i2}, \ldots, t_{im}\}$, the cluster mean is

$$m_i = \frac{t_{i1} + t_{i2} + \ldots + t_{im}}{m}$$

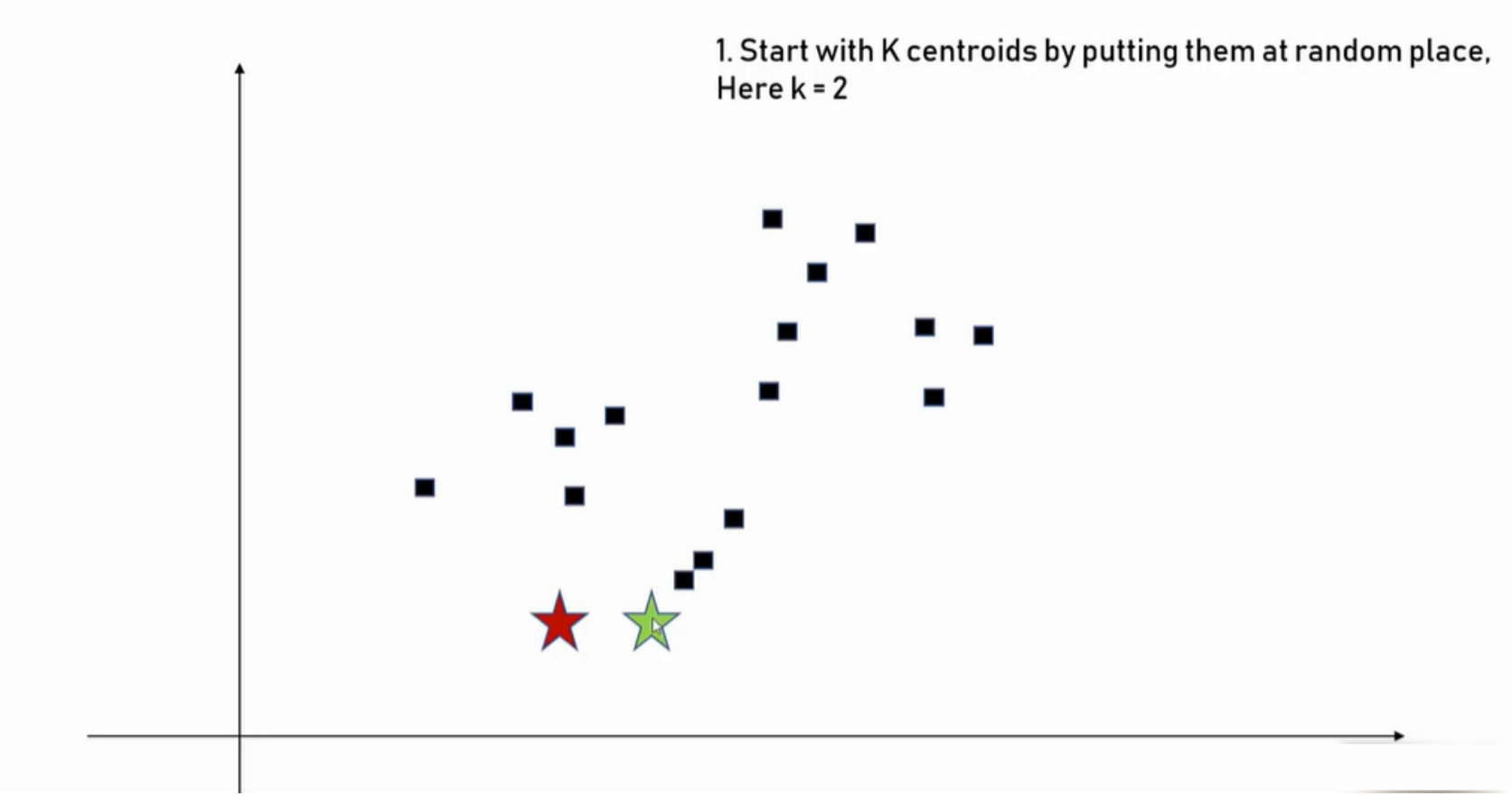
The k-Means Clustering



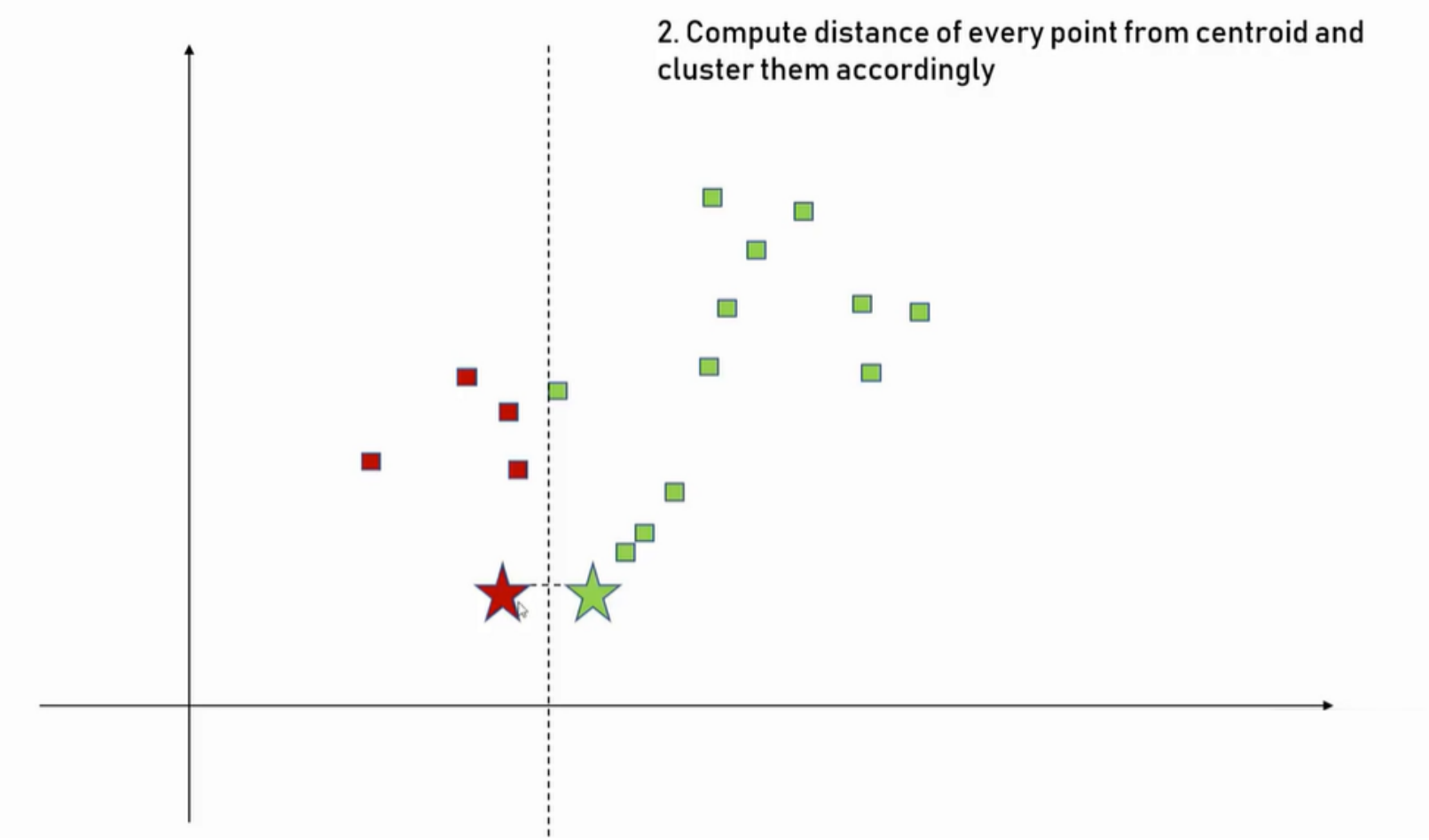
The k-Means Clustering



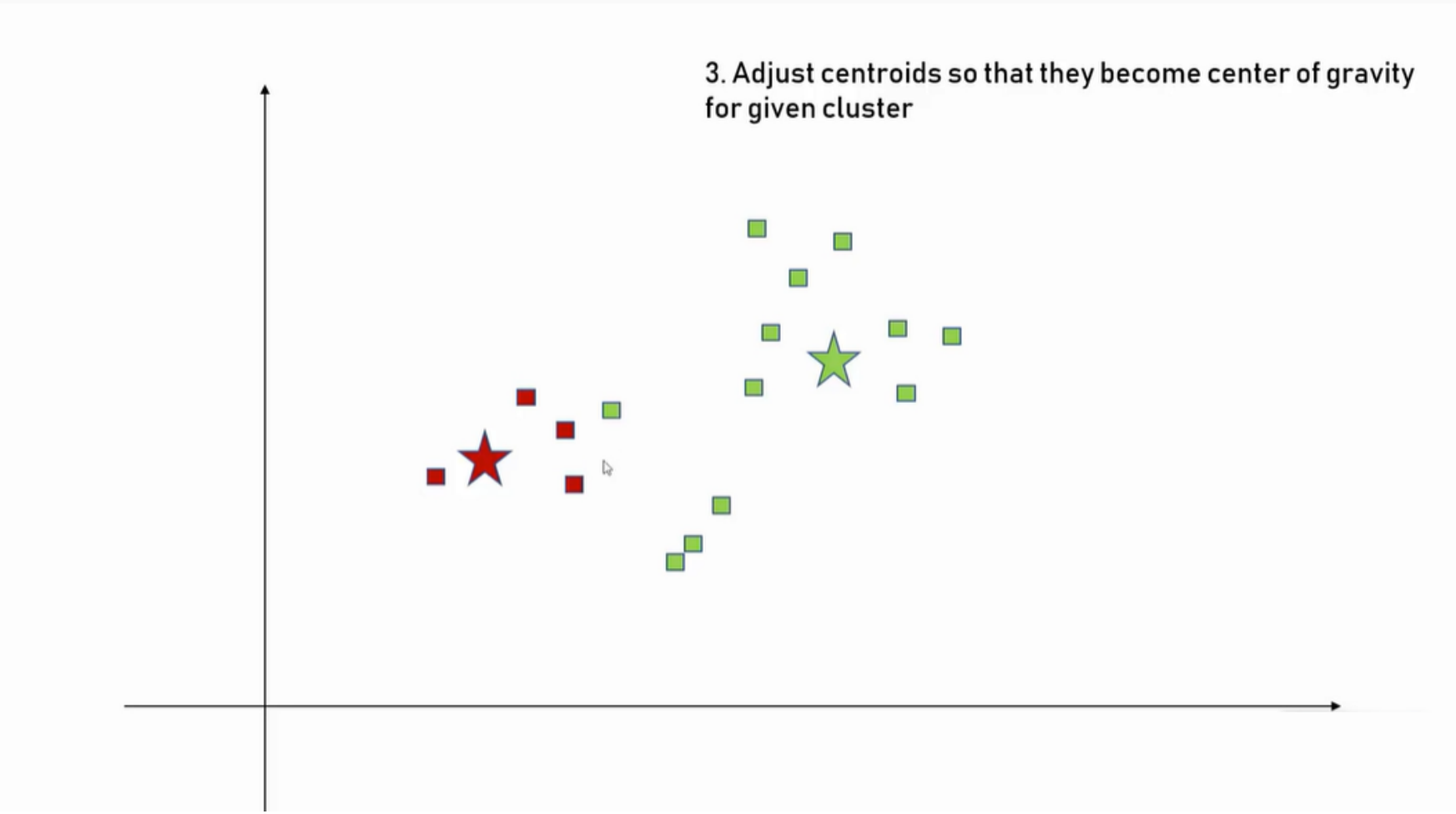
The k-Means Clustering



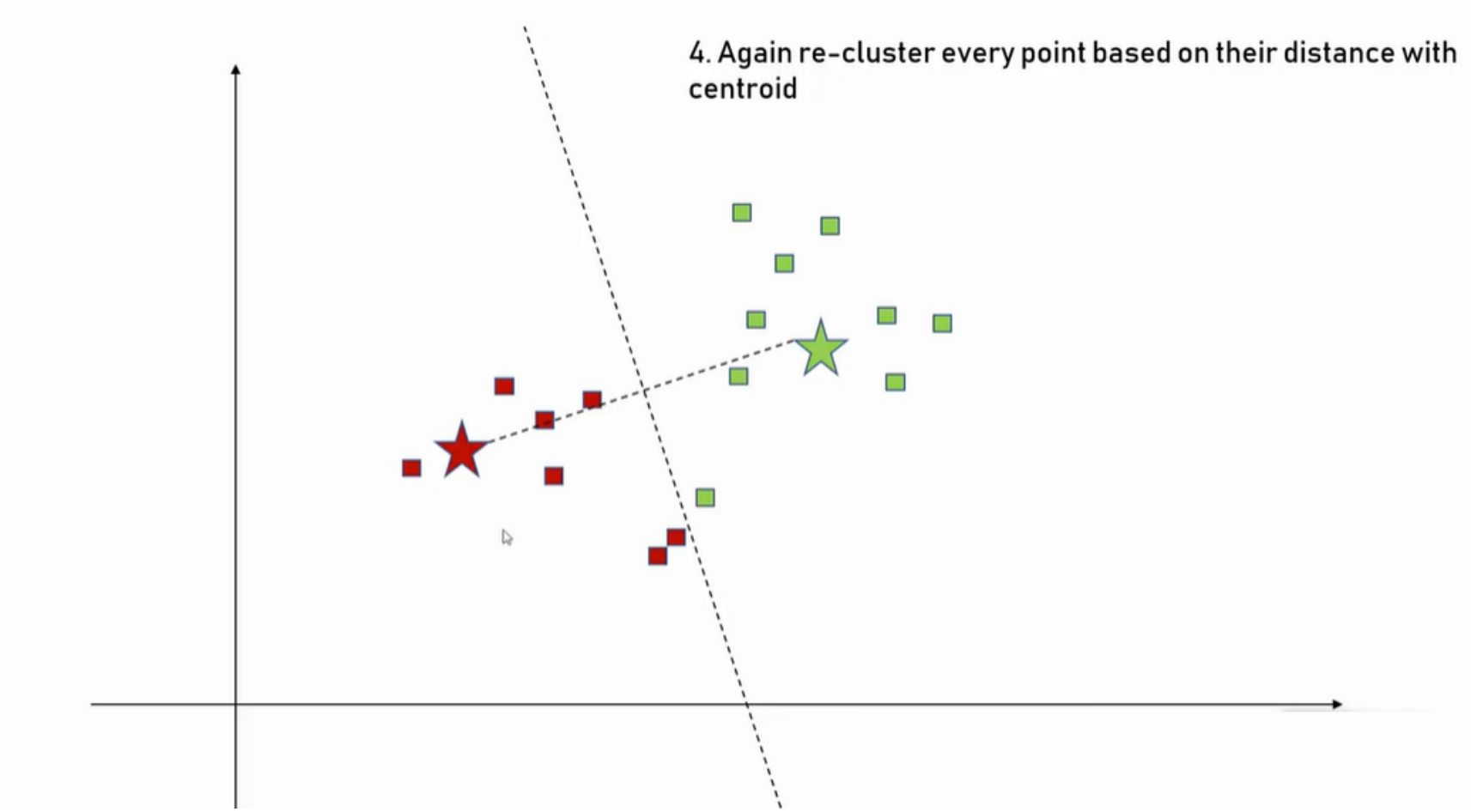
The k-Means Clustering



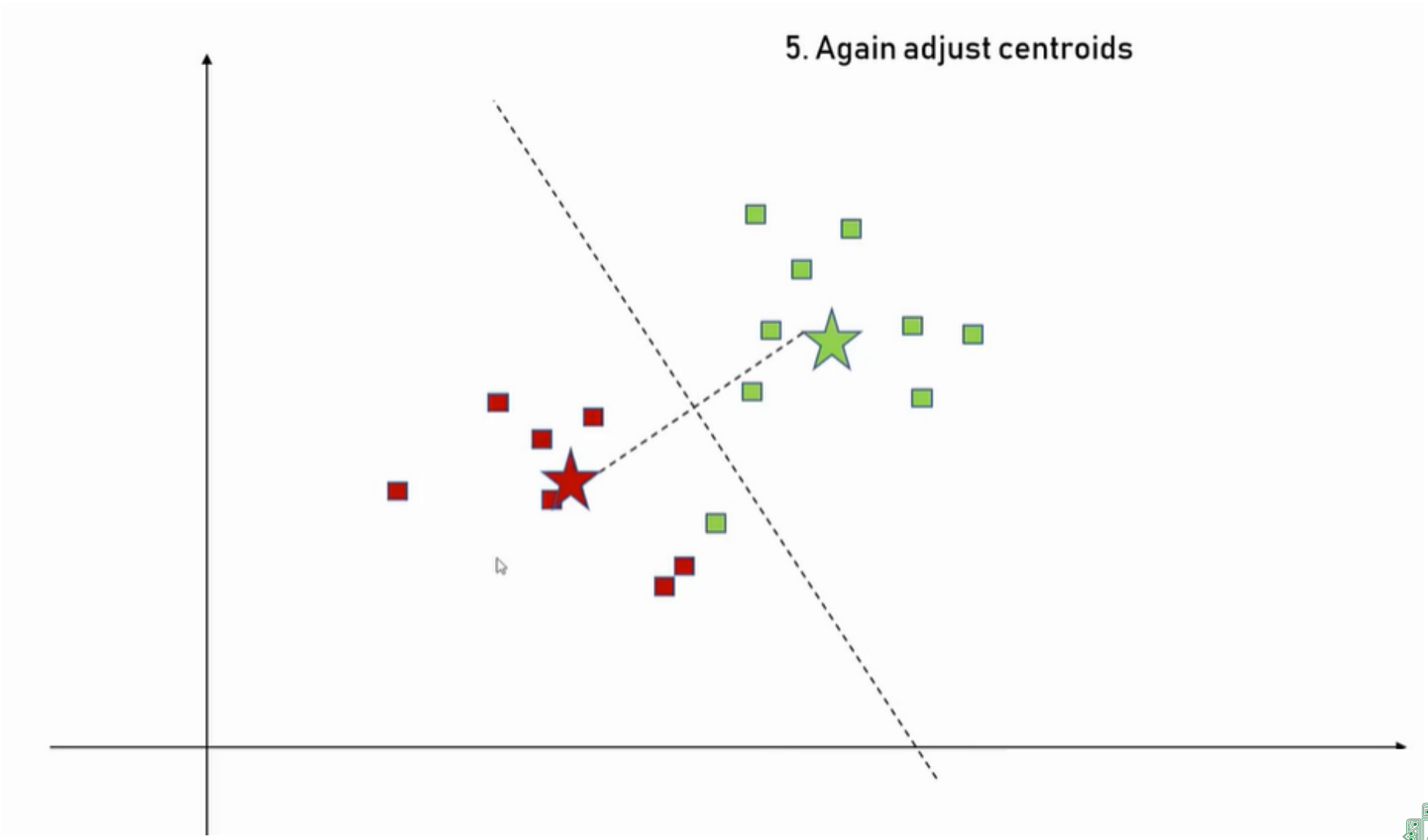
The k-Means Clustering



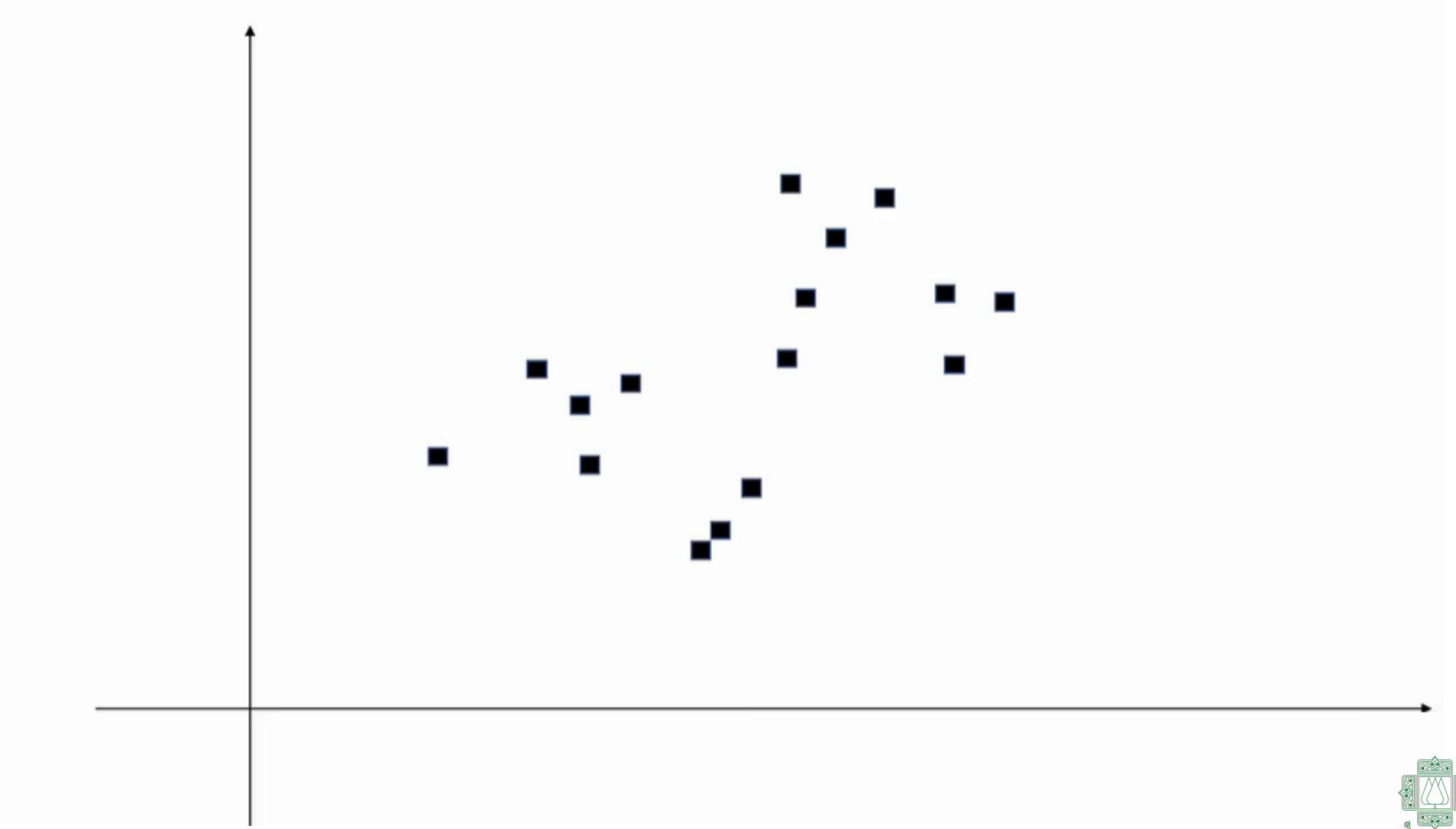
The k-Means Clustering



The k-Means Clustering



The k-Means Clustering



Example: K-Means Clustering

- given: $\{2, 4, 10, 12, 3, 20, 30, 11, 25\}$, $k = 2$
- randomly assign means: $m_1 = 3$, $m_2 = 7$

Table: Add caption

t_i	2	4	10	12	3	20	30	11	25
$t_i - m_1$	-1	1	7	9	0	17	27	8	22
$t_i - m_2$	-5	-3	3	5	-4	13	23	4	18

- $K_1 = \{2, 3, 4\}, K_2 = \{10, 12, 20, 30, 11, 25\} \Rightarrow m_1 = 3, m_2 = 18$
- $K_1 = \{2, 3, 4, 10\}, K_2 = \{12, 20, 30, 11, 25\} \Rightarrow m_1 = 4.75, m_2 = 19.6$
- $K_1 = \{2, 3, 4, 10, 11, 12\}, K_2 = \{20, 30, 25\} \Rightarrow m_1 = 7, m_2 = 25$
- stop as the clusters with these means are the same

Algorithm of k-means clustering

ALGORITHM 5.6

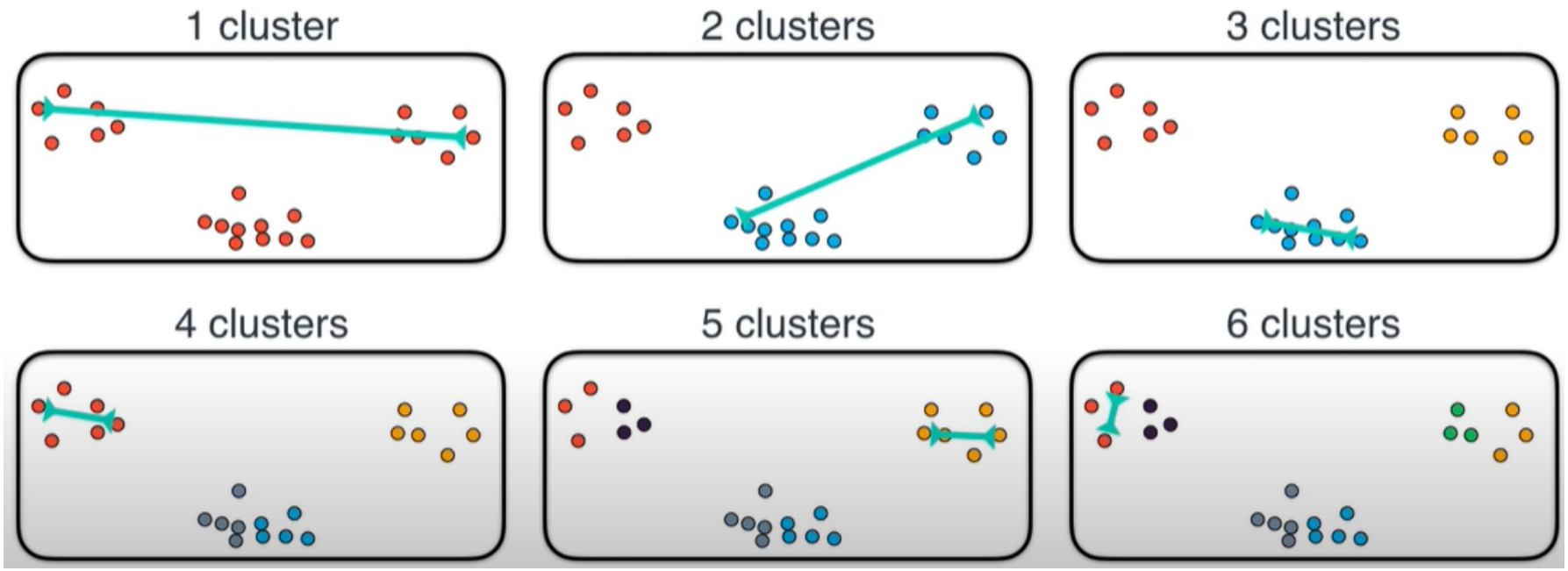
```
Input:
   D = {t1, t2, ..., tn}    //Set of elements
   k      //Number of desired clusters
Output:
   K      //Set of clusters
K-means algorithm:
   assign initial values for means m1, m2, ..., mk;
   repeat
      assign each item ti to the cluster which has the closest mean;
      calculate new mean for each cluster;
   until convergence criteria is met;
```

The K-means algorithm is illustrated in Example 5.4.

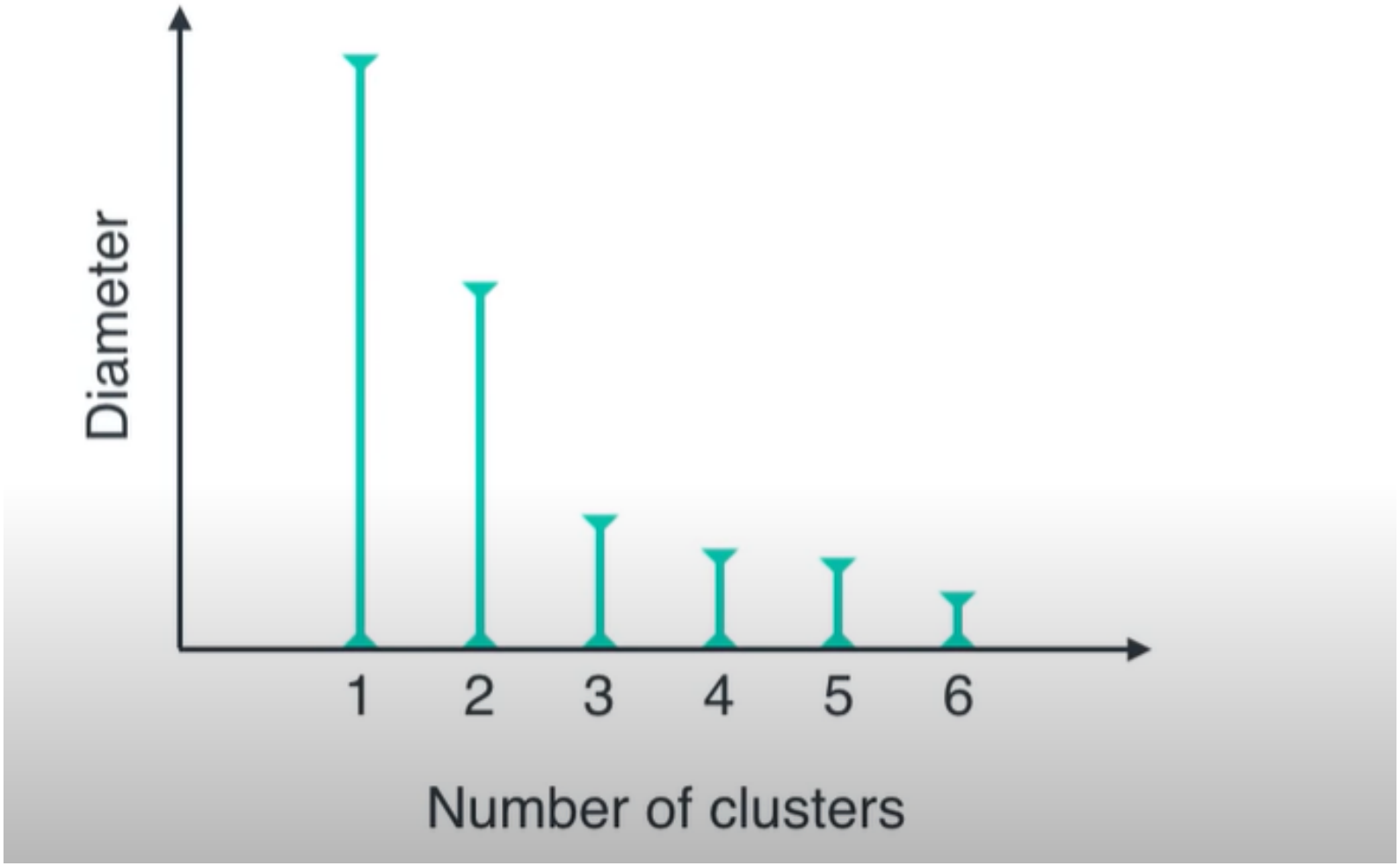
Determining Optimal Clusters

1. Elbow method
2. Silhouette method
3. Gap statistic

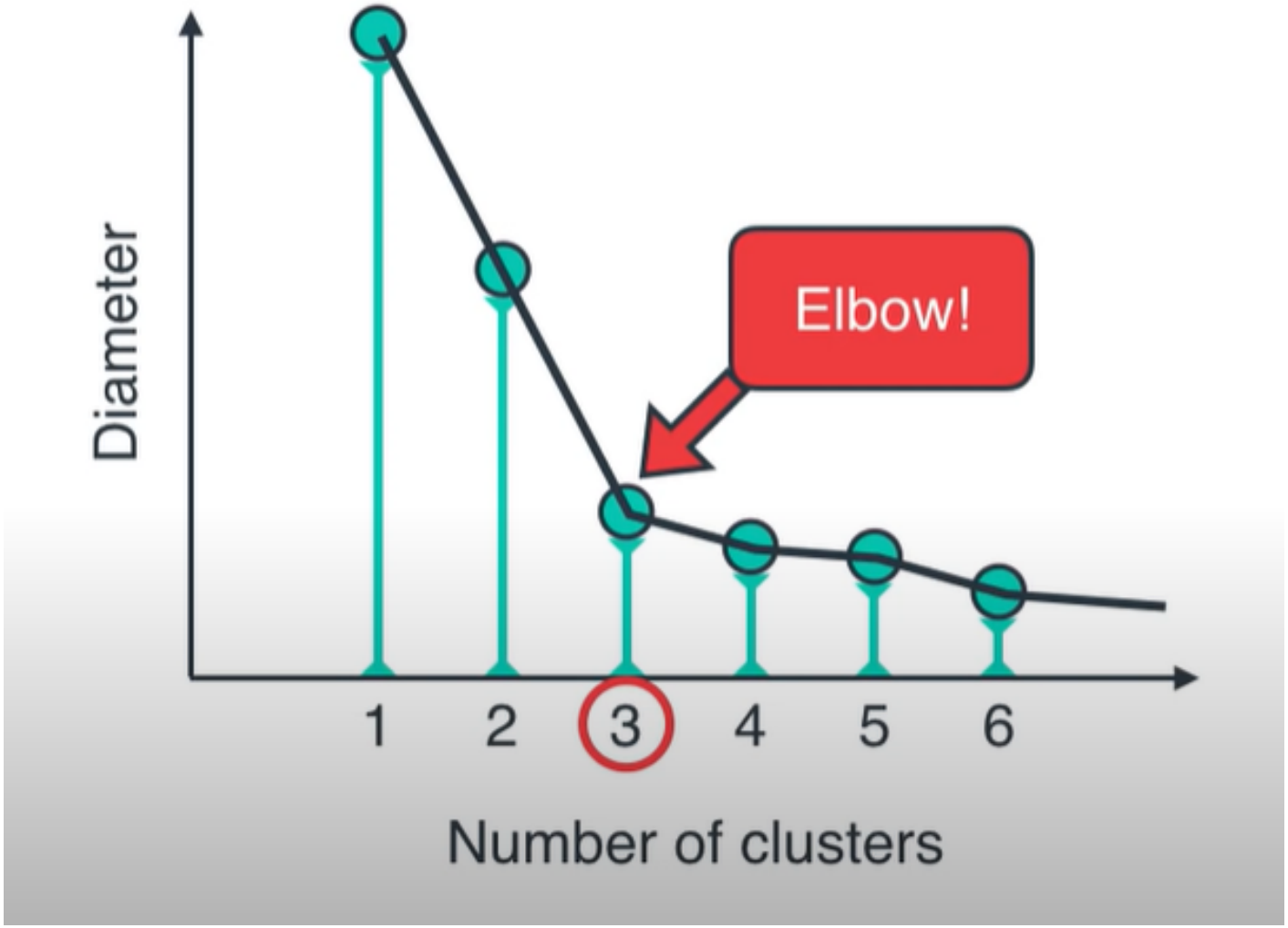
Elbow Method for determining the value of k



Elbow Method for determining the value of k



Elbow Method for determining the value of k



Elbow Method: Optimal Clusters Size

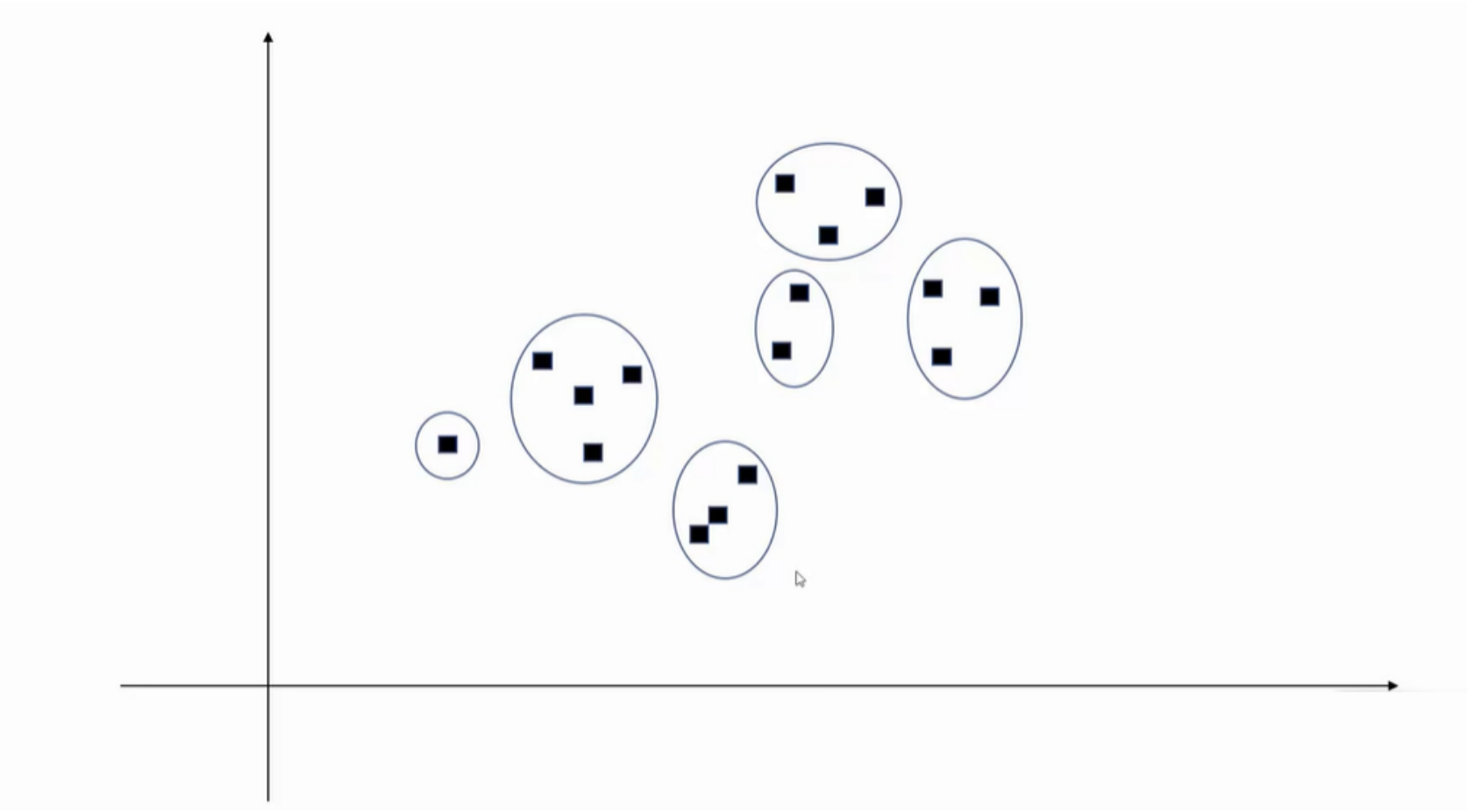
alternatively, we can use the following algorithm to define the optimal clusters:

1. compute clustering algorithm (e.g., k-means clustering) for different values of k
   — for instance, by varying k from 1 to 10 clusters
2. for each k, calculate the total within-cluster sum of square (WSS) or sum of square error (SSE) which is defined

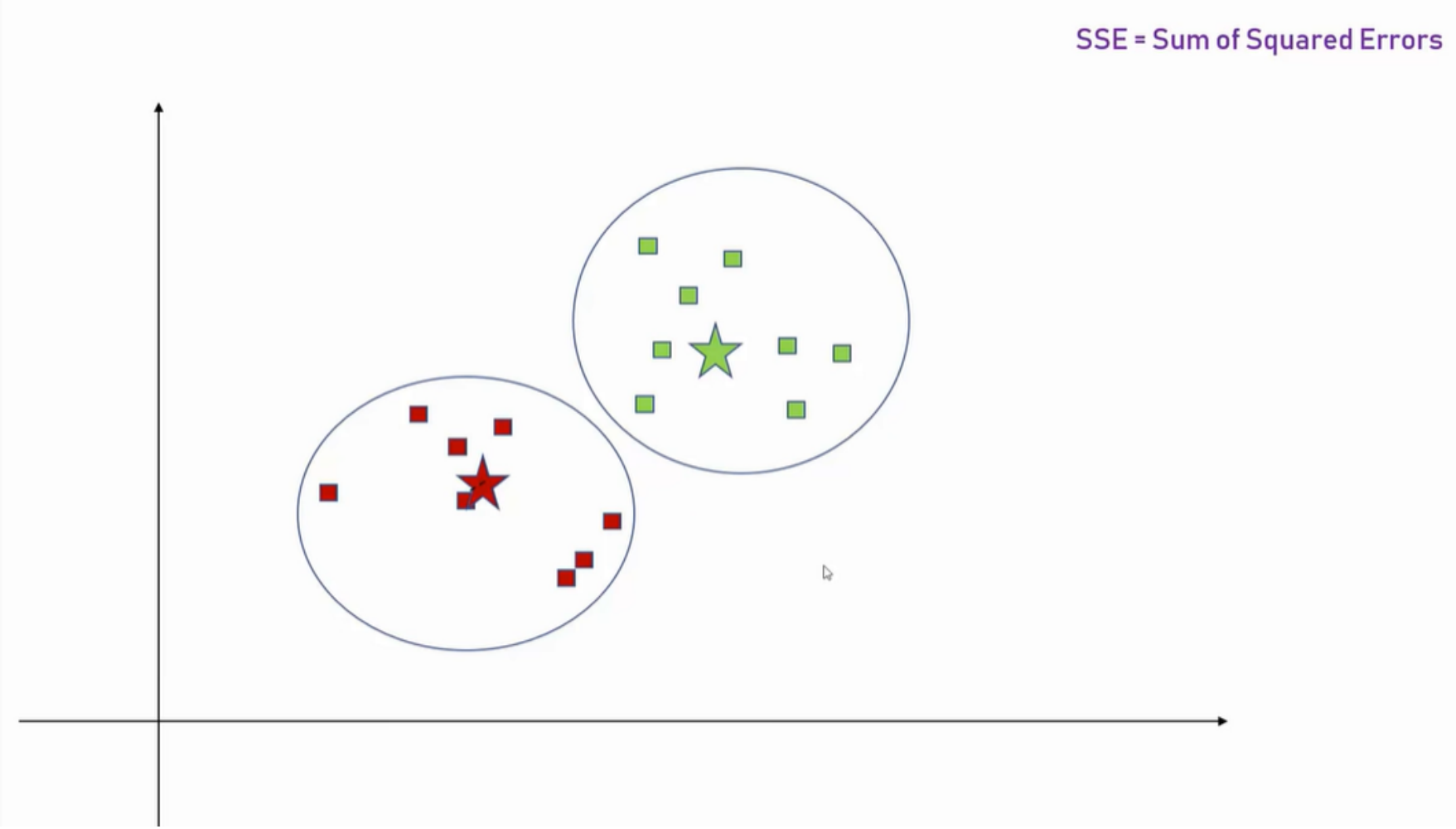
$$\sum_{k=1}^{K} W(C_k)$$

   where C_k is the kth cluster and $W(C_k)$ is the within-cluster variation (also called sum of square error within the cluster
3. plot the curve of WSS according to the number of clusters k
4. the location of a bend (knee/elbow) in the plot is generally considered as an indicator of the appropriate number of clusters

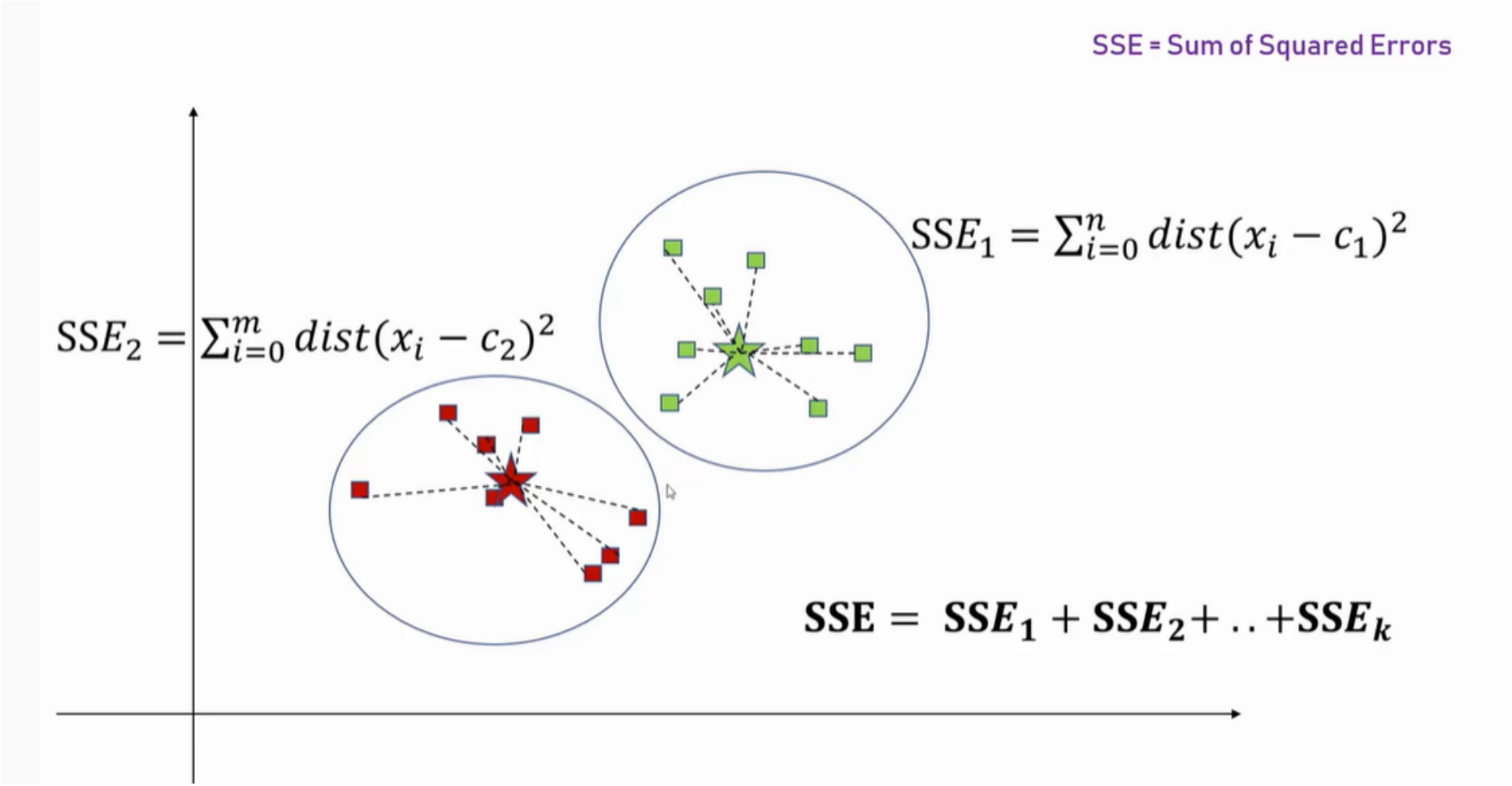
What is the value of k?



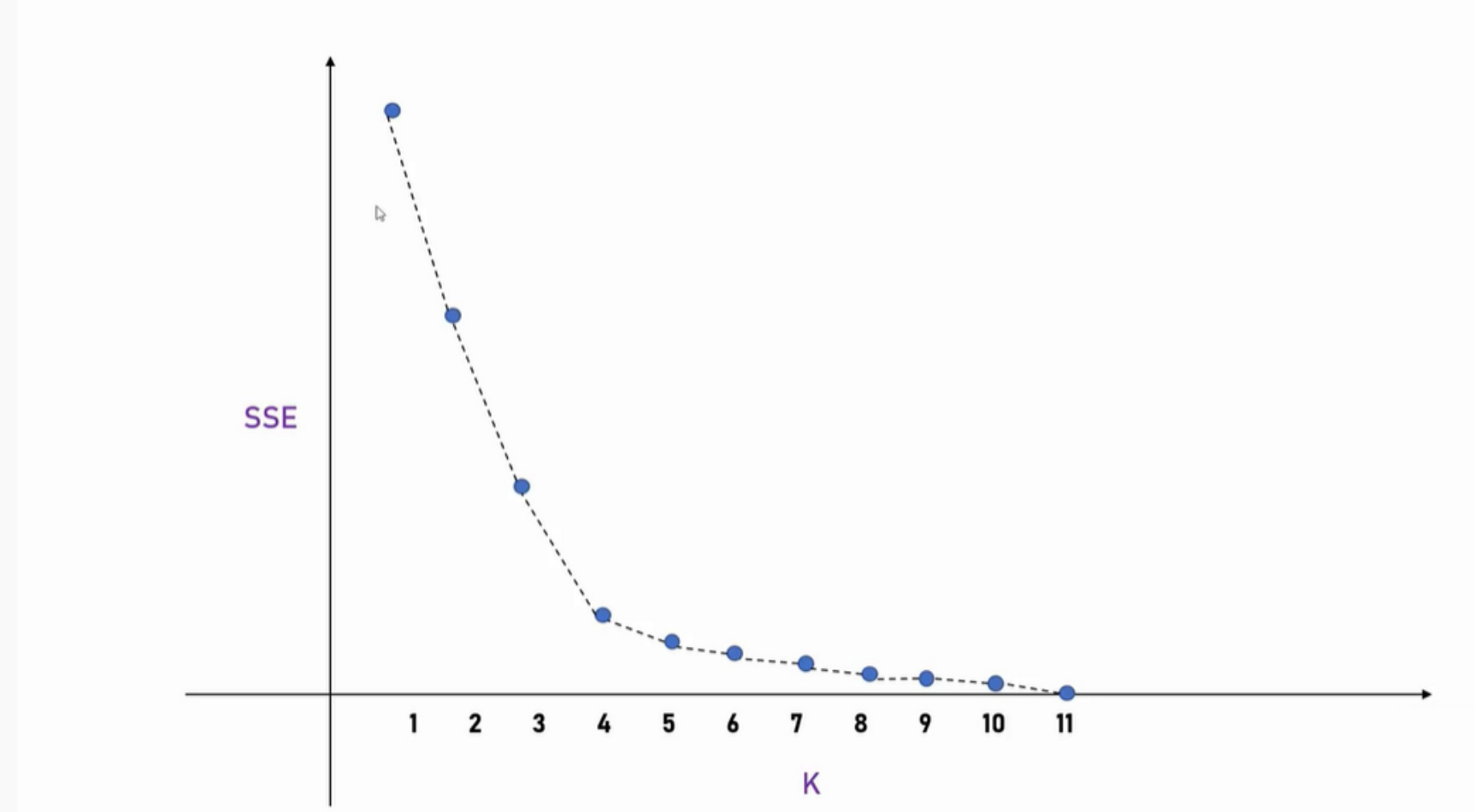
What is the value of k?



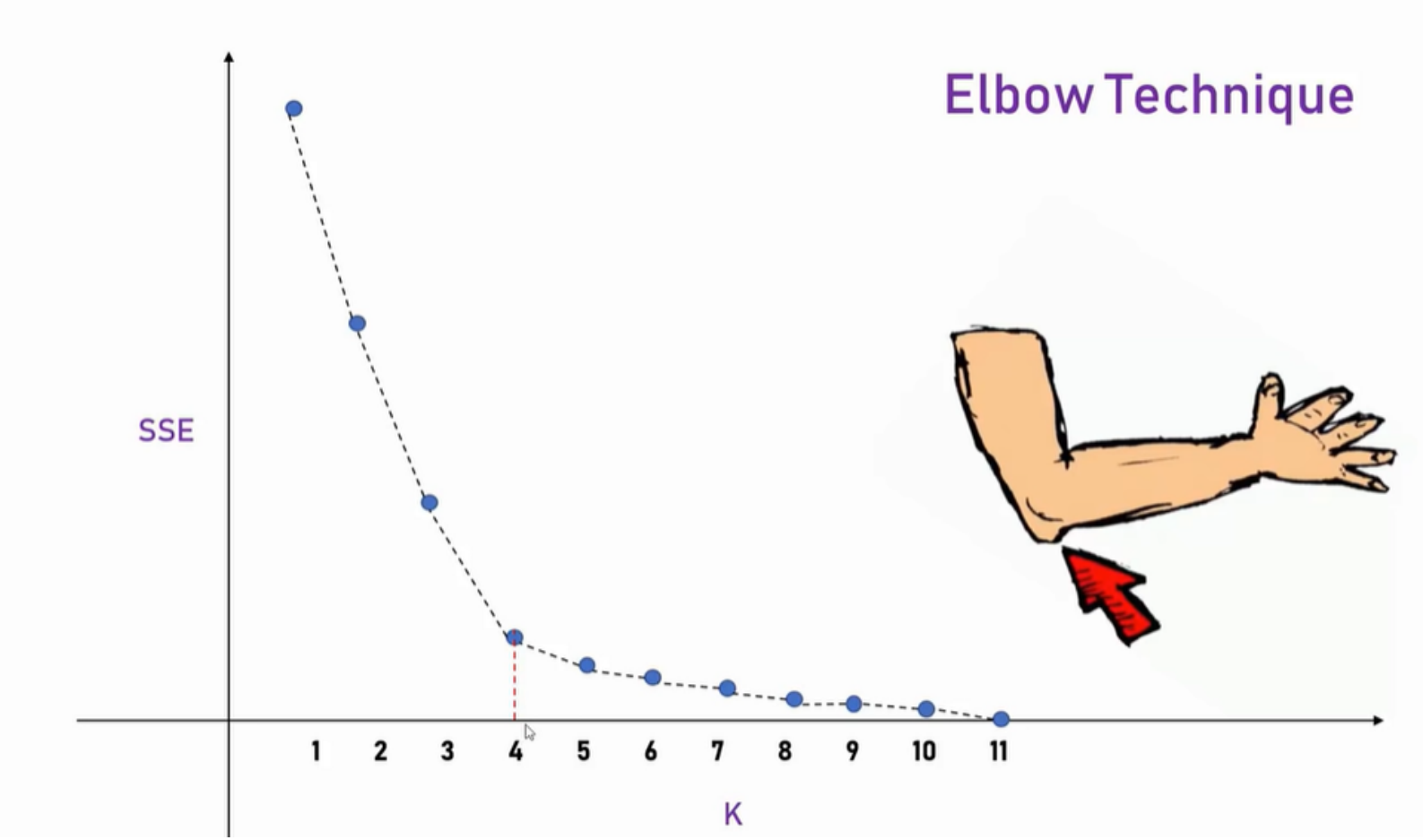
What is the value of k?



What is the value of k?



What is the value of k?



Python code: K-means Clustering

```
#K-means cluster
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline

from sklearn.datasets import load_iris
import pandas as pd

data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df)
y_predicted
```

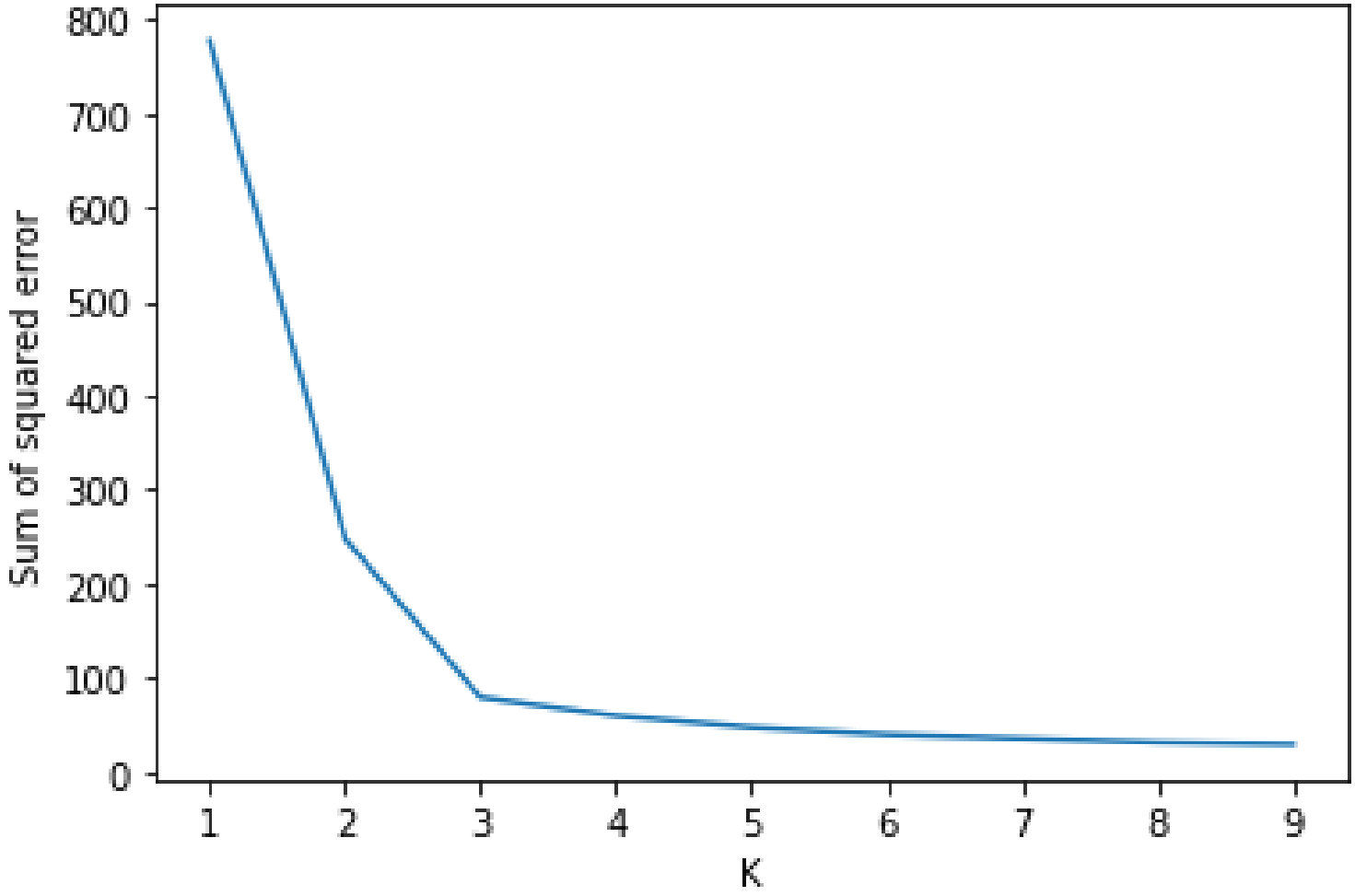
Python code: *k*-means Clustering

```
df['cluster']=y_predicted
df.head()
# See center of the cluster
km.cluster_centers_

# Draw a scatter plot
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1['sepal length (cm)'],df1['sepal width
    (cm)'],color='green')
plt.scatter(df2['sepal length (cm)'],df2['sepal width
    (cm)'],color='red')
plt.scatter(df3['sepal length (cm)'],df3['sepal width
    (cm)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='pur
plt.xlabel('Sepal length (cm)')
plt.ylabel('Sepal width (cm)')
plt.legend()
```

Python code: k-means Clustering

```
# Elbow plot
sse = []
k_rng = range(1,10)
for k in k_rng:
km = KMeans(n_clusters=k)
km.fit(df)
sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

What is the value of k for iris data?



Python code: Hierarchical Clustering

```
from sklearn.datasets import load_iris
import pandas as pd

data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()
import seaborn as sns
sns.pairplot(df)
from sklearn.cluster import AgglomerativeClustering
groups = AgglomerativeClustering(n_clusters=3,
    affinity='euclidean', linkage='ward')
groups .fit_predict(df)
plt.scatter(df['sepal length (cm)'] ,df['sepal width (cm)'],
    c= groups.labels_, cmap='cool')
```

Python code: Silhouetter Clustering

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples,
    silhouette_score

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np

# Generating the sample data from make_blobs
# This particular setting has one distinct cluster and 3
    clusters placed close
# together.
X, y = make_blobs(n_samples=500,
n_features=2,
centers=4,
cluster_std=1,
center_box=(-10.0, 10.0),
```

```
from sklearn.cluster import KMeans

wcss=[]
for i in range(1,11):
kmeans=KMeans(n_clusters=i, init='k-means++',random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
clusterer = KMeans(n_clusters=4, random_state=10)
cluster_labels = clusterer.fit_predict(X)
print(cluster_labels)
```

```
for n_clusters in range_n_clusters:
# Create a subplot with 1 row and 2 columns
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_size_inches(18, 7)

# The 1st subplot is the silhouette plot
# The silhouette coefficient can range from -1, 1 but in
    this example all
# lie within [-0.1, 1]
ax1.set_xlim([-0.1, 1])
# The (n_clusters+1)*10 is for inserting blank space between
    silhouette
# plots of individual clusters, to demarcate them clearly.
ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

# Initialize the clusterer with n_clusters value and a
    random generator
# seed of 10 for reproducibility.
clusterer = KMeans(n_clusters=n_clusters, random_state=10)
cluster_labels = clusterer.fit_predict(X)

# The silhouette_score gives the average value for all the
    samples.
# This gives a perspective into the density and separation
    of the formed
# clusters
silhouette_avg = silhouette_score(X, cluster_labels)
print("For n_clusters =", n_clusters,
"The average silhouette_score is :", silhouette_avg)
```

```
# Compute the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(X,
    cluster_labels)

y_lower = 10
for i in range(n_clusters):
# Aggregate the silhouette scores for samples belonging to
# cluster i, and sort them
ith_cluster_silhouette_values = \
sample_silhouette_values[cluster_labels == i]

ith_cluster_silhouette_values.sort()

size_cluster_i = ith_cluster_silhouette_values.shape[0]
y_upper = y_lower + size_cluster_i

color = cm.nipy_spectral(float(i) / n_clusters)
ax1.fill_betweenx(np.arange(y_lower, y_upper),
0, ith_cluster_silhouette_values,
facecolor=color, edgecolor=color, alpha=0.7)

# Label the silhouette plots with their cluster numbers at
    the middle
ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

# Compute the new y_lower for next plot
y_lower = y_upper + 10 # 10 for the 0 samples

ax1.set_title("The silhouette plot for the various
    clusters.")
ax1.set_xlabel("The silhouette coefficient values")
ax1.set_ylabel("Cluster label")

# The vertical line for average silhouette score of all the
    values
ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([]) # Clear the yaxis labels / ticks
ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
```

```
# 2nd Plot showing the actual clusters formed
colors = cm.nipy_spectral(cluster_labels.astype(float) /
    n_clusters)
ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0,
    alpha=0.7,
c=colors, edgecolor='k')

# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):
ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
s=50, edgecolor='k')

ax2.set_title("The visualization of the clustered data.")
ax2.set_xlabel("Feature space for the 1st feature")
ax2.set_ylabel("Feature space for the 2nd feature")

plt.suptitle(("Silhouette analysis for KMeans clustering on
    sample data "
"with n_clusters = %d" % n_clusters),
fontsize=14, fontweight='bold')

plt.show()
```

R Code: k-Means Cluster

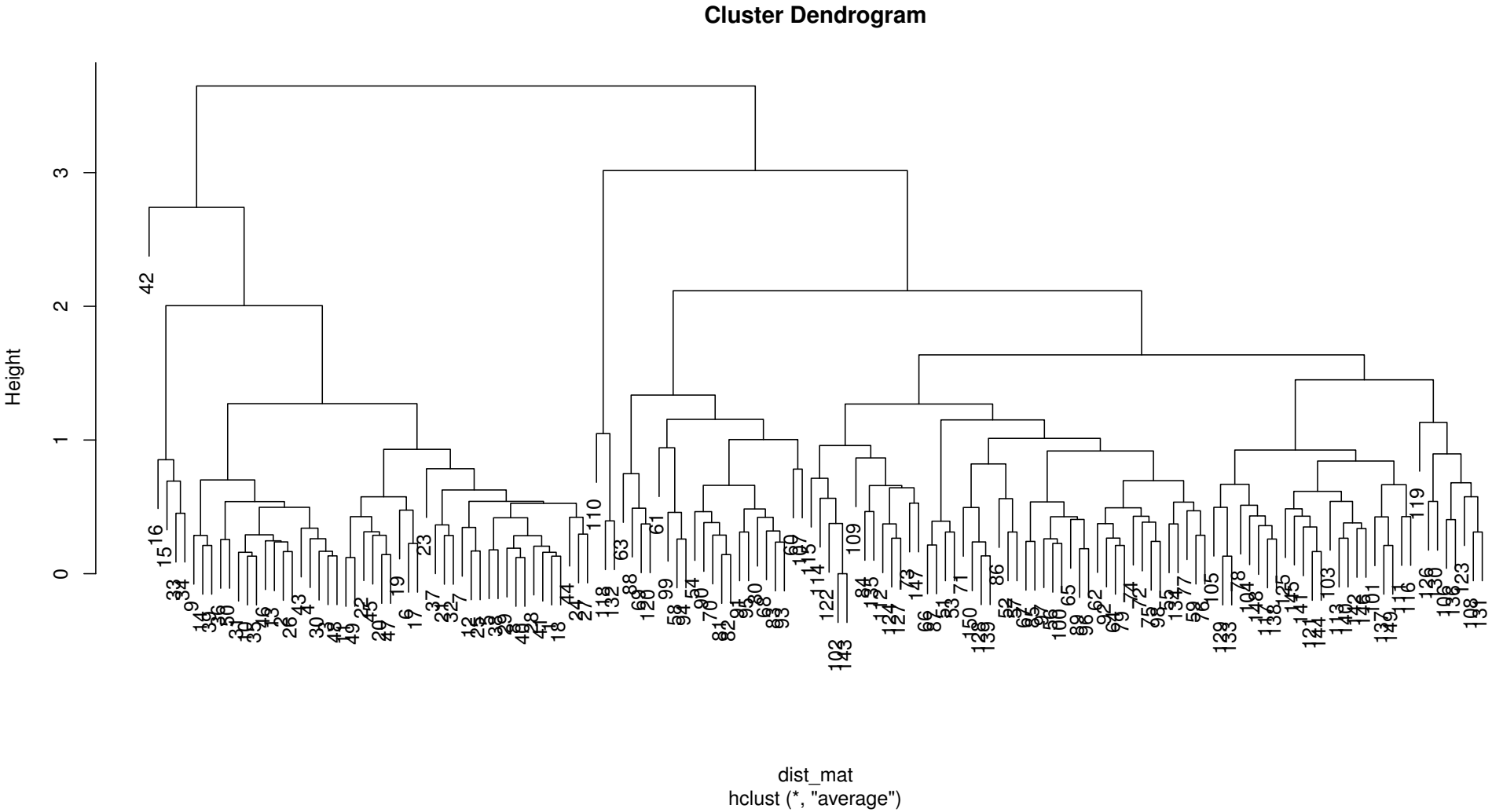
```
# Example 1
x<-c(2,4,10,12,3,20,30,11,25)
example1<-kmeans(x, 2, nstart = 25)
print(example1)
# Cluster size
example1$size
# Cluster means
example1$centers
# Cluster members
cbind(x, example1$cluster)
```

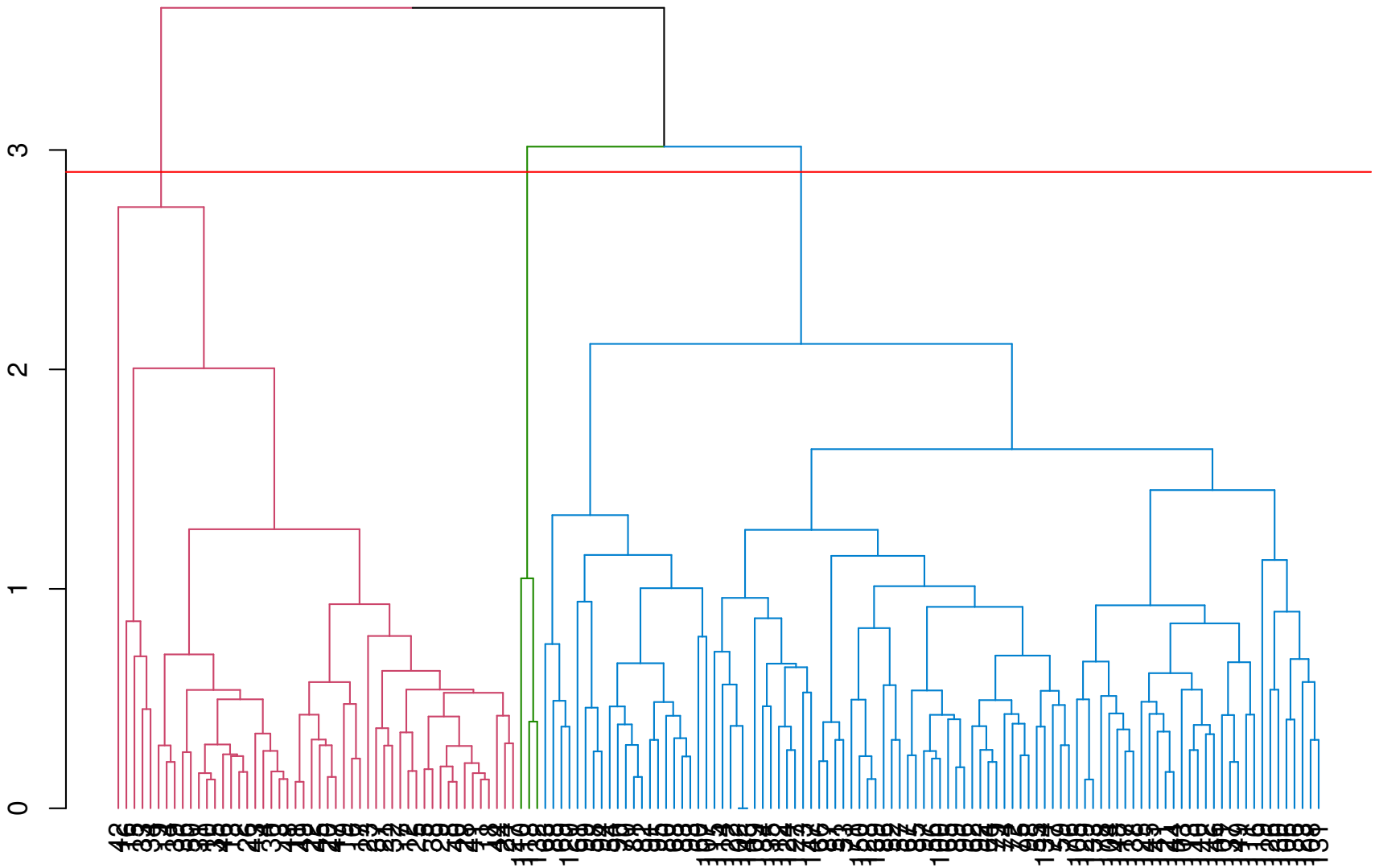
R Code: k-Means Cluster

```
### Example 2: iris dataset
# Remove species column (5) and scale the data
iris.scaled <- scale(iris[, -5])
# Elbow method for kmeans
fviz_nbclust(iris.scaled, kmeans, method = "wss") +
geom_vline(xintercept = 3, linetype = 2)
example2<-kmeans(iris.scaled,centers=3, nstart = 25)
print(example2)
# Cluster size
example2$size
# Cluster means
example2$centers
# Cluster members
cbind(iris.scaled, example2$cluster)
fviz_cluster(example2, data = iris.scaled)
```

R Code: Hierarchical Clustering

```
## iris dataset
dist_mat <- dist(iris.scaled, method = 'euclidean')
print(dist_mat)
hclust_avg <- hclust(dist_mat, method = 'average')
# hclust_single<- hclust(dist_mat, method = 'single')
# hclust_complete<- hclust(dist_mat, method = 'complete')
plot(hclust_avg)
cut_avg <- cutree(hclust_avg, k = 3)
plot(hclust_avg)
abline(h = 2.9, col = 'red')

suppressPackageStartupMessages(library(dendextend))
avg_dend_obj <- as.dendrogram(hclust_avg)
avg_col_dend <- color_branches(avg_dend_obj, h = 3)
plot(avg_col_dend)
abline(h = 2.9, col = 'red')
```


Cluster Dendrogram
Height
0
1
2
3
dist_mat
hclust (*, "average")


0
1
2
3

Chapter 13: Parallel Ensemble Learning — Bagging Methods

13 Chapter 13: Parallel Ensemble Learning — Bagging Methods

13.1 Problem & Motivation

13.2 Ensemble Learning

13.3 Bootstrapping

13.4 Bagging

13.5 Random Forest

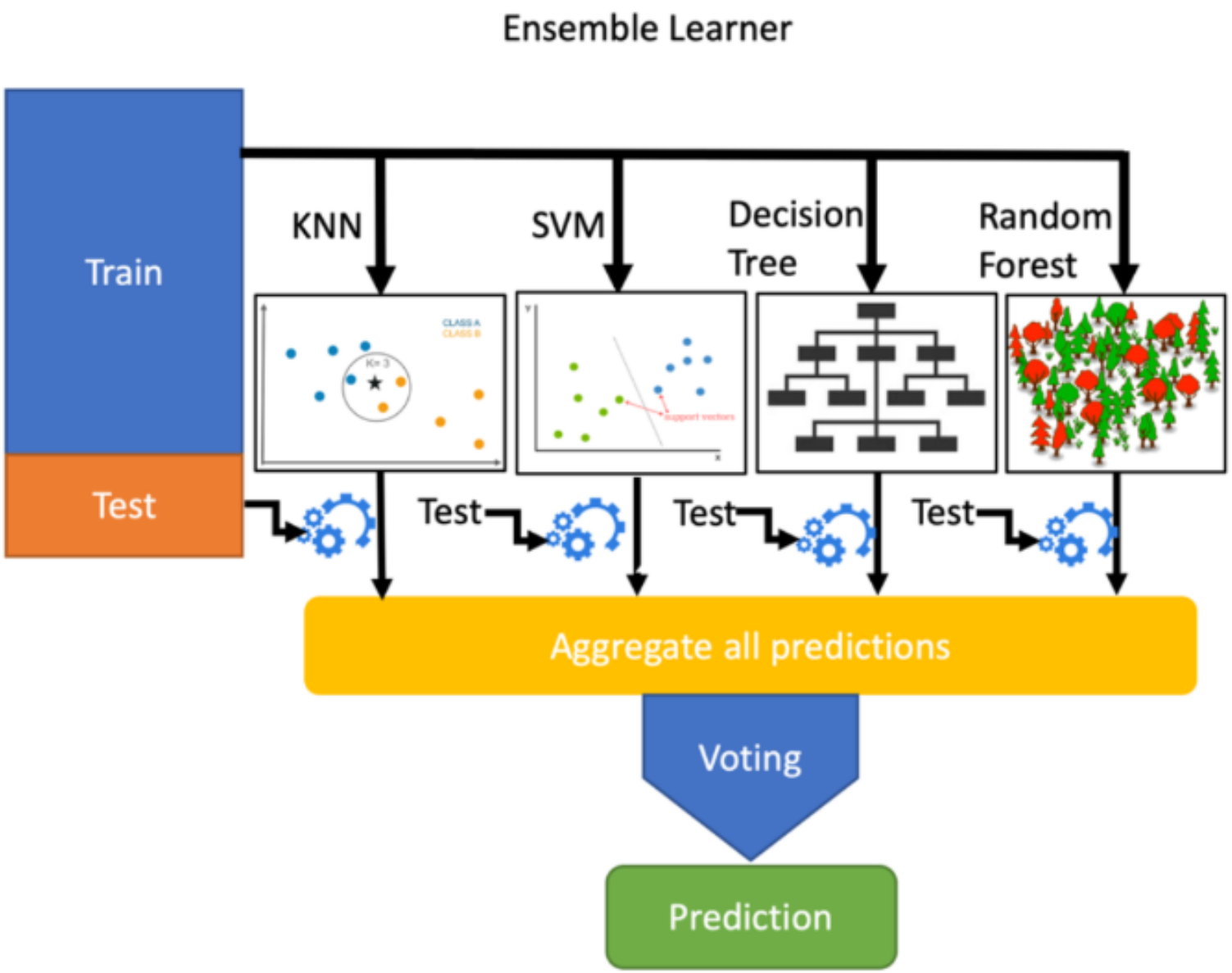
13.6 Python Code: Bagging methods

13.7 R Code: Boasting Methods

Ensemble learning

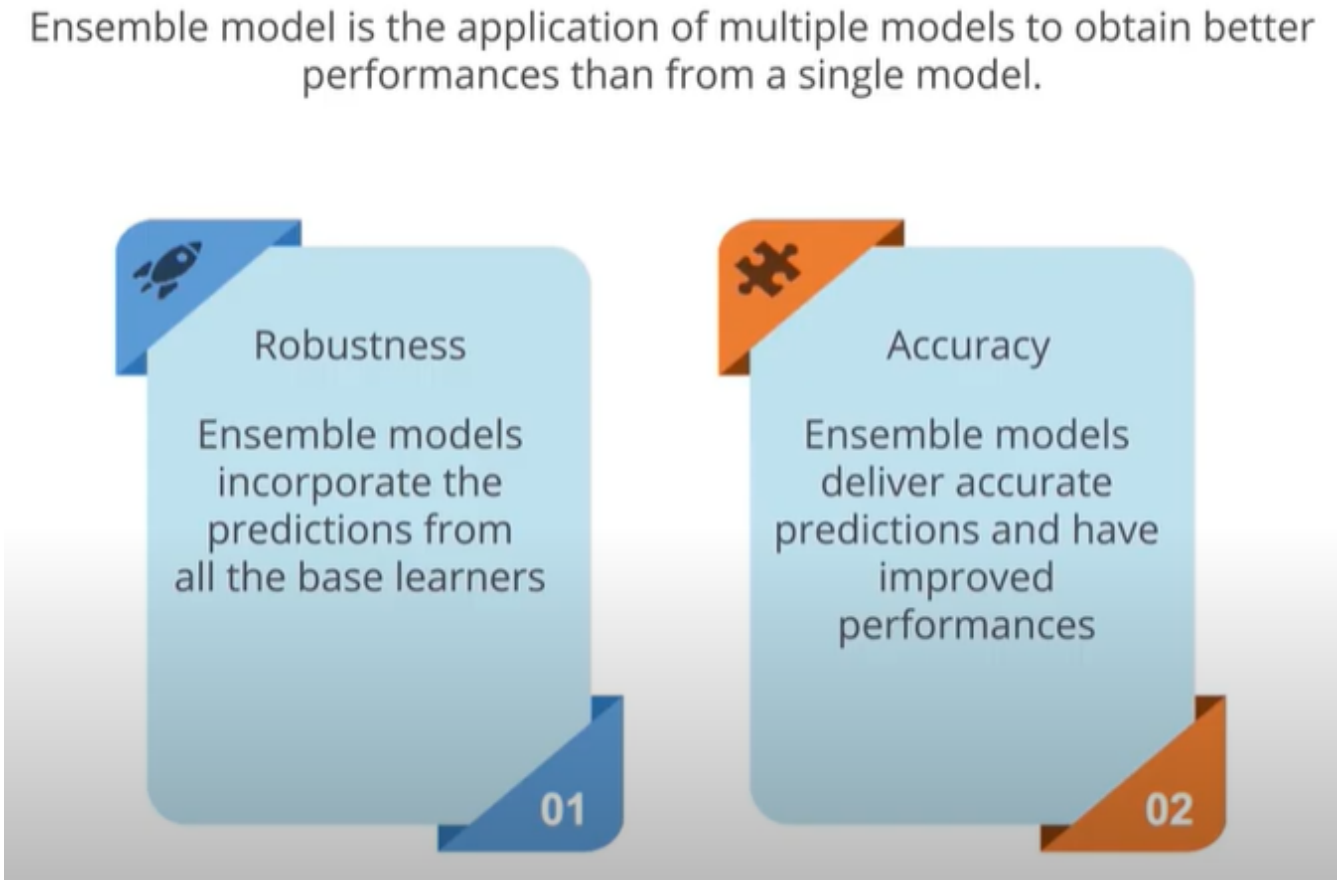
- a process in which decisions from multiple machine learning models are combined to reduce errors and improve prediction when compared to a single ML model
- then the maximum voting technique is used on aggregated decisions (or predictions in machine learning) to deduce the final prediction.
- the **goal of ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator

Ensemble Learner



the performance of ensemble model is based on two factors

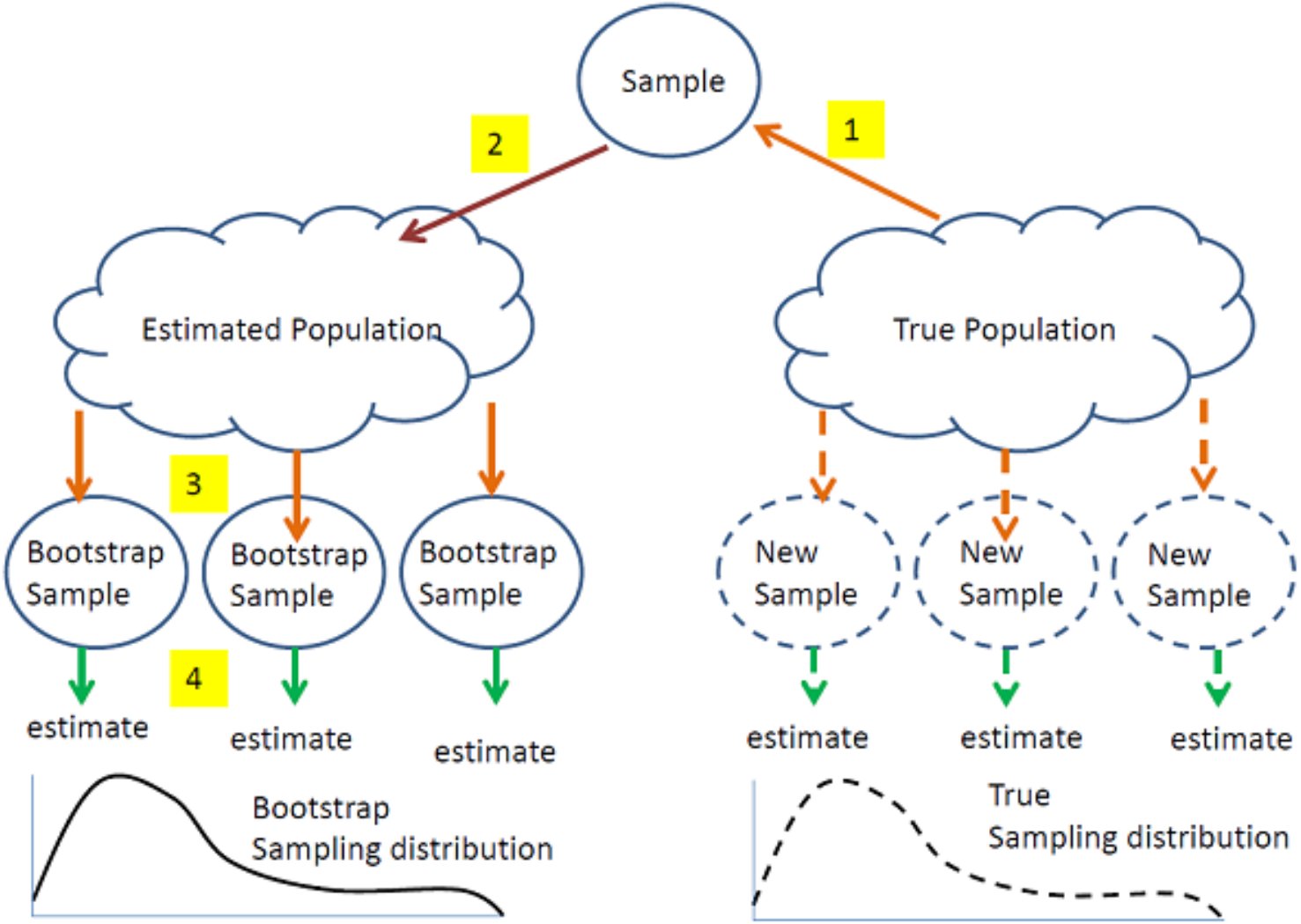
1. robustness
2. accuracy



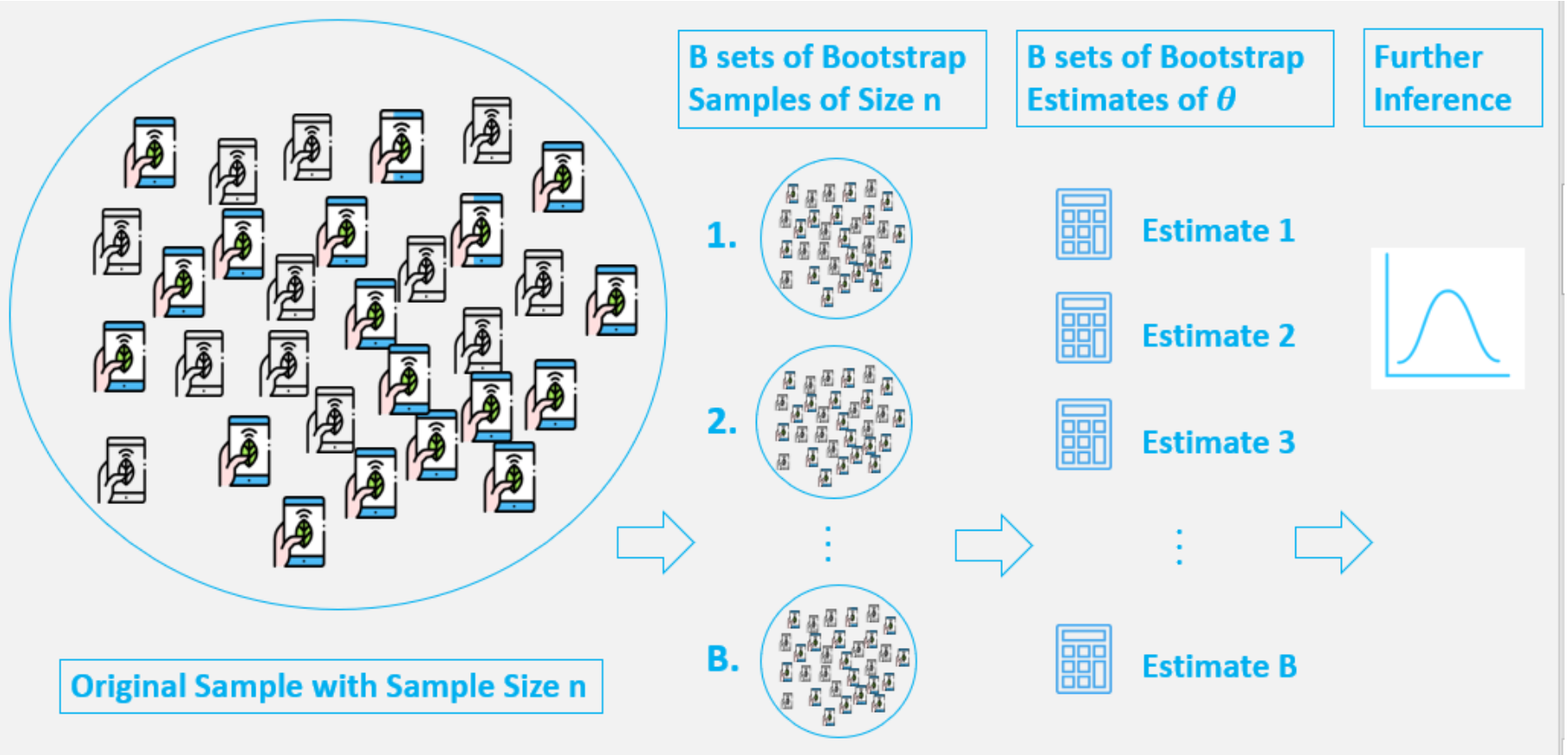
types of ensemble learning (the heart of machine learning):

1. averaging methods (parallel ensemble) ⇒ for reducing variance
   - bagging
   - Random Forest algorithm
2. boosting methods (sequential ensemble) ⇒ for reducing bias
   - AdaBoost
   - Boosted Regression Tree
   - Gradient Boosting
   - XGBoost
   - LPBoost
   - TotalBoost
   - BrownBoost
   - MadaBoost
   - LogitBoost
   - Gentle Boost and others
3. stacking & blending ⇒ for improving predictions

Preliminary Concept: Bootstrap (resampling) Method



Preliminary Concept: Bootstrap (resampling) Method



Preliminary Concept: Bootstrap Alggorithm

1. first, draw B bootstrap samples

$$x_1^*, x_2^*, \ldots, x_B^*$$

with replacement from the observed data $x_1, x_2, \ldots, x_n$ (it is also called nonparametric bootstrap)

2. evaluate the bootstrap replications

$$\widehat{\theta}_1^*, \widehat{\theta}_2^*, \ldots, \widehat{\theta}_B^*$$

3. compute the bootstrap estimate of θ is an average (you can also use median for skewed distribution of bootstrapped statistics) of all $\widehat{\theta}_1^*, \widehat{\theta}_2^*, \ldots, \widehat{\theta}_B^*$, that is

$$\bar{\widehat{\theta}}^* = \frac{1}{B} \sum_{b=1}^{B} \widehat{\theta}_b^*$$

4. bootstrap bias estimator

$$\text{estimated bias}(\widehat{\theta}) = \bar{\widehat{\theta}}^* - \widehat{\theta}$$

5. estimate standard error $F(\widehat{\theta})$ or better approximate standard error $F(\widehat{\theta}^*)$ by the sample deviation of the B replications

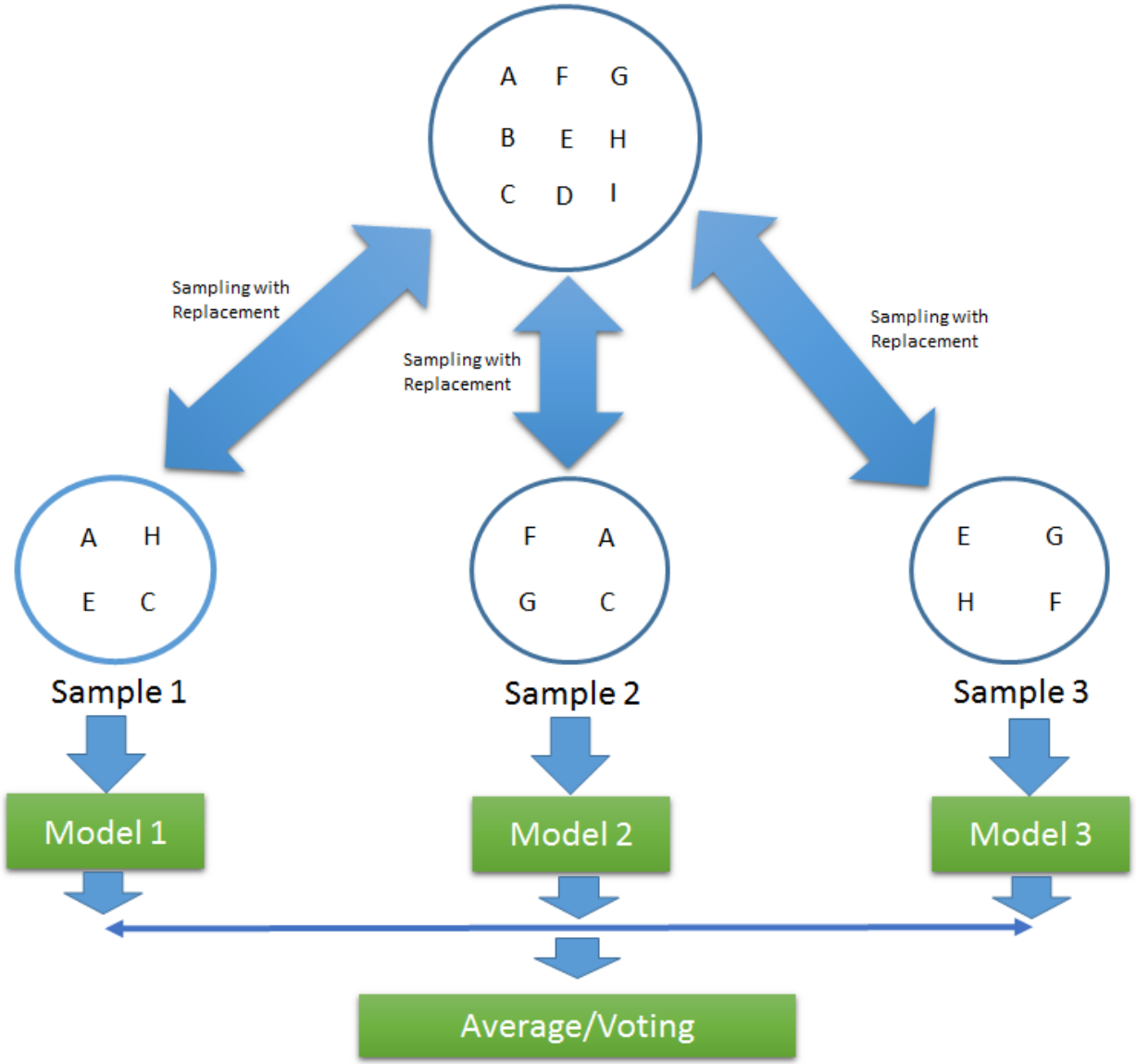
$$\widehat{se}(\widehat{\theta}) = \left\{ \frac{1}{B+1} \sum_{b=1}^{B} \left(\widehat{\theta}_b^* - \bar{\widehat{\theta}}^* \right)^2 \right\}^{\frac{1}{2}}$$

this methods is a nonparametric bootstrap methods

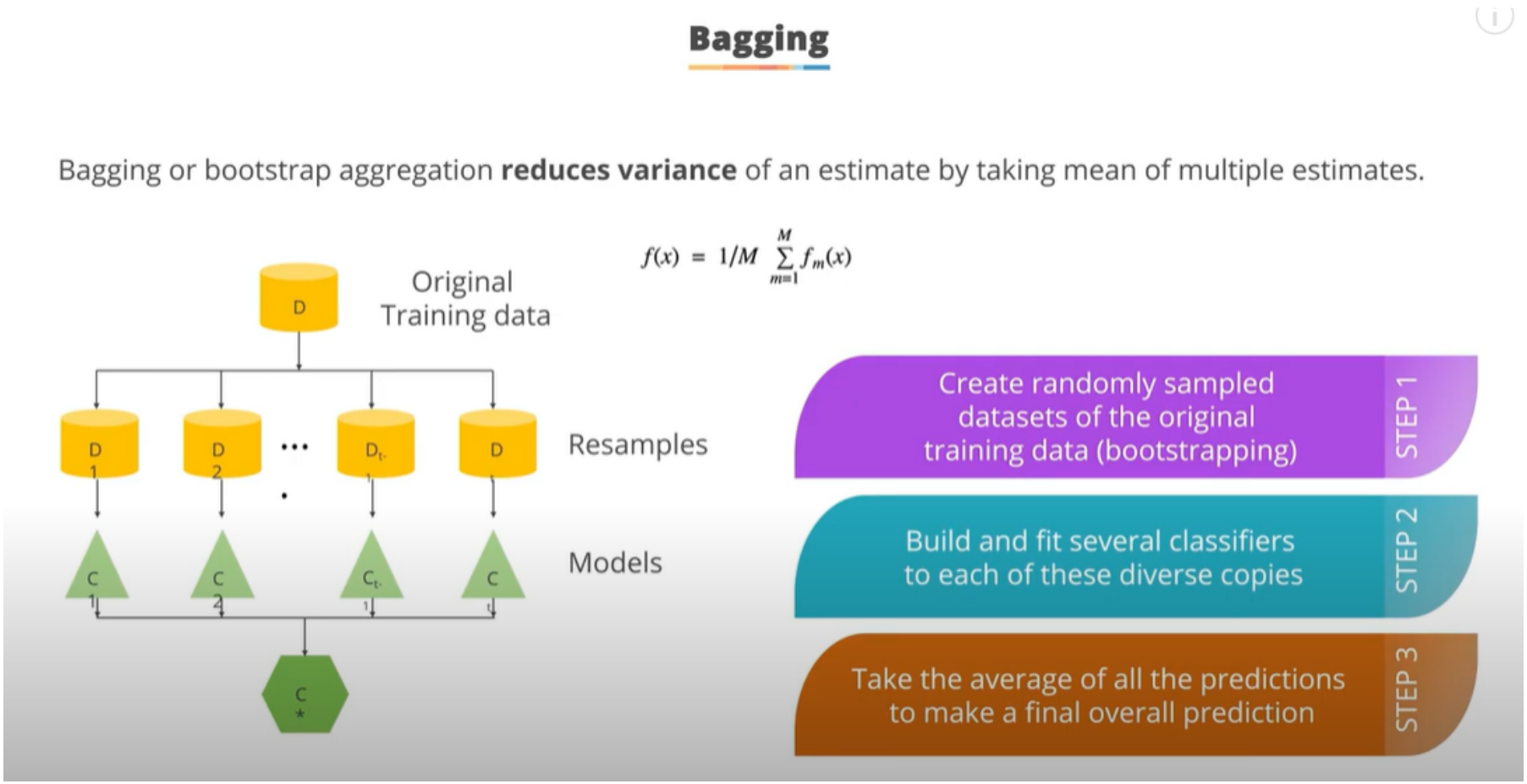
Bagging

- bagging is an acronym for **b**ootstrap **agg**regat**ing**
- bagging (**b**ootstrap **agg**regat**ing**) algorithm for creating multiple different models from a single training dataset
- it is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model
- bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees
- it helps to avoid overfitting

Bagging


A F G
B E H
C D I
Sampling with Replacement
Sampling with Replacement
Sampling with Replacement
A H
E C
F A
G C
E G
H F
Sample 1
Sample 2
Sample 3
Model 1
Model 2
Model 3
Average/Voting

Bagging


Bagging
Bagging or bootstrap aggregation **reduces variance** of an estimate by taking mean of multiple estimates.
$f(x) = 1/M \sum_{m=1}^{M} f_m(x)$
Original Training data
D
Resamples
Models
Create randomly sampled datasets of the original training data (bootstrapping)
STEP 1
Build and fit several classifiers to each of these diverse copies
STEP 2
Take the average of all the predictions to make a final overall prediction
STEP 3

Example: let's assume we have a sample dataset of 1000 instances (x) and we are using the CART algorithm. Bagging of the CART algorithm would work as follows:

- create many (e.g. 100) random sub-samples of our dataset with replacement
- train a CART model on each sample
- given a new dataset, calculate an average of the output (for regression) or a maximum number vote(, i.e., mode of the output for classification) from each model

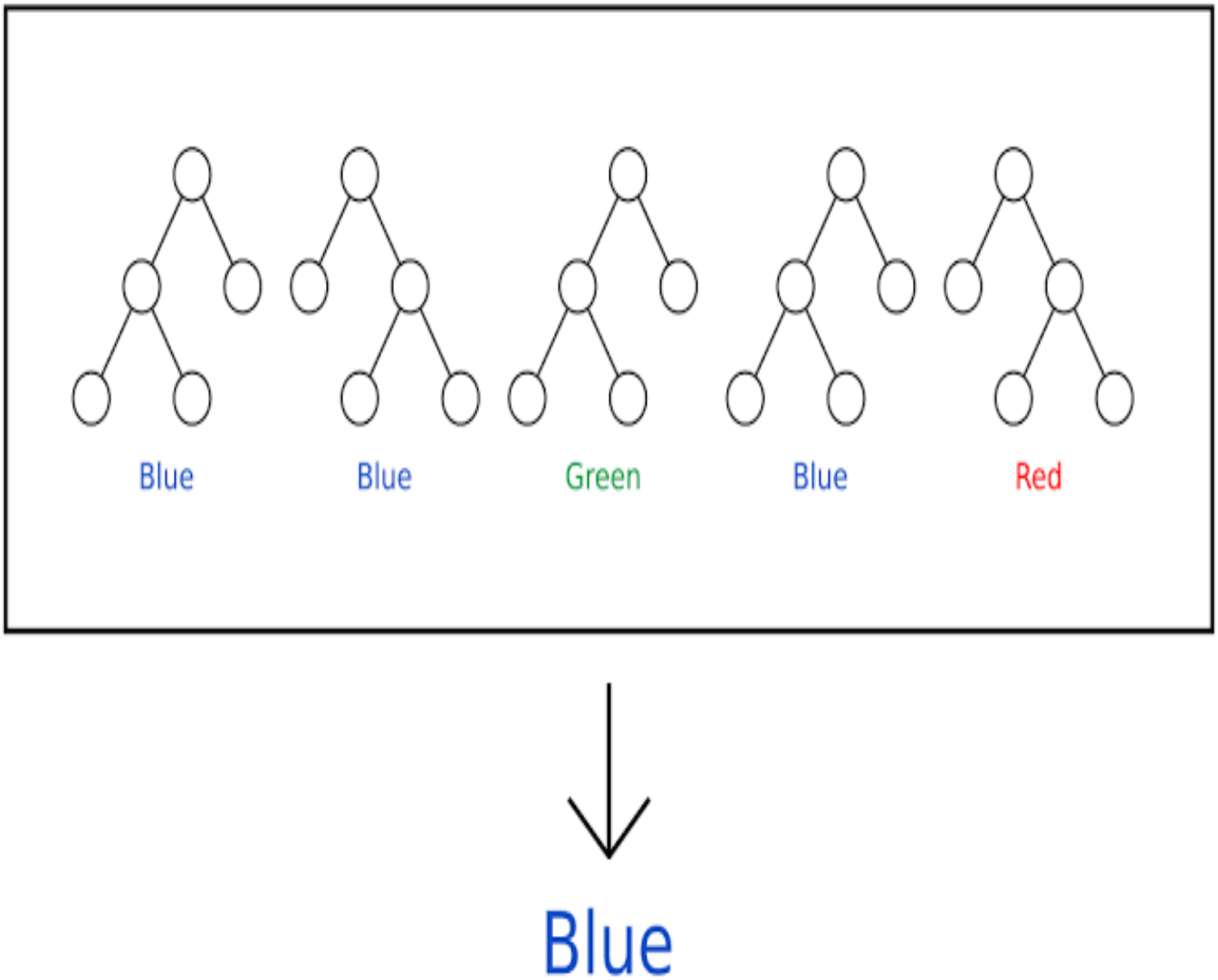
for example, if we had 5 bagged decision trees that made the following class predictions for a in input sample: blue, blue, red, blue and red, we would take the most frequent class and predict blue

- when bagging with decision trees, we are less concerned about individual trees overfitting the training data
- these trees will have both high variance and low bias
- these are important characterize of sub-models when combining predictions using bagging

Random Forest

random forest is an example of ensemble learning, in which we combine multiple machine learning algorithms to obtain better predictive performance

- why the name "Random"?
  - a random sampling of training data set when building trees
  - random subsets of features considered when splitting nodes
- a technique known as bagging is used to create an ensemble of trees where multiple training sets are generated with replacement
- the random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by the maximum of voting
- it reduces the overfitting by averaging or voting the result


Blue
Blue
Green
Blue
Red
Blue

Bagging: for regression

- equal weights are assigned to different models despite some models performing better than others
  — weighted averaging can solve the problem

Python code: Bagging methods

```
# Model fit using Logistic Regression
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import GridSearchCV
# Load the breast cancer dataset
bc = datasets.load_breast_cancer()
X = bc.data
y = bc.target
# Create training and test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.25, random_state=1, stratify=y)
# Pipeline Estimator
pipeline = make_pipeline(StandardScaler(),
LogisticRegression(random_state=1))
# Fit the model
pipeline.fit(X_train, y_train)
# Model scores on test and training data
print('Model test Score: %.3f, ' %pipeline.score(X_test,
    y_test),
'Model training Score: %.3f' %pipeline.score(X_train,
    y_train))
```

Python code: Bagging methods

```
# Model fit using Bagging Classifier
# Pipeline Estimator
pipeline = make_pipeline(StandardScaler(),
LogisticRegression(random_state=1))
# Instantiate the bagging classifier
bgclassifier = BaggingClassifier(base_estimator=pipeline,
    n_estimators=100,
max_features=10,
max_samples=100,
random_state=1, n_jobs=5)
# Fit the bagging classifier
bgclassifier.fit(X_train, y_train)
# Model scores on test and training data
print('Model test Score: %.3f, ' %bgclassifier.score(X_test,
    y_test),
'Model training Score: %.3f' %bgclassifier.score(X_train,
    y_train))
```

R Code: Ensemble Learning

```
# Bagging
library(adabag)
library(caret)
set.seed(1)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
mod = bagging(Species~., data=trainData, coob=TRUE)
print(mod)
summary(mod)
pred=predict(mod, testData)
confusionMatrix(table(testData$Species,pred$class))
```

Chapter 14: Sequential Ensemble Learning — Boosting Methods

14 Chapter 14: Sequential Ensemble Learning — Boosting Methods

14.1 Problem & Motivation

14.2 Boosting Methods

14.3 AdaBoost Algorithm

14.4 Boosted Regression Tree

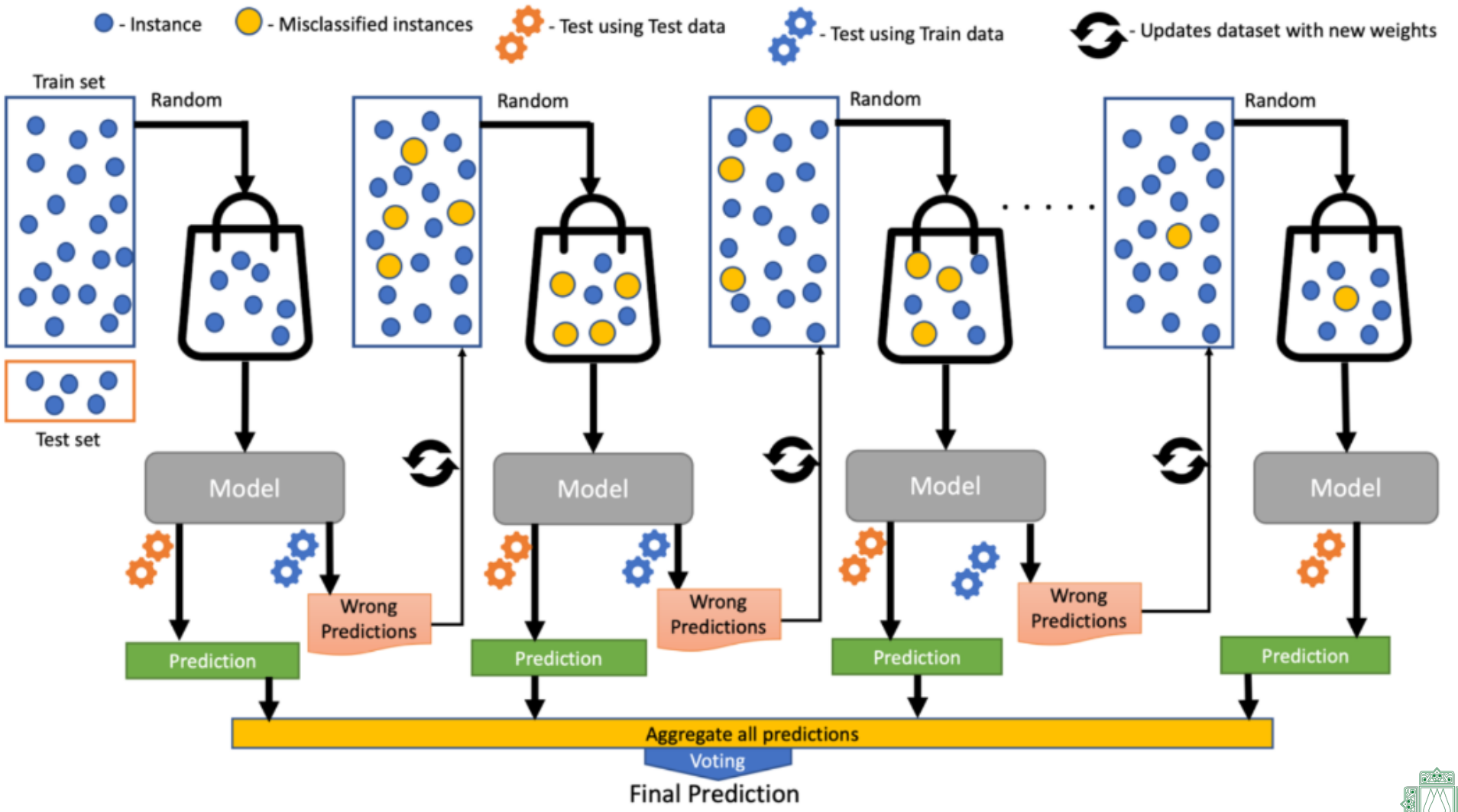
14.5 Python Code: Boasting methods

14.6 R Code: Boasting Methods

Boosting Methods

- boosting generally means increasing performance
- the motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."
- it is a sequential ensemble learning technique to convert a weak classification algorithm or weak learners into strong learners to increase the accuracy of the model
- a weak classifier is one whose error rate is only slightly better than random guessing
- boosting algorithm: improving week learners (models) and creating an aggregated model to improve model accuracy
- the purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data

Internal Working of Boosting Algorithm



What is AdaBoost?

- AdaBoost is short for Adaptive Boosting
- AdaBoost was the first really successful boosting algorithm developed for binary classification
- the most popular boosting algorithm due to Freund and Schapire (1997) called "AdaBoost.M1"
- AdaBoost.M1 algorithm is also known as "Discrete AdaBoost"
- modern boosting methods build on AdaBoost, most notably stochastic gradient boosting machines
- AdaBoost was designed in such a way that at every step the sample distribution was adapted to put more weight on misclassified samples and less weight on correctly classified samples
  —the final prediction is a weighted average of all the weak learners, where more weight is placed on stronger learners

Schematic of AdaBoost

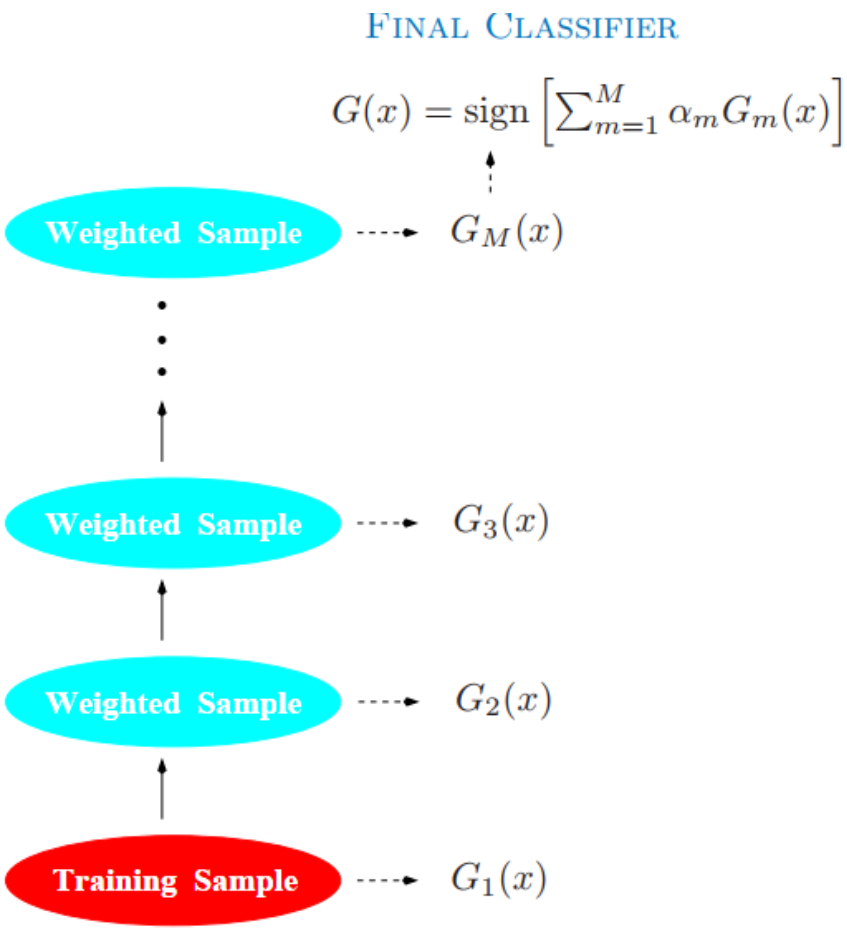


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Algorithm of AdaBoost

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost: Toy Example

Step I

- suppose you select a binary decision tree and you select f_1 based on information gain or gini index
- suppose the decision tree classify 6 correctly and 1 wrongly
- set initial weight $w_i = 1/N \quad ; \quad i = 1, \ldots, 7$

	f_1	f_2	f_3	Output	Sample Weight
1	zz	zz	zz	Yes	1/7
2	zz	zz	zz	No	1/7
3	zz	zz	zz	Yes	1/7 $\Leftrightarrow$ wrongly classify
4	zz	zz	zz	No	1/7
5	zz	zz	zz	Yes	1/7
6	zz	zz	zz	No	1/7
7	zz	zz	zz	Yes	1/7

AdaBoost: Toy Example

Step II: calculate total error

- calculate total error = 1/7 (initial weight of the missclassified tuple)
- compute performance of stump (α_m) by using total error

$$\alpha_1 = \log\left(\frac{1 - err_1}{err_1}\right) = \log((1 - 1/7)/(1/7)) = 0.778151$$

- update the weight:
  - for missclassified observation
    w_3 = inital weight $\times\, e^{\alpha_1} = \frac{1}{7} \times e^{0.778151} = 0.3111$
  - for correctly classified observations
    $w_i = w_i \times e^{-\alpha_1} = \frac{1}{7} \times e^{-0.778151} = 0.0656$

AdaBoost: Toy Example

	f_1	f_2	f_3	Output	Sample Weight	updated weight
1	zz	zz	zz	Yes	1/7	0.0656
2	zz	zz	zz	No	1/7	0.0656
3	xx	xx	xx	Yes	1/7	0.3111
4	zz	zz	zz	No	1/7	0.0656
5	zz	zz	zz	Yes	1/7	0.0656
6	zz	zz	zz	No	1/7	0.0656
7	zz	zz	zz	Yes	1/7	0.0656

AdaBoost: Toy Example

- note that the sum of all updated weight w_i is not equal to 1 whereas the sum of initial weight is 1
- therefore, we need normalized updated weight w_i which can be obtained by following

$$\text{normalized weight} = w_i / \sum_{m=1}^{7} w_i = w_i / 0.7047$$

- based on normalized weight w_i, make interval for each tuple
- now draw a random number, suppose it is 0.5654

AdaBoost: Toy Example

	f_1	f_2	f_3	Output	Normalized w_i	Interval
1	zz	zz	zz	Yes	0.093089	(to 0.0931)
2	zz	zz	zz	No	0.093089	(0.0931–0.1862)
3	zz	zz	zz	Yes	0.441464	(0.1862–0.6276)
4	zz	zz	zz	No	0.093089	(0.6276–0.7207)
5	zz	zz	zz	Yes	0.093089	(0.7207–0.8138)
6	zz	zz	zz	No	0.093089	(0.8138–0.9069)
7	zz	zz	zz	Yes	0.093089	(0.9069–1)

- since random number 0.5654 is belongs to the interval (0.1962–0.6276), then select corresponding taupe (in this case it is id 3)
- similarly, select weighted sample
- based on the weighted sample, apply binary decision tree algorithm to identify missclassified observations

AdaBoost: Toy Example

Step III

- repeat **Step II** and
- continue iteration up to minimize total error compare to the initial stage

Algorithm of AdaBoost

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Boosted Regression Tree: Toy Example

id	Name	Salary$y$	base model $\widehat{y}_0$	r_{i0}	r_{i1} (say)	$\widehat{y}_1$
1	zzz	50	75	-25	-23	52
2	zzz	70	75	-5	-3	72
3	zzz	85	75	10	3	78
4	zzz	95	75	20	23	98

let a square function $\sum(y_i - \widehat{y})^2$ and we want to minimize this with respect to $\widehat{y}$

1. so we get, $\widehat{y} = \bar{y}$ and it is a base model, i.g., $\widehat{y}_0 = \bar{y}$
2. residual is $r_0 = y - \widehat{y}$
3. make a decision tree based on residual and input feature $S_0 = \{r_{i0}, x_i\}$ and then predict r_{i0} and obtain residual r_{i1}
4. compute $\widehat{y}_1 = \widehat{y}_0 + r_1 \Rightarrow$ predicted value is too far from the actual value y?
5. need again decision tree on $S_1 = \{r_{i1}, x_i\}$

Boosted Regression Tree

- boosted trees are grown sequentially; each tree is grown using information from previously grown trees
- the basic algorithm for boosted regression trees can be generalized to the following where x represents features and y represents response
  (i). fit a decision tree to the data y and consider initial predicted data $F_1(x)$
  (ii). then fit the next decision tree to the residuals of the previous: $h_1(x) = y - F_1(x)$
  (iii). add this new tree to our algorithm: $F_2(x) = F_1(x) + h_1(x)$
  (iv). fit the next sequential decision tree to the residuals of $F_2 : h_2(x) = y - F_2(x)$
  (v). and this new tree to the algorithm: $F_3(x) = F_2(x) + f_1(x)$
  (vi). continue this process until some mechanism (i.e. cross validation) tells us to stop

the basic algorithm for boosted regression trees can be generalized to the following where the final model is simply a stagewise additive model of b individual regression trees:

$$F(x) = \sum_{b=1}^{B} F_b(x)$$

Python code:

```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
# Classification Report
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Load data
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split dataset into training and test dataset
from sklearn.model_selection import train_test_split
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3) # 70% training and 30% test
```

```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50, learning_rate=1) #
    It uses DecisionTreeClassifier as default weak learner
    for training purpos
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# predict for the test dataset and generate classification
    report
predictionsModel = model.predict(X_test)
print('Classification
    Report:\n',classification_report(y_test,
    predictionsModel))
print('Confusion Matrix:\n',confusion_matrix(y_test,
    predictionsModel))
print('Accuracy Score:',accuracy_score(y_test,
    predictionsModel))
```

Use Base Learners: SVM

```
# Base Learners SVM
from sklearn.ensemble import AdaBoostClassifier
# Import Support Vector Classifier
from sklearn.svm import SVC
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
svc=SVC(probability=True, kernel='linear')
# Create adaboost classifer object
abc =AdaBoostClassifier(n_estimators=50,
    base_estimator=svc,learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

R Code: Ensemble Learning –Boosting

```
# Adaboost algorithm
library(adabag)
library(caret)
data("iris")
pairs(Species~., data=iris, col=iris$Species)
## Make training and test data sets.
set.seed(1)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
head(trainData)
# Classification with the Adabag Boosting
model = boosting(Species~., data=trainData, boos=TRUE,
    mfinal=50)
print(names(model))
print(model$trees[1])
pred=predict(model, testData)
pred
confusionMatrix(pred$confusion)
# print the probability of each class in test data.
result = data.frame(testData$Species, pred$prob, pred$class)
print(result)
```

R Code: Ensemble Learning

```
# Gradient Boosting
require(gbm)
## Make training and test data sets.
set.seed(1)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
model=gbm(Species~.,data = trainData,distribution =
    "gaussian",n.trees = 10000,
shrinkage = 0.01, interaction.depth = 4)
model

summary(model)
?gbm
```