ECEN 377: Engineering Applications of AI

Dr. Mahmoud Nabil
mnmahmoud@ncat.edu

North Carolina A & T State University

September 12, 2024

Outline

Welcome and Instructor Introduction

- **Instructor:** Mahmoud N. Mahmoud
- **Background:** Brief overview of your experience in AI and engineering
- **Contact:** mnmahmoud@ncat.edu, Office Hours: TR from 1:00pm to 4:00pm

Course Objectives

- Understand fundamental concepts of machine learning and AI
- Apply machine learning techniques to solve engineering problems
- Gain hands-on experience with Python programming for ML
- Explore advanced topics and current trends in AI

Course Structure and Schedule

Weekly Topics Overview

- Introduction to ML, AI, and Basics of Python
- Minimization Maximization
- KNN and Decision Trees
- Linear Regression
- Logistic Regression and Optimization
- Polynomial Regression
- Decision Trees & Random Forest
- Classification Model Performance
- Ensemble Learning
- Support Vector Machines
- Neural Networks
- ML in Practice

Lecture and Lab Format

- 1.5 hours lecture + 1.5 hours lab each week

Key Dates

- Two Exams: Week 5 & 10
- Project Deadlines: Week 14
- Final Exam: TBD

Assessment Breakdown

- Assignments/Quizes: 20%
- Final Project: 15%
- Exams: 65 (2+Final)%

Outline

What is Artificial Intelligence and Machine Learning?

What is Artificial Intelligence and Machine Learning?

" AI: The set of all tasks in which a computer can make decisions or imitate human behaviors "

What is Artificial Intelligence and Machine Learning?

" AI: The set of all tasks in which a computer can make decisions or imitate human behaviors "

Humans make decisions based on:

- Logic And Reasoning
- Experience

What is Artificial Intelligence and Machine Learning?

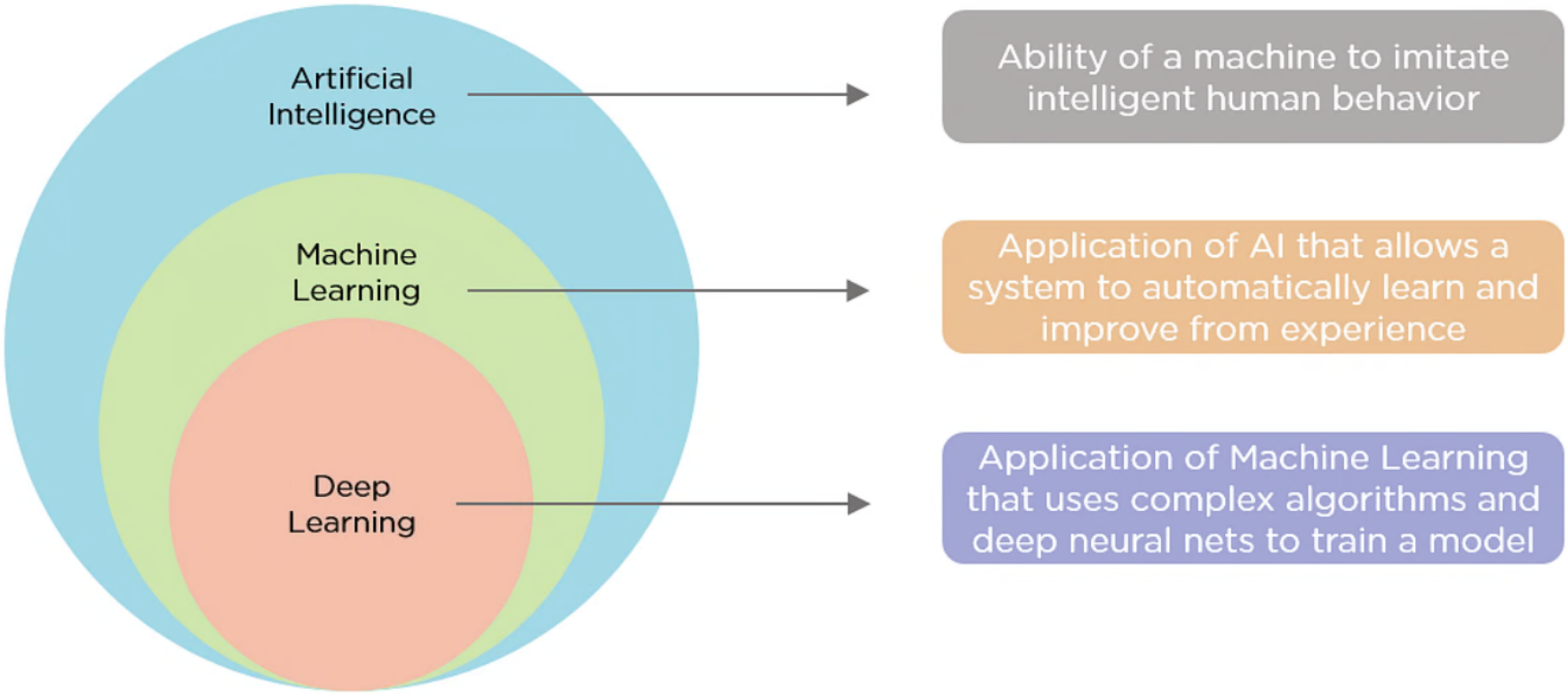
" AI: The set of all tasks in which a computer can make decisions or imitate human behaviors "

Humans make decisions based on:

- Logic And Reasoning
- Experience

" AI Make decisions based on input data "

Overview of AI and Machine Learning

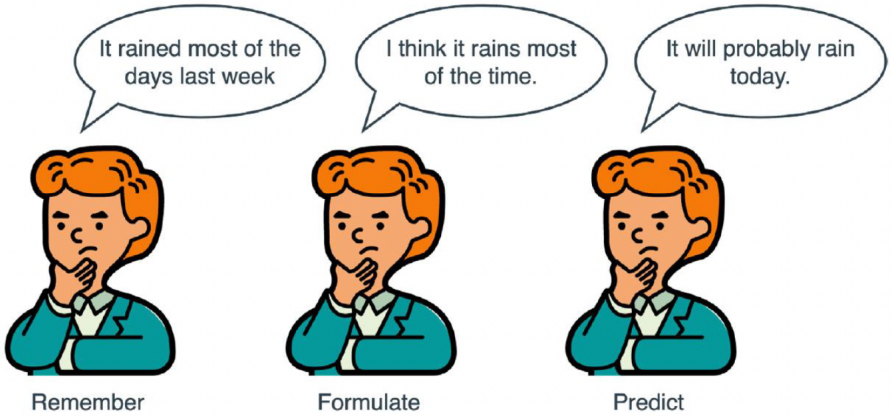


How Do Humans Learn?

How Do Humans Learn?

3 steps for humans to learn:

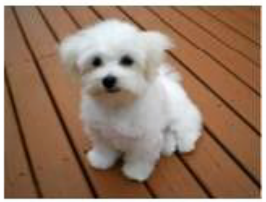
1. We remember past situations that were similar.
2. We formulate a general rule (Model).
3. We use this rule to predict what may happen in the future.



Outline

What is Model?

"A set of rules that represent our data and can be used to make predictions"

$f($  $) \longrightarrow dog$

$f($  $) \longrightarrow cat$

Example: Spam or Ham?!

- Bob likes to send us a lot of emails.
- In particular, a lot of his emails are spam.
- It is Saturday, and we just got a notification of an email from Bob.
- Can we guess if it is spam or ham without looking at the email?

Example: Spam or Ham?!

Model 1

- **Remember**
  - 4 of the last 10 emails from Bob were spam.
- **Formulate (Model)**
  - 40% of Bob's emails are spam. Should it be true?!
- **Predict**
  - This new email is Ham.

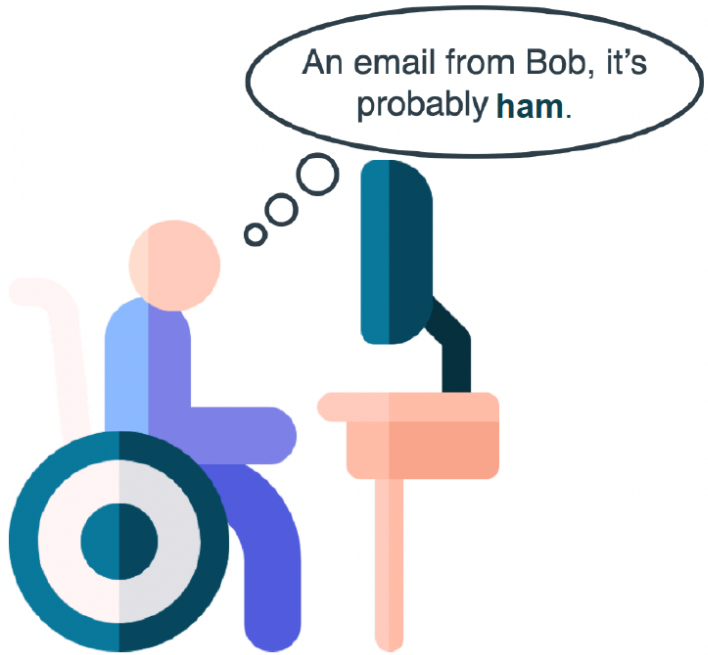
Example: Spam or Ham?!

Model 2

- **Remember**

Mo	Tu	Sa	Su	Su	We	Fr	Sa	Tu	Th
H	H	S	S	S	H	H	S	H	H

- **Formulate (Model)**
  - Every email that Bob sends during the week is ham, and during the weekend is spam.
- **Predict**
  - Today is Saturday, this email is spam.



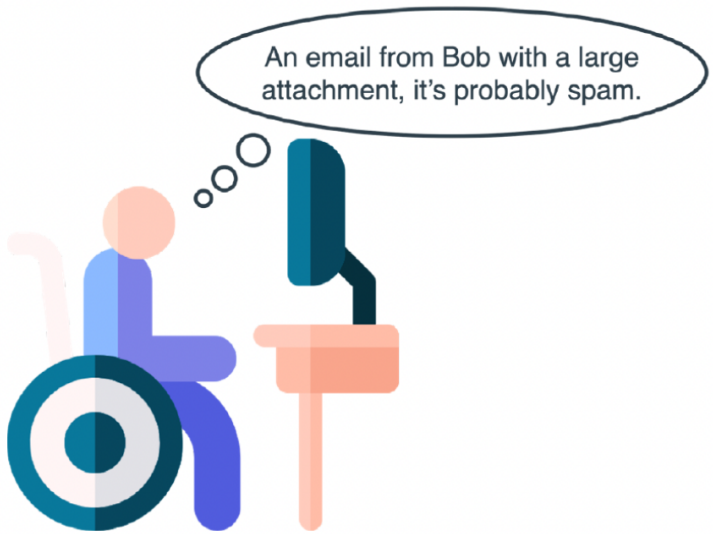
Example: Spam or Ham?!

Model 3

- **Remember**

1KB	2KB	16KB	20KB	18KB	3KB	5KB	25KB	1KB	3KB
H	H	S	S	S	H	H	S	H	H

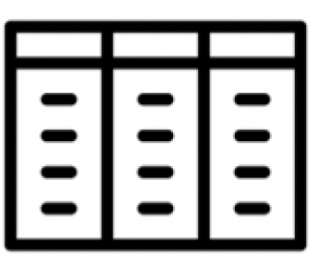
- **Formulate (Model)**
  - Any email larger of size 10KB or more is spam, and any email of size less than 10KB is ham.



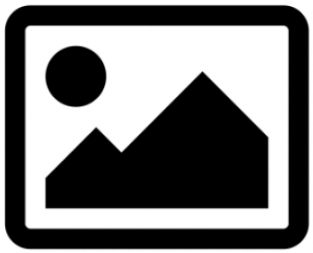
- **Predict**
  - The email size is 19 KB, this email is spam.

What is Feature?

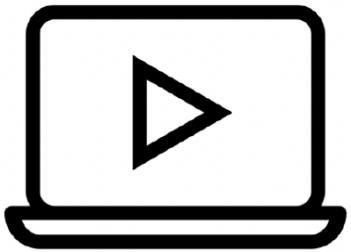
" Any property or characteristic of the data that the model can use to make predictions"



Tabular



Image



Video

Text

Audio

Example: Spam or Ham?!

Using the 2 features:

Model 4

- **If** an email is larger than 10 KB or it is sent on the weekend, then it is classified as spam.
- **Otherwise**, it is classified as ham.

Model 5

- **If** the email is sent during the week, then it must be larger than 15 KB to be classified as spam.
- **If** it is sent during the weekend, then it must be larger than 5 KB to be classified as spam.
- **Otherwise**, it is classified as ham.

Model 6

- **Consider** the number of the day, where Monday is 0, Tuesday is 1, and so on.
- If we add the number of the day to the size (in KB), and the result is 12 or more, then it is spam.
- **Otherwise**, it is classified as ham.

How Do Machines Learn?

3 Steps for Machines to Learn

- **Remember**: Look at a huge amount of data.
- **Formulate**: Create models by going through many rules and formulas, and check which model fits the data best.
- **Predict**: Use the model to make predictions about future data.

Machines can look at a huge amount of data and formulate a model quickly.

Email Classification Models

More than 2 features:

Model 7

- If the email has two or more spelling mistakes, then it is spam.
- Otherwise, if it has an attachment larger than 20 KB, it is spam.
- Otherwise, if the sender is not in our contact list, it is spam.
- Otherwise, if it has the words "free" and "prize", it is spam.
- Otherwise, it is classified as ham.

Model 8

- If 10 X (number of spelling mistakes) - 4 X (# of appearances of the word "free") - 10 X (# of appearances of the word "prize") ≥ 10 , then we classify the message as spam.
- Otherwise, we classify it as ham.

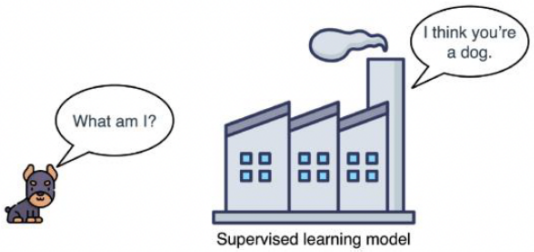
How Could These Problems Be Solved?!

- Predicting housing prices based on their size, number of rooms, location, etc.
- Detecting spam and non-spam emails based on the words in the email, the sender, etc.
- Recommending videos or movies to a user (for example, in YouTube, IMDB, etc.).
- Playing games like chess or Go.

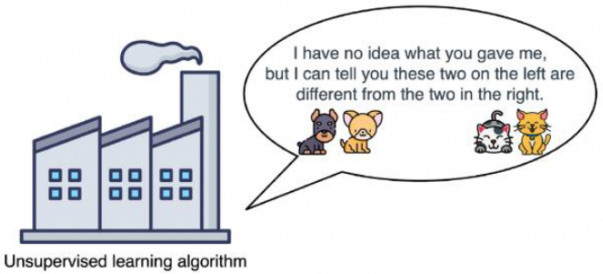
They are different, right?

Machine Learning Models Types

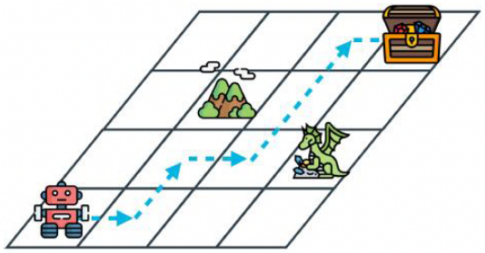
Supervised Learning



Unsupervised Learning



Reinforcement Learning



Let's talk first about the difference between Data, Labels, Predictions, and Features

What are Data, Labels, Predictions, Features?

- **Data** is simply information.

What are Data, Labels, Predictions, Features?

- **Data** is simply information.
- **Labels** are values that we try to predict.

What are Data, Labels, Predictions, Features?

- **Data** is simply information.
- **Labels** are values that we try to predict.
- **Predictions** are the guesses that the model makes.

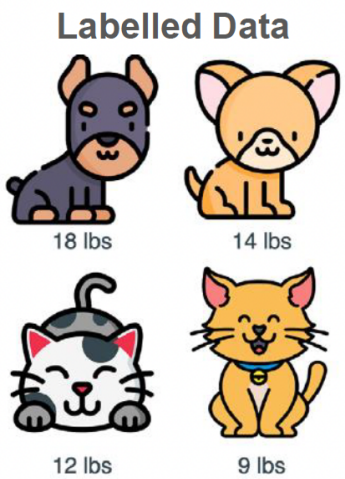
What are Data, Labels, Predictions, Features?

- **Data** is simply information.
- **Labels** are values that we try to predict.
- **Predictions** are the guesses that the model makes.
- **Features** are any property of the data that the model can use to make predictions.

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

Data types


Labelled Data
Dog
Dog
Cat
Cat

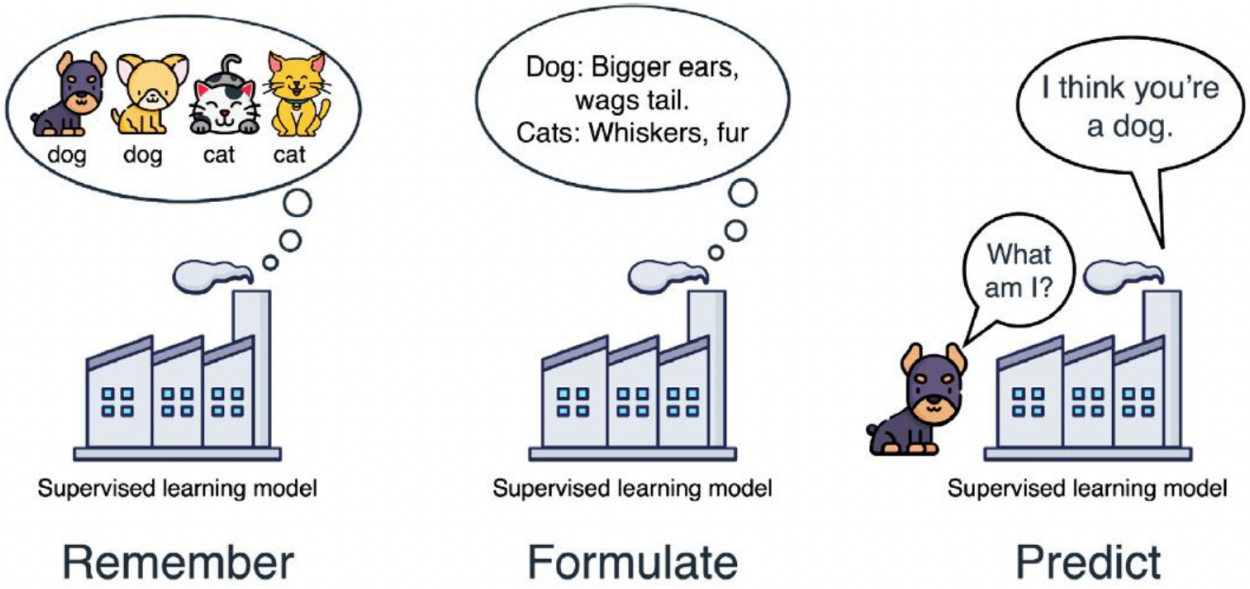

Labelled Data
18 lbs
14 lbs
12 lbs
9 lbs


Unlabelled Data

Outline

Outline

Supervised Learning



What are supervised models types?

Supervised Learning

Definition

- Learning from labeled data to predict outcomes or classify data.

Examples

Task	Example
Classification	Spam email detection
Regression	Predicting house prices

Table: Types of Supervised Learning Models

Classification Task: Credit Approval Application

Definition

- A classification task involves predicting a discrete value based on input features.

Example Dataset

Client ID	Income	Credit Score	Approval
001	$50,000	700	Approved
002	$45,000	650	Denied
003	$60,000	720	Approved

Table: Example of Labeled Data: Credit Approval

Regression Task: House Price Prediction

Definition

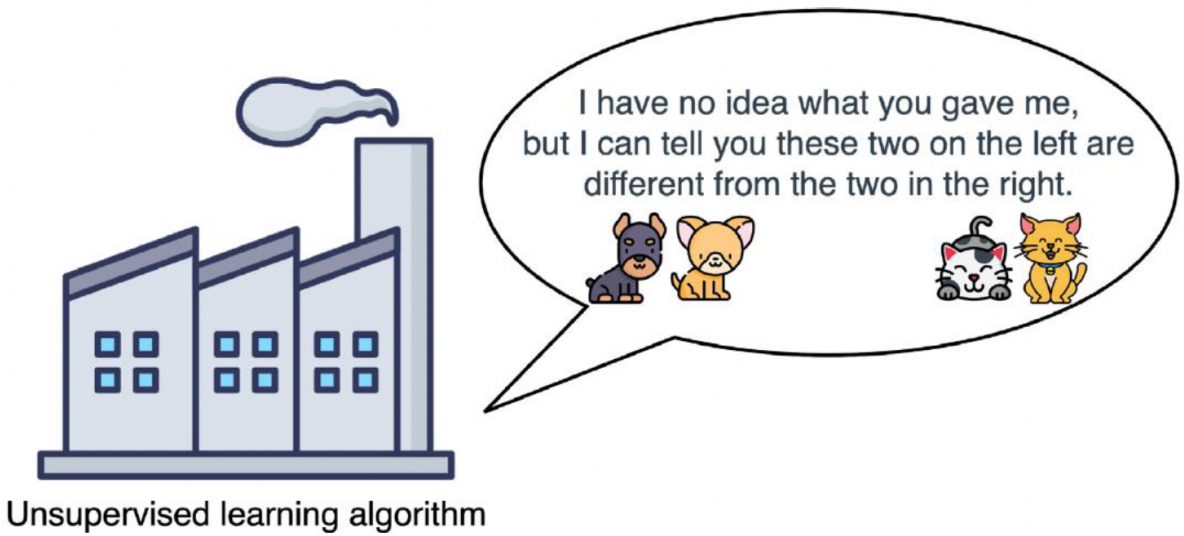
- A regression task involves predicting a continuous value based on input features.

Example Dataset

House ID	Size (sq ft)	Number of Bedrooms	Price
001	1,500	3	$300,000
002	2,000	4	$400,000
003	1,800	3	$350,000

Table: Example of Regression Task: House Price Prediction

Unsupervised Learning



What are supervised models types?

Outline

Unsupervised Learning

Definition

- Finding hidden patterns or intrinsic structures in unlabeled data.

Examples

Task	Example
Clustering	group data into clusters.
Dimensionality Reduction	simplify with fewer features.
Generative Algorithms	be able to generate new data points.

Table: Types of Unsupervised Learning Models

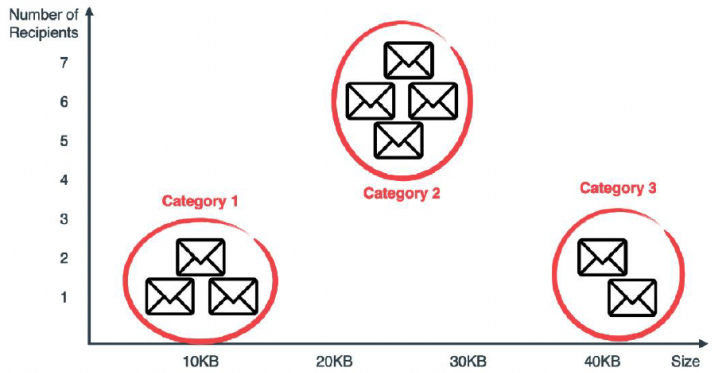
Clustering Task: Customers Segmentation

Definition

- A clustering task involves grouping data points into clusters based on their similarity.

Example Dataset

Email	Size	Recipients
1	8	1
2	12	1
3	43	1
4	10	2
5	40	2
6	25	5
7	23	6
8	28	6
9	26	7



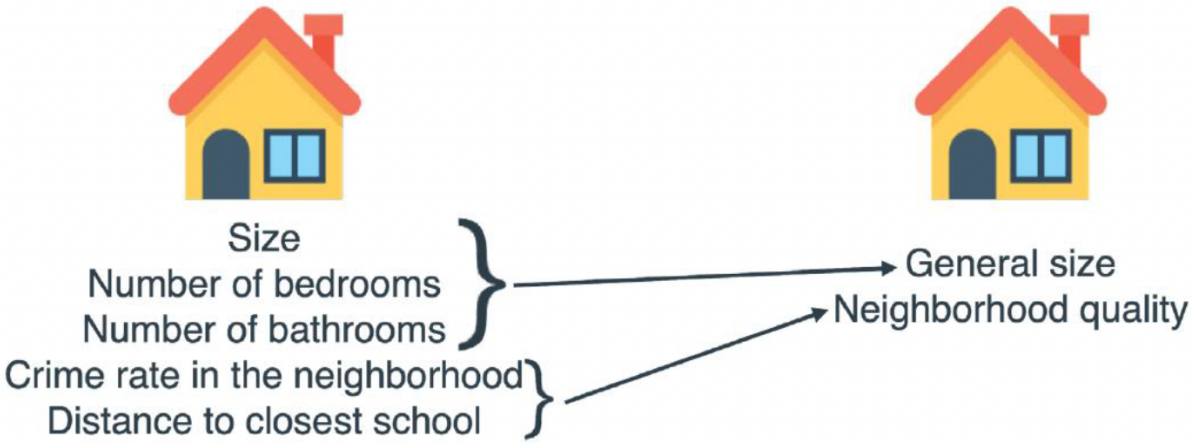
What are the problems here?

Dimensionality Reduction: Feature Reduction

Definition

- Dimensionality reduction involves reducing the number of features in a dataset while preserving important information.

Example



Generative Algorithms

Definition

- Generate new data points that does not exist before.

Example

Let's check this link:

- http://thispersondoesnotexist.com

Key Differences: Supervised vs. Unsupervised Learning

Comparison

Aspect	Supervised Learning	Unsupervised Learning
Data Type	Labeled Data	Unlabeled Data
Objective	Prediction/Classification	Pattern Discovery
Examples	Email Classification, House Price Prediction	Customer Segmentation, Dimensionality Reduction

Table: Key Differences Between Supervised and Unsupervised Learning

Supervised Learning Algorithms

Supervised Learning Algorithms

- **Linear Regression**: Predicts a continuous target variable.
- **Logistic Regression**: Classifies data into binary categories.
- **K-Nearest Neighbors (KNN)**: Classifies data based on the closest training examples in the feature space.
- **Decision Trees**: Uses a tree-like model of decisions and their possible consequences.
- **Support Vector Machines (SVM)**: Finds the hyperplane that best separates different classes.
- **Neural Networks**: Models complex relationships using layers of nodes (neurons).
- **Naive Bayes**: Applies Bayes' theorem with strong (naive) independence assumptions between features.

Unsupervised Learning Algorithms

Unsupervised Learning Algorithms

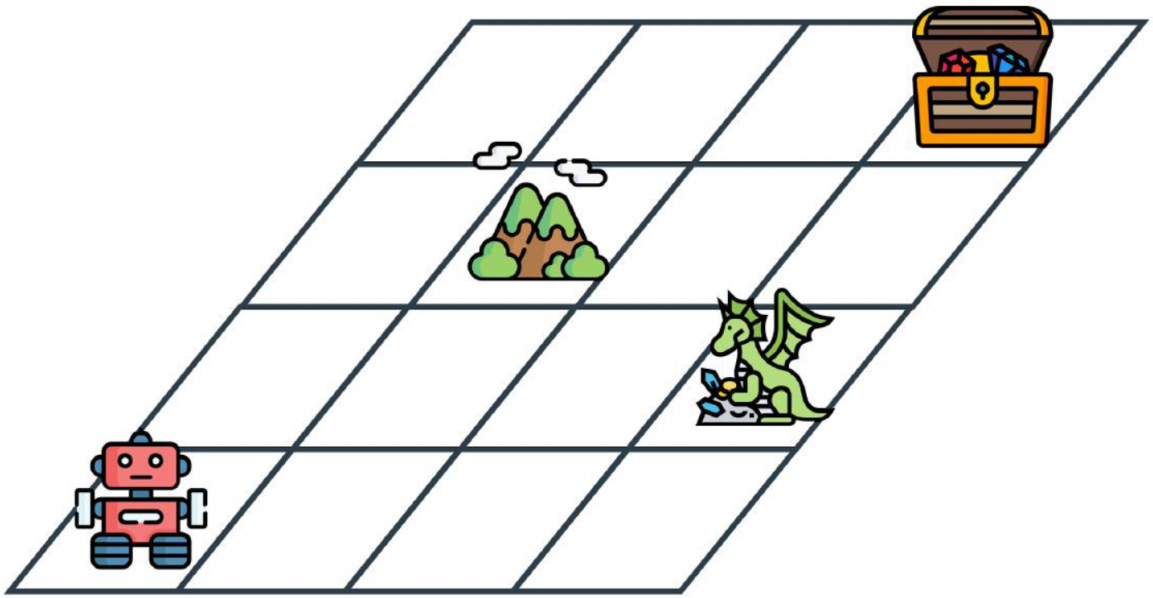
- **K-Means Clustering**: Partitions data into K distinct clusters based on distance metrics.
- **Hierarchical Clustering**: Builds a hierarchy of clusters using a tree-like structure.
- **Principal Component Analysis (PCA)**: Reduces the dimensionality of data while preserving variance.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: Visualizes high-dimensional data in lower dimensions.
- **Independent Component Analysis (ICA)**: Separates multivariate signals into additive, independent components.

Outline

Reinforcement Learning

Definition

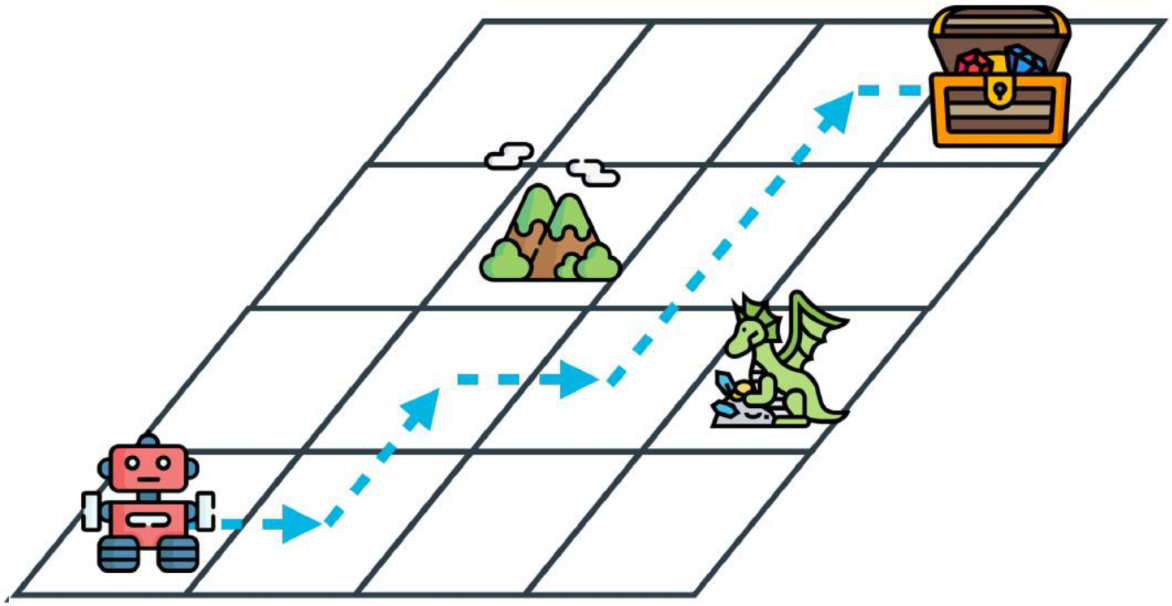
- Learning by interacting with the environment.



Reinforcement Learning

Definition

- Learning by interacting with the environment. (rewarding mechanism)



Reinforcement Learning

Definition

- Learning by interacting with the environment. (rewarding mechanism)

Example

Data Science and Machine Learning Workflow

1. **Data Collection**
   - Gather raw data from sources such as databases, APIs, or sensors.
2. **Data Preprocessing**
   - Clean and transform raw data into a usable format.
   - Example: Removing duplicates, handling missing values, and scaling numerical features.
3. **Model Training**
   - Train a machine learning model using preprocessed data.
   - Select and apply appropriate algorithms to build the model.
4. **Model Evaluation**
   - Assess the performance of the trained model using metrics such as accuracy or F1-score.
5. **Model Deployment**
   - Integrate the trained model into a production environment for real-time or batch predictions.

Exercise 1

For each of the following scenarios, state if it is an example of supervised or unsupervised learning. Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) A recommendation system on a social network that recommends potential friends to a user.

Exercise 1

For each of the following scenarios, state if it is an example of supervised or unsupervised learning. Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) A recommendation system on a social network that recommends potential friends to a user.

(b) A system in a news site that divides the news into topics.

Exercise 1

For each of the following scenarios, state if it is an example of supervised or unsupervised learning. Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) A recommendation system on a social network that recommends potential friends to a user.

(b) A system in a news site that divides the news into topics.

(c) The Google autocomplete feature for sentences.

Exercise 1

For each of the following scenarios, state if it is an example of supervised or unsupervised learning. Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) A recommendation system on a social network that recommends potential friends to a user.

(b) A system in a news site that divides the news into topics.

(c) The Google autocomplete feature for sentences.

(d) A recommendation system on an online retailer that recommends users what to buy based on their past history.

Exercise 1

For each of the following scenarios, state if it is an example of supervised or unsupervised learning. Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) A recommendation system on a social network that recommends potential friends to a user.

(b) A system in a news site that divides the news into topics.

(c) The Google autocomplete feature for sentences.

(d) A recommendation system on an online retailer that recommends users what to buy based on their past history.

(e) A system in a credit card company that captures fraudulent transactions.

Exercise 2

For each of the following applications of Machine Learning, would you use regression or classification to solve it? Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) An online store predicting how much money a user will spend on their site.

Exercise 2

For each of the following applications of Machine Learning, would you use regression or classification to solve it? Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) An online store predicting how much money a user will spend on their site.

(b) Alexa decoding voice and turning it into text.

Exercise 2

For each of the following applications of Machine Learning, would you use regression or classification to solve it? Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) An online store predicting how much money a user will spend on their site.

(b) Alexa decoding voice and turning it into text.

(c) Selling or buying stock from a particular company.

Exercise 2

For each of the following applications of Machine Learning, would you use regression or classification to solve it? Explain your answers. In cases of ambiguity, pick one and explain why you picked it.

(a) An online store predicting how much money a user will spend on their site.

(b) Alexa decoding voice and turning it into text.

(c) Selling or buying stock from a particular company.

(d) YouTube recommending a video to a user.

Outline

Outline

Introduction to Python for Machine Learning

Importance of Python in ML

- Python is a versatile and widely-used programming language in data science and machine learning.
- It offers simplicity and readability, making it accessible for both beginners and experts.
- Python has a strong community and extensive documentation, which supports the rapid development and deployment of ML models.

Key Libraries

- **NumPy**: Essential for numerical computing and handling large arrays and matrices. Provides support for mathematical functions and random number generation.
- **Pandas**: Key library for data manipulation and analysis. Offers data structures like DataFrames for handling structured data.
- **Matplotlib**: Widely used for creating static, animated, and interactive visualizations. Helps in plotting graphs and charts to visualize data and results.
- **Scikit-learn**: Popular library for implementing machine learning algorithms and models. Provides tools for classification, regression, clustering, and model evaluation.

Introduction to Python Tutorial

Why Learn Python?

- Python is easy to learn and read.
- Widely used in data science and machine learning.
- Extensive libraries and community support.

Getting Started with Python

- Installing Python
- Setting up a development environment
- Basic syntax and operations

Installing Python

Step-by-Step Guide

- Download the latest version of Python from `python.org`
- Follow the installation instructions for your operating system
- Verify the installation by running `python --version` in your terminal

Setting Up a Development Environment

Using Jupyter Notebooks

- Jupyter Notebooks are great for interactive coding and data visualization
- Install Jupyter using `pip install jupyter`
- Start a new notebook by running `jupyter notebook` in your terminal

Basic Syntax and Operations

Hello, World!

- The classic first program in Python

Code:

```
# hello_world.py
print("Hello, World!")
```

Variables and Data Types

Defining Variables

- Python is dynamically typed
- No need to declare the type of a variable

Code:

```
# variables.py
# Defining variables
a = 10
b = 3.14
c = "Hello"
d = [1, 2, 3]
e = {"key": "value"}
```

Common Data Types

- Integers, Floats, Strings, Lists, Dictionaries, etc.

Output: Variables and Data Types

Running the Code

- Output generated by running the `Code/variables.py` script

Output:

```
x = 5
y = 3.14
name = Alice
```

Arithmetic and Logical Operations in Python

Arithmetic Operations

Code:

```
# Arithmetic operations
a = 10
b = 5

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
floor_division = a // b
modulus = a % b
exponentiation = a ** b
```

Logical Operations

Code:

```
# Logical operations
x = True
y = False

logical_and = x and y
logical_or = x or y
logical_not = not x
```

Comparison Operations in Python

Comparison Operations

Code:

```
# Comparison operations
a = 10
b = 5
comparison_eq = a == b
comparison_neq = a != b
comparison_lt = a < b
comparison_gt = a > b
```

Control Structures: Conditionals

If Statements

- Control the flow of the program based on conditions

Code:

```
# conditionals.py
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

Control Structures: Conditionals

If Statements

- Control the flow of the program based on conditions

Code:

```
# conditionals.py
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

Running the Code

- Output generated by running the `Code/conditionals.py` script

Output:

```
x is greater than 10
y is positive
```

Control Structures: Loops

For and While Loops

- Iterate over a sequence of elements

Code:

```
# loops.py
# For loop
for i in range(5):
    print(f"For loop iteration: {i}")

# While loop
i = 0
while i < 5:
    print(f"While loop iteration: {i}")
    i += 1
```

Output: Loops

Running the Code

- Output generated by running the `Code/loops.py` script

Output:

```
0
1
2
3
4
Sum: 10
```

Functions in Python

Defining Functions

- Functions help in code reuse and organization
- Defined using the `def` keyword

Code:

```
# functions.py
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
print(greet("Bob"))
```

Output: Functions

Running the Code

- Output generated by running the `Code/functions.py` script

Output:

```
Hello, Alice!
Hello, Bob!
```

Outline

Lists

Lists in Python

- Ordered, mutable collection of items
- Supports indexing, slicing, and various methods

```
my_list = [1, 2, 3, 4, 5]
# Accessing elements
first_element = my_list[0]
last_element = my_list[-1]
# Modifying elements
my_list[2] = 10
# Adding elements
my_list.append(6)
# Removing elements
my_list.remove(4)
# Slicing the list
sub_list = my_list[1:4]
print("Original List: ", [1, 2, 3, 4, 5])
print("Modified List: ", my_list)
print("First Element: ", first_element)
print("Last Element: ", last_element)
print("Sub List: ", sub_list)
```

Tubles

Tubles in Python

- Ordered, immutable collection of items
- Supports indexing and slicing, but not modification

```
# Defining a tuple
my_tuple = (1, 2, 3, 4, 5)
# Accessing elements
first_element = my_tuple[0]
last_element = my_tuple[-1]
# Slicing the tuple
sub_tuple = my_tuple[1:4]
# Tuples are immutable, so you cannot
 ↪  modify them directly
# However, you can create a new tuple
 ↪  with the modified values
new_tuple = my_tuple + (6,)
print("Original Tuple: ", (1, 2, 3, 4,
 ↪  5))
print("New Tuple: ", new_tuple)
print("First Element: ", first_element)
print("Last Element: ", last_element)
print("Sub Tuple: ", sub_tuple)
```

Dictionaries

Dictionaries in Python

- Unordered collection of key-value pairs
- Keys must be unique and immutable
- Values can be of any data type

```
# Defining a dictionary
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}
# Accessing elements
name = my_dict["name"]
age = my_dict["age"]
# Adding a new key-value pair
my_dict["occupation"] = "Engineer"
# Modifying an existing key-value pair
my_dict["city"] = "San Francisco"
# Removing a key-value pair
del my_dict["age"]
# Checking if a key exists
has_city = "city" in my_dict
```

Iterating Over Lists, Tuples, and Dictionaries

Exmaple 1

```
# Iterating over a list
my_list = [1, 2, 3, 4, 5]
list_elements = []
for item in my_list:
    list_elements.append(f"List item: {item}")
```

Exmaple 2

```
# Iterating over a tuple
my_tuple = ('apple', 'banana', 'cherry')
tuple_elements = []
for item in my_tuple:
    tuple_elements.append(f"Tuple item: {item}")
```

Iterating Over Lists, Tuples, and Dictionaries

Exmaple 1

```
for key in my_dict:
    print(dict[key])
```

Exmaple 2

```
my_dict = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}
dict_elements = []
for key, value in my_dict.items():
    dict_elements.append(f"Key: {key}, Value: {value}")
```

Sets in Python

Defining and Using Sets

- Sets are collections of unique elements.
- Python provides built-in operations to work with sets, such as union, intersection, difference, and symmetric difference.

```
# sets.py

# Defining sets
my_set = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}

# Set operations
union_set = my_set | my_set2
intersection_set = my_set & my_set2
difference_set = my_set - my_set2

# Iterate over set
for element in union_set:
    print(element)
```

Exercises

1. Write a function `sum_all` that can take any number of arguments and returns their sum.
2. Define a function `is_even` that takes an integer and returns True if the number is even, and False otherwise.
3. Create a function `filter_even` that takes a list of numbers and a function as arguments. Use the function to filter out even numbers from the list.
4. Write a function `is_palindrome(s)` that checks if a given string s is a palindrome (reads the same backward as forward). The function should ignore spaces, punctuation, and case. For example, `is_palindrome("A man, a plan, a canal, Panama")` should return `True`.

Exercises

1. Develop a function `generate_primes(n)` that generates a list of prime numbers less than or equal to n. For example, `generate_primes(10)` should return [2, 3, 5, 7].
2. Write a function `are_anagrams(s1, s2)` that checks if two strings s1 and s2 are anagrams of each other. For example, `are_anagrams("listen", "silent")` should return True

Outline

Modules and Packages

Overview

- Modules are Python files (.py) that contain functions, classes, and variables.
- Packages are collections of modules organized in directories.
- Using modules and packages helps organize code and reuse functionality.

Common Python Libraries

- NumPy: For numerical computing.
- Pandas: For data manipulation.
- Matplotlib: For data visualization.
- Scikit-learn: For machine learning.

NumPy for Numerical Computing

Array Statistics Example

```
import numpy as np

# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
# Perform basic operations
mean_value = np.mean(arr)
sum_value = np.sum(arr)
max_value = np.max(arr)

print("Array:", arr)
print("Mean:", mean_value)
print("Sum:", sum_value)
print("Max:", max_value)
```

Array Operations

Array Operations Example

```
import numpy as np

# Create arrays
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])

# Basic operations
addition = arr1 + arr2
subtraction = arr1 - arr2
multiplication = arr1 * arr2
division = arr1 / arr2

print("Array 1:", arr1)
print("Array 2:", arr2)
print("Addition:\n", addition)
print("Subtraction:\n", subtraction)
print("Multiplication:\n", multiplication)
print("Division:\n", division)
```

Array Reshaping

Array Operations Example

```
import numpy as np

# Create an array
arr = np.arange(1, 13)  # Array from 1 to 12

# Reshape the array
reshaped_arr = arr.reshape(3, 4)  # Reshape to 3x4 array

# Transpose the array
transposed_arr = reshaped_arr.T

print("Original Array:\n", arr)
print("Reshaped Array:\n", reshaped_arr)
print("Transposed Array:\n", transposed_arr)
```

Advanced Array Operations

Advanced Array Operations Example

```
import numpy as np

# Create arrays
arr = np.array([1, 2, 3, 4, 5])
arr2 = np.array([10, 20, 30, 40, 50])
# Mathematical functions
sqrt_arr = np.sqrt(arr)
log_arr = np.log(arr)
exp_arr = np.exp(arr)
# Dot product
dot_product = np.dot(arr, arr2)

print("Array:", arr)
print("Square Root:\n", sqrt_arr)
print("Logarithm:\n", log_arr)
print("Exponential:\n", exp_arr)
print("Dot Product:", dot_product)
```

Random Number Generation

Random Number Generation

```
import numpy as np

# Generate random numbers
random_ints = np.random.randint(0, 10, size=5)
random_floats = np.random.random(size=5)

print("Random Integers:", random_ints)
print("Random Floats:", random_floats)
```

Outline

Pandas for Data Manipulation

Overview

- Pandas provides data structures like DataFrames for data manipulation and analysis.
- Ideal for handling structured data and performing complex data operations.

Code Example

```
import pandas as pd
# Create a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
# Display the DataFrame
print("DataFrame:\n", df)
```

Pandas for Data Manipulation

Output Example

```
DataFrame:
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

DataFrame Operations

Example

```
import pandas as pd
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
# Add a new column
df['Salary'] = [70000, 80000, 120000]
# Filter rows
filtered_df = df[df['Age'] > 30]
# Describe the DataFrame
description = df.describe()

print("DataFrame with Salary Column:\n", df)
print("\nFiltered DataFrame (Age > 30):\n", filtered_df)
print("\nDataFrame Description:\n", description)
```

DataFrame Operations

Output Example

```
DataFrame with Salary Column:
      Name  Age         City  Salary
0    Alice   25     New York   70000
1      Bob   30  Los Angeles   80000
2  Charlie   35      Chicago  120000

Filtered DataFrame (Age > 30):
      Name  Age     City  Salary
2  Charlie   35  Chicago  120000

DataFrame Description:
             Age        Salary
count   3.000000      3.000000
mean   30.000000  85000.000000
std     5.000000  25000.000000
min    25.000000  70000.000000
25%    27.500000  72500.000000
50%    30.000000  80000.000000
75%    32.500000  95000.000000
max    35.000000  120000.000000
```

DataFrame Operations

Example

```
import pandas as pd
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
# Add a new column
df['Salary'] = [70000, 80000, 120000]
# Filter rows
filtered_df = df[df['Age'] > 30]
# Describe the DataFrame
description = df.describe()

print("DataFrame with Salary Column:\n", df)
print("\nFiltered DataFrame (Age > 30):\n", filtered_df)
print("\nDataFrame Description:\n", description)
```

DataFrame Operations

Output Example

```
DataFrame with Salary Column:
      Name  Age         City  Salary
0    Alice   25     New York   70000
1      Bob   30  Los Angeles   80000
2  Charlie   35      Chicago  120000

Filtered DataFrame (Age > 30):
      Name  Age     City  Salary
2  Charlie   35  Chicago  120000

DataFrame Description:
             Age        Salary
count   3.000000      3.000000
mean   30.000000  85000.000000
std     5.000000  25000.000000
min    25.000000  70000.000000
25%    27.500000  72500.000000
50%    30.000000  80000.000000
75%    32.500000  95000.000000
max    35.000000  120000.000000
```

DataFrame Indexing and Selection

Example

```
import pandas as pd
# Create a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
# Select rows by label
row_by_label = df.loc[1]
# Select rows by position
row_by_position = df.iloc[2]
# Select a specific cell
cell_value = df.at[1, 'City']
print("Row by Label:\n", row_by_label)
print("\nRow by Position:\n", row_by_position)
print("\nSpecific Cell Value:\n", cell_value)
```

DataFrame Indexing and Selection

Output Example

```
Row by Label:
 Name            Bob
Age              30
City    Los Angeles
Name: 1, dtype: object

Row by Position:
 Name          Charlie
Age                35
City          Chicago
Name: 2, dtype: object

Specific Cell Value:
 Los Angeles
```

Handling Missing Data

Example

```
import pandas as pd
import numpy as np
# Create a DataFrame with missing values
data = {
    'Name': ['Alice', 'Bob', 'Charlie', np.nan],
    'Age': [25, np.nan, 35, 40],
    'City': ['New York', 'Los Angeles', np.nan, 'Chicago']
}
df = pd.DataFrame(data)

# Handle missing data
df_filled = df.fillna({'Name': 'Unknown',
    'Age': df['Age'].mean(), 'City': 'Unknown'})

print("Original DataFrame with Missing Values:\n", df)
print("\nDataFrame after Filling Missing Values:\n", df_filled)
```

Handling Missing Data

Output Example

```
Original DataFrame with Missing Values:
       Name   Age         City
0     Alice  25.0     New York
1       Bob   NaN  Los Angeles
2   Charlie  35.0          NaN
3       NaN  40.0      Chicago

DataFrame after Filling Missing Values:
       Name   Age         City
0     Alice  25.0     New York
1       Bob  30.0  Los Angeles
2   Charlie  35.0      Unknown
3   Unknown  40.0      Chicago
```

Outline

Matplotlib for Data Visualization

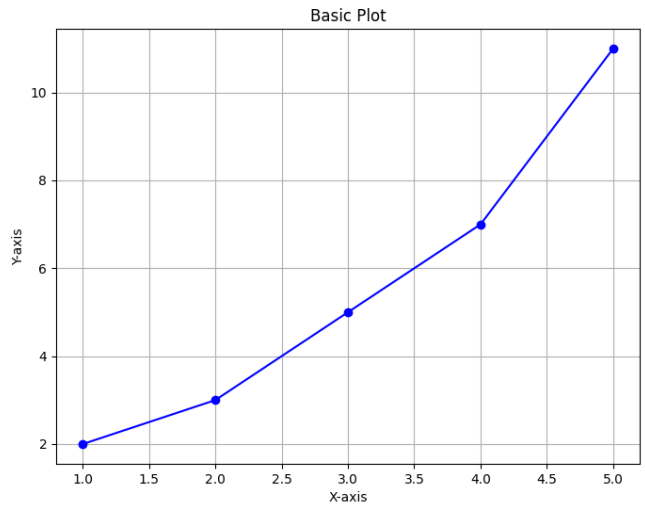
Matplotlib Basics

- Creating plots and charts
- Customizing visualizations

Code:

```
import matplotlib.pyplot as plt
# Data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
# Basic Plot
plt.figure(figsize=(4, 2))
plt.plot(x, y, marker='o', linestyle='-',
 ↪  color='b')
plt.title('Basic Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
```

Output:



Outline

Scikit-learn for Machine Learning I

Scikit-learn Basics

- Implementing ML algorithms
- Model training, evaluation, and prediction

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
↪  classification_report
import seaborn as sns
import matplotlib.pyplot as plt
# Load Wine dataset
wine = load_wine()
X = wine.data
y = wine.target
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
↪  random_state=42)
# Train a KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
```

Scikit-learn for Machine Learning II

```
15  knn.fit(X_train, y_train)
16  # Predict on test data
17  y_pred = knn.predict(X_test)
18  # Calculate accuracy
19  accuracy = accuracy_score(y_test, y_pred)
20  print(f"Accuracy: {accuracy:.2f}")
21  # Confusion matrix
22  cm = confusion_matrix(y_test, y_pred)
23  print(f"Confusion Matrix:\n{cm}")
24  # Classification report
25  cr = classification_report(y_test, y_pred, target_names=wine.target_names)
26  print(f"Classification Report:\n{cr}")
27  # Plot confusion matrix
28  sns.heatmap(cm, annot=True, cmap='viridis', fmt='d',
    ↪  xticklabels=wine.target_names, yticklabels=wine.target_names)
29  plt.xlabel('Predicted')
30  plt.ylabel('True')
31  plt.title('Confusion Matrix')
32  plt.savefig('sklearn_train_model_wine_output.png')
33  plt.show()
```

Questions