Supervised Machine Learning

Bharat Kumar

Lecturer in Computer Science

Machine

- computer, algorithm, or model — that processes input data, applies learning algorithms, and produces outputs or predictions

Learning

- "Learning is any process by which a system improves performance from experience."

  Herbert A. Simon

Machine Learning

- "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed."

Samuel A. L. (1959)

"A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

Mitchell T. (1997). *Machine Learning*. McGraw-Hill

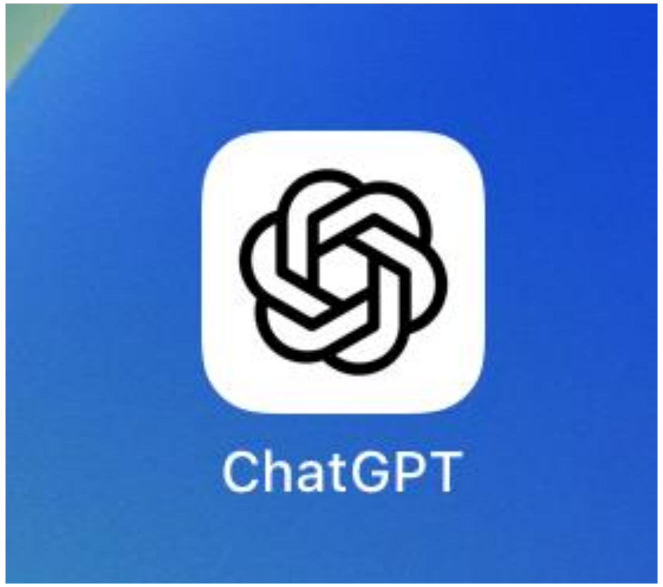
Machine Learning

- Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and statistical models that enable computers to perform specific tasks without being explicitly programmed.

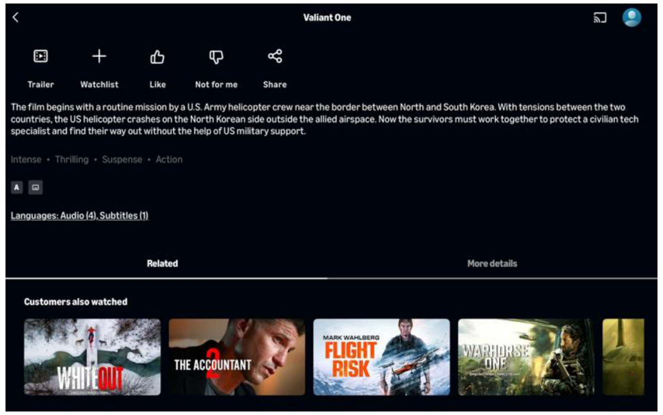
- these systems improve their performance on a task through experience, typically by identifying patterns in data.

Applications of Machine Learning

- Google Search
- ChatGPT
- Face recognition
- Movie Recommendations
- Siri , Alex
- Self-driving-car



Google
Google Search
I'm Feeling Lucky
Google Gemini, copy a holiday list and add it to my Google Calendar
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ

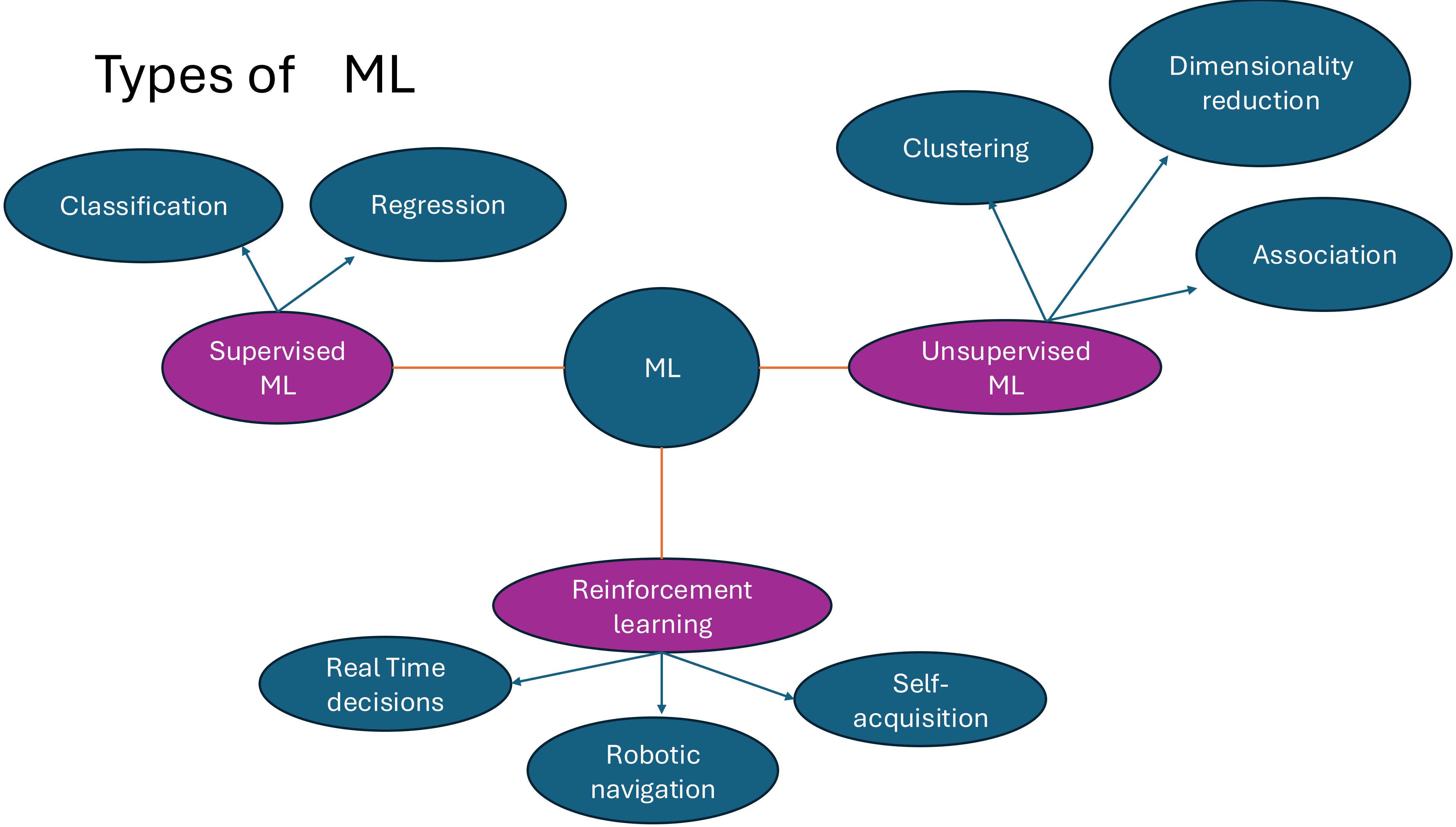


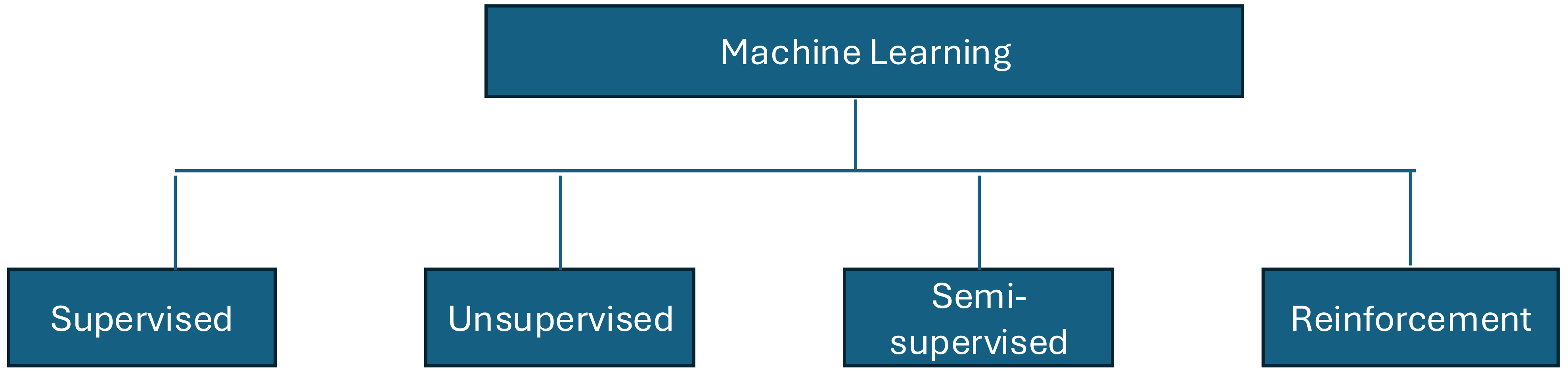
The film begins with a routine mission by a U.S. Army helicopter crew near the border between North and South Korea. With tensions between the two countries, the US helicopter crashes on the North Korean side outside the allied airspace. Now the survivors must work together to protect a civilian tech specialist and find their way out without the help of US military support.

Intense • Thrilling • Suspense • Action

Languages: Audio (4), Subtitles (1)

Types of ML


Dimensionality reduction
Clustering
Classification
Regression
Association
Supervised ML
ML
Unsupervised ML
Reinforcement learning
Real Time decisions
Self-acquisition
Robotic navigation


Machine Learning
Supervised
Unsupervised
Semi-supervised
Reinforcement

Types of Machine Learning

1 **Supervised Learning**
— Learning from labelled data (input-output pairs).

2 **Unsupervised Learning**
— Learning patterns from unlabelled data.

3 **Semi-Supervised Learning**
— Learning from a small amount of labelled data and a large amount of unlabelled data.

4 **Reinforcement Learning**
— Learning through interaction with an environment, using rewards and penalties to improve performance.

Types of Machine Learning

Type	Description	Examples
Supervised Learning	Learns from labeled data	Spam detection, Image classification
Unsupervised Learning	Finds patterns in unlabeled data	Customer segmentation, Clustering
Semi-Supervised Learning	Uses small labeled + large unlabeled data	Medical image classification
Reinforcement Learning	Learns by trial and error with rewards	Game playing, Robotics

Supervised Learning

- **Definition:**
  The algorithm learns from labeled data, meaning the input comes with the correct output.
- **Goal:**
  Learn a function that maps inputs to outputs.
- **Applications:**
  - Email spam detection (spam or not spam)
  - Image classification (cat or dog)
  - Medical diagnosis (disease or no disease)
  - Credit scoring (good or bad risk)
- **Common Algorithms:**
  - Linear Regression
  - Logistic Regression
  - Decision Trees
  - Support Vector Machines (SVM)
  - Neural Networks
  - k-Nearest Neighbors (k-NN)

Unsupervised Learning

- **Definition:**
  The algorithm works with unlabeled data and tries to find hidden patterns or structure.
- **Goal:**
  Discover the underlying structure or distribution.
- **Applications:**
  - Customer segmentation
  - Anomaly detection
  - Market basket analysis
  - Dimensionality reduction
- **Common Algorithms:**
  - k-Means Clustering
  - Hierarchical Clustering
  - Principal Component Analysis (PCA)
  - Autoencoders

Reinforcement Learning

- **Definition:**
  The algorithm learns by interacting with an environment, receiving rewards or penalties for actions taken.
- **Goal:**
  Learn a sequence of actions that maximizes cumulative reward.
- **Applications:**
  - Game playing (chess, Go, video games)
  - Robotics (navigation, control)
  - Self-driving cars
  - Stock trading strategies
- **Common Algorithms:**
  - Q-Learning
  - Deep Q-Networks (DQN)
  - Policy Gradient Methods
  - Actor-Critic Methods

Semi-Supervised Learning

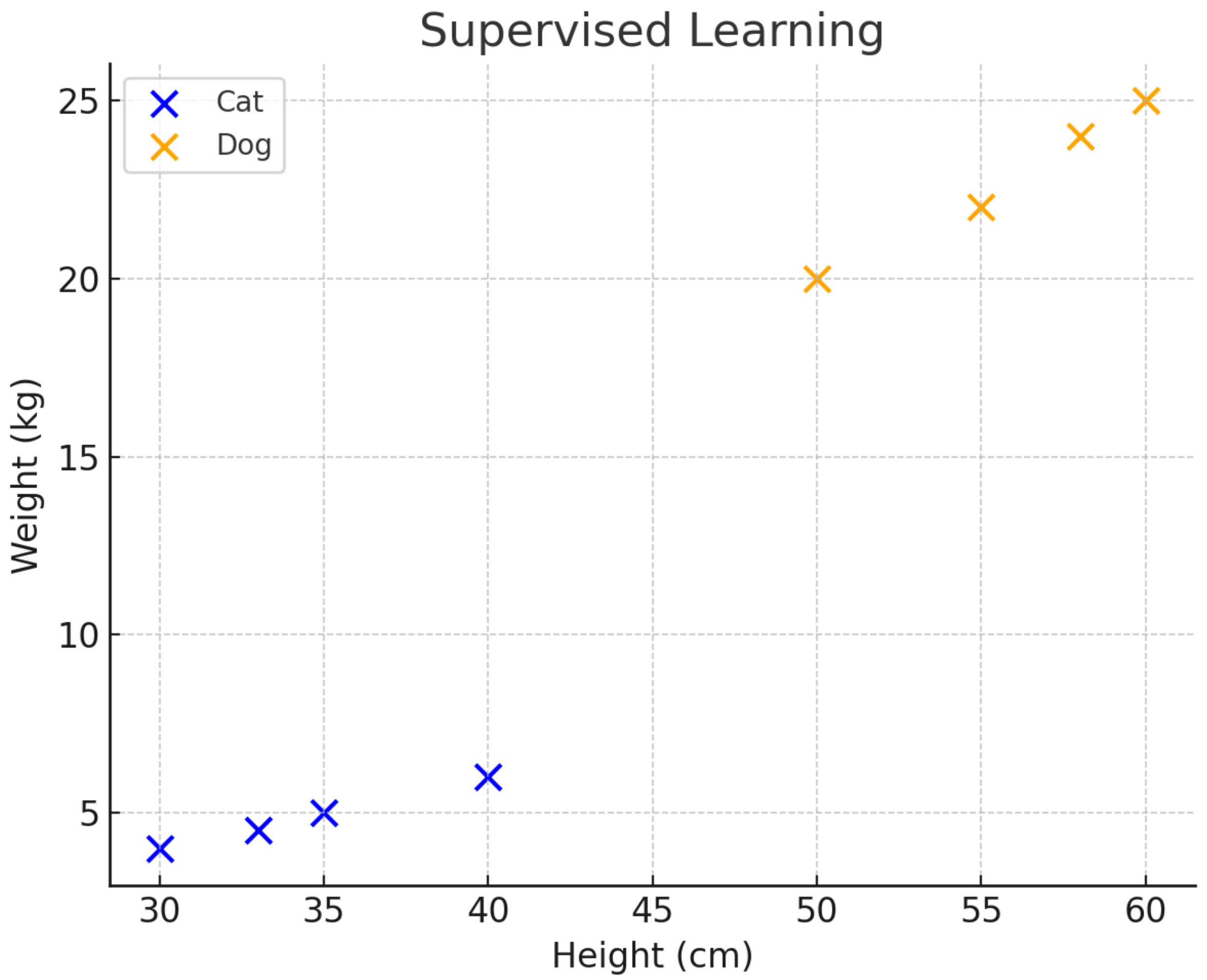
- **Definition:**
  Uses a small amount of labeled data with a large amount of unlabeled data.
- **Applications:**
  - When labeling data is expensive (medical images, legal documents)
  - Natural Language Processing (NLP)
- **Techniques:**
  - Self-training
  - Co-training
  - Graph-based methods

Key terminology

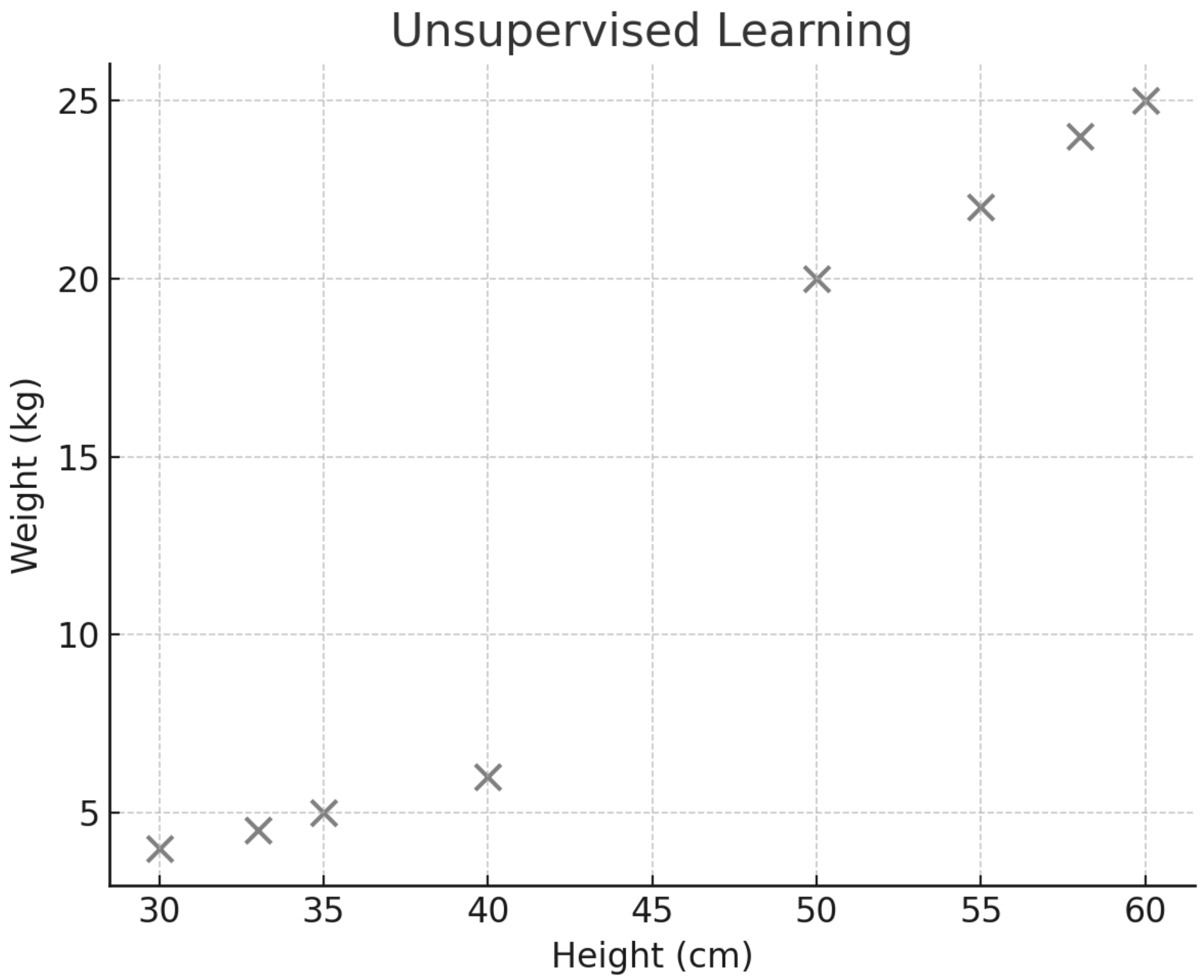
Term	Explanation	Example
Data	Information fed into the system	Customer purchases
Features (X)	Inputs, independent variables	Age, Income
Label (Y)	Output, dependent variable	Loan approved (Yes/No)
Training Set	Data used to train model	Historical data
Test Set	New unseen data	Future data
Model	Learned mapping function	Credit approval model

Terminology

- **Dataset**: A collection of data points or examples used for training and testing the ML model.
- **Feature**: An individual measurable property or characteristic of a data point.
- **Label (or Target)**: The output or result the model is trying to predict.
- **Model**: A mathematical representation learned by the algorithm from the training data.
- **Training**: The process of feeding data into the model to help it learn patterns.
- **Testing**: Evaluating the model's performance using unseen data.
- **Overfitting**: When a model performs well on training data but poorly on test data.
- **Underfitting**: When a model fails to capture patterns even in the training data.
- **Supervised Learning**: The model learns from labeled data.
- **Unsupervised Learning**: The model learns patterns from unlabeled data.



Supervised Learning
Cat
Dog
Weight (kg)
Height (cm)



Unsupervised Learning
Weight (kg)
Height (cm)

Cat and Dog dataset

Labelled Data

Index	Height (cm)	Weight (kg)	Label
1	30	4	Cat (0)
2	35	5	Cat (0)
3	40	6	Cat (0)
4	33	4.5	Cat (0)
5	50	20	Dog (1)
6	60	25	Dog (1)
7	55	22	Dog (1)
8	58	24	Dog (1)

Unlabelled Data

Index	Height (cm)	Weight (kg)
1	30	4
2	35	5
3	40	6
4	33	4.5
5	50	20
6	60	25
7	55	22
8	58	24

```python
# Features: Height (cm), Weight (kg)
X = [
    [30, 4],
    [35, 5],
    [40, 6],
    [33, 4.5],
    [50, 20],
    [60, 25],
    [55, 22],
    [58, 24]
]

# Labels: 0 = Cat, 1 = Dog
y = [0, 0, 0, 0, 1, 1, 1, 1]
```

🐈 🐈 🐈 🐈 🐈 🐕‍🦺 🐕‍🦺 🐕‍🦺 🐕‍🦺

Iris Dataset : Activity by students

- Perform the DataFrame Operations for preprocessing

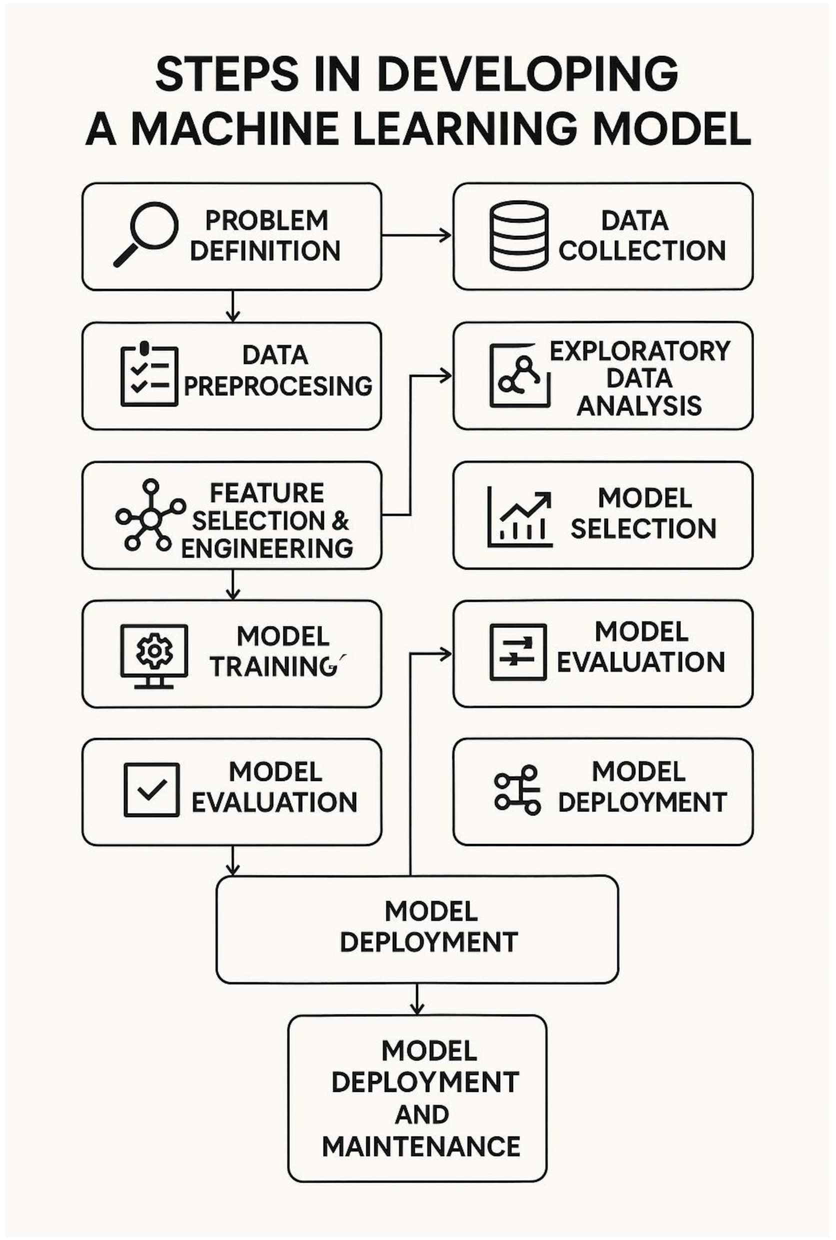
Key tasks of machine learning

- **Classification**: Predicting a categorical label (e.g., spam or not spam).
- **Regression**: Predicting a continuous value (e.g., house price).
- **Clustering**: Grouping similar data points together (e.g., customer segmentation).
- **Dimensionality Reduction**: Reducing the number of input features while retaining important information (e.g., PCA).
- **Anomaly Detection**: Identifying unusual or rare data points.

How to choose the right algorithm ?

- **Type of Data**: Whether it's numerical, categorical, text, or images.
- **Problem Type**: Classification, regression, clustering, etc.
- **Model Complexity**: Simple models like linear regression vs. complex ones like neural networks.
- **Accuracy Requirements**: Trade-off between performance and interpretability.
- **Training Time and Resources**: Some algorithms require extensive computation.
- **Scalability**: Consider how well the algorithm handles large datasets.

Steps in developing machine learning



Steps in Developing a Machine Learning Model

- **Define the Problem**: Clearly identify what you want the model to predict or classify.
- **Collect Data**: Gather relevant and sufficient data.
- **Preprocess Data**: Clean, normalize, and transform data for use in algorithms.
- **Split Data**: Divide into training and testing sets (e.g., 80/20 split).
- **Choose Algorithm**: Select the appropriate machine learning technique.
- **Train the Model**: Fit the model to training data.
- **Evaluate the Model**: Test the model on unseen data.
- **Tune Parameters**: Improve performance using hyperparameter optimization.
- **Deploy Model**: Use the trained model in real-world applications.
- **Monitor and Update**: Continually evaluate and retrain as needed.

why python?

- **Free and Open Source**
- **Large Community**
- **Simple Syntax** → easier to learn.
- **Powerful Libraries**:
  - NumPy → Numerical computing
  - Pandas → Data manipulation
  - Scikit-Learn → ML algorithms
  - TensorFlow, PyTorch → Deep learning
  - Matplotlib, Seaborn → Visualization

Numpy library

- NumPy is a foundational library in Python for numerical computations. It provides support for large multi-dimensional arrays and matrices, along with mathematical functions to operate on them.

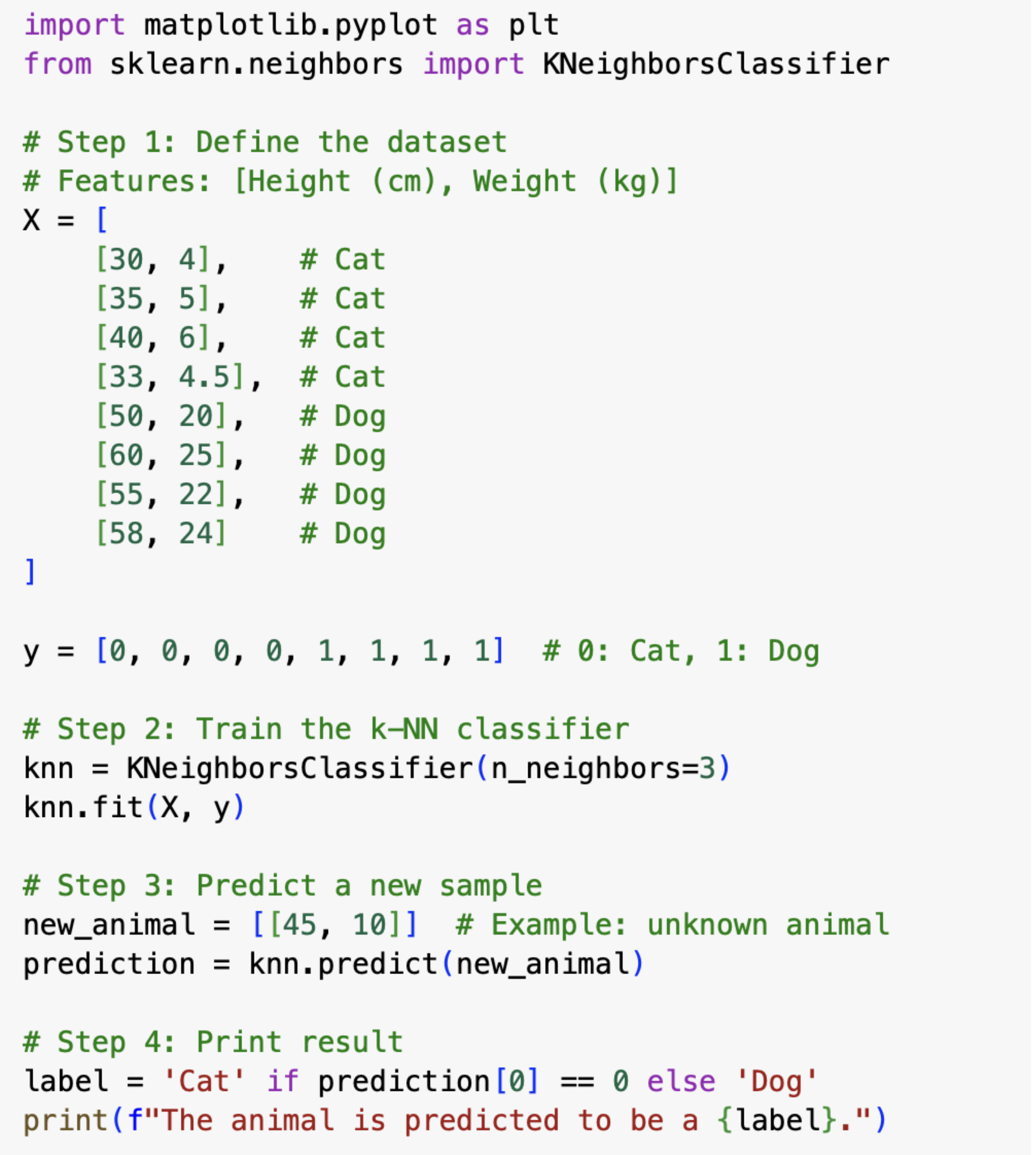
Numpy

```python
import numpy as np

# Step 1: Create a NumPy array
data = np.array([10, 20, 30, 40, 50])
print("Original Array:", data)

# Step 2: Calculate basic statistics
print("Mean:", np.mean(data))
print("Maximum:", np.max(data))
print("Minimum:", np.min(data))

# Step 3: Create a 2D array (matrix)
matrix = np.array([[1, 2], [3, 4]])
print("\nMatrix:\n", matrix)

# Step 4: Transpose and matrix multiplication
transpose = matrix.T
print("\nTranspose:\n", transpose)

product = np.dot(matrix, transpose)
print("\nMatrix × Transpose:\n", product)
```

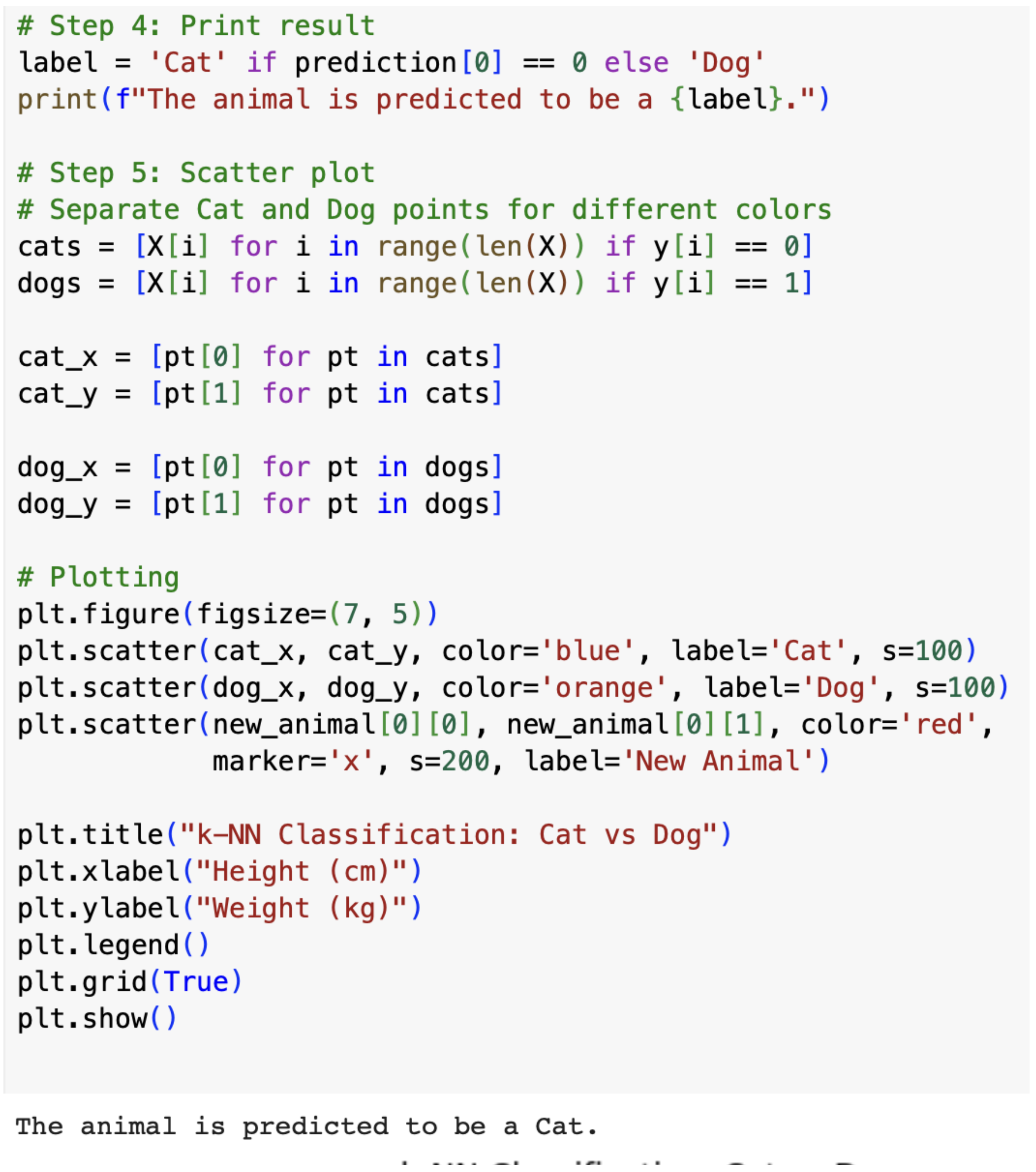
```
Original Array: [10 20 30 40 50]
Mean: 30.0
Maximum: 50
Minimum: 10

Matrix:
 [[1 2]
 [3 4]]

Transpose:
 [[1 3]
 [2 4]]

Matrix × Transpose:
 [[ 5 11]
 [11 25]]
```

k-Nearest Neighbors classification algorithm

- k-Nearest Neighbors (k-NN) is a simple, instance-based classification algorithm. It classifies new data points based on the majority class among its k nearest neighbors in the feature space. Distance metrics like Euclidean distance are used to determine "closeness."
- Steps:
    - Store all labelled training examples.
    - For a new data point, compute its distance from all training points.
    - Identify the k closest examples.
    - Assign the class based on majority voting.



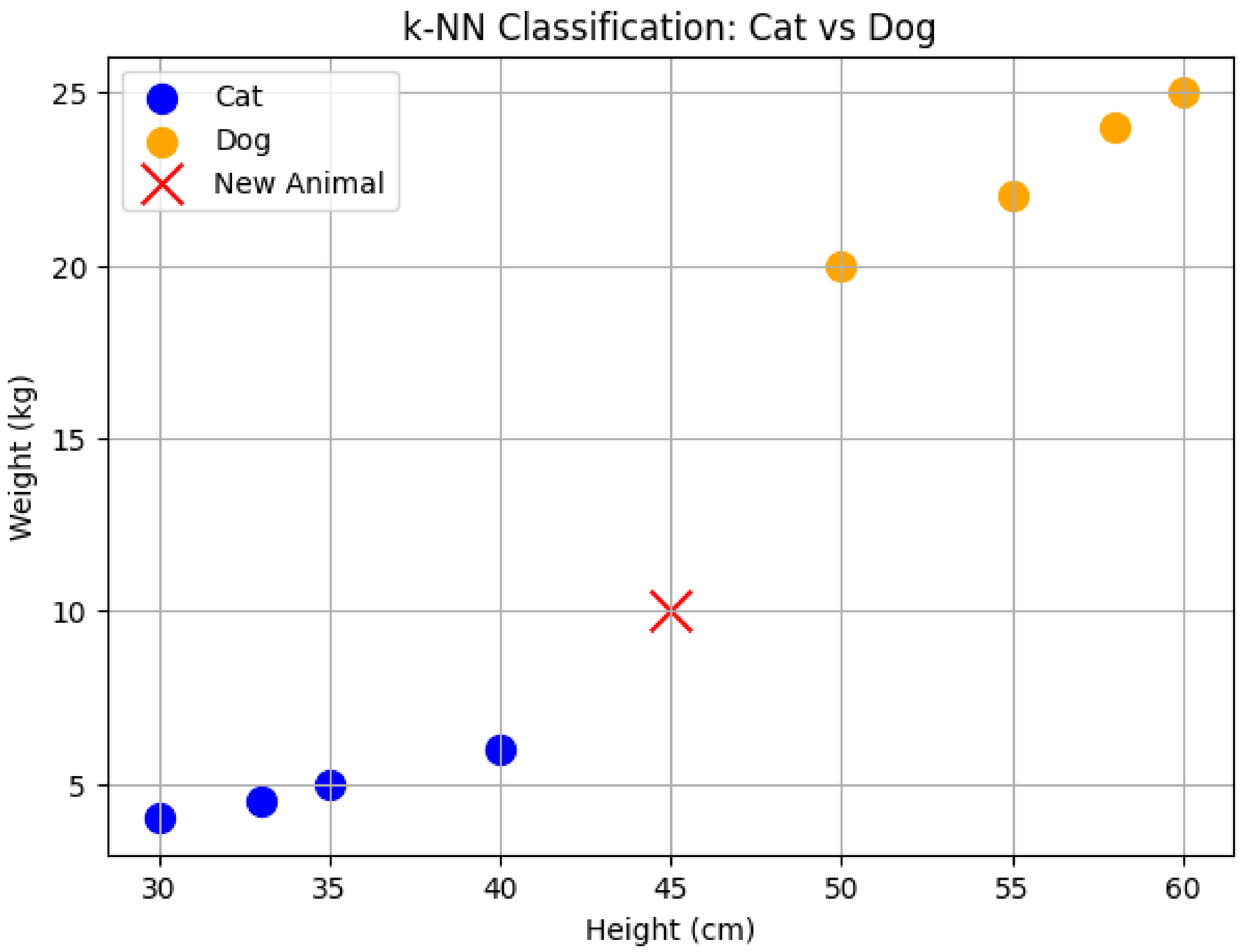
```python
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

# Step 1: Define the dataset
# Features: [Height (cm), Weight (kg)]
X = [
    [30, 4],    # Cat
    [35, 5],    # Cat
    [40, 6],    # Cat
    [33, 4.5],  # Cat
    [50, 20],   # Dog
    [60, 25],   # Dog
    [55, 22],   # Dog
    [58, 24]    # Dog
]

y = [0, 0, 0, 0, 1, 1, 1, 1]  # 0: Cat, 1: Dog

# Step 2: Train the k-NN classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)

# Step 3: Predict a new sample
new_animal = [[45, 10]]  # Example: unknown animal
prediction = knn.predict(new_animal)

# Step 4: Print result
label = 'Cat' if prediction[0] == 0 else 'Dog'
print(f"The animal is predicted to be a {label}.")
```



```python
# Step 4: Print result
label = 'Cat' if prediction[0] == 0 else 'Dog'
print(f"The animal is predicted to be a {label}.")
```

```python
# Step 5: Scatter plot
# Separate Cat and Dog points for different colors
cats = [X[i] for i in range(len(X)) if y[i] == 0]
dogs = [X[i] for i in range(len(X)) if y[i] == 1]

cat_x = [pt[0] for pt in cats]
cat_y = [pt[1] for pt in cats]

dog_x = [pt[0] for pt in dogs]
dog_y = [pt[1] for pt in dogs]

# Plotting
plt.figure(figsize=(7, 5))
plt.scatter(cat_x, cat_y, color='blue', label='Cat', s=100)
plt.scatter(dog_x, dog_y, color='orange', label='Dog', s=100)
plt.scatter(new_animal[0][0], new_animal[0][1], color='red',
            marker='x', s=200, label='New Animal')

plt.title("k-NN Classification: Cat vs Dog")
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.legend()
plt.grid(True)
plt.show()
```

```
The animal is predicted to be a Cat.
```


k-NN Classification: Cat vs Dog
Cat
Dog
New Animal
Weight (kg)
Height (cm)
5
10
15
20
25
30
35
40
45
50
55
60

Parsing and importing data from a text file

data.txt

```
45,170
50,165
60,180
55,175
```

```python
# Open and read the file
with open('data.txt', 'r') as file:
    lines = file.readlines()

# Parse the data into a list of lists
data = []
for line in lines:
    values = line.strip().split(',')     # split on comma
    data.append([float(val) for val in values])  # convert to float

print("Parsed Data:")
for row in data:
    print(row)
```

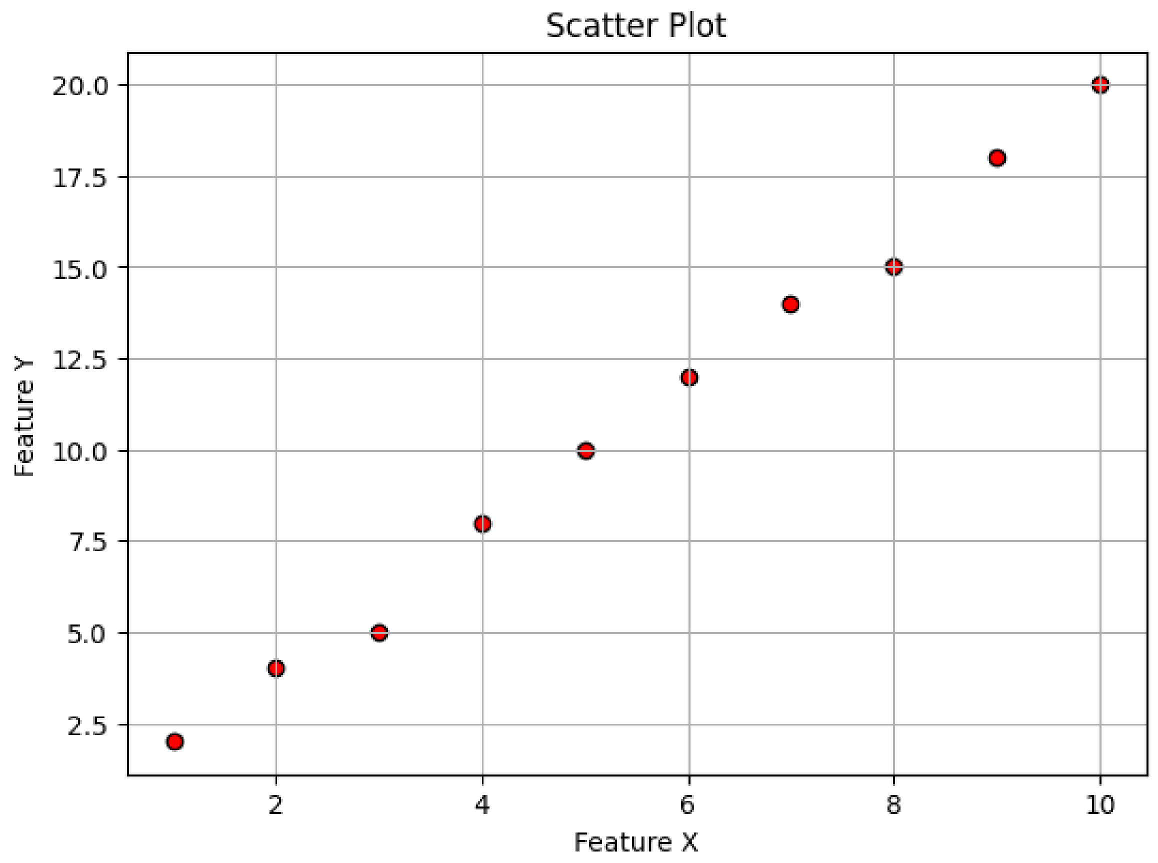
Output

```
[
    [ 45, 170],
    [50, 165] ,
    [60, 180] ,
    [55, 175]
]
```

Creating scatter plots with Matplotlib

- Scatter plots help visualize relationships between two numeric variables.

```python
import matplotlib.pyplot as plt

# Fixed sample data for scatter plot
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 8, 10, 12, 14, 15, 18, 20]

# Create scatter plot
plt.figure(figsize=(7, 5))
plt.scatter(x, y, color='teal', edgecolor='black')
plt.title("Sample Scatter Plot")
plt.xlabel("Feature X")
plt.ylabel("Feature Y")
plt.grid(True)
plt.show()
```



Normalizing numeric values

- Min-Max Normalization is a technique to scale numeric data so that all values fall within a specific range, typically 0 to 1.

- Why ?

Normalization is a key data preprocessing step used to scale numeric features so they fall within a common range. It helps machine learning algorithms perform accurately, efficiently, and fairly.

Min Max Normalization

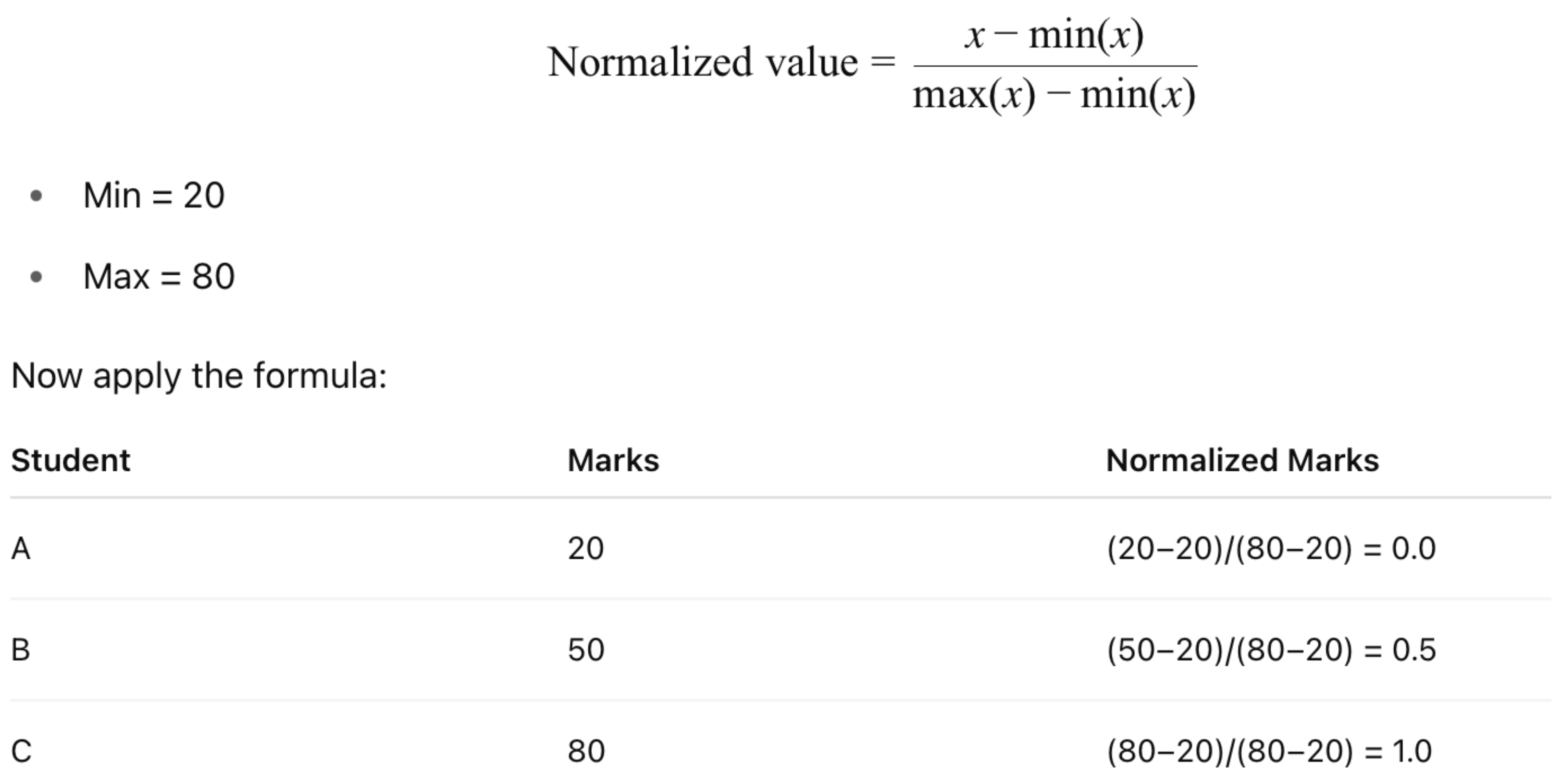
The formula for Min-Max Normalization is:

$$x_{\text{normalized}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Where:

- x is the original value
- $\min(x)$ is the minimum value in the column
- $\max(x)$ is the maximum value in the column

This formula maps the **minimum value to 0**, the **maximum value to 1**, and everything in between to a number between 0 and 1.



$$\text{Normalized value} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Min = 20
- Max = 80

Now apply the formula:

Student	Marks	Normalized Marks
A	20	(20–20)/(80–20) = 0.0
B	50	(50–20)/(80–20) = 0.5
C	80	(80–20)/(80–20) = 1.0

- Now all values are between 0 and 1
- Useful in machine learning because it makes all features equal in scale
- Prevents large numbers (like salary or age) from dominating smaller ones (like marks or height)

End of Unit - I