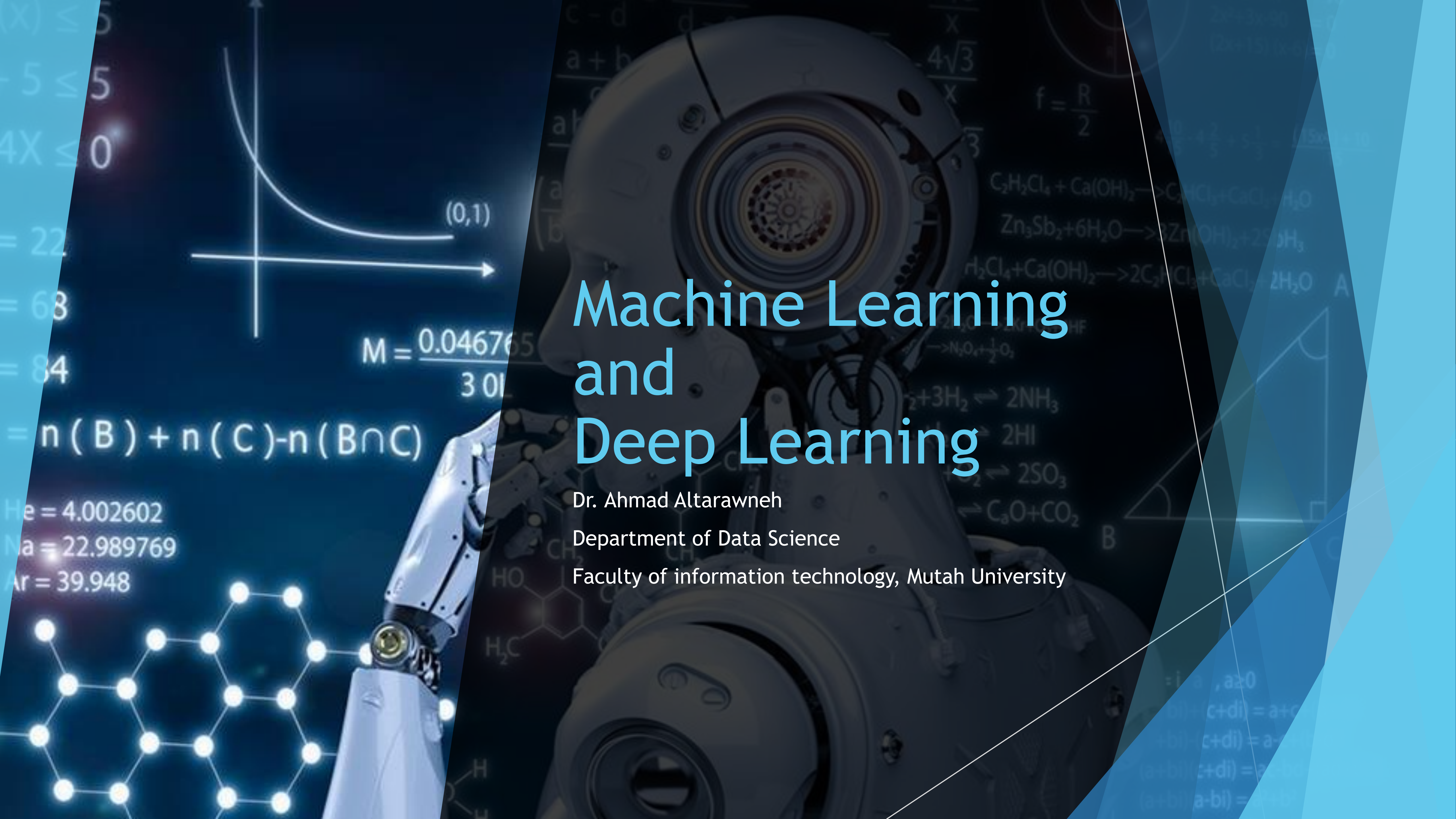


Machine Learning and Deep Learning

Dr. Ahmad Altarawneh

Department of Data Science

Faculty of information technology, Mutah University

Outline

- What is Machine Learning?
- Machine Learning types
  - Supervised learning
    - Classification
    - Regression
  - Unsupervised learning
  - Reinforcement learning
- ML applications

What is Machine learning

- Machine learning is a subset of artificial intelligence (AI) that involves the development of algorithms that allow computers to perform tasks without explicit programming; Learning from historical data

Cont.

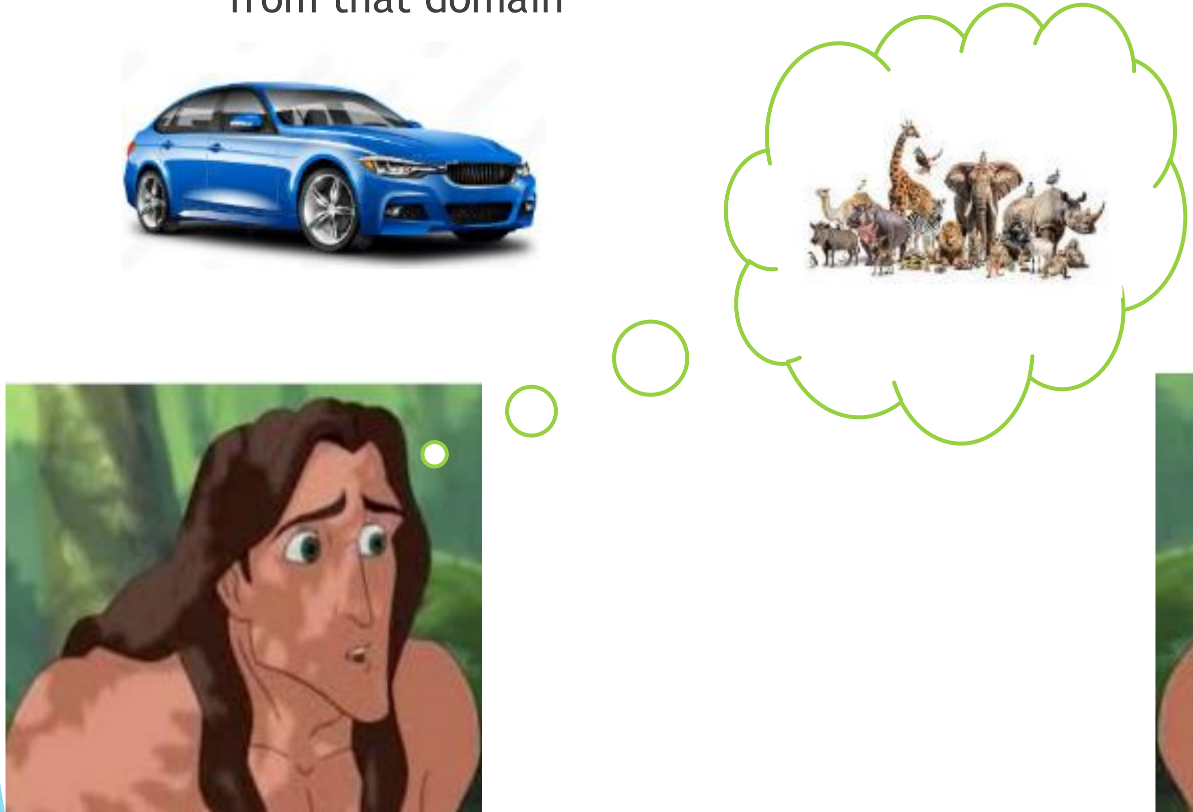
- If we show an object to a human and ask him, what is this object

Humans do this effortlessly if provided with the required experience: historical data.

It is a car

Cont.

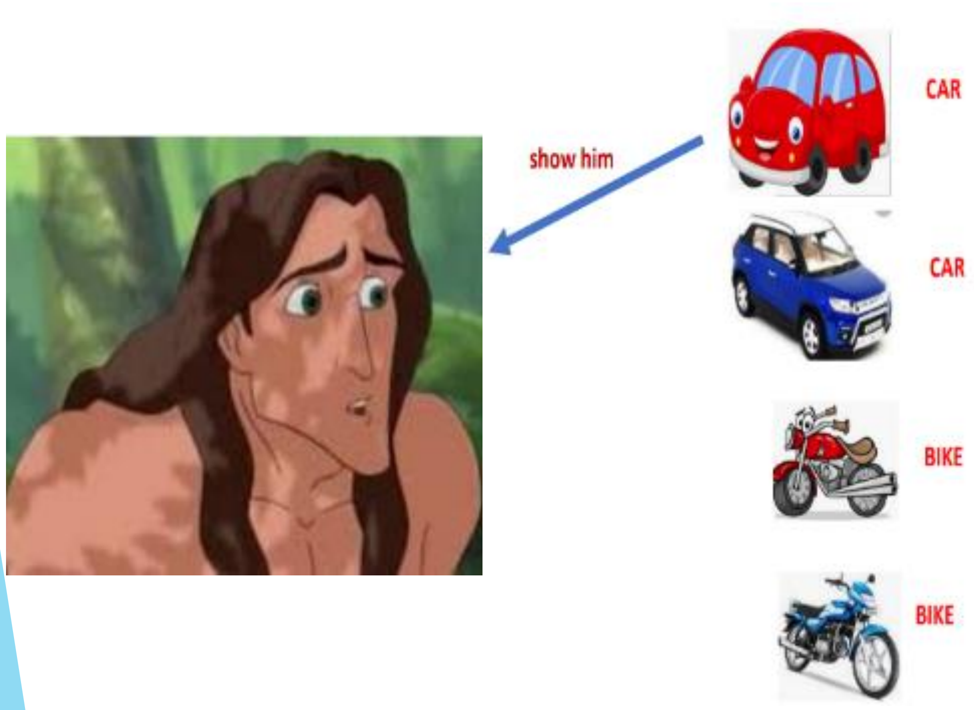
- What if humans do not have experience with a specific domain
  - They cannot answer questions, and cannot behave efficiently on a required task from that domain

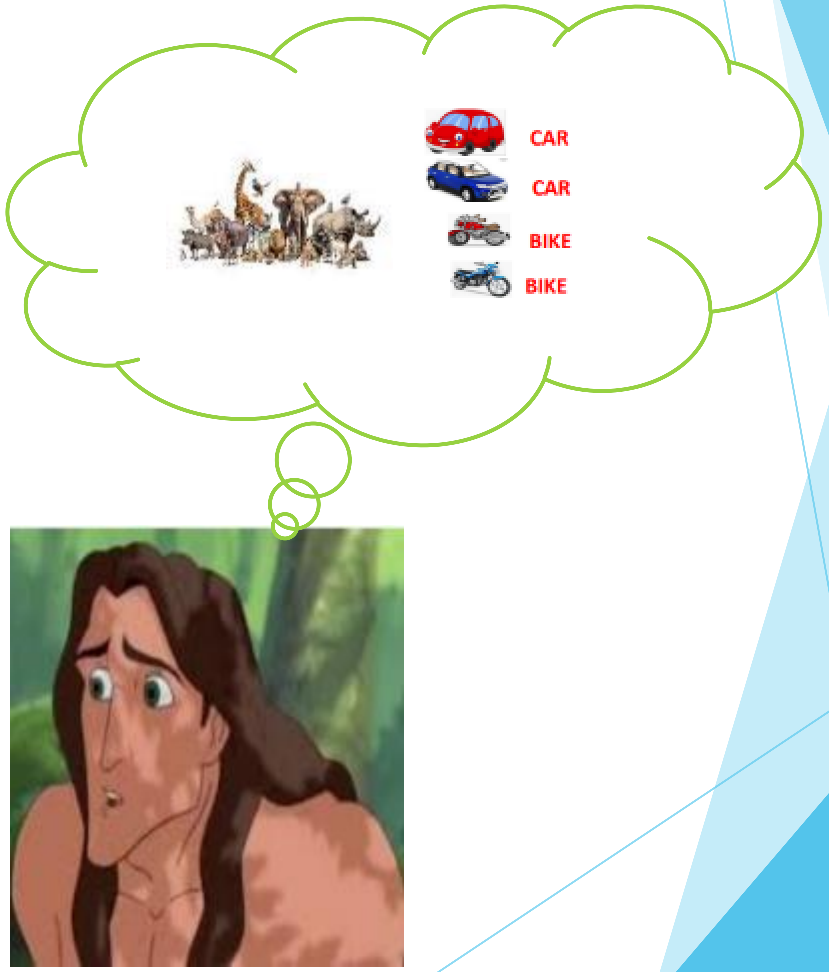


Never seen it before

Cont.

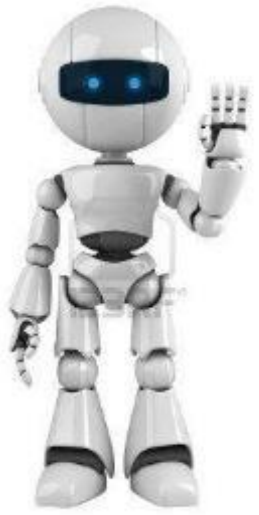
- However, humans can observe and learn





Cont.

- Can we do the same thing with the machine?
- Machines can perform basic operations, or explicitly programmed tasks
  - Addition
  - Subtraction
  - Comparison
  - Plotting charts

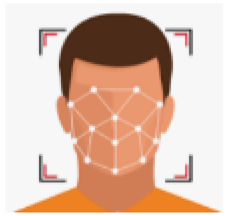


Cont.

- We want the machine to perform tasks like humans, without explicit programming
  - Identify objects
  - Predict prices
  - Correct grammar like experts
  - Recognize faces

16000$ in 2025

I ~~made~~ met him yesterday

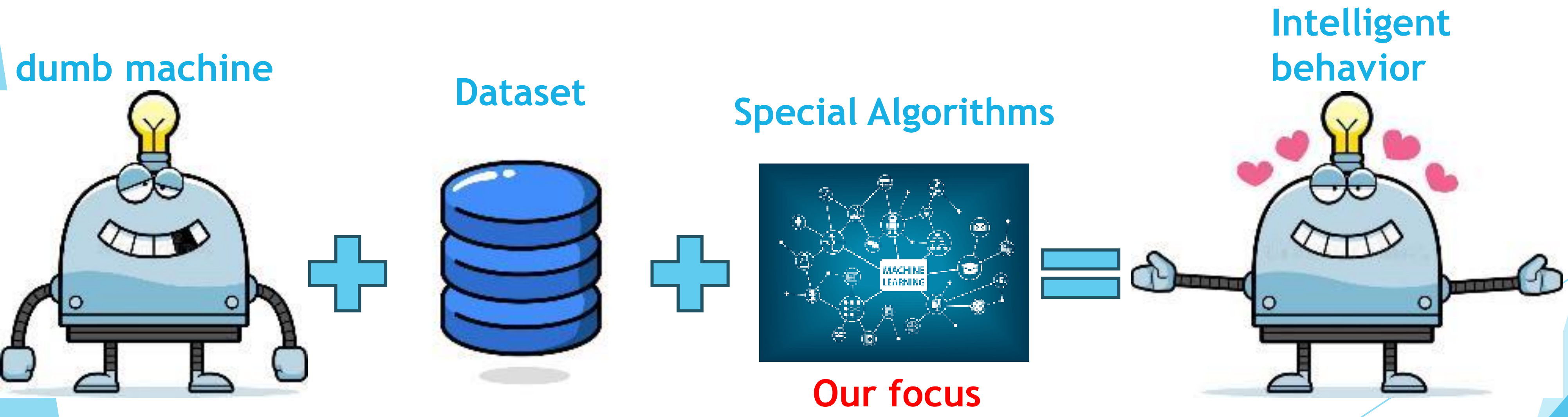


Cont.

- What do we do?

  Just like, what we did to humans.

  we need to provide experience to the machine.



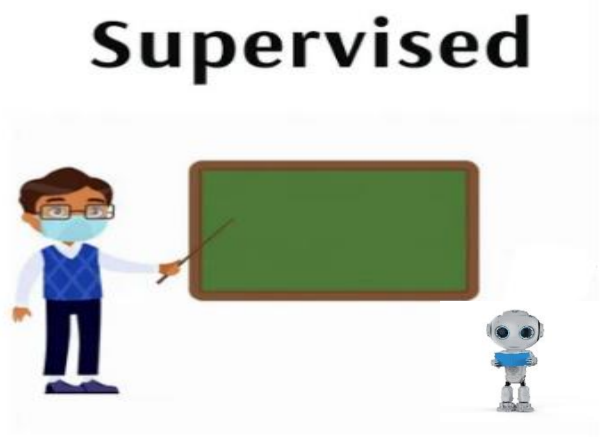
Cont.

- Given a machine learning problem
- Identify and create the appropriate dataset
- Perform computations to learn
- Output the decision

ML types

- Humans can solve different day-to-day life problems, where decisions need to be made
- Depending on the nature of the problem, machine learning tasks can be broadly divided into:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

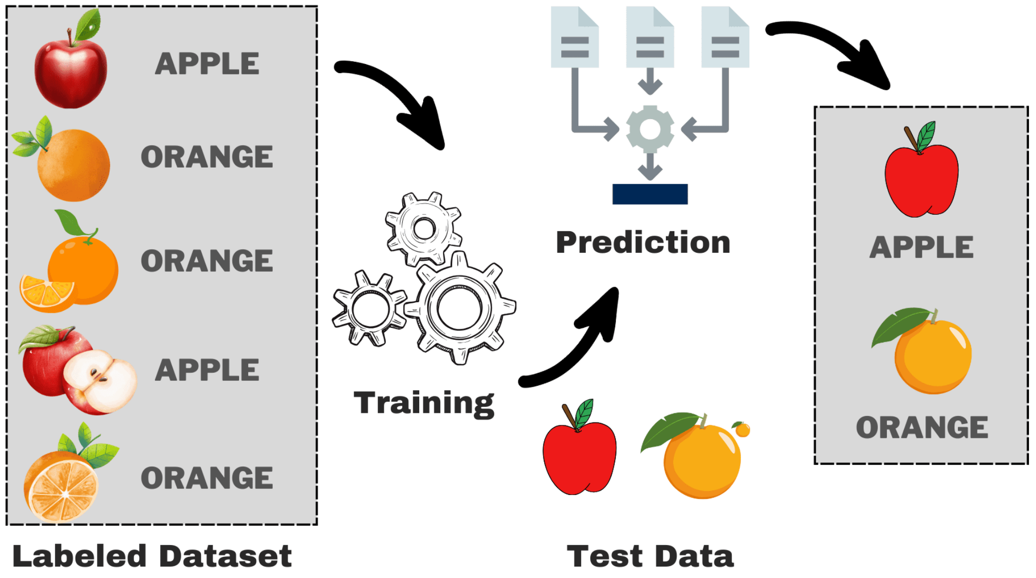
Supervised learning



- In this learning approach, there will be a supervisor (teacher, expert) in the learning process
- The supervisor shows examples along with their answers (labels)
- The machine tries to find relations between the input and the output provided by the expert
- During the evaluation the supervisor tests the machine on a new set of examples without providing the answers
  - The new examples belong to the same categories, on which the machine was trained
  - to test if the machine found the correct relations between the inputs and outputs given during the training
- Supervised learning can be divided into two types
  - Classification
  - Regression

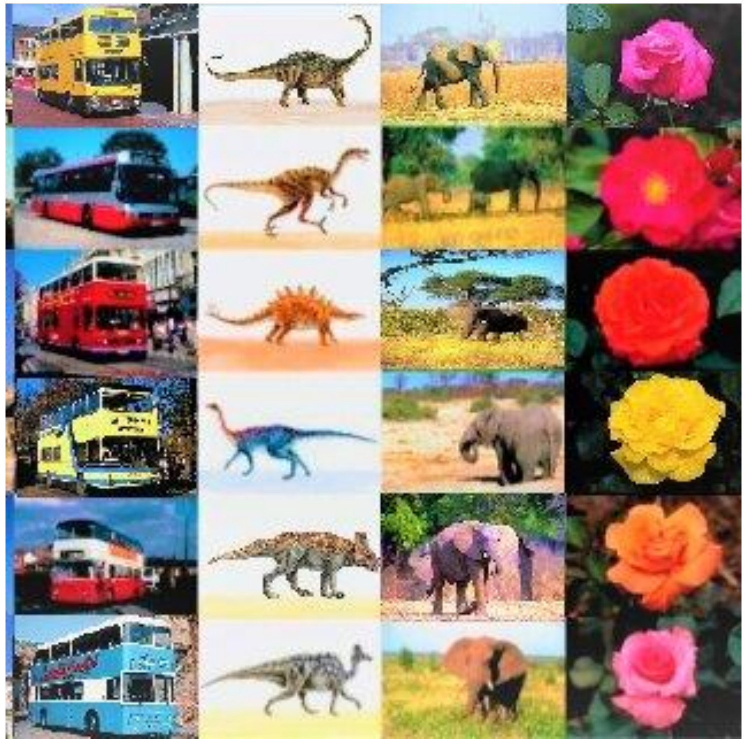
Classification

- In this type the answers provided to the machine during the training are integers or categories
  - Examples, 0,1,2,3 or Apple, Car, Face, Table, etc.
- The inputs can be any data: structured or unstructured
  - E.g., Images



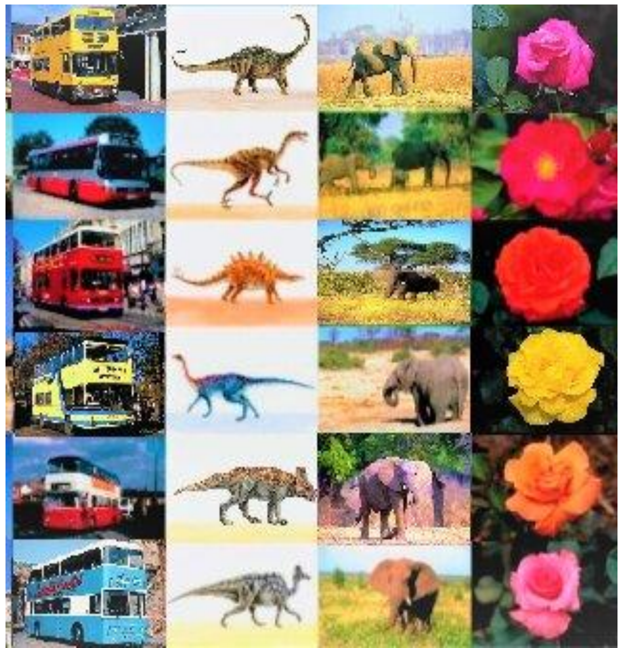
Cont. Classification

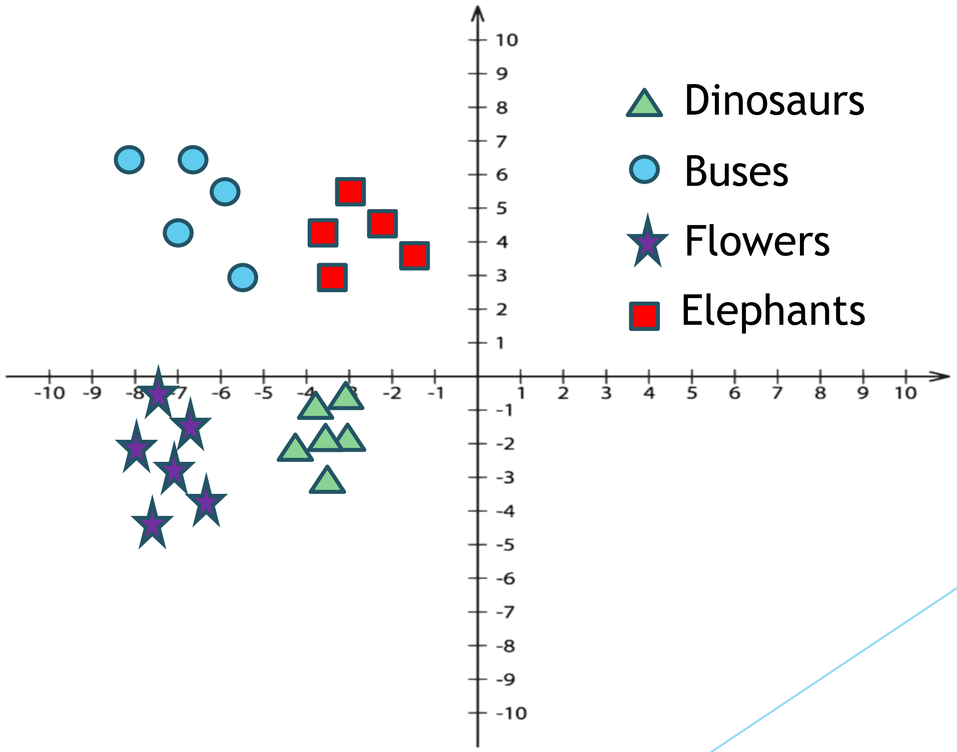
- One example of classification is object classification
- We want the machine to distinguish image content from each other
- The following figure has input examples that belong to 4 classes (labels)
  - Corel-1K dataset



Cont. Classification

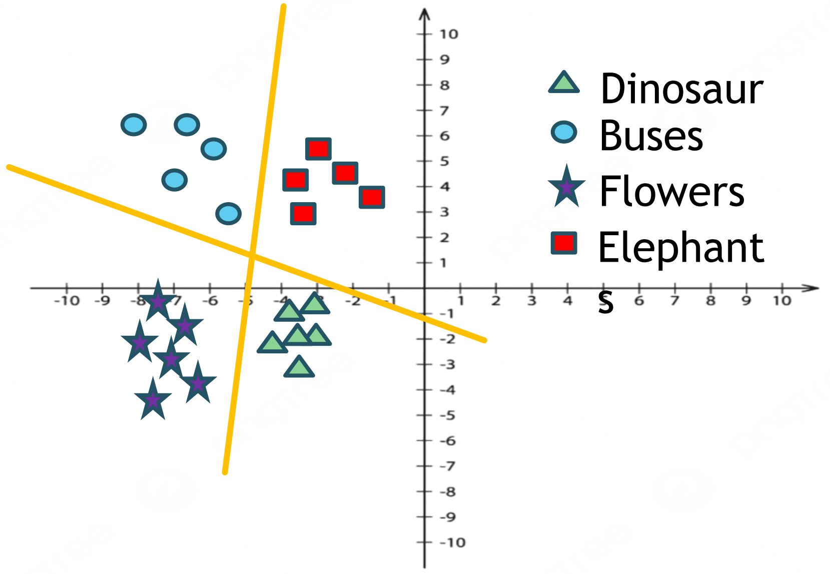
- Remember the pattern recognition flowchart diagram?
- These inputs might be processed to be converted to be digestible by the ML algorithms

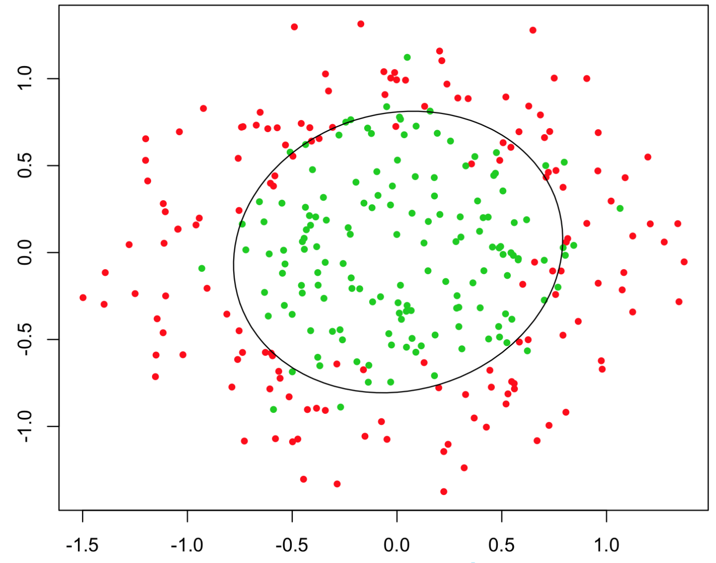




Cont. Classification

- The algorithms responsible for solving classification problems are called CLASSIFIERS
- They draw a kind of decision boundary to distinguish these classes from each other
- The decision boundaries are not necessarily linear



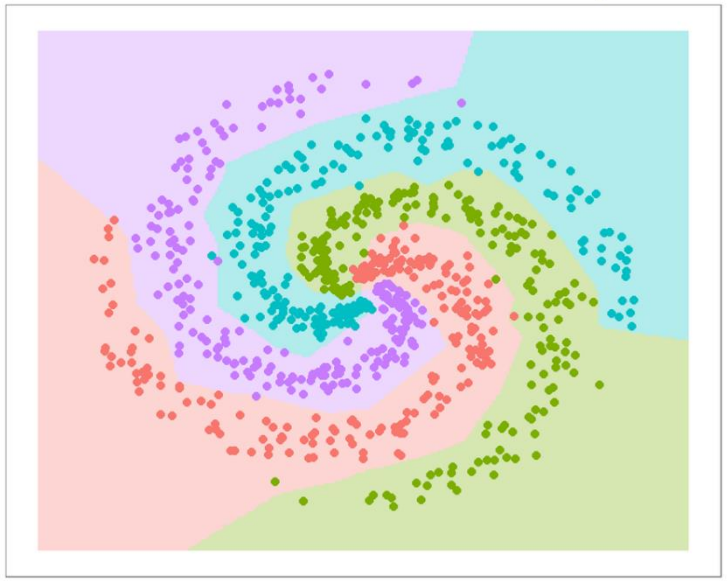


Binary classification with non-linear decision boundary

Cont. Classification

- When there are only two tables the classification is called binary classification problem
- The problems with more than two labels (classes or categories) are called multi-class classification problems
  - Some classifiers form very complex decision boundaries

Neural Network Decision Boundary

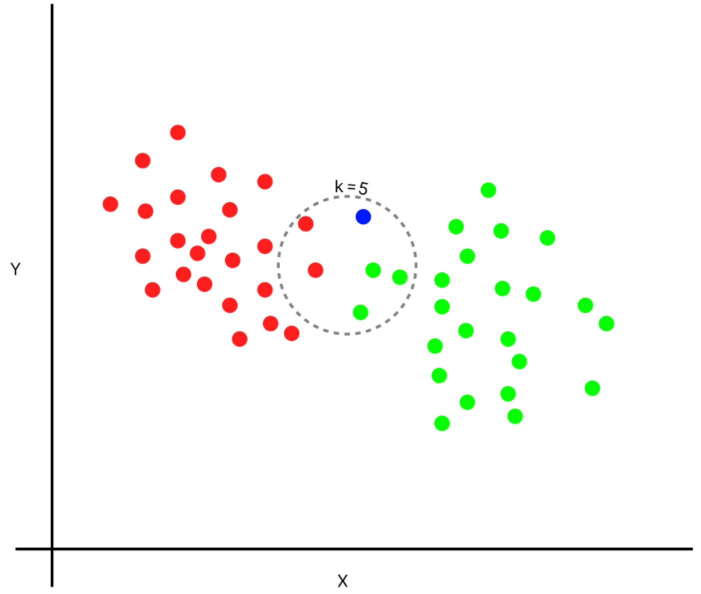


Cont. Classification

- Well-known classifiers which you have studied so far
  - K-nearest neighbors
  - Naïve Bayesian
  - Decision trees
  - Support vector machines (SVM)
  - Linear discriminant Analysis (LDC)
  - Perceptron: will be reviewed later in the ANN's chapter

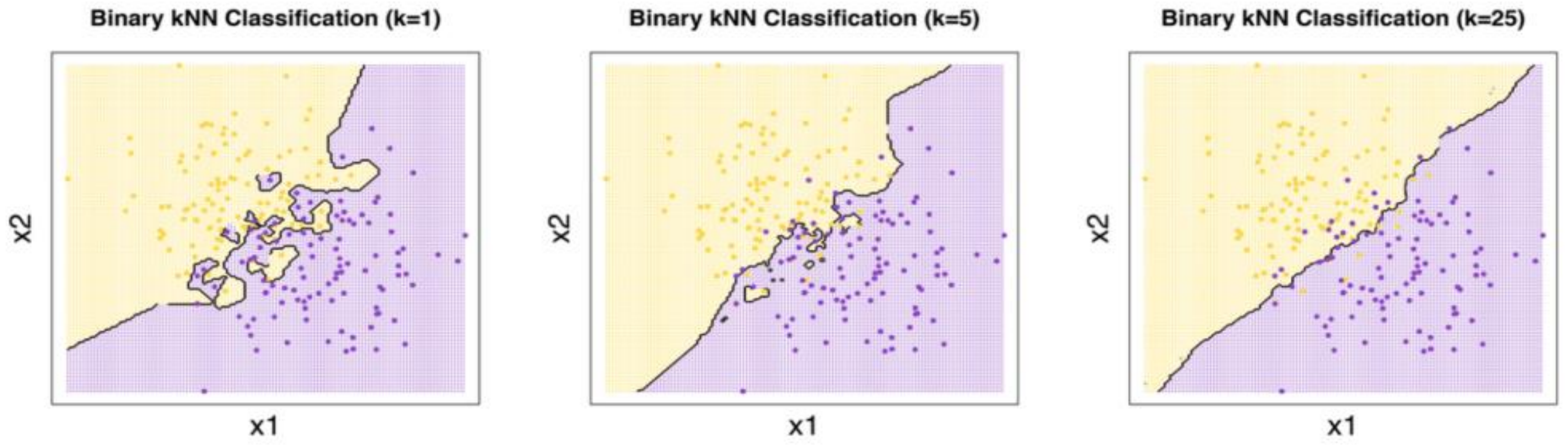
Cont. Classification
KNN

- KNN is one of the simplest and easiest to implement-classifiers
- It has no model as the model is the training data itself
- KNN looks for the closest example in the dataset from the new instance
    - A majority vote of the K closest examples in the training set determines the label of the new instance



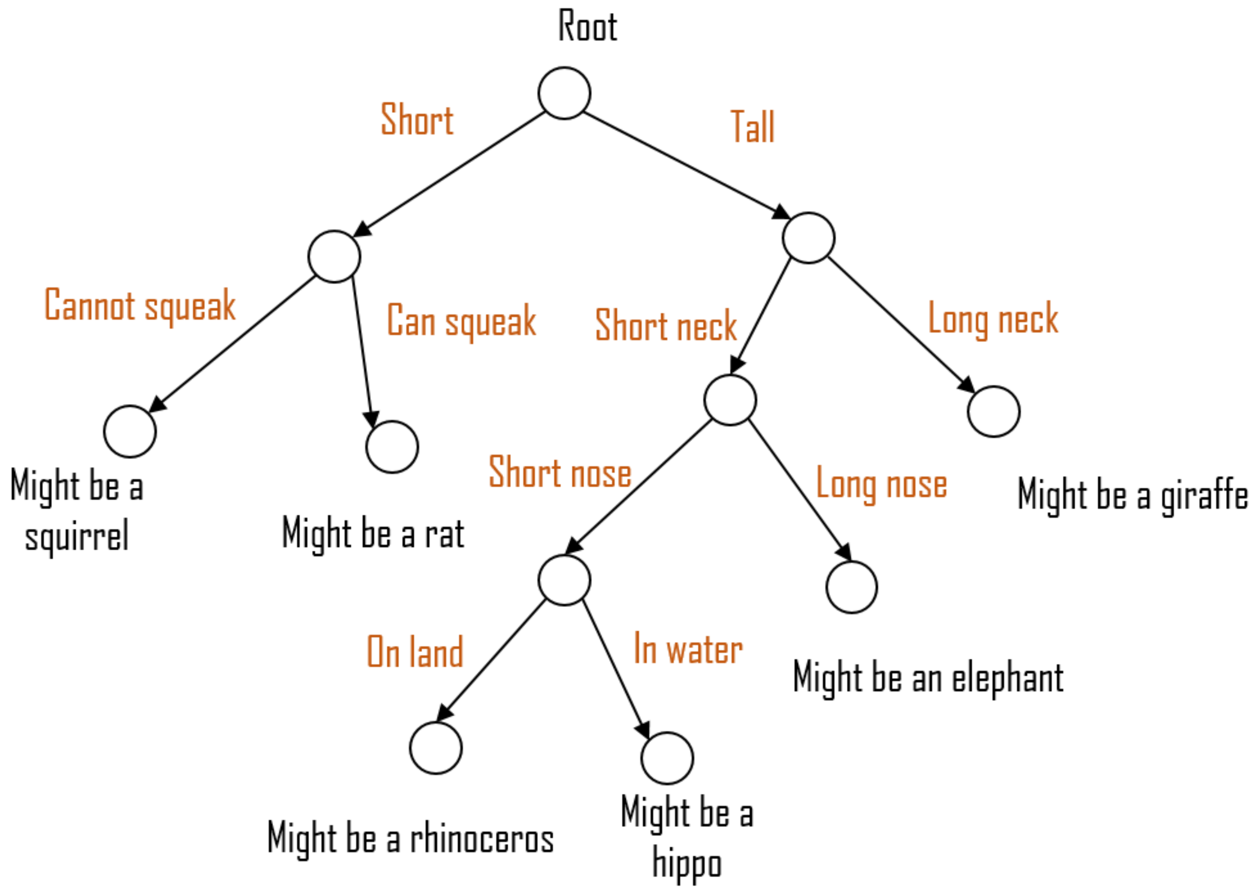
Cont. Classification
KNN

- The complexity of the decision boundary of KNN is determined by K and the distance metric used
  - Small K value leads to complex decision boundaries and potential overfitting
  - Large K can lead to smooth decision boundaries and potential underfitting, as the classification of a point is based on more distant neighbors
  - It is better to experiment with various distance metrics and K values 1, 3, and 5, etc. depending upon the size of the training data



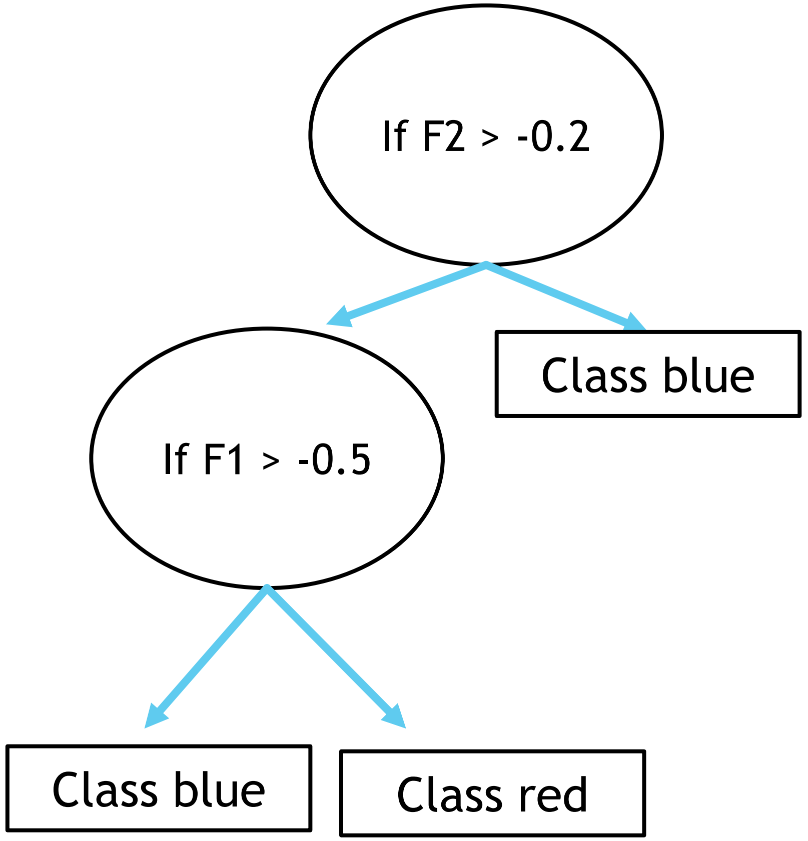
Cont. Classification
Decision trees

- Decision trees try to classify input data by forming a set of rules that control the decision boundaries in order to distinguish examples from different classes from each other
- Split the features that matter in the classification starting from the strongest features, the features that provide the best separability between classes
- *Information gain* or *Gini index* algorithms are examples of algorithms used to rank the feature for splitting

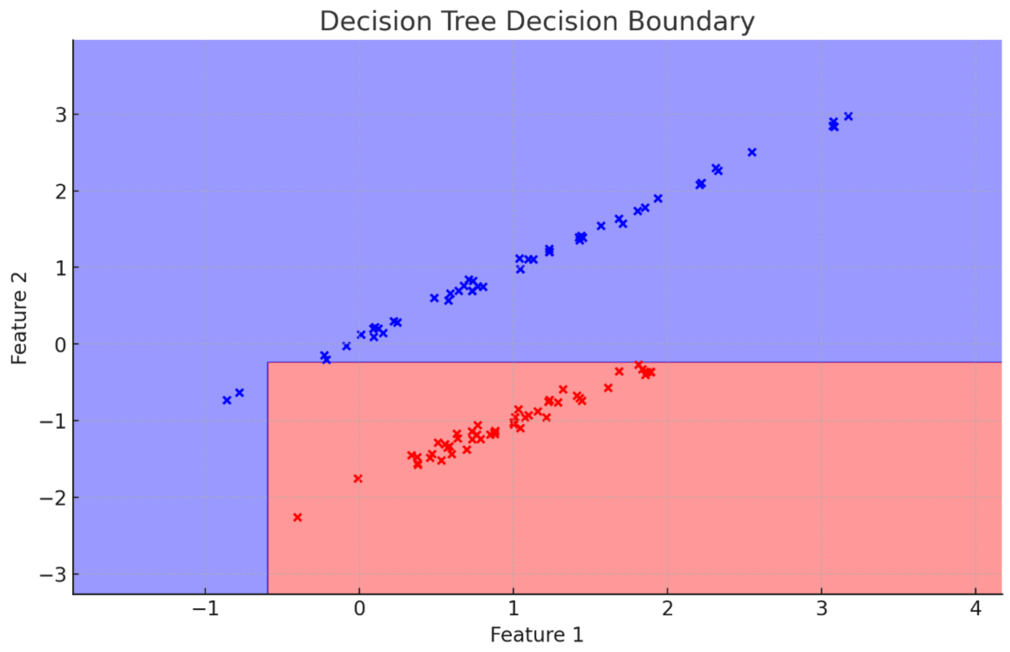


Cont. Classification
Decision trees

- See the following example

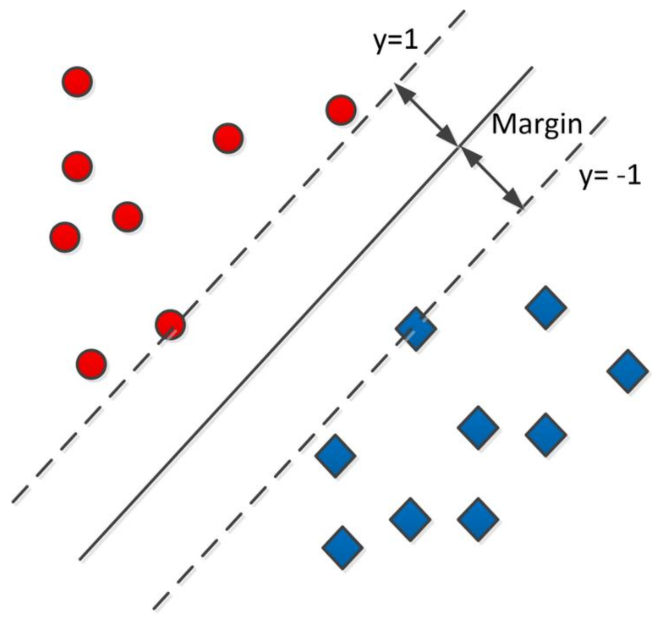


The decision boundaries of the DT are always axis-aligned



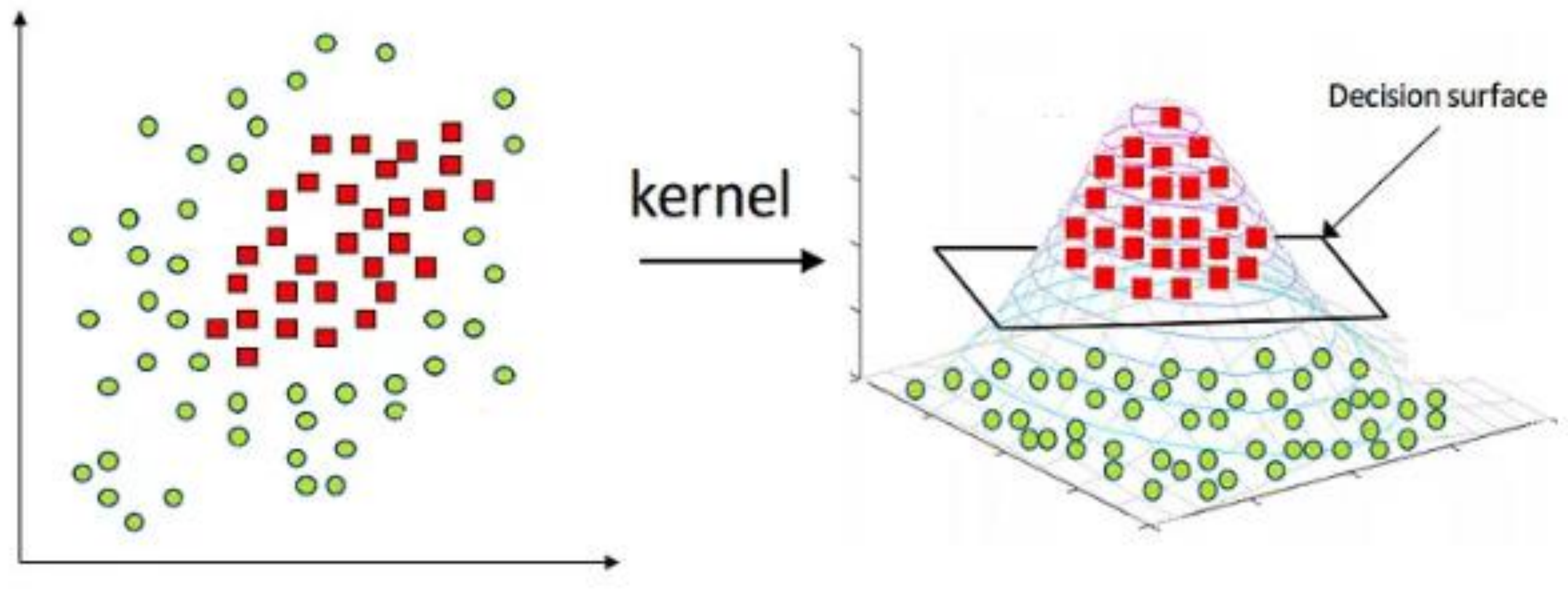
Cont. Classification
SVM

- Support vector machines are the choice when we want to maximize the margin between the classes
- It provides a better generalization



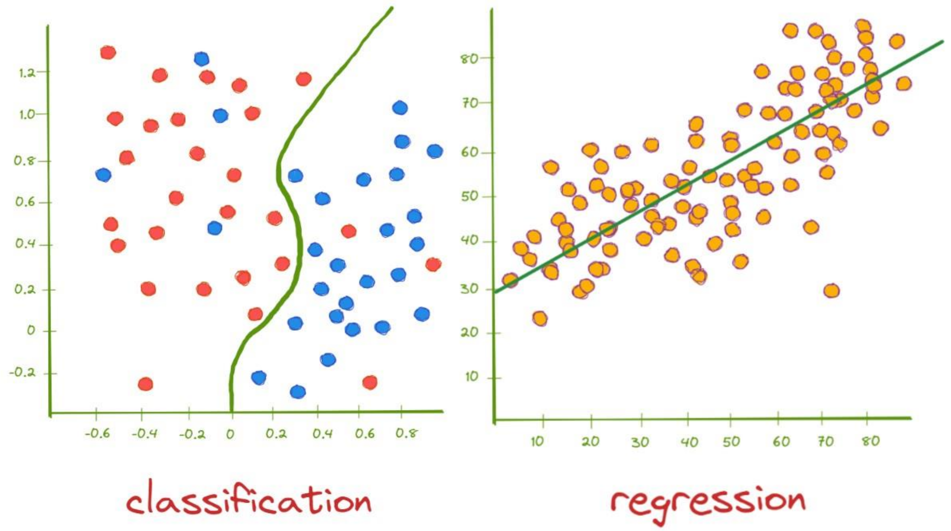
Cont. Classification
SVM

- SVM can solve non-linear classification tasks by the help of kernels



Regression

- Regression tasks differ from classification tasks in the type of output
- In classification the output is categorical, while in regression the output is real (continuous values)
  - E.g., prices, heights, weights, etc.
- In this kind of problem, the model aim is to fit the data points instead of separating them
  - The algorithms work on this kind of problem are called REGRESSORS



Regression
Linear regression

- Linear regression is the simplest method for regression as it tries to fit a line to the data to use it for predicting future data
- The line is identified by slope (m or w sometimes) and intercept (b)

$$y = mx + b$$

- These parameters can be calculated using the following formula

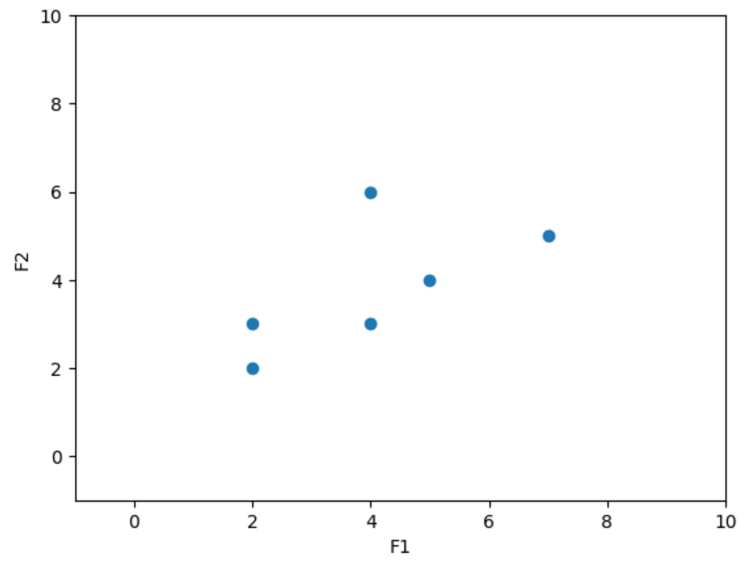
$$m = \frac{n\sum(xy) - \sum(x)\sum(y)}{n\sum(x^2) - (\sum(x))^2}$$

$$b = \frac{\sum(y) - m\sum(x)}{n}$$

Regression
Linear regression

- As an example, see the following points

```
F1 = [4,2,5,4,7,2]
F2 = [3,2,4,6,5,3]
```



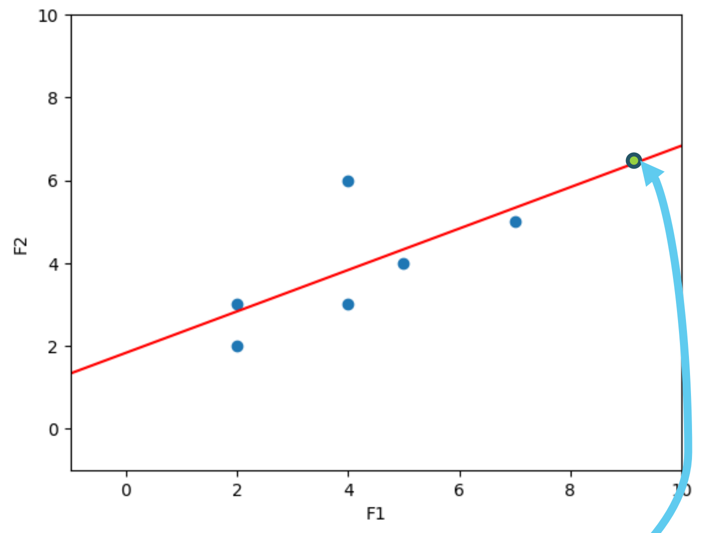
- The question is what is the model (line in linear regression) that fits these data points

Regression

Linear regression

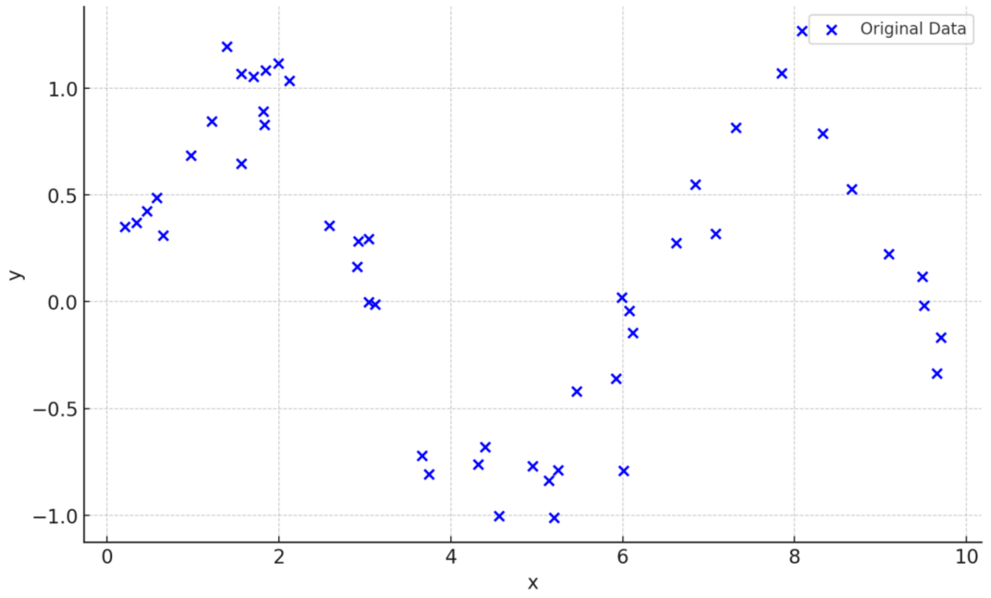
- By applying the equations of linear regression, we find
  - m = 0.5 and b = 1.8
  - so, the model is $y = 0.5x + 1.83$
  - let us fit it on the data and see how it looks like
- we can use this model to predict future points
  - what is the F2 value when F1 is 9

$$F2 = 0.5 * 9 + 1.83$$
$$F2 = 6.33$$



Regression

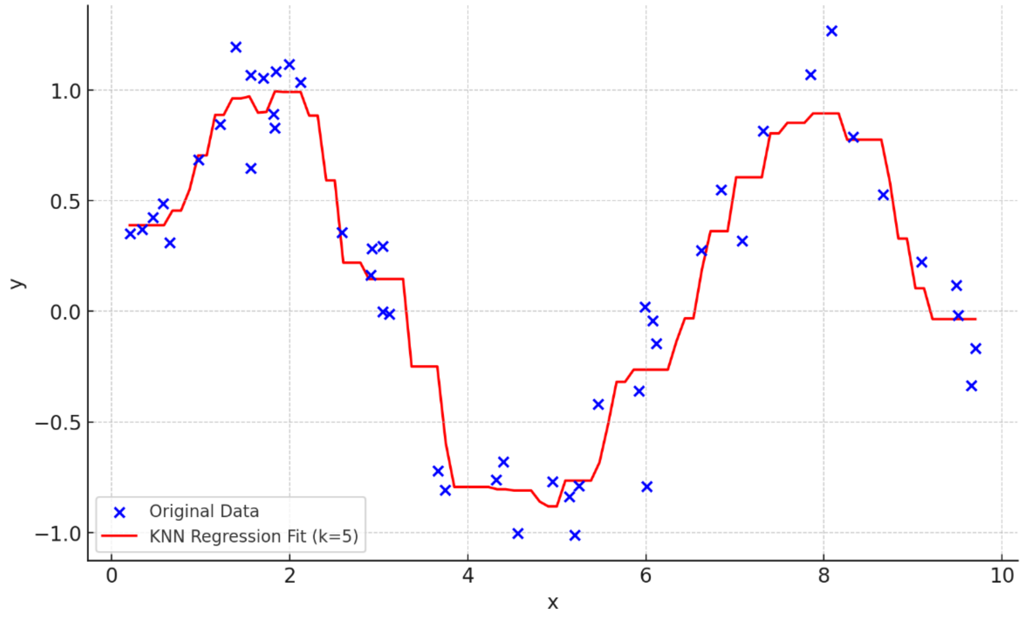
- However, the data might and will not always be linear!
- What is the solution when the regression problem has nonlinear relation



Use non-linear regressors

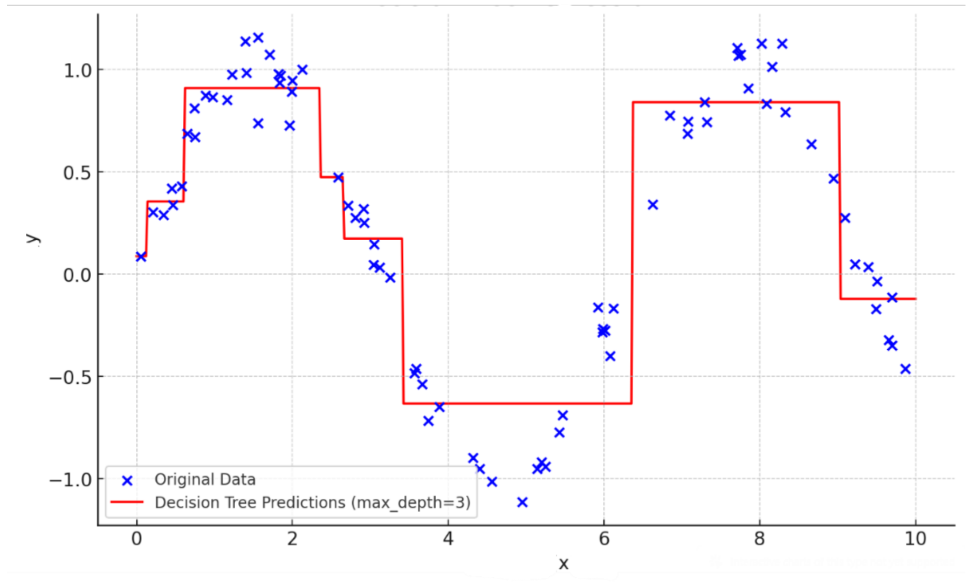
Regression
KNN regressor

- Knn regressor works similarly to knn classifier
- However, instead of taking the major class of the K closest points, we take the average of the labels of the K closest points



Regression
Decision tree regressor

- DT can be used as a regressor as well, check this
  - https://www.youtube.com/watch?v=UhY5vPfQIrA
- The split and impurity measures are based on the variance



More regressors
Polynomial

- **Polynomial regression** is a method used for modeling **non-linear relationships** by fitting a **linear model** to a **transformed version of the input data.** In polynomial regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x$$

- a new set of features is created for each sample based on the degree of the function
- then we solve a linear system of equations to find the betas

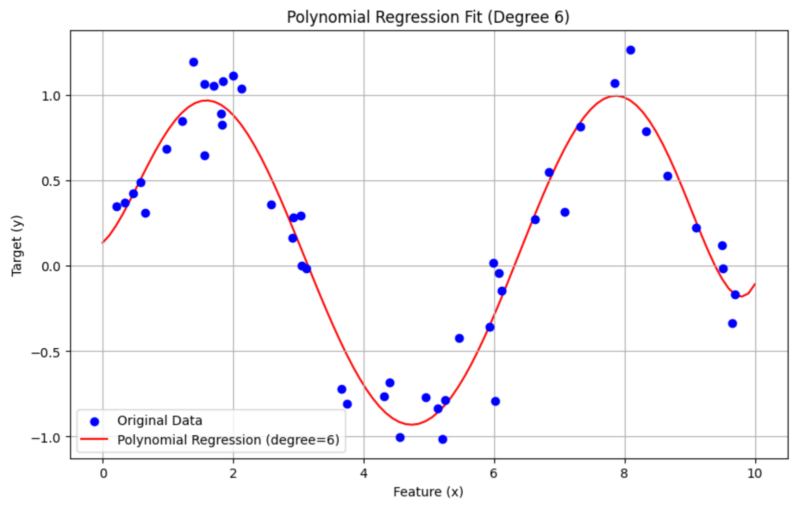
More regressors

Polynomial

- Beta vector contains the coefficients, sometimes it is called weights (w) vector
- they can be found using the following formula

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

  - X is the matrix of transformed polynomial features,
  - y is the vector of actual target values,
  - w = $[\beta_0, \beta_1, \beta_2, \ldots, \beta_d]$ is the vector of coefficients.



```python
import numpy as np
import matplotlib.pyplot as plt


np.random.seed(42)
x = np.sort(np.random.rand(50) * 10).reshape(-1, 1)
y = np.sin(x).ravel() + np.random.normal(0, 0.2, x.shape[0])

def generate_polynomial_features(x, degree):
    n_samples = x.shape[0]
    X_poly = np.ones((n_samples, degree + 1))

    for d in range(1, degree + 1):
        X_poly[:, d] = x.ravel() ** d

    return X_poly

def fit_polynomial_model(X_poly, y):
    w = (X^T * X)^-1 * X^T * y
    XTX = np.dot(X_poly.T, X_poly)
    XTy = np.dot(X_poly.T, y)

    # Compute coefficients (w)=> betas
    w = np.linalg.inv(XTX).dot(XTy)
    return w

def predict_polynomial(X_poly, w):
    return np.dot(X_poly, w)
```

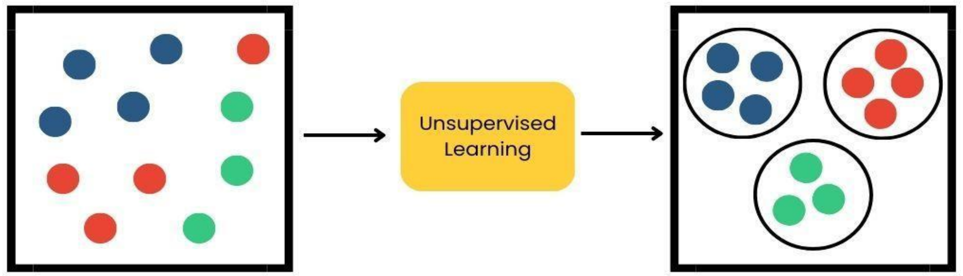
```python
degree = 3

X_poly = generate_polynomial_features(x, degree)
w = fit_polynomial_model(X_poly, y)

x_fit = np.linspace(0, 10, 100).reshape(-1, 1)
X_fit_poly = generate_polynomial_features(x_fit, degree)
y_poly_pred = predict_polynomial(X_fit_poly, w)

plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='blue', label='Original Data',
zorder=2)
plt.plot(x_fit, y_poly_pred, color='red',
label=f'Polynomial Regression (degree={degree})', zorder=1)

plt.xlabel('Feature (x)')
plt.ylabel('Target (y)')
plt.legend()
plt.grid(True)
plt.title(f'Polynomial Regression Fit (Degree {degree})')
plt.show()
```

More regressors
ANN

- ANN is another method used to solve regression problems
- We will talk about it heavily in later chapters

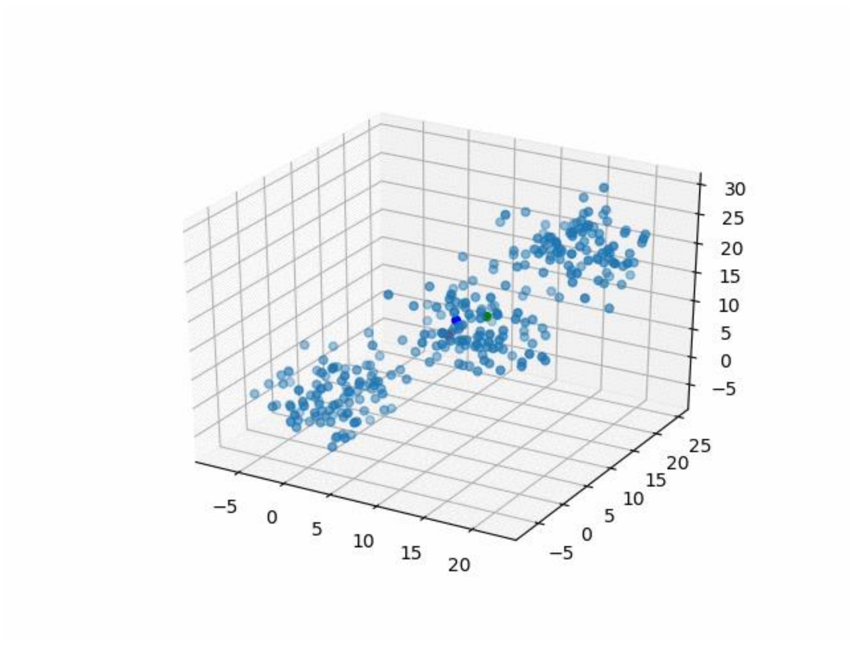
Unsupervised learning

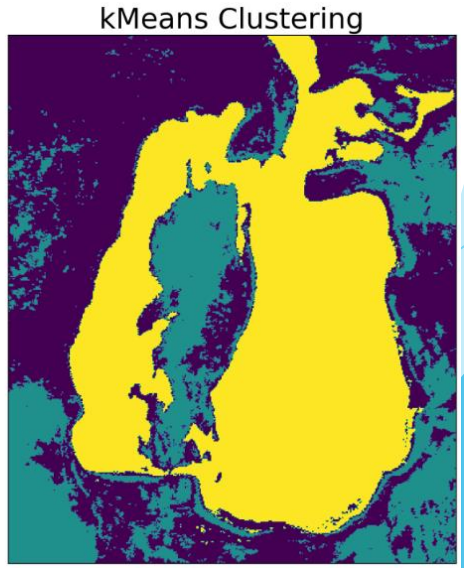
- In unsupervised learning the task is to find relations between samples, without teacher, supervisor
  - There is no labels
- The machine tries to label these examples based on some criterion
  - For example, the similarity between features of different examples indicates that these examples should have the same labels
- Example group the images with similar content together



Unsupervised learning
K-means: recalling

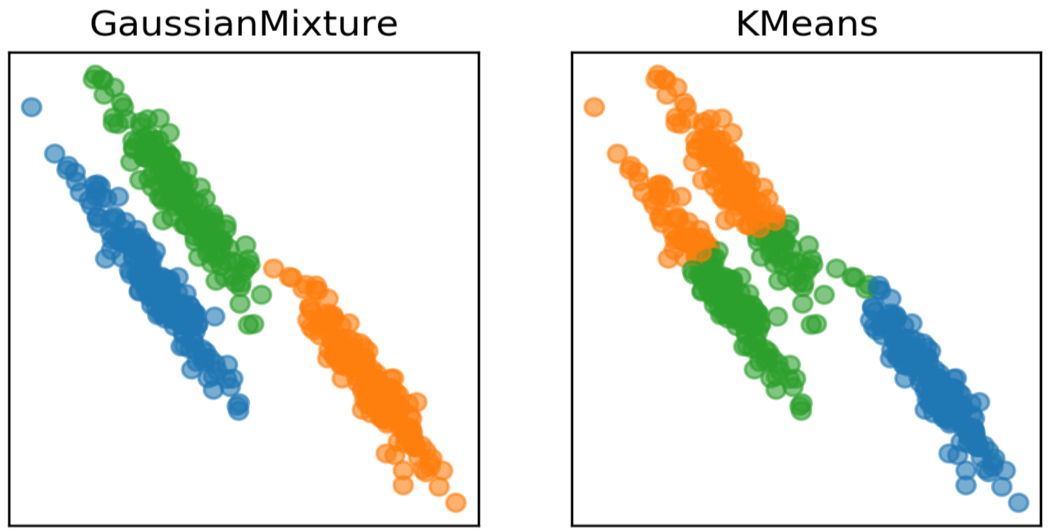
- K-means is the most popular algorithm for clustering
- It tries to group the examples that are close to each other in the same group
- there will be K groups, each of which has instances similar to each other





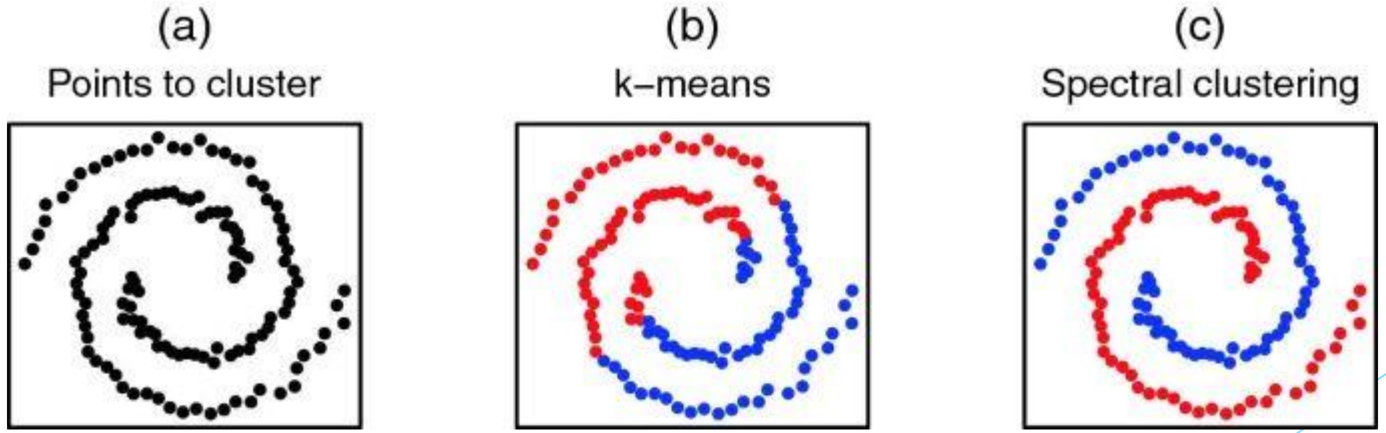
Unsupervised learning
Gaussian Mixture Models (GMM)

- K-means decision boundaries are hard and cannot fit data with more complex distributions
- two parameters control the GMM:
  - Mean: controls the location, center, of the model
  - Covariance matrix (Variance in 1d): controls the **shape**, **size**, and **orientation** of the Gaussian distribution, essentially defining its **spread**



Unsupervised learning
Spectral clustering

- Spectral clustering is effective for non-normally distributed or non-convex clusters.
- It first constructs a similarity graph, then computes the eigenvectors corresponding to the smallest non-zero eigenvalues of the graph Laplacian matrix.
- The data points are then represented by the rows of these eigenvectors
- A clustering algorithm like K-Means is applied to group the points in this transformed space.



Unsupervised learning
Spectral clustering

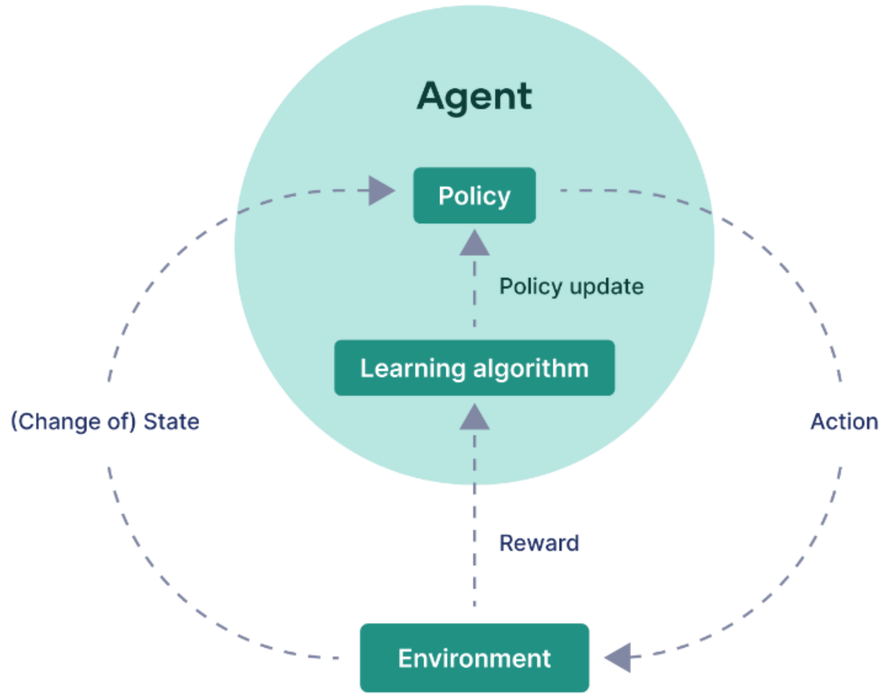
- Despite the power of spectral clustering, it has some serious limitations:
  - Scalability to a large dataset: spectral clustering can be applied to datasets with a small number of samples, e.g., a few thousand samples
  - The time and space complexity of the similarity matrix is high
  - The **eigenvectors** used in spectral clustering come from the **Laplacian matrix**, which may not have a direct or intuitive interpretation with respect to the original data

Reinforcement learning

- **Reinforcement Learning (RL)** is a type of machine learning where an **agent** learns to make decisions by interacting with an **environment.**
- The agent takes actions in different situations (called **states**) and receives **rewards** or **penalties** as feedback.
- The goal of the agent is to learn a **policy** that maximizes the cumulative reward over time by balancing exploration (trying new actions) and exploitation (choosing the best-known action)

Reinforcement learning

- **Reinforcement Learning** is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment and receiving feedback (rewards or penalties). The agent's objective is to maximize the cumulative reward by learning an optimal policy through trial and error



Reinforcement learning

- **Key Elements of RL:**
  1. **Agent**: The learner or decision-maker.
  2. **Environment**: The external system with which the agent interacts.
  3. **State**: The current situation or configuration of the environment.
  4. **Action**: The decision or move the agent makes.
  5. **Reward**: Feedback from the environment in response to the agent's action, indicating success or failure.
  6. **Policy (π)**: A strategy or rule that the agent follows to choose actions based on states.
  7. **Value Function**: The expected cumulative reward from a given state or state-action pair.
  8. **Exploration vs. Exploitation**: Balancing between trying new actions to discover more (exploration) and using known actions to get rewards (exploitation)

Reinforcement learning
Q-learning

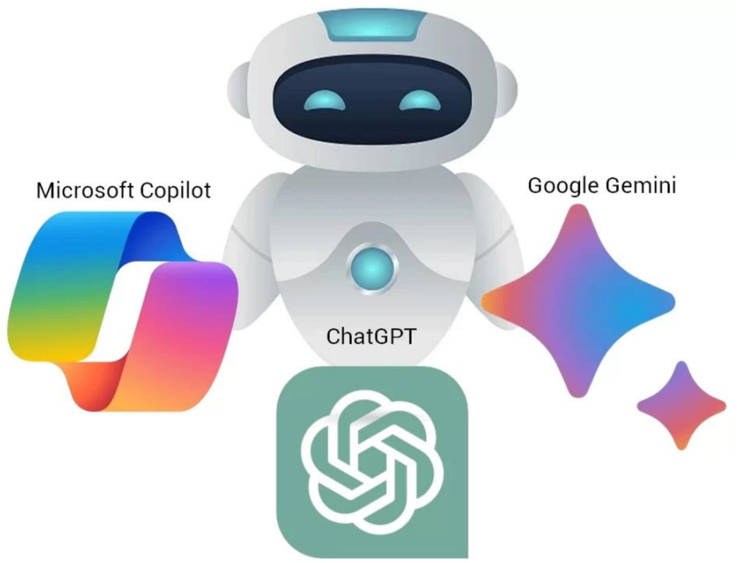
- **Q-Learning** is one of the most popular **model-free reinforcement learning** algorithms.
- It is used to find the optimal action-selection policy for any given finite set of states and actions
  - The "Q" stands for **quality**, which represents how useful a given action is in gaining future rewards.

Reinforcement learning
Neuroevolution

- **Neuroevolution (NE)** is somewhat different from reinforcement learning (RL), but it can still be applied to solve reinforcement learning tasks.
- In reinforcement learning (RL), the agent typically learns using gradient-based algorithms, such as Q-Learning or Policy Gradients, to update its policy based on the rewards it receives from interacting with the environment.
- In contrast, in **Neuroevolution (NE)**, the agent learns by evolving a population of neural networks.
  - Instead of using gradients, NE uses genetic algorithm to modify the characteristics (Brain, Neural Network) of the **best-performing agents** from a population, based on their **fitness** (rewards or performance in the environment).
  - The fittest agents are selected to produce new agents (brains) for the next generation, improving performance over time.

ML applications

- There are plenty of applications that rely on ML to perform
- For example, **Chatbots & Virtual Assistants**:
- These models are heavily trained on a huge amount of data using complex learning algorithms
    - Transformers

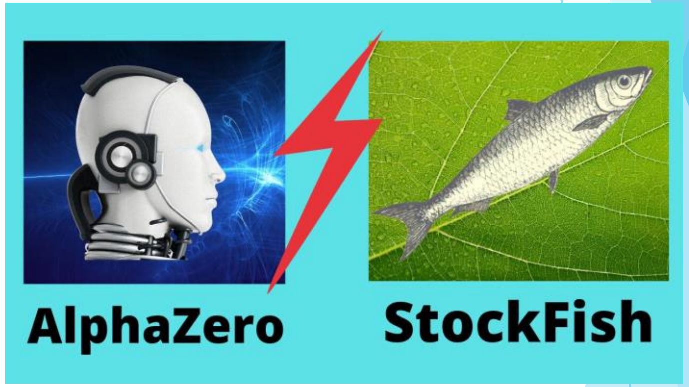


ML applications
Translation

- Translating long paragraphs, documents, or websites between languages while maintaining fluency
- Such models are trained on vast number of scripts translated to and from different languages
- Such applications use special kind of deep learning algorithms to perform:
  - Transformers
  - Recurrent neural networks (RNN) => Long-short-term memory (LSTM)
  - Hybrid deep models => LSTM with attention or LSTM with CNN

ML applications
Game playing

- Agents are trained to play games using neuroevolution and reinforcement learning algorithms
  - AlphaZero uses reinforcement learning to learn chess
    - Unlikely to be defeated by any expert human
  - **OpenAI Five (reinorcment learning)** plays Dota2 at different level
  - defeated 5 world champions (OG team) in a 5vs5 game
    - https://www.youtube.com/watch?v=pkGa8ICQJS8



ML applications

Robotics

- Robots learn how to walk using ML, more specifically reinforcement learning methods
  - Deep Q-Learning (DQN)

- Military applications anti-drawn agents
  - Sthir Stab 640
  - remember the black-mirror episode, Metalhead?

Ending

- Vast Applications: AI and ML are transforming industries, from healthcare to finance, and beyond.
- Understanding the Fundamentals: By learning the basics of ML, you gain insight into how these technologies work and impact our daily lives.
- Achievable Mastery: These innovations aren’t magic; with effort and the right tools, you too can contribute and even surpass current advancements.
- Your Potential: With continuous learning and dedication, you'll be able to create advanced models capable of driving groundbreaking solutions and innovations