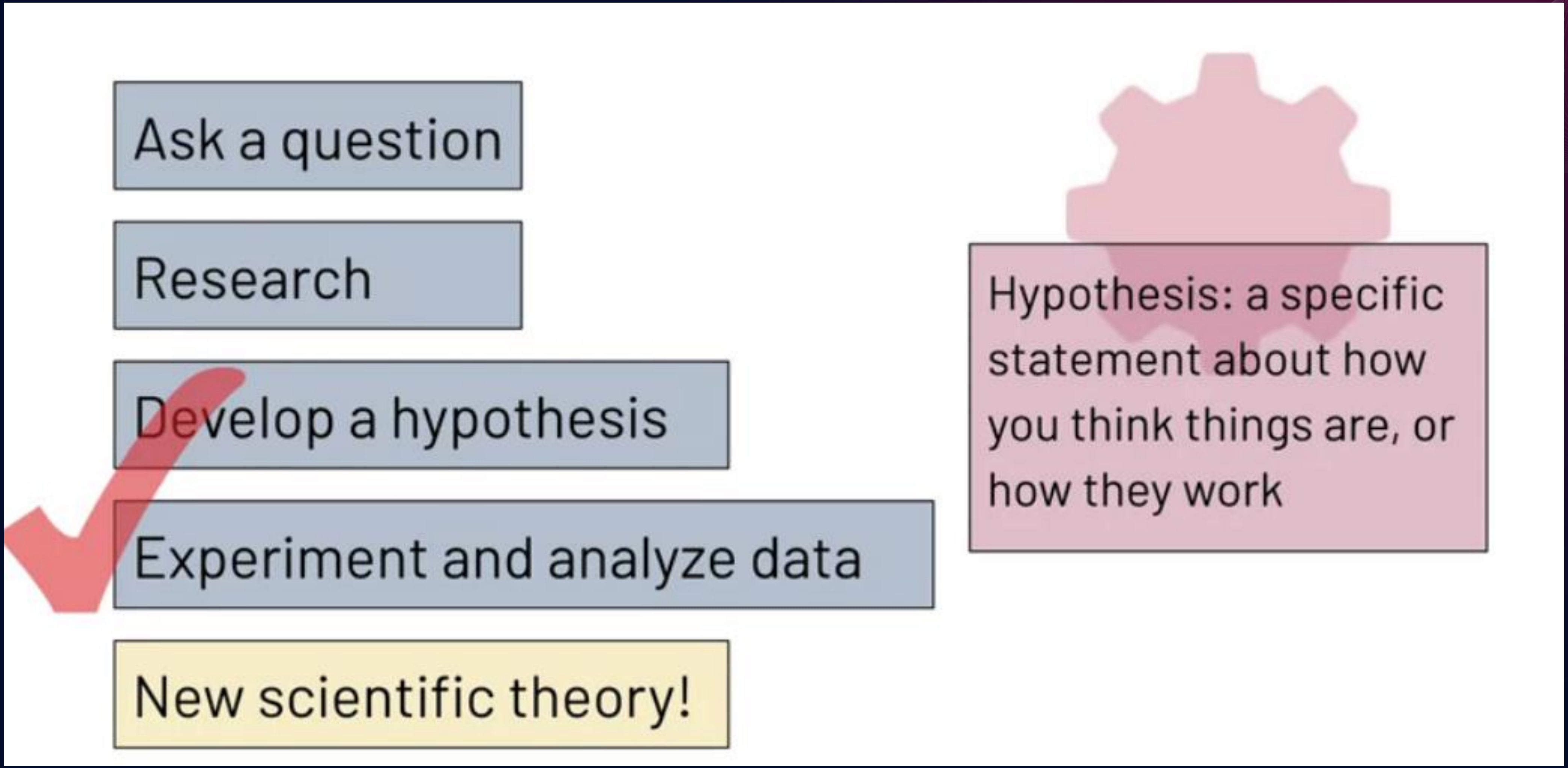
CS 156:Introduction to Artificial Intelligence

Instructor: Dr. Sayma Akther

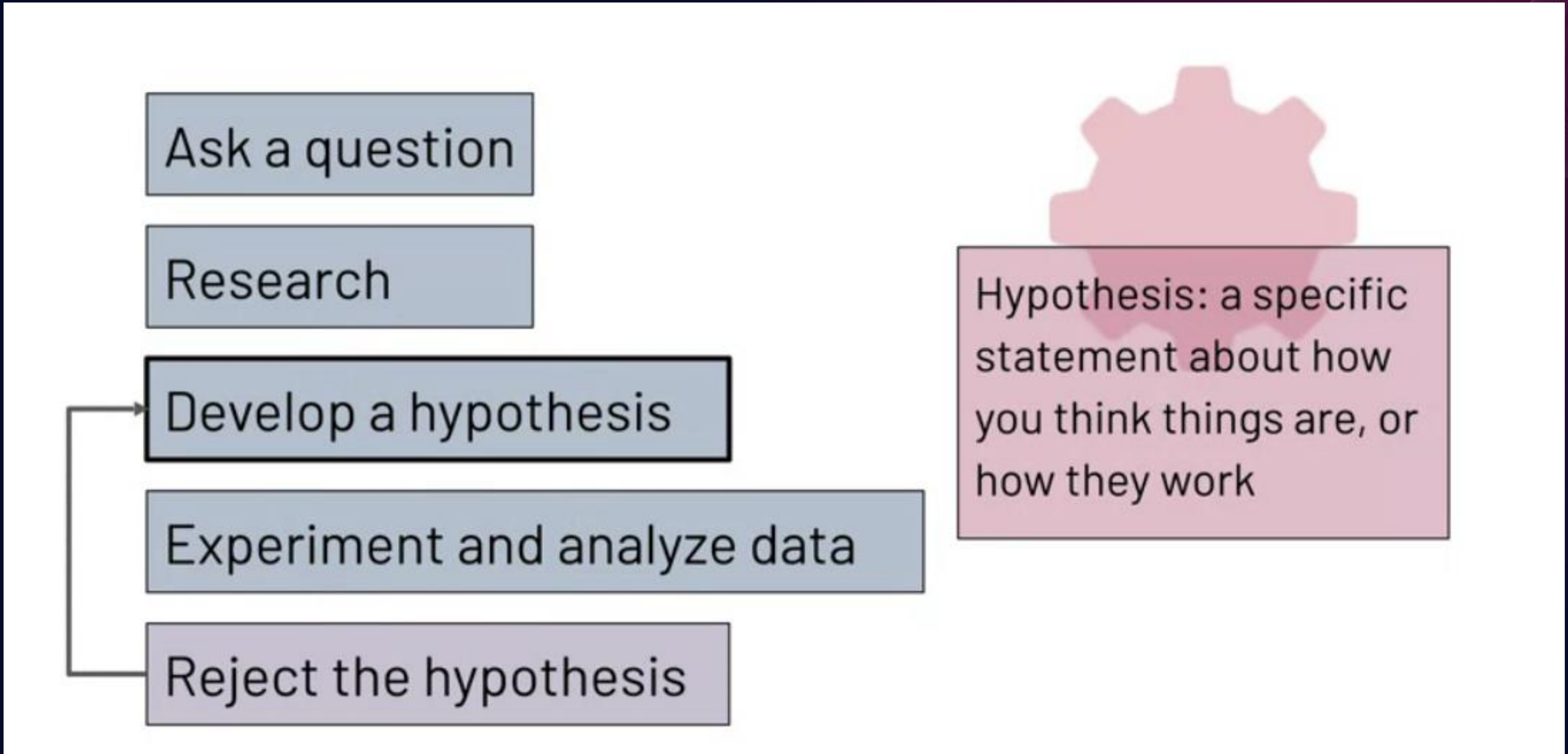
San José State University

Classification Models in Machine Learning



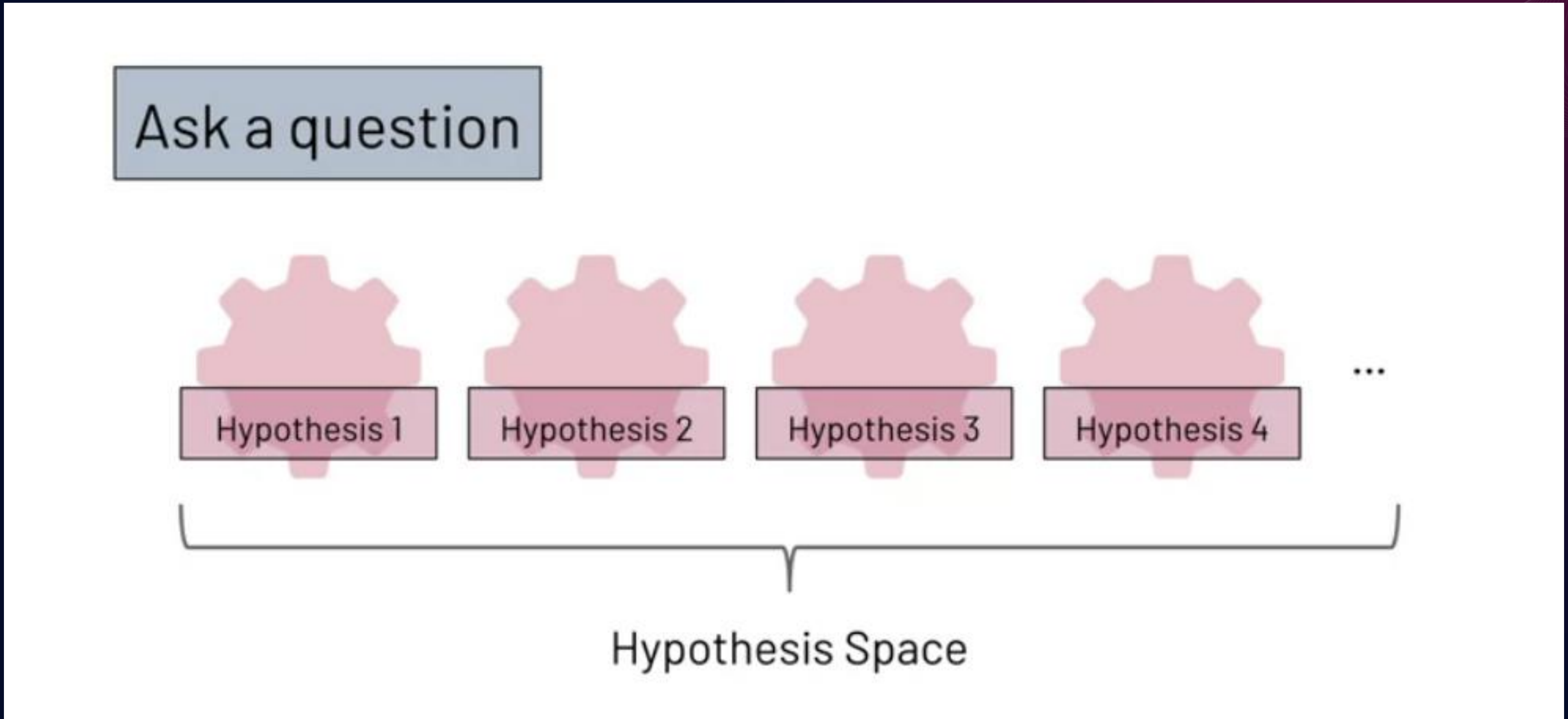
slide~superdatascience

Classification Models in Machine Learning

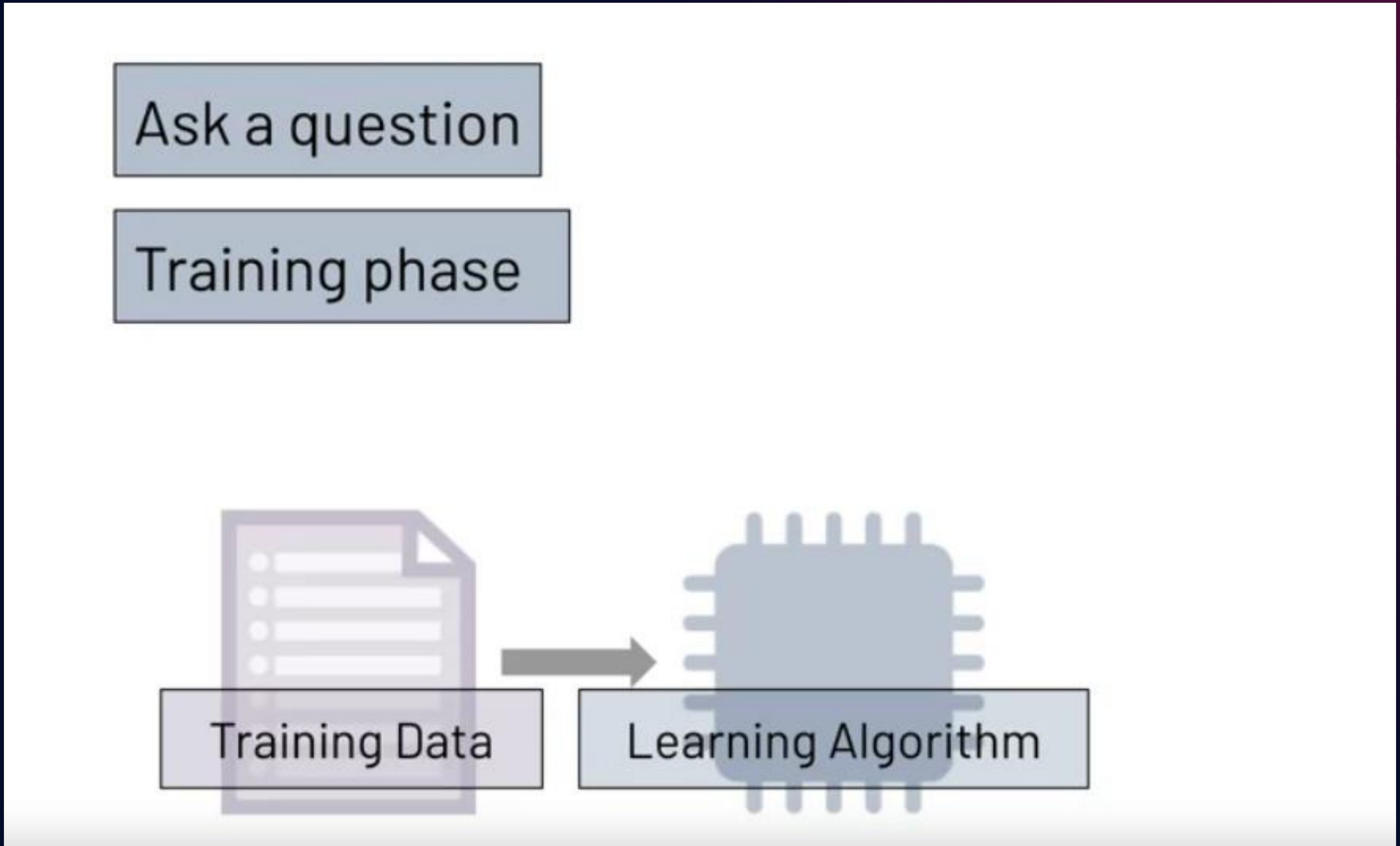


slide~superdatascience

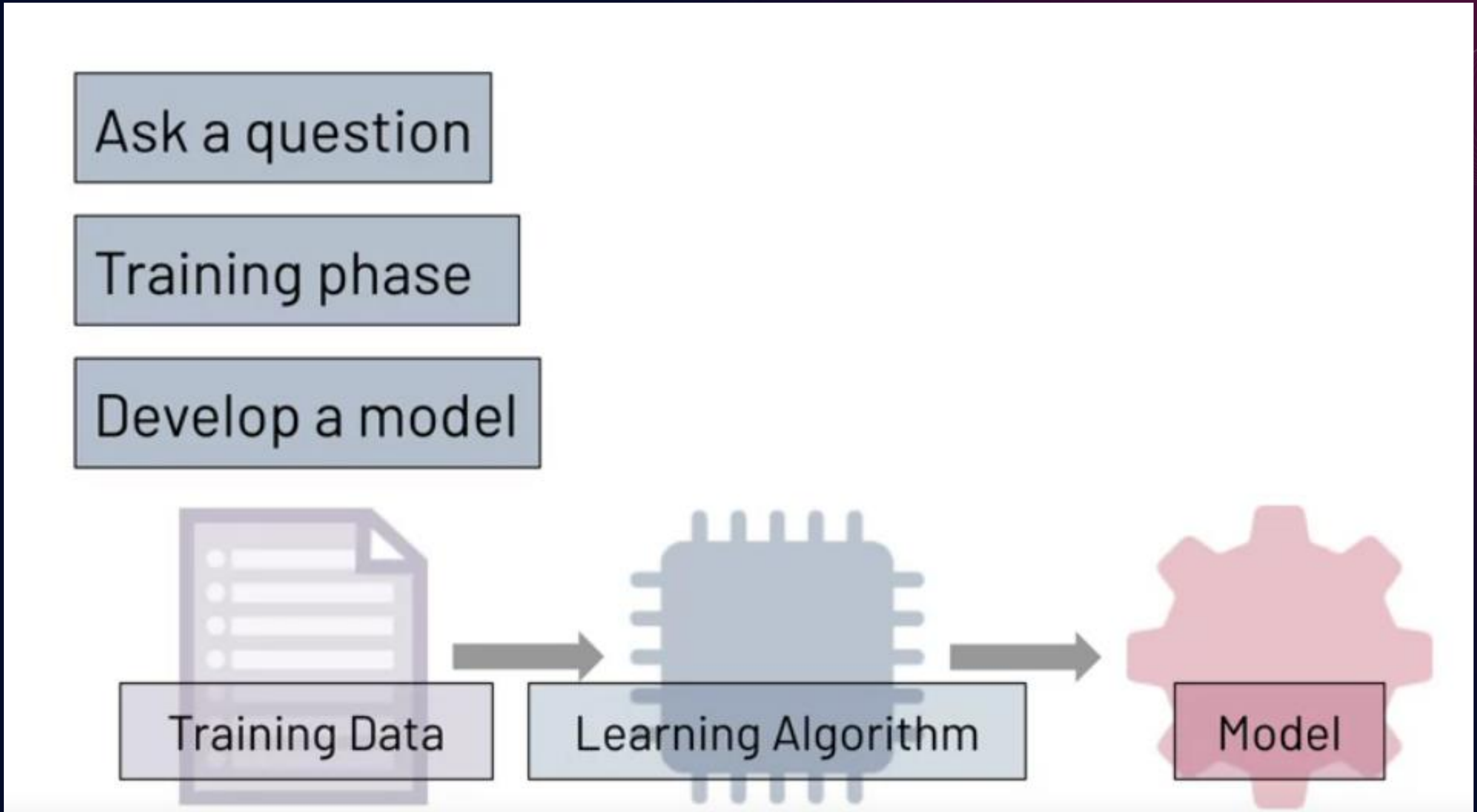
Classification Models in Machine Learning



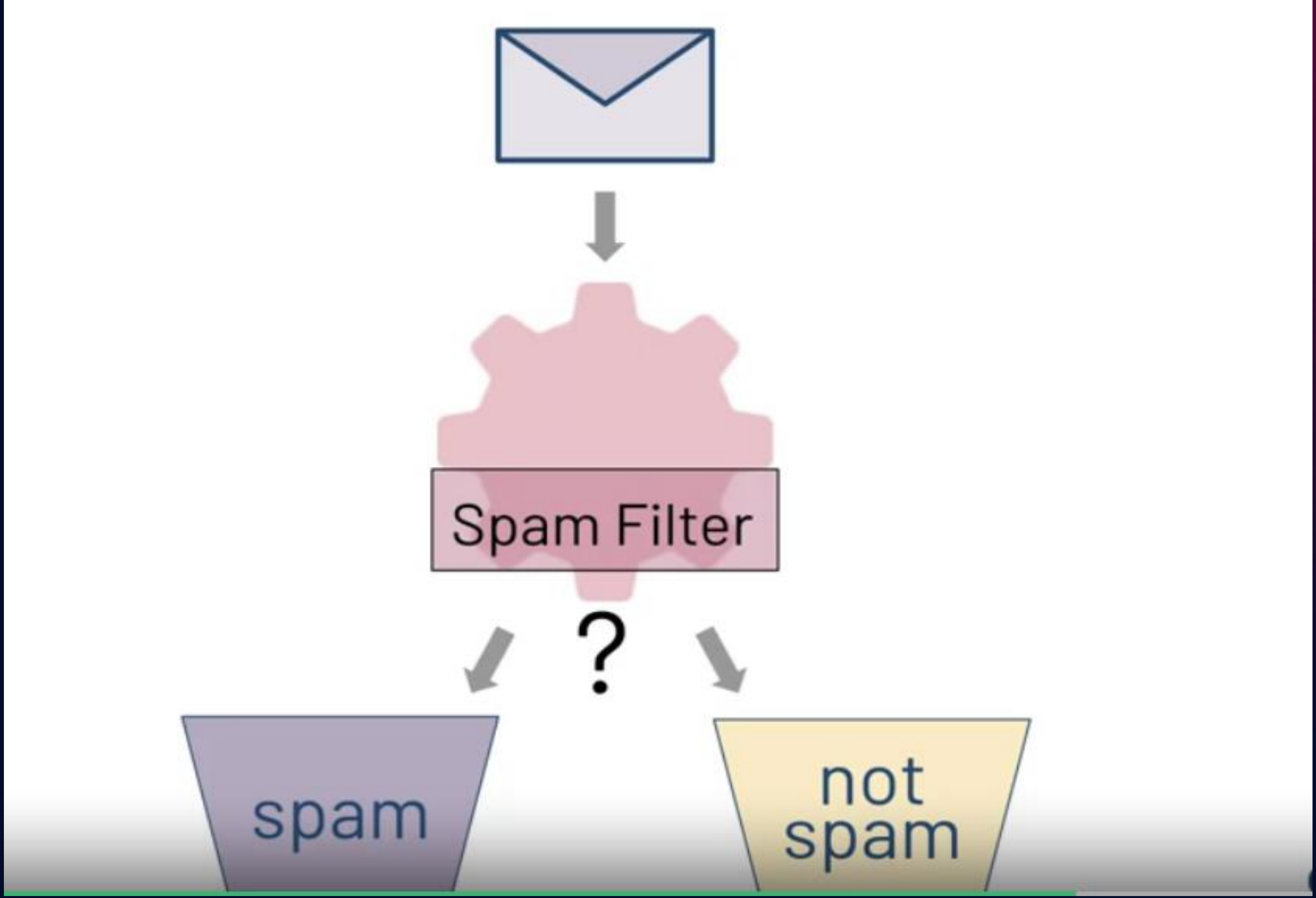
Classification Models in Machine Learning



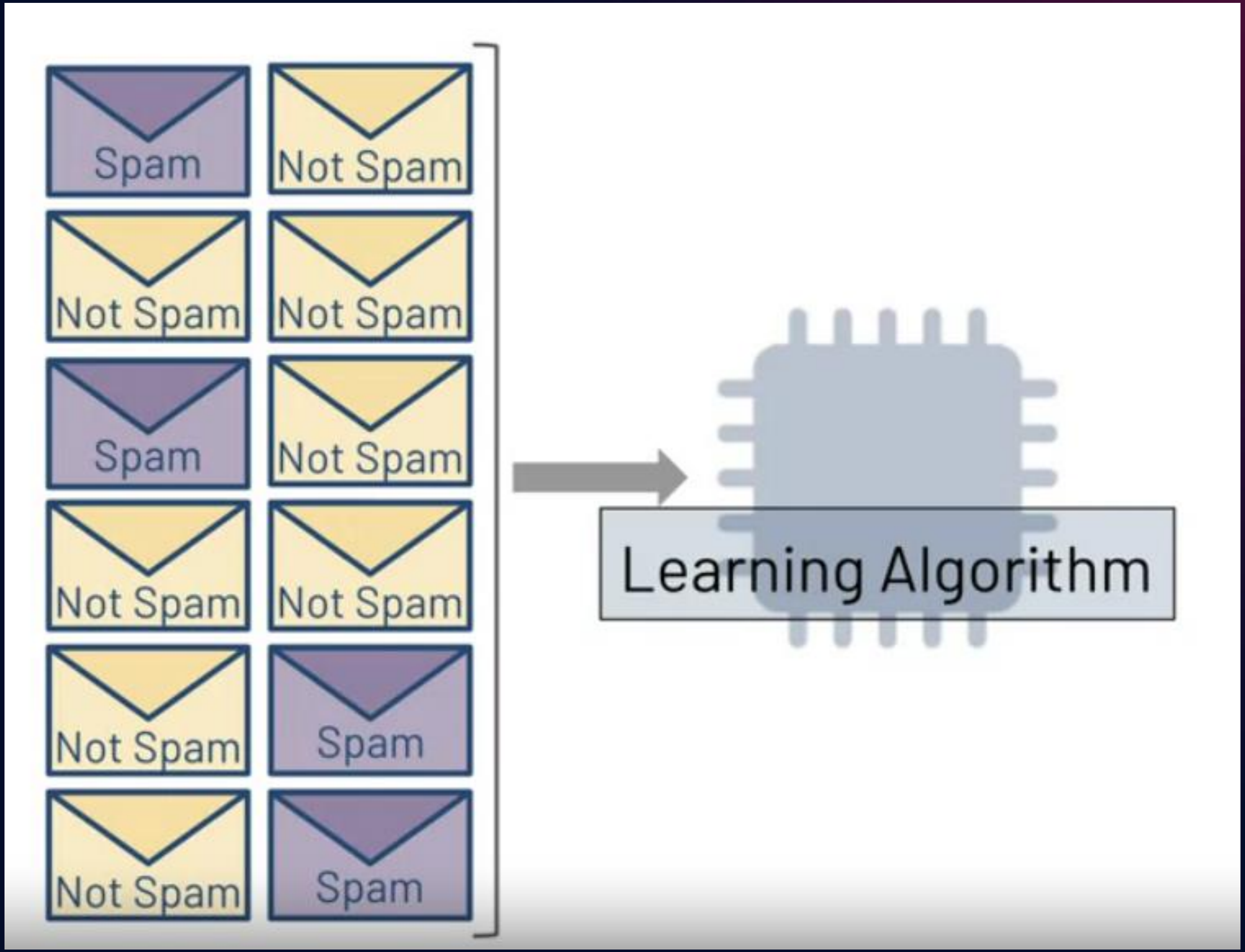
Classification Models in Machine Learning



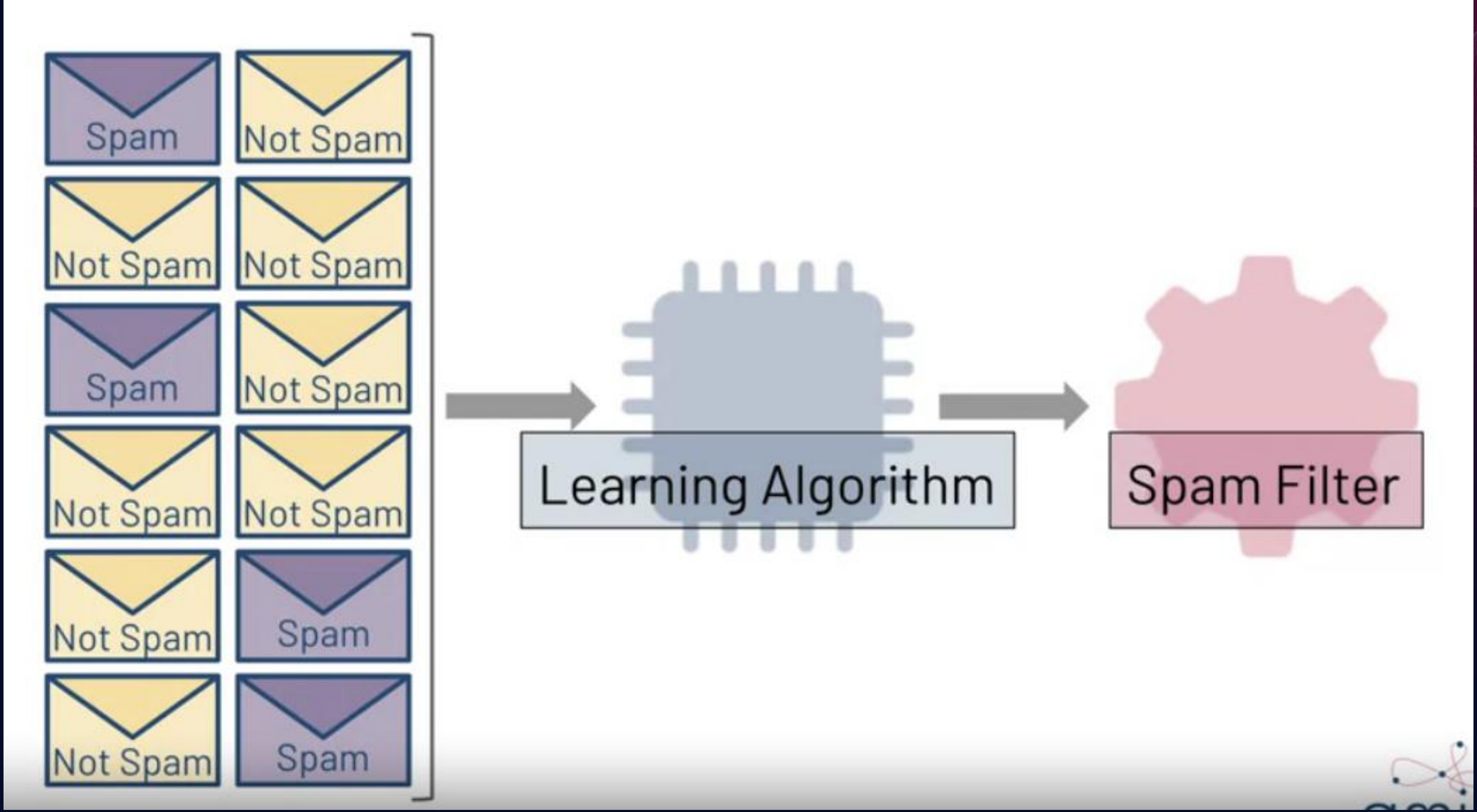
Classification Models in Machine Learning



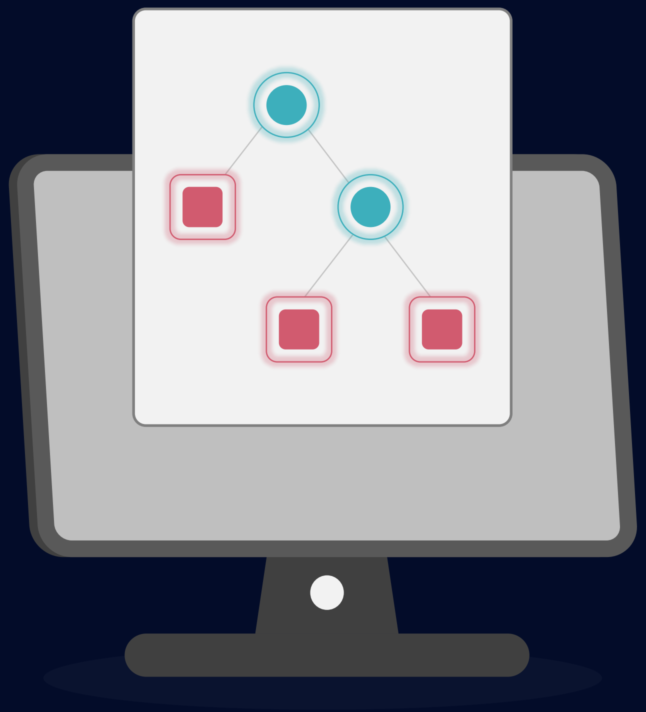
Classification Models in Machine Learning



Classification Models in Machine Learning



Decision Tree



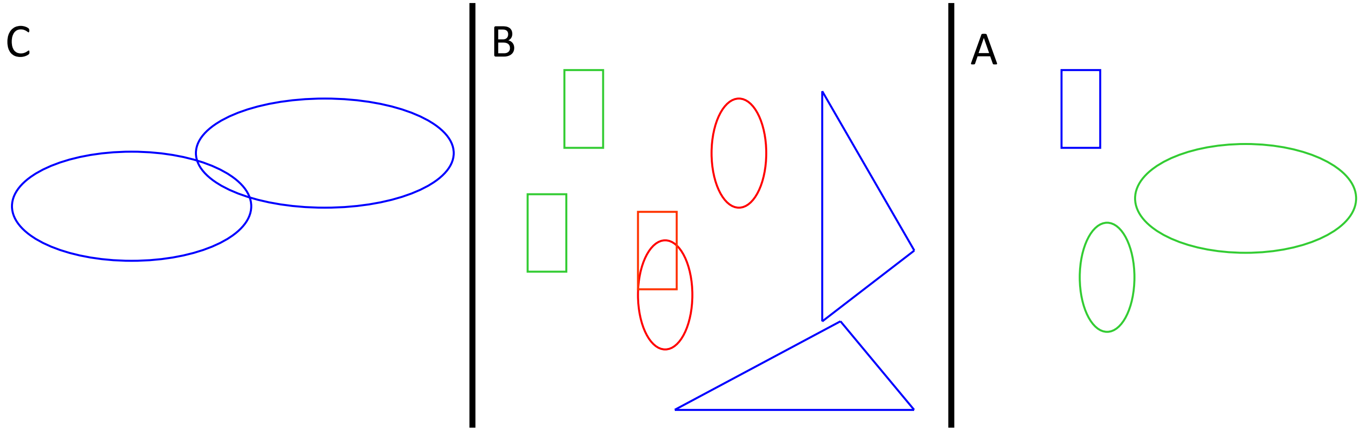
It is a handy tool with many applications. Decision trees can be used to solve classification and regression issues

The name indicates that it displays the predictions coming from a series of feature-based splits using a flowchart-like tree structure

It all starts with a root node and ends with a leaf choice

Decision Trees

- A hierarchical data structure that represents data by implementing a divide and conquer strategy
- Can be used as a non-parametric classification and regression method
- Given a collection of examples, learn a decision tree that represents it.
- Use this representation to classify new examples

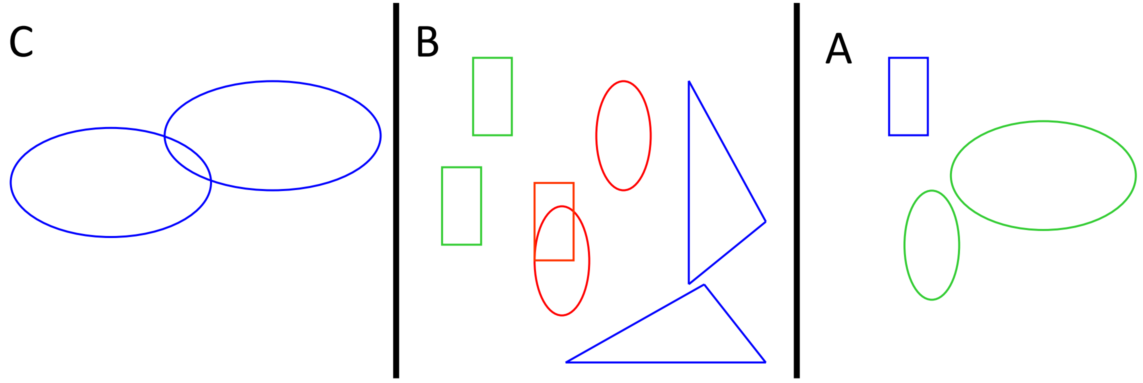


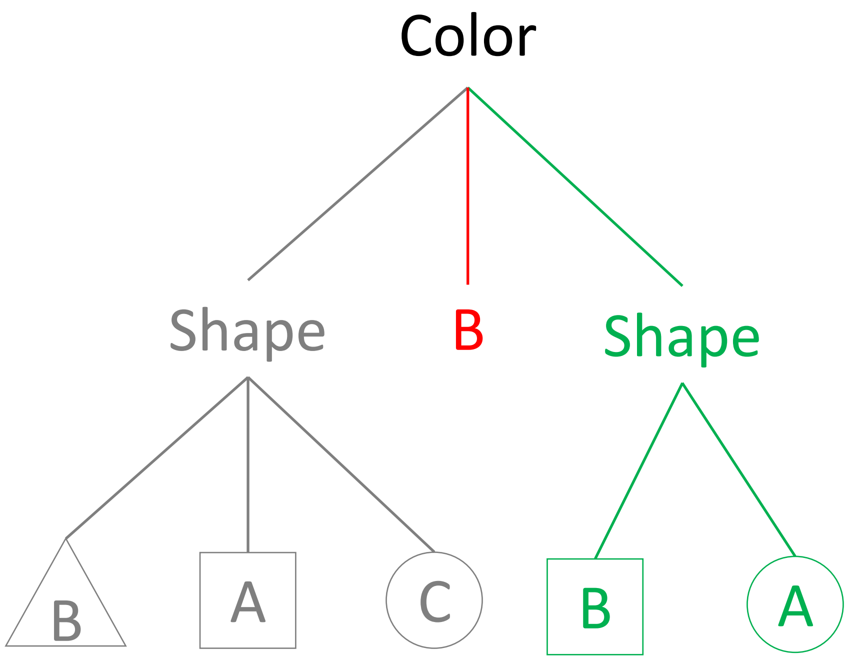
The Representation

- Decision Trees are classifiers for instances represented as feature vectors
  - color={red, blue, green} ; shape={circle, triangle, rectangle} ; label= {A, B, C}
- Nodes are tests for feature values
- There is one branch for each value of the feature
- Leaves specify the category (labels)
- Can categorize instances into multiple disjoint categories

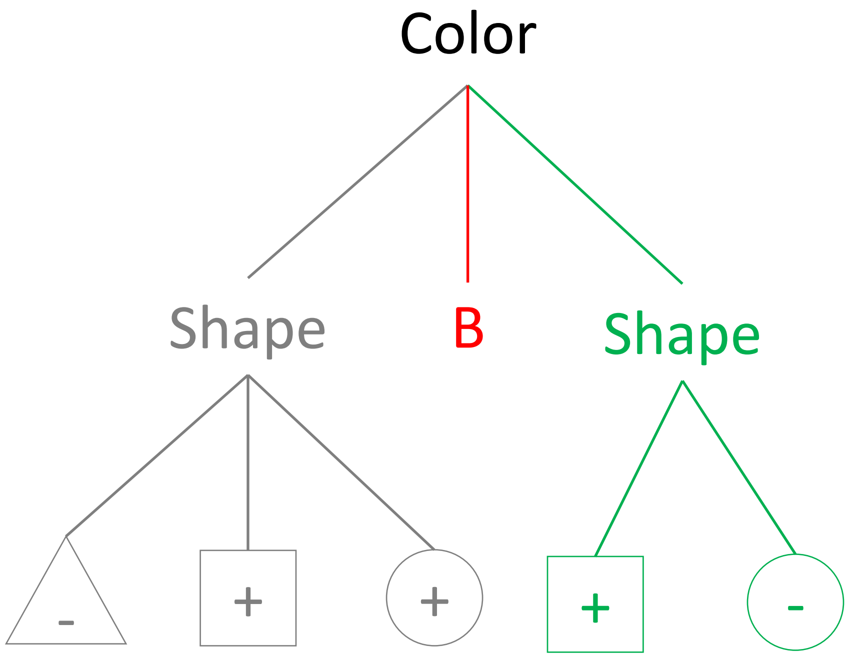
Evaluation of a Decision Tree

Learning a Decision Tree



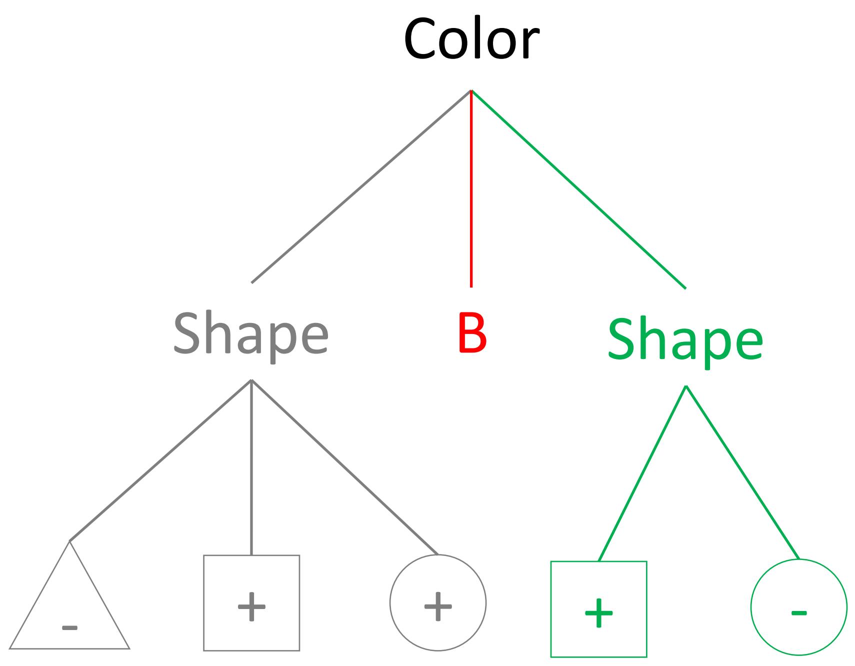


Expressivity of Decision Trees



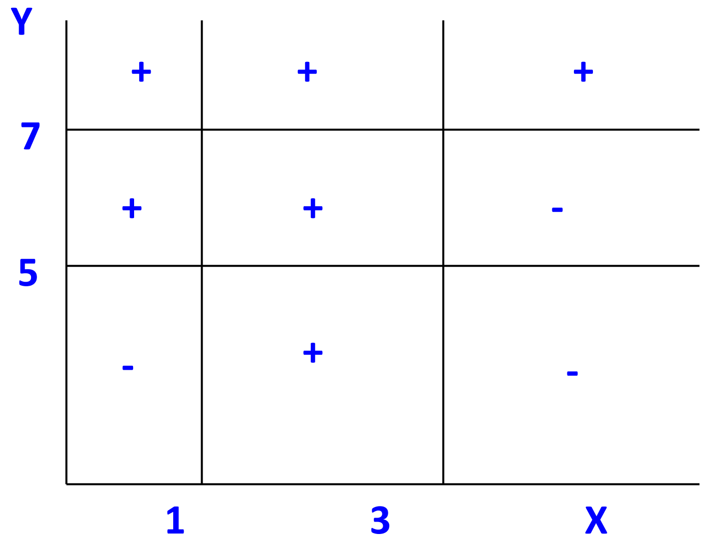
Decision Trees

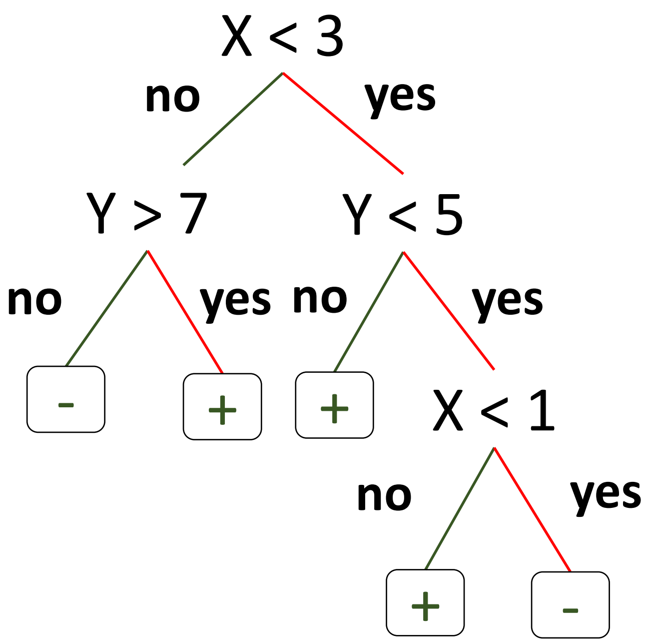
- Output is a discrete category. Real valued outputs are possible (regression trees)
- There are efficient algorithms for processing large amounts of data (but not too many features)
- There are methods for handling **noisy data** (classification noise and attribute noise) and for handling missing attribute values



Decision Boundaries

- Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes
- In this case, the tree divides the features space into axis-parallel rectangles, each labeled with one of the labels





Today’s key concepts

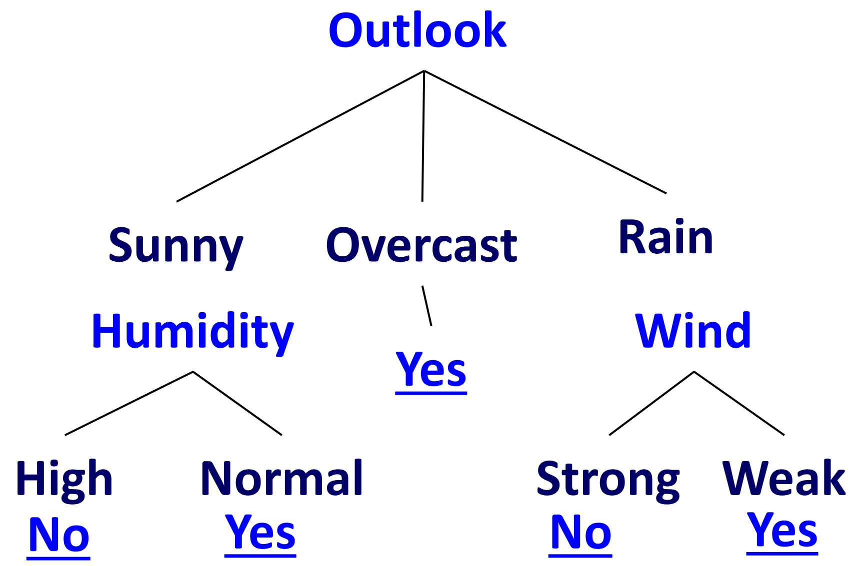
- Learning decision trees (ID3 algorithm)
  - Greedy heuristic (based on information gain)
    Originally developed for discrete features

Learning decision trees (ID3 algorithm

Decision Trees

- Can represent any Boolean Function
- Can be viewed as a way to compactly represent a lot of data.
- Natural representation: (20 questions)
- The evaluation of the Decision Tree Classifier is easy

- Clearly, given data, there are many ways to represent it as a decision tree.
- Learning a good representation from data is the challenge.



Will I play tennis today?

- **Features**
  - Outlook: {Sun, Overcast, Rain}
  - Temperature: {Hot, Mild, Cool}
  - Humidity: {High, Normal, Low}
  - Wind: {Strong, Weak}

- **Labels**
  - Binary classification task: Y = {+, -}

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Basic Decision Trees Learning Algorithm

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Algorithm?

LEARN-DECISION-TREE

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
      A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
      tree ← a new decision tree with root test A
      for each value v of A do
          exs ← {e : e ∈ examples and e.A = v}
          subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)
          add a branch to tree with label (A = v) and subtree subtree
      return tree
```

The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - The main decision in the algorithm is the selection of the next attribute to condition on.
- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.
  - The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$Entropy(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

- p_+ is the proportion of positive examples in S and
- p_- is the proportion of negatives examples in S
  - If all the examples belong to the same category: Entropy = 0
  - If all the examples are equally mixed (0.5, 0.5): Entropy = 1
  - Entropy = Level of uncertainty.
- In general, when p_i is the fraction of examples labeled i:

$$Entropy(S[p_1, p_2, \dots, p_k]) = -\sum_1^k p_i \log(p_i)$$

- Entropy can be viewed as the number of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 – can use less then 1 bit.

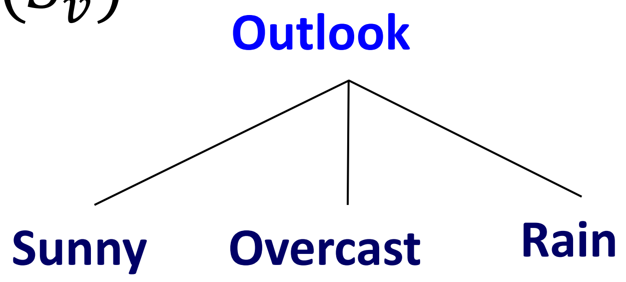
Information Gain

High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.

- The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(S)} \frac{|S_v|}{|S|} Entropy(S_v)$$



- Where:
  - S_v is the subset of S for which attribute a has value v, and
  - the entropy of partitioning the data is calculated by **weighing the entropy of each partition** by its size relative to the original set

- Partitions of low entropy (imbalanced splits) lead to high gain
- Go back to check which of the A, B splits is better

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

calculate current entropy

- $p_+ = \frac{9}{14} \quad p_- = \frac{5}{14}$
- $Entropy(Play) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$

$$= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$\approx 0.94$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(S)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Outlook = sunny:

$p_+ = 2/5 \quad p_- = 3/5$ **Entropy(O = S) = 0.971**

Outlook = overcast:

$p_+ = 4/4 \quad p_- = 0$ **Entropy(O = O) = 0**

Outlook = rainy:

$p_+ = 3/5 \quad p_- = 2/5$ **Entropy(O = R) = 0.971**

Expected entropy

= $\sum_{v \in values(S)} \frac{|S_v|}{|S|} Entropy(S_v)$

= (5/14)×0.971 + (4/14)×0 + (5/14)×0.971 = **0.694**

Information gain = 0.940 – 0.694 **= 0.246**

Information Gain: Humidity

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(S)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Humidity = high:

$p_+ = 3/7 \quad p_- = 4/7$ **Entropy(H = H) = 0.985**

Humidity = Normal:

$p_+ = 6/7 \quad p_- = 1/7$ **Entropy(H = N) = 0.592**

Expected entropy

= $\sum_{v \in values(S)} \frac{|S_v|}{|S|} Entropy(S_v)$

= (7/14)×0.985 + (7/14)×0.592 = **0.7785**

Information gain = 0.940 – 0.7785 **= 0.1615**

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246
Humidity: 0.1615
Wind:?
Temperature:?

→ Split on Outlook

Which feature to split on?

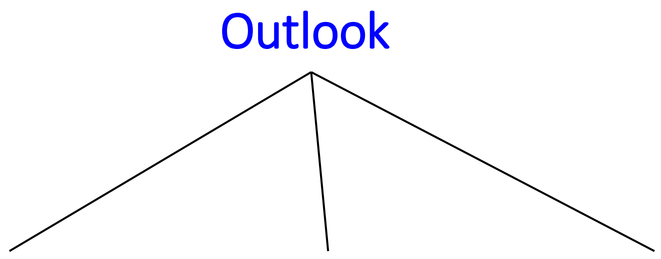
	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246
Humidity: 0.1615
Wind: 0.048
Temperature: 0.029

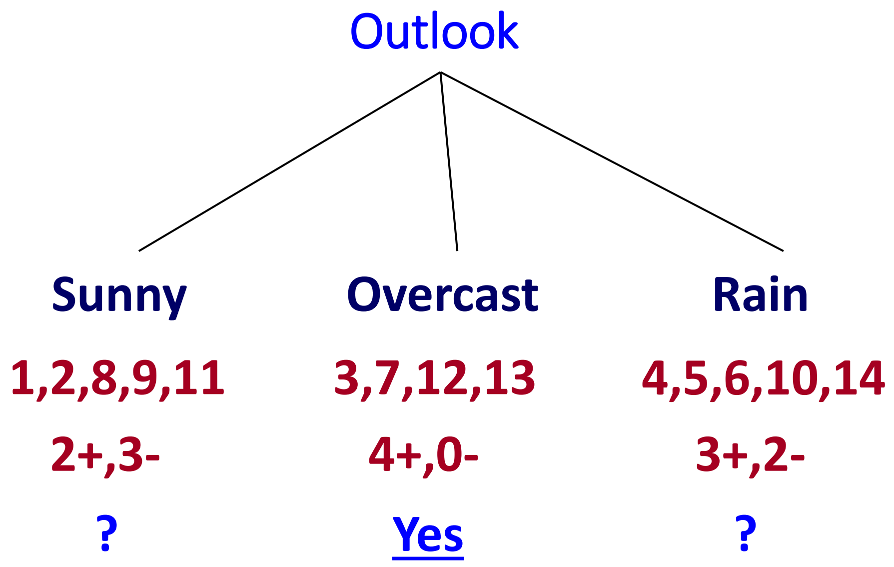
→ Split on Outlook

An Illustrative Example (III)



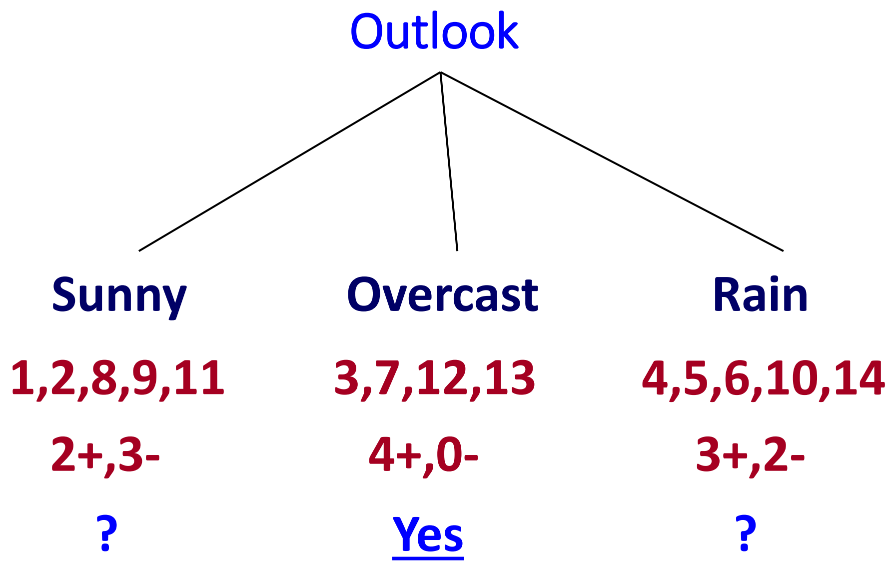
Gain(S,Humidity)=0.1615
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example (III)



	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example (III)



Continue until:
- Every attribute is included in path, or,
- All examples in the leaf have same label

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example (IV)

Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

$\text{Gain}(S_{sunny}, \text{Humidity}) =$ .97-(3/5) 0-(2/5) 0 = .97

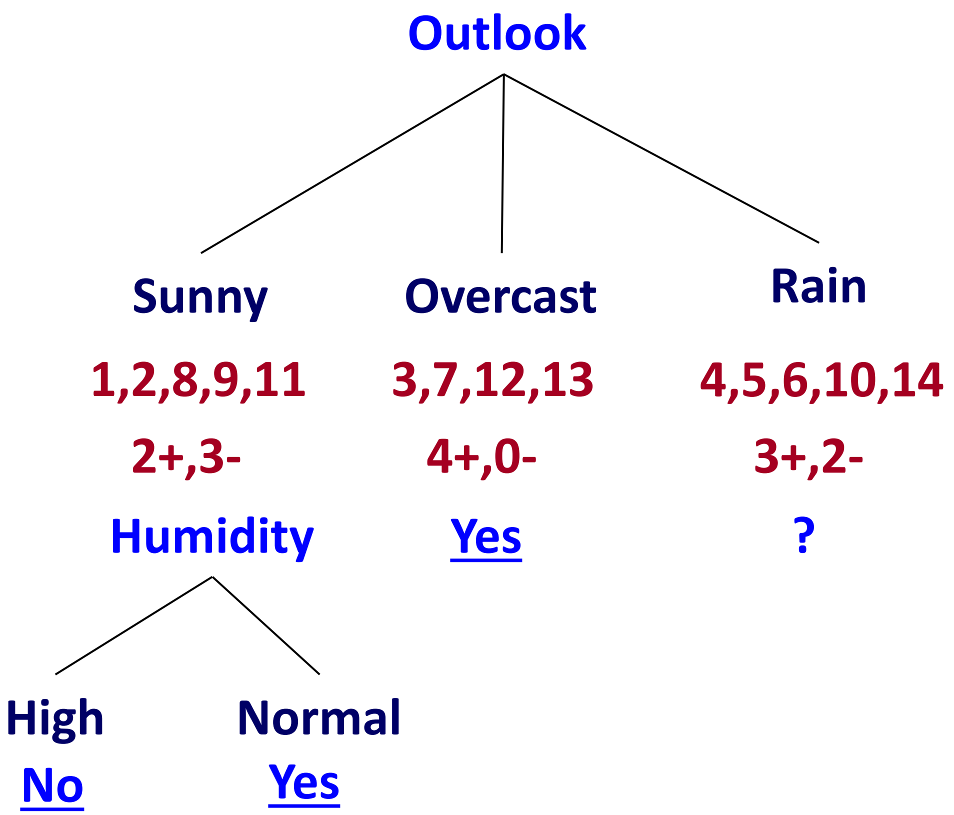
$\text{Gain}(S_{sunny}, \text{Temp}) =$ .97- 0-(2/5) 1 = .57

$\text{Gain}(S_{sunny}, \text{Wind}) =$ .97-(2/5) 1 - (3/5) .92= .02

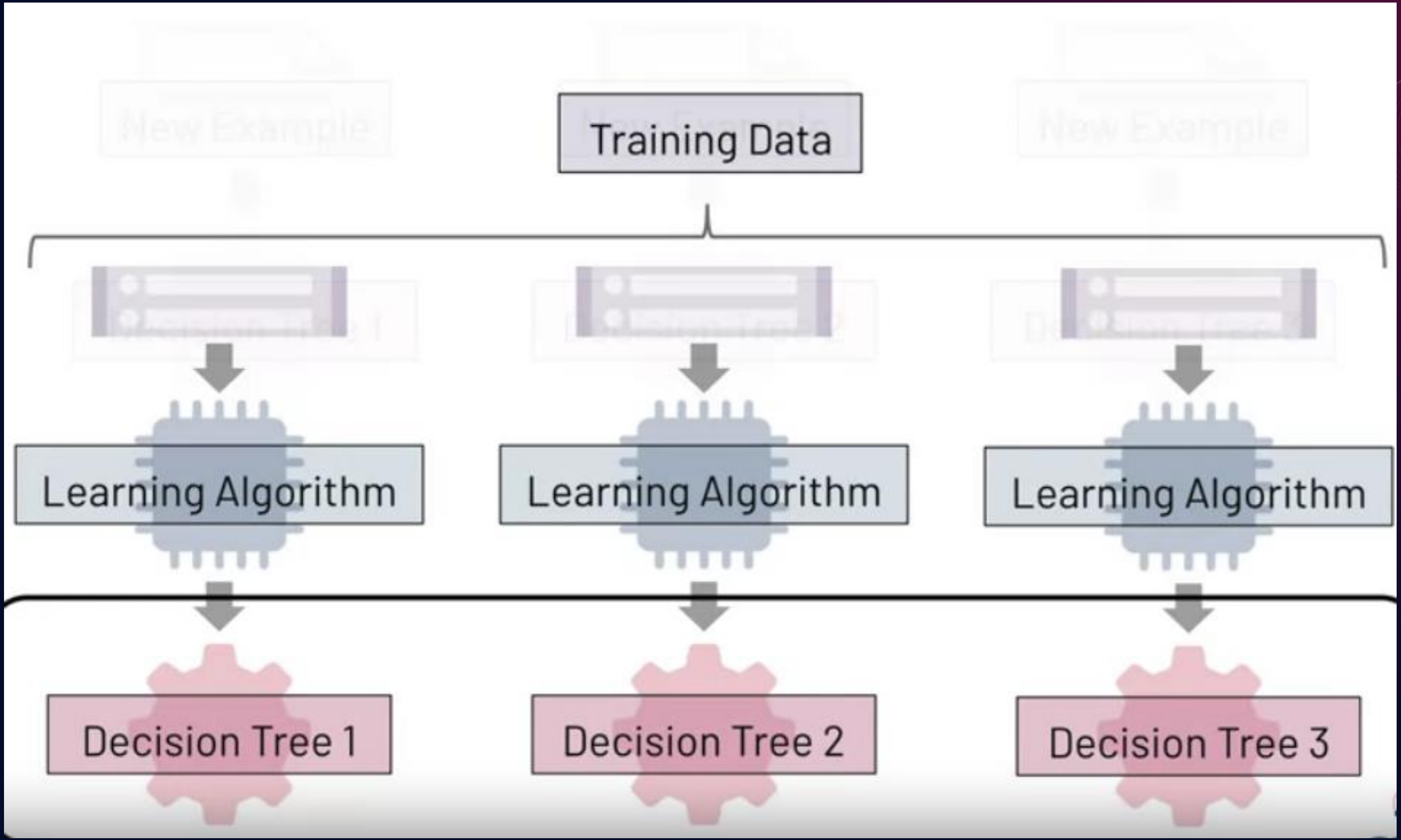
Split on Humidity

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example (V)



Classification Models: Decision Tree



Random Forest Algorithm

Ensemble Learning

Random Forest Algorithm

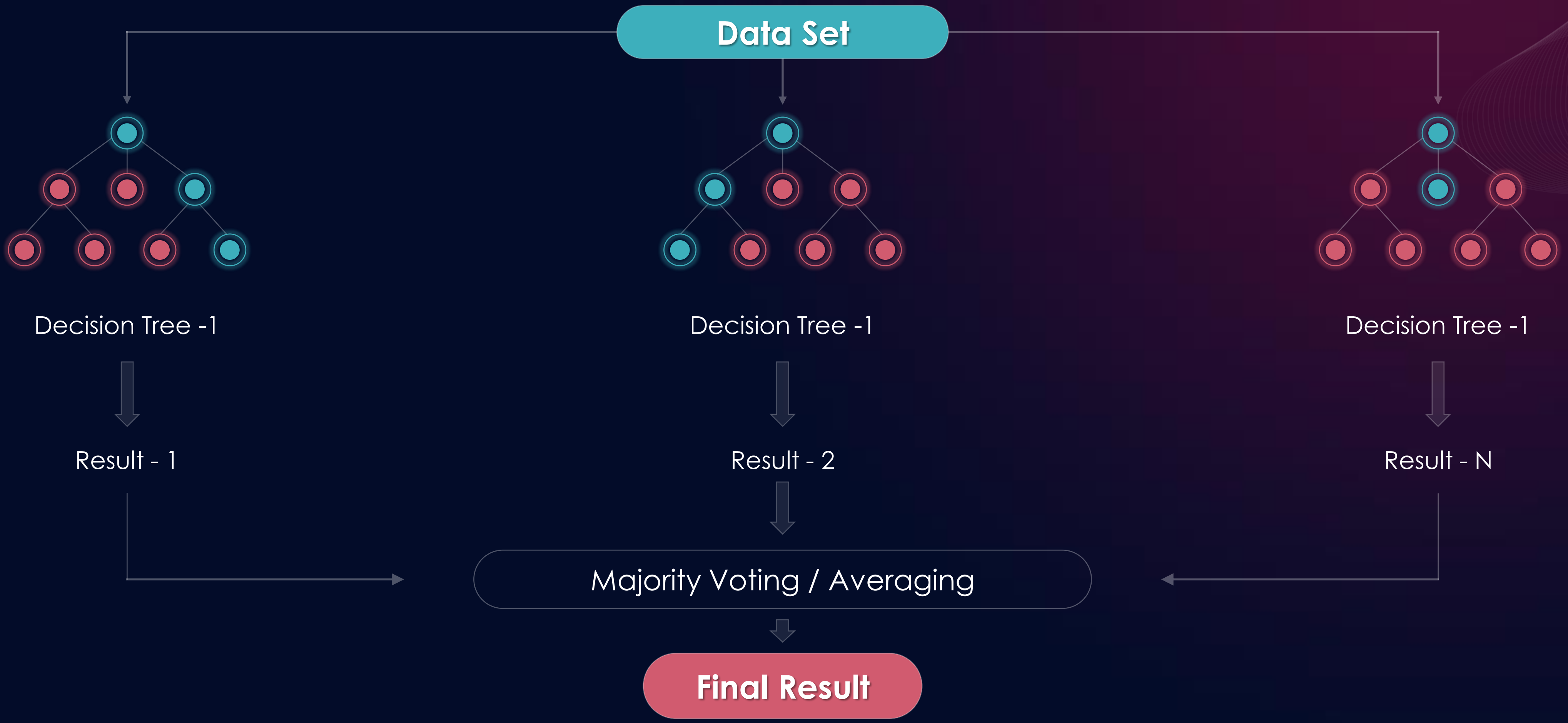
STEP 1: Pick at random K data points from the Training set.

STEP 2: Build the Decision Tree associated to these K data points.

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y t
for the data point in question, and assign the new data point the average across all of th
predicted Y values.

Random Forest Algorithm



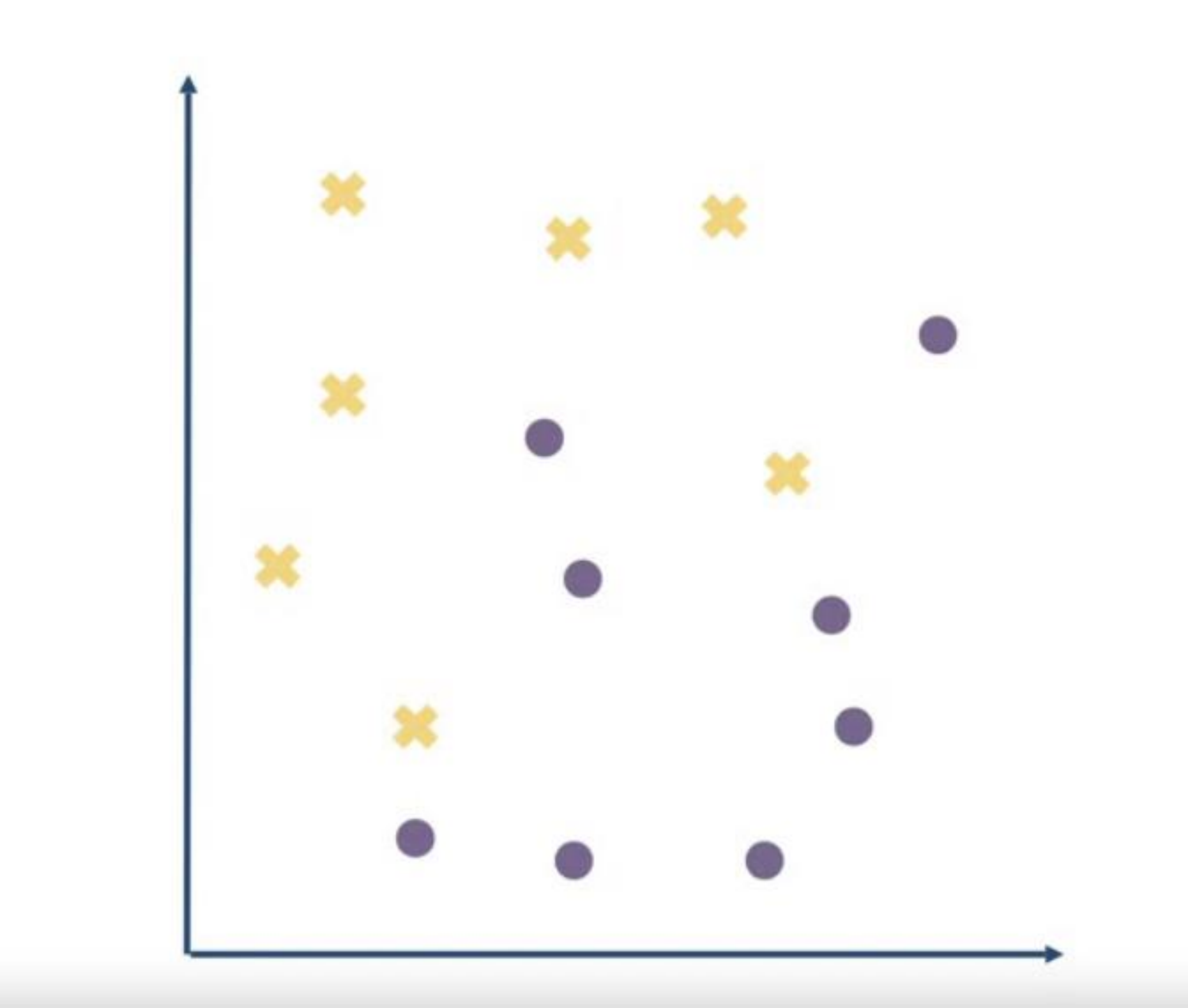
A Random Forest is a cluster of decision trees. Each tree is classed, and the tree "votes" for that class to classify a new item based on its properties. The forest chooses the categorization with the highest votes (over all the trees in the forest).

KNN (K- Nearest Neighbors) Algorithm

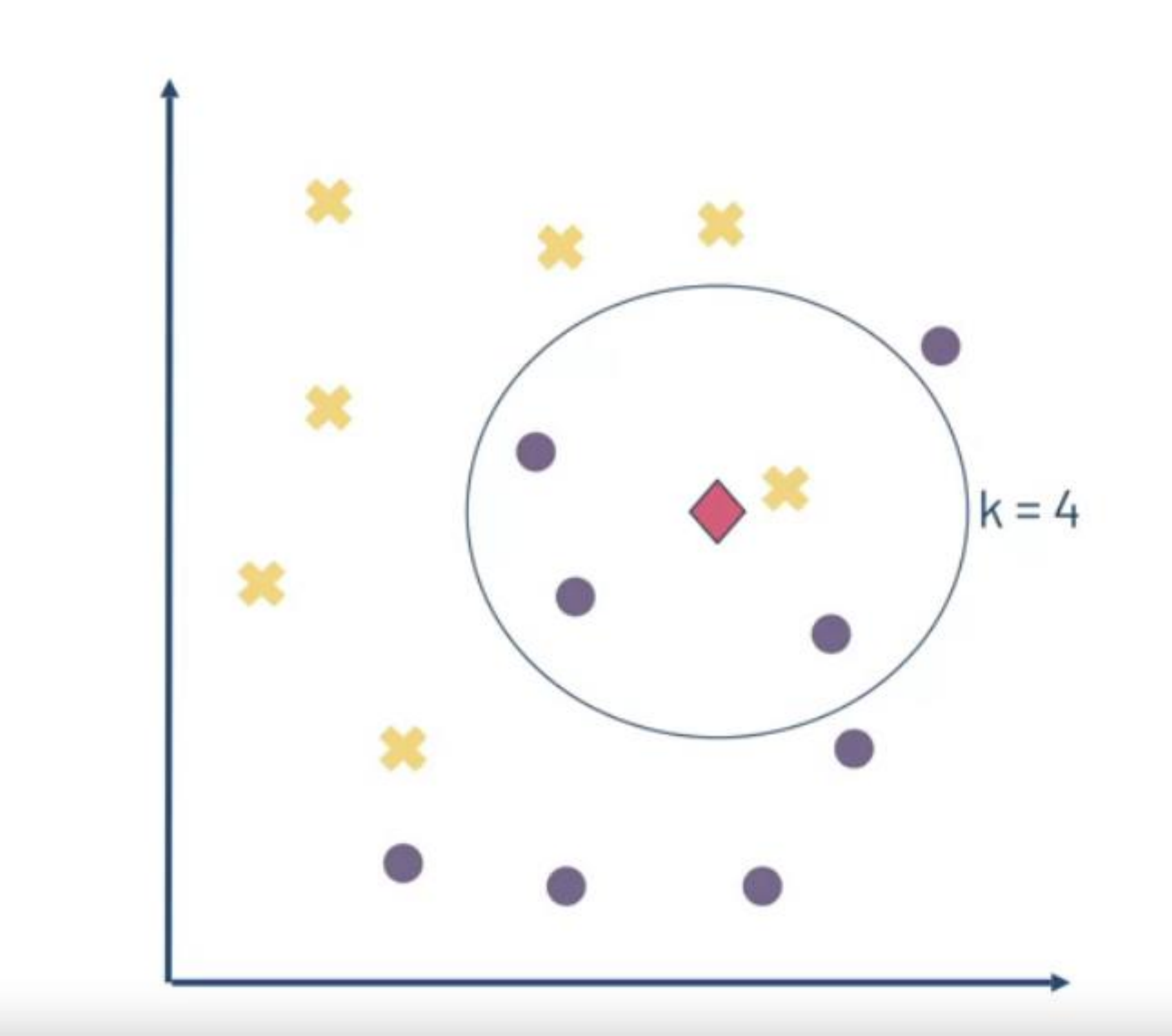
It's a simple algorithm that keeps all existing instances, and classifies new cases based on a majority vote of its k neighbors

The case is then given to the class with which it has greatest in common. A distance function is used to perform this task

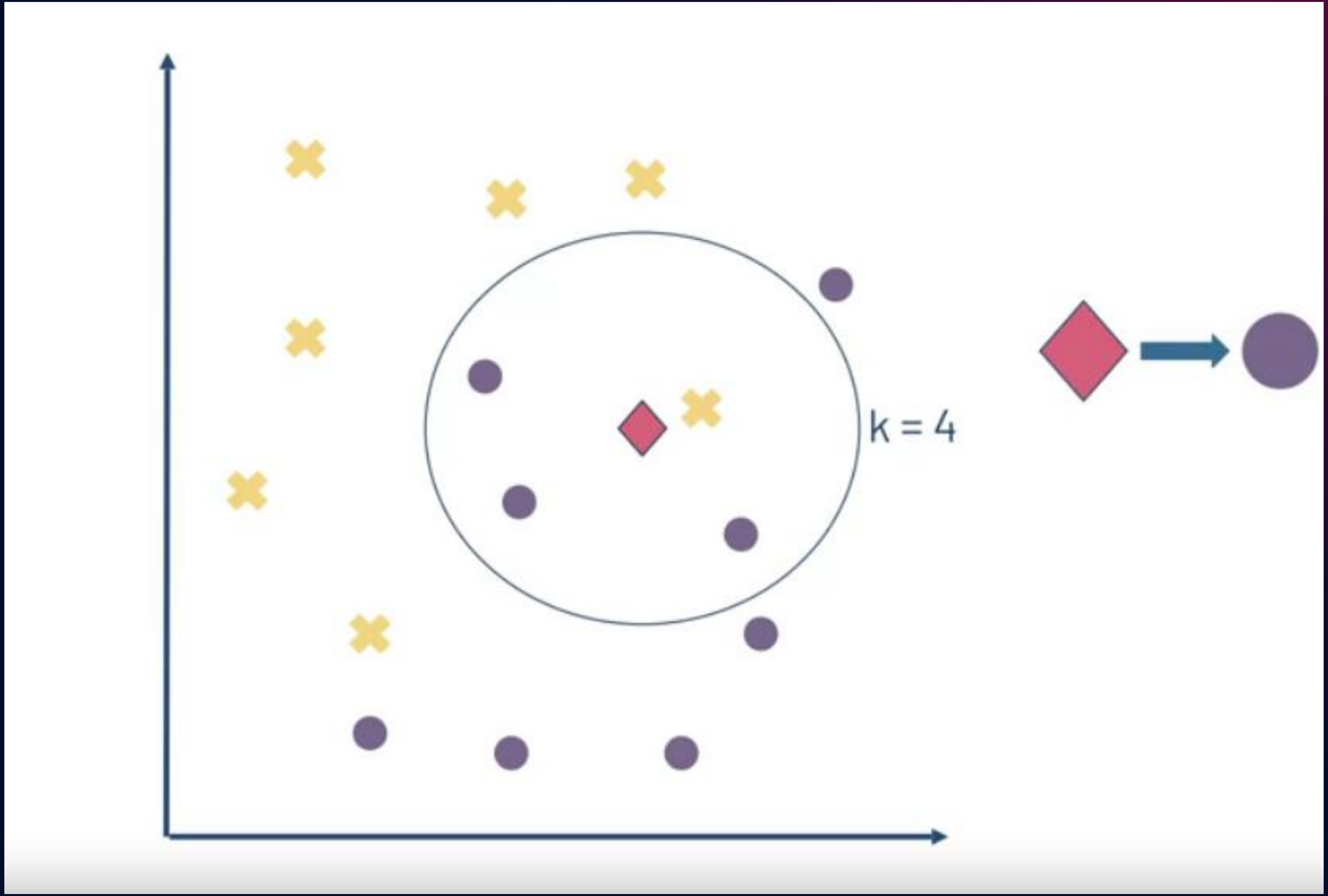
KNN (K- Nearest Neighbors) Algorithm



KNN (K- Nearest Neighbors) Algorithm

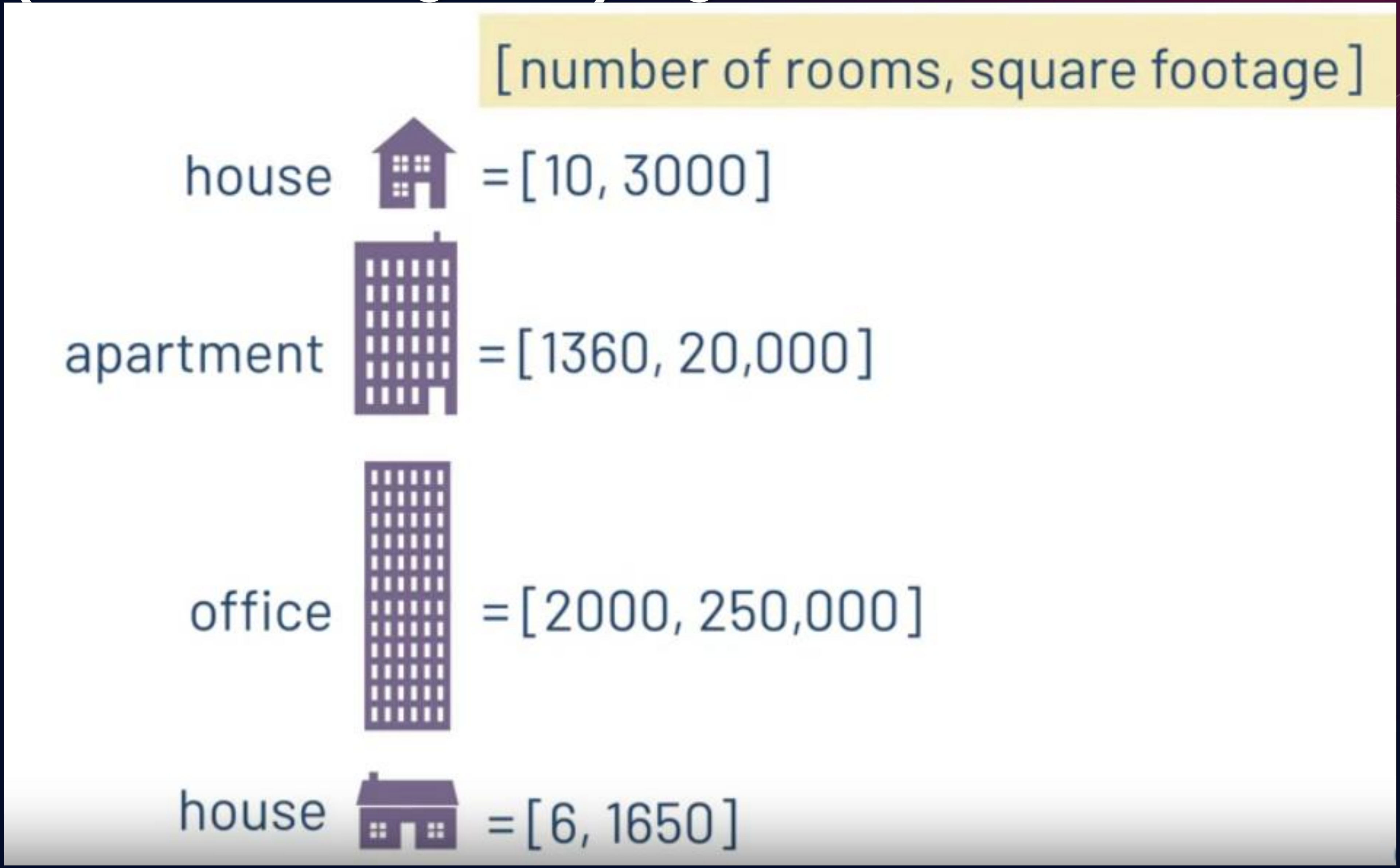


KNN (K- Nearest Neighbors) Algorithm

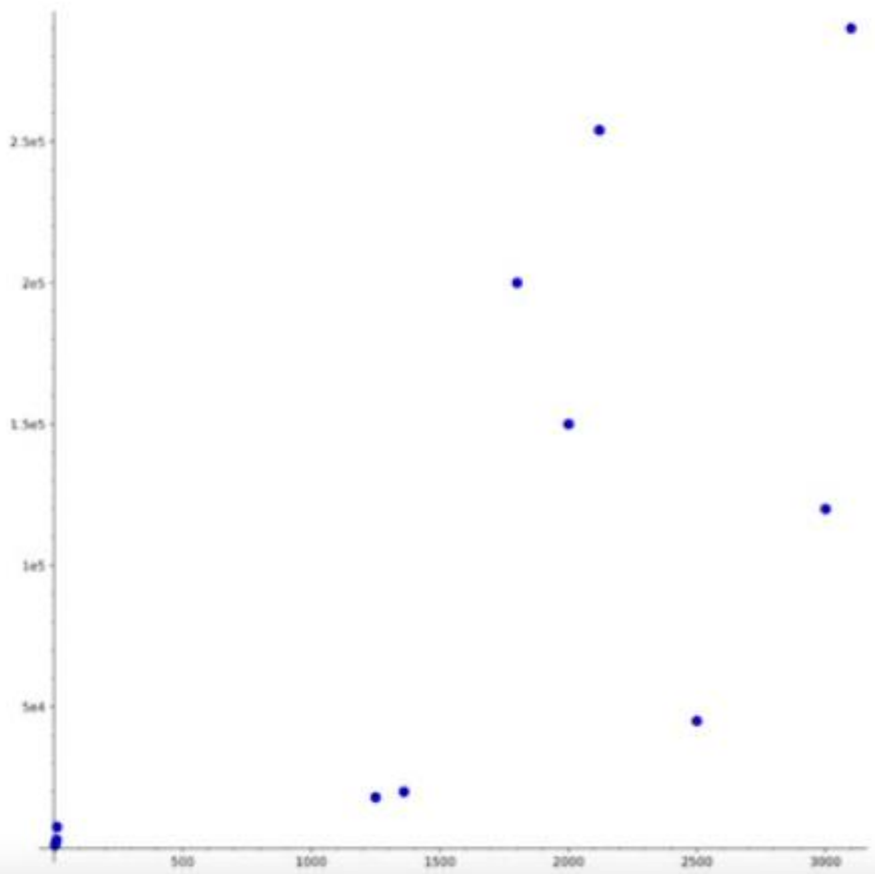


KNN (K- Nearest Neighbors) Algorithm

KNN (K- Nearest Neighbors) Algorithm



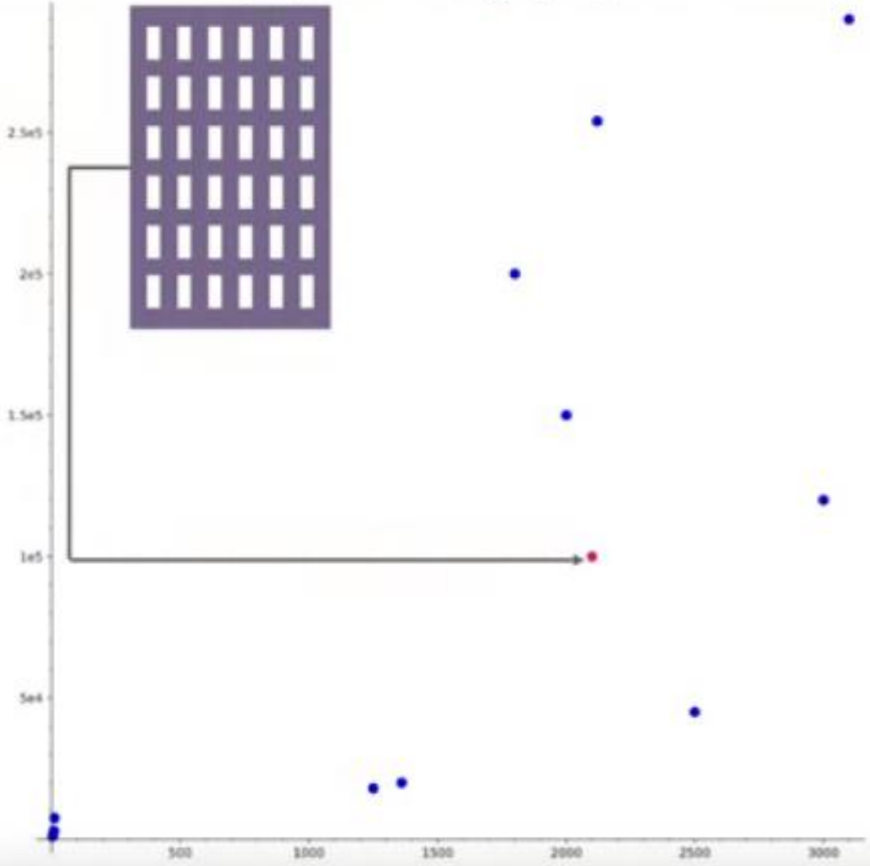
KNN (K- Nearest Neighbors) Algorithm



$$\begin{bmatrix} 10 & 3000 \\ 1360 & 20000 \\ 2000 & 250000 \\ 7 & 2050 \\ 4 & 1000 \\ 3100 & 290000 \\ 3000 & 120000 \\ 12 & 7500 \\ 2500 & 45000 \\ 1800 & 200000 \\ 2120 & 254000 \\ \vdots & \vdots \\ 1250 & 18000 \end{bmatrix} \begin{bmatrix} \text{house} \\ \text{apartment} \\ \text{office} \\ \text{house} \\ \text{house} \\ \text{office} \\ \text{apartment} \\ \text{house} \\ \text{apartment} \\ \text{office} \\ \text{office} \\ \vdots \\ \text{apartment} \end{bmatrix}$$

KNN (K- Nearest Neighbors) Algorithm

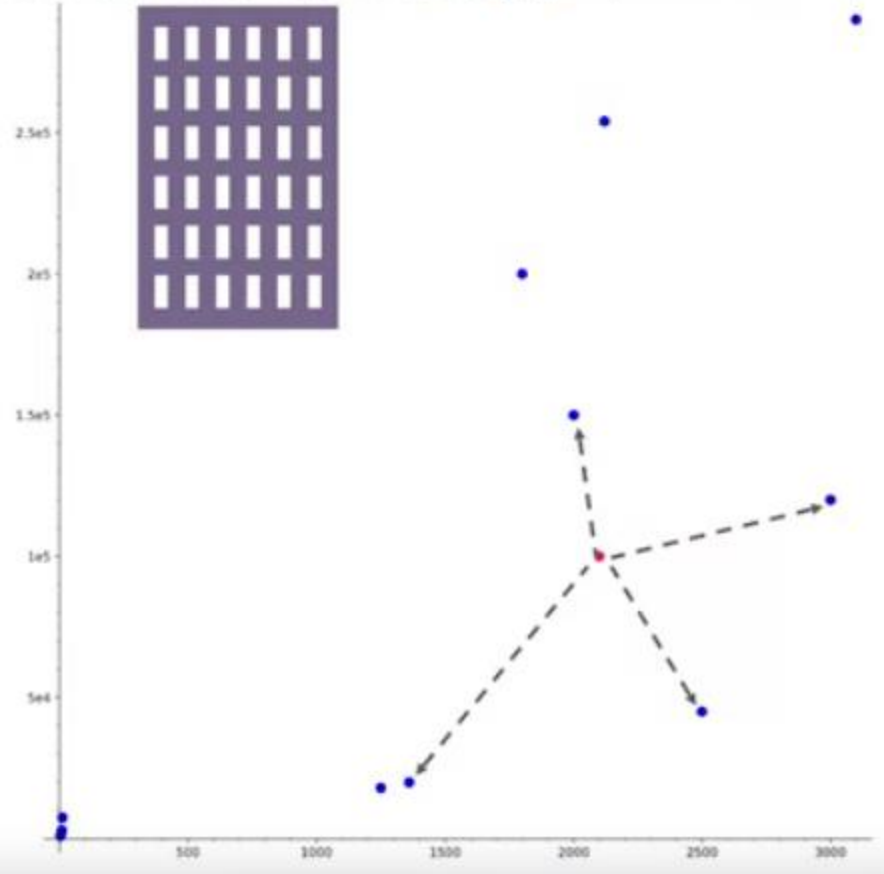
[number of rooms, square footage] = [2100 100000]



$$\begin{bmatrix} 10 & 3000 \\ 1360 & 20000 \\ 2000 & 250000 \\ 7 & 2050 \\ 4 & 1000 \\ 3100 & 290000 \\ 3000 & 120000 \\ 12 & 7500 \\ 2500 & 45000 \\ 1800 & 200000 \\ 2120 & 254000 \\ \vdots & \vdots \\ 1250 & 18000 \end{bmatrix} \begin{bmatrix} \text{house} \\ \text{apartment} \\ \text{office} \\ \text{house} \\ \text{house} \\ \text{office} \\ \text{apartment} \\ \text{house} \\ \text{apartment} \\ \text{office} \\ \text{office} \\ \vdots \\ \text{apartment} \end{bmatrix}$$

KNN (K- Nearest Neighbors) Algorithm

$$[\text{number of rooms, square footage}] = \begin{bmatrix} 2100 & 100000 \end{bmatrix}$$

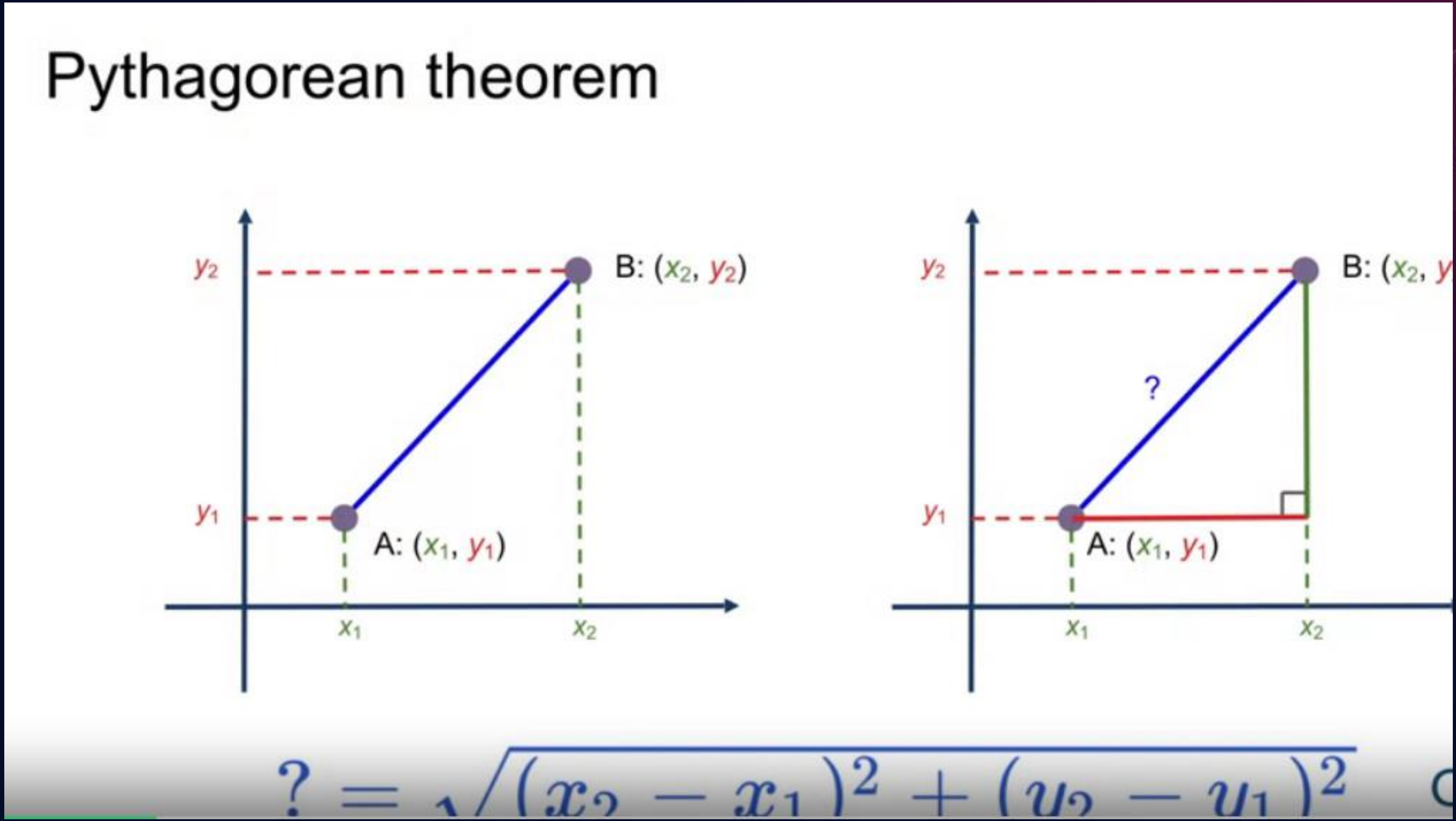


$$\begin{bmatrix} 10 & 3000 \\ 1360 & 20000 \\ 2000 & 250000 \\ 7 & 2050 \\ 4 & 1000 \\ 3100 & 290000 \\ 3000 & 120000 \\ 12 & 7500 \\ 2500 & 45000 \\ 1800 & 200000 \\ 2120 & 254000 \\ \vdots & \vdots \\ 1250 & 18000 \end{bmatrix} \quad \begin{bmatrix} \text{house} \\ \text{apartment} \\ \text{office} \\ \text{house} \\ \text{house} \\ \text{office} \\ \text{apartment} \\ \text{house} \\ \text{apartment} \\ \text{office} \\ \text{office} \\ \vdots \\ \text{apartment} \end{bmatrix}$$

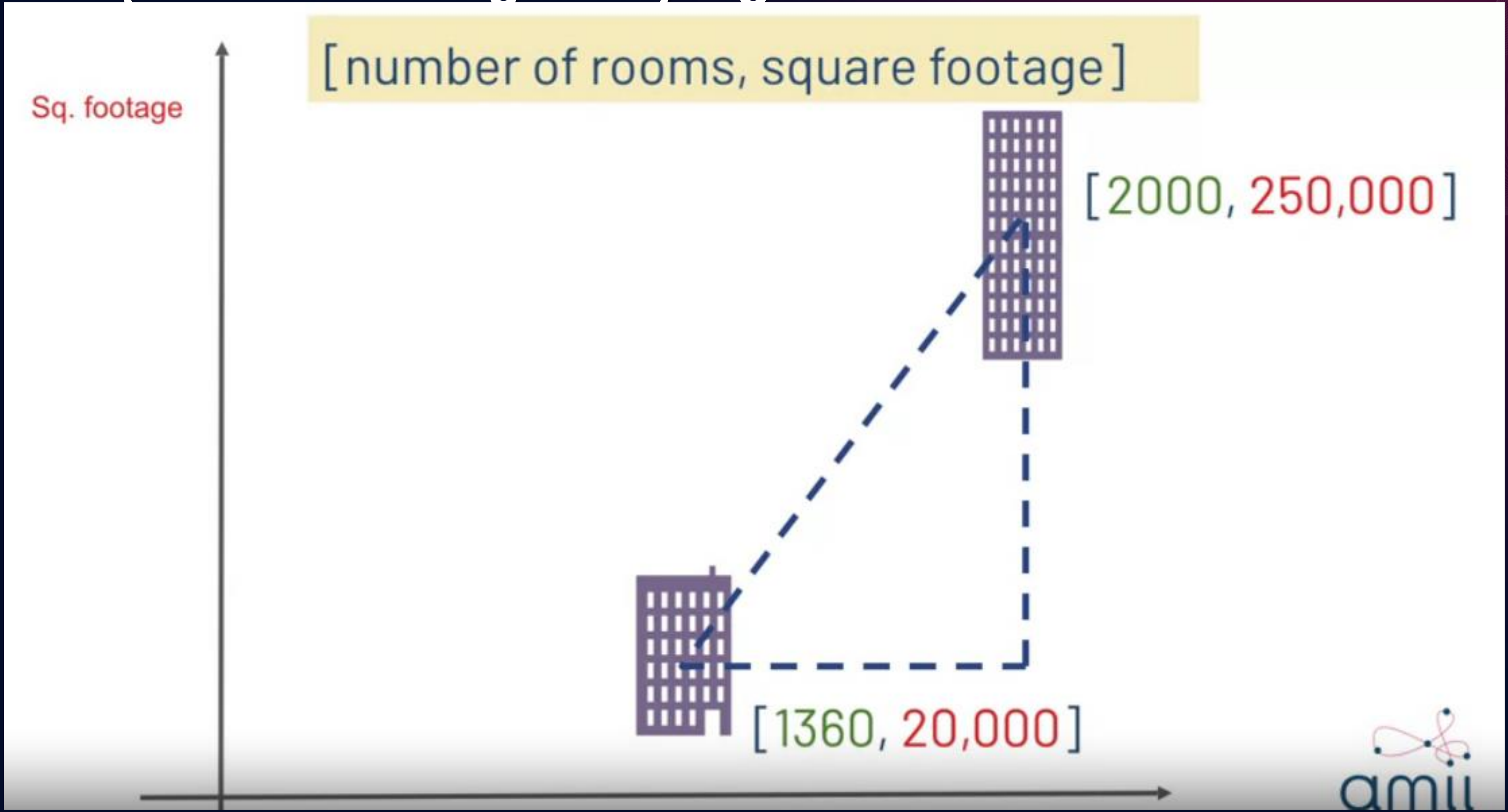
KNN (K- Nearest Neighbors) Algorithm

$$\text{m examples}\left\{\underbrace{\begin{bmatrix} 10 & 3000 & 150 & 1 & \cdots & 0 \\ 1360 & 20000 & 220 & 1 & \cdots & 1 \\ 2000 & 150000 & 300 & 0 & \cdots & 2 \\ 7 & 2050 & 45 & 0 & \cdots & 0 \\ 4 & 1000 & 13 & 1 & \cdots & 0 \\ 3100 & 290000 & 480 & 0 & \cdots & 2 \\ 3000 & 120000 & 200 & 1 & \cdots & 1 \\ 12 & 7500 & 16 & 1 & \cdots & 0 \\ 4120 & 45000 & 305 & 0 & \cdots & 1 \\ 1800 & 200000 & 122 & 1 & \cdots & 2 \\ 2120 & 254000 & 320 & 0 & \cdots & 2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1250 & 18000 & 100 & 0 & \cdots & 1 \end{bmatrix}}_{\text{n features}}\right.$$

KNN (K- Nearest Neighbors) Algorithm

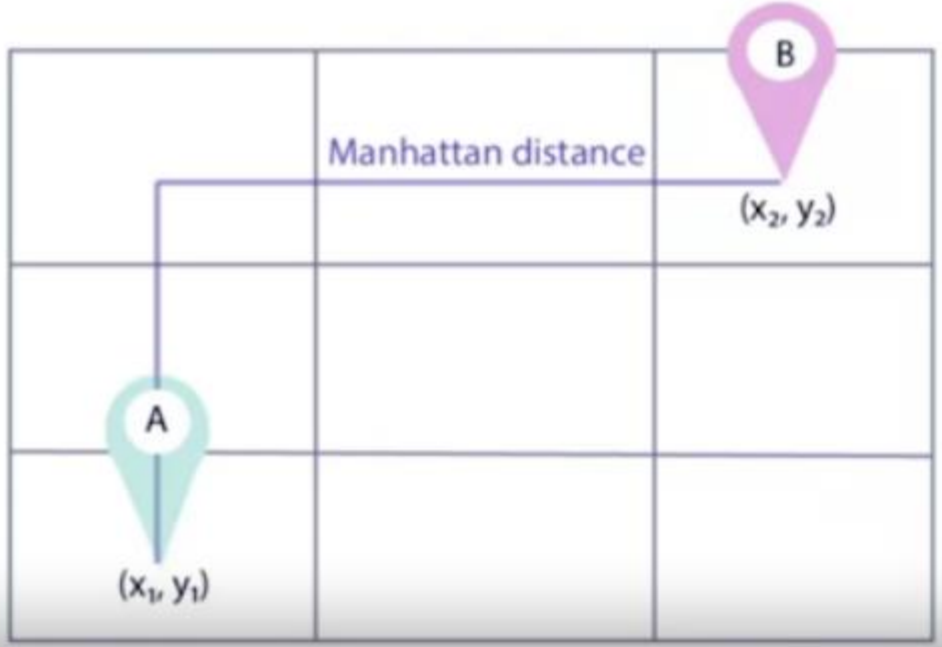


KNN (K- Nearest Neighbors) Algorithm



KNN (K- Nearest Neighbors) Algorithm

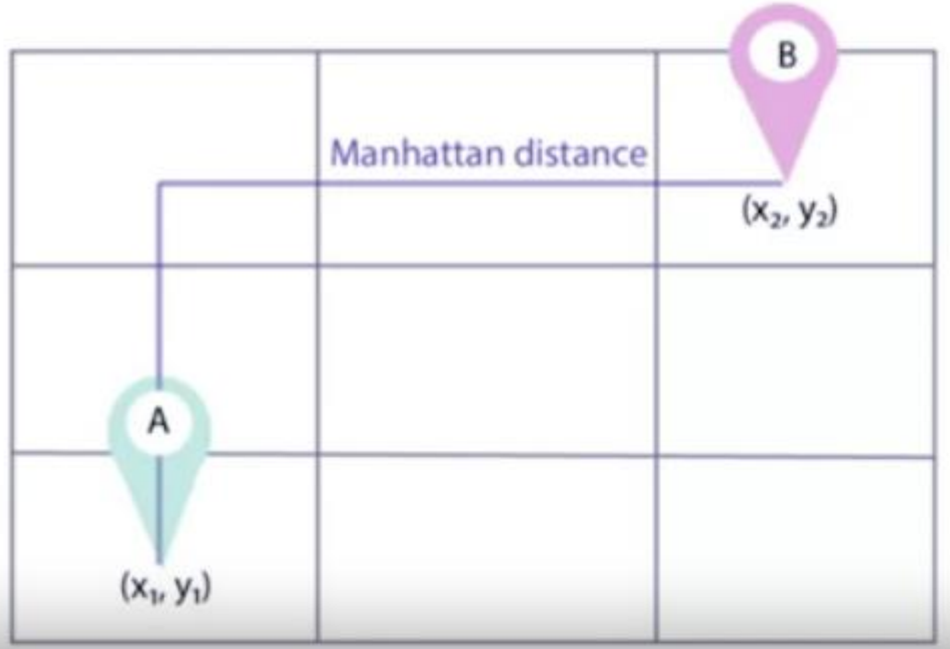
Manhattan distance



KNN (K- Nearest Neighbors) Algorithm

Manhattan distance

$$D_M(A, B) = \sum_{j=1}^{m} \left| X_{(A,j)} - X_{(B,j)} \right|$$



KNN (K- Nearest Neighbors) Algorithm

Hamming distance

$$D_H(A, B) = \sum_{j=1}^{m} \left| X_{(A,j)} - X_{(B,j)} \right|$$

$$X_{(A,j)} = X_{(B,j)} \Rightarrow D_{H_j} = 0$$

$$X_{(A,j)} \neq X_{(B,j)} \Rightarrow D_{H_j} = 1$$