# Module 1

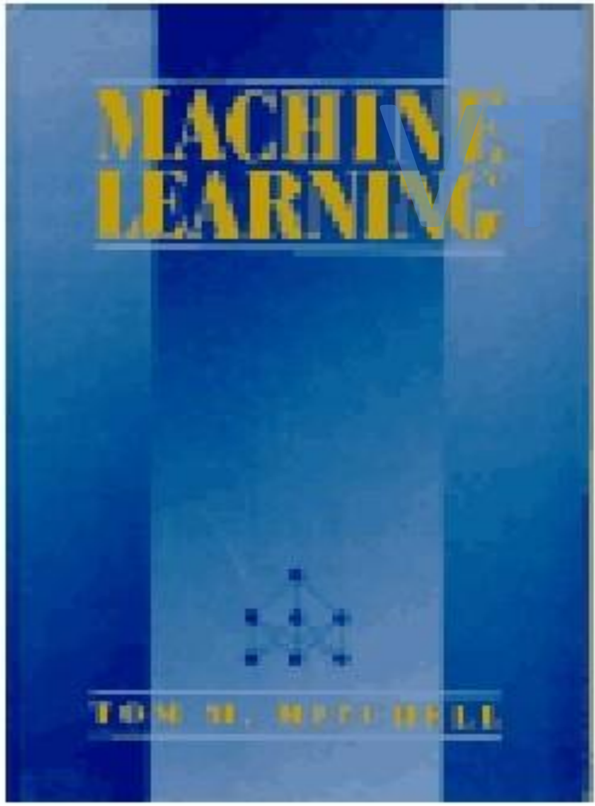
# Introduction to Machine Learning

**Prof. Mahesh G Huddar**

**Dept. of Computer Science and Engineering**

# Books

Machine Learning by Tom M. Mitchell



# Syllabus

**Module – 1** **10 Hours**

Introduction: Well posed learning problems, designing a Learning system, Perspective and Issues in Machine Learning. Concept Learning: Concept learning task, Concept learning as search, Find-S algorithm, Version space, Candidate Elimination algorithm, Inductive Bias.

**Text Book1, Sections: 1.1 – 1.3, 2.1-2.5, 2.7**

**Module – 2** **10 Hours**

Decision Tree Learning: Decision tree representation, Appropriate problems for decision tree learning, Basic decision tree learning algorithm, hypothesis space search in decision tree learning, Inductive bias in decision tree learning, Issues in decision tree learning.

**Text Book1, Sections: 3.1-3.7**

**Module – 3** **08 Hours**

Artificial Neural Networks: Introduction, Neural Network representation, Appropriate problems, Perceptrons, Backpropagation algorithm.

**Text book 1, Sections: 4.1 – 4.6**

**Module – 4** **10 Hours**

Bayesian Learning: Introduction, Bayes theorem, Bayes theorem and concept learning, ML and LS error hypothesis, ML for predicting probabilities, MDL principle, Naive Bayes classifier, Bayesian belief networks, EM algorithm

**Text book 1, Sections: 6.1 – 6.6, 6.9, 6.11, 6.12**

**Module – 5** **12 Hours**

Evaluating Hypothesis: Motivation, Estimating hypothesis accuracy, Basics of sampling theorem, General approach for deriving confidence intervals, Difference in error of two hypothesis, Comparing learning algorithms.

Instance Based Learning: Introduction, k-nearest neighbor learning, locally weighted regression, radial basis function, cased-based reasoning,

Reinforcement Learning: Introduction, Learning Task, Q Learning

**Text book 1, Sections: 5.1-5.6, 8.1-8.5, 13.1-13.3**

# Syllabus - Lab

1. Implement and demonstrate the **FIND-S** algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.
2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination** algorithm to output a description of the set of all hypotheses consistent with the training examples.
3. Write a program to demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.
4. Build an **Artificial Neural Network** by implementing the Backpropagation algorithm and test the same using appropriate data sets.
5. Write a program to implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.
6. Assuming a set of documents that need to be classified, use the **naïve Bayesian Classifier** model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall or your data set.
7. Write a program to construct a **Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.
8. Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set for clustering using **k-Means** algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.
9. Write a program to implement **k-Nearest Neighbor algorithm** to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.
10. Implement the non-parametric **Locally Weighted Regression** algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

# MACHINE LEARNING

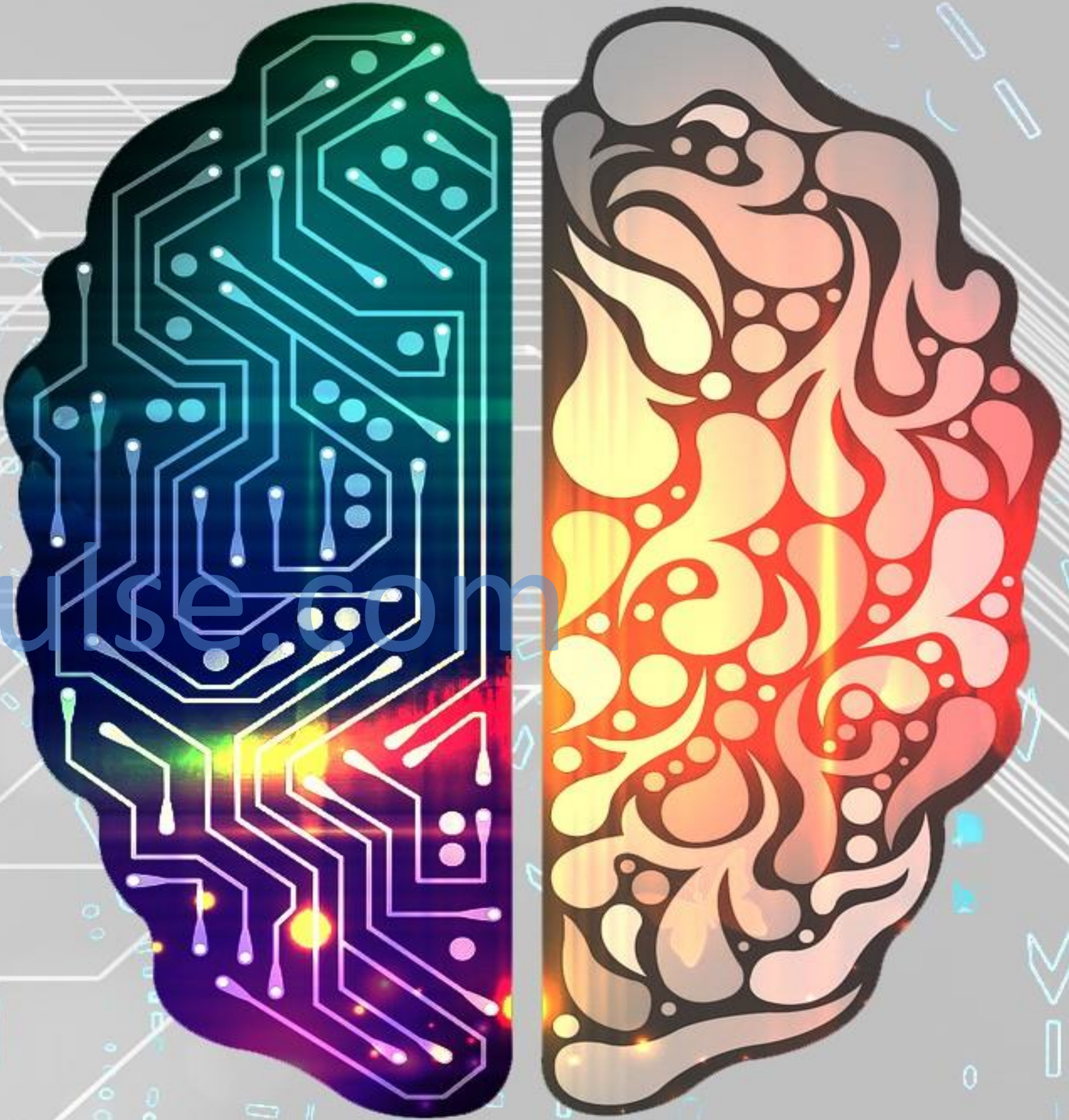


Image hosted by: WittySparks.com | Image source: Pixabay.com

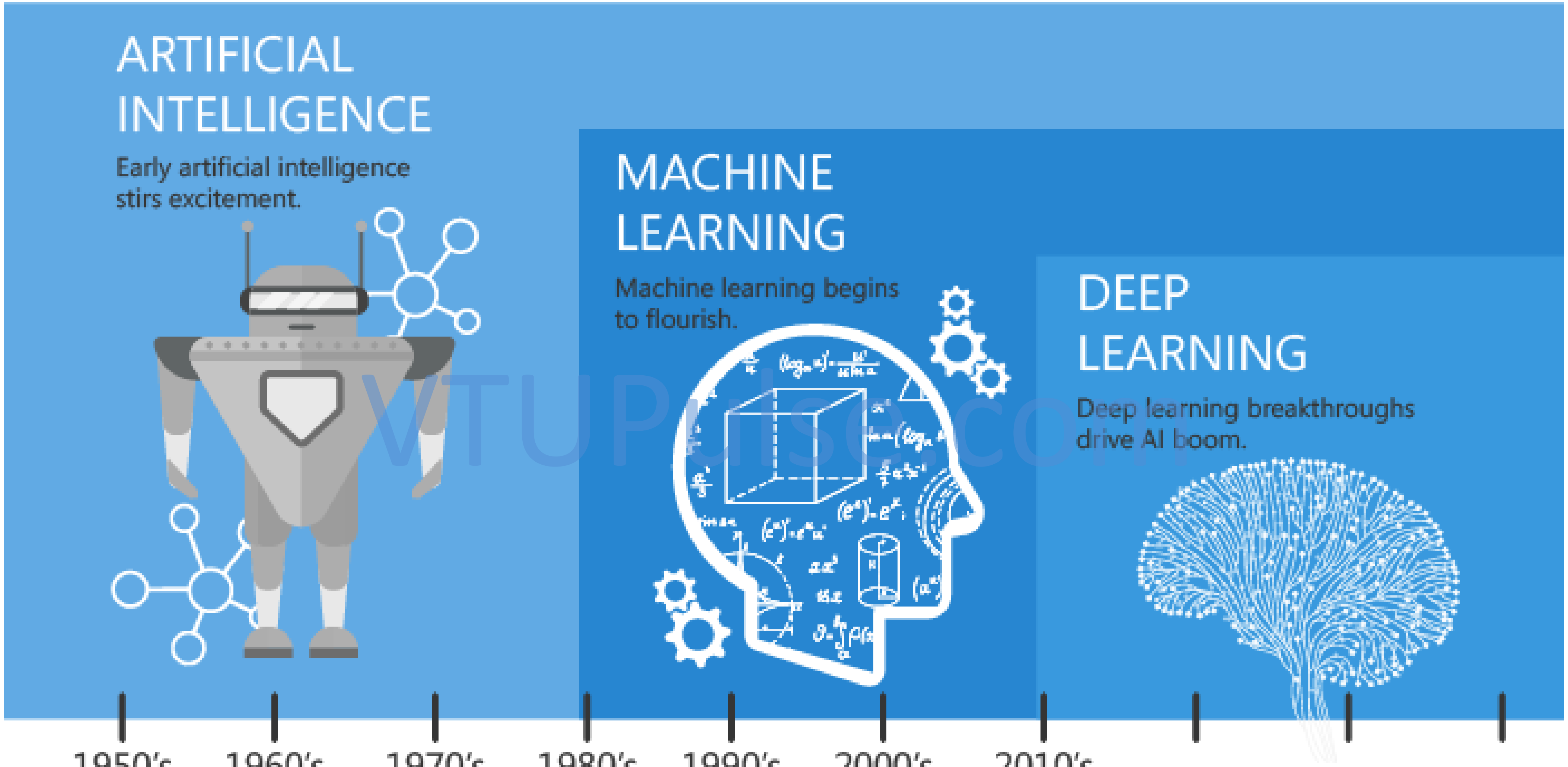
# A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
- “Machine learning is the next Internet”  (Tony Tether, Director, DARPA)
- Machine learning is the hot new thing”  (John Hennessy, President, Stanford)
- “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)
- “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)
- “Machine learning is today’s discontinuity” (Jerry Yang, CEO, Yahoo)

# Introduction to Machine Learning

- We have seen Machine Learning as a buzzword for the past few years, the reason for this might be the high amount of data production by applications, the increase of computation power in the past few years and the development of better algorithms.
- You may already be using a device or application that utilizes it.
- For example, GMAIL, WhatsApp, E-Commerce Website, Video Sharing Platforms, a wearable fitness tracker like Fitbit, or an intelligent home assistant like Google Home.

# History of Machine Learning



ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.

MACHINE LEARNING

Machine learning begins to flourish.

DEEP LEARNING

Deep learning breakthroughs drive AI boom.

Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

# What is Machine Learning..?

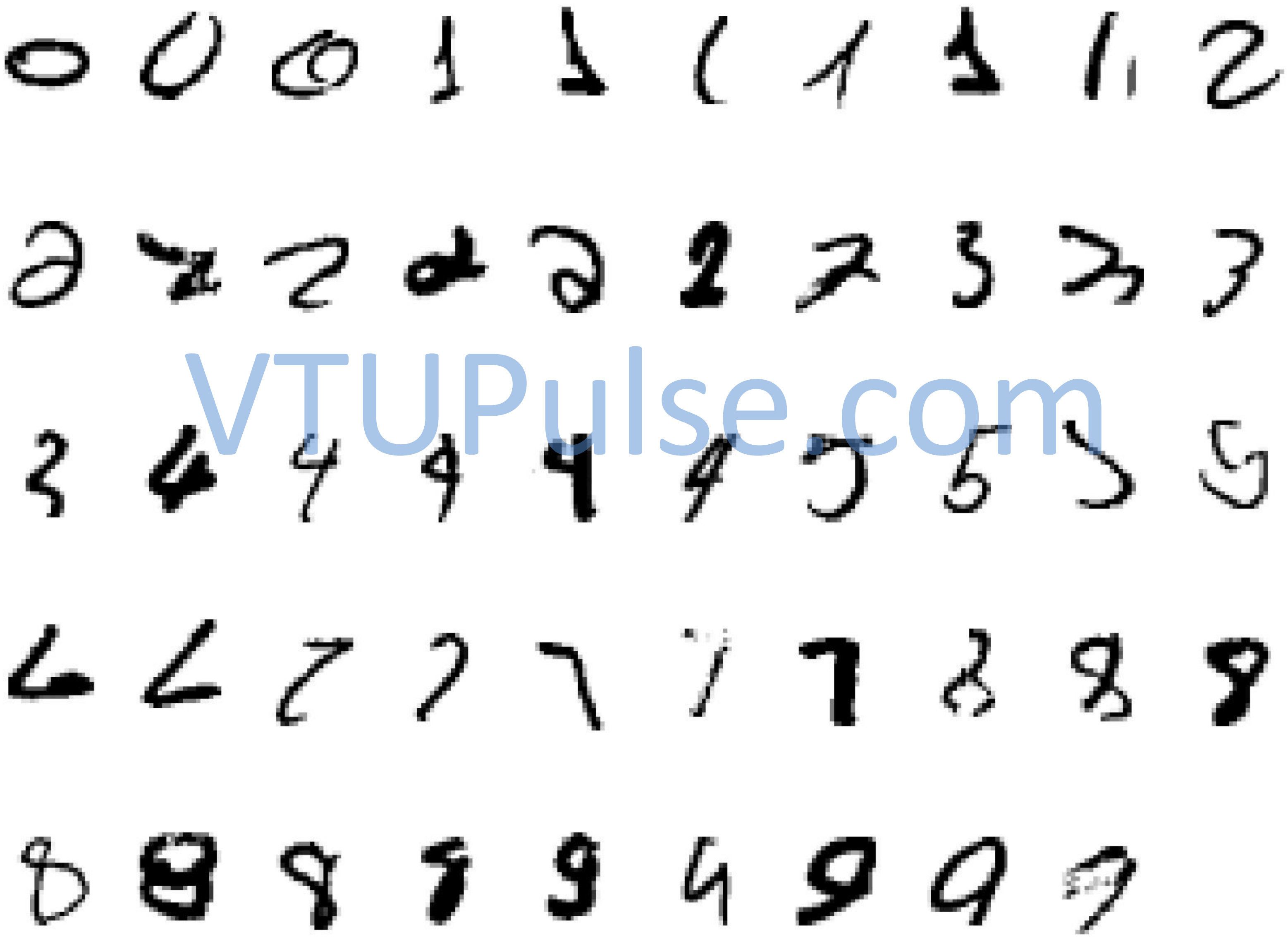
- “Learning is any process by which a system improves performance from experience.” - Herbert Simon

- A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

# What is Machine Learning..?

- Definition by Tom Mitchell (1998):
- Machine Learning is the study of algorithms that
  - improve their performance P
  - at some task T
  - with experience E.
- A well-defined Machine Learning task is given by <P, T, E>.

# A classic example of a task that requires machine learning



# Handwritten Digit Recognition Problem

- **Task T:** Recognizing and Classifying handwritten words within images
- **Performance P:** percent of words correctly classified
- **Experience E:** a database of handwritten words with given classifications

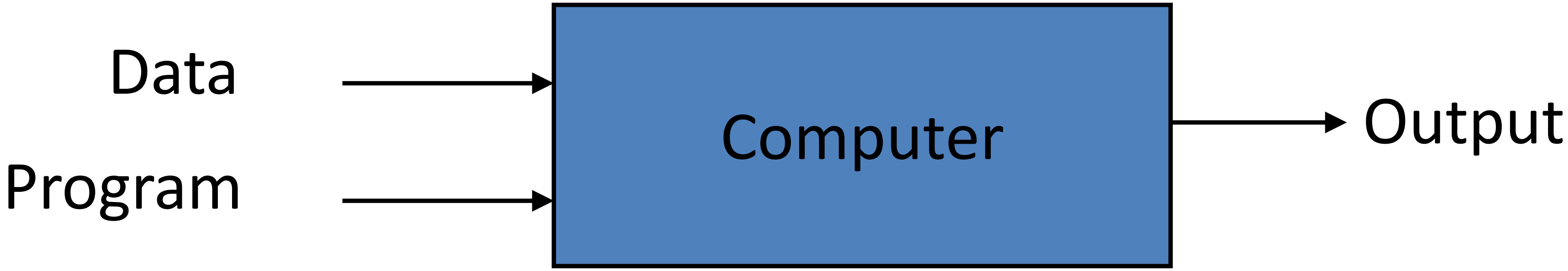
# A robot driving learning problem

- **Task T:** driving on public four-lane highways using vision sensors
- **Performance P:** average distance traveled before an error (as judged by human overseer)
- **Experience E:** a sequence of images and steering commands recorded while observing a human driver

# A checkers learning problem

- **Task T:** playing checkers
- **Performance P:** percent of games won against opponents
- **Experience E:** playing practice games against itself

## Traditional Programming



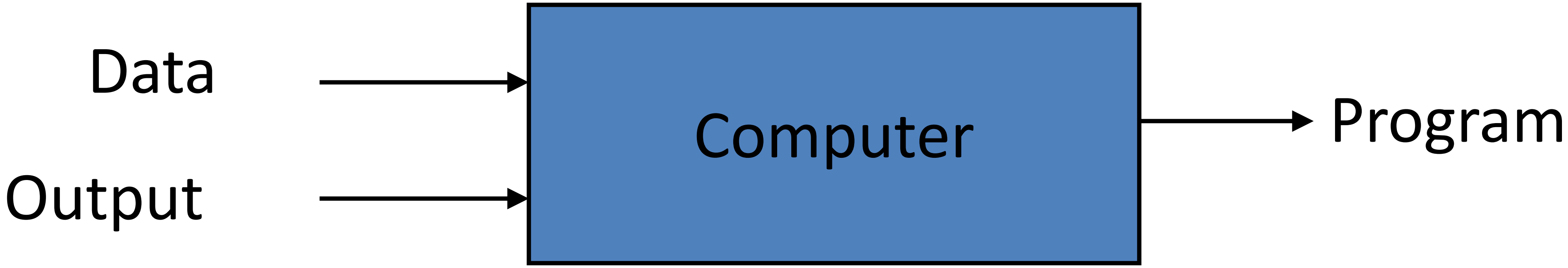
Data

Program

Computer

Output

## Machine Learning



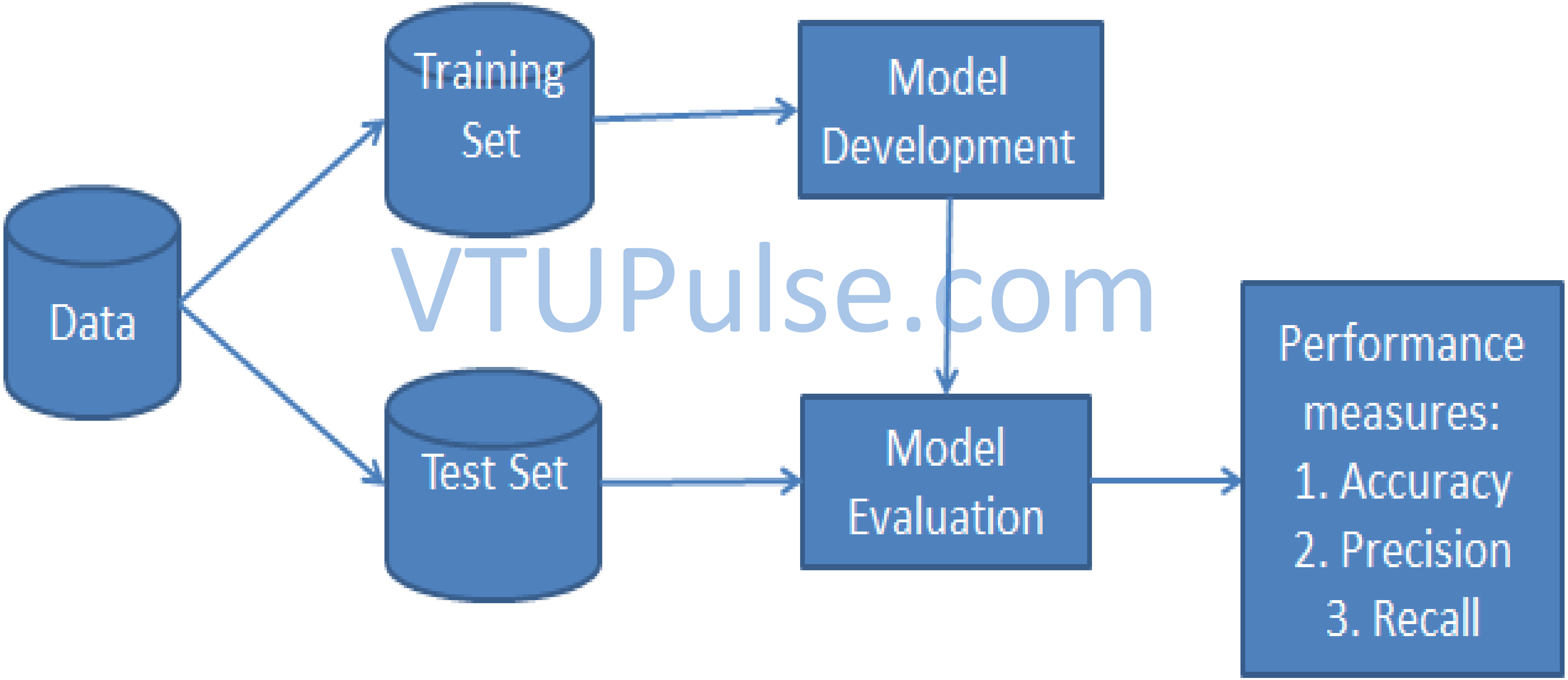
Data

Output

Computer

Program

# How Machine Learning Works?

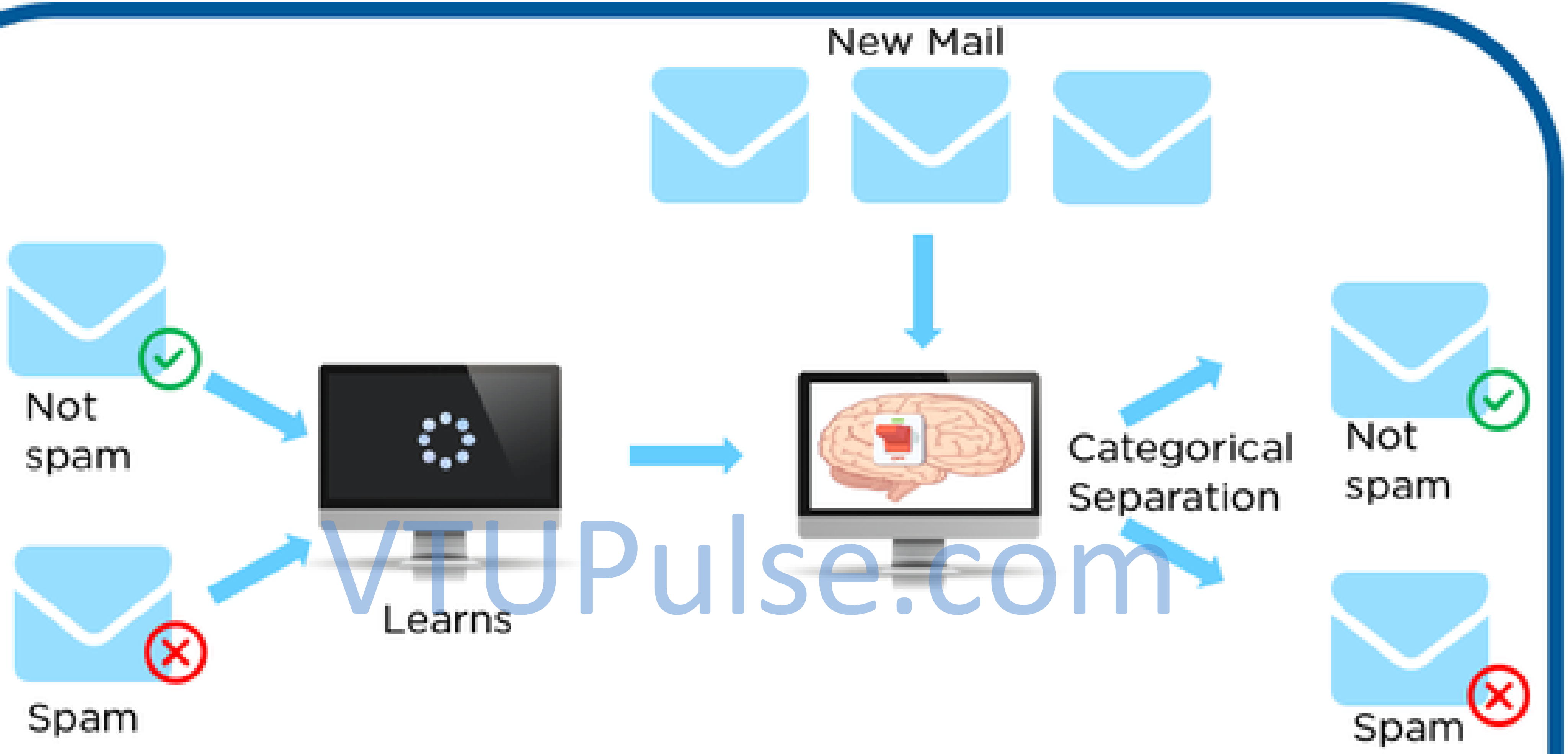


# Types of Learning Algorithms

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**

# Supervised Learning

- In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label.
- The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.
- As shown in the example, we have initially taken some data and marked them as ‘Spam’ or ‘Not Spam’. This labeled data is used by the training supervised model, this data is used to train the model.
- Once it is trained we can test our model by testing it with some test new mails and checking of the model is able to predict the right output.

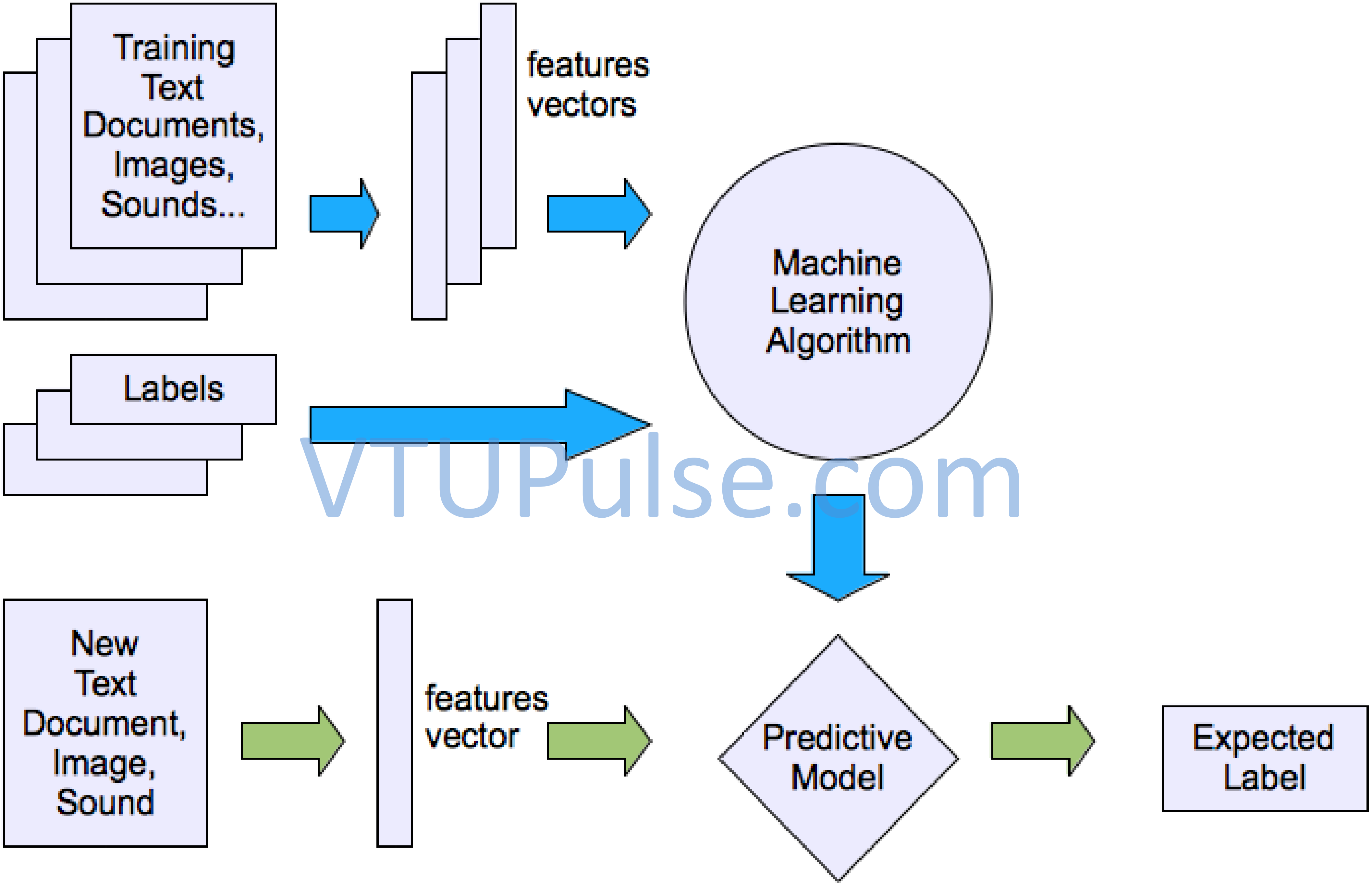


Enables the machine to be trained to classify observations into some class

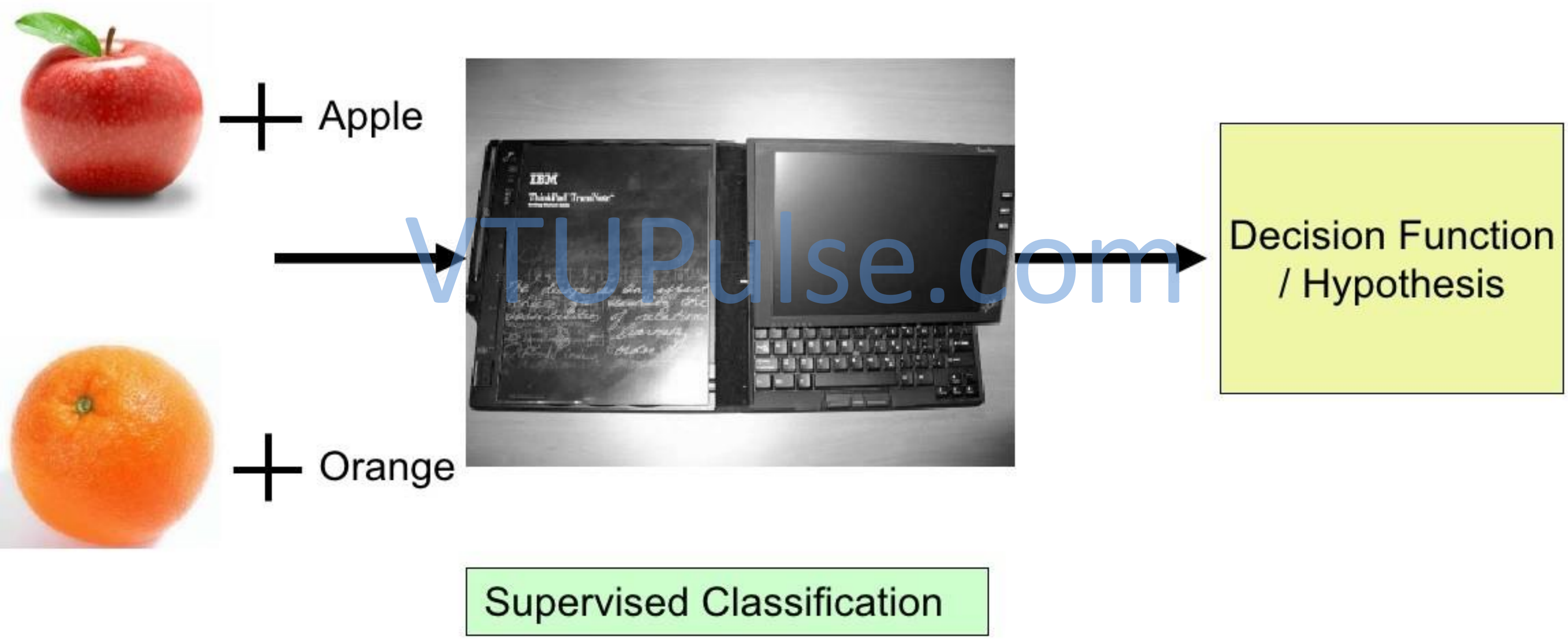
# Types of Supervised learning

- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.


Training Text Documents, Images, Sounds...
features vectors
Machine Learning Algorithm
Labels
New Text Document, Image, Sound
features vector
Predictive Model
Expected Label

# Supervised Learning



Apple

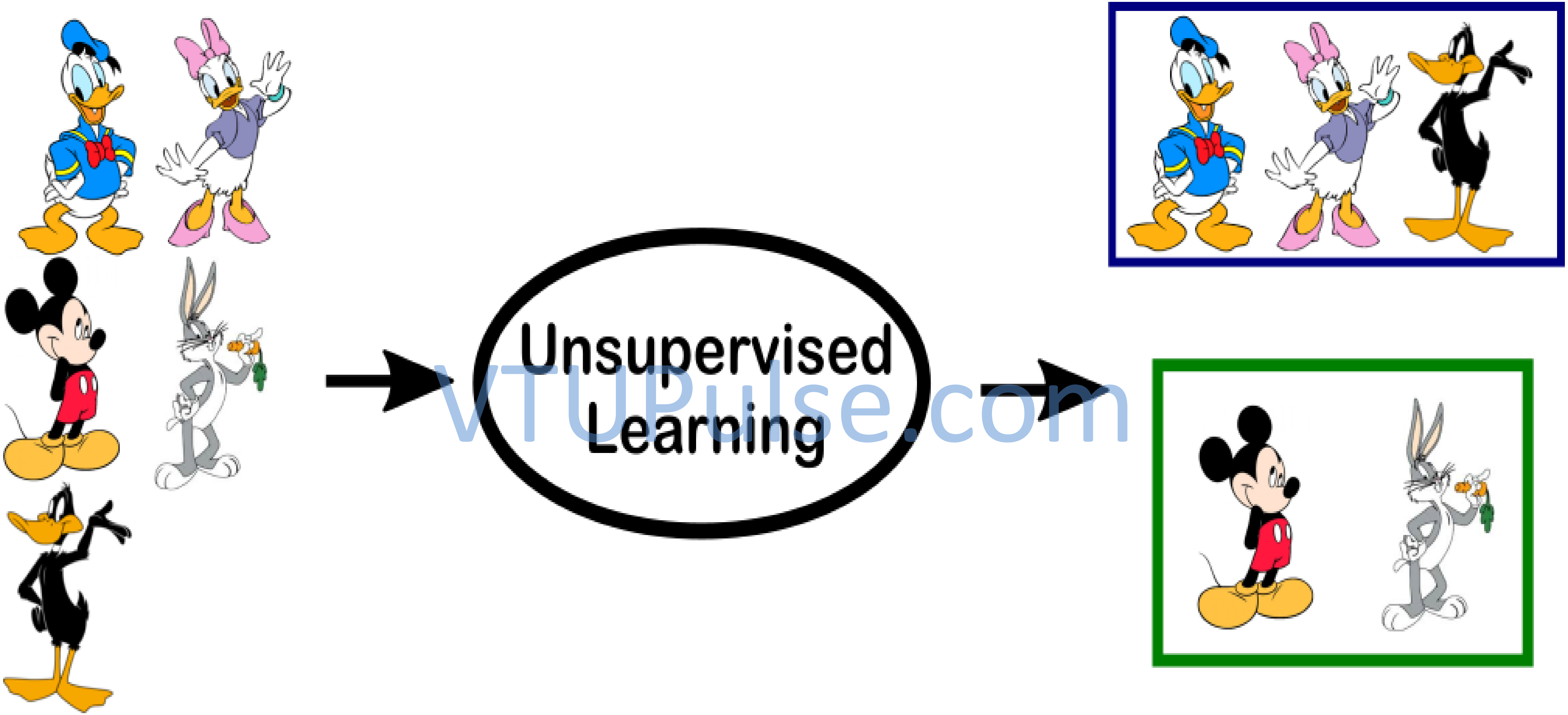
Orange

Decision Function / Hypothesis

Supervised Classification

# Unsupervised Learning Algorithm

- In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training.
- In the example, we have given some characters to our model which are 'Ducks' and 'Not Ducks'.
- In our training data, we don't provide any label to the corresponding data.
- The unsupervised model is able to separate both the characters by looking at the type of data and models the underlying structure or distribution in the data in order to learn more about it.

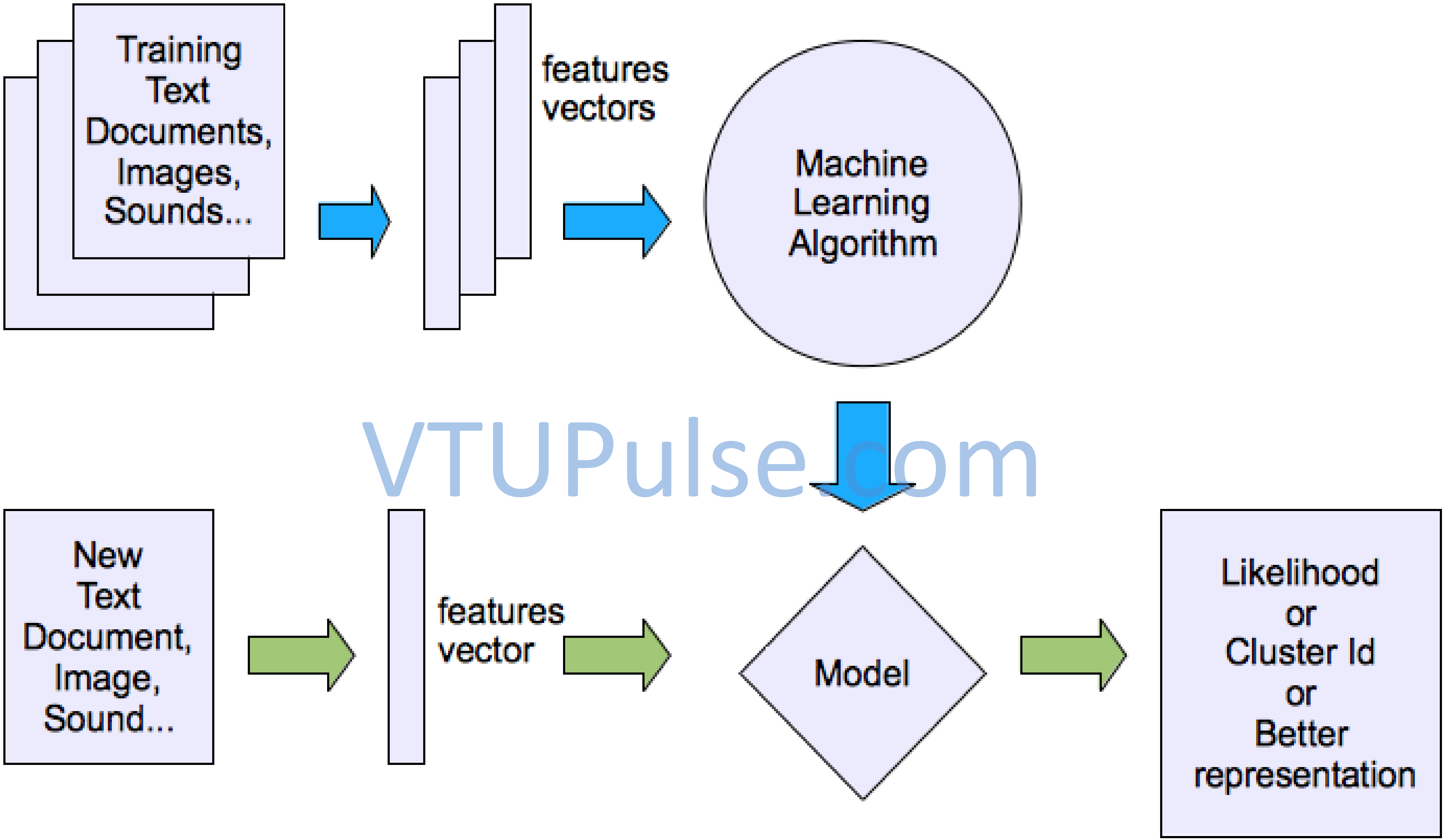


Unsupervised Learning

# Types of Unsupervised learning

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.


Training Text Documents, Images, Sounds...
features vectors
Machine Learning Algorithm
New Text Document, Image, Sound...
features vector
Model
Likelihood or Cluster Id or Better representation

# Unsupervised Learning

Decision Function / Hypothesis

Unsupervised Classification

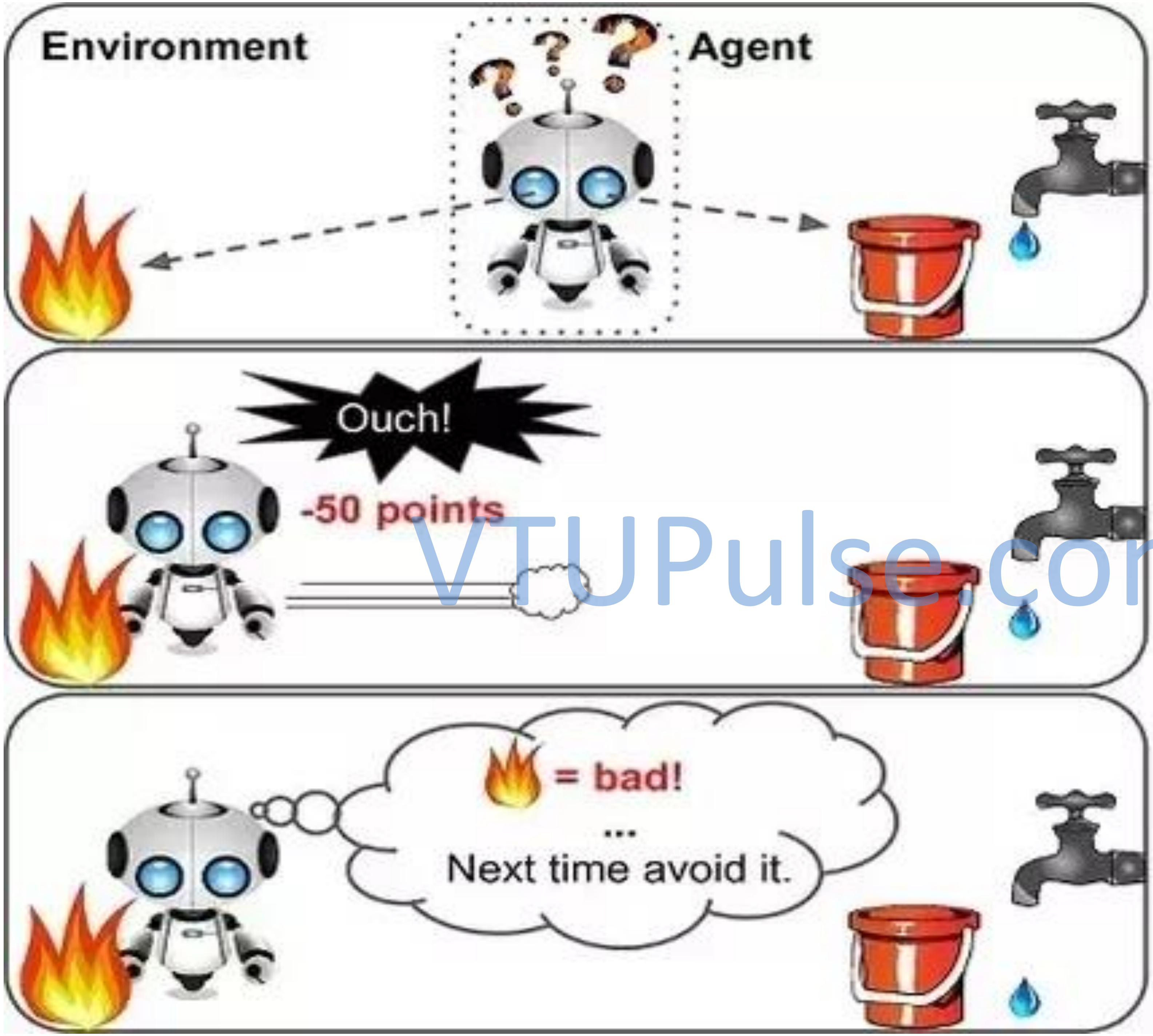
# Reinforcement Learning

- A reinforcement learning algorithm, or agent, learns by interacting with its environment.
- The agent receives rewards by performing correctly and penalties for performing incorrectly.
- The agent learns without intervention from a human by maximizing its reward and minimizing its penalty.
- It is a type of dynamic programming that trains algorithms using a system of reward and punishment.

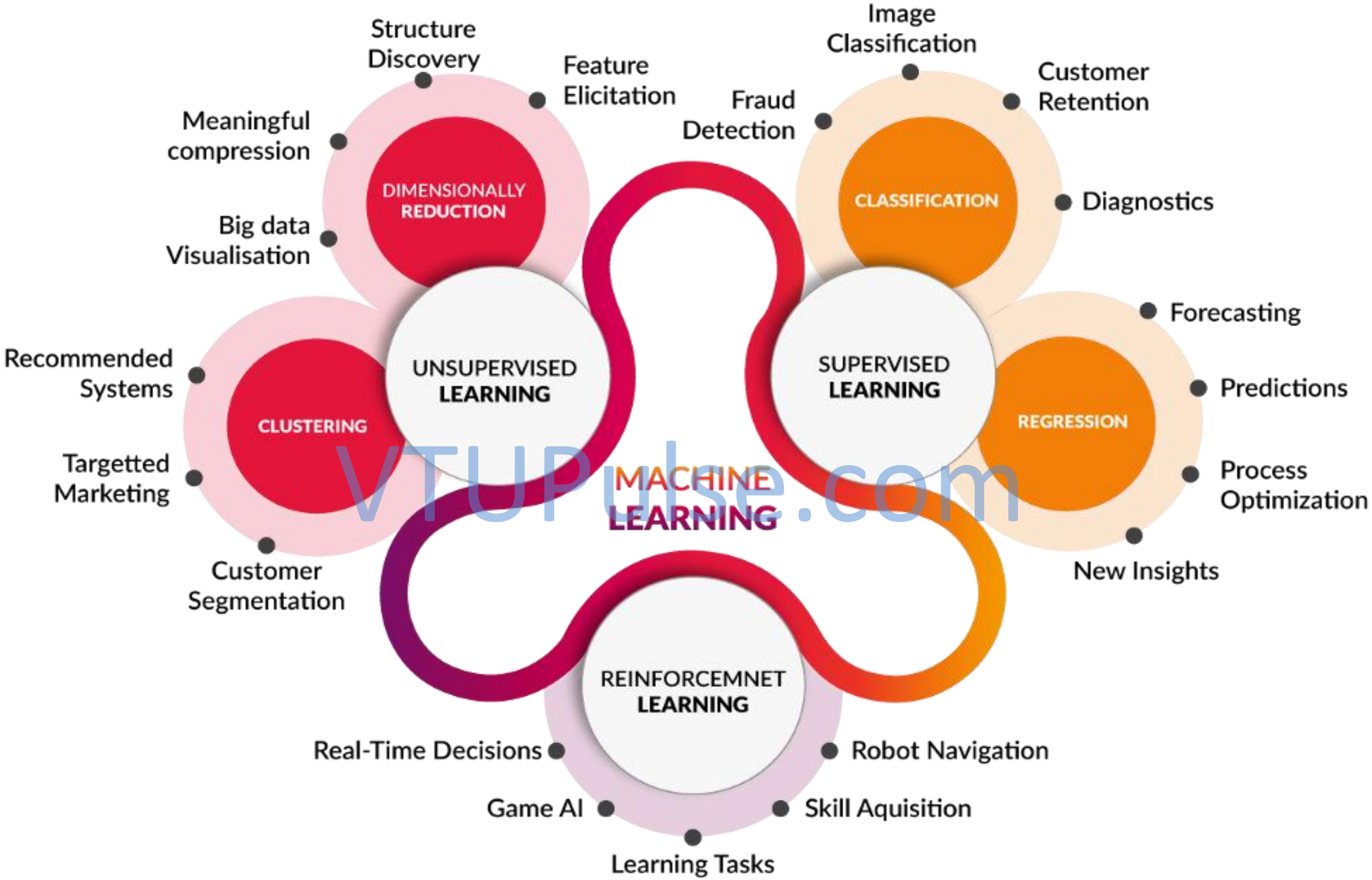
# Reinforcement Learning

- In the example, we can see that the agent is given 2 options i.e. a path with water or a path with fire.
- A reinforcement algorithm works on reward a system i.e. if the agent uses the fire path then the rewards are subtracted and agent tries to learn that it should avoid the fire path.

- If it had chosen the water path or the safe path then some points would have been added to the reward points, the agent then would try to learn what path is safe and what path isn’t.
- It is basically leveraging the rewards obtained, the agent improves its environment knowledge to select the next action.



1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found


Structure Discovery
Feature Elicitation
Meaningful compression
DIMENSIONALLY REDUCTION
Big data Visualisation
Image Classification
Customer Retention
Fraud Detection
CLASSIFICATION
Diagnostics
UNSUPERVISED LEARNING
SUPERVISED LEARNING
Recommended Systems
CLUSTERING
Targetted Marketing
Customer Segmentation
Forecasting
Predictions
REGRESSION
Process Optimization
New Insights
MACHINE LEARNING
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

# Applications of Machine Learning

- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Generating patterns:
  - Generating images or motion sequences
- Recognizing anomalies:
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant or unusual sound in your car engine.
- Prediction:
  - Future stock prices or currency exchange rates

# Influence of Disciplines on Machine Learning

**Artificial intelligence**

- Learning symbolic representations of concepts. Machine learning as a search problem. Learning as an approach to improving problem solving. Using prior knowledge together with training data to guide learning.

**Bayesian methods**

- Bayes' theorem as the basis for calculating probabilities of hypotheses. The naive Bayes classifier. Algorithms for estimating values of unobserved variables.

**Computational complexity theory**

- Theoretical bounds on the inherent complexity of different learning tasks, measured in terms of the computational effort, number of training examples, number of mistakes, etc. required in order to learn.

**Control theory**

- Procedures that learn to control processes in order to optimize predefined objectives and that learn to predict the next state of the process they are controlling.

# Influence of Disciplines on Machine Learning

**Information theory**

- Measures of entropy and information content. Minimum description length approaches to learning. Optimal codes and their relationship to optimal training sequences for encoding a hypothesis.

**Philosophy**

- Occam's razor, suggesting that the simplest hypothesis is the best. Analysis of the justification for generalizing beyond observed data.

**Psychology and neurobiology**

- The power law of practice, which states that over a very broad range of learning problems,
- people's response time improves with practice according to a power law. Neurobiological studies motivating artificial neural network models of learning.

**Statistics**

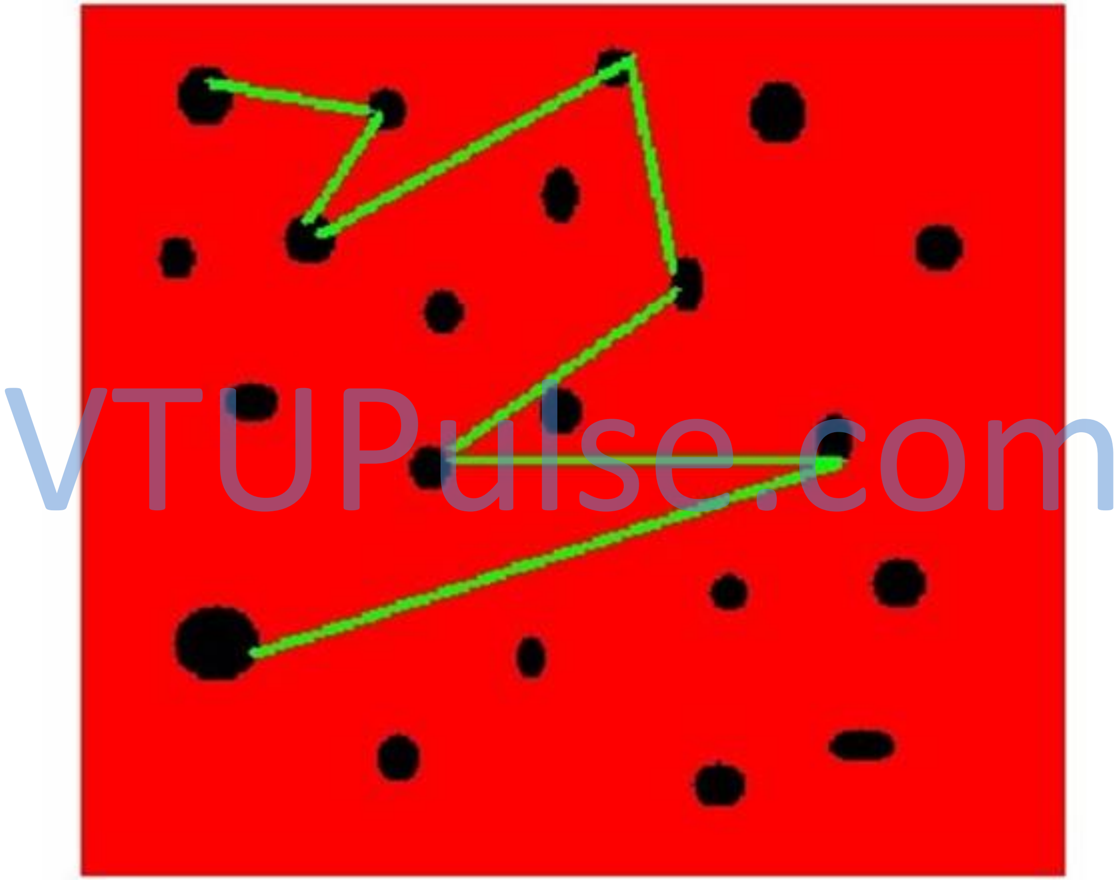
- Characterization of errors (e.g., bias and variance) that occur when estimating the accuracy of a hypothesis based on a limited sample of data. Confidence intervals, statistical tests.

# DESIGNING A LEARNING SYSTEM

# CONCEPT LEARNING

- The problem of inducing general functions from specific training examples is central to learning.
- Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- What is Concept Learning…?
- **“A task of acquiring potential hypothesis (solution) that best fits the given training examples.”**

# CONCEPT LEARNING



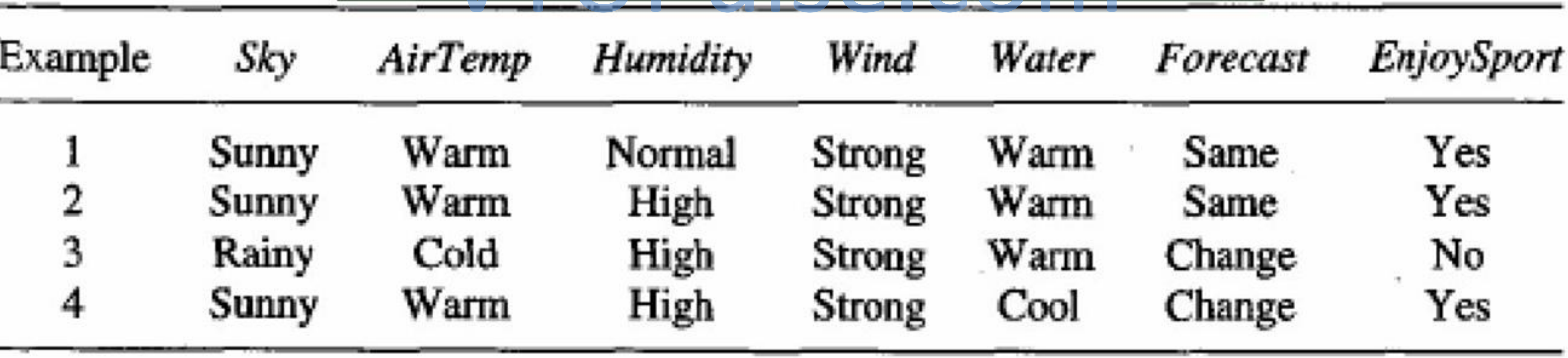
# A CONCEPT LEARNING TASK

- Consider the example task of learning the target concept "**days on which XYZ enjoys his favorite water sport.**"
- Table describes a set of example days, each represented by a set of ***attributes.*** The attribute ***EnjoySport*** indicates whether or not XYZ enjoys his favorite water sport on this day.
- The task is to learn to predict the value of ***EnjoySport*** for an arbitrary day, based on the values of its other attributes.

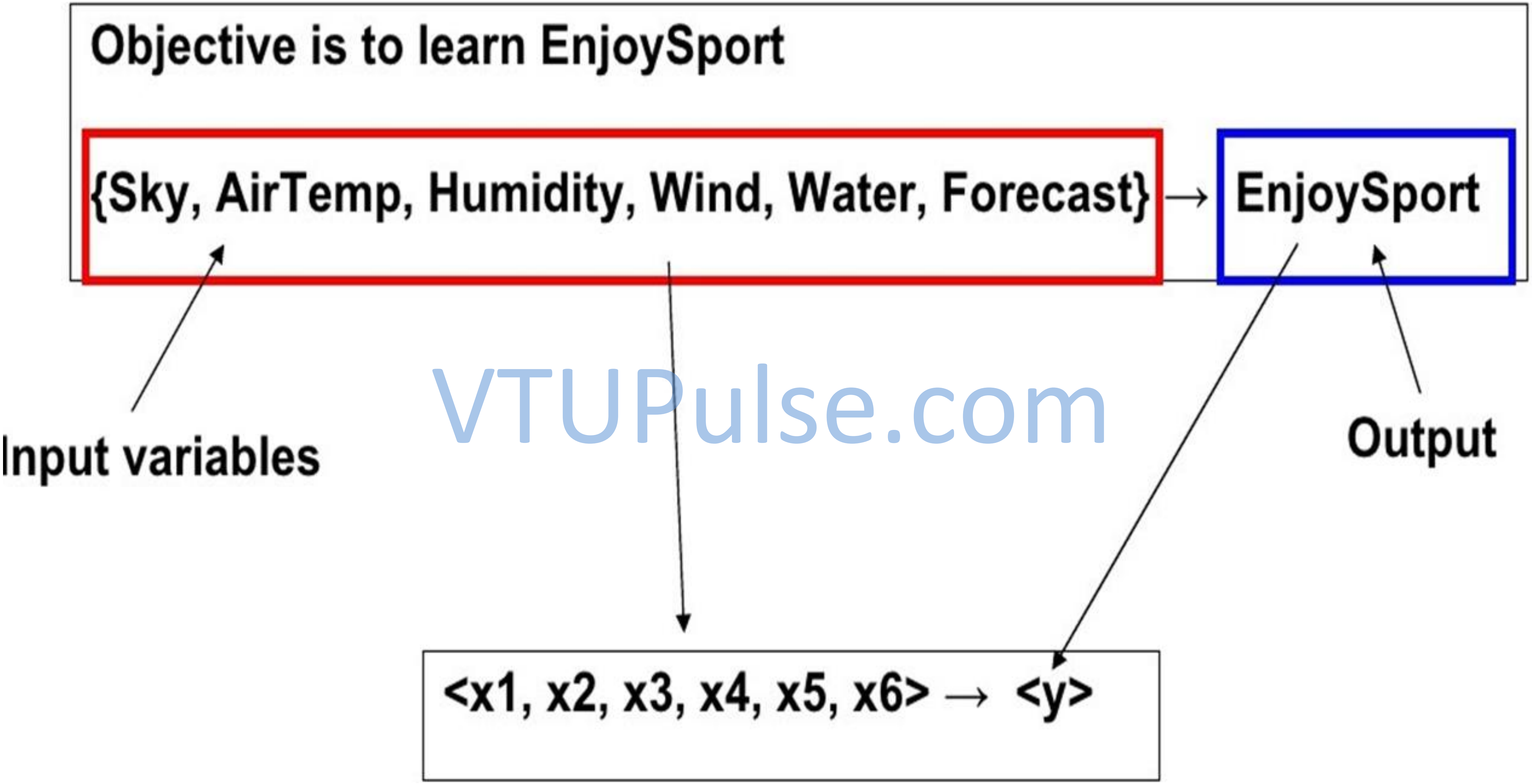
**Objective is to learn EnjoySport**

**{Sky, AirTemp, Humidity, Wind, Water, Forecast} → EnjoySport**

**Tom enjoys his favorite water sports**



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |



Objective is to learn EnjoySport

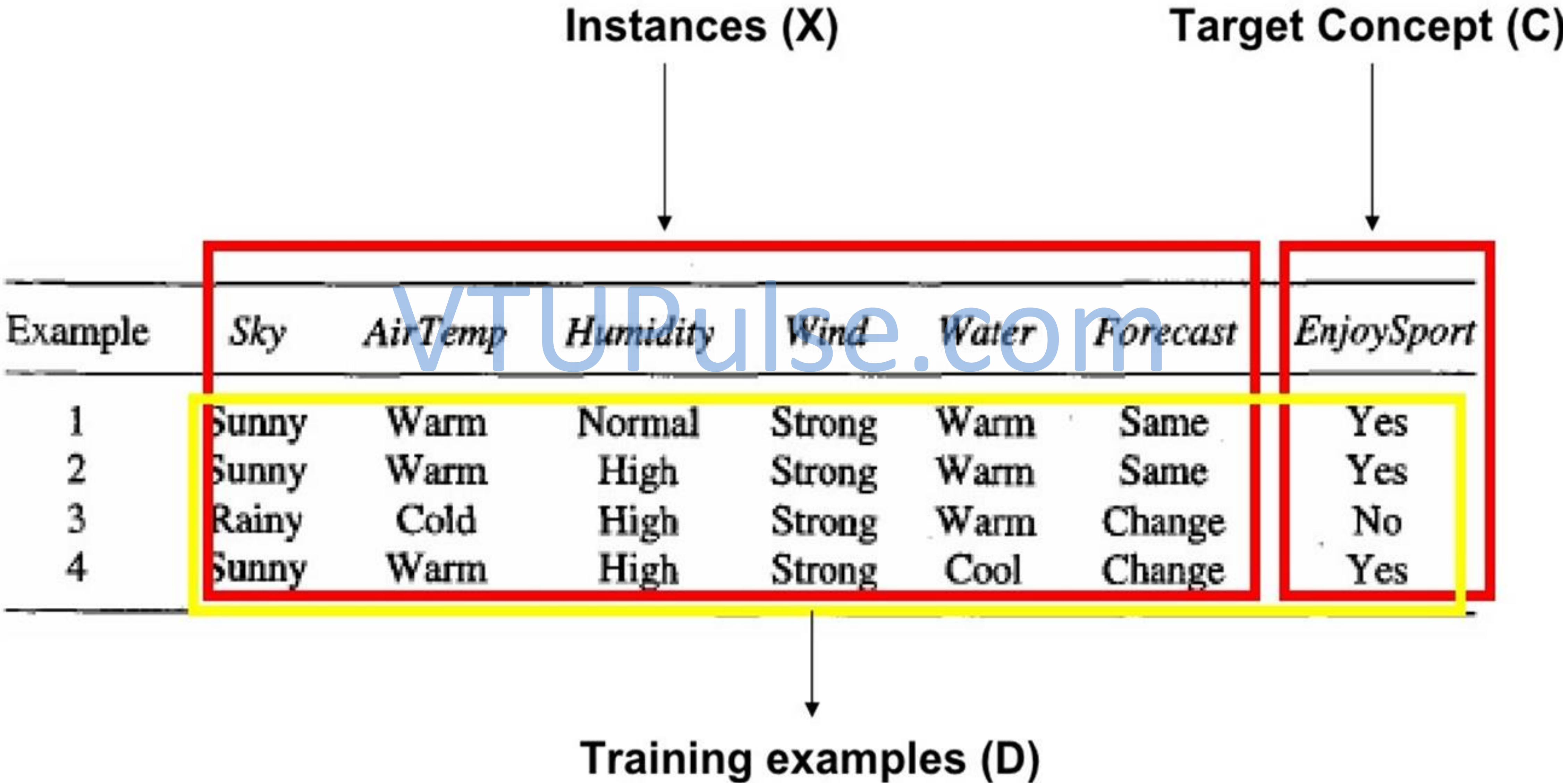
{Sky, AirTemp, Humidity, Wind, Water, Forecast} → EnjoySport

Input variables

Output

<x1, x2, x3, x4, x5, x6> → <y>

# A CONCEPT LEARNING TASK - Notation



Instances (X)

Target Concept (C)

| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

Training examples (D)

# A CONCEPT LEARNING TASK

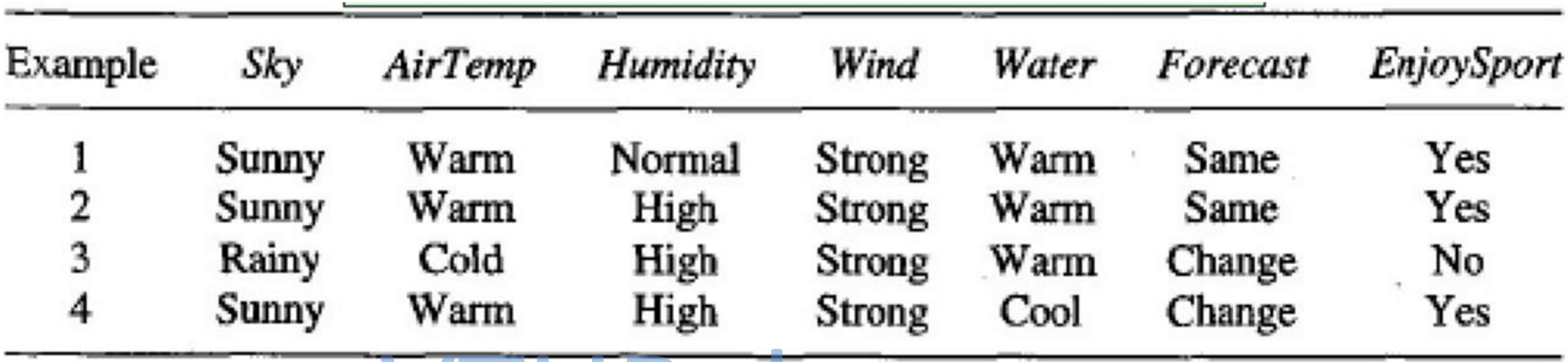
- What hypothesis representation shall we provide to the learner in this case?
- Let us begin by considering a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.
- In particular, let each hypothesis be a vector of six constraints, specifying the values of the six attributes **Sky, AirTemp, Humidity, Wind, Water,** and **Forecast.**

- For each attribute, the hypothesis will either
  - indicate by a **"?'** that any value is acceptable for this attribute,
  - specify a single required value (e.g., **Warm)** for the attribute, or
  - indicate by a **"ø"** that no value is acceptable.

# A CONCEPT LEARNING TASK

- If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (h(x) = 1).
- To illustrate, the hypothesis that Prabhas enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression

# A CONCEPT LEARNING TASK



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

1. **(Sunny, Warm, Normal, Strong, Warm, Same)**

2. **(Rainy, Warm, High, Strong, Warm, Same)**

3. **(Rainy, Cold, High, Strong, Warm, Change)**

4. **(?, Cold, High, ?, ?, ?)**

# A CONCEPT LEARNING TASK

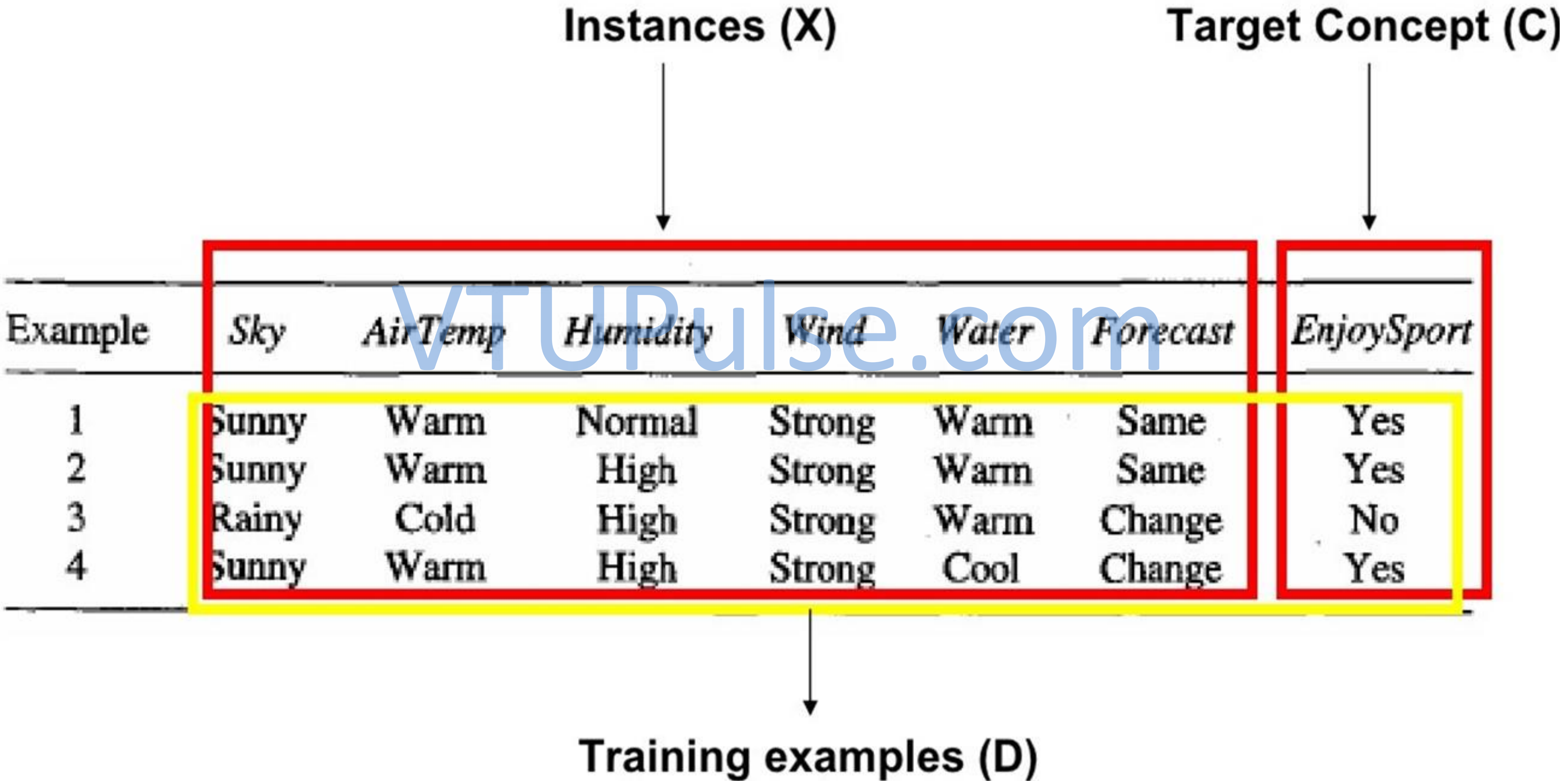
- The most general hypothesis-that every day is a positive example-is represented by

**(?, ?, ?, ?, ?, ?)**

- and the most specific possible hypothesis-that no day is a positive example-is represented by

**(ø, ø, ø, ø, ø, ø)**

# A CONCEPT LEARNING TASK - Notation



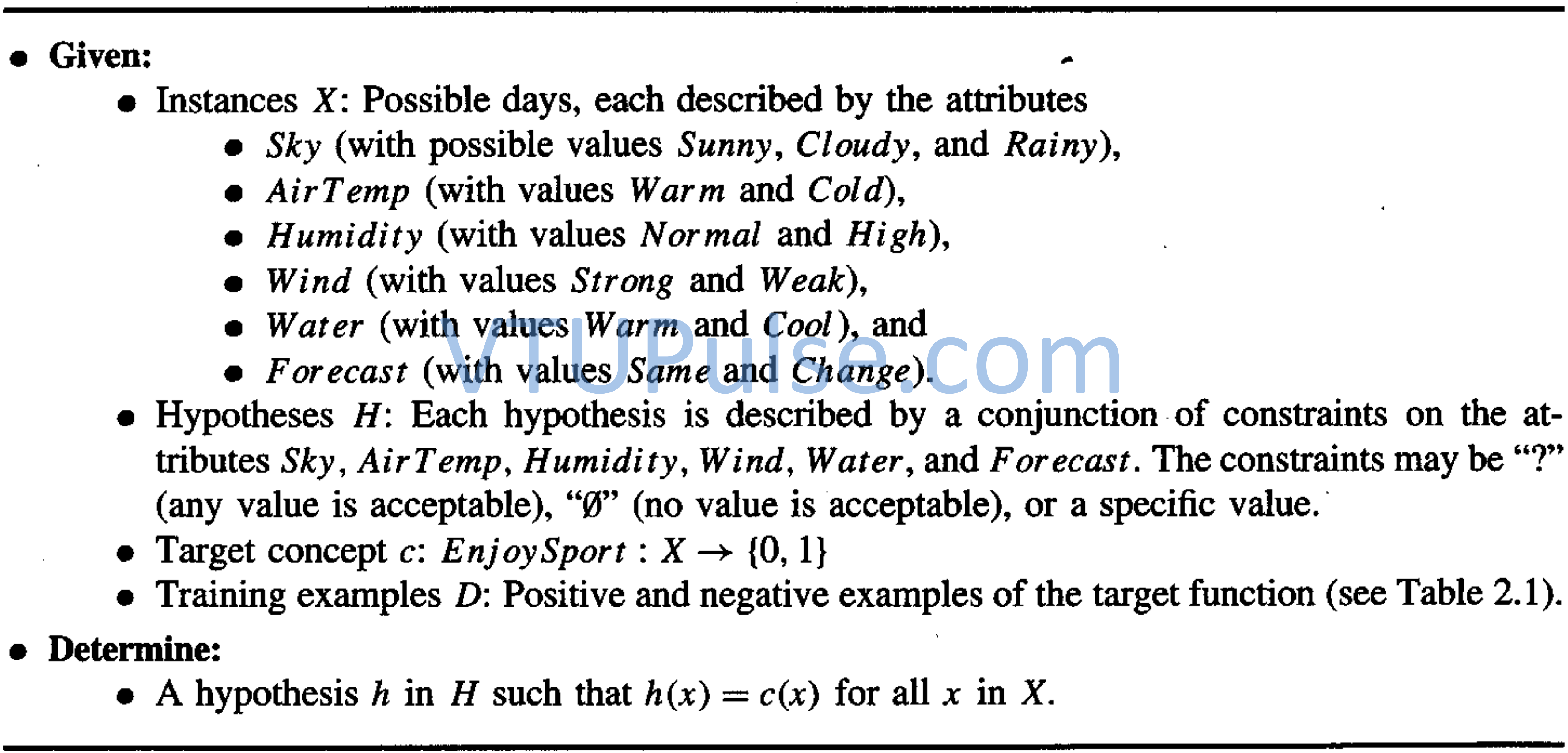
Instances (X)

Target Concept (C)

| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

Training examples (D)

# A CONCEPT LEARNING TASK - Notation



- **Given:**
  - Instances $X$: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses $H$: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept $c$: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples $D$: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis $h$ in $H$ such that $h(x) = c(x)$ for all $x$ in $X$.

# A CONCEPT LEARNING TASK - Search

- Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- The goal of this search is to find the hypothesis that best fits the training examples.

- It is important to note that by selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

# A CONCEPT LEARNING TASK - Search

## Instance Space:

- Consider, for example, the instances X and hypotheses H in the ***EnjoySport*** learning task.
- Given that the attribute ***Sky*** has three possible values, and that ***AirTemp, Humidity, Wind, Water,*** and ***Forecast*** each have two possible values, the instance space X contains exactly **3 . 2 . 2 . 2 . 2 . 2 = 96** distinct instances.

# A CONCEPT LEARNING TASK - Search

F1 - > A, B

F2 - > X, Y

**3 * 3 = 9 + 1 = 10**

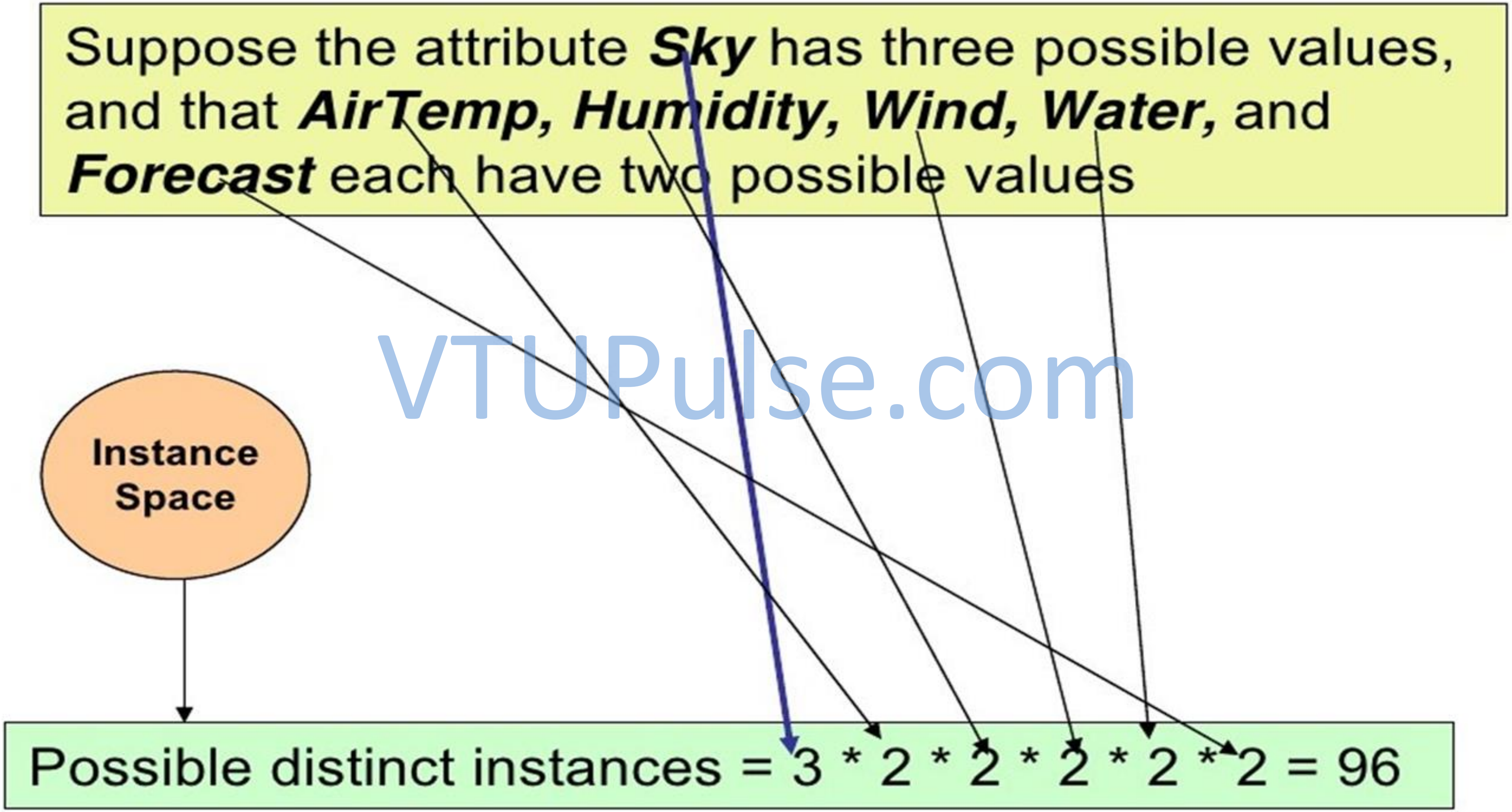
(?, ø)

**Instance Space:** (A, X), (A, Y), (B, X), (B, Y) – **4 Instances**

**Hypothesis Space:** (A, X), (A, Y), (A, ø), (A, ?), (B, X), (B, Y), (B, ø), (B, ?), (ø, X), (ø, Y), (ø, ø), (ø, ?), (?, X), (?, Y), (?, ø), (?, ?) - **16**

**Hypothesis Space:** (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y (?, ?) (ø, ø)- **10**

# A CONCEPT LEARNING TASK – Instance Space



Suppose the attribute ***Sky*** has three possible values, and that ***AirTemp, Humidity, Wind, Water,*** and ***Forecast*** each have two possible values

**Instance Space**

Possible distinct instances = 3 * 2 * 2 * 2 * 2 * 2 = 96

# A CONCEPT LEARNING TASK - Search

## Hypothesis Space

- Similarly there are **5 . 4 . 4 . 4 . 4 . 4 = 5120** syntactically distinct hypotheses within H.
- Notice, however, that every hypothesis containing one or more "**ø**" symbols represents the empty set of instances; that is, it classifies every instance as negative.
- Therefore, the number of semantically distinct hypotheses is only **1 + (4 . 3 . 3 . 3 . 3 . 3) = 973.**
- Our EnjoySport example is a very simple learning task, with a relatively small, finite hypothesis space.

# A CONCEPT LEARNING TASK – Hypothesis Space

Hypothesis Space: A set of all possible hypotheses

Possible syntactically distinct Hypotheses for *EnjoySport*

= 5 * 4 * 4 * 4 * 4 * 4 = 5120

- Sky has three possible values
- Fourth value don't care (?)
- Fifth value is empty set Ø

# General-to-Specific Ordering of Hypotheses

- To illustrate the general-to-specific ordering, consider the two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)

  h2 = (Sunny, ?, ?, ?, ?, ?)

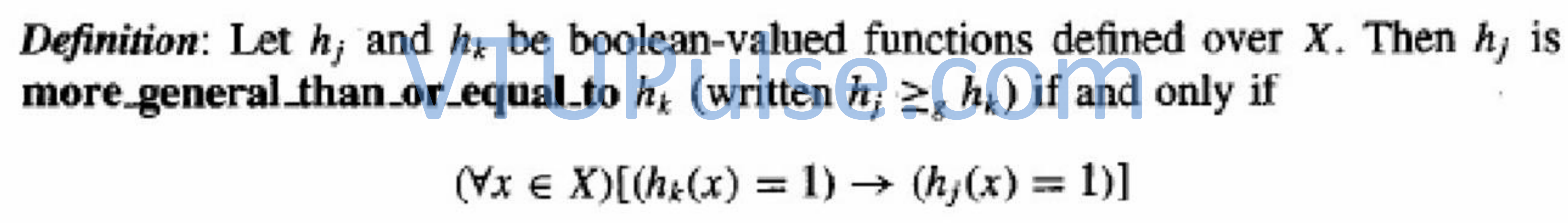
- Now consider the sets of instances that are classified positive by hl and by h2. Because h2 imposes fewer constraints on the instance, it classifies more instances as positive.
- In fact, any instance classified positive by h1 will also be classified positive by h2. Therefore, we say that h2 is more general than h1.

# More General Than hypothesis

- For any instance x in X and hypothesis h in H, we say that x satisjies h if and only if h(x) = 1.
- We define the ***more_general_than_or_equal***e_to relation in terms of the sets of instances that satisfy the two hypotheses:

# More General Than hypothesis

Given hypotheses **hj** and **hk**, **hj** is **more_general_than_or_equal_to hk** if and only if any instance that satisfies **hk** also satisfies ***hj.***



***Definition***: Let $h_j$ and $h_k$ be boolean-valued functions defined over $X$. Then $h_j$ is **more_general_than_or_equal_to** $h_k$ (written $h_j \geq_g h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

We can also say that ***hj*** is ***more_specific_than hk*** when ***hk*** is ***more_general_than hj.***

S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI.

*Inculcating Values, Promoting Prosperity*

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

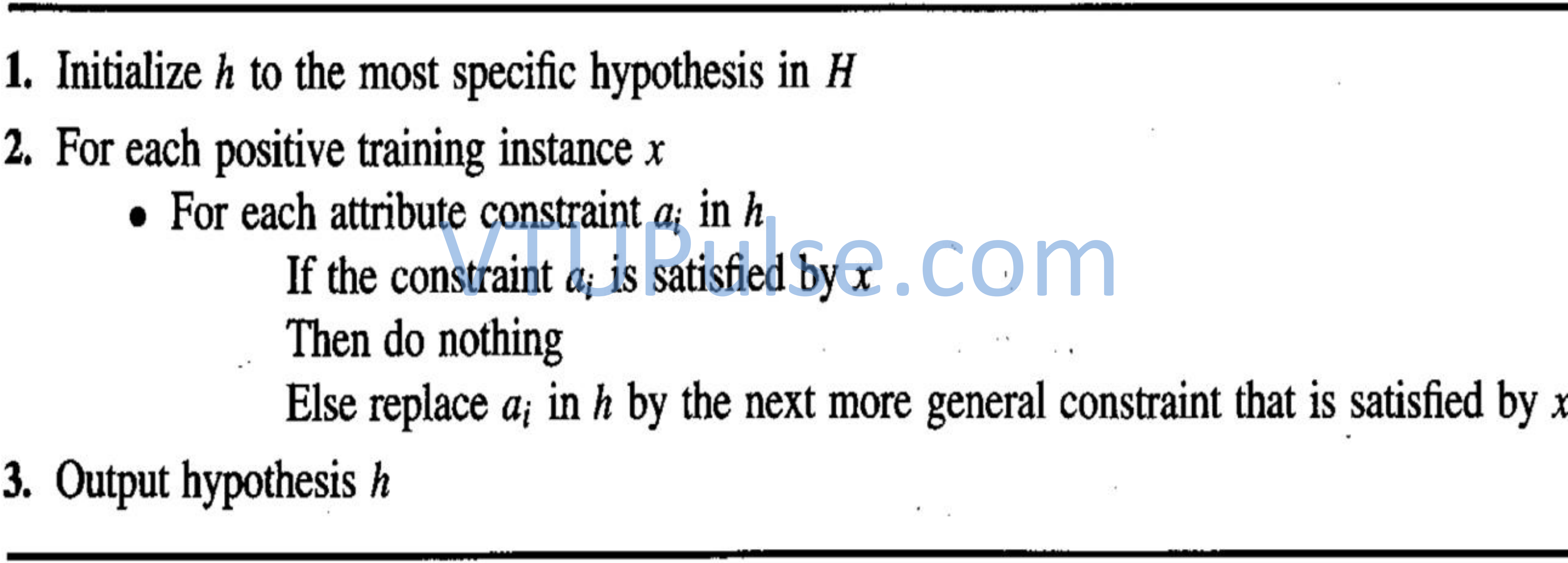
# FIND-S Algorithm

## Finding A Maximally Specific Hypothesis

Prof. Mahesh G Huddar

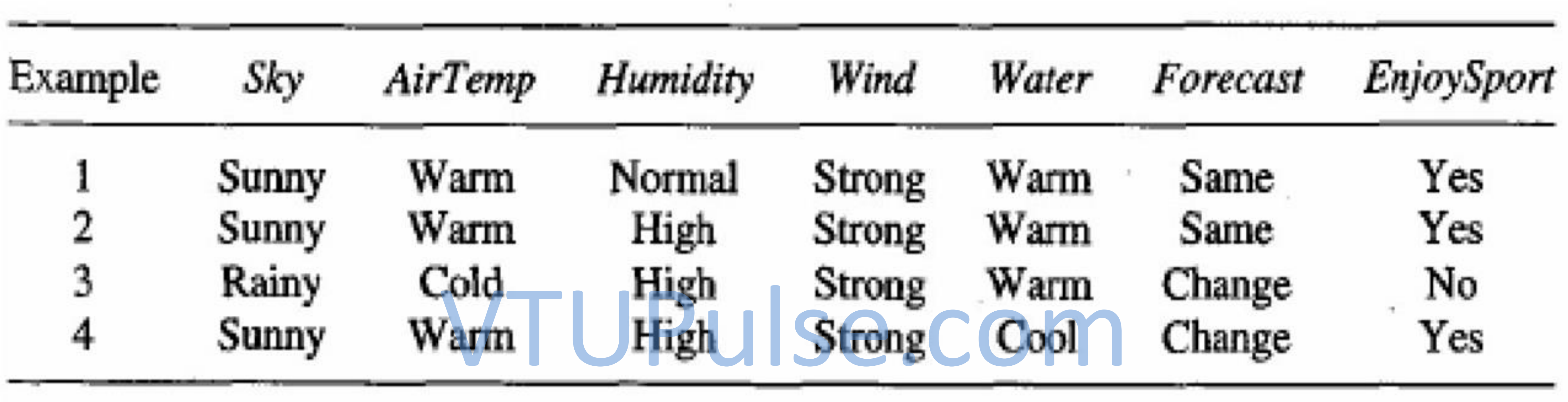
Dept. of Computer Science and Engineering

# FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

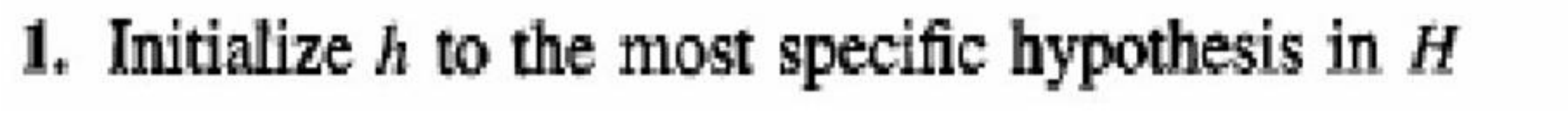


```
1. Initialize $h$ to the most specific hypothesis in $H$
2. For each positive training instance $x$
    • For each attribute constraint $a_i$ in $h$
            If the constraint $a_i$ is satisfied by $x$
            Then do nothing
            Else replace $a_i$ in $h$ by the next more general constraint that is satisfied by $x$
3. Output hypothesis $h$
```

# FIND-S: Step-1



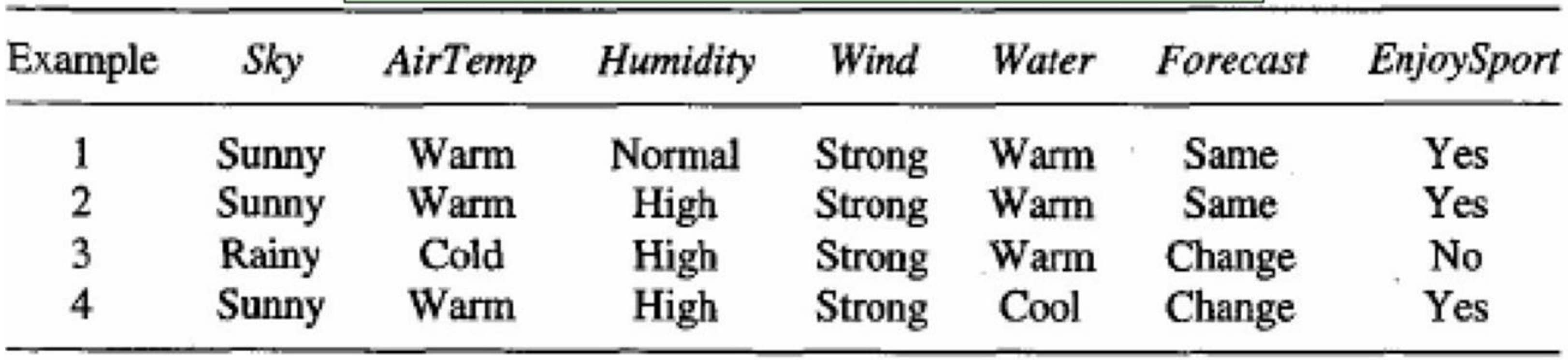
| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |



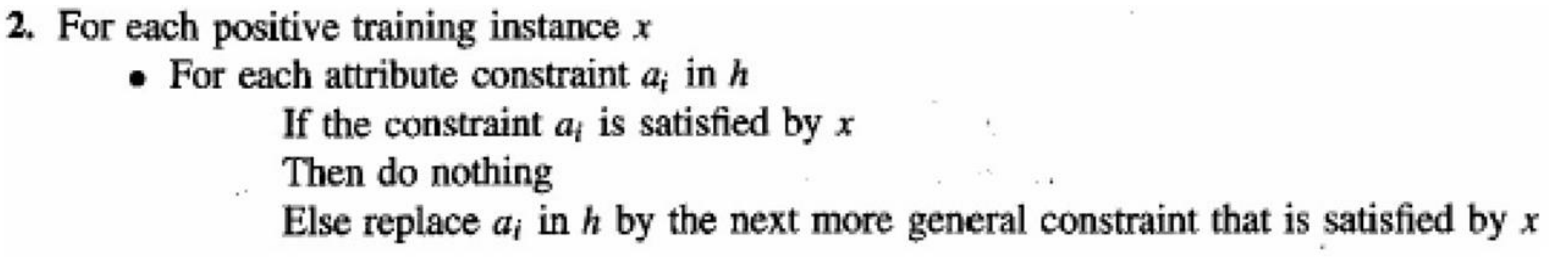
1. Initialize $h$ to the most specific hypothesis in $H$

h0 = <Ø, Ø, Ø, Ø, Ø, Ø>

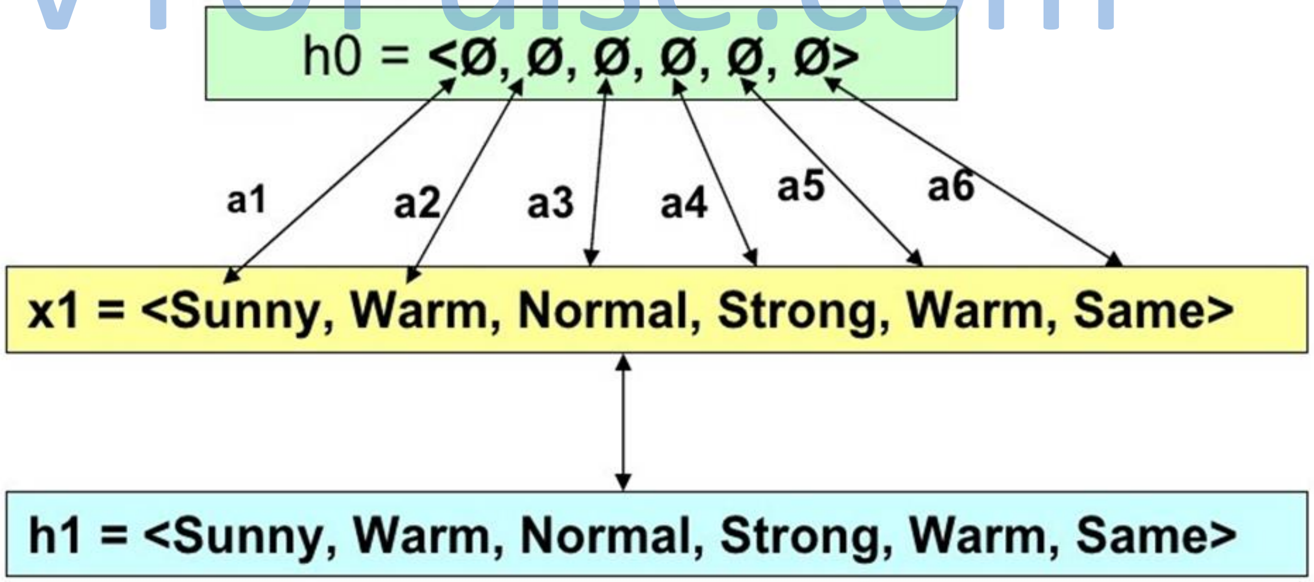
# FIND-S: Step-2



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |



```
2. For each positive training instance x
       • For each attribute constraint a_i in h
             If the constraint a_i is satisfied by x
             Then do nothing
             Else replace a_i in h by the next more general constraint that is satisfied by x
```



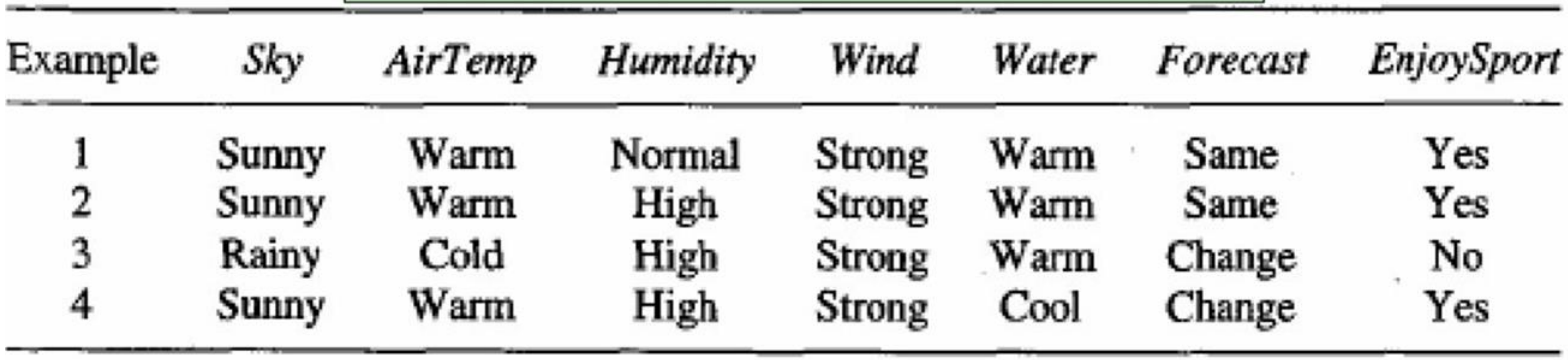
h0 = <Ø, Ø, Ø, Ø, Ø, Ø>

x1 = <Sunny, Warm, Normal, Strong, Warm, Same>

h1 = <Sunny, Warm, Normal, Strong, Warm, Same>

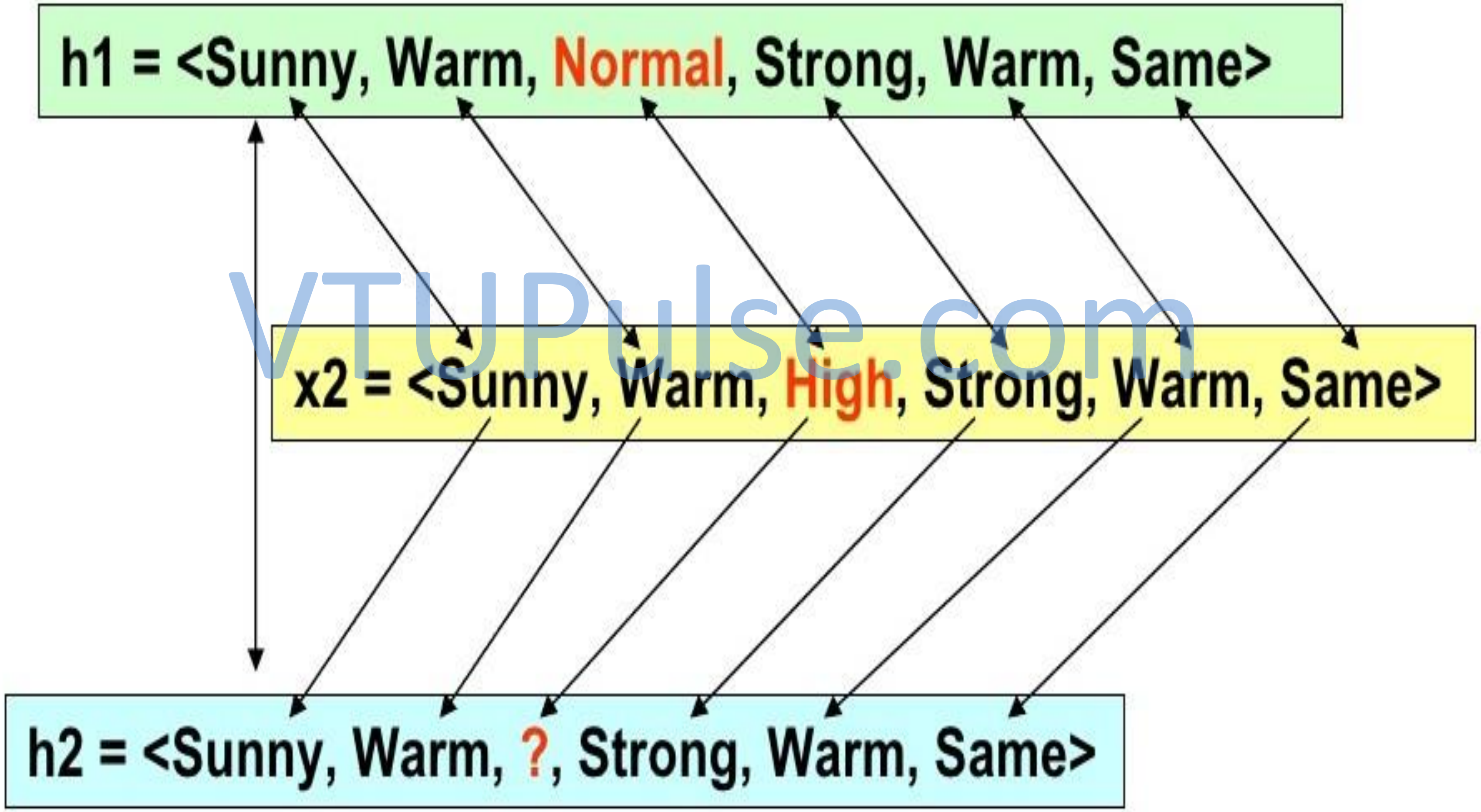
Iteration 1

# FIND-S: Step-2



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

Iteration 2

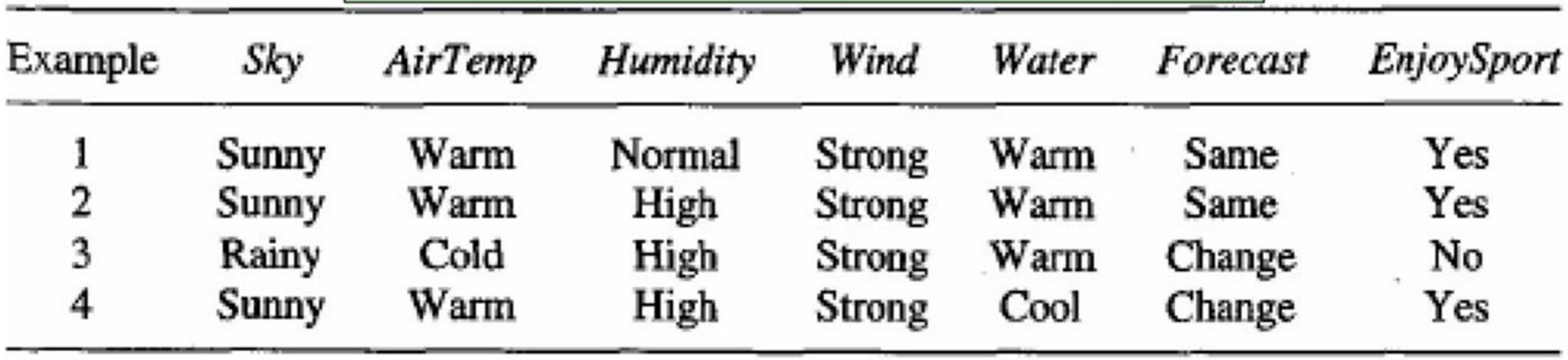


h1 = <Sunny, Warm, Normal, Strong, Warm, Same>

x2 = <Sunny, Warm, High, Strong, Warm, Same>

h2 = <Sunny, Warm, ?, Strong, Warm, Same>

# FIND-S: Step-2



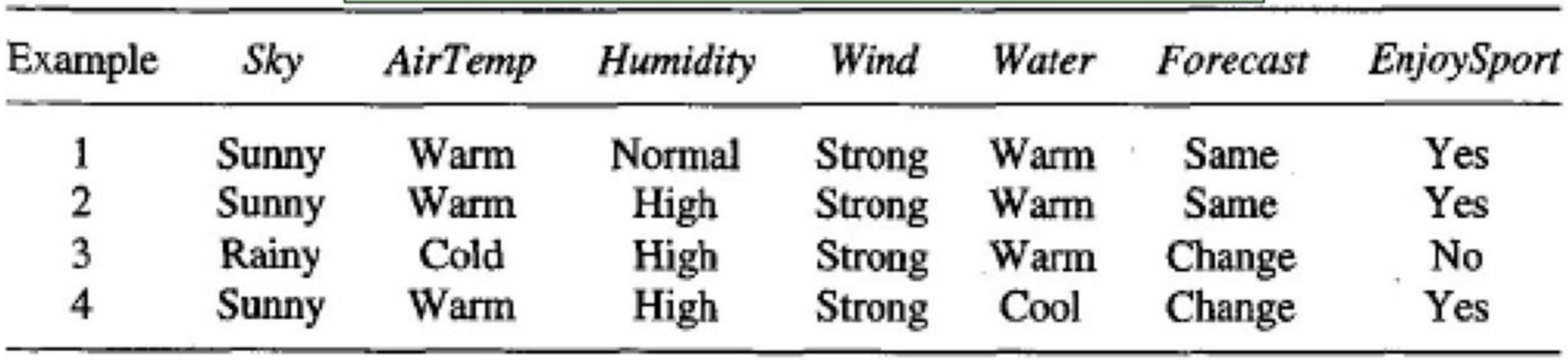
| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

Iteration 3

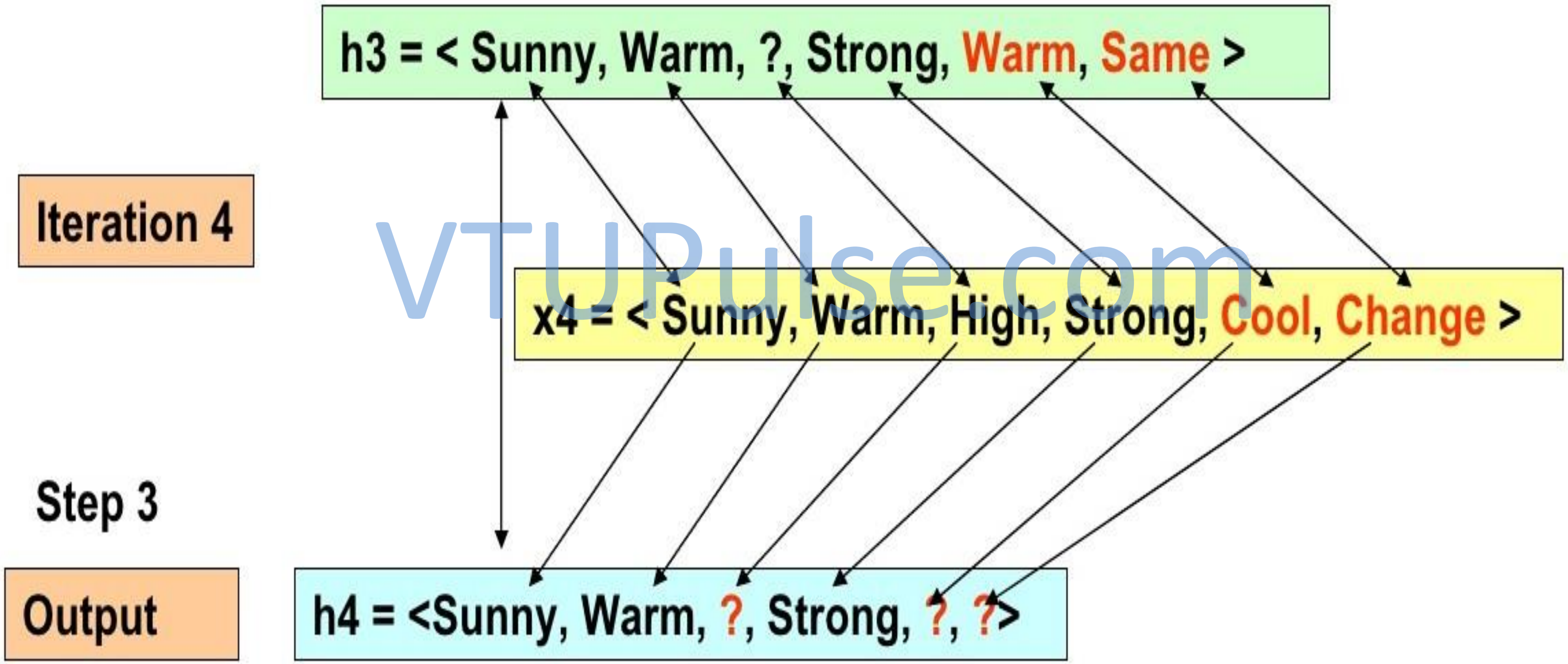
Ignore

h3 = <Sunny, Warm, ?, Strong, Warm, Same>

# FIND-S: Step-2



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |



Iteration 4

h3 = < Sunny, Warm, ?, Strong, Warm, Same >

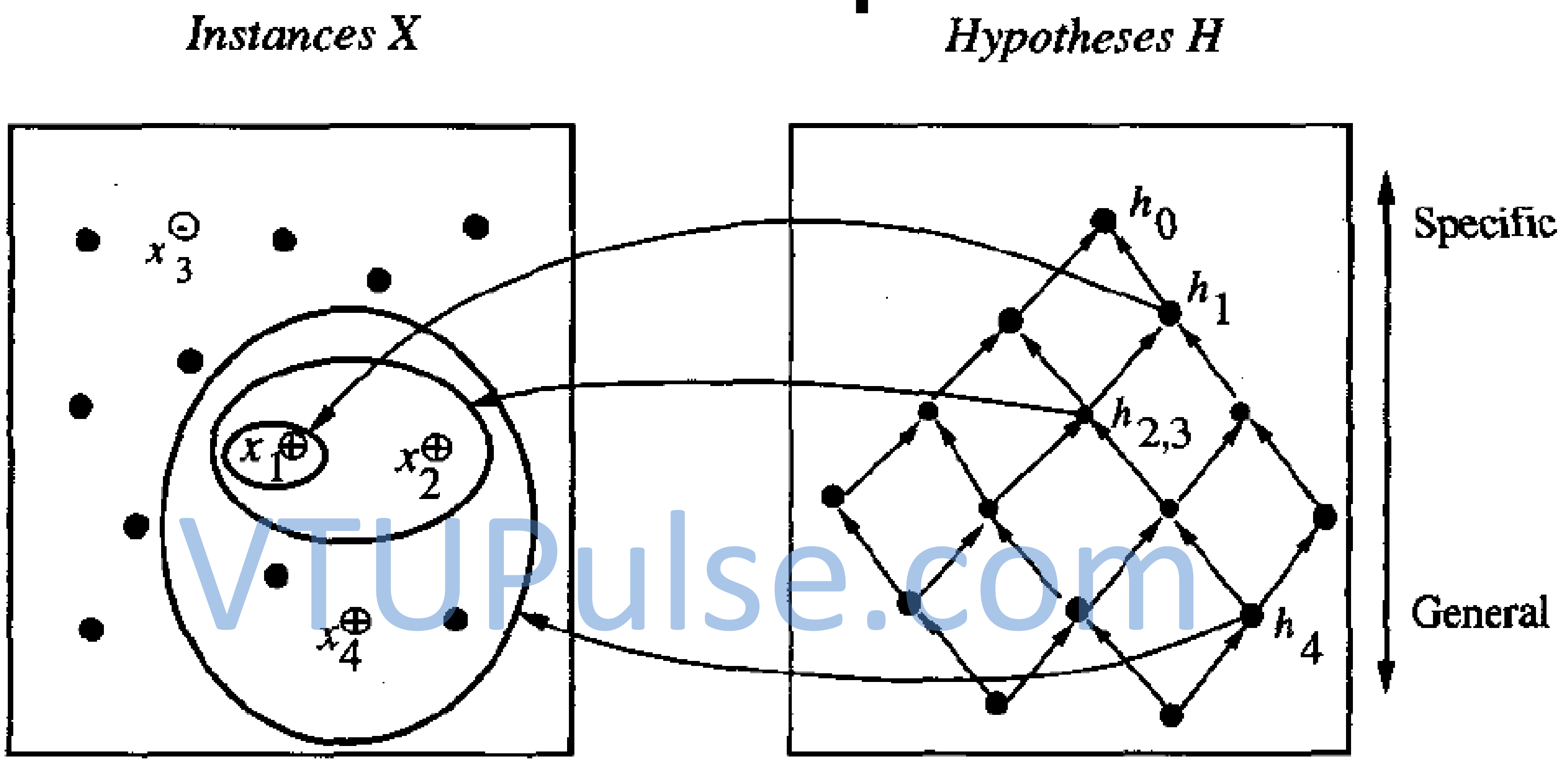
x4 = < Sunny, Warm, High, Strong, Cool, Change >

Step 3

Output

h4 = <Sunny, Warm, ?, Strong, ?, ?>

# FIND-S: Step-2



$x_1$ = <*Sunny Warm Normal Strong Warm Same*>, +

$x_2$ = <*Sunny Warm High Strong Warm Same*>, +

$x_3$ = <*Rainy Cold High Strong Warm Change*>, -

$x_4$ = <*Sunny Warm High Strong Cool Change*>, +

$h_0$ = <∅, ∅, ∅, ∅, ∅, ∅>

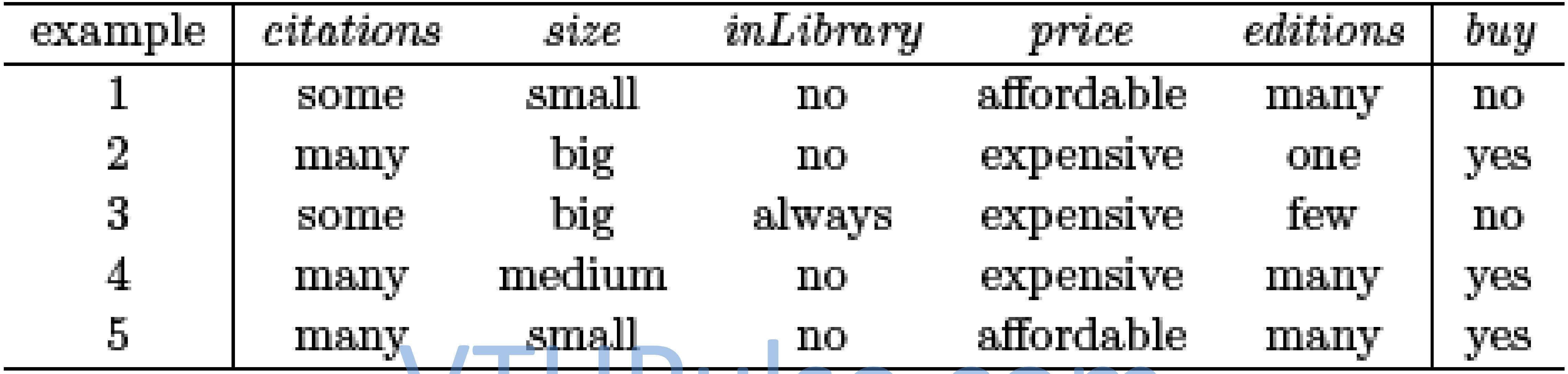
$h_1$ = <*Sunny Warm Normal Strong Warm Same*>

$h_2$ = <*Sunny Warm ? Strong Warm Same*>

$h_3$ = <*Sunny Warm ? Strong Warm Same*>

$h_4$ = <*Sunny Warm ? Strong ? ?* >

# FIND-S Algorithm Solved Example - 2

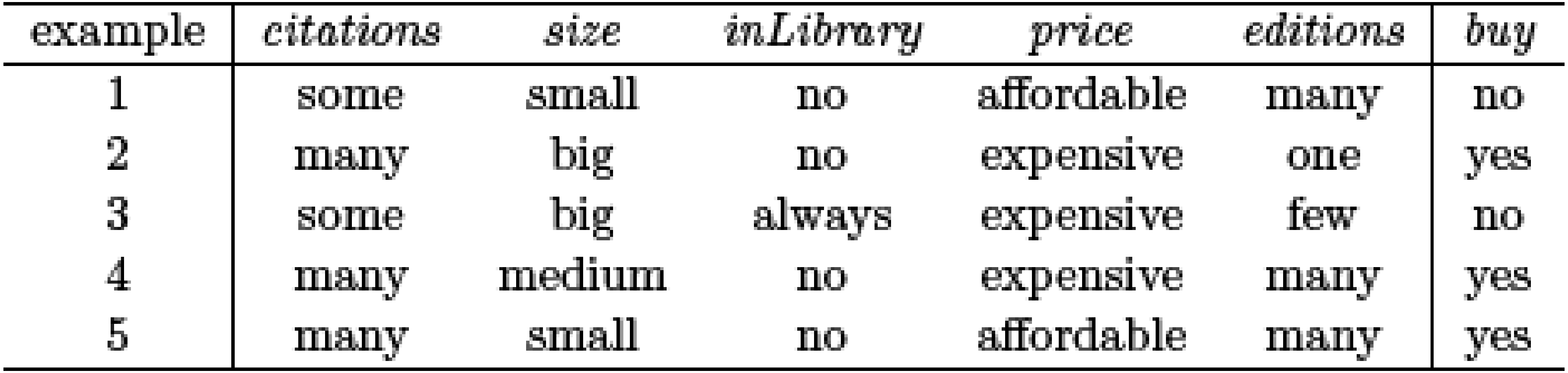


| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

1. **How many concepts are possible for this instance space?**

2. **How many hypotheses can be expressed by the hypothesis language?**

3. **Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.**



| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

**1. How many concepts are possible for this instance space?**

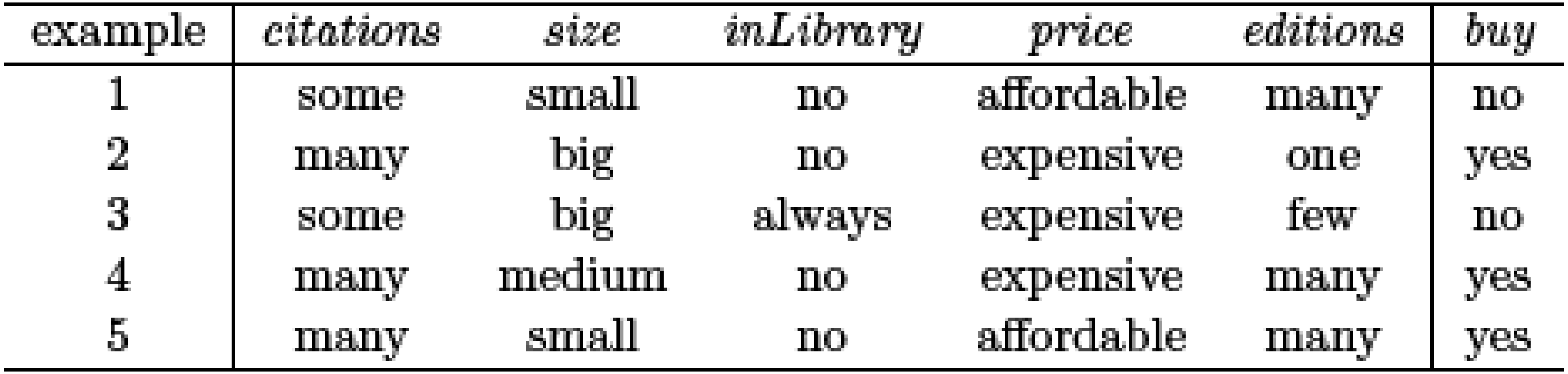
**Solution: 2 * 3 * 2 * 2 * 3 = 72**

**2. How many hypotheses can be expressed by the hypothesis language?**

**Solution: 4 * 5 * 4 * 4 * 5 = 1600**

**Semantically Distinct Hypothesis = ( 3 * 4 * 3 * 3 * 4 ) + 1 = 433**

Step 1:



| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

h0 = (ø, ø, ø, ø, ø)

Step 2:

X1 = (some, small, no, expensive, many) – No

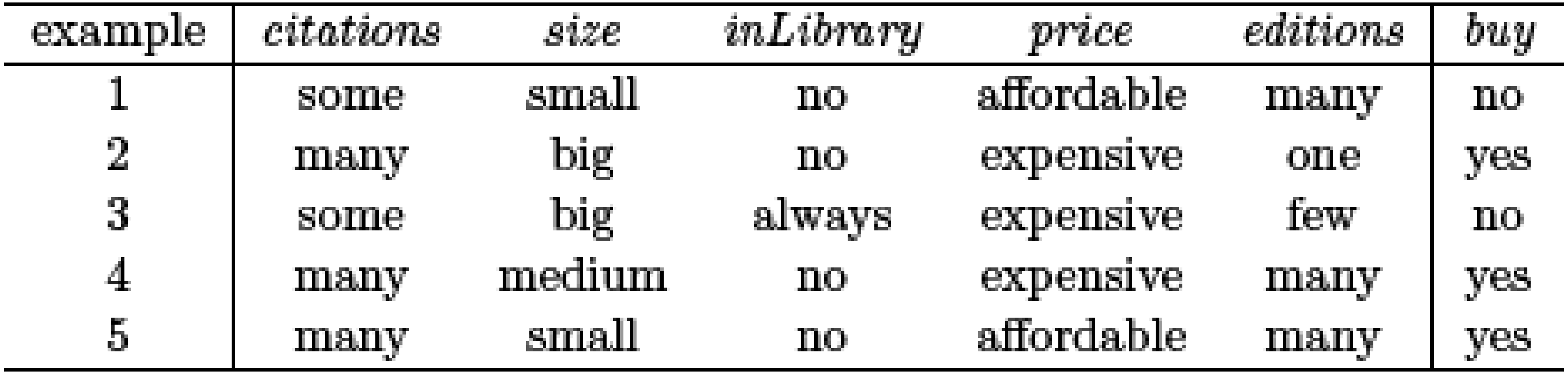
Negative Example Hence Ignore

h1 = (ø, ø, ø, ø, ø)

X2 = (many, big, no, expensive, one) – Yes

h2 = (many, big, no, expensive, one)

Step 2:



| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

h2 = (many, big, no, expensive, one)

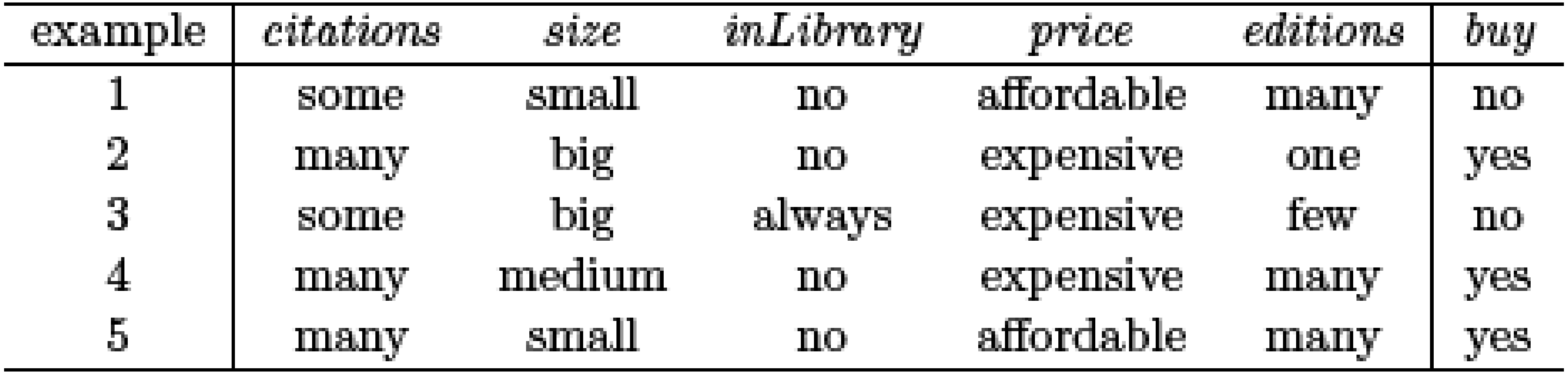
X3 = (some, big, always, expensive, few) – No

Negative example hence Ignore

h3 = (many, big, no, expensive, one)

X4 = (many, medium, no, expensive, many) – Yes

h4 = (many, ?, no, expensive, ?)



| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

**Step 2:**

h4 = (many, ?, no, expensive, ?)

**X5 = (many, small, no, affordable, many) – Yes**

h5 = (many, ?, no, ?, ?)

**Step 3:**

**Final Hypothesis is:**

h5 = (many, ?, no, ?, ?)

# FIND – S Algorithm in PYTHON

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on the given set of training data samples. Read the training data from a .CSV file

```
import csv
 a = []
 with open('enjoysport.csv', 'r') as csvfile:
    next(csvfile)
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)

print("\nThe total number of training instances are : ",len(a))

num_attribute = len(a[0])-1
print("No of Attributess: “, num_attribute)
print("\nThe initial hypothesis is : ")
hypothesis = ['0']*num_attribute
print(hypothesis)
```

```
[  ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'],
   ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'],
   ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'],
   ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] ]

The total number of training instances are :  4
No of Attributes : 6
The initial hypothesis is :
['0', '0', '0', '0', '0', '0']
```

```
for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        print ("\nInstance ", i+1, "is", a[i], " and is Positive Instance")
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'
        print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")
    if a[i][num_attribute] == 'no':
        print ("\nInstance ", i+1, "is", a[i], " and is Negative Instance Hence Ignored")
        print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")
```

```
[  ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'],
   ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'],
   ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'],
   ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] ]

The total number of training instances are :  4

The initial hypothesis is :
['0', '0', '0', '0', '0', '0']
```

```
Instance 1 is ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'] and is Positive Instance
The hypothesis for the training instance 1 is: ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
Instance 2 is ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'] and is Positive Instance
The hypothesis for the training instance 2 is: ['sunny', 'warm', '?', 'strong', 'warm', 'same']
Instance 3 is ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'] and is Negative Instance Hence Ignored
The hypothesis for the training instance 3 is: ['sunny', 'warm', '?', 'strong', 'warm', 'same']
Instance 4 is ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] and is Positive Instance
The hypothesis for the training instance 4 is: ['sunny', 'warm', '?', 'strong', '?', '?']
```

# FIND – S Algorithm in PYTHON

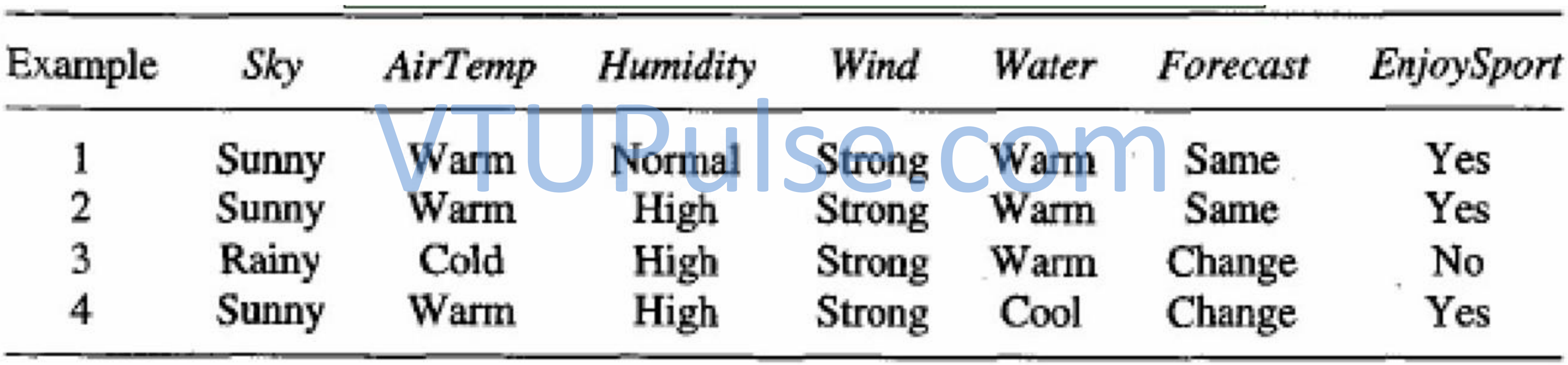
Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on the given set of training data samples. Read the training data from a .CSV file

```
print("\nThe Maximally specific hypothesis for the training instance is ")
print(hypothesis)
```

```
The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?','?']
```

# FIND – S Algorithm in PYTHON

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on the given set of training data samples. Read the training examples from .CSV file.



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

```
import csv
 a = []
 with open('enjoysport.csv', 'r') as csvfile:
    next(csvfile)
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)
 print("\nThe total number of training instances are : ",len(a))
 num_attribute = len(a[0])-1
print("No of Attributess: “, num_attribute)
print("\nThe initial hypothesis is : ")
hypothesis = ['0']*num_attribute
print(hypothesis)

for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        print ("\nInstance ", i+1, "is", a[i], " and is Positive Instance")
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'
        print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")

    if a[i][num_attribute] == 'no':
        print ("\nInstance ", i+1, "is", a[i], " and is Negative Instance Hence Ignored")
        print("The hypothesis for the training instance", i+1, " is: " , hypothesis, "\n")

print("\nThe Maximally specific hypothesis for the training instance is ")
print(hypothesis)
```

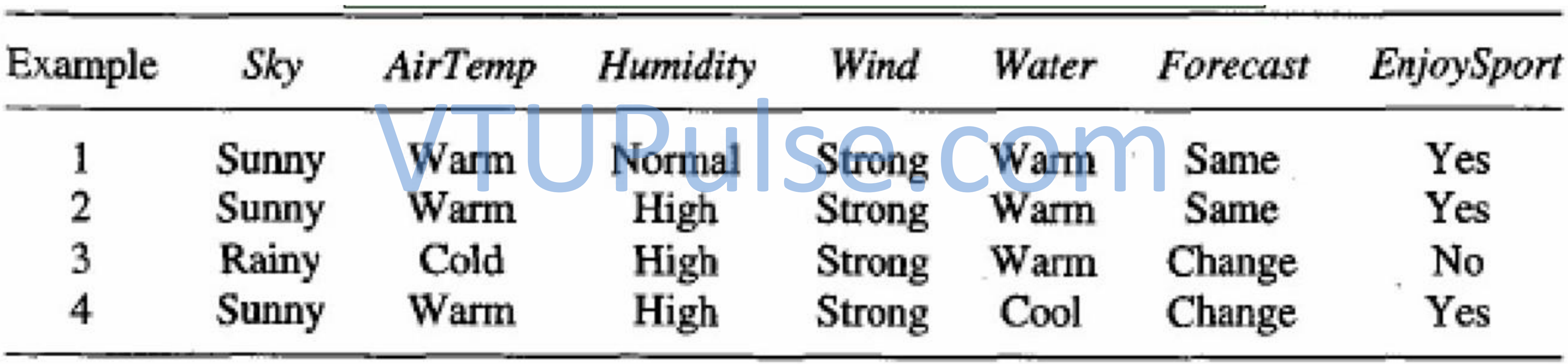
```
[  ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'],
   ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'],
   ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'],
   ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] ]

The total number of training instances are :  4
No of Attributes : 6
The initial hypothesis is :
['0', '0', '0', '0', '0', '0']

Instance 1 is ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'] and is Positive
Instance
The hypothesis for the training instance 1 is: ['sunny', 'warm', 'normal', 'strong', 'warm',
'same']
Instance 2 is ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'] and is Positive
Instance
The hypothesis for the training instance 2 is: ['sunny', 'warm', '?', 'strong', 'warm',
'same']
Instance 3 is ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'] and is Negative
Instance Hence Ignored
The hypothesis for the training instance 3 is: ['sunny', 'warm', '?', 'strong', 'warm',
'same']
Instance 4 is ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] and is Positive
Instance
The hypothesis for the training instance 4 is: ['sunny', 'warm', '?', 'strong', '?', '?']
The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
```

# FIND – S Algorithm in PYTHON

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on the given set of training data samples. Read the training examples from .CSV file.



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

# Candidate Elimination Algorithm in PYTHON

```
import numpy as np

import pandas as pd


data = pd.read_csv(path+'/enjoysport.csv')

concepts = np.array(data.iloc[:,0:-1])

print("\nInstances are:\n",concepts)

target = np.array(data.iloc[:,-1])

print("\nTarget Values are: ",target)
```

```
Instances are:

[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'high' 'strong' 'warm' 'same']

['rainy' 'cold' 'high' 'strong' 'warm' 'change']

['sunny' 'warm' 'high' 'strong' 'cool' 'change']]


Target Values are: ['yes' 'yes' 'no' 'yes']
```

# Candidate Elimination Algorithm in PYTHON

```
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and genearal_h")
    print("\nSpecific Boundary: ", specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)

    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "yes":
            print("Instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("Instance is Negative ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'

        print("Specific Bundary after ", i+1, "Instance is ", specific_h)
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
```

```
Instances are:

[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

['sunny' 'warm' 'high' 'strong' 'warm' 'same']

['rainy' 'cold' 'high' 'strong' 'warm' 'change']

['sunny' 'warm' 'high' 'strong' 'cool' 'change']]


Target Values are: ['yes' 'yes' 'no' 'yes']
```

```
import numpy as np
import pandas as pd

data = pd.read_csv(path+'/enjoysport.csv')
concepts = np.array(data.iloc[:,0:-1])
print("\nInstances are:\n",concepts)
target = np.array(data.iloc[:,-1])
print("\nTarget Values are: ",target)

def learn(concepts, target):
   specific_h = concepts[0].copy()
   print("\nInitialization of specific_h and genearal_h")
   print("\nSpecific Boundary: ", specific_h)
   general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
   print("\nGeneric Boundary: ",general_h)

   for i, h in enumerate(concepts):
      print("\nInstance", i+1 , "is ", h)
      if target[i] == "yes":
         print("Instance is Positive ")
         for x in range(len(specific_h)):
            if h[x]!= specific_h[x]:
               specific_h[x] ='?'
               general_h[x][x] ='?'

      if target[i] == "no":
         print("Instance is Negative ")
         for x in range(len(specific_h)):
            if h[x]!= specific_h[x]:
               general_h[x][x] = specific_h[x]
            else:
               general_h[x][x] = '?'
```

```
[  ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'],
   ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'],
   ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'],
   ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] ]

The total number of training instances are :  4
No of Attributes : 6
The initial hypothesis is :
['0', '0', '0', '0', '0', '0']

Instance 1 is ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'] and is Positive
Instance
The hypothesis for the training instance 1 is: ['sunny', 'warm', 'normal', 'strong', 'warm',
'same']
Instance 2 is ['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'] and is Positive
Instance
The hypothesis for the training instance 2 is: ['sunny', 'warm', '?', 'strong', 'warm',
'same']
Instance 3 is ['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'] and is Negative
Instance Hence Ignored
The hypothesis for the training instance 3 is: ['sunny', 'warm', '?', 'strong', 'warm',
'same']
Instance 4 is ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes'] and is Positive
Instance
The hypothesis for the training instance 4 is: ['sunny', 'warm', '?', 'strong', '?', '?']
The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
```

# Unanswered Questions of FIND-S Algorithm

1. Has the learner converged to the correct target concept? Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the ***only*** hypothesis in ***H*** consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.

2. Why prefer the most specific hypothesis? In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis over the most general, or some other hypothesis of intermediate generality.

# Unanswered Questions of FIND-S Algorithm

3. Are the training examples consistent? In most practical learning problems there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples. We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

4. What if there are several maximally specific consistent hypotheses? In the hypothesis language H for the ***EnjoySport*** task, there is always a unique, most specific hypothesis consistent with any set of positive examples. However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.

# Consistent Hypothesis and Version Space

- ***The idea*****:** output a description of the set of *all hypotheses* ***consistent*** with the training examples (correctly classify training examples).
- ***Version Space*****:** a representation of the set of hypotheses which are *consistent* with *D*

1. an explicit list of hypotheses **(List-Then-Eliminate)**
2. a compact representation of hypotheses which exploits the *more_general_than* partial ordering **(Candidate-Elimination)**

# Consistent Hypothesis and Version Space

An hypothesis $h$ is **consistent** with a set of training examples $D$ iff $h(x) = c(x)$ for each example in $D$

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

| Example | Citations | Size | InLibrary | Price | Editions | Buy |
|---|---|---|---|---|---|---|
| 1 | Some | Small | No | Affordable | One | No |
| 2 | Many | Big | No | Expensive | Many | Yes |

**h1 = (?, ?, No, ?, Many)** **– Consistent**

**h2 = (?, ?, No, ?, ?)** **– Not Consistent**

# Consistent Hypothesis and Version Space

- The version space $VS_{H,D}$ is the subset of the hypothesis from $H$ *consistent* with the training example in $D$

$$VS_{H,D} \equiv \{h \in H \mid Consistent(h, D)\}$$

# *List-Then-Eliminate* algorithm

Version space as list of hypotheses

1. *VersionSpace* $\leftarrow$ a list containing every hypothesis in *H*

2. For each training example, $\langle x, c(x) \rangle$ Remove from *VersionSpace* any hypothesis *h* for which $h(x) \neq c(x)$

3. Output the list of hypotheses in *VersionSpace*

# Consistent Hypothesis and Version Space

- F1 - > A, B
- F2 - > X, Y
- **Instance Space:** (A, X), (A, Y), (B, X), (B, Y) – **4 Examples**

- **Hypothesis Space:** (A, X), (A, Y), (A, **ø**), (A, ?), (B, X), (B, Y), (B, **ø**), (B, ?), (**ø**, X), (**ø**, Y), (**ø**, **ø**), (**ø**, ?), (**?**, X), (**?**, Y), (**?**, **ø**), (**?**, ?) - **16 Hypothesis**
- **Semantically Distinct Hypothesis :** (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (**?**, X), (**?**, Y (**?**, ?), (**ø**, **ø**) – **10**

# Consistent Hypothesis and Version Space

- Version Space: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y) (?, ?), (ø, ø),
- Training Instances

| F1 | F2 | Target |
|---|---|---|
| A | X | Yes |
| A | Y | Yes |

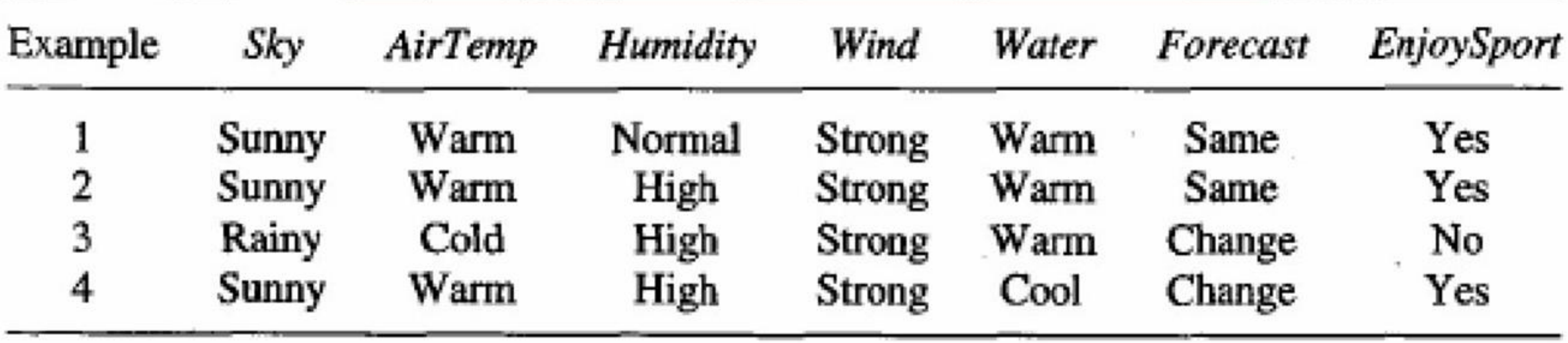
- Consistent Hypothesis are: (A, ?), (?, ?)

# *List-Then-Eliminate* algorithm

Problems

- The hypothesis space must be finite
- Enumeration of all the hypothesis, rather inefficient

# Candidate Elimination Algorithm  Solved Example - 1



| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| 2 | Sunny | Warm | High | Strong | Warm | Same | Yes |
| 3 | Rainy | Cold | High | Strong | Warm | Change | No |
| 4 | Sunny | Warm | High | Strong | Cool | Change | Yes |

$S_0$: ⟨∅, ∅, ∅, ∅, ∅. ∅⟩

$S_1$: ⟨*Sunny,Warm, Normal, Strong, Warm, Same*⟩

$S_2$: $S_3$: ⟨*Sunny,Warm, ?, Strong, Warm, Same*⟩

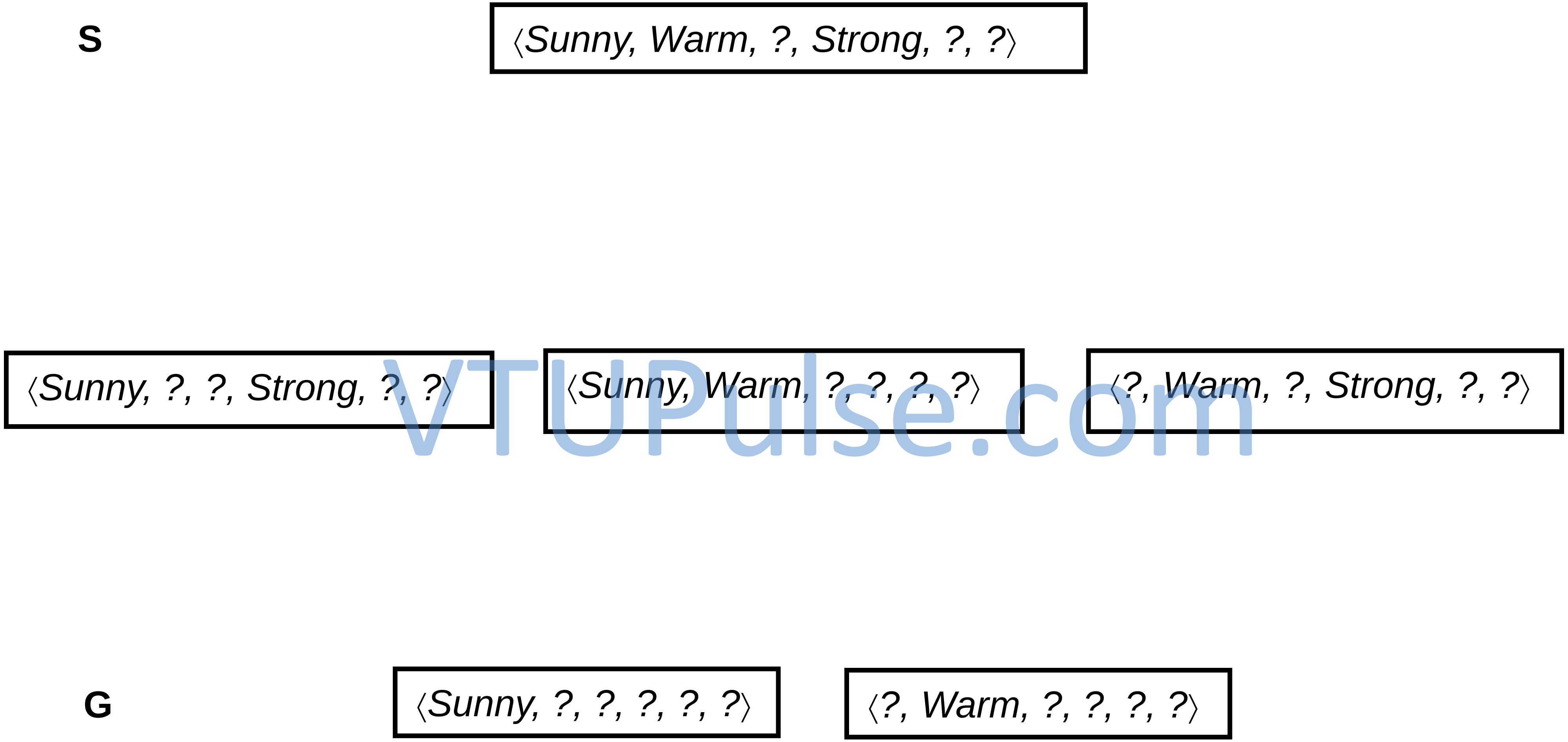
$S_4$ ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

$G_4$: ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩

$G_3$: ⟨*Sunny,?,?,?,?,?*⟩ ⟨*?,Warm,?,?,?,?*⟩ ⟨*?,?,Normal,?,?,?*⟩ ⟨*?, ?,?,?,Cool,?*⟩ ⟨*?,?,?,?,?,Same*⟩

$G_{0:}$ $G_{1:}$ $G_{2:}$ ⟨?, ?, ?, ?, ?, ?⟩

# Learned Version Space by Candidate Elimination Algorithm



S

*⟨Sunny, Warm, ?, Strong, ?, ?⟩*

*⟨Sunny, ?, ?, Strong, ?, ?⟩*

*⟨Sunny, Warm, ?, ?, ?, ?⟩*

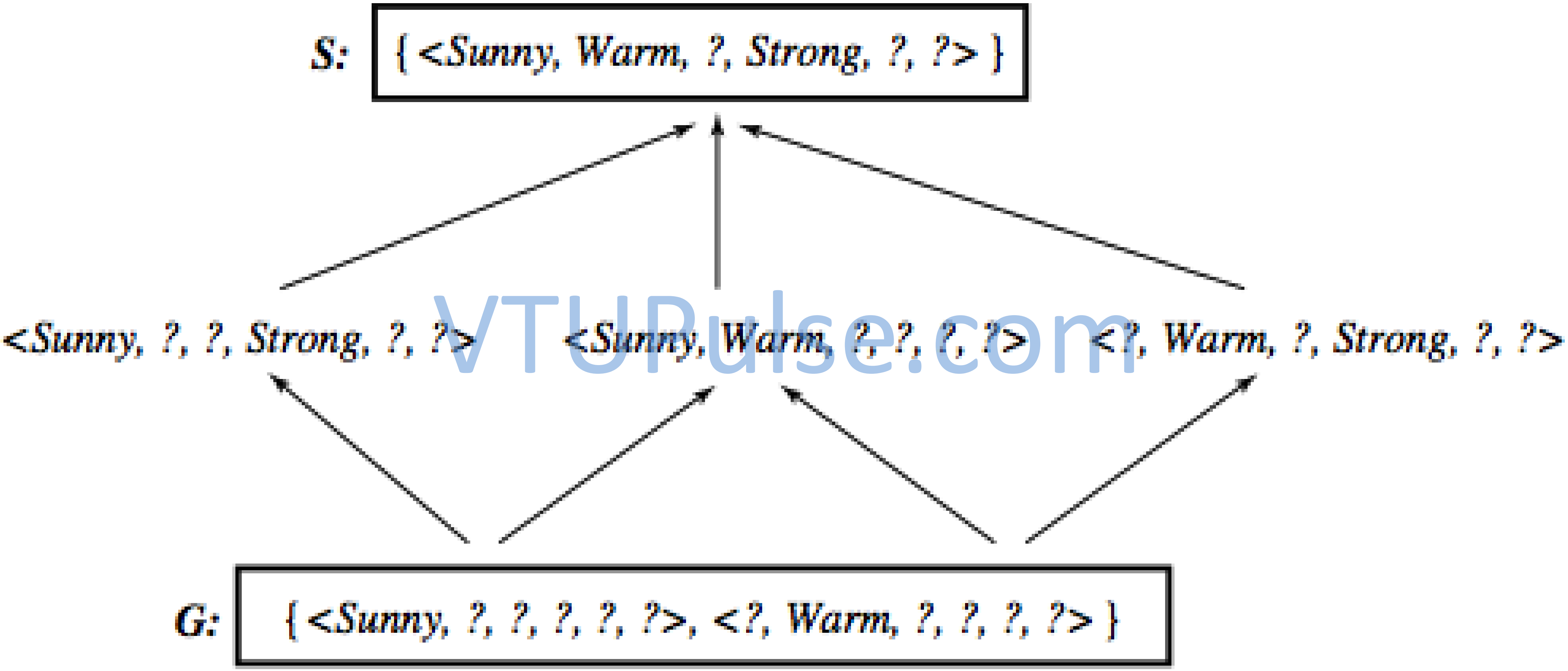
*⟨?, Warm, ?, Strong, ?, ?⟩*

G

*⟨Sunny, ?, ?, ?, ?, ?⟩*

*⟨?, Warm, ?, ?, ?, ?⟩*

# Learned Version Space by Candidate Elimination Algorithm

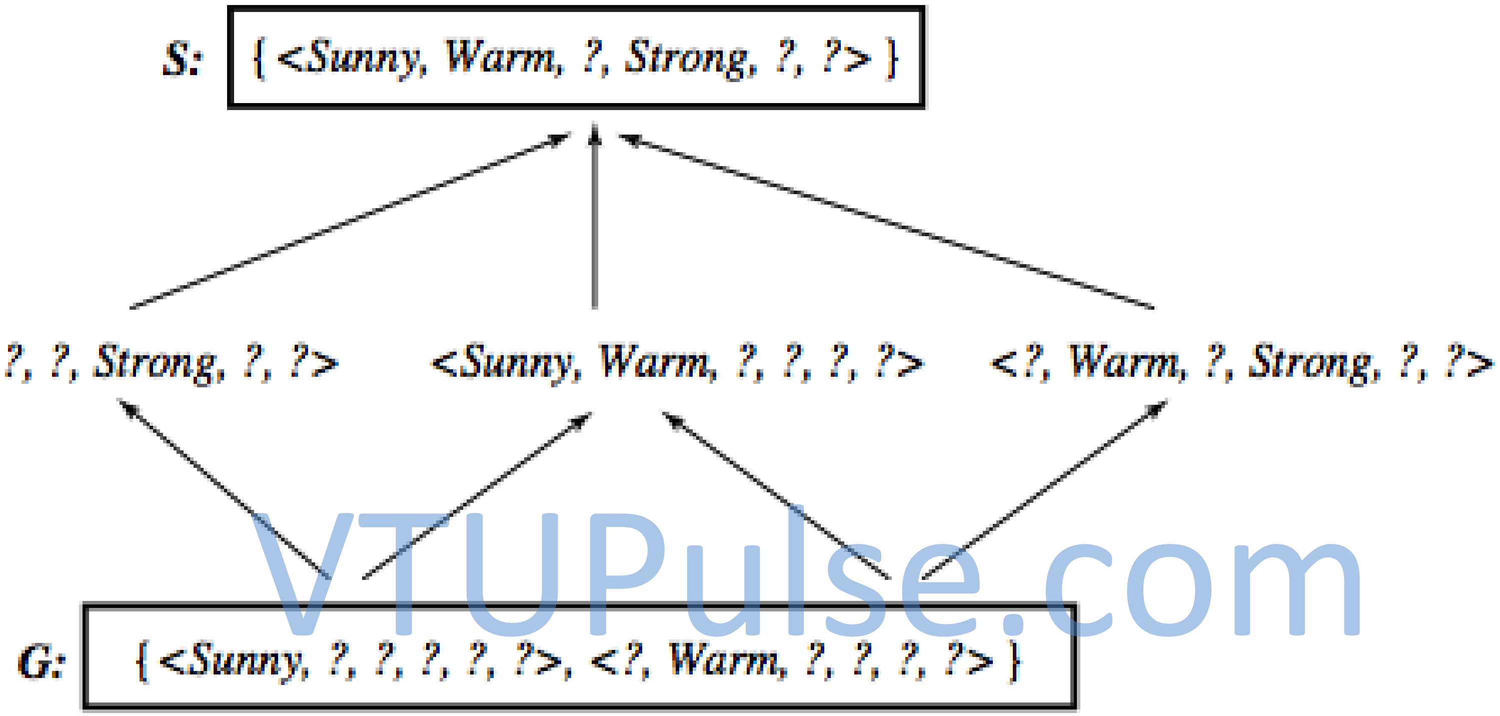


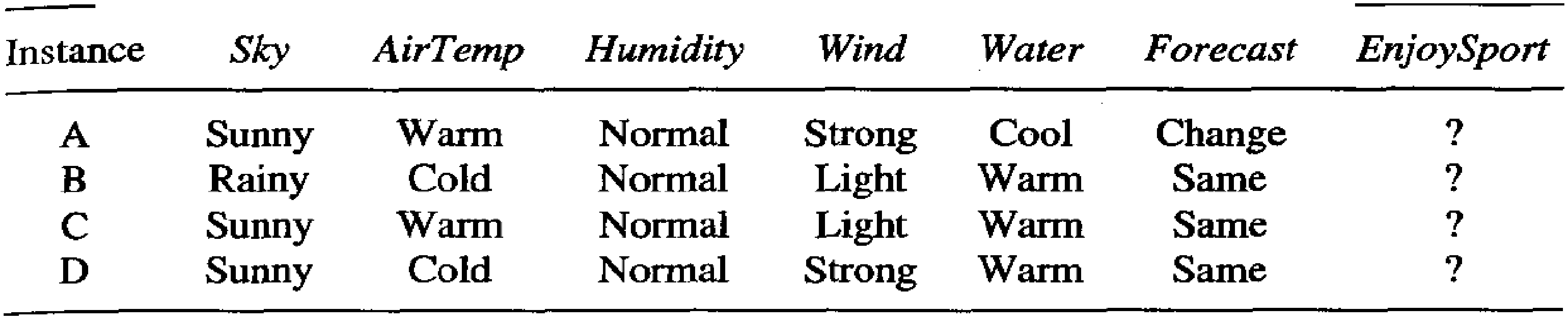
*S:* { <*Sunny*, *Warm*, ?, *Strong*, ?, ?> }

<*Sunny*, ?, ?, *Strong*, ?, ?> <*Sunny*, *Warm*, ?, ?, ?, ?> <?, *Warm*, ?, *Strong*, ?, ?>

*G:* { <*Sunny*, ?, ?, ?, ?, ?>, <?, *Warm*, ?, ?, ?, ?> }

# New instances to be classified





| Instance | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| A | Sunny | Warm | Normal | Strong | Cool | Change | ? |
| B | Rainy | Cold | Normal | Light | Warm | Same | ? |
| C | Sunny | Warm | Normal | Light | Warm | Same | ? |
| D | Sunny | Cold | Normal | Strong | Warm | Same | ? |

# Candidate Elimination Algorithm  Solved Example - 2

| SIze | Color | Shape | Class / Label |
|---|---|---|---|
| Big | Red | Circle | No |
| Small | Red | Triangle | No |
| Small | Red | Circle | Yes |
| Big | Blue | Circle | No |
| Small | Blue | Circle | Yes |

S0: (0, 0, 0)

S1: (0, 0, 0)

S2: (0, 0, 0)

S3: (Small, Red, Circle)

S4: (Small, Red, Circle)

S5: (Small, ?, Circle)

**S: G: (Small, ?, Circle)**

G5: (Small, ?, Circle)

G4: (Small, ?, Circle)

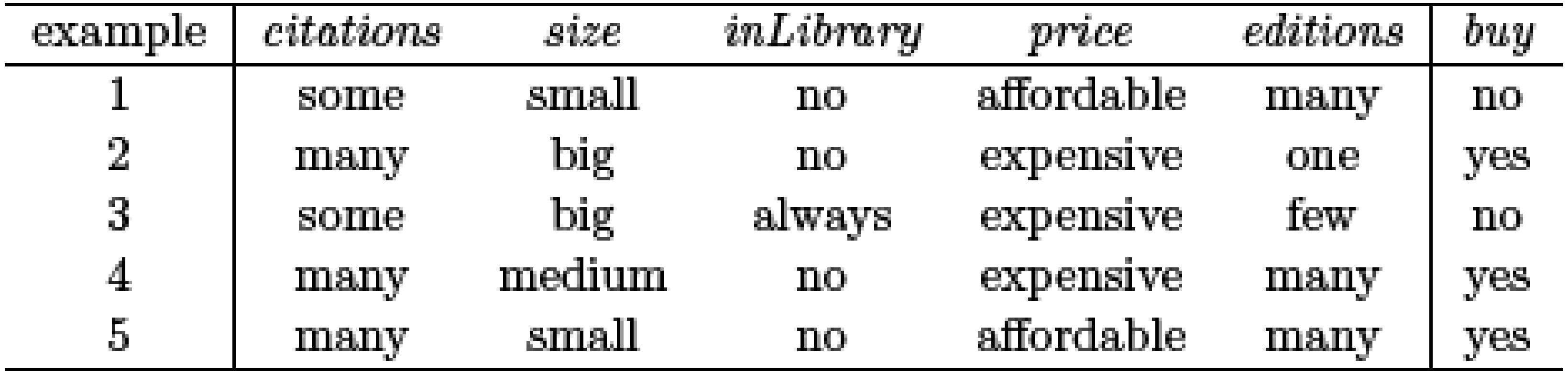
G3: (Small, ?, Circle)

G2: (Small, Blue, ?)    (Small, ?, Circle)    (?, Blue, ?)    (Big, ?, Triangle)    (?, Blue, Triangle)

G1: (Small, ?, ?)    (?, Blue, ?)    (?, ?, Triangle)

G0: (?, ?, ?)

# Candidate Elimination Algorithm  Solved Example - 3



| example | *citations* | *size* | *inLibrary* | *price* | *editions* | *buy* |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | medium | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

- S0:  (0, 0, 0, 0, 0)
- S1: (0, 0, 0, 0, 0)
- S2: (Many, Big, No, Exp, One)
- S3: (Many, Big, No, Exp, One)
- S4: (Many, ?, No, Exp, ?)
- **S5: (Many, ?, No, ?, ?)**
- (Many, ?, No, ?, ?)
- **G5: (Many, ?, ?, ?, ?)**
- G4: (Many,?,?,?,?) (Many,?,?,Exp,?) (?,?,No,exp,?)
- G3:  (Many,?,?,?,?)  (Many,big,?,?,?)  (?,Big,no,?,?)  (?,Big,?,Aff,?)  (?,Big,?,?,Many)  (?,Big,?,?,One)  (Many,?,?,Exp,?) (?,Small,?,Exp,?) (?,Medium,?,Exp,?) (?,?,No,exp,?) (?,?,?,Exp,one) (?,?,?,Exp,many) (?,?,?,?,One)
- G2: (Many,?,?,?, ?) (?, Big,?,?,?) (?,?,?,Exp,?) (?,?,?,?,One)
- G1: (Many,?,?,?, ?) (?, Big,?,?,?) (?,Medium,?,?,?) (?,?,Always,?,?) (?,?,?,Exp,?) (?,?,?,?,One) (?,?,?,?,Few)
- G0: (?, ?, ?, ?, ?)

**Final Hypothesis Set: (Many, ?, No, ?, ?) (Many, ?, ?, ?, ?)**

# Candidate Elimination Algorithm Solved Example - 3

- S0: (0, 0, 0, 0, 0)
- S1: (0, 0, 0, 0, 0)
- S2: (Many, Big, No, Exp, One)
- S3: (Many, ?, No, Exp, ?)
- S4: (Many, ?, No, ?, ?)

| Example | Citations | Size | InLibrary | Price | Editions | Buy |
|---|---|---|---|---|---|---|
| 1 | Some | Small | No | Affordable | One | No |
| 2 | Many | Big | No | Expensive | Many | Yes |
| 3 | Many | Medium | No | Expensive | Few | Yes |
| 4 | Many | Small | No | Affordable | Many | Yes |

**Final Hypothesis Set: (Many, ?, No, ?, ?) (Many, ?, ?, ?, ?)**

- G4: (Many,?,?,?, ?)
- G3: (Many,?,?,?, ?) (?,?,?,Exp,?)
- G2: (Many,?,?,?, ?) (?, Big,?,?,?) (?,?,?,Exp,?)
- G1: (Many,?,?,?, ?) (?, Big,?,?,?) (?,Medium,?,?,?) (?,?,Always,?,?) (?,?,?,Exp,?) (?,?,?,?,One) (?,?,?,?,Few)
- G0: (?, ?, ?, ?, ?)

# Candidate Elimination Algorithm  Solved Example - 3

- S0:  (0, 0, 0, 0, 0)
- S1: (0, 0, 0, 0, 0)
- S2: (Many, Big, No, Exp, Many)
- S3: (Many, ?, No, Exp, ?)
- S4: (Many, ?, No, ?, ?)

| Example | Citations | Size | InLibrary | Price | Editions | Buy |
|---|---|---|---|---|---|---|
| 1 | Some | Small | No | Affordable | One | No |
| 2 | Many | Big | No | Expensive | Many | Yes |
| 3 | Many | Medium | No | Expensive | Few | Yes |
| 4 | Many | Small | No | Affordable | Many | Yes |

**Final Hypothesis Set: (Many, ?, No, ?, ?) (Many, ?, ?, ?, ?)**

- G4: (Many,?,?,?, ?)
- G3: (Many,?,?,?, ?)    (?,?,?,Exp,?)
- G2: (Many,?,?,?, ?)    (?, Big,?,?,?)    (?,?,?,Exp,?)    (?,?,?,?,Many)
- G1: (Many,?,?,?, ?)    (?, Big,?,?,?) (?,Medium,?,?,?)    (?,?,?,Exp,?)    (?,?,?,?,Many) (?,?,?,?,Few)
- G0: (?, ?, ?, ?, ?)

# Candidate elimination algorithm

```
For each training example $d$, do:
$If\ d$ is positive example
        Remove from $G$ any hypothesis $h$ inconsistent with $d$
        For each hypothesis $s$ in $S$ not consistent with $d$:
        •       Remove $s$ from $S$
        •       Add to $S$ all minimal generalizations of $s$ consistent with $d$ and having a generalization in $G$
        •       Remove from $S$ any hypothesis with a more specific $h$ in $S$
$If\ d$ is negative example
        Remove from $S$ any hypothesis $h$ inconsistent with $d$
        For each hypothesis $g$ in $G$ not consistent with $d$:
        •       Remove $g$ from $G$
        •       Add to $G$ all minimal specializations of $g$ consistent with $d$ and having a specialization in $S$
        •       Remove from $G$ any hypothesis having a more general hypothesis in $G$
```

# Observations

- The  learned Version Space correctly describes the target concept, provided:
    1. There are no errors in the training examples
    2. There is some hypothesis that correctly describes the target concept

- If $S$ and $G$ converge to a single hypothesis the concept is exactly learned
- An empty version space means no hypothesis in $H$ is consistent with training examples

# Extending the hypothesis space

| | ***Sky*** | ***AirTemp*** | ***Humidity*** | ***Wind*** | ***Water*** | ***Forecast*** | ***EnjoyS*** |
|---|---|---|---|---|---|---|---|
| 1 | *Sunny* | *Warm* | *Normal* | *Strong* | *Cool* | *Change* | *YES* |
| 2 | *Cloudy* | *Warm* | *Normal* | *Strong* | *Cool* | *Change* | *YES* |
| 3 | *Rainy* | *Warm* | *Normal* | *Strong* | *Cool* | *Change* | *NO* |

- No hypothesis consistent with the three examples with the assumption that the target is a conjunction of constraints

  $\langle ?, \mathit{Warm}, \mathit{Normal}, \mathit{Strong}, \mathit{Cool}, \mathit{Change} \rangle$ is too general

- Target concept exists in a different space $H'$, including disjunction and in particular the hypothesis

  *Sky=Sunny* or *Sky=Cloudy*

# Hypothesis space and bias

- What if H does not contain the target concept?
- Can we improve the situation by extending the hypothesis space?
- Will this influence the ability to generalize?
- These are general questions for inductive inference, addressed in the context of Candidate-Elimination
- Suppose we include in H every possible hypothesis … including the ability to represent disjunctive concepts

# An unbiased learner

- Every possible subset of $X$ is a possible target

  $|H'| = 2^{|X|}$, or $2^{96}$ (vs $|H| = 973$, a strong bias)

- This amounts to allowing conjunction, disjunction and negation

  $\langle Sunny, ?, ?, ?, ?, ?\rangle$ V $< Cloudy, ?, ?, ?, ?, ?\rangle$

  Sunny(Sky) V Cloudy(Sky)

- We are guaranteed that the target concept exists

# No generalization without bias!

- $VS$ after presenting three positive instances $x_1$, $x_2$, $x_3$, and two negative instances $x_4$, $x_5$

  $S = \{(x_1 \vee x_2 \vee x_3)\}$

  $G = \{\neg(x_4 \vee x_5)\}$

  … all subsets including $x_1 x_2 x_3$ and not including $x_4 x_5$
- We can only classify precisely examples already seen!
- Take a majority vote?
  - Unseen instances, e.g. $x$, are classified positive (and negative) by half of the hypothesis
  - For any hypothesis $h$ that classifies $x$ as positive, there is a complementary hypothesis $\neg h$ that classifies $x$ as negative

# No inductive inference without a bias

- *A learner that makes no a priori assumptions regarding the identity of the target concept, has no rational basis for classifying unseen instances*
- The *inductive bias* of a learner are the assumptions that justify its inductive conclusions or the policy adopted for generalization
- Different learners can be characterized by their bias
- See next for a more formal definition of *inductive bias* …

# The Inductive Learning Hypothesis

**The Inductive Learning Hypothesis**

Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

# Inductive bias: definition

- Given:
  - a concept learning algorithm *L* for a set of instances *X*
  - a concept *c* defined over *X*
  - a set of training examples for *c*: $D_c = \{\langle x, c(x)\rangle\}$
  - $L(x_i, D_c)$ outcome of classification of $x_i$ after learning

- Inductive inference ( $\succ$ ):

$$D_c \wedge x_i \succ L(x_i, D_c)$$

- The *inductive bias* is defined as a minimal set of assumptions ***B***, such that ( $\vdash$ for deduction)

$$\forall (x_i \in X) [ (B \wedge D_c \wedge x_i) \vdash L(x_i, D_c) ]$$

# Inductive bias of Candidate-Elimination

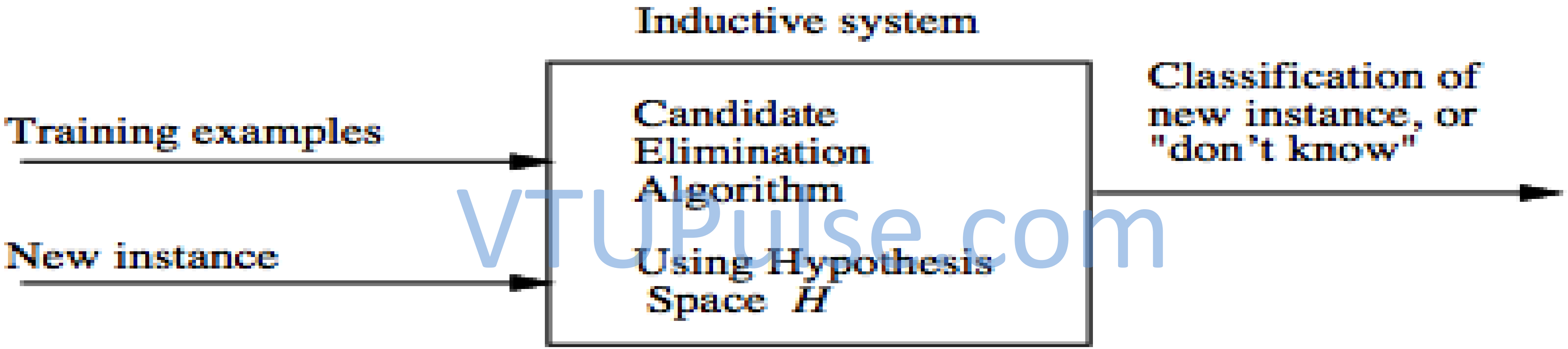
- Assume $L$ is defined as follows:
  - compute $VS_{\mathrm{H,D}}$
  - classify new instance by complete agreement of all the hypotheses in $VS_{\mathrm{H,D}}$
- Then the inductive bias of Candidate-Elimination is simply

  $B \equiv (c \in \mathrm{H})$

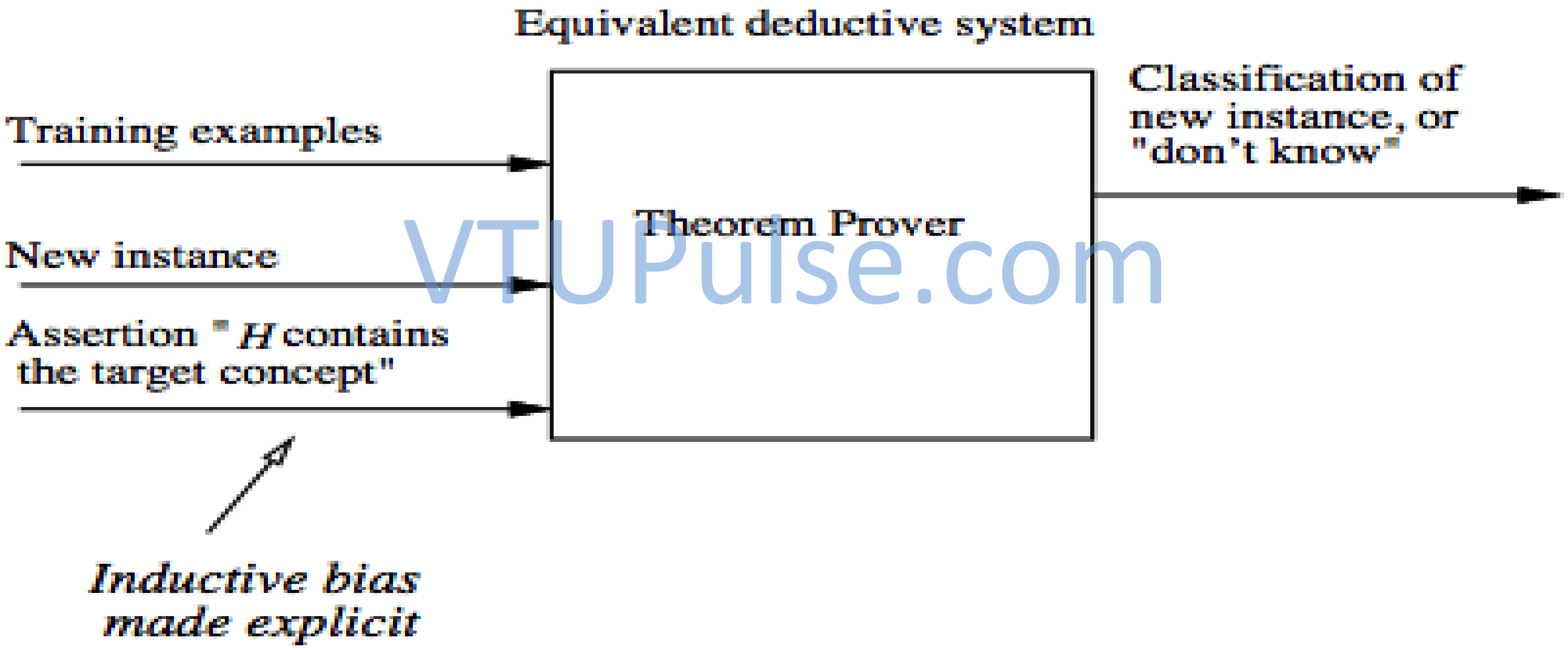
- In fact by assuming $c \in H$:

  1. $c \in VS_{\mathrm{H,D}}$ , in fact $VS_{\mathrm{H,D}}$ includes all hypotheses in $H$ consistent with D
  2. $L(x_i, D_c)$ outputs a classification "by complete agreement", hence any hypothesis, including $c$, outputs $L(x_i, D_c)$

# Inductive system


Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space $H$
Classification of new instance, or "don't know"

# Equivalent deductive system


Equivalent deductive system
Training examples
New instance
Assertion " H contains the target concept"
Inductive bias made explicit
Theorem Prover
Classification of new instance, or "don't know"

# Each learner has an inductive bias

- Three learner with three different inductive bias:
  1. *Rote learner*: no inductive bias, just stores examples and is able to classify only previously observed examples
  2. *CandidateElimination*: the concept is a conjunction of constraints.
  3. *Find-S*: the concept is in *H* (a conjunction of constraints) plus "all instances are negative unless seen as positive examples” (stronger bias)
  - The stronger the bias, greater the ability to generalize and classify new instances (greater inductive leaps).

# Issues in Machine Learning

1. What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? **Which algorithms perform best for which types of problems and representations?**
2. **How much training data is sufficient?** What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?
3. When and how can **prior knowledge** held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct?

# Issues in Machine Learning

4. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
5. What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?
6. How can the learner automatically alter **its** representation to improve its ability to represent and learn the target function?