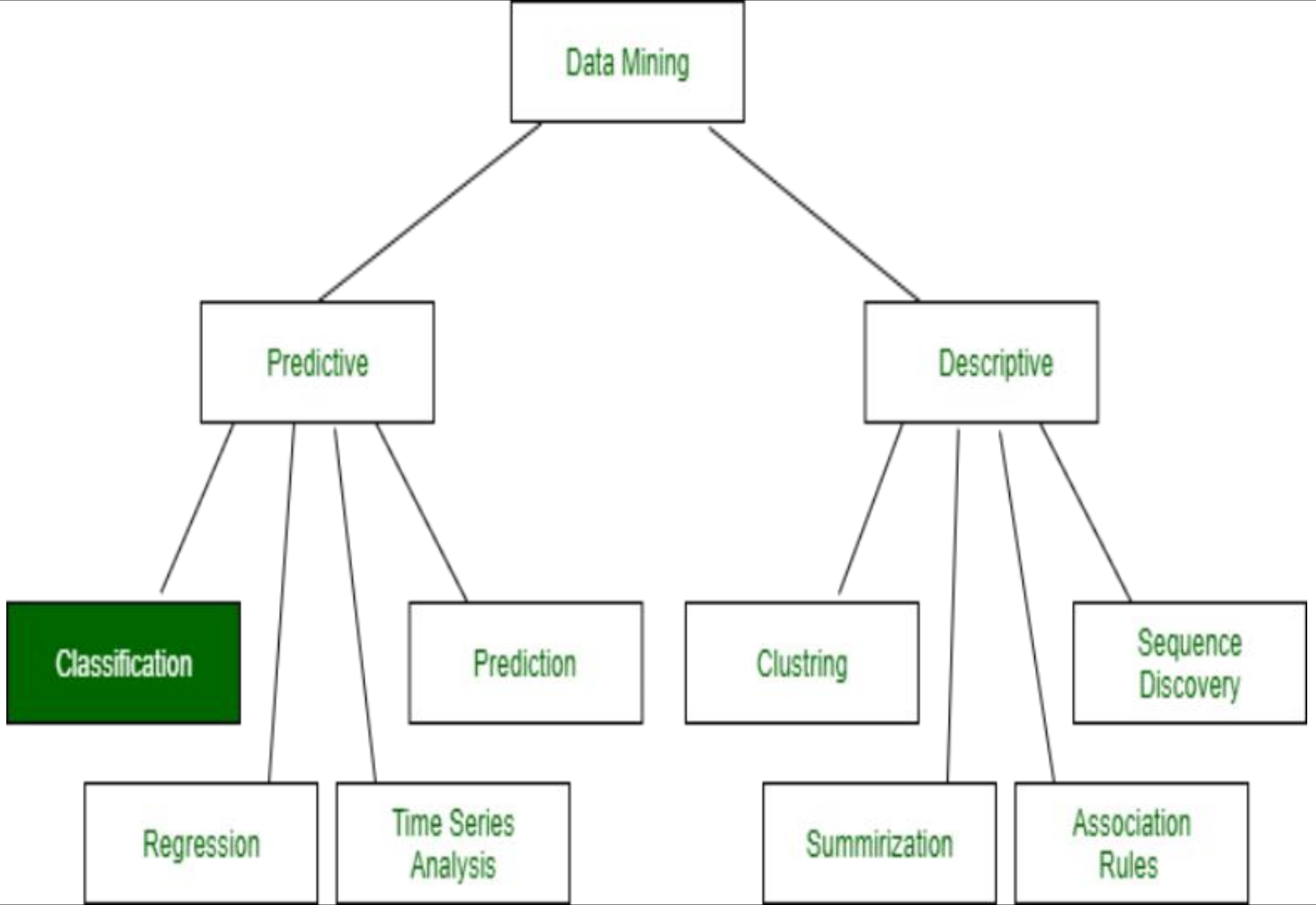
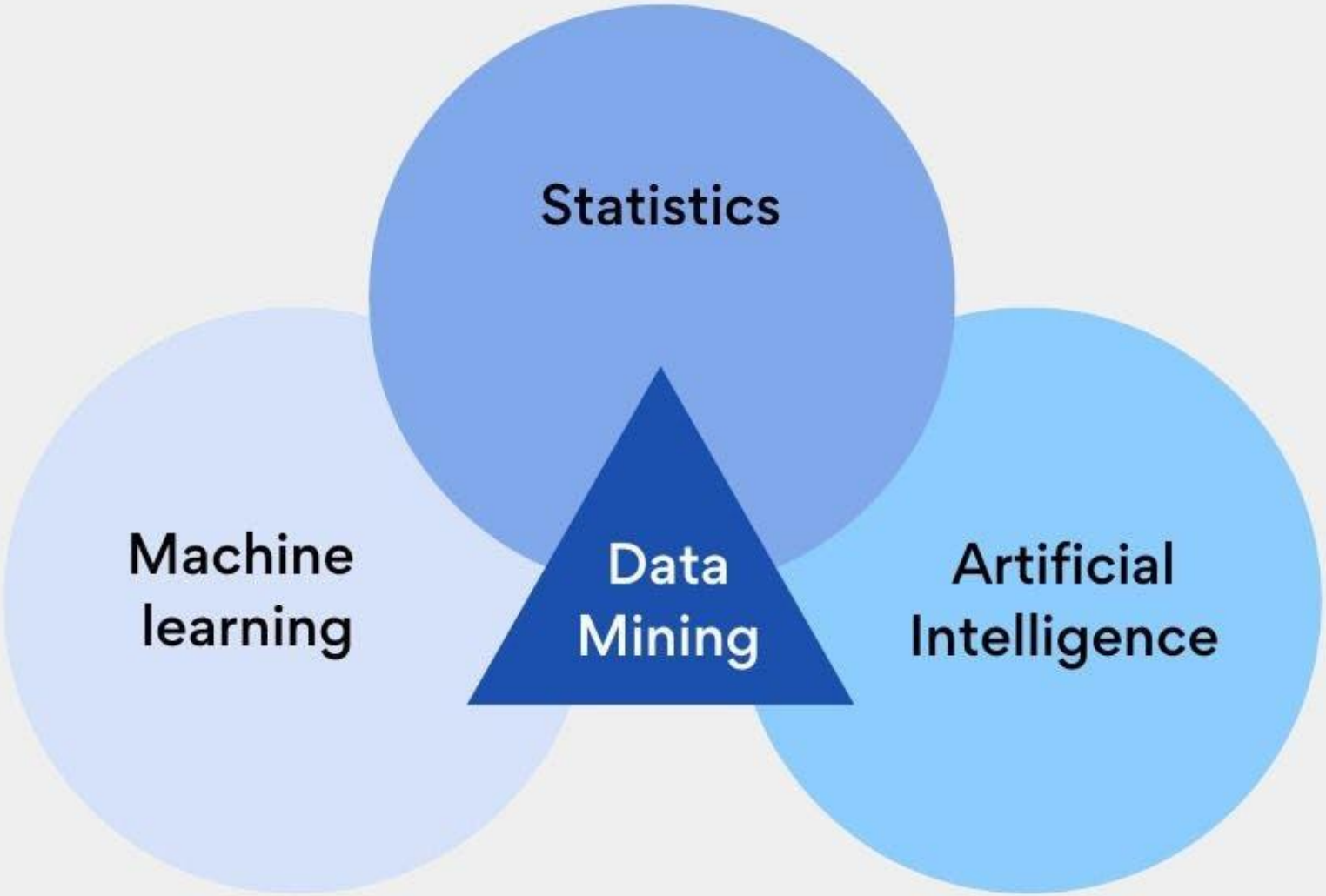
Regression & Classification

What we have studied so far?


Data Mining
Predictive
Descriptive
Classification
Prediction
Clustring
Sequence Discovery
Regression
Time Series Analysis
Summirization
Association Rules


Statistics
Machine learning
Data Mining
Artificial Intelligence

TURING

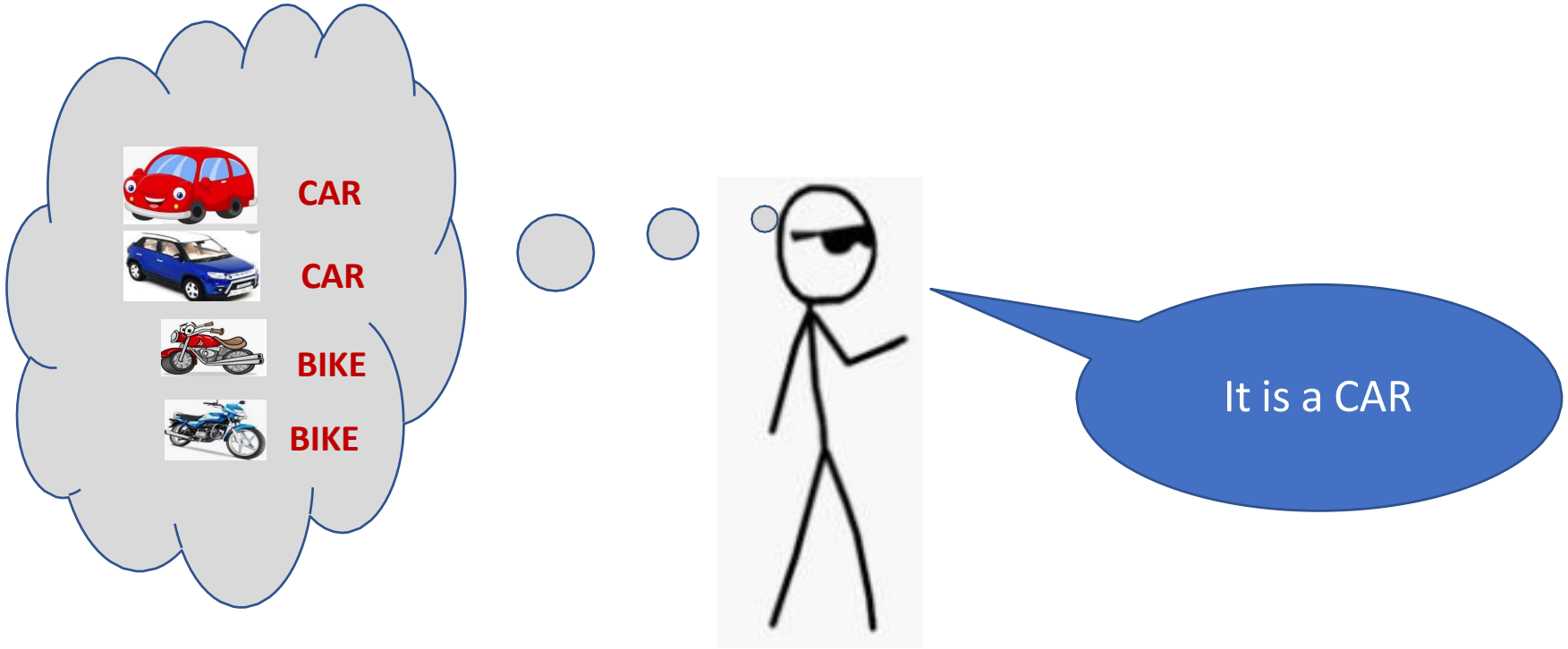

I'm a Data Scientist
R or Python
HTML

What is Machine Learning? (Layman's term)

Human can learn from past experience
and make decision of its own

What is this object?

What is this object?



Let us ask the same question to him

What is this object?

Let us ask the same question to him

What is this object?

[But, he is a human being. He can observe and learn]

Let us make him learn

show him

Let us make him learn

show him

CAR

CAR

BIKE

BIKE

Let us ask the same question now

What is this object?

CAR

CAR

BIKE

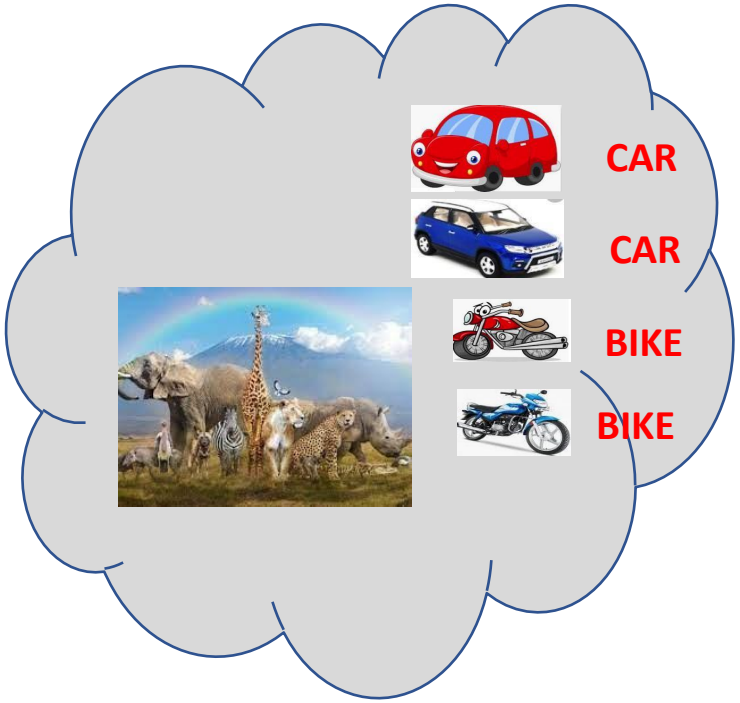
BIKE

Past experience

Let us ask the same question now

CAR

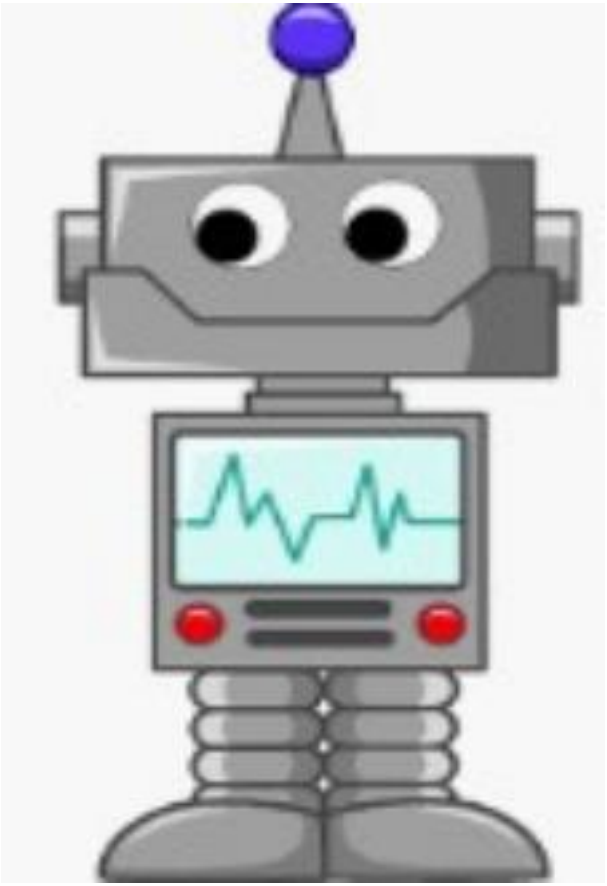
What is this object?



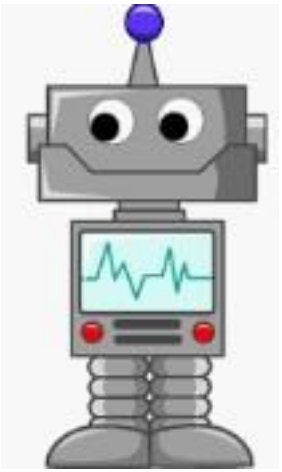
So Humans can learn from their experience and make decisions based on that.

What about a Machine?

Can it learn from past experience and make decisions on its own?



What about a Machine ?



Machines follow instructions

[It can not take decision of its own]

What about a Machine ?

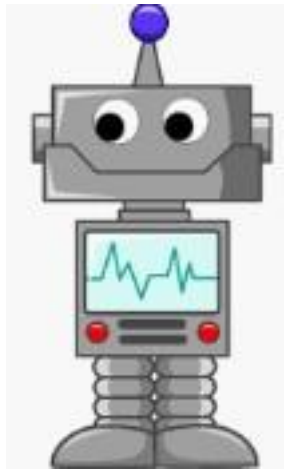
We can ask a machine

- To perform an arithmetic operations such as
  - Addition
  - Multiplication
  - Division

Machines follow instructions

What about a Machine ?

- Comparison
- Print
- Plotting a chart



Machines follow instructions

What is Machine Learning?

[We want a machine to act like a human]

What is Machine Learning?

[to identify this object.]

What is Machine Learning?

Price in 2025?

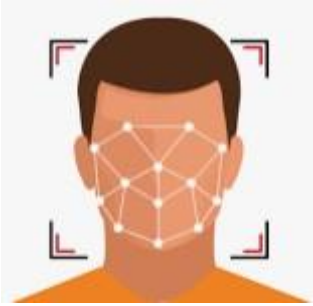
[predict the price in future]

What is Machine Learning?

I ~~made~~ met him yesterday

[Natural Language understand, and correct grammar]

What is Machine Learning?



recognize face

[Recognize Faces]

What is Machine Learning?

[What do we do?]

Just like, what we did to human, we need to provide

experience to the machine.

What is Machine Learning?

+

Dataset

[
This what we called as Data or Training dataset.

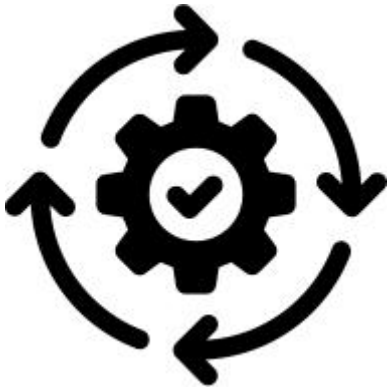
So, we first need to provide training dataset to the machine.
]

What is Machine Learning?

+

Dataset

+



[Then, devise algorithms and execute programs on the data

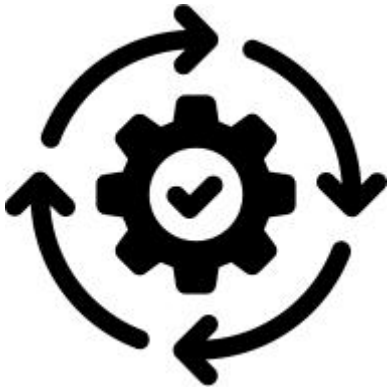
With respect to the underlying target tasks]

What is Machine Learning?

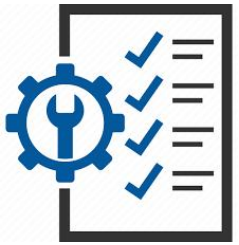
+

Dataset

+



+



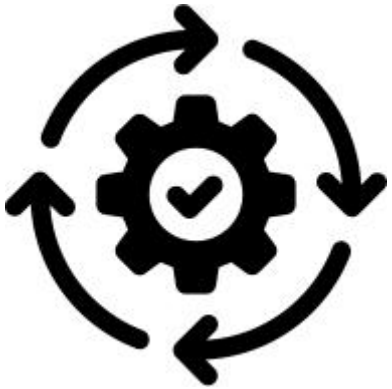
[Then, using the programs, Identify required rules]

What is Machine Learning?

+

Dataset

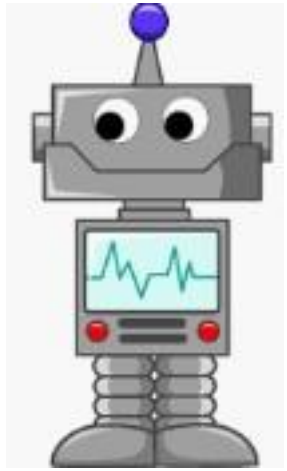
+



+

[extract required patterns]

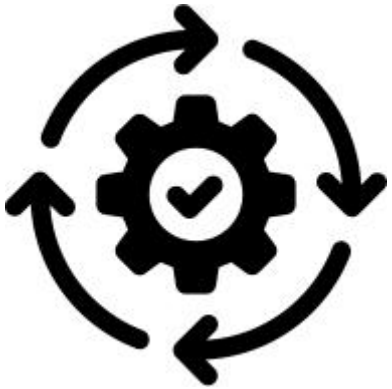
What is Machine Learning?



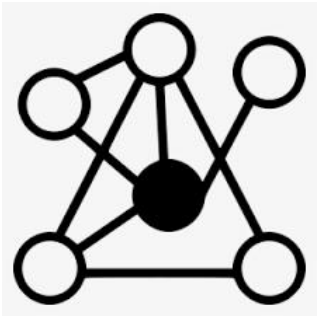
+

Dataset

+

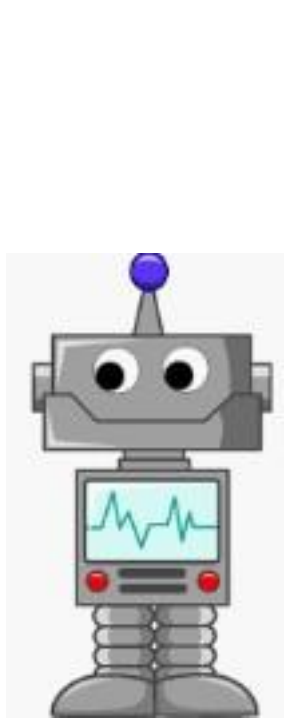


+



[Identify relations]

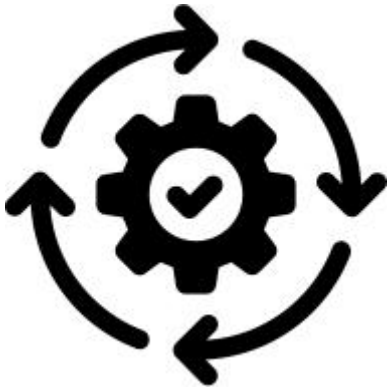
What is Machine Learning?



+

Dataset

+



+

=

[So that machine can derive inferences from the data]

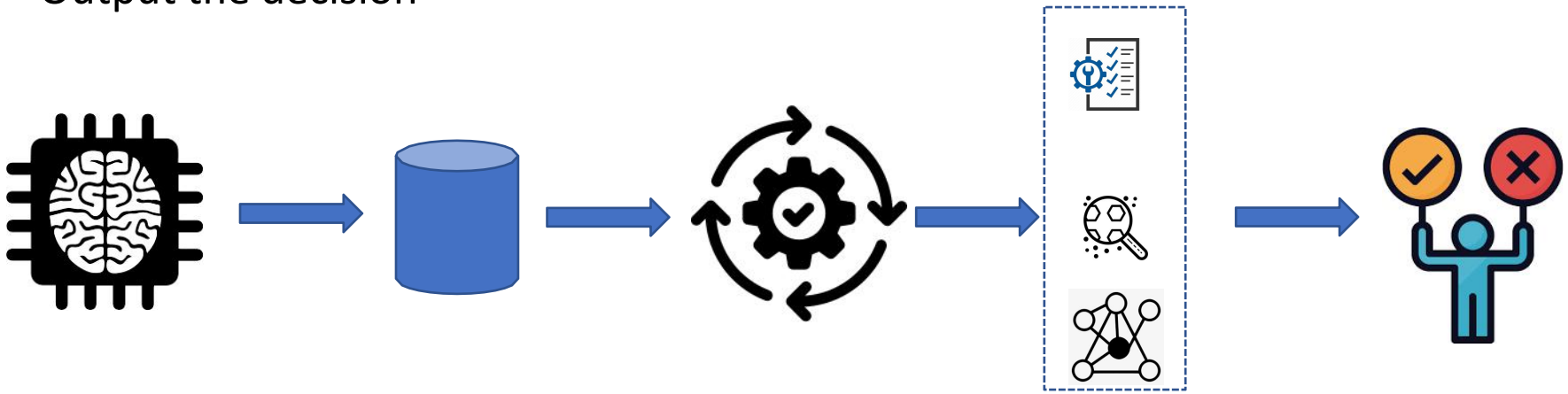
What is Machine Learning?

In Summary, Machine Learning is the ability of machine to learn things based on past data and draw inferences. These inferences can be used to make decisions about new data.

In summary, what is machine learning?

Given a machine learning problem

- Identify and create the appropriate dataset
- Perform computation to learn
  - Required rules, pattern and relations
- Output the decision



Machine Learning Paradigms

- Supervised
- Unsupervised Learning
- Reinforcement learning

[We as human being solve various types of problem in our day-to-day life, <pause> Various decisions need to be taken.
Depending on the nature of the problem, machine learning tasks can be broadly divided in]

What is Supervised Learning?

CAR

CAR

+

BIKE

BIKE

Samples Labels

= **Training Dataset**

[In supervised learning, we need some thing called a Labelled Training Dataset]

What is Supervised Learning?




CAR

CAR

BIKE

BIKE

Samples

Labels

= Training Dataset

$f($  $,\quad\quad)=$

[Given a labelled dataset, the task is to devise a function which takes the dataset, and a new sample, and produces an output value.]

What is Supervised Learning?




CAR

CAR

BIKE

BIKE

Samples Labels

= Training Dataset

$f($  ,  $)=$

[Given a labelled dataset, the task is to devise a function which takes the dataset, and a new sample, and produces an output value.]

What is Supervised Learning?




CAR

CAR

BIKE

BIKE

Samples

Labels

= Training Dataset

$f($  ,  $)$ = CAR

[Given a labelled dataset, the task is to devise a function which takes the dataset, and a new sample, and produces an output value.]

What is Supervised Learning?


CAR


CAR

+


BIKE


BIKE

Samples Labels

= **Training Dataset**

Classification

$f($  ,  $)$ = CAR

[If the possible output values of the function are predefined and discrete/categorical, it is called Classification

What is Supervised Learning?


CAR


CAR

\+


BIKE


BIKE

Samples Labels

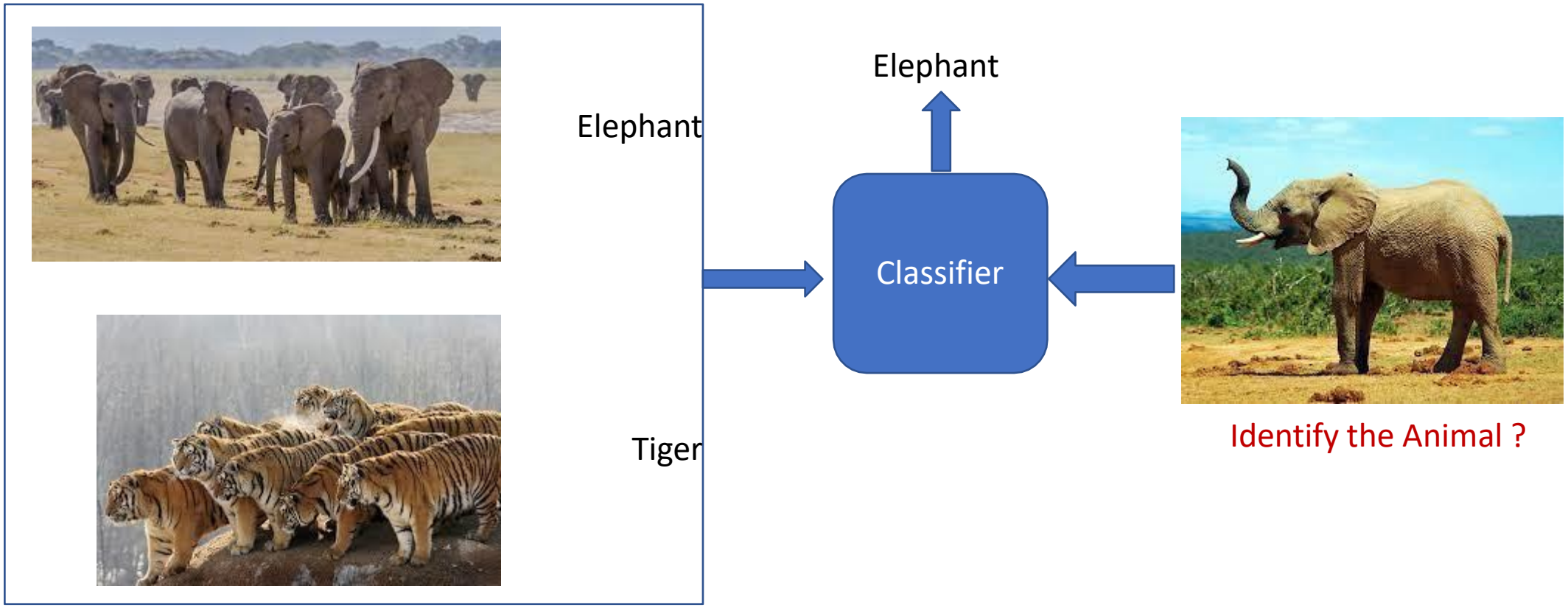
= **Training Dataset**

Classification

$f($  $)=$ CAR

[Predefined classes means, it will produce output only from the labels defined in the dataset. For example, even if we input a bus, it will produce either CAR or BIKE]

Classifier


Elephant
Tiger
Classifier
Elephant
Identify the Animal ?
Dataset

Regression

Dataset

Regression

$f($  ,  $) = 20500.50$

[If the possible output values of the function are continuous real values, then it is called Regression

[
The classification and Regression problems are supervised, because the decision depends on the characteristics of the ground truth labels or values present in the dataset, which we define as experience
]

What is Unsupervised Learning


~~CAR~~


~~CAR~~


~~BIKE~~


~~BIKE~~

Dataset

[In the unsupervised learning, we do not need to know the labels or Ground truth values]

What is Unsupervised Learning

Dataset

Clustering

[The task is to identify the patterns like group the similar objects together]

What is Unsupervised Learning

Dataset

Association Rules Mining

[Association rules like]

More Example Unsupervised Learning

Dataset

More Example Unsupervised Learning

Dataset

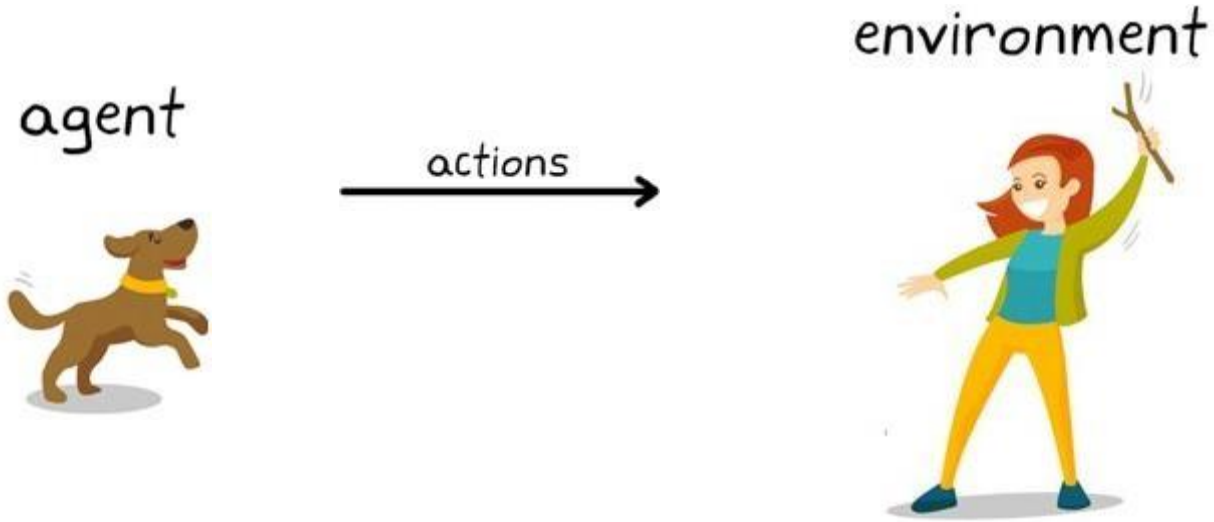
More Example Unsupervised Learning

What is Reinforcement Learning

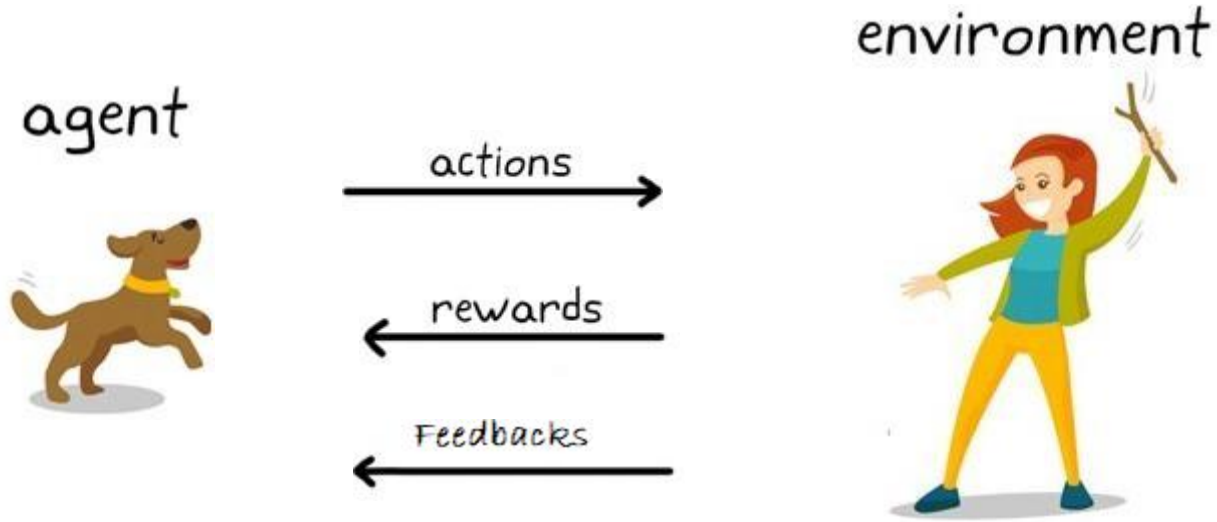
[It is also known as learning from trials and errors]

What is Reinforcement Learning

What is Reinforcement Learning



What is Reinforcement Learning



Another Example

Agent

Task

Environment

Reinforcement Learning

Punishment

Reinforcement Learning

Reward

Reinforcement Learning

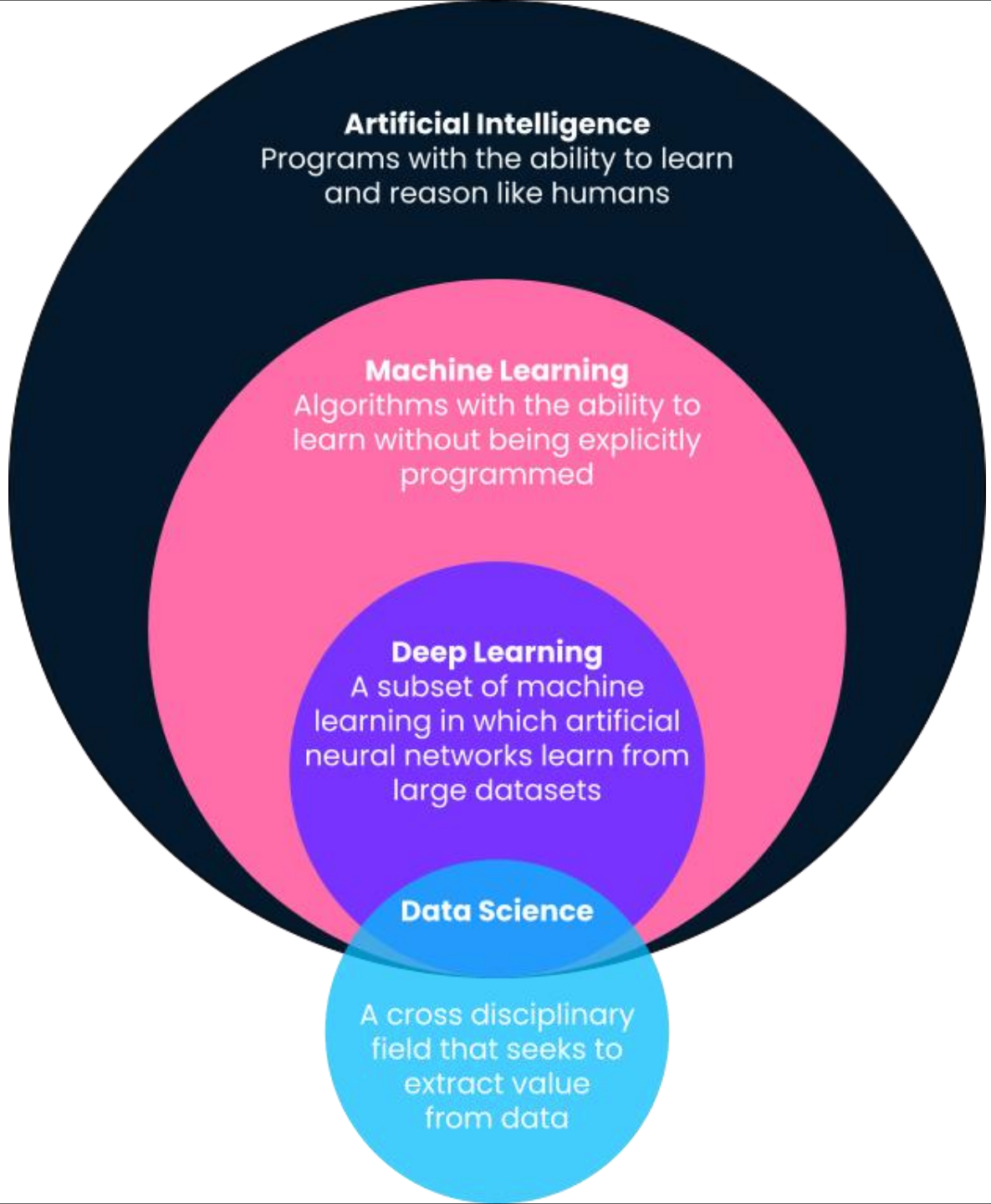
Reward

Baby Learn from the Trials and Errors

Reinforcement Learning

SUMMARY

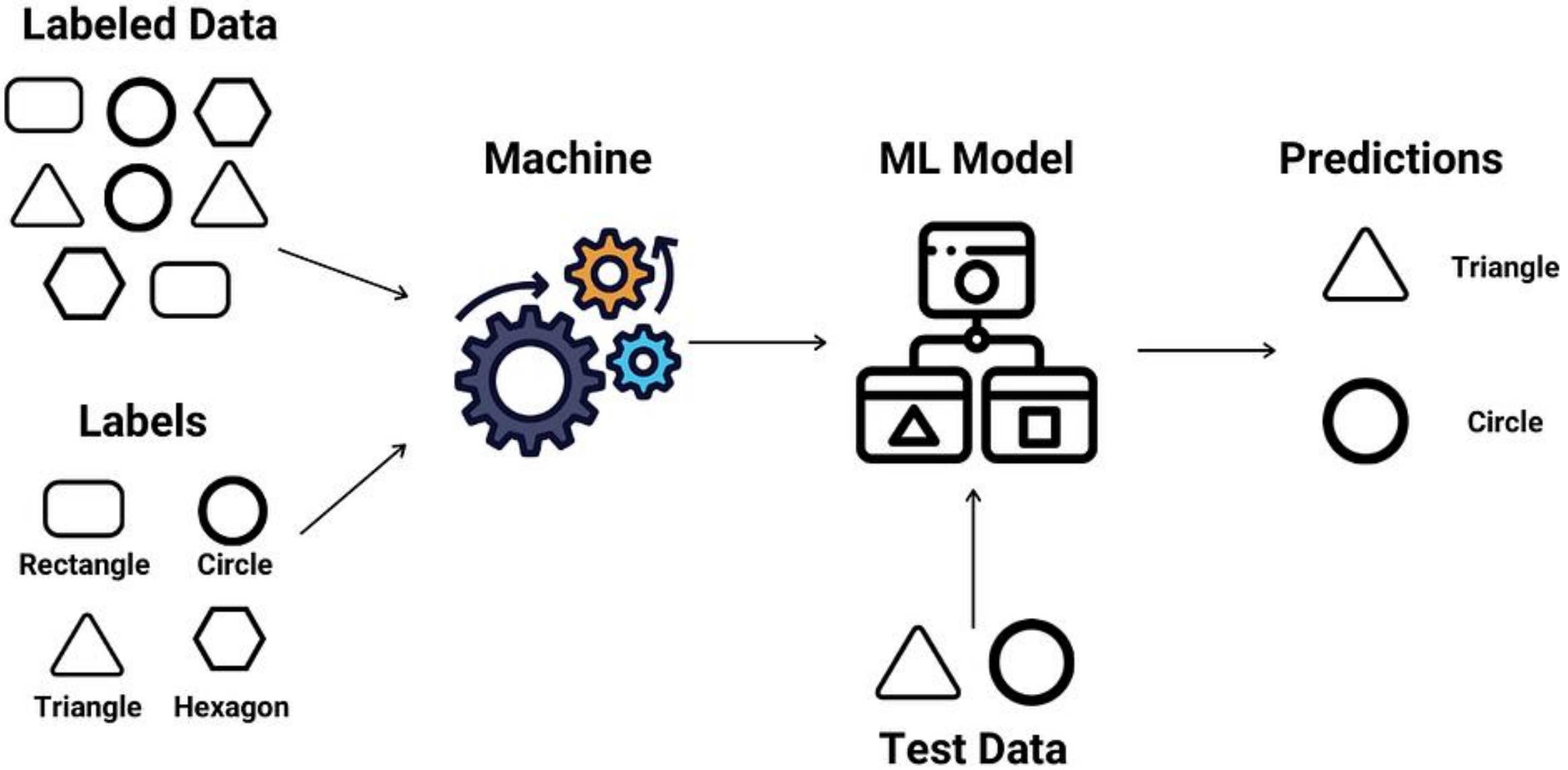
- What is Machine Learning?
  - ML is the field of study that gives computer algorithms the ability to learn without being explicitly programmed.
  - In the ML, we train algorithm with a set of known data and then we give it some new data and ask the algorithm to predict a reasonable result.
- Types of Machine Learning
  - Supervised learning
  - Unsupervised learning
  - Semi-Supervised learning
  - Reinforcement learning


Artificial Intelligence
Programs with the ability to learn and reason like humans
Machine Learning
Algorithms with the ability to learn without being explicitly programmed
Deep Learning
A subset of machine learning in which artificial neural networks learn from large datasets
Data Science
A cross disciplinary field that seeks to extract value from data

Supervised Learning

- In supervised learning, the model is trained on labeled data, meaning that both the input (features) and the output (target or class label) are known during training.
- The goal is to learn a mapping from the input features to the output labels so that the model can make predictions on new, unseen data.
- There are mainly two types of tasks in supervised learning -
  - Classification
    - Classification models classify our outputs to certain categories.
    - Like Whether given patient has cancer or not, Given email is spam or not, etc.
  - Regression
    - Regression models are for labeling outputs with continuous values.
    - Like predicting house prices, predicting tomorrow's temperature, etc.

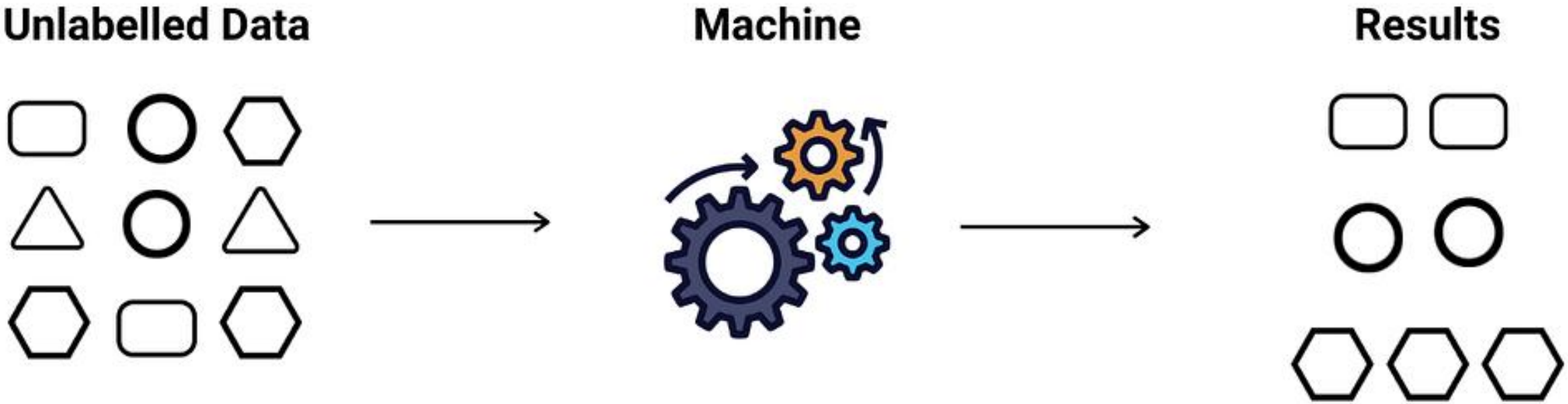
Supervised Learning



Unsupervised Learning

- In unsupervised learning, the model is trained on data without labeled outputs.
- The goal is to identify patterns, groupings, or structures in the data. The model does not have predefined labels to guide it, so it must discover these on its own.
- Some examples of unsupervised learning tasks are -
  - Clustering
  - Dimensionality Reduction

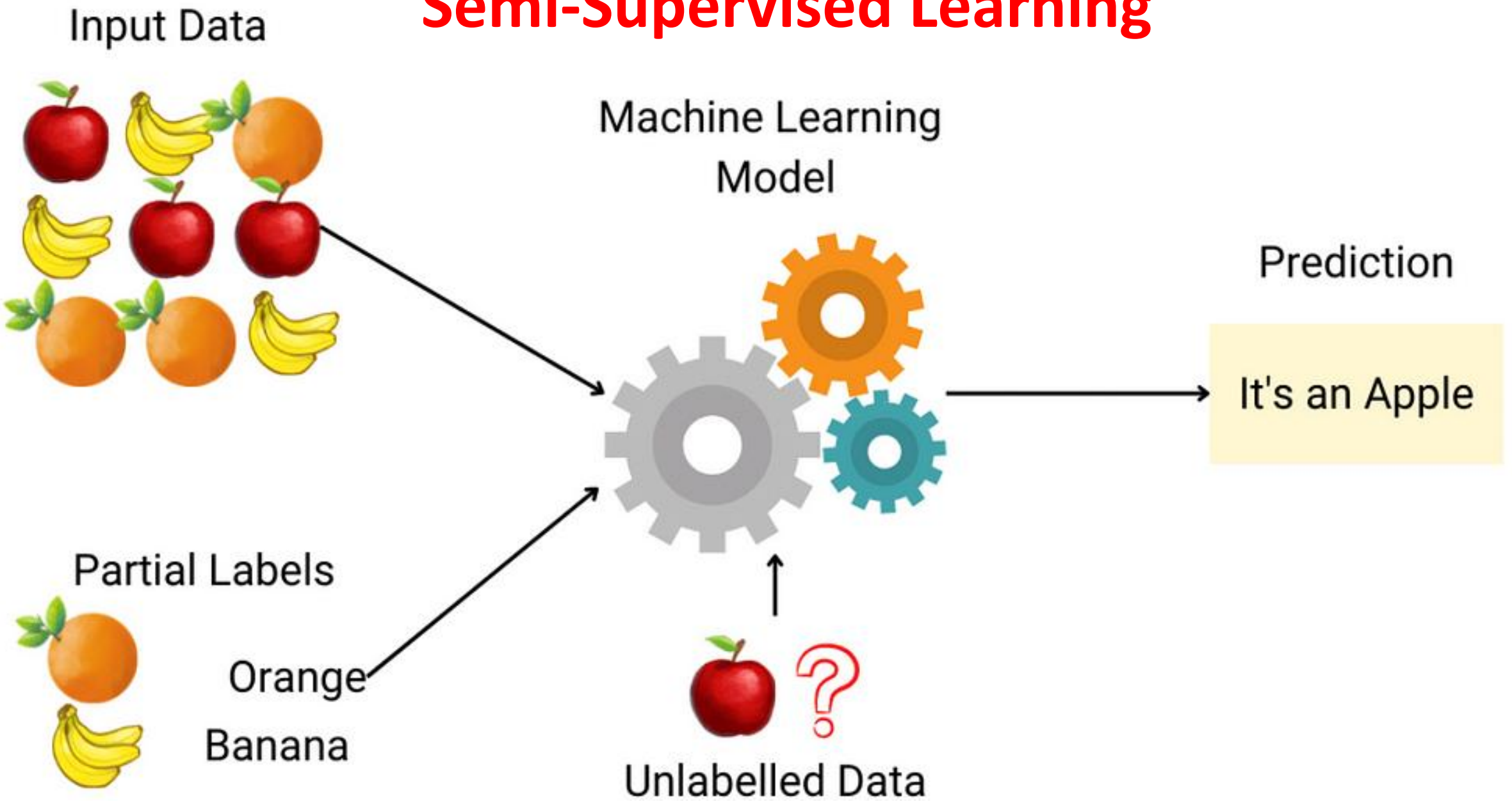
Unsupervised Learning



Semi-Supervised Learning

- Semi-supervised learning is a mix of supervised and unsupervised learning. The model is trained on a small amount of labeled data and a larger amount of unlabeled data.
- It is useful when labeling data is expensive or time-consuming, but there is an abundance of unlabeled data available.

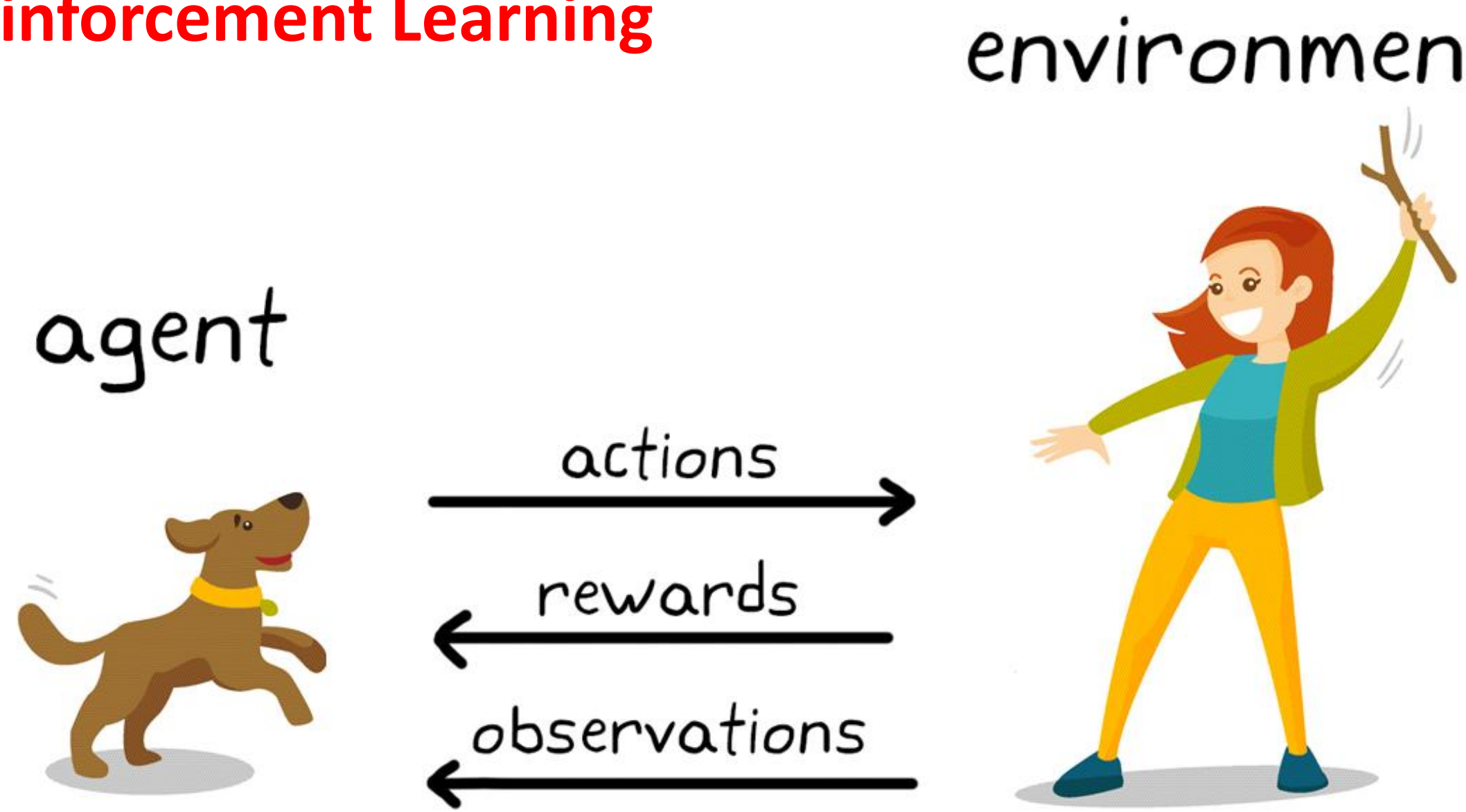
Semi-Supervised Learning

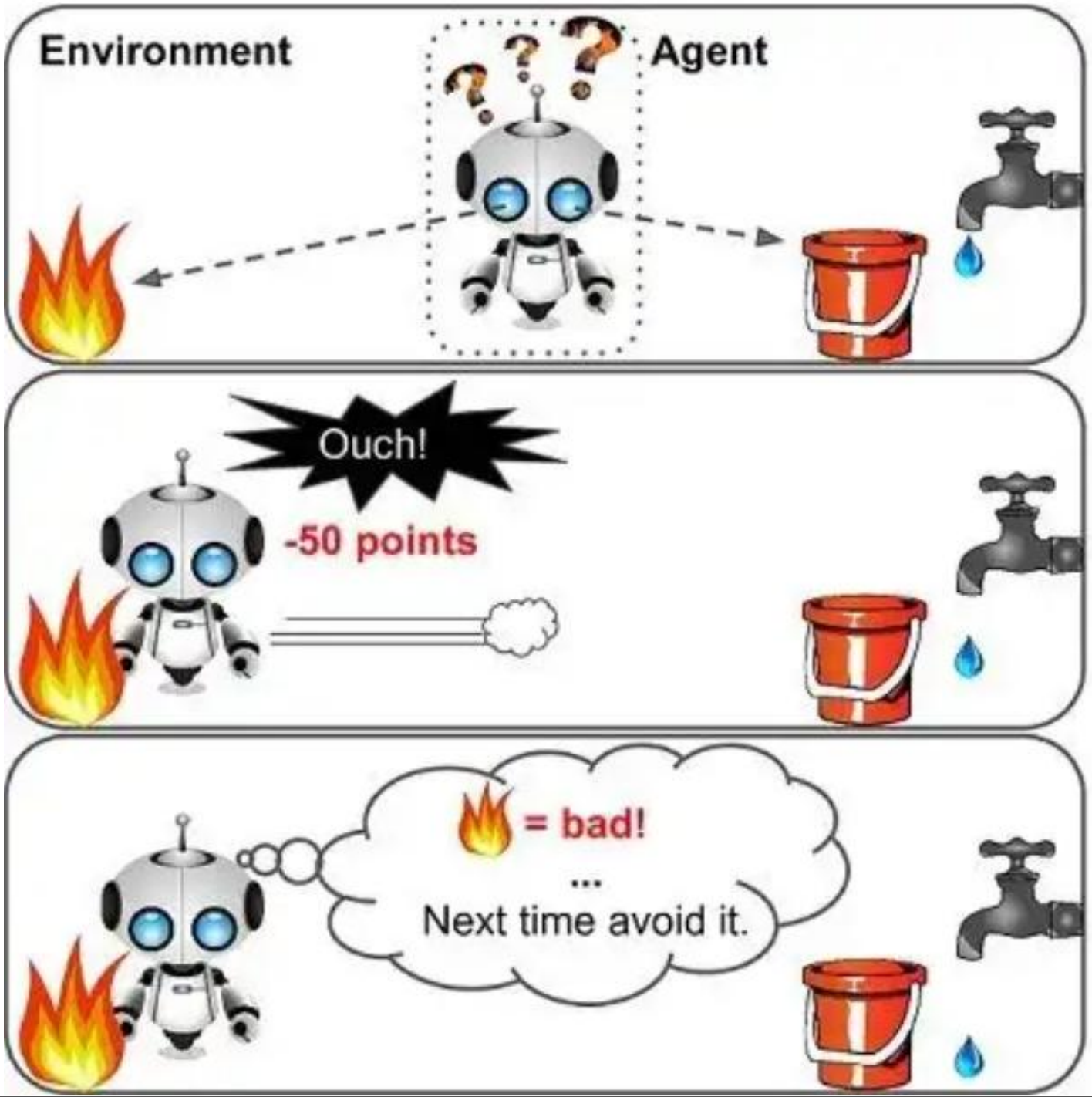


Reinforcement Learning

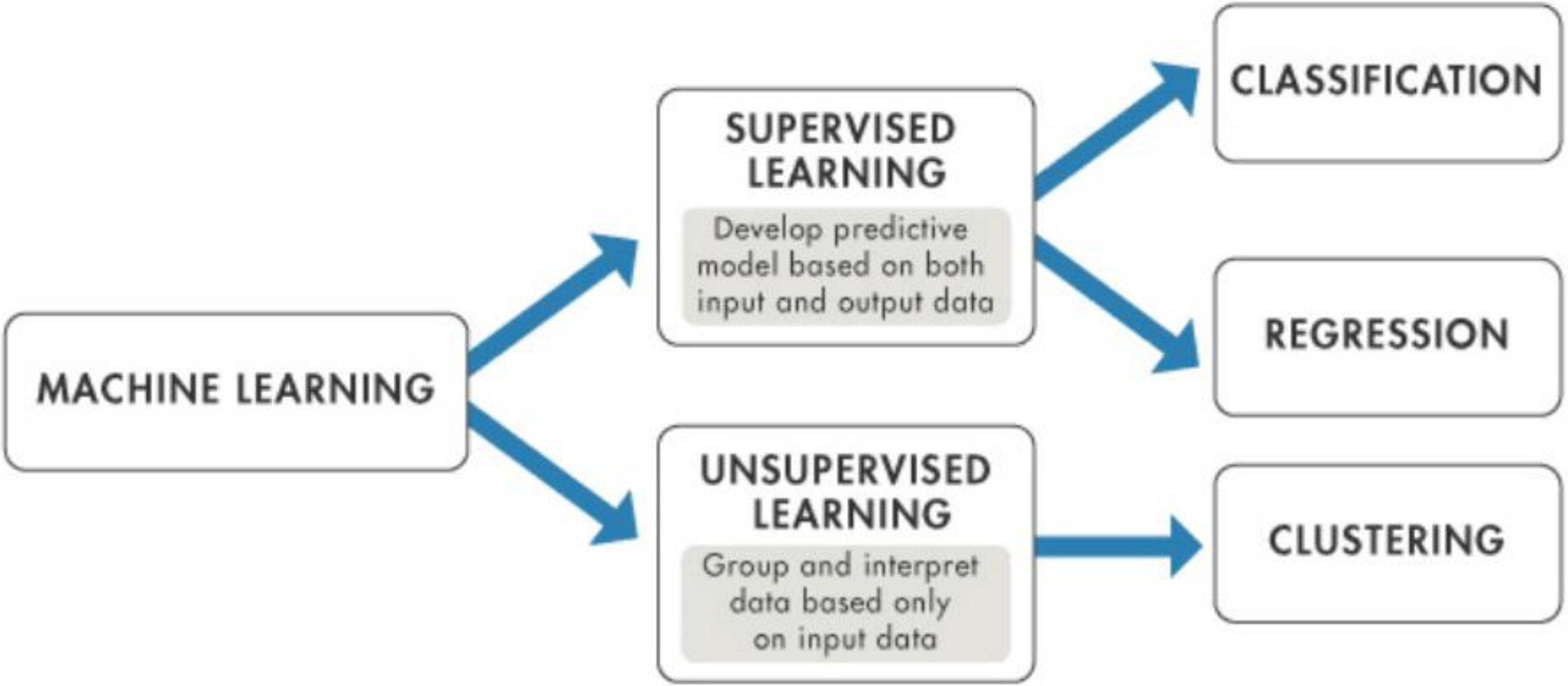
- Reinforcement learning (RL) involves an agent that learns to make decisions by interacting with an environment.
- The agent receives feedback in the form of rewards or penalties based on its actions.
- The goal is to learn a strategy (policy) that maximizes the cumulative reward over time.

Reinforcement Learning

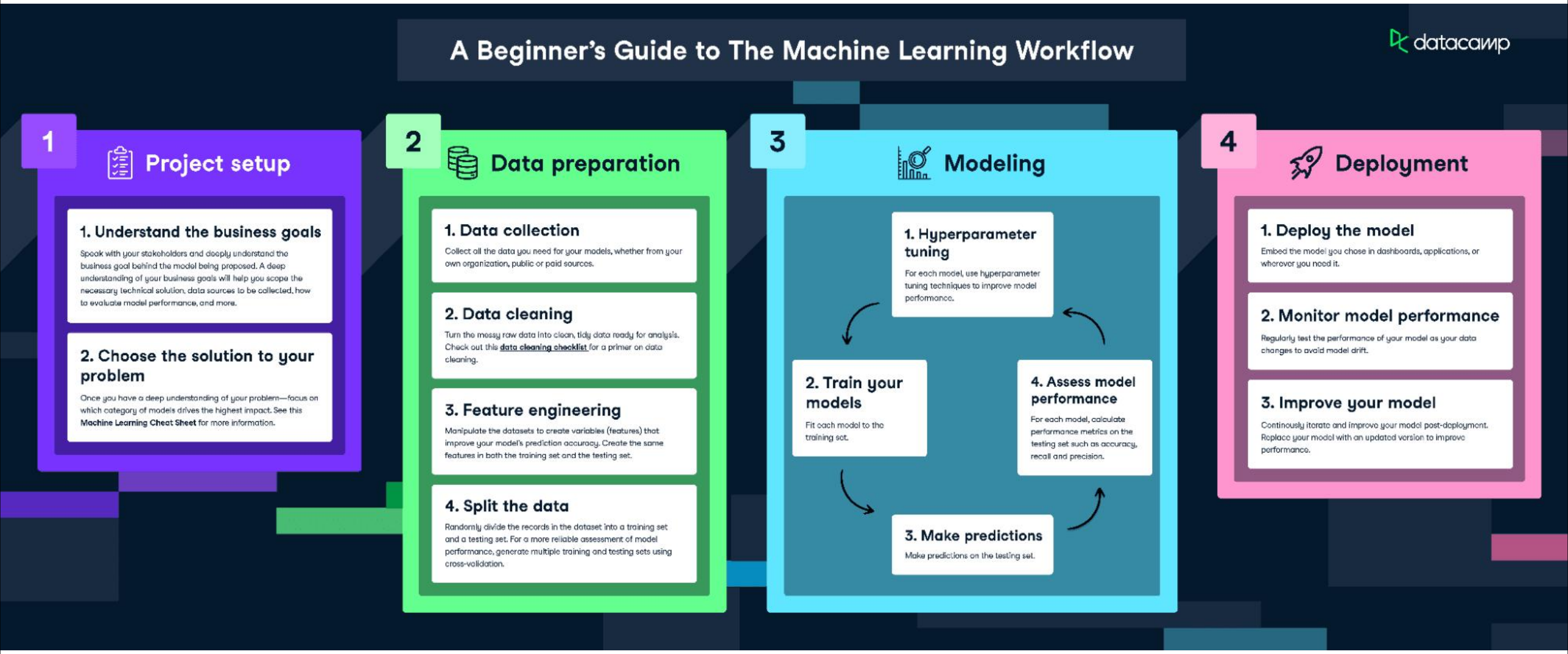




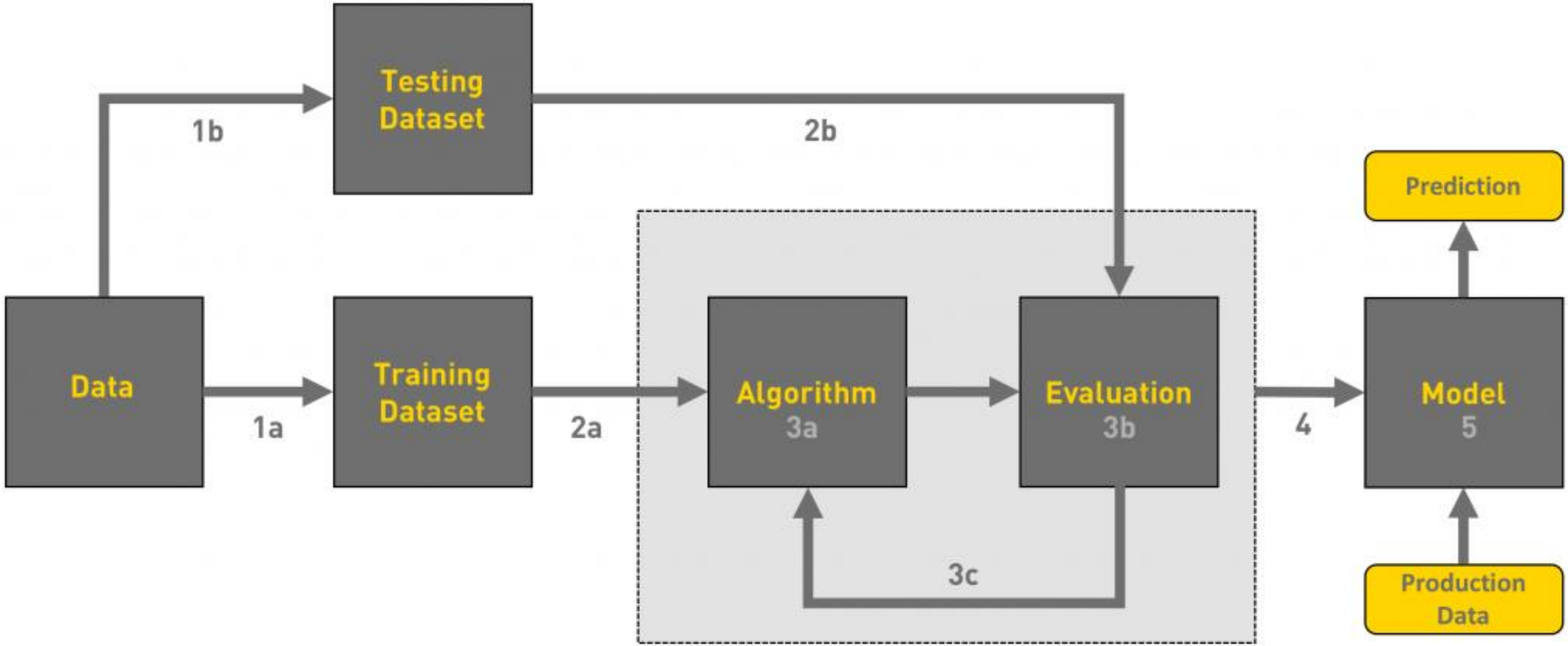
1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

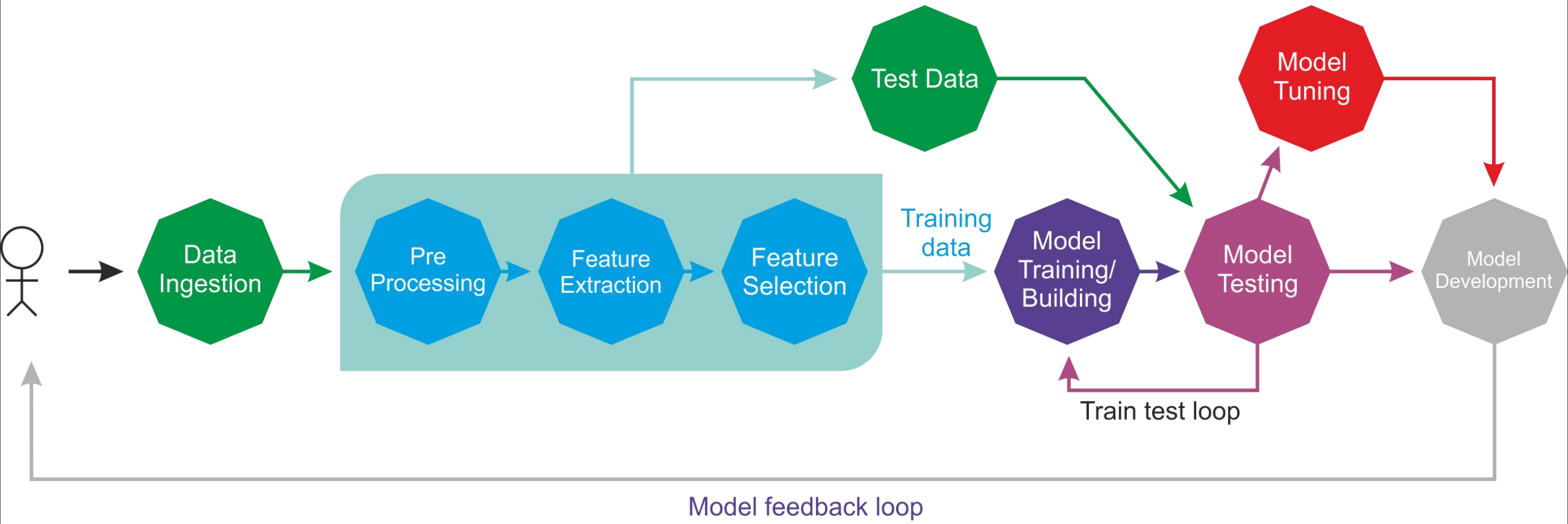
Machine Learning Workflow



Machine Learning Workflow

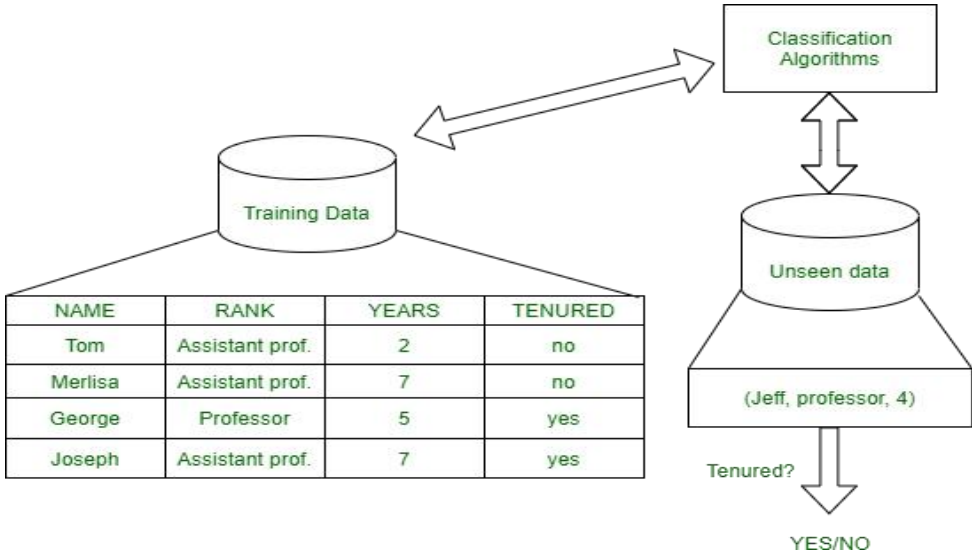


Machine Learning Workflow



CLASSIFICATION

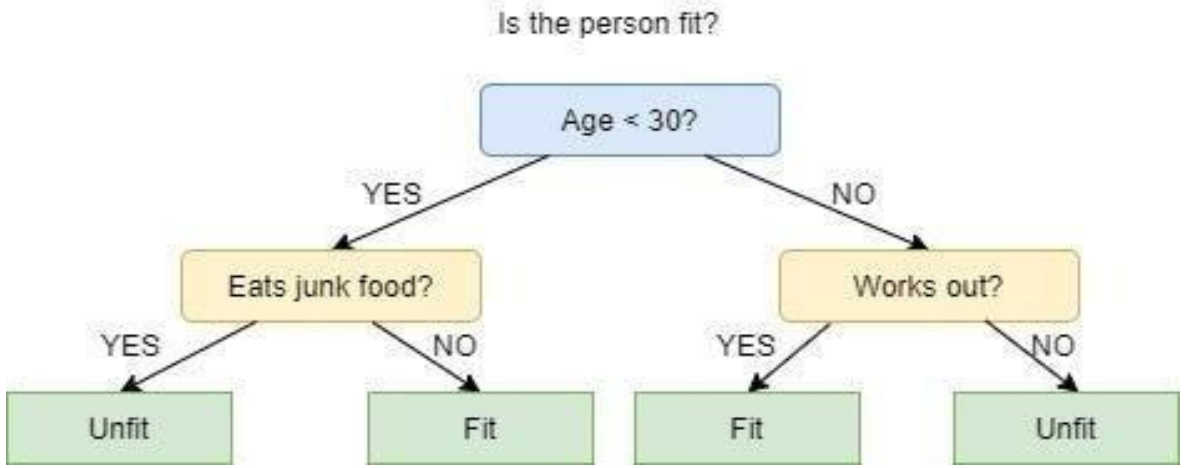
- Classification is a task that involves separating data points into different classes i.e., assigning a class label to each data point instance in a dataset based on its features. The goal of classification is to build a model that accurately predicts the class labels of new instances based on their features.
- There are two main types of classification: **binary classification** and **multi-class classification.** Binary classification involves classifying instances into two classes, such as “spam” or “not spam” in email classifier, while multi-class classification involves classifying instances into more than two classes.



NAME	RANK	YEARS	TENURED
Tom	Assistant prof.	2	no
Merlisa	Assistant prof.	7	no
George	Professor	5	yes
Joseph	Assistant prof.	7	yes

Decision Trees

- A decision tree is a structure that contains nodes and edges and is built from a dataset.
- Each node is either used to make a decision (known as decision node) or represent an outcome (known as leaf node).
- The picture depicts below represent a decision tree that is used to classify whether a person is Fit or Unfit.
- The decision nodes here are questions like 'Is the person less than 30 years of age?', 'Does the person eat junk?', etc. and the leaves are one of the two possible outcomes Fit and Unfit.
- Looking at the Decision Tree, following decisions can be looked at: if a person is less than 30 years of age and doesn't eat junk food then he is **Fit**, if a person is less than 30 years of age and eats junk food then he is **Unfit** and so on.
- The initial node is called the root node (colored in blue), the final nodes are called the leaf nodes (colored in green) and the rest of the nodes are called intermediate or internal nodes. The root and intermediate nodes represent the decisions while the leaf nodes represent the outcomes.

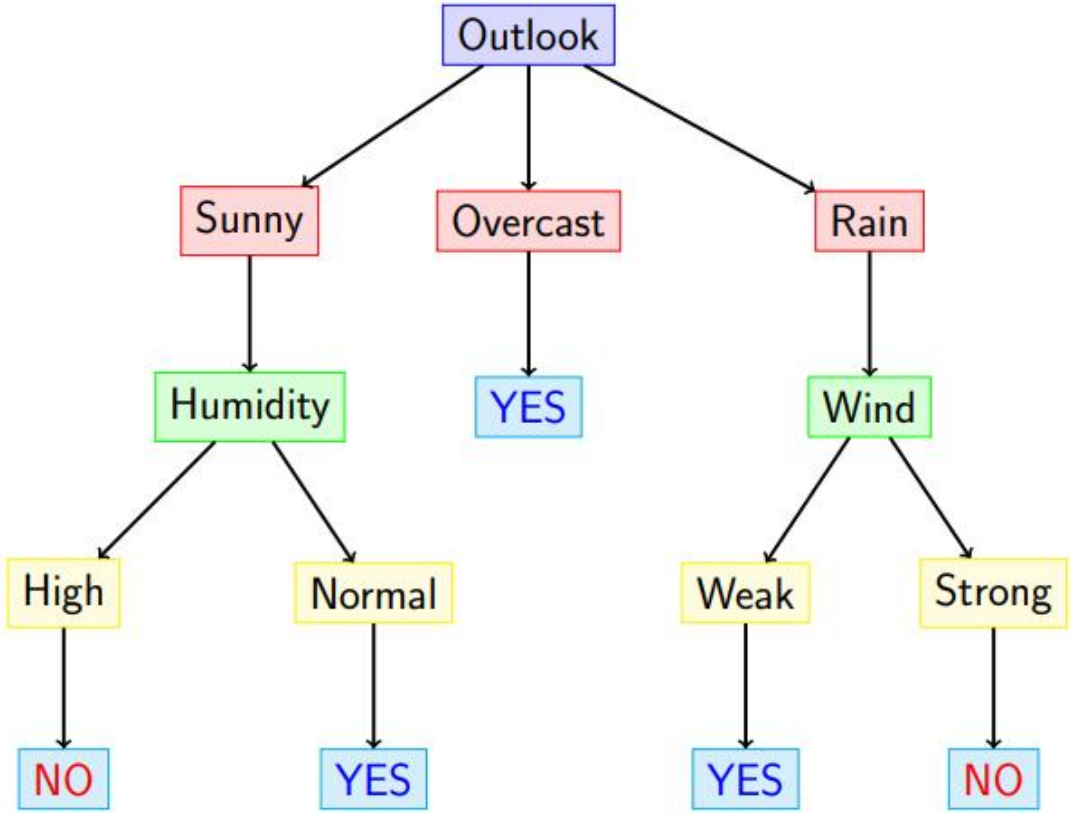


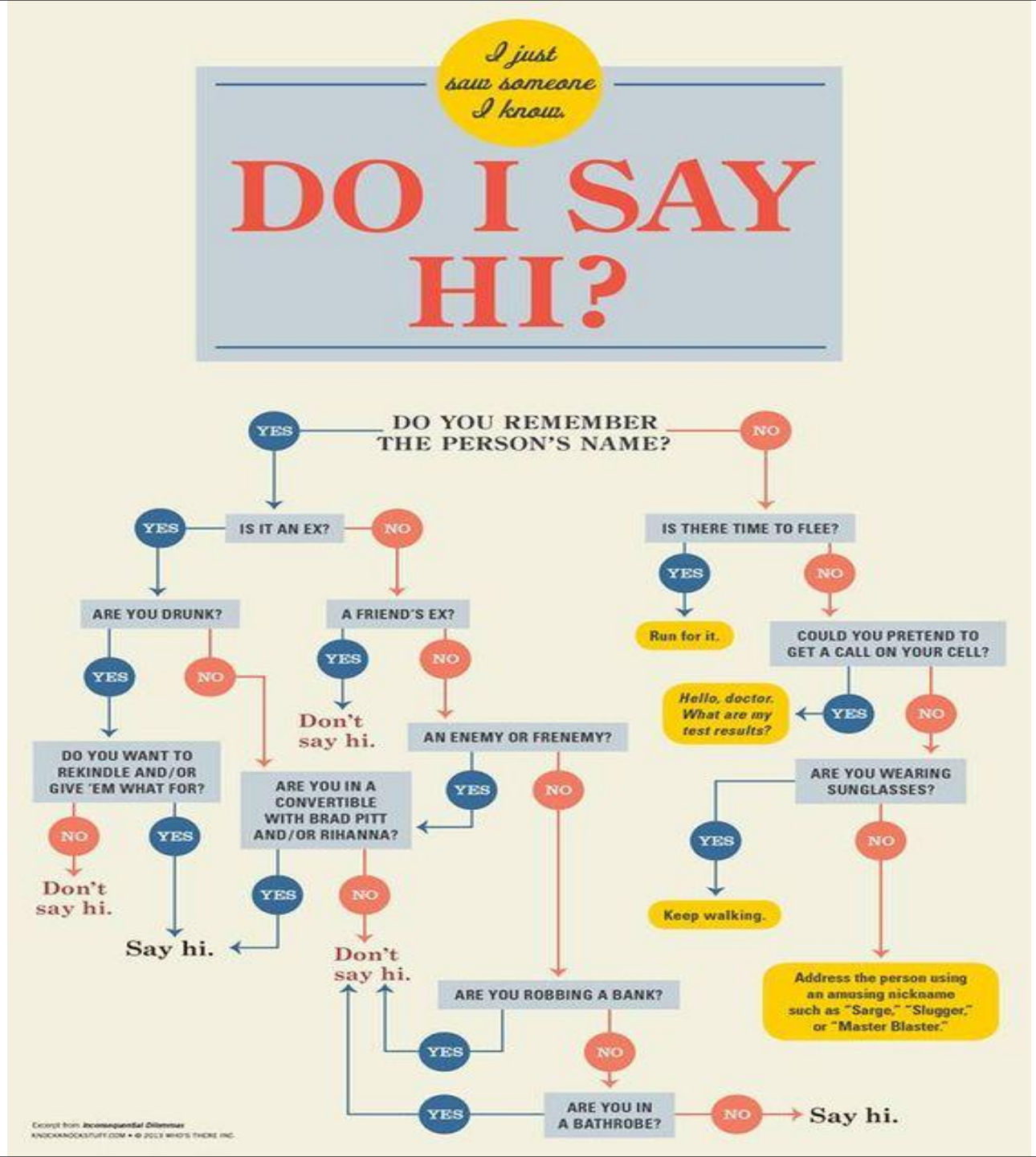
How are Decision tree used for classification?

- Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree.
- A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

EXAMPLE

- Should I play tennis today based on the weather forecast, it's raining with strong wind?




I just saw someone I know.
DO I SAY HI?
DO YOU REMEMBER THE PERSON'S NAME?
YES
NO
IS IT AN EX?
YES
NO
ARE YOU DRUNK?
YES
NO
DO YOU WANT TO REKINDLE AND/OR GIVE 'EM WHAT FOR?
NO
YES
Don't say hi.
Say hi.
A FRIEND'S EX?
YES
NO
Don't say hi.
AN ENEMY OR FRENEMY?
YES
NO
ARE YOU IN A CONVERTIBLE WITH BRAD PITT AND/OR RIHANNA?
YES
NO
Say hi.
Don't say hi.
ARE YOU ROBBING A BANK?
YES
NO
ARE YOU IN A BATHROBE?
YES
NO
Say hi.
IS THERE TIME TO FLEE?
YES
NO
Run for it.
COULD YOU PRETEND TO GET A CALL ON YOUR CELL?
Hello, doctor. What are my test results?
YES
NO
ARE YOU WEARING SUNGLASSES?
YES
NO
Keep walking.
Address the person using an amusing nickname such as "Sarge," "Slugger," or "Master Blaster."
Excerpt from Inconsequential Dilemmas

Why are decision tree classifiers so popular?

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting and therefore is appropriate for exploratory knowledge discovery.
- Decision trees can handle multidimensional data.
- Their representation of acquired knowledge in tree form is intuitive and generally easy to understand by humans.
- The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy. However, successful use may depend on the data at hand.
- Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems.

DECISION TREE INDUCTION

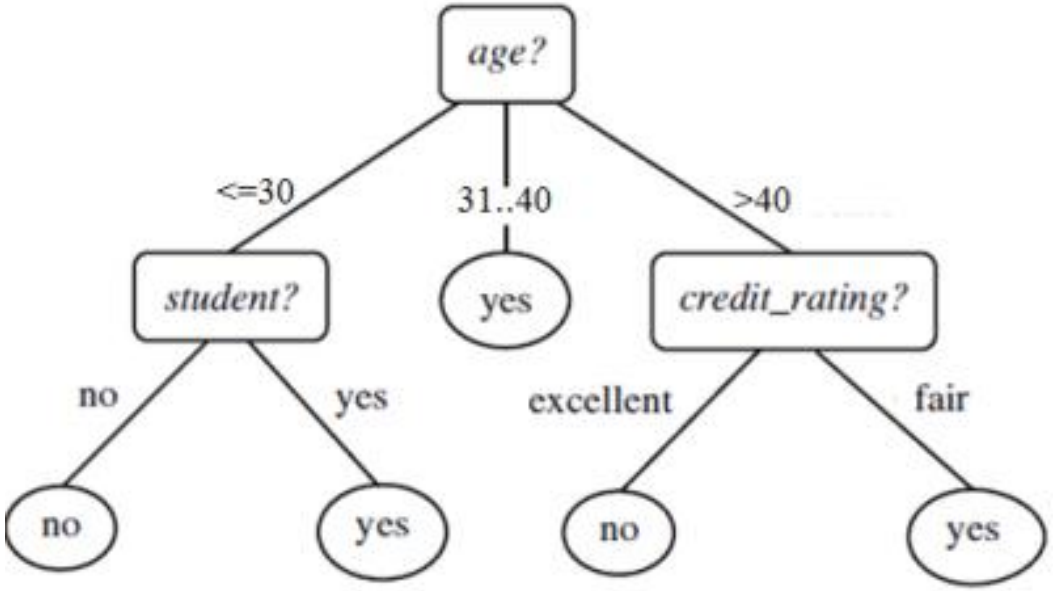
- The process of learning a decision tree model from training dataset is called Decision Tree Induction.

Training Data Set: ***buys_computer***

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

attributes

class attribute

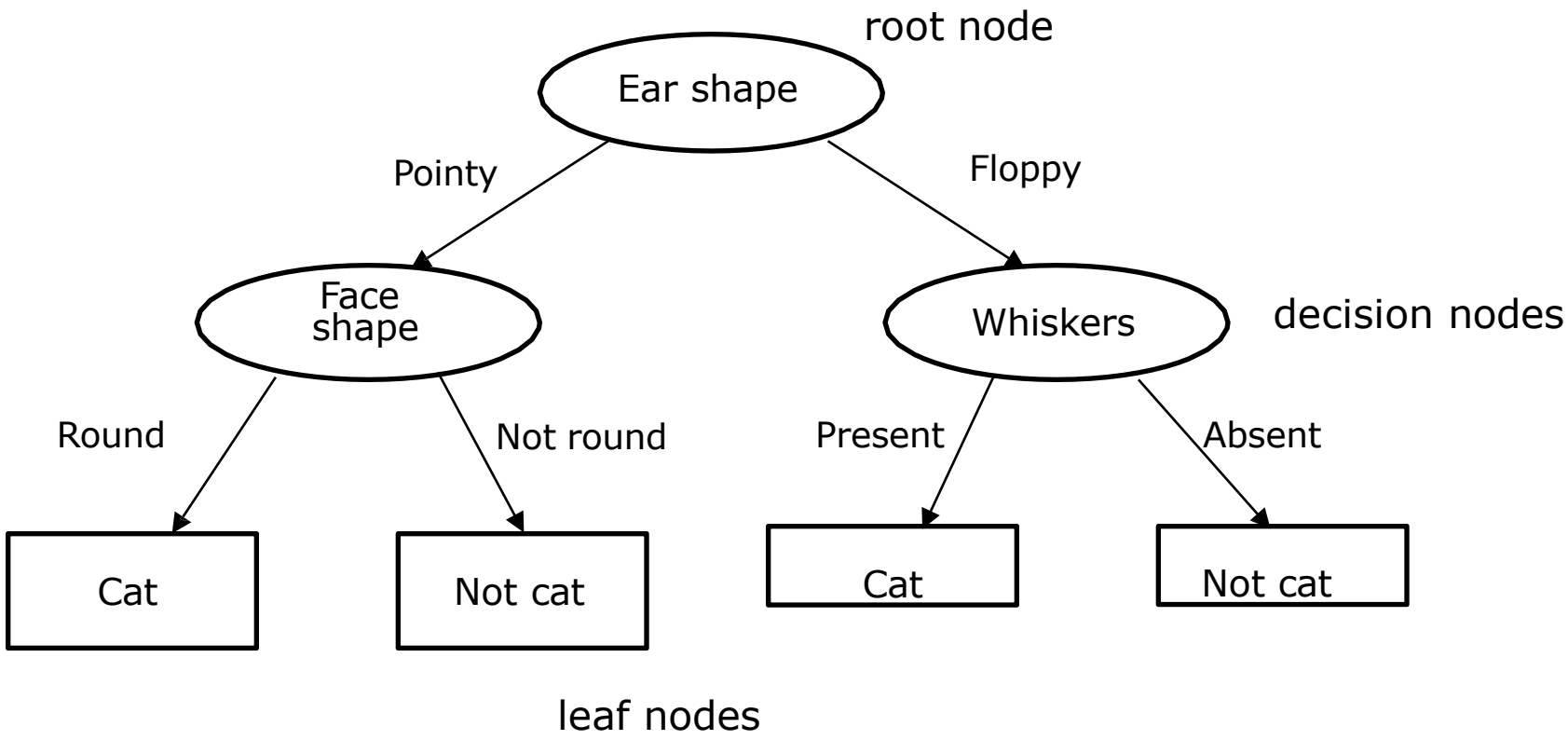


Cat classification example

	Ear shape (x_1)	Face shape (x_2)	Whiskers (x_3)	Cat
	Pointy	Round	Present	1
	Floppy	Not round	Present	1
	Floppy	Round	Absent	0
	Pointy	Not round	Present	0
	Pointy	Round	Present	1
	Pointy	Round	Absent	1
	Floppy	Not round	Absent	0
	Pointy	Round	Absent	1
	Floppy	Round	Absent	0
	Floppy	Round	Absent	0

Categorical (discrete values)

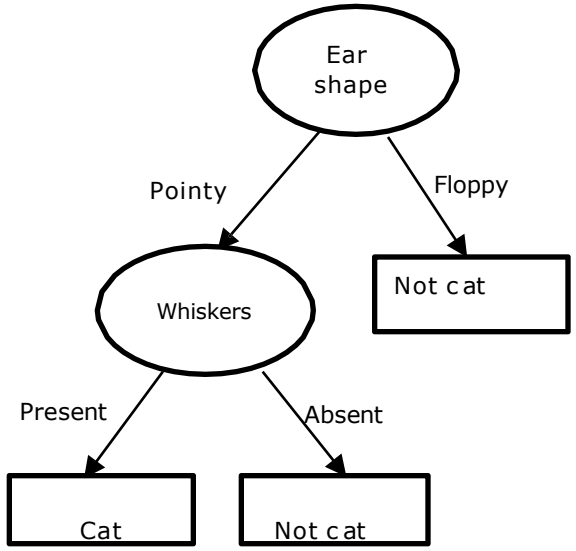
Decision Tree

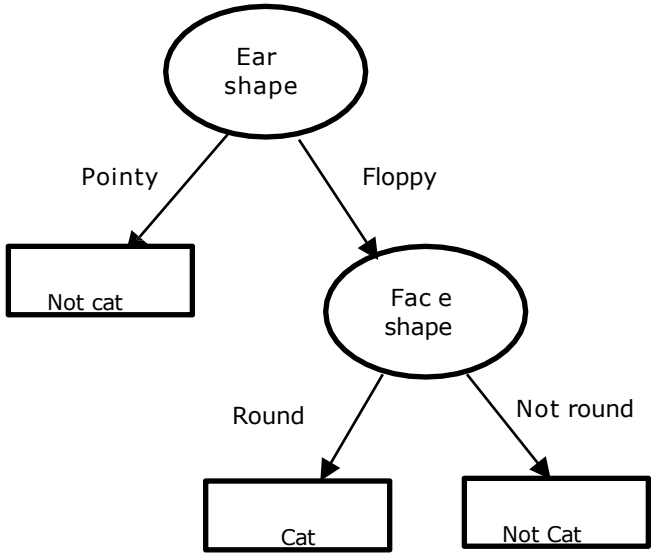


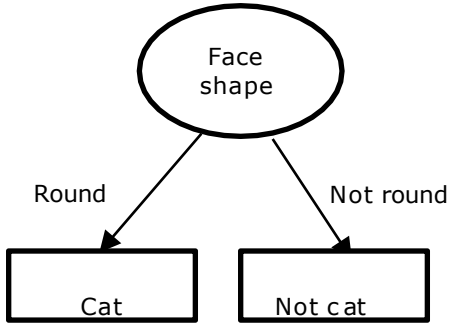
New test example

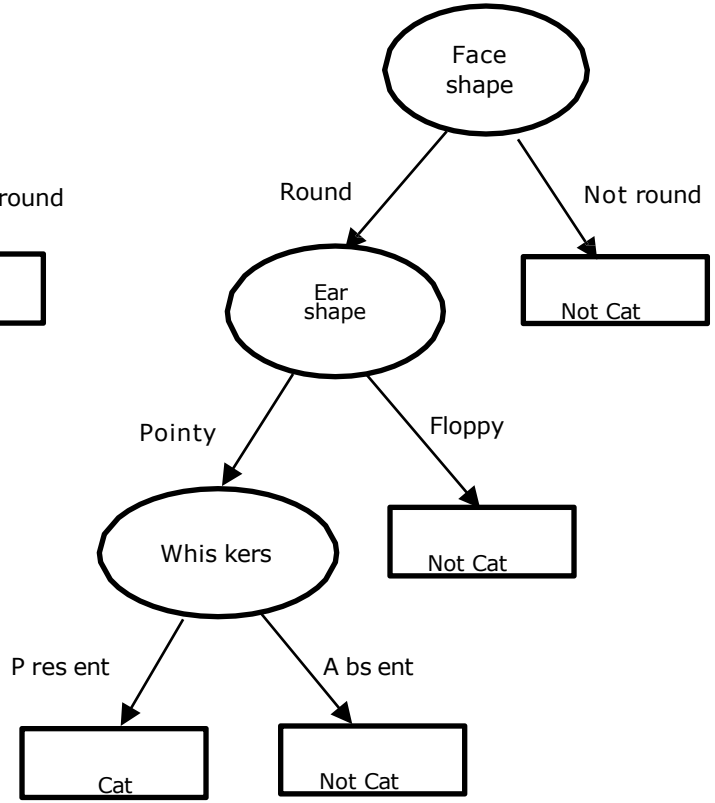
Ear shape: Pointy
Face shape: Round
Whiskers: Present

Decision Tree



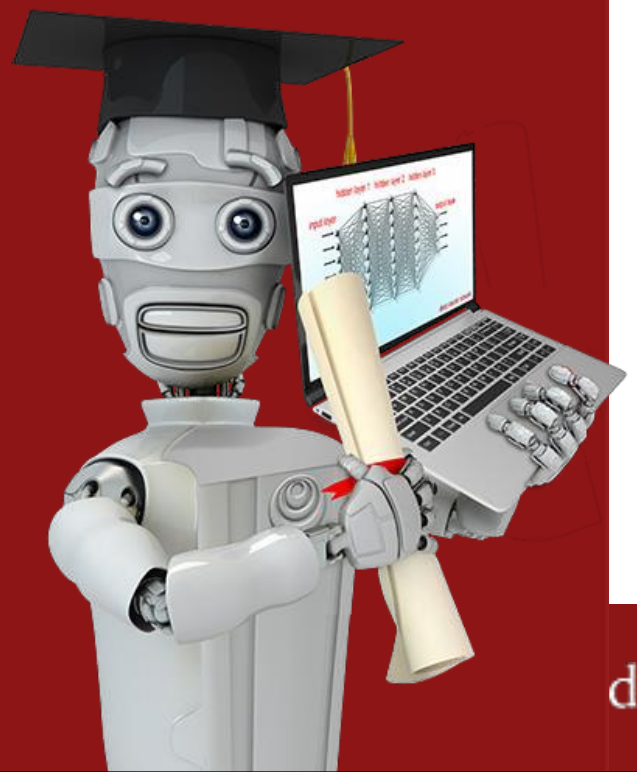






DeepLearning.AI

Stanford ONLINE

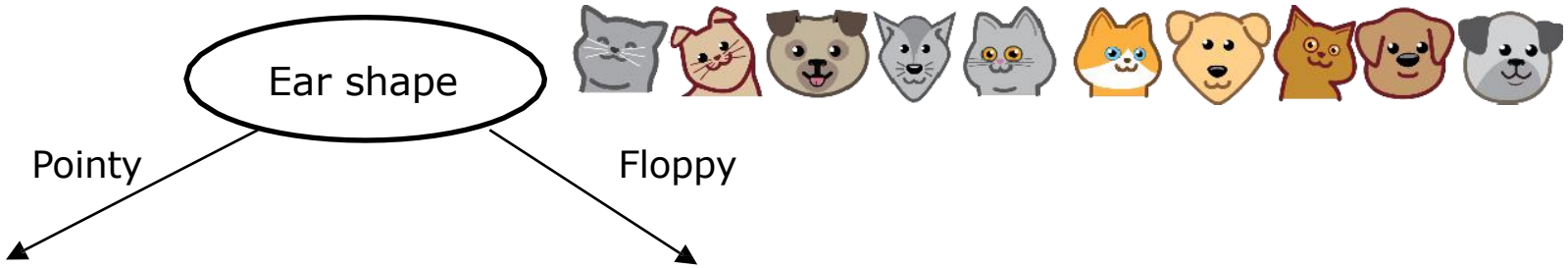


Decision Trees

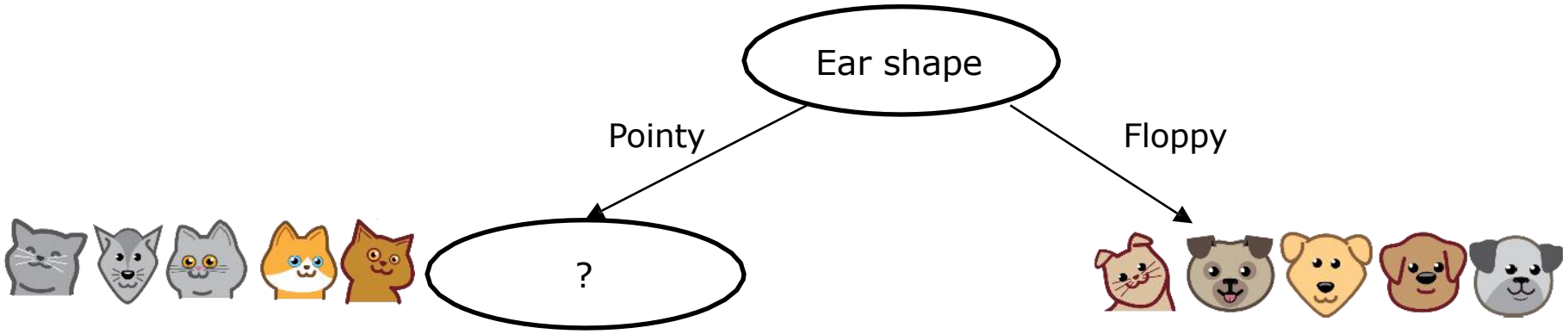
Learning Process

Decision Tree Learning

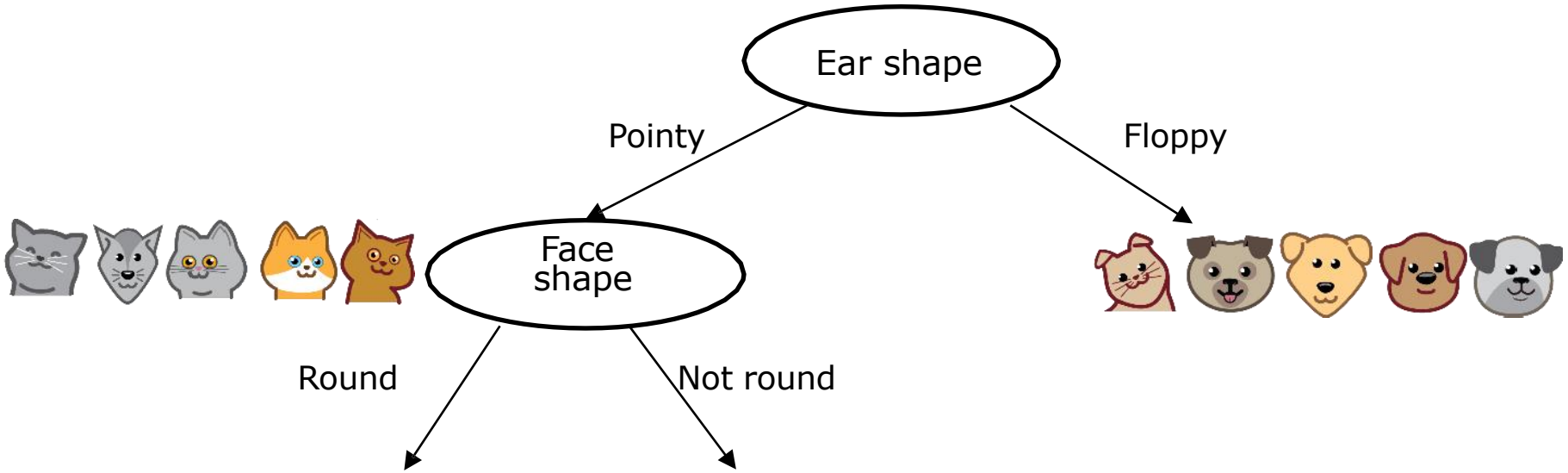
Decision Tree Learning



Decision Tree Learning

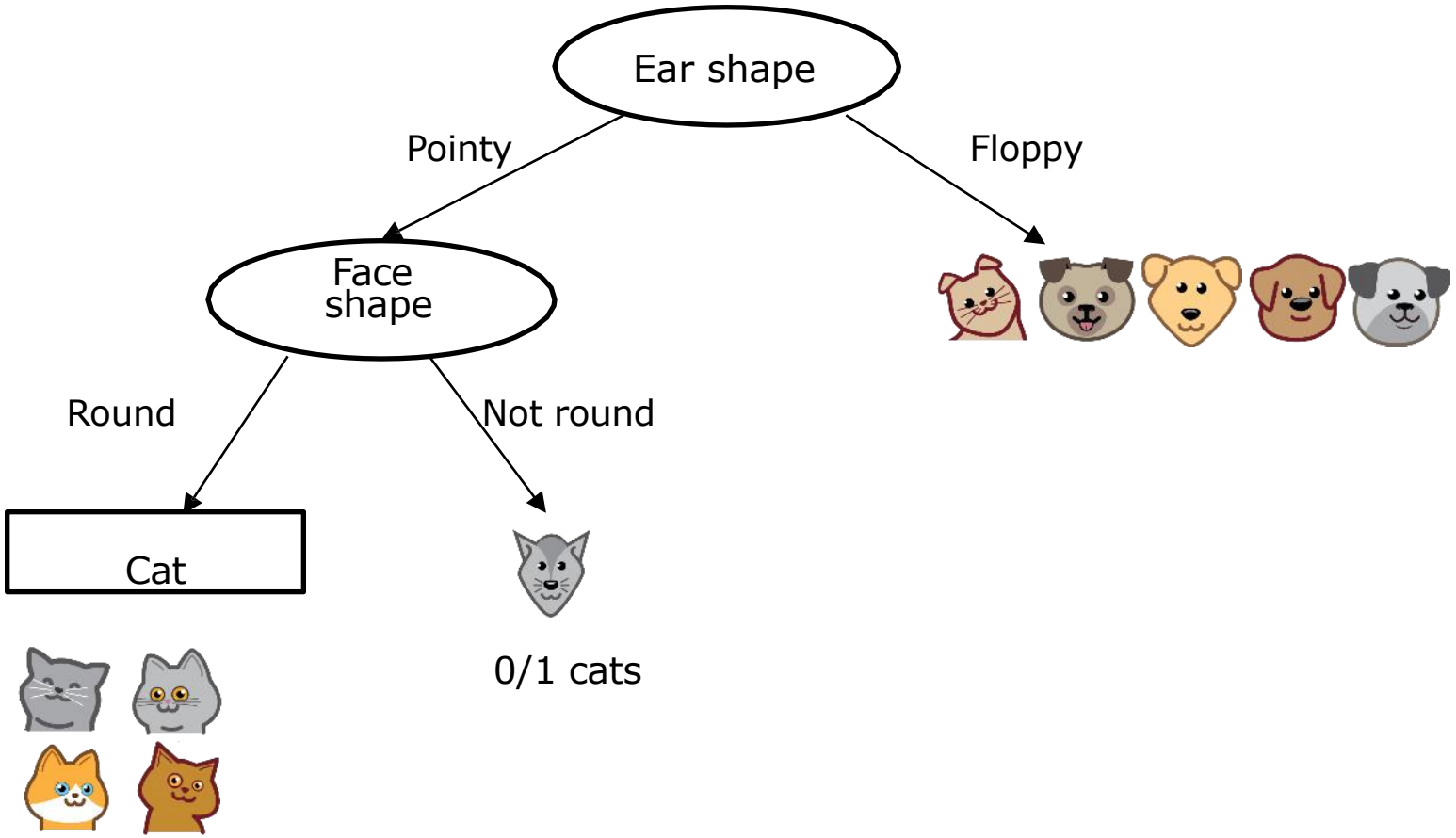


Decision Tree Learning

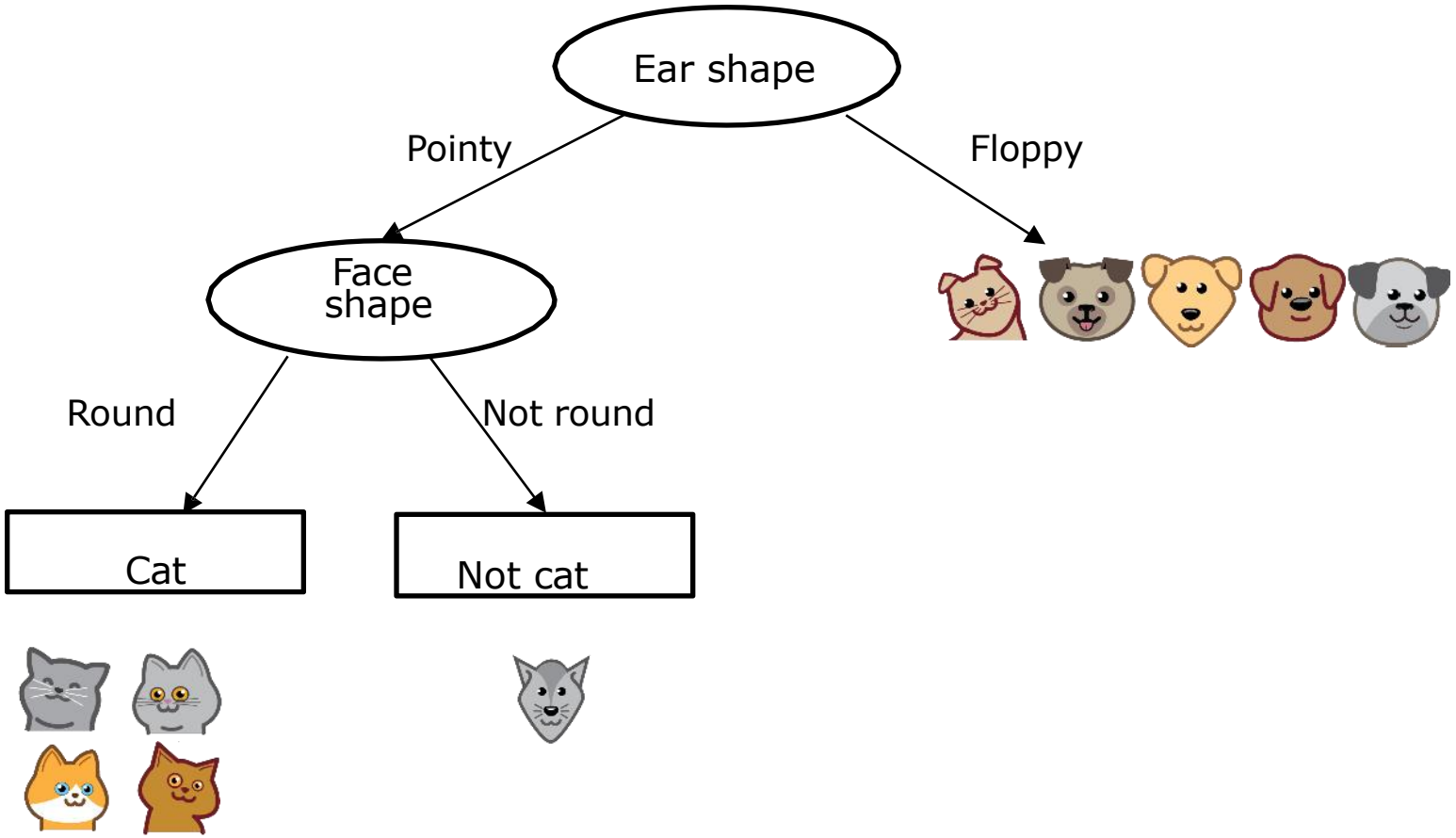


4/4 cats

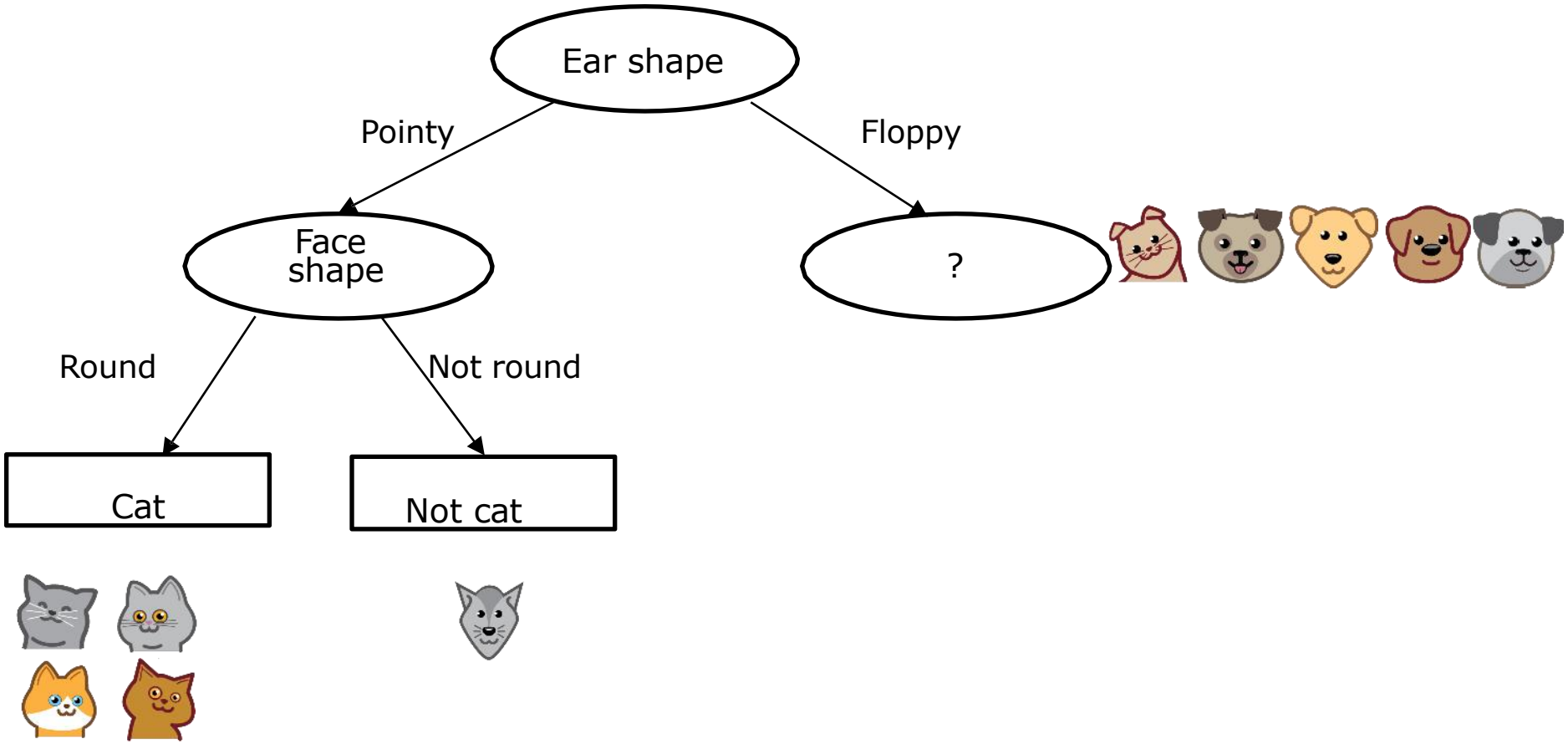
Decision Tree Learning



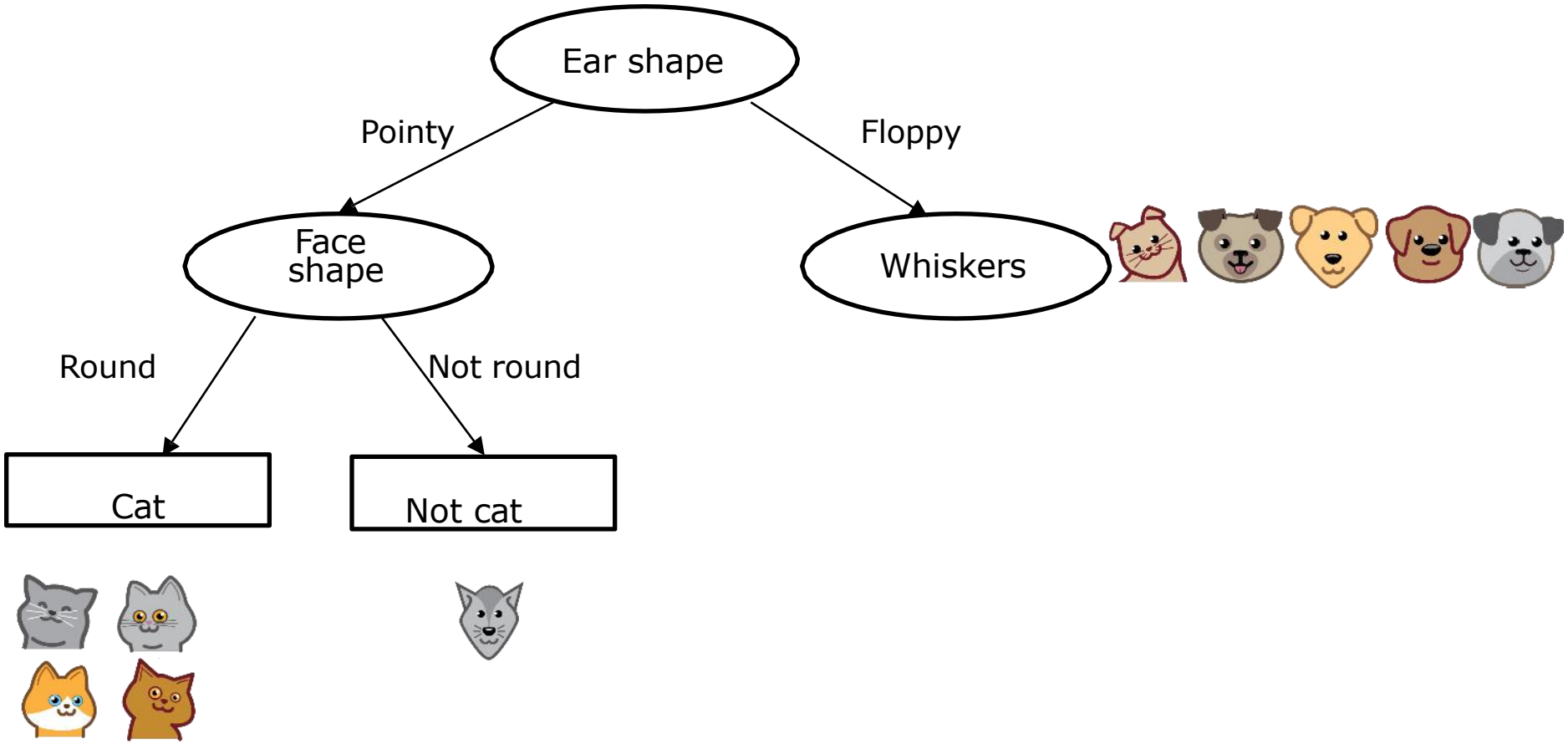
Decision Tree Learning



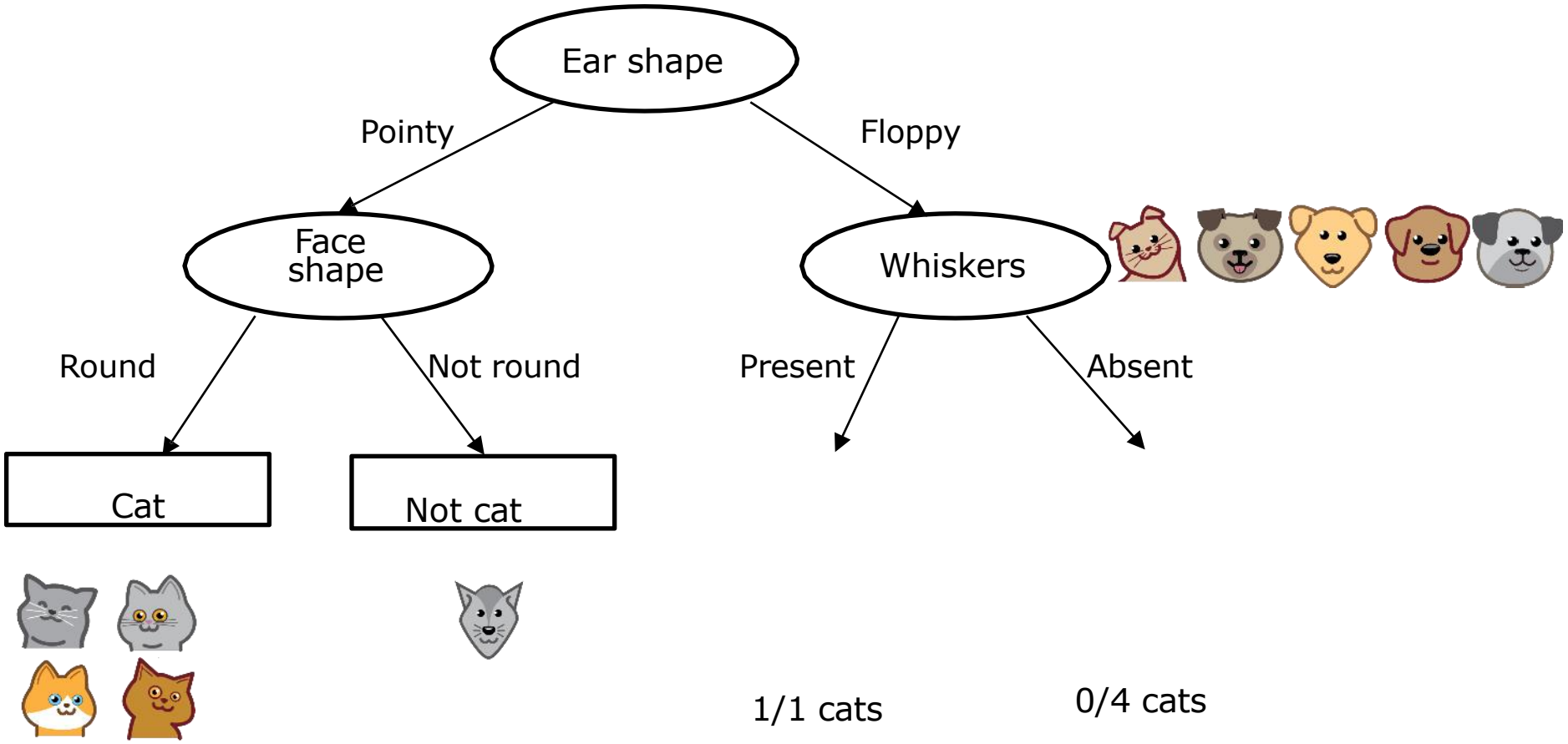
Decision Tree Learning



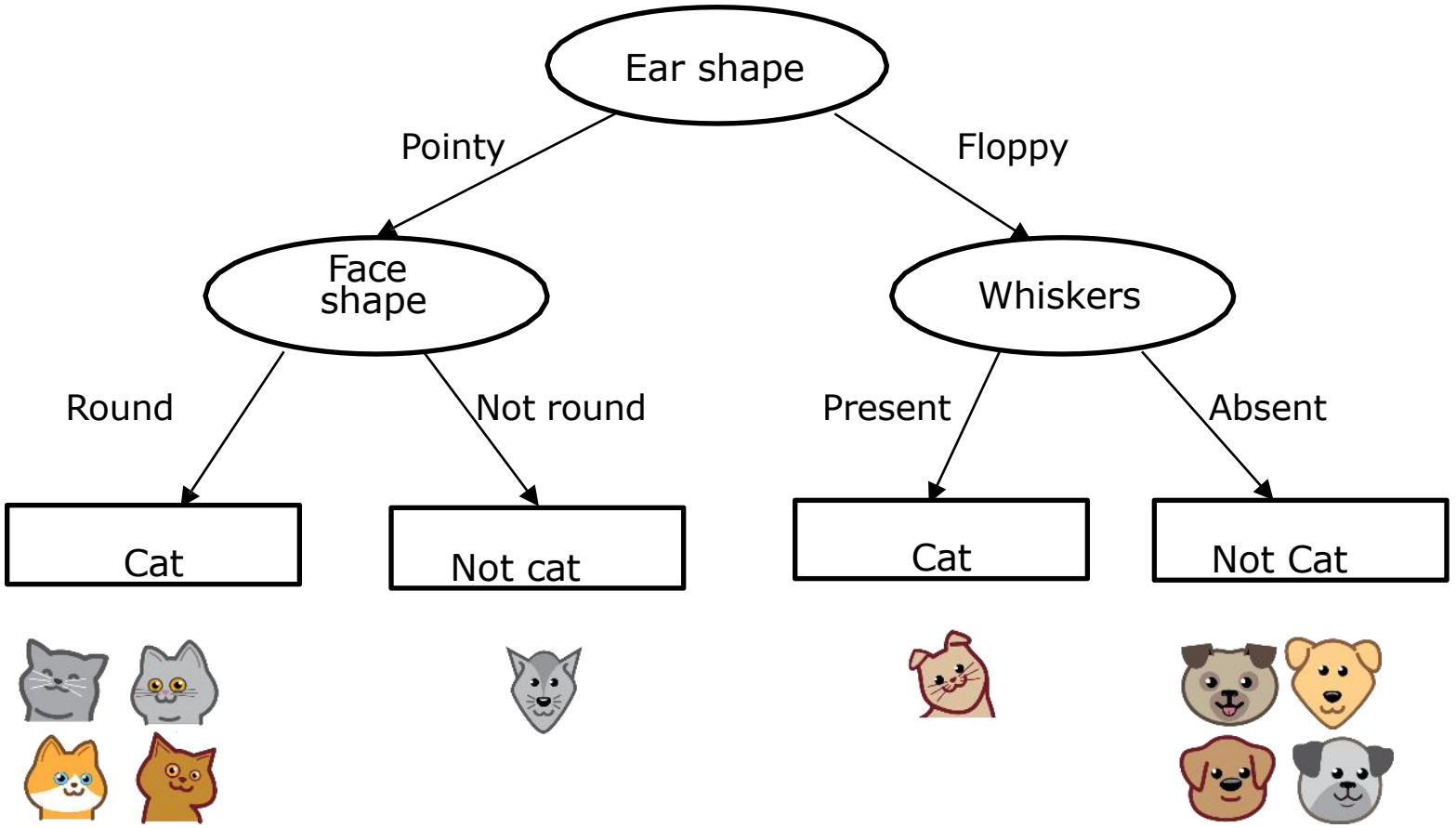
Decision Tree Learning



Decision Tree Learning



Decision Tree Learning



Decision Tree Learning

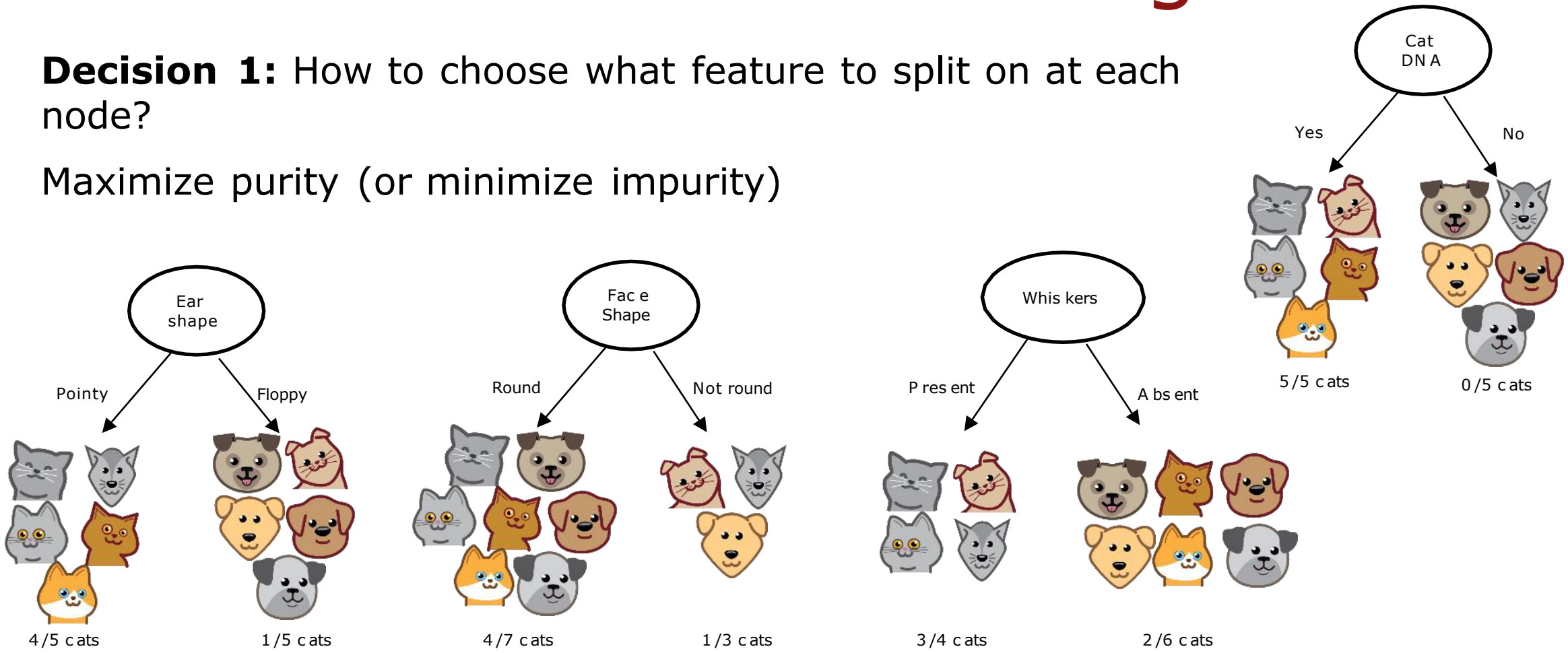
Decision 1: How to choose what feature to split on at each node?

Maximize purity (or minimize impurity)

Decision Tree Learning

Decision 1: How to choose what feature to split on at each node?

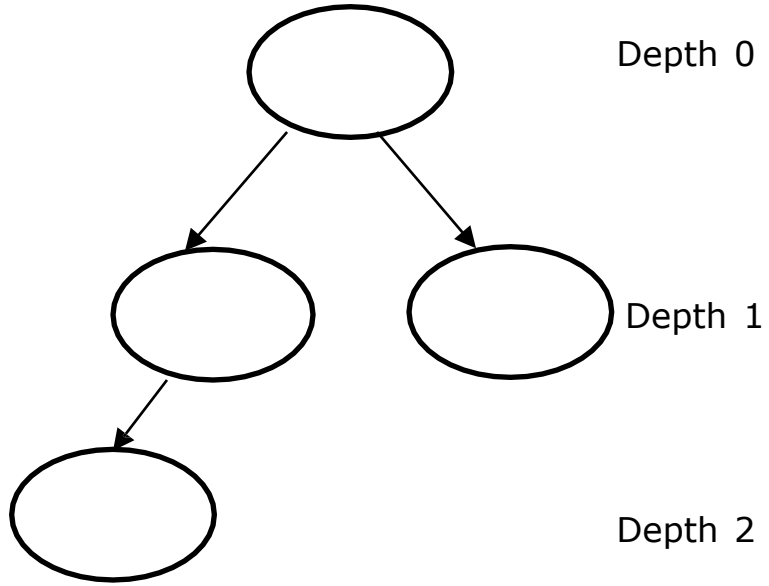
Maximize purity (or minimize impurity)



Decision Tree Learning

Decision 2: When do you stop splitting?

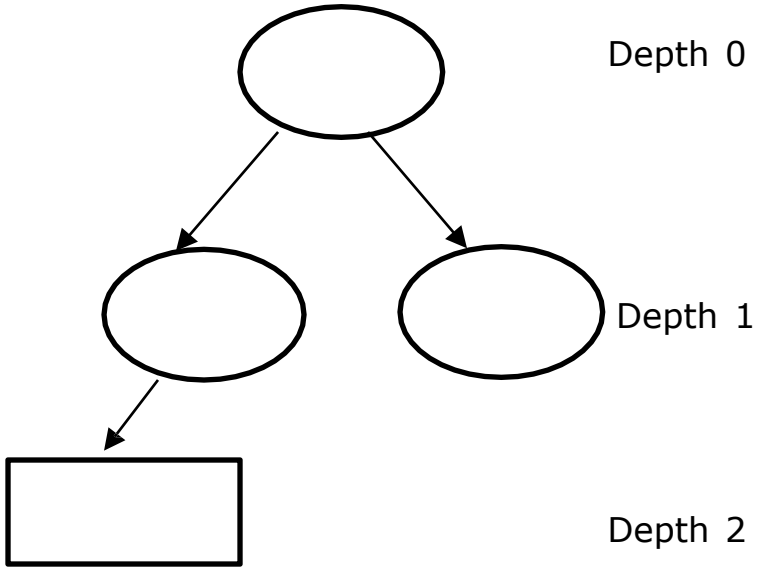
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth



Decision Tree Learning

Decision 2: When do you stop splitting?

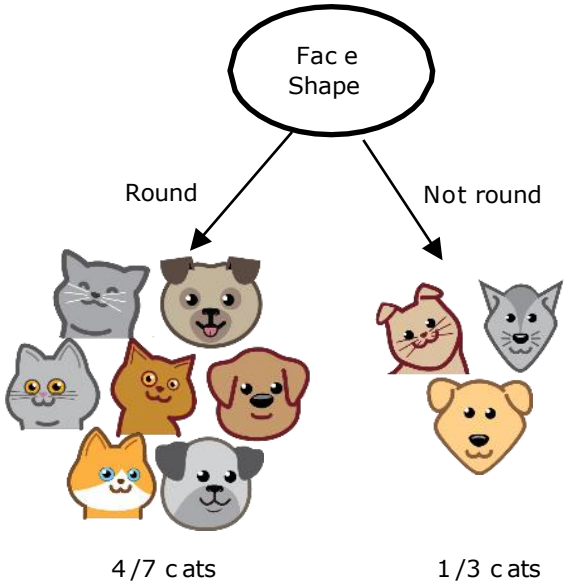
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth



Decision Tree Learning

Decision 2: When do you stop splitting?

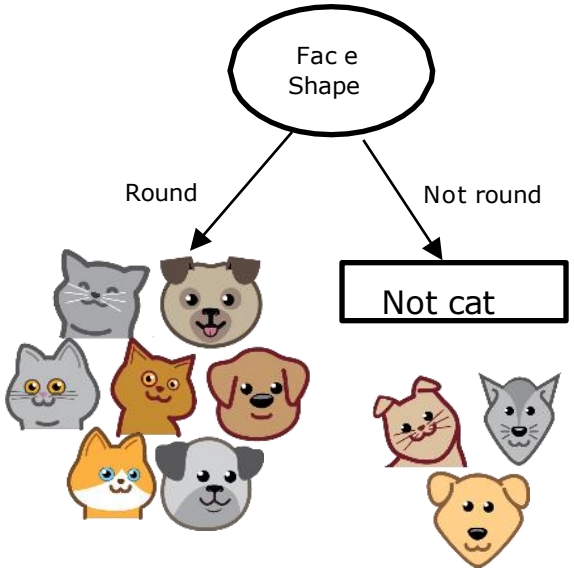
- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



Decision Tree Learning

Decision 2: When do you stop splitting?

- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



Decision Tree Algorithm

Principle

- Basic algorithm (adopted by ID3, C4.5 and CART): a **greedy algorithm**
- Tree is constructed in a top-down recursive divide-and-conquer manner
- Attributes are categorical (if continuous-valued, they are discretized in advance)
- Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node

Iterations

- At start, all the training tuples are at the root
- Tuples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)

Stopping conditions

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
- There are no samples left

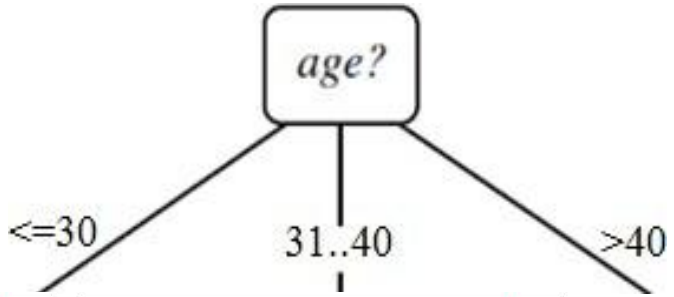
Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute**
that splits this data set.

Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



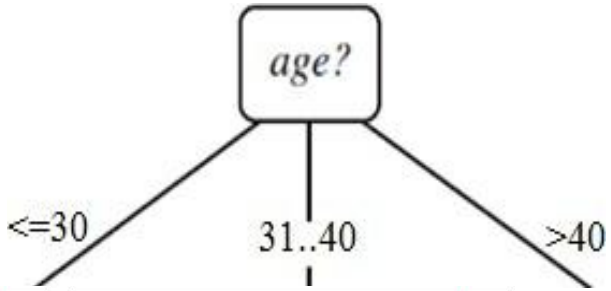
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

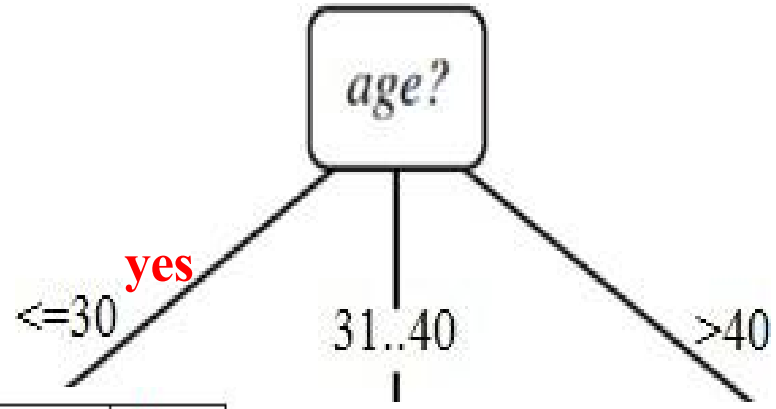
age	income	student	cr	buys
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

yes

all examples are yes

Decision Tree Induction Algorithm: Example

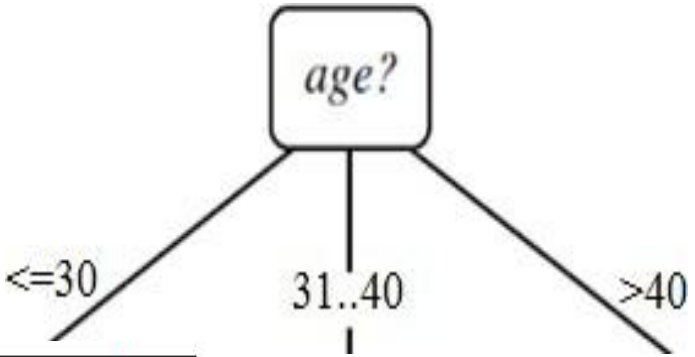


age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute**
that splits this data set.

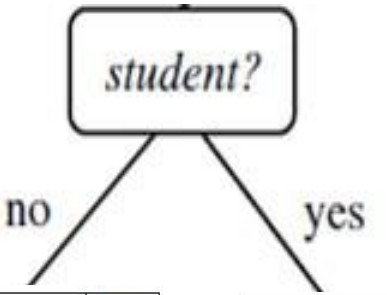
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

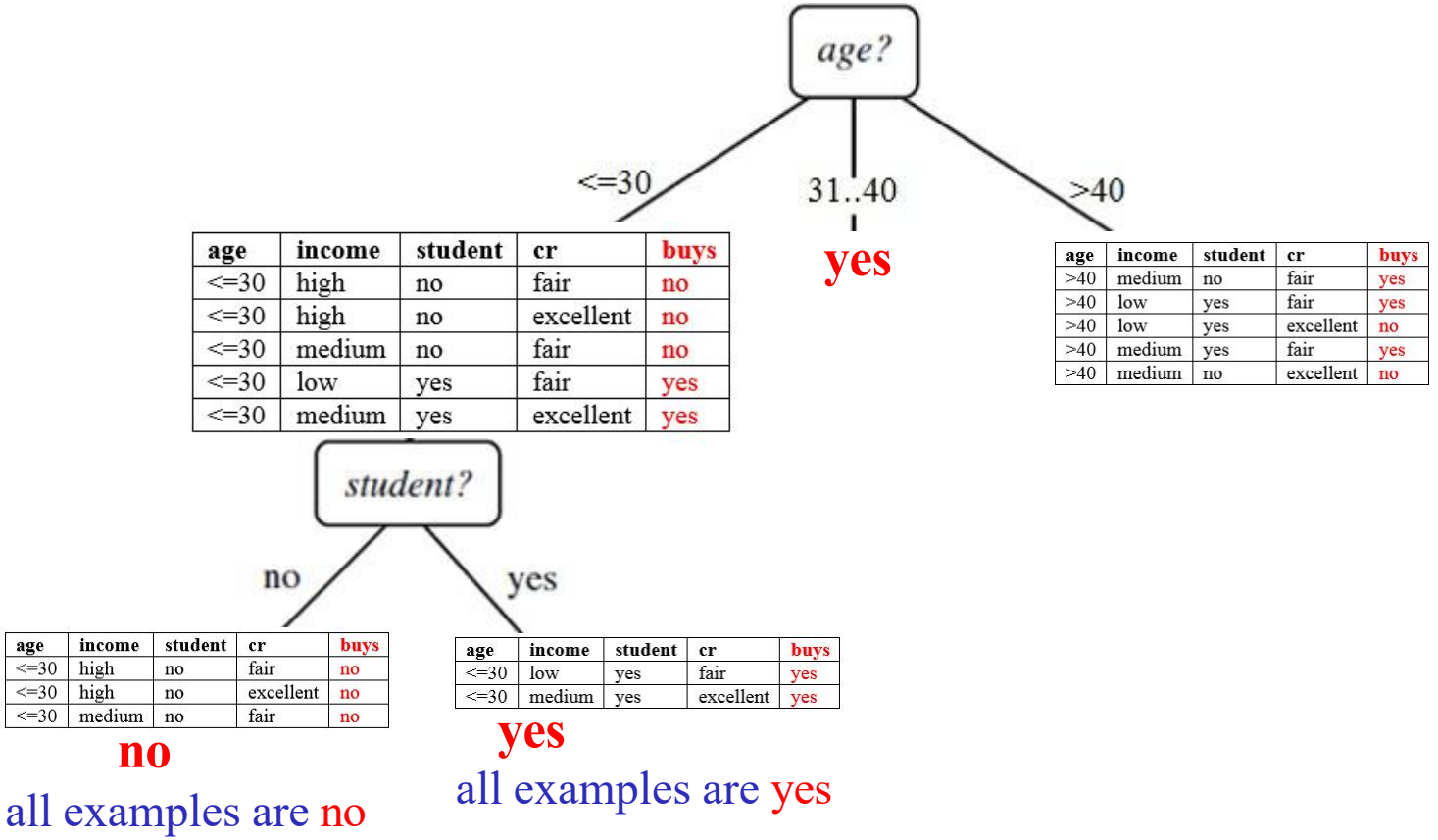
age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no

age	income	student	cr	buys
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

Decision Tree Induction Algorithm: Example



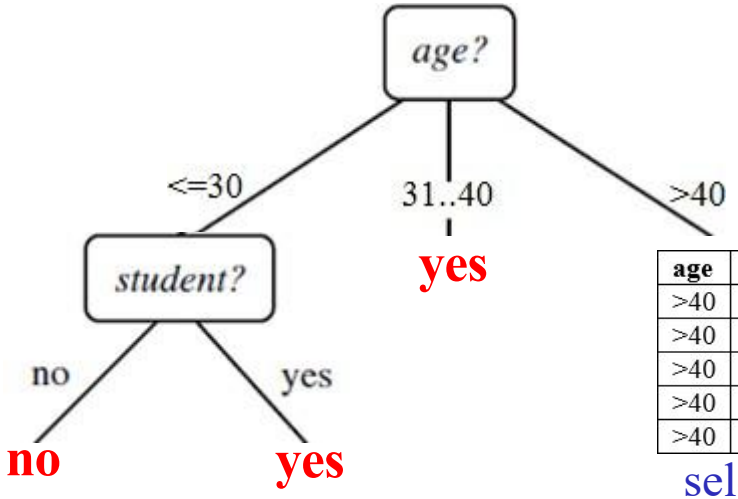
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no

age	income	student	cr	buys
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

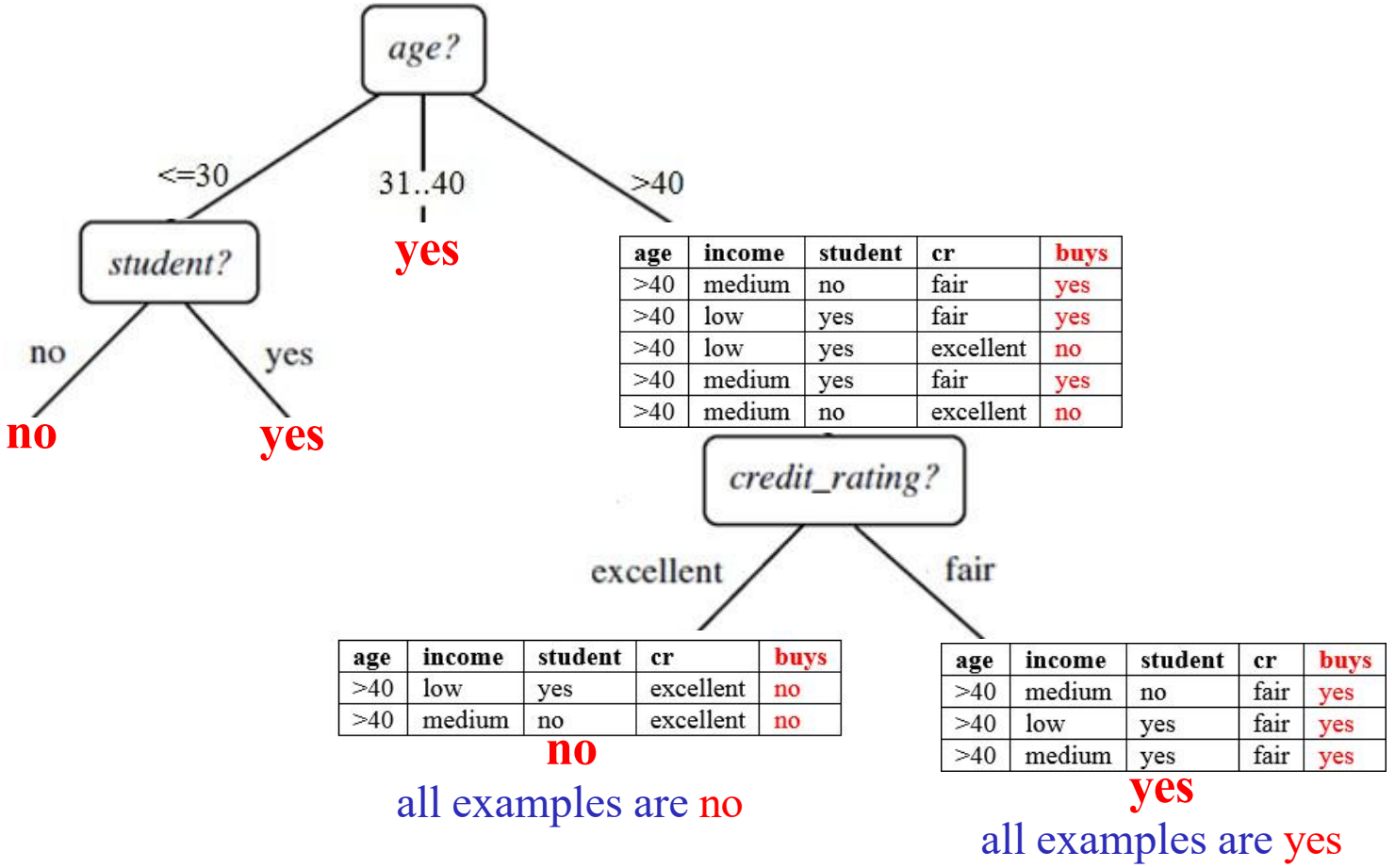
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute**
that splits this data set.

Decision Tree Induction Algorithm: Example

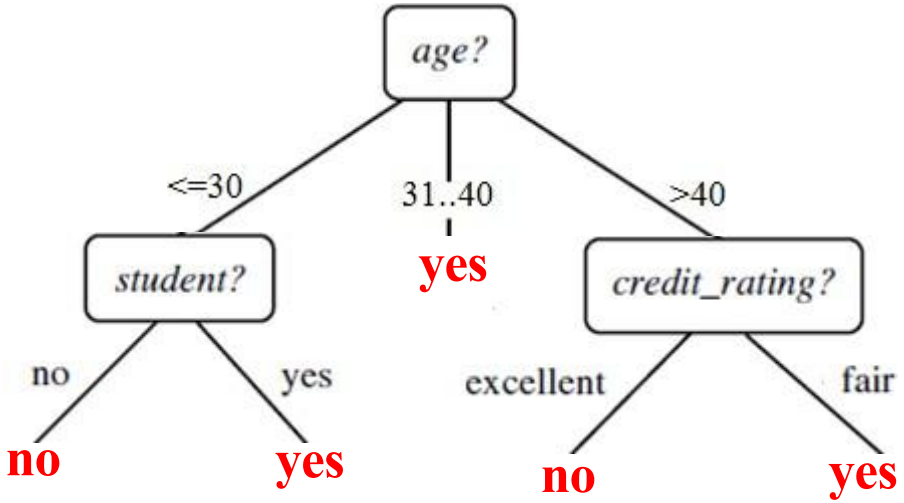


age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	cr	buys
>40	low	yes	excellent	no
>40	medium	no	excellent	no

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	medium	yes	fair	yes

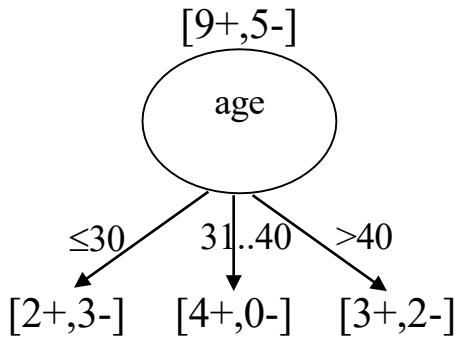
Decision Tree Induction Algorithm: Example

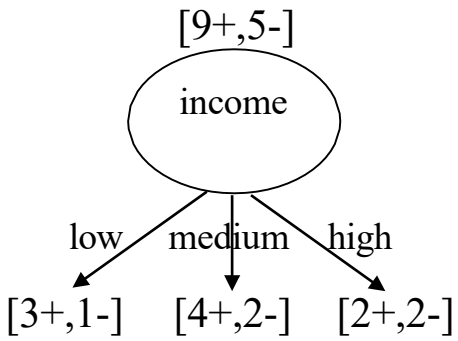


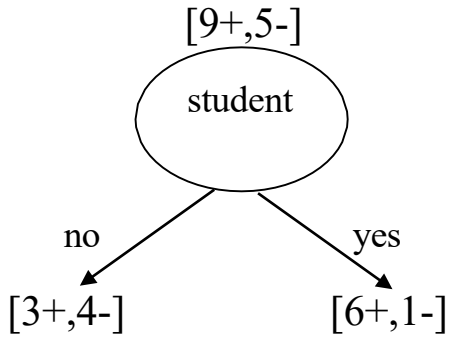
How to Determine the Best Split

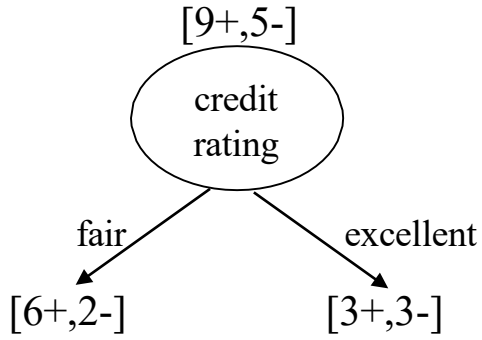
- There are 9 positive examples and 5 negative examples.
- **Which test condition (attribute) is the best?**

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no









How to Determine the Best Split

- Greedy approach:
  - Nodes with **homogeneous (purer)** class distribution are preferred

- Need a **measure of node impurity**:

[5+,5-]	**[9+,1-]**
Non-homogeneous	Homogeneous
High degree of impurity	Low degree of impurity

- **Measures of Node Impurity:**
  - **Entropy, Gini index**

Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain

   Gain = P - M

or equivalently, lowest impurity measure after splitting (M)

Attribute Selection Measures

- An **attribute selection measure** is a heuristic for *selecting the splitting criterion* that **best** separates a given data set D of class-labeled training tuples into individual classes.
  - to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure (i.e., all of the tuples that fall into a given partition would belong to the same class).

- The **attribute having the best score for the measure** is chosen as the splitting attribute for the given tuples.

- Three popular **attribute selection measures:**

  information gain, ***gain ratio***, and ***gini index***.

ENTROPY

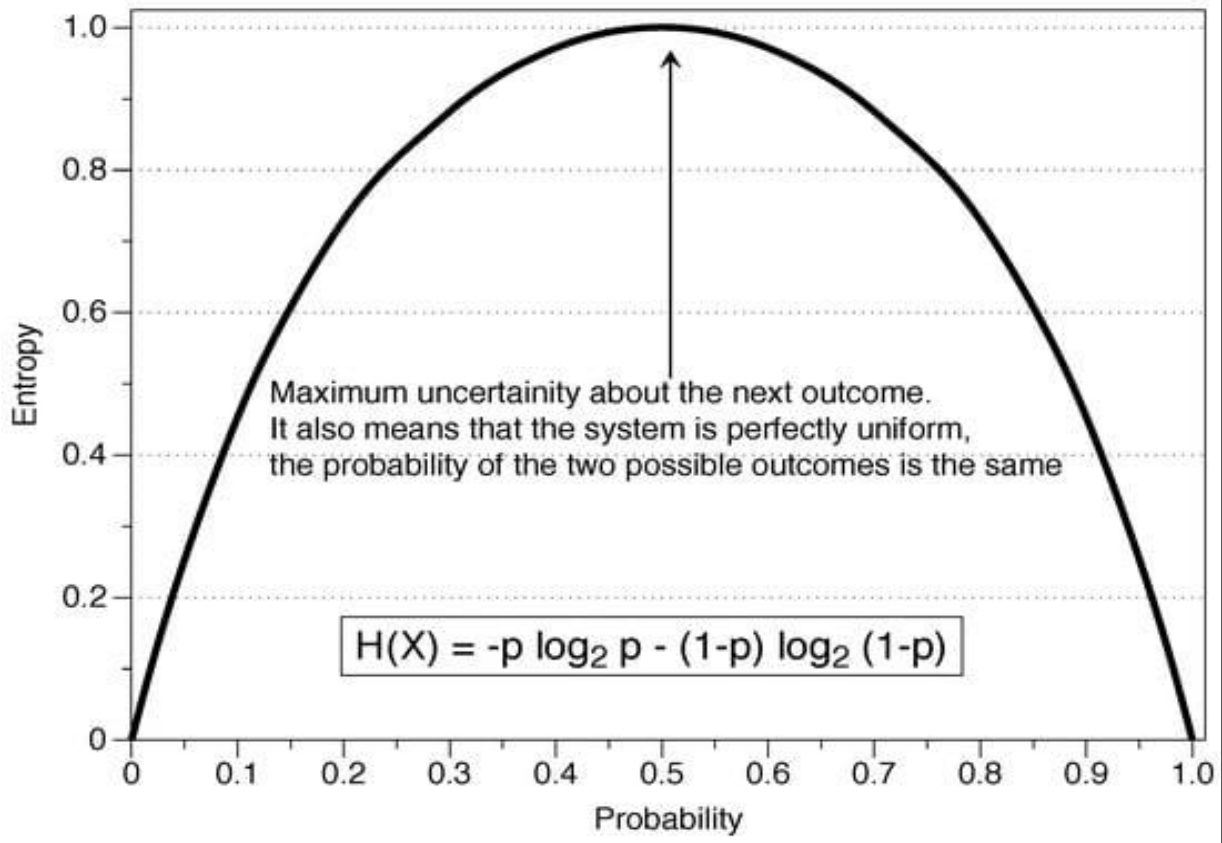
- Entropy is a measure of the amount of uncertainity in the dataset.
- If the data is completely homogenous (i.e. all data samples belong to the same class) then the entropy is 0. Else if the data is divided (50-50%) entropy is 1, otherwise entropy would be in between 0 to 1.
- The goal of a decision tree is to split the data in a way that reduces uncertainty (i.e., entropy) at each step, leading to more "pure" subsets of data.

$$Entropy(D) = H(D) = -\sum_{i=1}^{m} p_i \, log_2(p_i)$$

- $\boldsymbol{m}$ is the number of classes.
- p_i is the nonzero probability that an arbitrary sample/tuple in D belongs to class C_i and is estimated by $\frac{|C_{i,D}|}{|D|}$
- A log function to the base 2 is used, because the information is encoded in bits.
- $Entropy(D)$ or $Info(D)$ is just the average amount of information needed to identify the class label of a tuple in D.

Entropy

- **High Entropy**
  - D is from a uniform like distribution.
  - Flat histogram
  - Values sampled from it are less predictable i.e. more uncertainty or randomness.
- **Low Entropy**
  - D is from a varied (peaks and valleys) distribution.
  - Histogram has many lows and highs
  - Values sampled from it are more predictable.



Assume there are two classes, *P* and *N*

Let the set of examples *D* contain *p* elements of class *P* and *n* elements of class *N*

The amount of information, needed to decide if an arbitrary example in *D* belongs to *P* or *N* is defined as

$$Info(\mathrm{D}) = I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Measure of Impurity: Entropy

- Given a collection S, containing positive and negative examples of some target concept, the ***entropy of S*** relative to this boolean classification is:

$$\mathbf{Entropy(S) = -p_+ log_2 p_+ - p_- log_2 p_-}$$

- **S** is a sample of training examples
- $\mathbf{p_+}$ is the proportion of positive examples
- $\mathbf{p_-}$ is the proportion of negative examples

Measure of Impurity: Entropy

Entropy([12+,5-] =

$-(12/17)\log_2(12/17) - (5/17)\log_2(5/17) = 0.874$

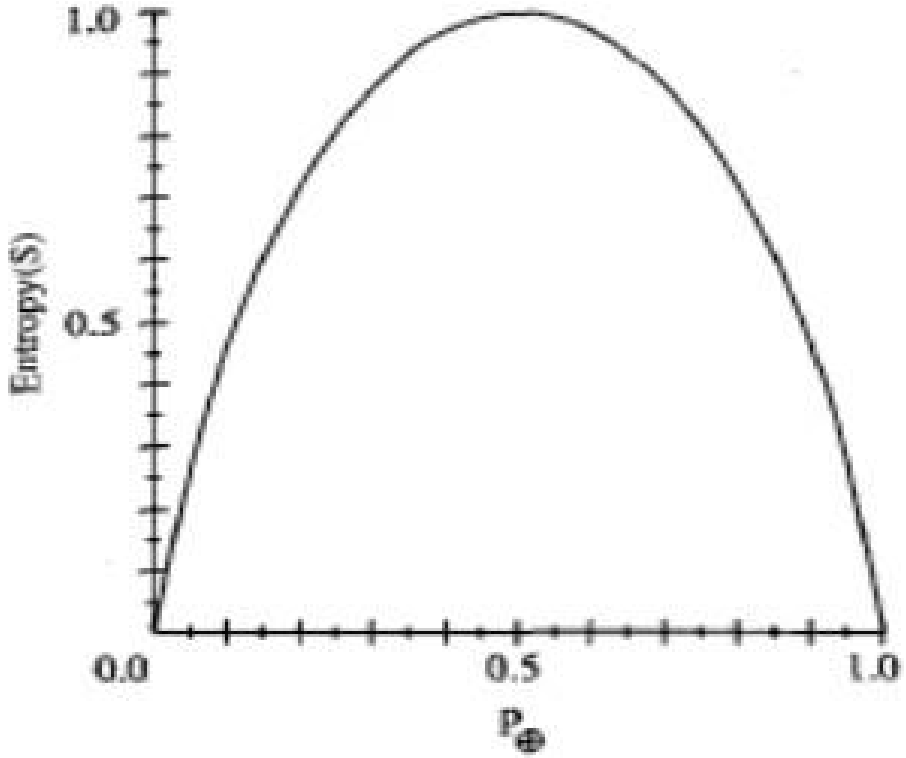
Entropy([8+,8-] =

$-(8/16)\log_2(8/16) - (8/16)\log_2(8/16) = 1.0$

Entropy([8+,0-] =

$-(8/8)\log_2(8/8) - (0/8)\log_2(0/8) = 0.0$

- **It is assumed that $0 \log_2(0)$ is 0**



Entropy – Non-Boolean Target Classification

- If the target attribute can take on **c** different values, then the entropy of **S** relative to this **c-wise** classification is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- ***p_i is the proportion of S belonging to class i.***
- The logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits.
- If the target attribute can take on c possible values, the entropy can be as large as $\log_2 c$.

Entropy – Informaton Theory

- Entropy(S) = ***expected number of bits needed to encode*** class (+ or -) of randomly drawn members of S (under the optimal, shortest length-code)
  - if p+ is 1, the receiver knows the drawn example will be positive, so no message need be sent, and the entropy is zero.
  - if p+ is 0.5, one bit is required to indicate whether the drawn example is positive or negative.
  - if p+ is 0.8, then a collection of messages can be encoded using on average less than 1 bit per message by assigning shorter codes to collections of positive examples and longer codes to less likely negative examples.
- Information theory optimal length code assign $\mathbf{-\log_2 p}$ bits to messages having probability p.
- So the ***expected number of bits to encode*** (+ or -) of random member of S:

$$\mathbf{- p_+ \log_2 p_+ - p_- \log_2 p_-}$$

Attribute Selection Measure: Information Gain

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- **Expected information (entropy)** needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- **Information needed** (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- **Information gained by branching on attribute A**

$$Gain(A) = Info(D) - Info_A(D)$$

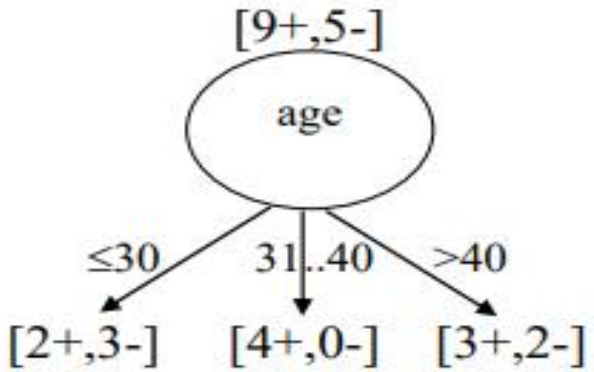
Attribute Selection Measure: Information Gain

- There are 9 positive examples and 5 negative examples.

  D: [9+,5-]

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$\text{Info}(D) = \text{Entropy}(D) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

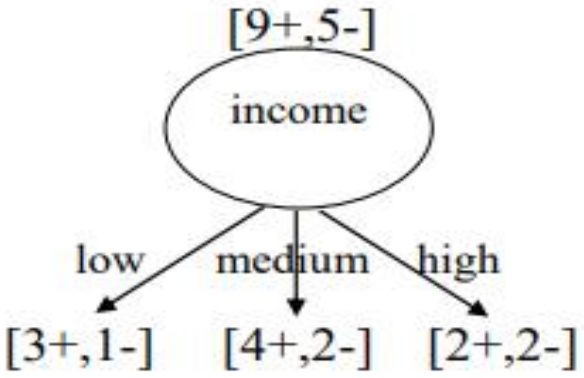


$$\text{Info}_{age}(D) = 5/14 * \text{Info}([2+,3-]) + 4/14 * \text{Info}([4+,0-]) + 5/14 * \text{Info}([3+,2-])$$
$$= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971$$
$$= 0.694$$

$$\text{Gain}(age) = \text{Info}(D) - \text{Info}_{age}(D)$$
$$= 0.940 - 0.694 = 0.246$$

Attribute Selection Measure: Information Gain

$$\text{Info}(D) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

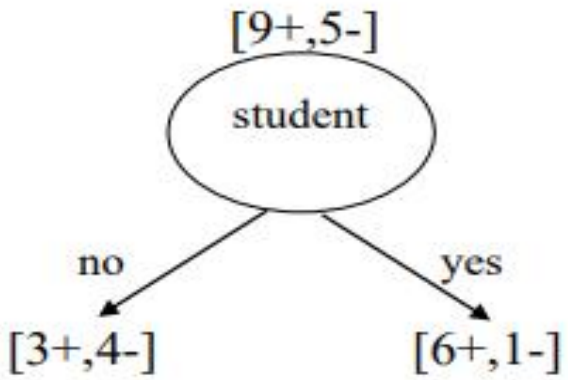


$$\text{Info}_{\text{income}}(D) = 4/14 * \text{Info}([3+,1-]) + 6/14 * \text{Info}([4+,2-]) + 4/14 * \text{Info}([2+,2-])$$
$$= (4/14)*0.811 + (6/14)*0.911 + (4/14)*1.0$$
$$= 0.911$$

$$\text{Gain}(\text{income}) = \text{Info}(D) - \text{Info}_{\text{income}}(D)$$
$$= 0.940 - 0.911 = 0.029$$

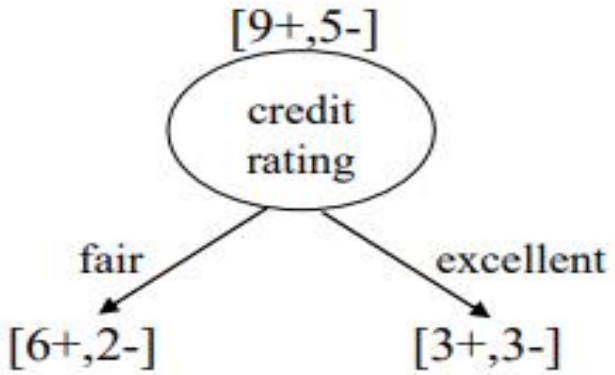
Attribute Selection Measure: Information Gain

$$\text{Info}(D) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$



$$\text{Info}_{\text{student}}(D) = 7/14 * \text{Info}([3+,4-]) + 7/14 * \text{Info}([6+,1-])$$
$$= (7/14)*0.985 + (7/14)*0.592 = 0.789$$

$$\text{Gain}(\text{student}) = \text{Info}(D) - \text{Info}_{\text{student}}(D)$$
$$= 0.940 - 0.789 = 0.151$$



$$\text{Info}_{cr}(D) = 8/14 * \text{Info}([6+,2-]) + 6/14 * \text{Info}([3+,3-])$$
$$= (8/14)*0.811 + (6/14)*1.0 = 0.892$$

$$\text{Gain}(cr) = \text{Info}(D) - \text{Info}_{cr}(D)$$
$$= 0.940 - 0.892 = 0.048$$

Methods for Expressing Test Conditions
Splitting Criterion

- **Depends on attribute types**
  - Binary
  - Nominal
  - Ordinal
  - Continuous

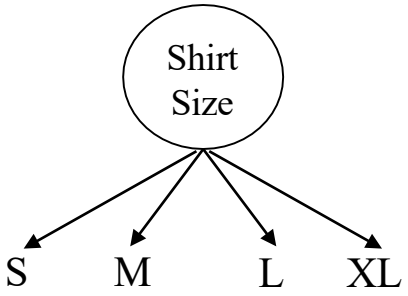
- **Depends on number of ways to split**
  - 2-way split (Binary split)
  - Multi-way split

Test Condition for Ordinal Attributes

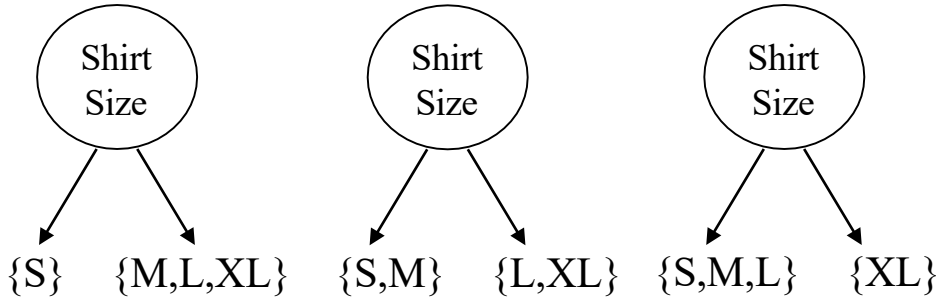
- Multi-way split:
  - Use as many partitions as distinct values of the attribute.

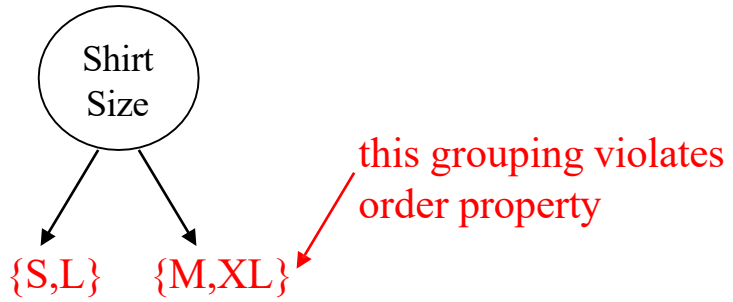
Example:

ShirtSize: {S,M,L,XL}



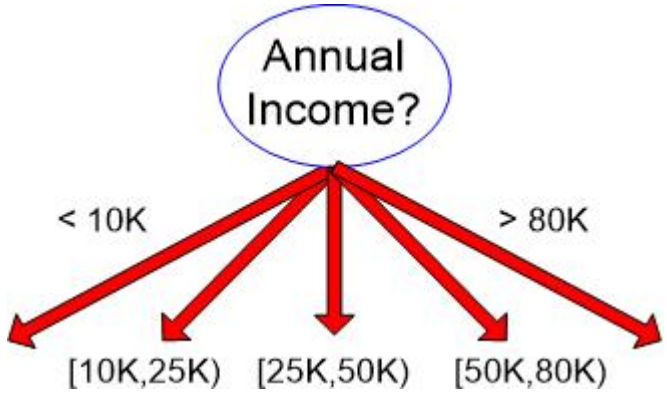
- Binary split:
  - Divide attribute values into two non-empty subsets.
  - Preserve order property among attribute values





Test Condition for Continuous Attributes

- Multi-way split:
  - Discretize the continuous attribute.
  - Discretization to form an ordinal categorical attribute



- Binary split:
  - Divide attribute values into two non-empty subsets from a cut-off point.
  - Binary Decision: (A≤v) or (A> v)
    - consider all possible splits and finds the best cut

Computing Information Gain for Continuous Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the ***best split point*** for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A $\leq$ split-point, and D2 is the set of tuples in D satisfying A > split-point

EXAMPLE

Training Data Set: ***buys_computer***

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

ID3 Algorithm

- During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3.
- ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (i.e.,repeatedly) dichotomizes (i.e., divides) features into two or more groups at each step.
- It is a classification algorithm that follows a top-down greedy approach by selecting a best features that yields maximum **Information Gain(IG)** or **Minimum Entropy(H)**.
- The top-down approach means start building the tree from the top and the greedy approach means that at each iteration, select the best feature at the present moment to create a node.

ID3 ALGORITHM

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N.
```

ID3 Example

Using ID3 algorithm to build a Decision Tree to predict the weather

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

ID3 Example cont…

Step 1: Entropy of dataset is:

$H(S) = - p(Yes) * \log_2(p(Yes)) - p(No) * \log_2(p(No))$

$= - (9/14) * \log_2(9/14) - (5/14) * \log_2(5/14)$

$= - (-0.41) - (-0.53) = 0.94$

Step 2: Calculate entropy for all its categorical values and information gain for the features.

First feature – Outlook

Categorical values - sunny, overcast and rain

H(Outlook=sunny) = -(2/5)*log(2/5)-(3/5)*log(3/5) =0.971

H(Outlook=rain) = -(3/5)*log(3/5)-(2/5)*log(2/5) =0.971

H(Outlook=overcast) = -(4/4)*log(4/4)-0 = 0

Average Entropy Information for Outlook-

I(Outlook) = p(sunny) * H(Outlook=sunny) + p(rain) * H(Outlook=rain) + p(overcast) * H(Outlook=overcast) = (5/14)*0.971 + (5/14)*0.971 + (4/14)*0 = 0.693

Information Gain(S, Outlook) = H(S) - I(Outlook) = 0.94 - 0.693 = 0.247

ID3 Example cont…

Step 2: Calculate entropy for all its categorical values and information gain for the features.

Second feature – Temperature

Categorical values - hot, mild, cool

H(Temperature=hot) = -(2/4)*log(2/4)-(2/4)*log(2/4) = 1

H(Temperature=cool) = -(3/4)*log(3/4)-(1/4)*log(1/4) = 0.811

H(Temperature=mild) = -(4/6)*log(4/6)-(2/6)*log(2/6) = 0.9179

Average Entropy Information for Temperature -

I(Temperature) = p(hot)*H(Temperature=hot) + p(mild)*H(Temperature=mild) + p(cool)*H(Temperature=cool) = (4/14)*1 + (6/14)*0.9179 + (4/14)*0.811 = 0.9108

Information Gain(S, Temperature) = H(S) - I(Temperature) = 0.94 - 0.9108 = 0.0292

ID3 Example cont…

Step 2: Calculate entropy for all its categorical values and information gain for the features.

Third feature - Humidity

Categorical values - high, normal

H(Humidity=high) = -(3/7)*log(3/7)-(4/7)*log(4/7) = 0.983

H(Humidity=normal) = -(6/7)*log(6/7)-(1/7)*log(1/7) = 0.591

Average Entropy Information for Humidity -

I(Humidity) = p(high)*H(Humidity=high) + p(normal)*H(Humidity=normal)

= (7/14)*0.983 + (7/14)*0.591 = 0.787

Information Gain (S, Humidity) = H(S) - I(Humidity) = 0.94 - 0.787 = 0.153

ID3 Example cont…

Step 2: Calculate entropy for all its categorical values and information gain for the features.

Fourth feature – Wind

Categorical values - weak, strong

H(Wind=weak) = -(6/8)*log(6/8)-(2/8)*log(2/8) = 0.811

H(Wind=strong) = -(3/6)*log(3/6)-(3/6)*log(3/6) = 1

Average Entropy Information for Wind -

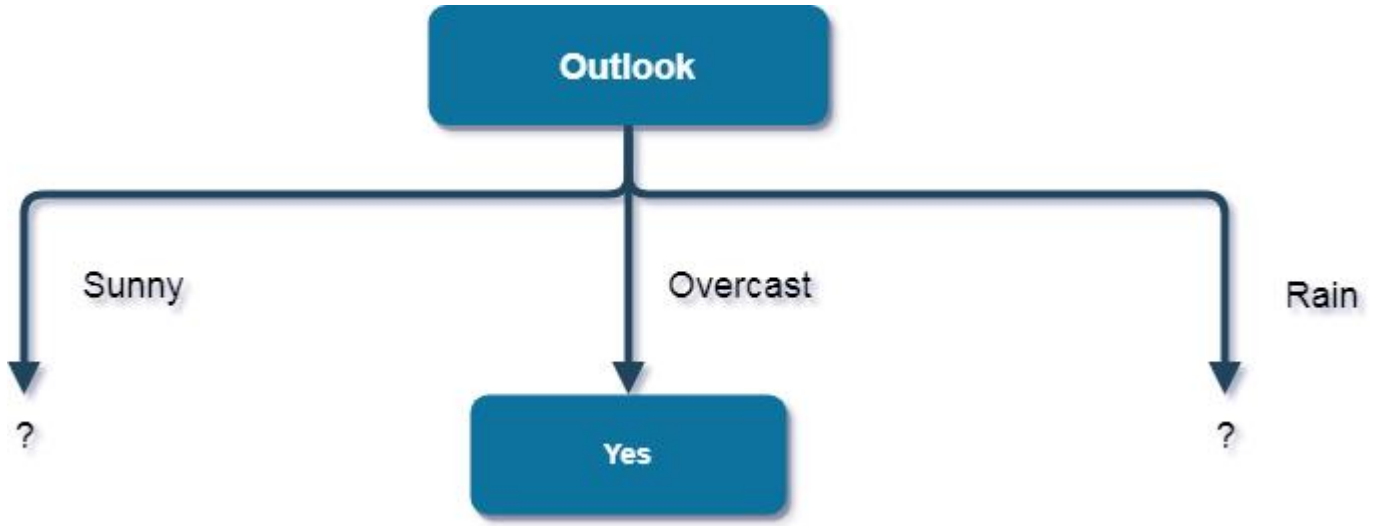
I(Wind) = p(weak)*H(Wind=weak) + p(strong)*H(Wind=strong)

= (8/14)*0.811 + (6/14)*1 = 0.892

Information Gain (S, Wind) = H(S) - I(Wind) = 0.94 - 0.892 = 0.048

ID3 Example cont…

Step 3: Find the feature with maximum information gain.
Here, the feature with maximum information gain is Outlook. So, the decision tree built so far is as follows.
Here, when Outlook = overcast, it is of pure class(Yes).
Now, the same procedure to be repeated for the data with rows consist of Outlook value as Sunny and then for Outlook value as Rain.



ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained
Now, finding the best feature for splitting the data with Outlook=Sunny values
Entropy of Sunny is -
$H(S) = - p(yes) * \log_2(p(yes)) - p(no) * \log_2(p(no))$
$= - (2/5) * \log_2(2/5) - (3/5) * \log_2(3/5)$
$= 0.971$
First feature – Temperature with categorical values - hot, mild, cool
H(Sunny, Temperature=hot) = -0-(2/2)*log(2/2) = 0
H(Sunny, Temperature=cool) = -(1)*log(1)- 0 = 0
H(Sunny, Temperature=mild) = -(1/2)*log(1/2)-(1/2)*log(1/2) = 1

Average Entropy Information for Temperature -
I(Sunny, Temperature) = p(Sunny, hot)*H(Sunny, Temperature=hot) + p(Sunny, mild)*H(Sunny, Temperature=mild) + p(Sunny, cool)*H(Sunny, Temperature=cool) = (2/5)*0 + (1/5)*0 + (2/5)*1 = 0.4

Information Gain (Sunny, Temperature) = H(Sunny) - I(Sunny, Temperature) = 0.971 - 0.4
= 0.571

ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained

Second feature - Humidity with categorical values - high, normal

H(Sunny, Humidity=high) = - 0 - (3/3)*log(3/3) = 0
H(Sunny, Humidity=normal) = -(2/2)*log(2/2)-0 = 0

Average Entropy Information for Humidity -
I(Sunny, Humidity) = p(Sunny, high)*H(Sunny, Humidity=high) + p(Sunny, normal)*H(Sunny, Humidity=normal) = (3/5)*0 + (2/5)*0 = 0

Information Gain (Sunny, Humidity)= H(Sunny) - I(Sunny, Humidity) = 0.971 - 0 = 0.971

ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained

Third feature - Wind with categorical values - weak, strong

H(Sunny, Wind=weak) = -(1/3)*log(1/3)-(2/3)*log(2/3) = 0.918
H(Sunny, Wind=strong) = -(1/2)*log(1/2)-(1/2)*log(1/2) = 1

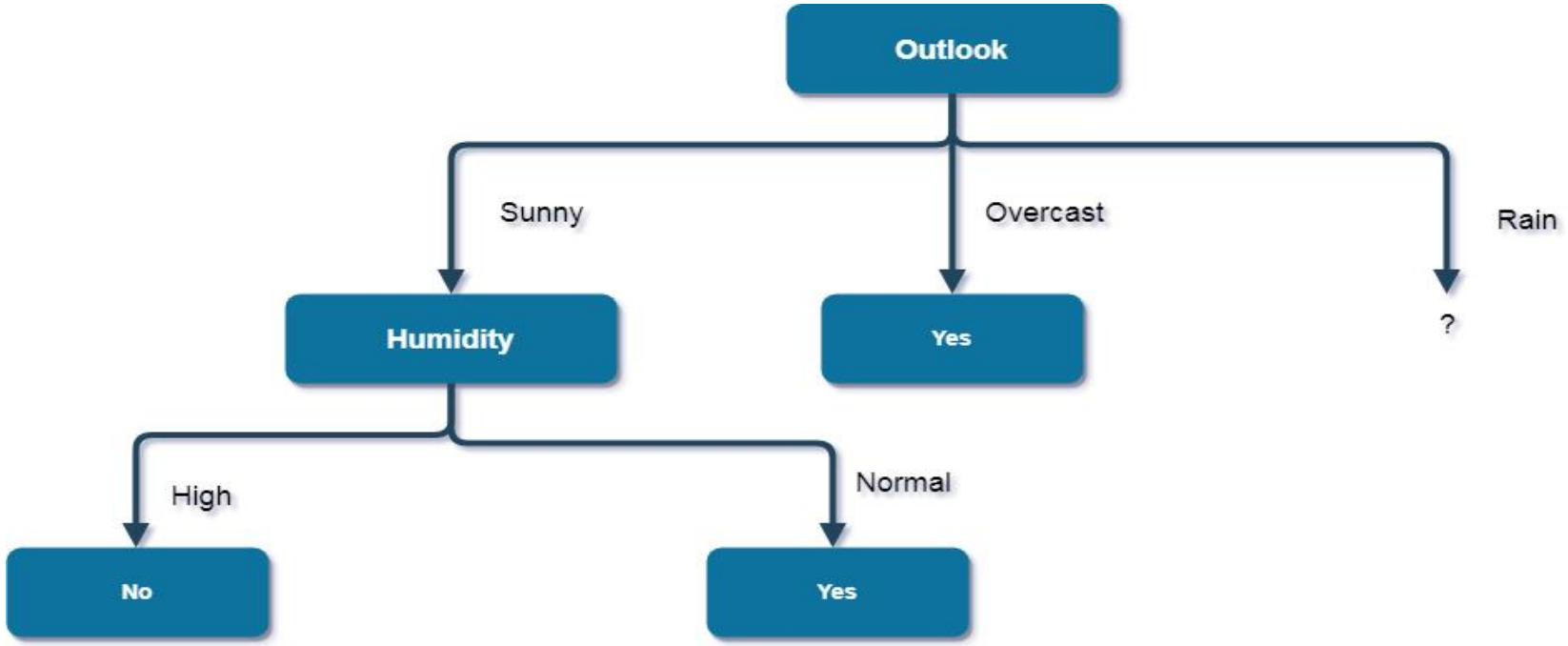
Average Entropy Information for Wind -

I(Sunny, Wind) = p(Sunny, weak)*H(Sunny, Wind=weak) + p(Sunny, strong)*H(Sunny, Wind=strong) = (3/5)*0.918 + (2/5)*1 = 0.9508

Information Gain (Sunny, Wind) = H(Sunny) - I(Sunny, Wind) = 0.971 - 0.9508 = 0.0202

ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained.
The feature with maximum information gain is Humidity. So, the decision tree built so far is as follows. Here, when Outlook = Sunny and Humidity = High, it is a pure class of category "no". And When Outlook = Sunny and Humidity = Normal, it is again a pure class of category "yes". Therefore, we don't need to do further calculations.



ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained.
Now, finding the best feature for splitting the data with Outlook=Rain values
Entropy of Rain is -
H(S) = - p(yes) * log2(p(yes)) - p(no) * log2(p(no))
= - (3/5) * log(3/5) - (2/5) * log(2/5)
= 0.971

First feature – Temperature with categorical values - mild, cool
H(Rain, Temperature=cool) = -(1/2)*log(1/2)- (1/2)*log(1/2) = 1
H(Rain, Temperature=mild) = -(2/3)*log(2/3)-(1/3)*log(1/3) = 0.918

Average Entropy Information for Temperature -
I(Rain, Temperature) = p(Rain, mild)*H(Rain, Temperature=mild) + p(Rain, cool)*H(Rain, Temperature=cool) = (2/5)*1 + (3/5)*0.918 = 0.9508

Information Gain (Rain, Temperature) = H(Rain) - I(Rain, Temperature) = 0.971 - 0.9508
= 0.0202

ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained.

Second feature - Wind with categorical values - weak, strong
H(Wind=weak) = -(3/3)*log(3/3)-0 = 0
H(Wind=strong) = 0-(2/2)*log(2/2) = 0

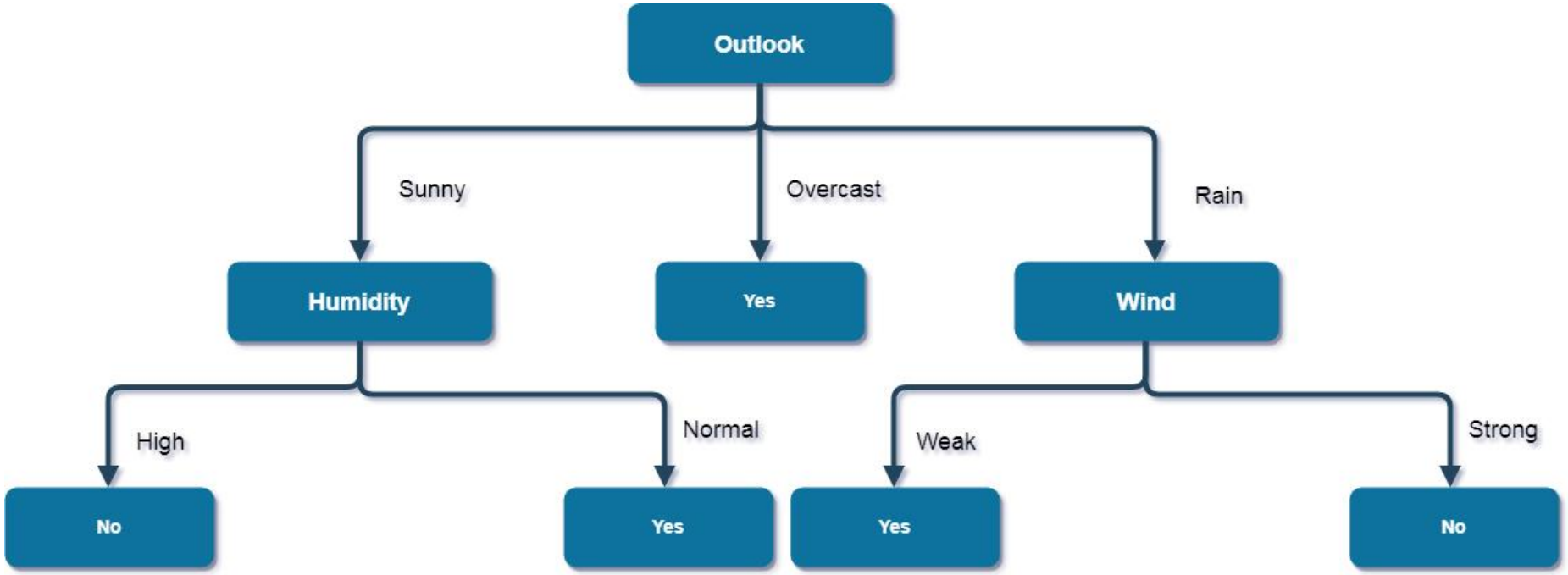
Average Entropy Information for Wind -
I(Wind) = p(Rain, weak)*H(Rain, Wind=weak) + p(Rain, strong)*H(Rain, Wind=strong)
= (3/5)*0 + (2/5)*0 = 0

Information Gain (Rain, Wind)= H(Rain) - I(Rain, Wind) = 0.971 – 0 = 0.971

ID3 Example cont…

Step 4: The same process to be repeated until the tree is obtained.

Here, the feature with maximum information gain is Wind. So, the decision tree built so far is as follows. Here, when Outlook = Rain and Wind = Strong, it is a pure class of category "no". When Outlook = Rain and Wind = Weak, it is again a pure class of category "yes". This is final desired tree for the given dataset.



Other Attribute Selection Measures

Gain Ratio

- The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.
- For example, consider an attribute that acts as a unique identifier, such as product_ID. A split on product_ID would result in a large number of partitions (as many as there are values), each one containing just one tuple. Because each partition is pure, the information required to classify data set D based on this partitioning would be Infoproduct_ID(D) = 0. Therefore the information gained by partitioning on this attribute is maximal. Clearly, such a partitioning is useless for classification.
- C4.5, a successor of ID3 uses an extension to information gain known as *gain ratio* which attempts to overcome this bias. It applies a kind of normalization to information gain using a "split information" value.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}. \qquad SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right).$$

Other Attribute Selection Measures

Gini Impurity

- The Gini impurity (or Gini in short) is used in CART. Using the notation previously described, the Gini measures the impurity of D, a data partition or a set of training tuples, as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2, \tag{6.8}$$

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. The sum is computed over m classes.

Model Overfitting

Training error: Train a model on the training dataset, then test the model on the same training set. The error rate is called "training error," which evaluates how well the model fits the training data.

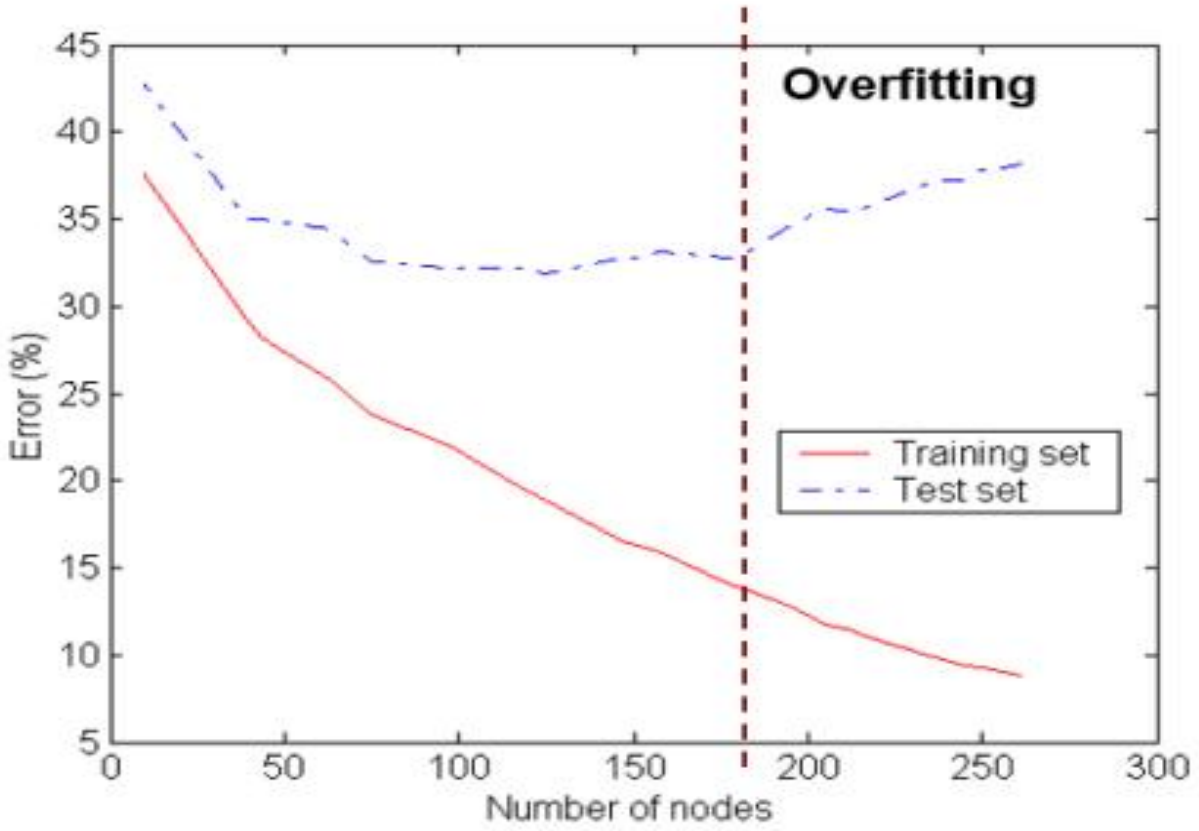
Test error: Test the model on a test dataset that is different from the training set. The error rate is called "test error," which evaluates how well the model generalizes to unseen data.

Overfitting means a model fits the training data very well but generalizes to unseen data poorly.

How do I know if my model is overfitting?

- Your model is overfitting if its training error is small (fits well with training data) but the test error is large (generalizes poorly to unseen data).

Underfitting and Overfitting



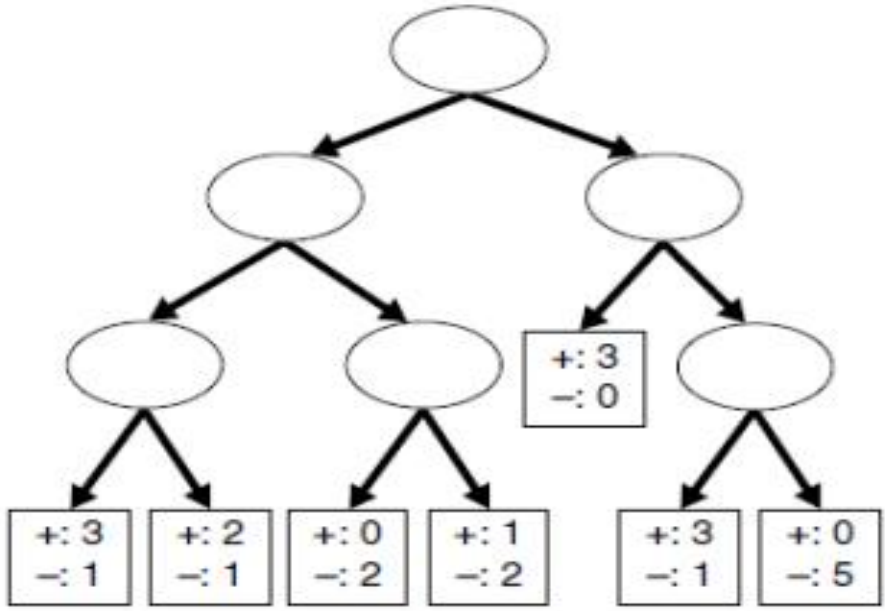
Overfitting due to
- Noise
- Insufficient Examples

Two approaches to **avoid overfitting**
- ***Prepruning***: Halt tree construction early-do not split a node if this would result in the goodness measure falling below a threshold
- ***Postpruning***: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees

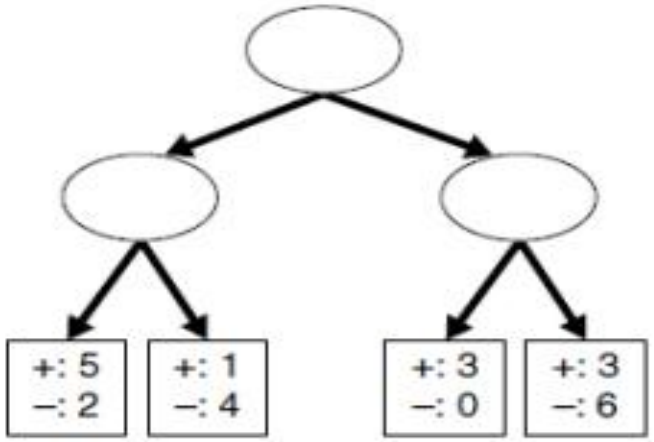
- **Overfitting** means a model fits the training data very well but generalizes unseen data poorly.
- **Overfitting:** complex models are more likely to overfit than simple models
- **Underfitting:** when model is too simple, both training and test errors are large

Occam's Razor

- **Occam's Razor: Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.**
  - For complex models, there is a greater chance that they are fitted accidentally by errors in data.
- **Therefore, one should include model complexity when evaluating a model.**



Decision Tree, T_L



Decision Tree, T_R

Avoid Overfitting in Classification

- The generated tree may overfit the training data -
  - Too many branches, some may reflect anomalies due to noise or outliers.
  - Result is in poor accuracy for unseen samples.
- Two approaches to avoid overfitting -
  - **Prepruning**: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold.
    - Difficult to choose an appropriate threshold.
  - **Postpruning:** Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees.
    - Use a set of data different from the training data to decide which is the "best pruned tree".

Extracting Classification Rules from Decision Trees

- Represent the knowledge in the form of IF-THEN rules.
- One rule is created for each path from the root to a leaf.
- Each attribute-value pair along a path forms a conjunction.
- The leaf node holds the class prediction.
- Rules are easier for humans to understand.

Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = "31…40" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Naïve Bayes Classification

Bayesian Classification

- **Bayesian classifiers** are ***statistical classifiers***.
  - They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on **Bayes Theorem.**
- A simple Bayesian classifier known as the **Naïve Bayesian Classifier** to be *comparable in performance* with decision tree and selected neural network classifiers.
  - Bayesian classifiers exhibits high accuracy and speed when applied to large databases.
- Even when **Bayesian methods** are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes Theorem

- P(A) is **prior probability** (**unconditional probability**) of event A.
- P(A|B) is **posterior probability** (**conditional probability**) of event A given that event B holds.
- P(A,B) is the **joint probability** of two events A and B.
  - The (unconditional) probability of the events A and B occurring together.
  - P(A,B) = P(B,A)

Bayes Theorem

$P(A|B) = P(A,B) / P(B)$ ➔ $P(A,B) = P(A|B)*P(B)$

$P(B|A) = P(B,A) / P(A)$ ➔ $P(B,A) = P(B|A)*P(A)$

Since $P(A,B) = P(B,A)$, we have $P(A|B)*P(B) = P(B|A)*P(A)$

Thus, we have **Bayes Theorem**

$$\mathbf{P(A|B) = P(B|A)*P(A) / P(B)}$$

$$\mathbf{P(B|A) = P(A|B)*P(B) / P(A)}$$

Bayes Theorem - Example

Bayes Theorem

P(A|B) = P(B|A)*P(A) / P(B)

P(B|A) = P(A|B)*P(B) / P(A

Sample Space for events A and B

A holds	T	T	F	F	T	F	T
B holds	T	F	T	F	T	F	F

P(A) = 4/7 P(B) = 3/7 P(A,B) = P(B,A) = 2/7

P(B|A) = 2/4 P(A|B) = 2/3

Is Bayes Theorem correct?

P(B|A) = P(A|B)*P(B) / P(A) = (2/3 * 3/7) / 4/7 = 2/4 ➔ CORRECT

P(A|B) = P(B|A)*P(A) / P(B) = (2/4 * 4/7) / 3/7 = 2/3 ➔ CORRECT

Bayes Theorem - Example

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time P(S|M) = 0.5
  - Prior probability of any patient having meningitis is 1/50,000 P(M) = 1/50,000
  - Prior probability of any patient having stiff neck is 1/20 P(S) = 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis? **P(M|S) ?**

$$\mathbf{P(M|S) = \frac{P(S|M)\,P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002}$$

Independence of Events

- The events A and B are **INDEPENDENT** if and only if **P(A,B) = P(A)*P(B)**

Example: Bit strings of length 3 is {000,001,010,011,100,101,110,111}

Event A: A randomly generated bit string of length three begins with a 1.

Event B: A randomly generated bit string of length three ends with a 1.

P(A) = 4/8 100,101,110,111 P(B) = 4/8 001,011,101,111

P(A,B) = 2/8 101,111 Are A and B independent? P(A)*P(B) = (4/8) * (4/8) = 16/64 = 2/8 = P(A,B)

➔ A and B are independent.

Event C: A randomly generated bit string of length three contains with two 1s.

P(C) = 3/8 011,101,110

P(A,C) = 2/8 101,110 Are A and C independent?

P(A)*P(C) = (4/8)*(3/8) = 12/64 = 3/16 ≠ 2/8

➔ A and C are NOT independent.

Bayes Theorem for Prediction

- Let **X** be a **data sample**: its class label is unknown.
- Let H be a **hypothesis** that X belongs to class C.
- **Classification** is to determine P(H|**X**), (i.e., **posteriori probability**): the probability that the hypothesis holds given the observed data sample **X.**
- P(H) (**prior probability**): the initial probability of H
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (**likelihood**): the probability of observing the sample **X**, given that the hypothesis H holds.

Bayes Theorem: $$\mathbf{P(H|X) = \frac{P(X|H)\,P(H)}{P(X)}}$$

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes.

Naïve Bayes Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an attribute vector $(x_1, x_2, \ldots, x_n)$
  - Attributes $A_1, A_2, \ldots, A_n$ have values $A_1=x_1, A_2=x_2, \ldots, A_n=x_n$
- Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.
- We are looking the classification of the tuple $(x_1, x_2, \ldots, x_n)$.
- **The classification of this tuple will be the class C_i that maximizes the following conditional probability**.

$$\mathbf{P(C_i \mid x_1, x_2, \ldots, x_n)}$$

Naïve Bayes Classifier

- To compute $P(C_i \mid x_1, x_2, \ldots, x_n)$ is almost impossible for a real data set.
- We use Bayes Theorem to find this conditional probability.

$$P(C_i \mid x_1, x_2, \ldots, x_n) = P(x_1, x_2, \ldots, x_n \mid C_i) * P(C_i) \ / \ P(x_1, x_2, \ldots, x_n)$$

- **Since $P(x_1, x_2, \ldots, x_n)$ is constant for all classes, we only look at the class C_i that maximizes the following formula.**

$$\mathbf{P(x_1, x_2, \ldots, x_n \mid C_i) * P(C_i)}$$

- We should compute $P(x_1, x_2, \ldots, x_n \mid C_i)$ and $P(C_i)$ from the training dataset.

Naïve Bayes Classifier
Computing Probabilities

- To compute $P(C_i)$ from the dataset is easy.

  $P(C_i) = N_{Ci} / N$ where N_{Ci} is the number of tuples belong to class C_i and N is the number of the total tuples in the dataset.

- But, to compute $P(x_1, x_2, \ldots, x_n | C_i)$ from the dataset is NOT easy.
  - In fact, it is almost impossible for a dataset with many attributes.
  - If we have n binary attributes, the number of possible tuples is 2^n.

Naïve Bayes Classifier

Computing Probabilities – Independence Assumption

- In order to compute $P(x_1, x_2, \ldots, x_n \mid C_i)$, we make independence assumption for attributes although this assumption may not be true.

  Independence Assumption: Attributes are conditionally independent (i.e., no dependence relation between attributes)

  $$\mathbf{P(x_1, x_2, \ldots, x_n \mid C_i) = P(x_1 \mid C_i) * P(x_2 \mid C_i) * \ldots * P(x_n \mid C_i)}$$

- If $\mathbf{A_k}$ **is categorical**,

  $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by

  $|C_i|$ (# of tuples of C_i in the dataset)

Naïve Bayes Classifier

Computing Probabilities – continuous-valued attribute

- If $\mathbf{A_k}$ **is a continuous-valued attribute**,

  $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$\mathbf{g(x, \mu, \sigma)} = \frac{\mathbf{1}}{\sqrt{\mathbf{2\pi\sigma^2}}}\, \mathbf{e}^{-\frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\mathbf{x-\mu}}{\sigma}\right)^{\mathbf{2}}}$$

  and $P(x_k|C_i)$ is $\mathbf{g(x_k, \mu_{C_i}, \sigma_{C_i})}$

- Or we can discretize the continuous-valued attribute first.

Naïve Bayes Classifier

Computing Probabilities from Training Dataset

Dataset has 14 tuples.

Two classes:

buyscomputer=yes

buyscomputer=no

P(bc=yes) = 9/14

P(bc=no) = 5/14

age	income	student	creditrating	buyscomputer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier

Computing Probabilities from Training Dataset

P(age=b31|bc=yes)=2/9
P(age=i31|bc=yes)=4/9
P(age=g40|bc=yes)=3/9

P(age=b31|bc=no)=3/5
P(age=i31|bc=no)=0
P(age=g40|bc=no)=2/5

P(inc=high|bc=yes)=2/9
P(inc=med|bc=yes)=4/9
P(inc=low|bc=yes)=3/9

P(inc=high|bc=no)=2/5
P(inc=med|bc=no)=2/5
P(inc=low|bc=no)=1/5

P(std=yes|bc=yes)=6/9
P(std=no|bc=yes)=3/9

P(std=yes|bc=no)=1/5
P(std=no|bc=no)=4/5

P(cr=exc|bc=yes)=3/9
P(cr=fair|bc=yes)=6/9

P(cr=exc|bc=no)=3/5
P(cr=fair|bc=no)=2/5

P(bc=yes) = 9/14
P(bc=no) = 5/14

age	income	student	creditrating	buyscomputer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier
Finding Classification

X: (age <= 30 , income = medium, student = yes, creditrating = fair)

P(X|bc=yes)=P(age=b30|bc=yes)*P(inc=med|bc=yes)*P(std=yes|bc=yes)*P(cr=fair|bc=yes)
= 2/9 * 4/9 * 6/9 * 6/9 = 0.044

P(X|bc=no)=P(age=b30|bc=no)*P(inc=med|bc=no)*P(std=yes|bc=no)*P(cr=fair|bc=no)
= 3/5 * 2/5 * 1/5 * 2/5 = 0.019

P(X|bc=yes)*P(bc=yes) = 0.044 * 9/14 = 0.028

P(X|bc=no)*P(bc=no) = 0.019 * 5/14 = 0.007

➔ Therefore, X belongs to class "buyscomputer = yes"

Confidence of the classification: 0.028/(0.028+0.007) = 0.80 80%

Naïve Bayes Classifier - Example

- Compute all probabilities for Naïve Bayes Classifier.
- Find the classification of the tuple (A=T,B=F)
- What is the confidence of that classification?

A	B	Class
T	F	Yes
T	F	No
T	T	Yes
T	F	Yes
F	T	Yes
F	T	No
F	F	No

Naïve Bayes Classifier - Example

P(yes) = 4/7 P(no) = 3/7

P(A=T|yes) = 3/4 P(A=T|no) = 1/3
P(A=F|yes) = 1/4 P(A=F|no) = 2/3

P(B=T|yes) = 2/4 P(B=T|no) = 1/3
P(B=F|yes) = 2/4 P(B=F|no) = 2/3

P(A=T,B=F|yes) = P(A=T|yes)*P(B=F|yes) = 3/4 * 2/4 = 6/16
P(A=T,B=F|no) = P(A=T|no)*P(B=F|no) = 1/3 * 2/3 = 2/9

P(A=T,B=F|yes)*P(yes) = 6/16 * 4/7 = 24/112 = 0.214
P(A=T,B=F|no)*P(no) = 2/9 * 3/7 = 6/63 = 0.095

Classification is YES

Confidence: 0.214 / (0.214+0.095) = 0.69 69%

A	B	Class
T	F	Yes
T	F	No
T	T	Yes
T	F	Yes
F	T	Yes
F	T	No
F	F	No

Naive Bayes Example

- Train a Naive Bayes Classifier for following golf training dataset (On left) and then use that classifier to classify given test data (on right) -

Day	Outlook	Temperature	Humidity	Wind	Play Golf
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Outlook	Temperature	Humidity	Windy
Sunny	Cool	High	True
Overcast	Mild	Normal	False

Naïve Bayes Classifier: Comments

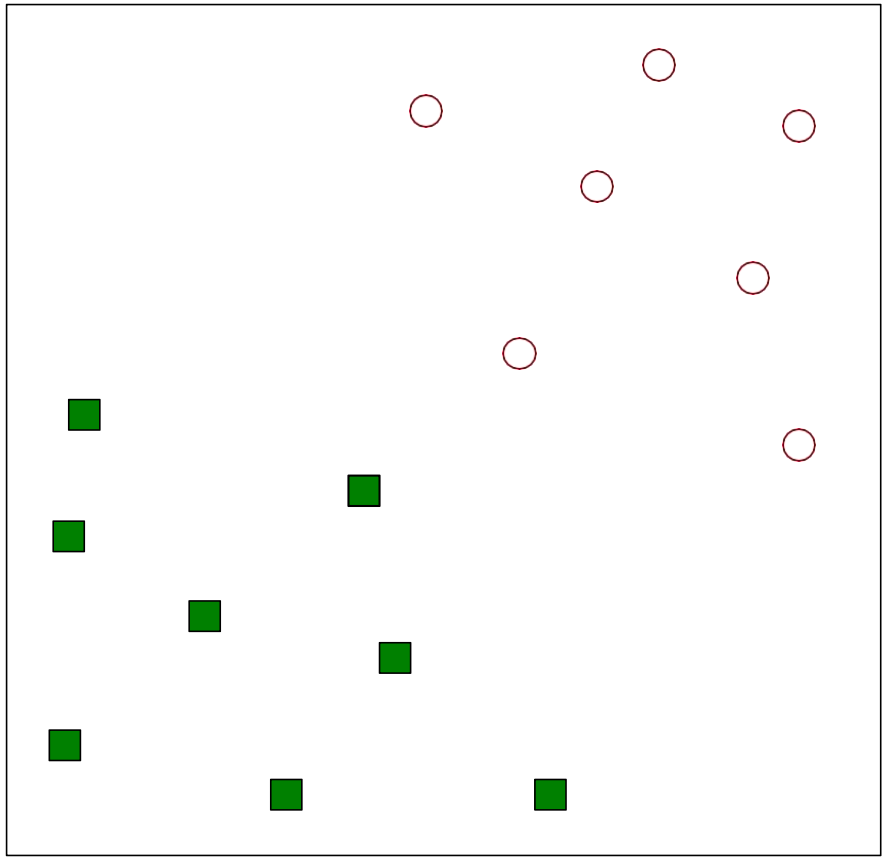
- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks

SUPPORT VECTOR MACHINES

Linear classifiers

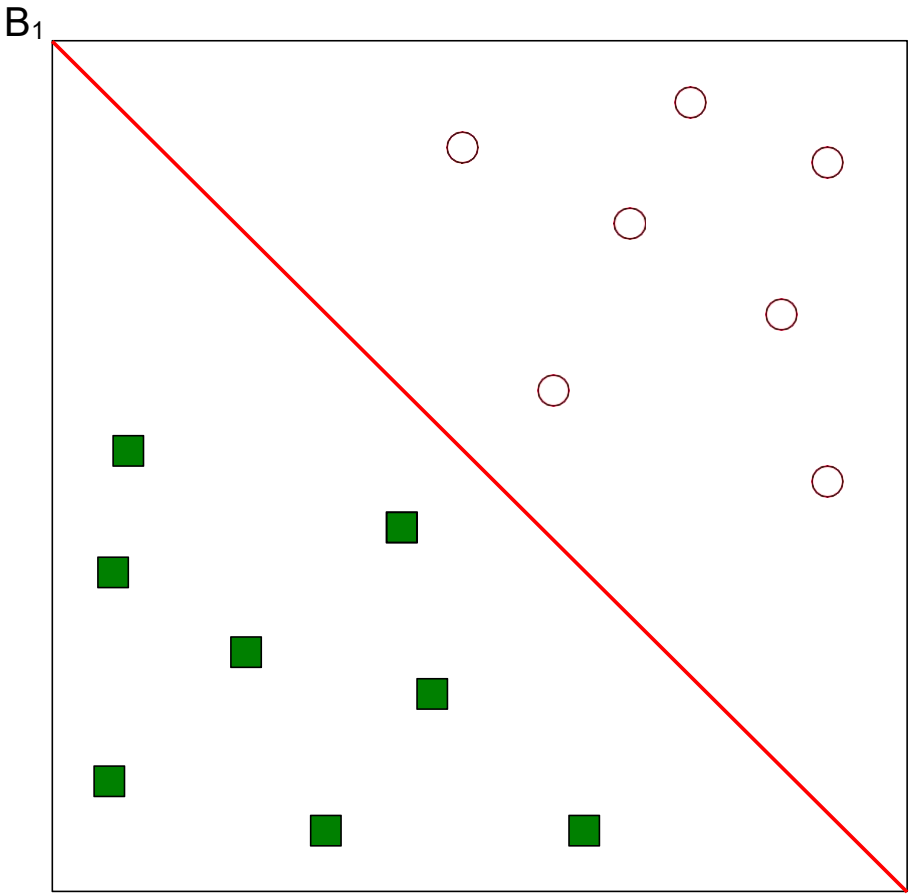
- SVMs are part of a family of classifiers that assumes that the classes are linearly separable
- That is, there is a hyperplane that separates (approximately, or exactly) the instances of the two classes.
- The goal is to find this hyperplane

Support Vector Machines



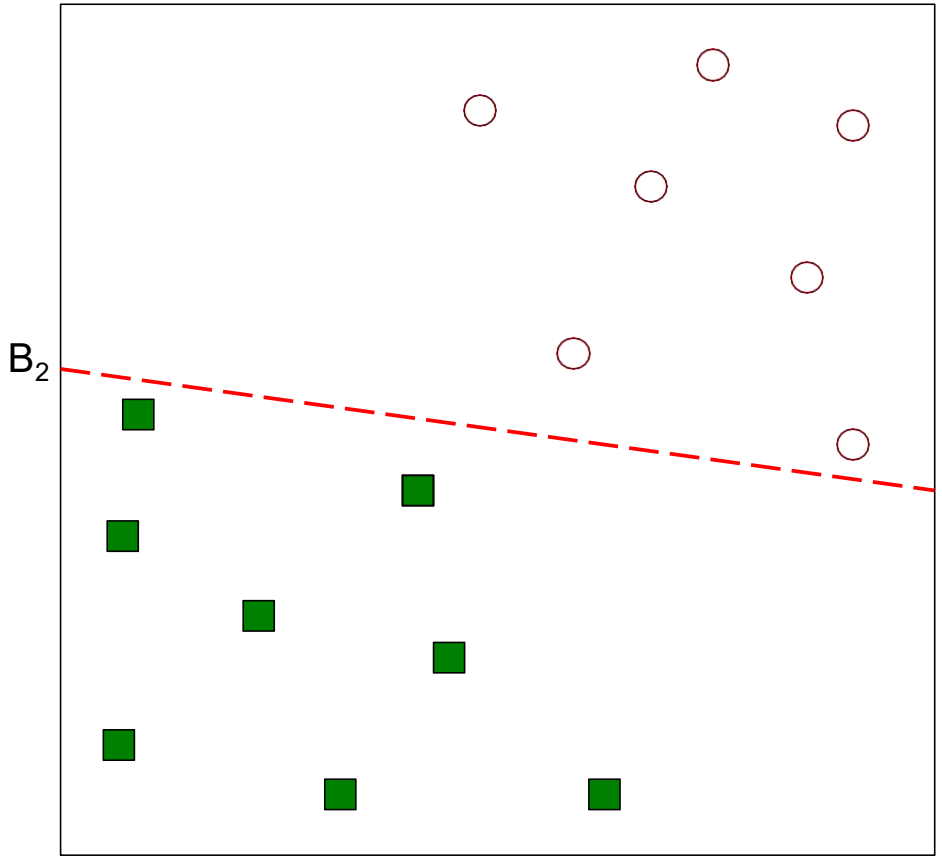
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



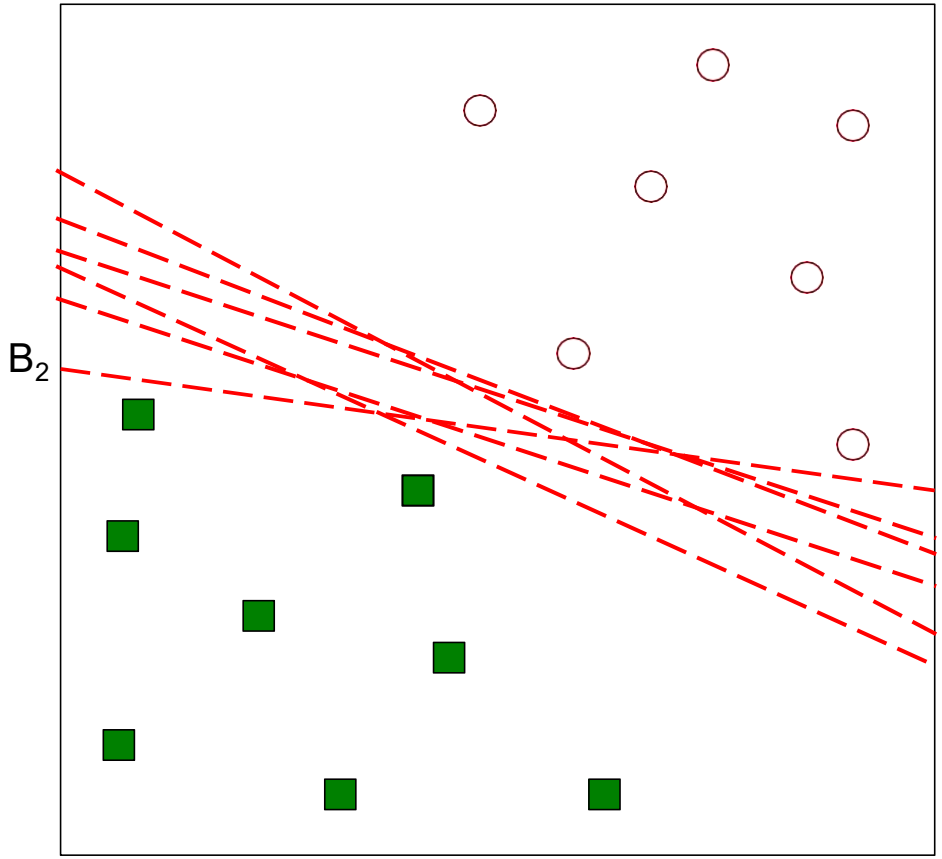
- One Possible Solution

Support Vector Machines



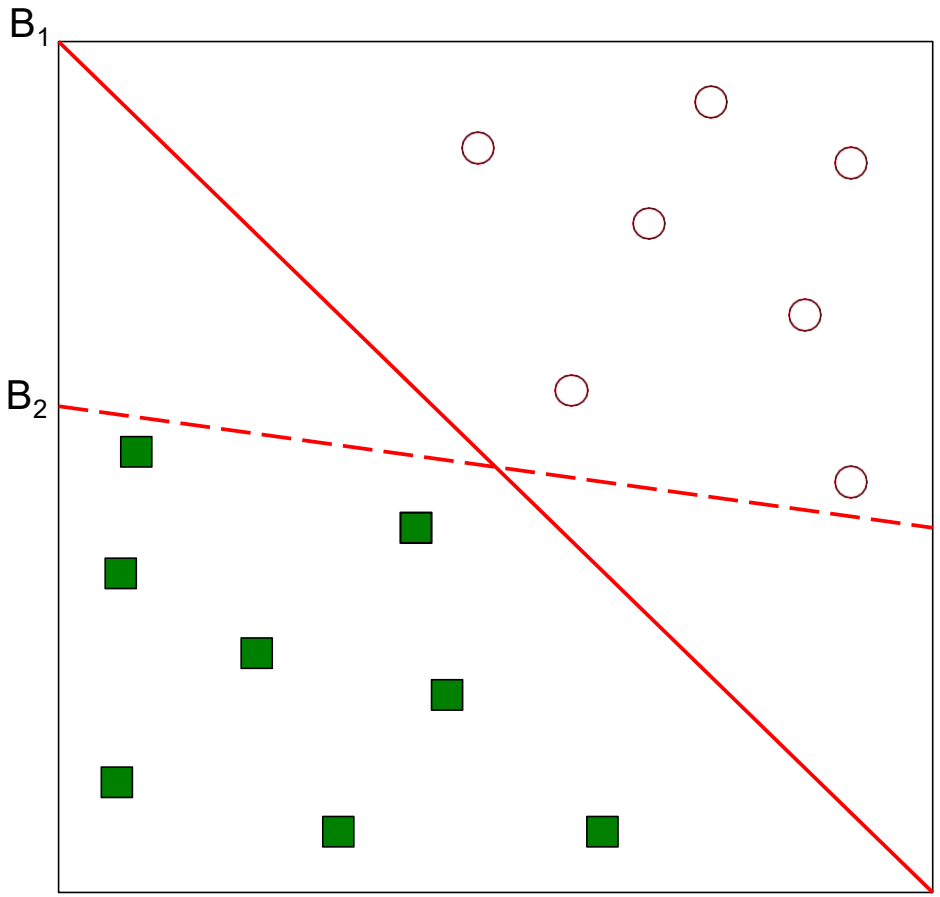
- Another possible solution

Support Vector Machines



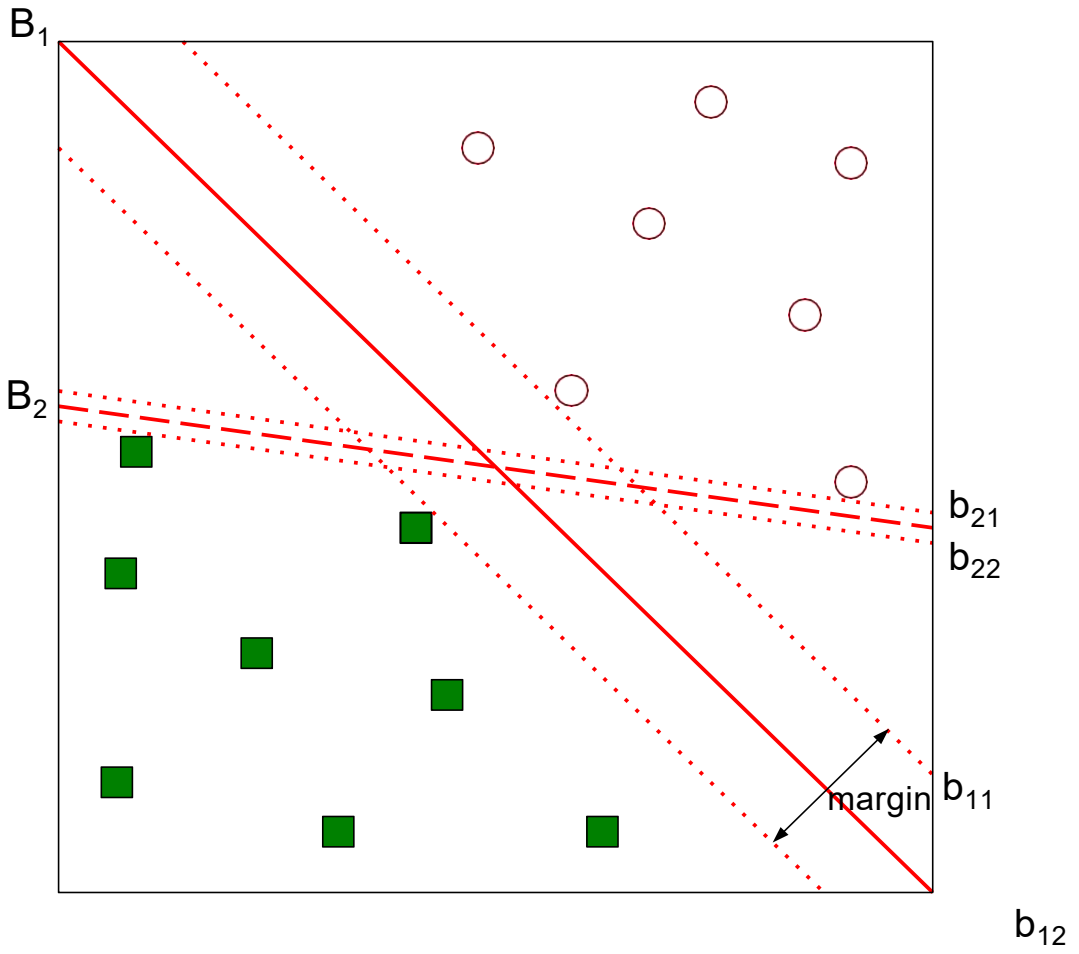
- Other possible solutions

Support Vector Machines

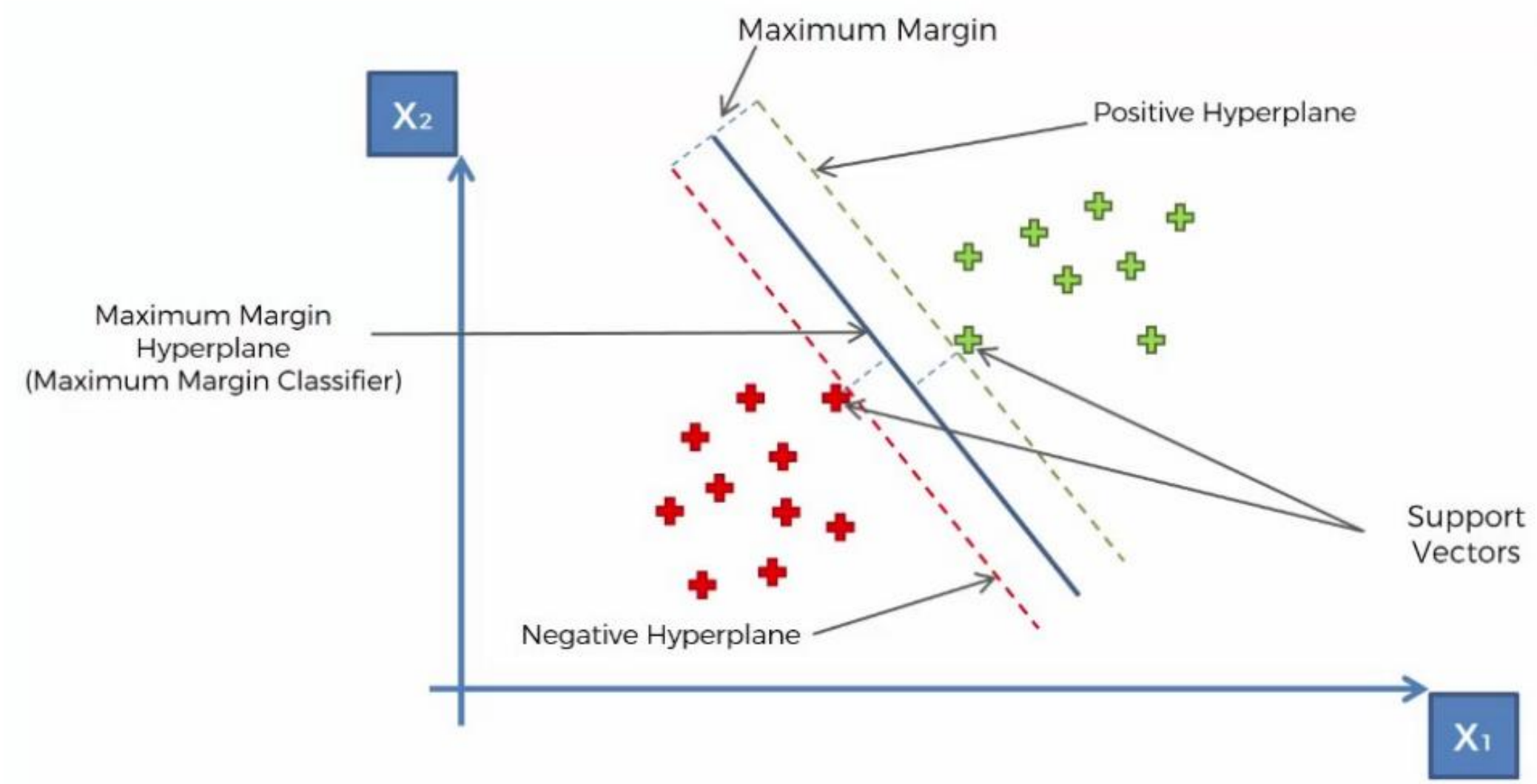


- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



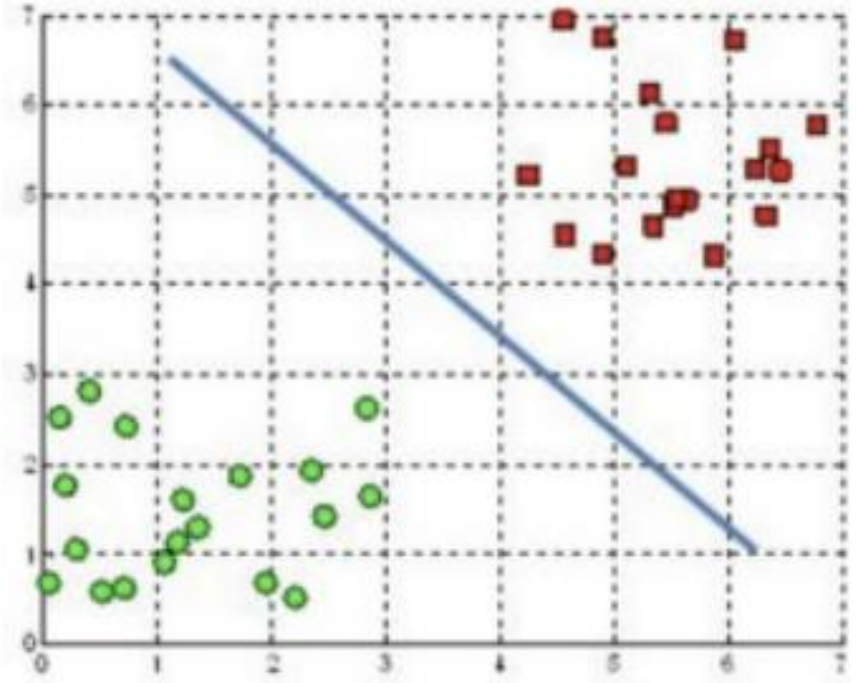
- Find hyperplane maximizes the margin : B1 is better than B2


Maximum Margin
X2
Positive Hyperplane
Maximum Margin
Hyperplane
(Maximum Margin Classifier)
Support
Vectors
Negative Hyperplane
X1

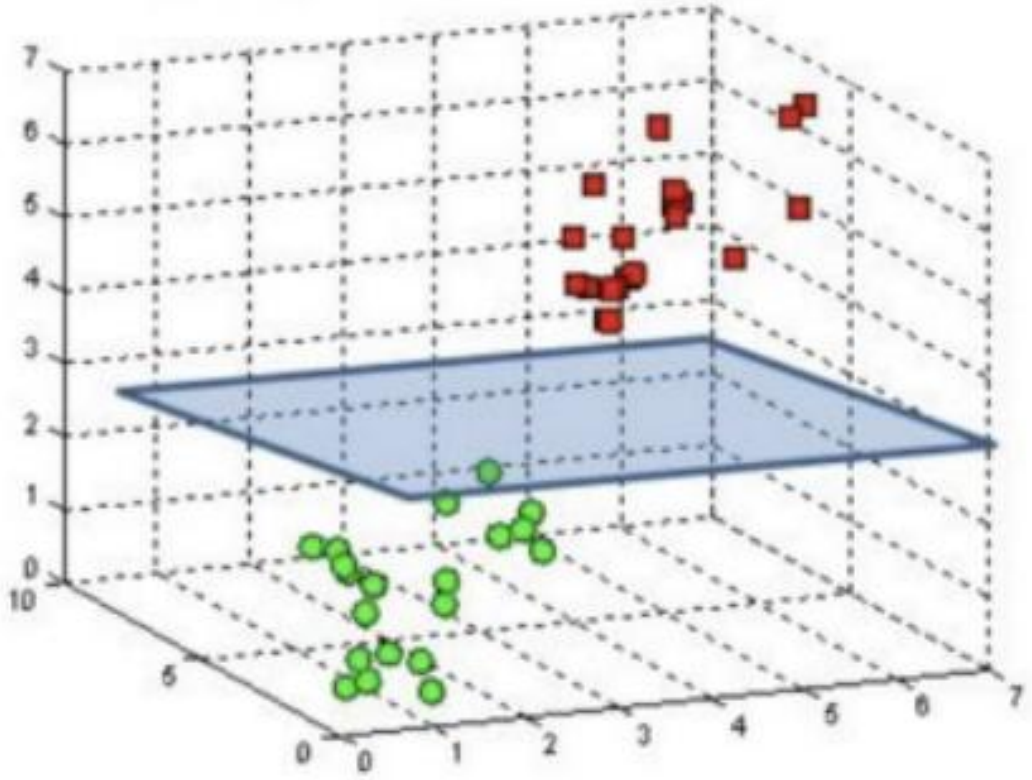
Definitions

- Support Vectors
  - Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.
- Hyperplane
  - A hyperplane is a decision plane which separates between a set of objects having different class memberships.

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



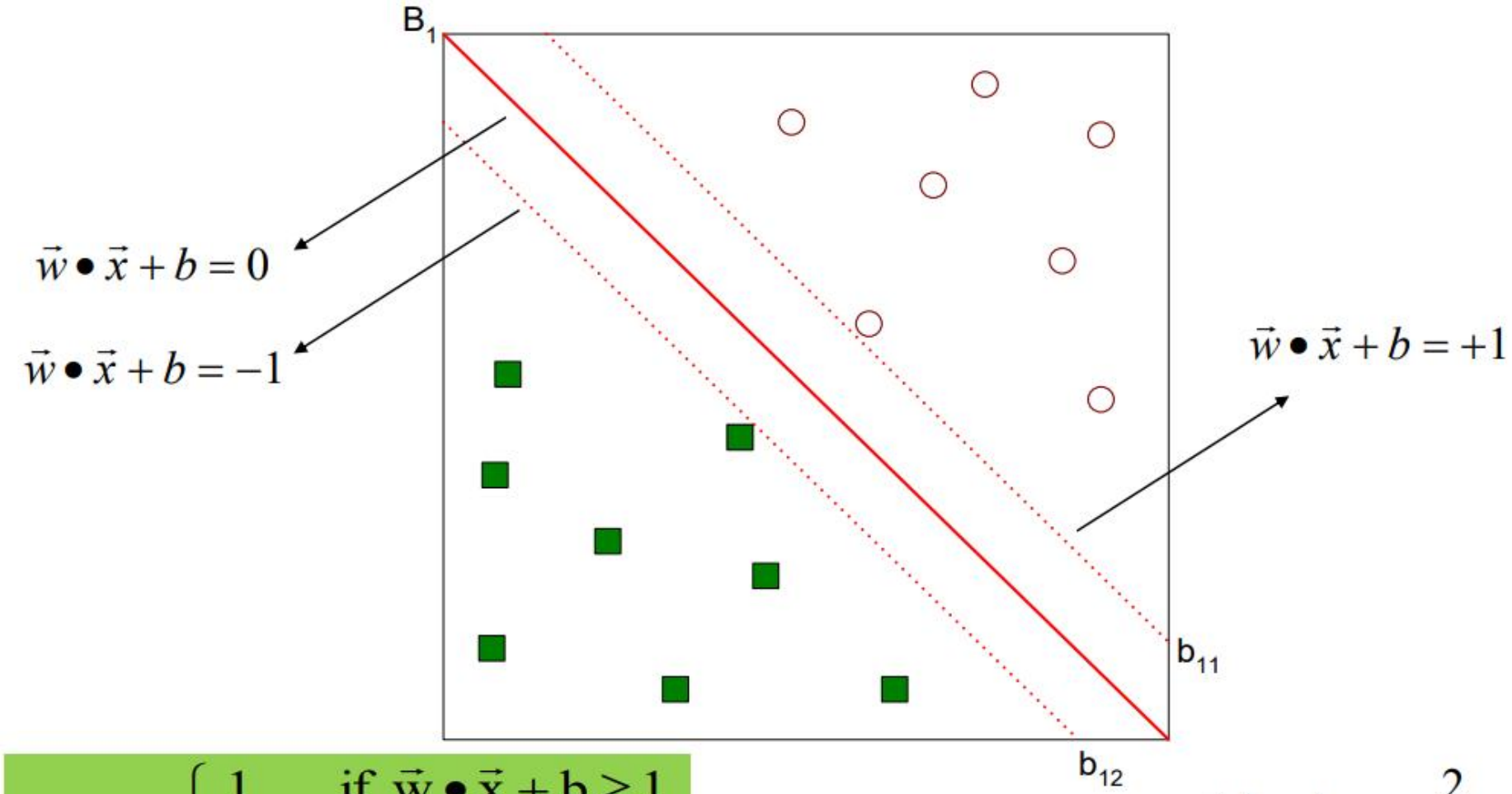
Definitions

- Margin
  - A margin is a gap between the two lines on the closest class points.
  - This is calculated as the perpendicular distance from the line to support vectors or closest points.
  - If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

How SVM Work?

- The main objective is to segregate the given dataset in the best possible way.
- The distance between the either nearest points is known as the margin.
- The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:
  - Generate hyperplanes which segregates the classes in the best way.
  - Select the right hyperplane with the maximum segregation from the either nearest data points.

Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Support Vector Machines

- We want to maximize: $Margin = \frac{2}{\|\vec{w}\|}$
- Which is equivalent to minimizing: $L(\vec{w}) = \frac{\|\vec{w}\|}{2}$
- But subjected to the following constraints:

$$\vec{w} \cdot \vec{x_i} + b \geq 1 \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 \text{ if } y_i = -1$$

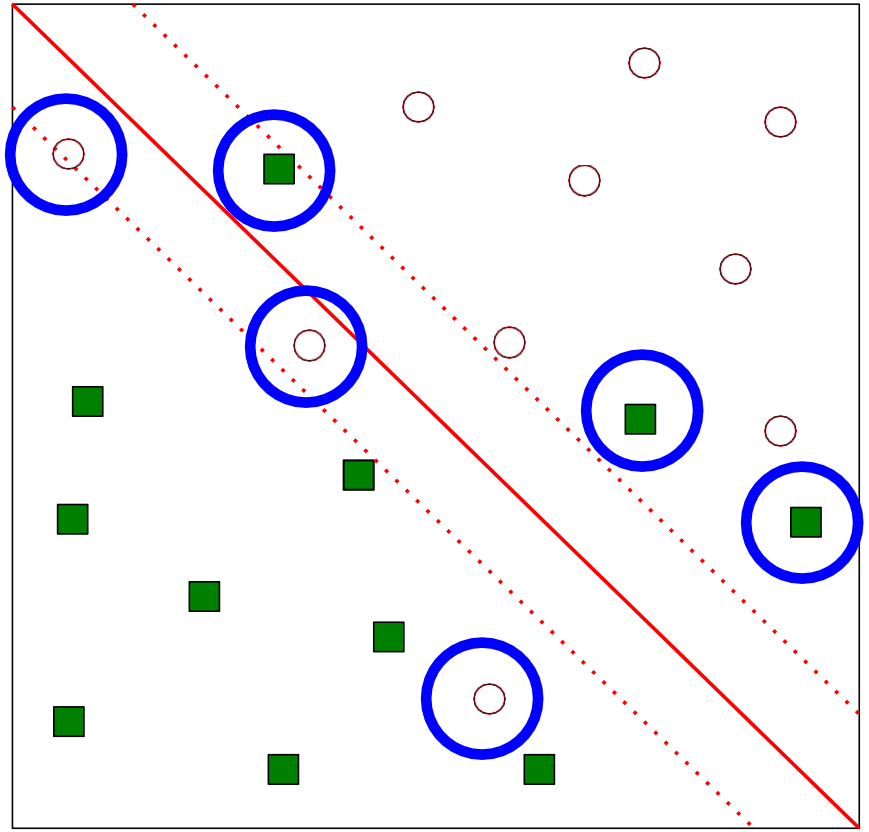
Concisely:

$$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1$$

- This is a constrained optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



LAZY LEARNERS

- '**Lazy**': Do not create a model of the training instances in advance
- When an instance arrives for testing, runs the algorithm to get the class prediction
- **Lazy Learners → Instance Based Classifiers**
  - **Instance-based Classifiers**: do not create a model but use training examples directly to classify unseen examples

Example, ***k* – nearest neighbour** classifier (***k* – NN classifier**)

Eager vs Lazy learners

- Eager learner
  - Generalized model from training data set is constructed
  - Using the model the class of test data set is predicted
  - Example – decision tree
- Lazy learner
  - Training dataset is stored
  - On querying similarity between test data and training set records is calculated to predict the class of test data
  - Example – k-Nearest Neighbour

- ***Lazy: less time in training but more time in predicting***
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Example Problem: Face Recognition

- We have a database of (say) 1 million face images
- We are given a new image and want to find the most similar images in the database
- Represent faces by (relatively) invariant values, e.g., ratio of nose width to eye width
- Each image represented by a large number of numerical features
- Problem: given the features of a new face, find those in the DB that are close in at least ¾ (say) of the features

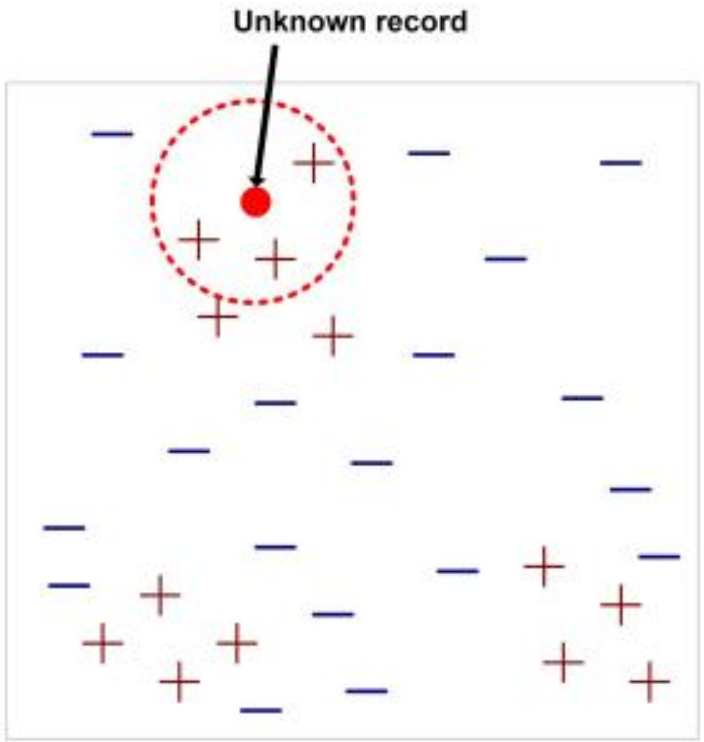
KNN

- The k-nearest neighbors algorithm, also known as KNN or k-NN, is a **non-parametric** and **lazy learning algorithm,** which uses proximity to make classifications about the grouping of an individual data point.
  - **Lazy learning algorithm** – It does not have a specialized training phase and uses all the data for training while classification.
  - **Non-parametric learning algorithm** – It doesn't assume anything about the underlying data.
- The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point.
- If k is set to 1, the instance will be assigned to the same class as its single nearest neighbor. If k is set to 5, the classes of 5 closest points are checked.
- Defining k can be a balancing act as different values can lead to over fitting or under fitting. **Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance.**
- The choice of k will largely depend on the input data as **data with more outliers or noise will likely perform better with higher values of k.**
- Overall, it is **recommended to have an odd number for k** to avoid ties in classification, and **cross-validation** tactics can help to choose the optimal k for the dataset.

k-Nearest-Neighbor Classifiers

- All instances correspond to points in the ***n-dimentional*** space
- The training tuples are described by *n* attributes.
- Each tuple represents a point in an *n-dimensional* space.
- A **k-nearest-neighbor classifier** searches the pattern space for the k training tuples that are closest to the unknown tuple.
- Prediction for test data is done on the basis of its neighbour.

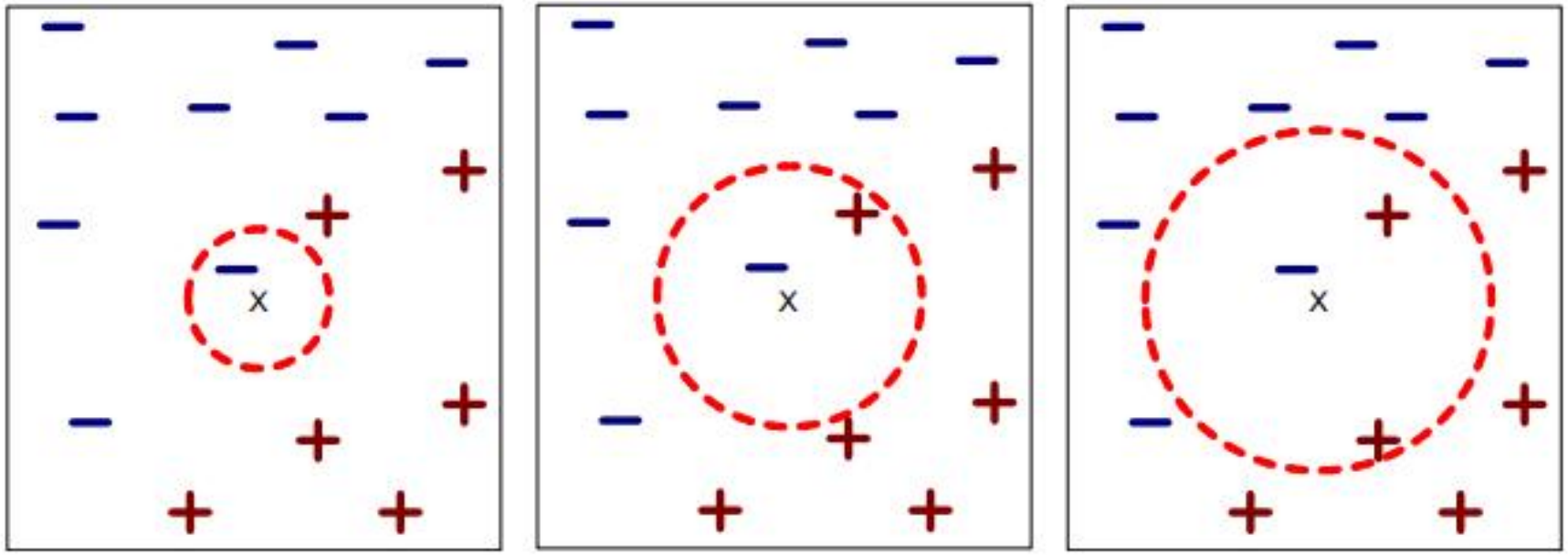
kNN— k Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

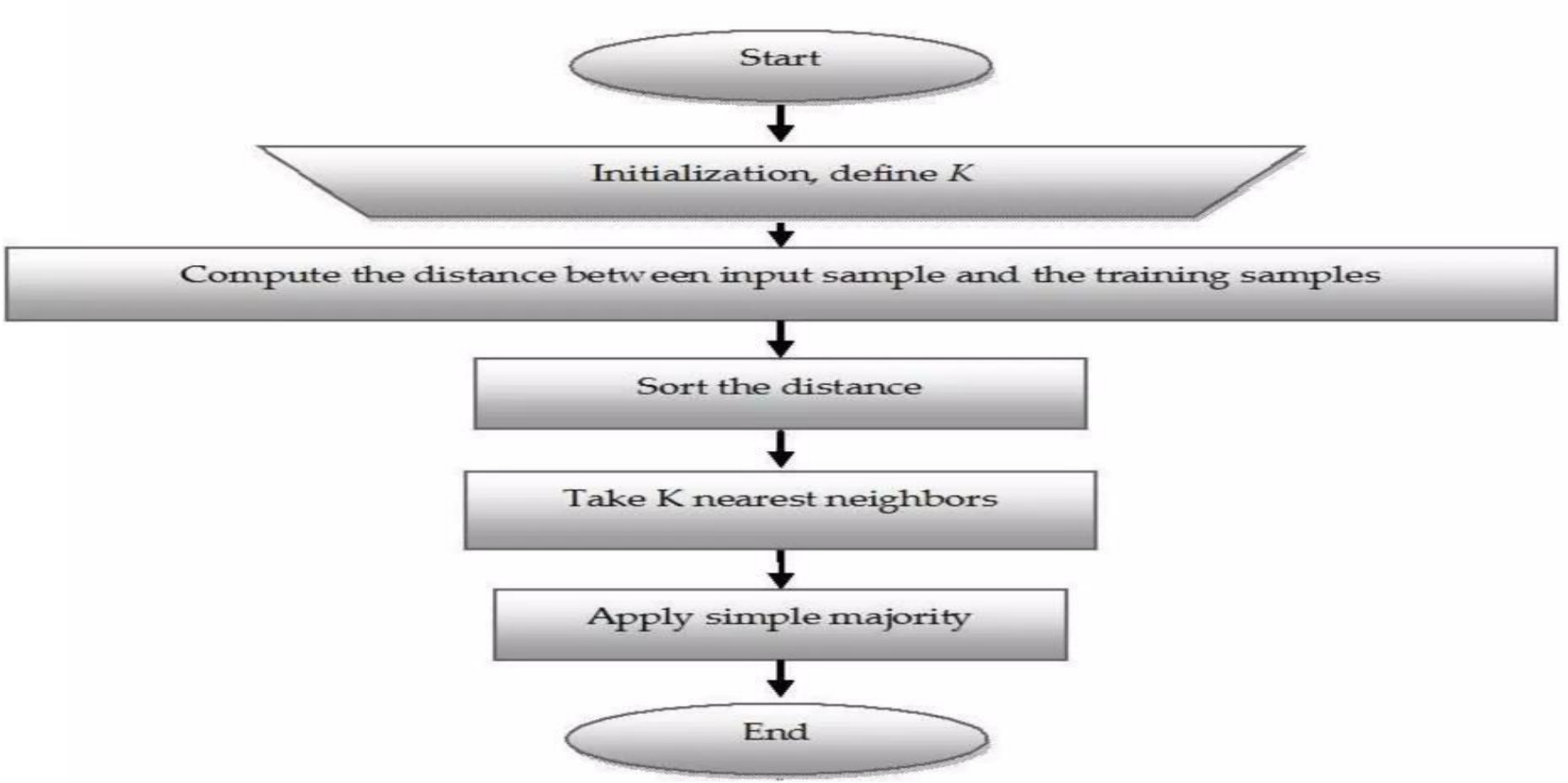
Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

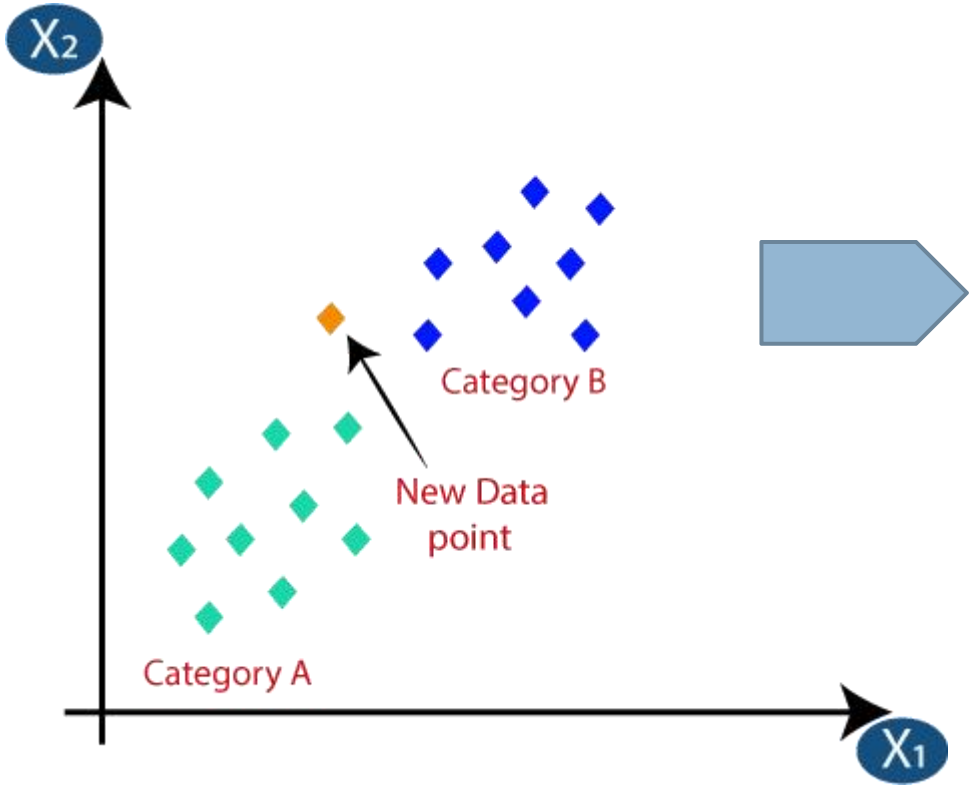
K-nearest neighbors of a record x are data points that have the k smallest distance to x

Working of KNN



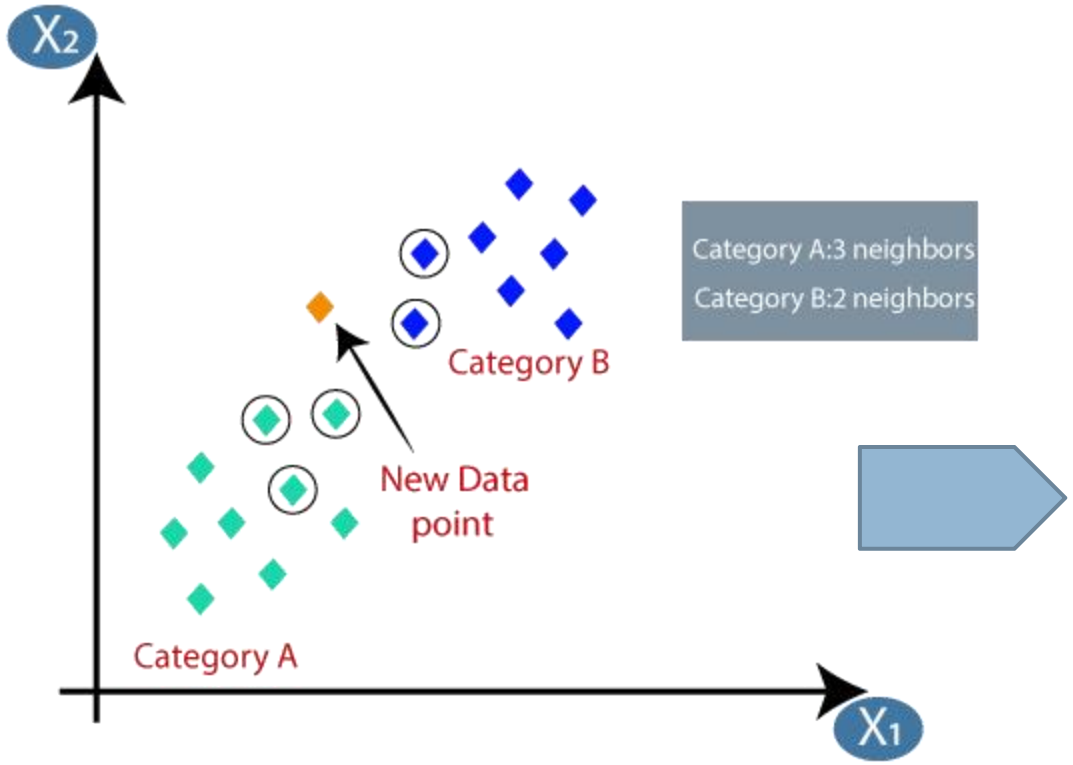
Working of KNN Example

Consider a new data point that need to put it in the required category



1. Firstly, choose the number of neighbors, so let k=5.
2. Next, calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points.
3. By calculating the Euclidean distance, the nearest neighbors are derived i.e., three nearest neighbors in category A and two nearest neighbors in category B.

Working of KNN Example cont…



As 3 nearest neighbors are from category A, hence this new data point must belong to category A.

KNN Example

Let given is the data from the questionnaires survey with two attributes (X1: Acid durability and X2: Strength) to classify whether a special paper tissue is good or not. Classify the new paper tissue with features: X1=3 and X2=7.

X1: Acid Durability	X2: Strength	Y: Class
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

KNN Solution

Step 1: Initialize and Define K

K=1 Over fitting Problem

Number of tuples is even, hence consider K as odd.

Let K = 3 for this problem

Step 2: Compute distance between input sample and training samples using Euclidian distance measure.

$$dist((x, y), (a, b)) = \sqrt{((x - a)^2 + (y - b)^2)}$$

X1: Acid Durability	X2: Strength	Y: Class	Distance from (3, 7)
7	7	Bad	sqrt((7-3)^2 + (7-7)^2)=4
7	4	Bad	sqrt((7-3)^2 + (4-7)^2)=5
3	4	Good	sqrt((3-3)^2 + (4-7)^2)=3
1	4	Good	sqrt((1-3)^2 + (4-7)^2)=3.6

KNN Solution cont…

Step 3: Determine K nearest neighbor those are at minimum distance and rank them.

X1: Acid Durability	X2: Strength	Y: Class	Distance from (3, 7)	Rank
7	7	Bad	4	3
7	4	Bad	5	4
3	4	Good	3	1
1	4	Good	3.6	2

Step 4: Apply Simple Majority

There are 2 “Good” and 1 “Bad”. Thus the new paper tissue will be classified as Good due to simple majority.

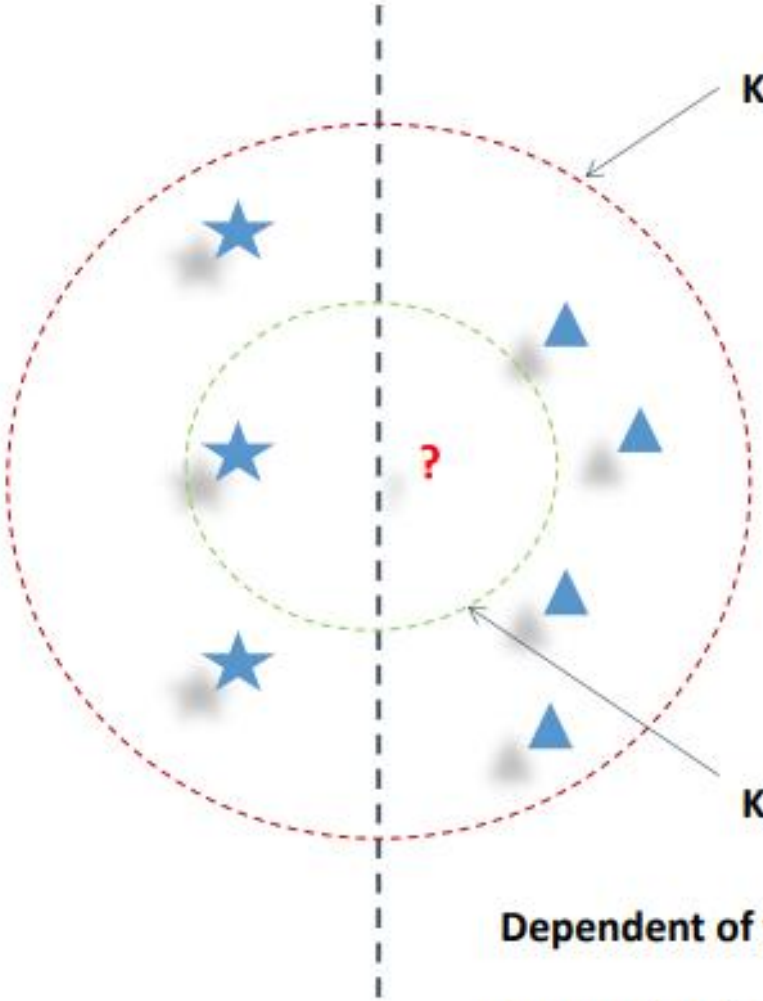
KNN Exercise

Given is the customer detail as customer's age, income, credit card no. and their corresponding class. Find the class label for the new data item (reflecting in red font):

Customer**:** RRR, Age: 37, Income: 50K, Cr Card: 2, Class- ?

Customer	Age	Income	No. of Credit Cards	Class
XXX	35	35K	3	No
YYY	22	50K	2	Yes
ZZZ	63	200K	1	No
PPP	59	170K	1	No
QQQ	25	40K	4	Yes
RRR	**37**	**50K**	**2**	**????**

Properties of KNN



Main questions:

– How many neighbors should we consider? That is, what is k?

– How do we measure distance?

– Should all points be weighted equally, or should some points have more influence than others?

Dependent of the data distributions.

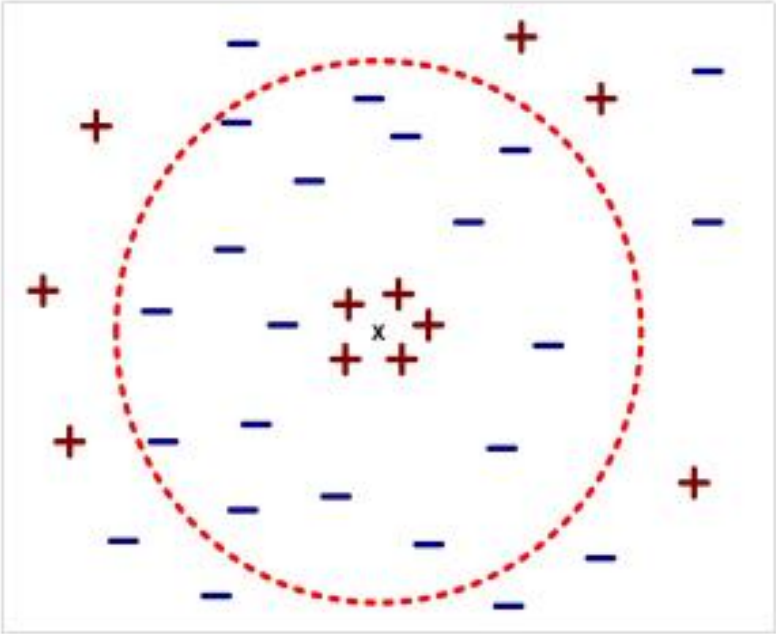
Can make mistakes at boundaries.

Determining a good value for k

- ***k*** can be determined experimentally.
- Starting with ***k = 1***, we use a test set to estimate the error rate of the classifier.
- This process can be repeated each time by incrementing k to allow for one more neighbor.
- The k value that gives the minimum error rate may be selected.
- In general, the larger the number of training tuples is, the larger the value of k will be.

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Summary Nearest Neighbor Classifiers

- k-NN classifiers are lazy learners
  - Unlike eager learners such as decision tree induction and rule-based systems, it does not build models explicitly
  - Classifying unknown records is relatively expensive
- Rely on local knowledge ("let the data speak for themselves") and not on global models to make decisions.
- k-NN classifiers rely on a distance function; the quality of the distance function is critical for the performance of a K-NN classifier.
- Capable to create quite complex decision boundaries which consists of edges of the Voronoi diagram.
- K-NN classifiers never actually compute decision boundaries, in contrast to decision trees/SVMs.
- k-NN classifiers obtain high accuracies and are quite popular in some fields, such as text data mining and in information retrieval, in general.

Evaluation of Forecasting Accuracy

- What makes a good forecast? Of course, a good forecast is an accurateforecast.
- A forecast “error” is the difference between an observed value and its forecast. The “error” does not mean a mistake, it means the unpredictable part of an observation.
- Error measure plays an important role in calibrating and refining forecasting model/method and helps the analyst to improve forecasting method.
- The choice of an error measure may vary according to the situation , number of time series available and on whether the task is to select the most accurate method or to calibrate a given model.
- The popular and highly recommended error measures are
  - Mean Square Error (MSE)
  - Root Mean Square Error (RMSE)
  - Mean Absolute Percentage Error (MAPE)

Mean Square Error (MSE)

MSE is defined as mean or average of the square of the difference between actual and estimated values. Mathematically it is represented as:

$$\mathrm{MSE} = \frac{\sum_{j=1}^{N} (\text{observation } (j) - \text{prediction } (j))^2}{N}$$

Month	1	2	3	4	5	6	7	8	9	10	11	12
Actual Demand	42	45	49	55	57	60	62	58	54	50	44	40
Forecasted Demand	44	46	48	50	55	60	64	60	53	48	42	38
Error	-2	-1	1	5	2	0	-2	-2	1	2	2	2
Squared Error	4	1	1	25	4	0	4	4	1	4	4	4

Sum of Square Error = 56 and MSE = 56 / 12 = 4.6667

Root Mean Square Error (RMSE)

It is just the square root of the mean square error. Mathematically it is represented as:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{j=1}^{N} (\text{observation } (j) - \text{prediction } (j))^2}{N}}$$

Month	1	2	3	4	5	6	7	8	9	10	11	12
Actual Demand	42	45	49	55	57	60	62	58	54	50	44	40
Forecasted Demand	44	46	48	50	55	60	64	60	53	48	42	38
Error	-2	-1	1	5	2	0	-2	-2	1	2	2	2
Squared Error	4	1	1	25	4	0	4	4	1	4	4	4

Sum of Square Error = 56, MSE = 56 / 12 = 4.6667, RMSE = SQRT(4.667) = 2.2

Mean Absolute Percentage Error (MAPE)

The formula to calculate MAPE is as follows:

$$\text{MAPE} = (100/n) \times \sum_{i=1}^{n} \frac{(|X'(t) - X(t)|}{X(t)}$$

Here, X'(t) represents the forecasted data value of point t and X(t) represents the actual data value of point t. Calculate MAPE for the below dataset.

Month	1	2	3	4	5	6	7	8	9	10	11	12
Actual Demand	42	45	49	55	57	60	62	58	54	50	44	40
Forecasted Demand	44	46	48	50	55	60	64	60	53	48	42	38

- MAPE is commonly used because it's easy to interpret and easy to explain. For example, a MAPE value of 11.5% means that the average difference between the forecasted value and the actual value is 11.5%.
- The lower the value for MAPE, the better a model is able to forecast values e.g. a model with a MAPE of 2% is more accurate than a model with a MAPE of 10%.

MAE

3. Mean Absolute Error (MAE)

Average of absolute differences between predicted and actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Regression

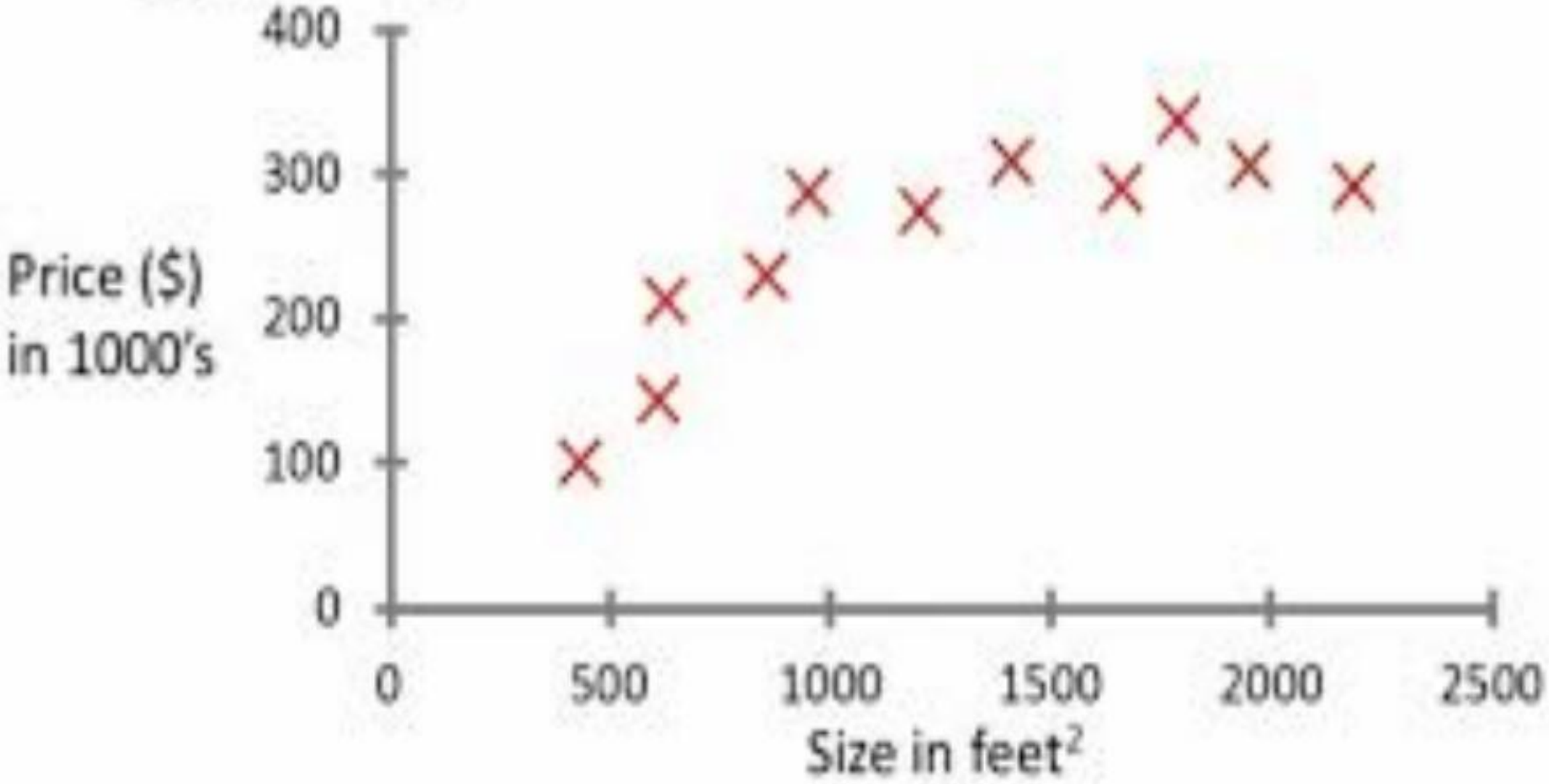
- Linear regression is a statistical technique that predicts a continuous value based on one or more independent attributes.
- For example -
  - Predicting the house price based on the area of house
  - Predicting the future income of a student based on the college she attended, her major in university, overall GPA, etc.
  - Predicting salary of an employee based on work experience.

REGRESSION: CASE STUDY

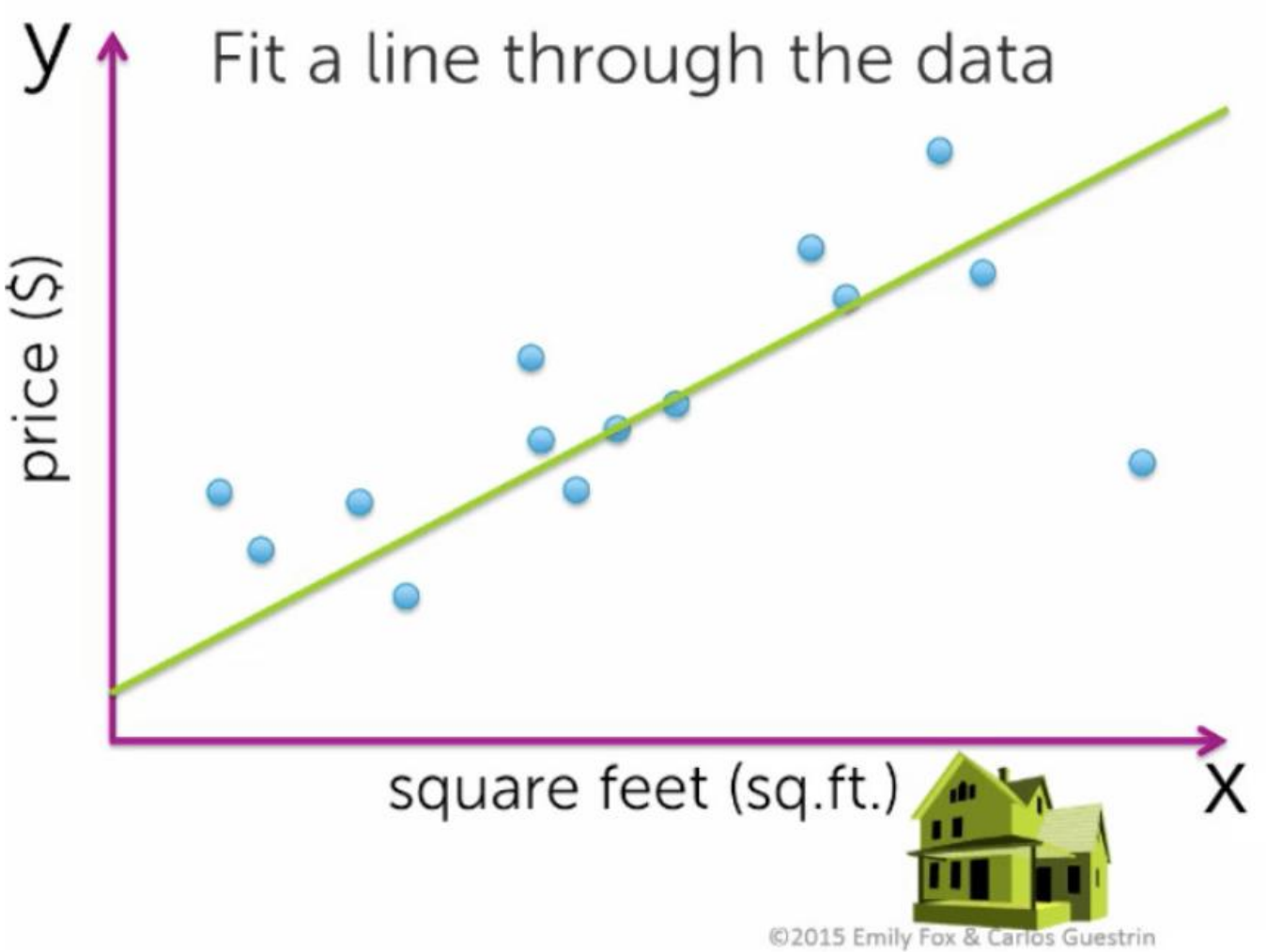
area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus	price
7420	4	2	3	yes	no	no	no	yes	2	yes	furnished	13300000
8960	4	4	4	yes	no	no	no	yes	3	no	furnished	12250000
9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished	12250000
7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished	12215000
7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished	11410000
7500	3	3	1	yes	no	yes	no	yes	2	yes	semi-furnished	10850000
8580	4	3	4	yes	no	no	no	yes	2	yes	semi-furnished	10150000
16200	5	3	2	yes	no	no	no	no	0	no	unfurnished	10150000
8100	4	1	2	yes	yes	yes	no	yes	2	yes	furnished	9870000
5750	3	2	4	yes	yes	no	no	yes	1	yes	unfurnished	9800000

- **Features/Attributes** - {area, bedrooms, bathrooms, etc.}
  - Also known as independent variables
- Target / Output - price
  - Also known as dependent variable
- **Feature Vector -** Vector representation of features like [area, bedrooms, bathrooms, …]

Housing price prediction.

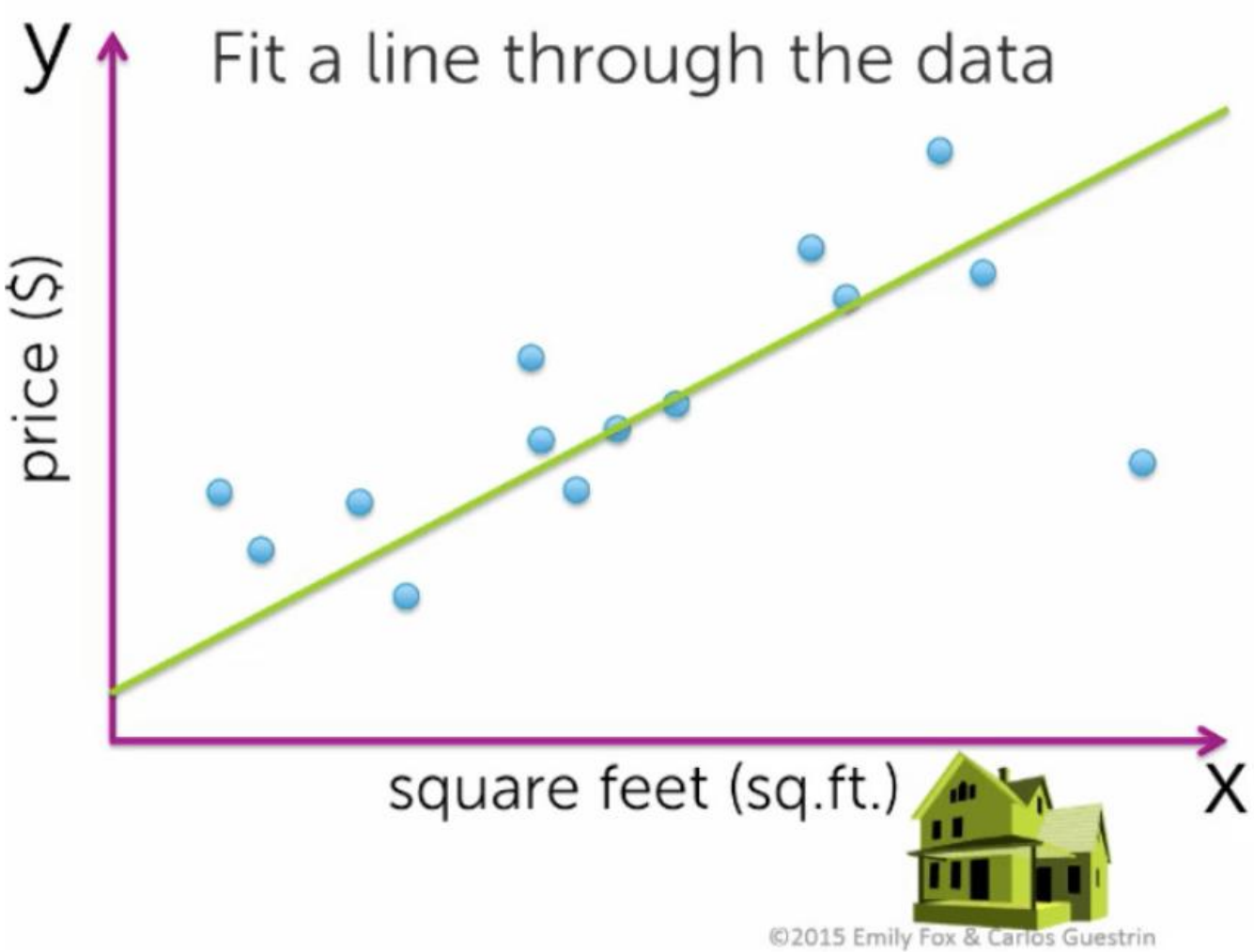


Simple Linear Regression: Model



- What do we need to define a line?
  - General line equation is **Y = M X + C**
  - Slope
  - Intercept

SIMPLE LINEAR REGRESSION: MODEL

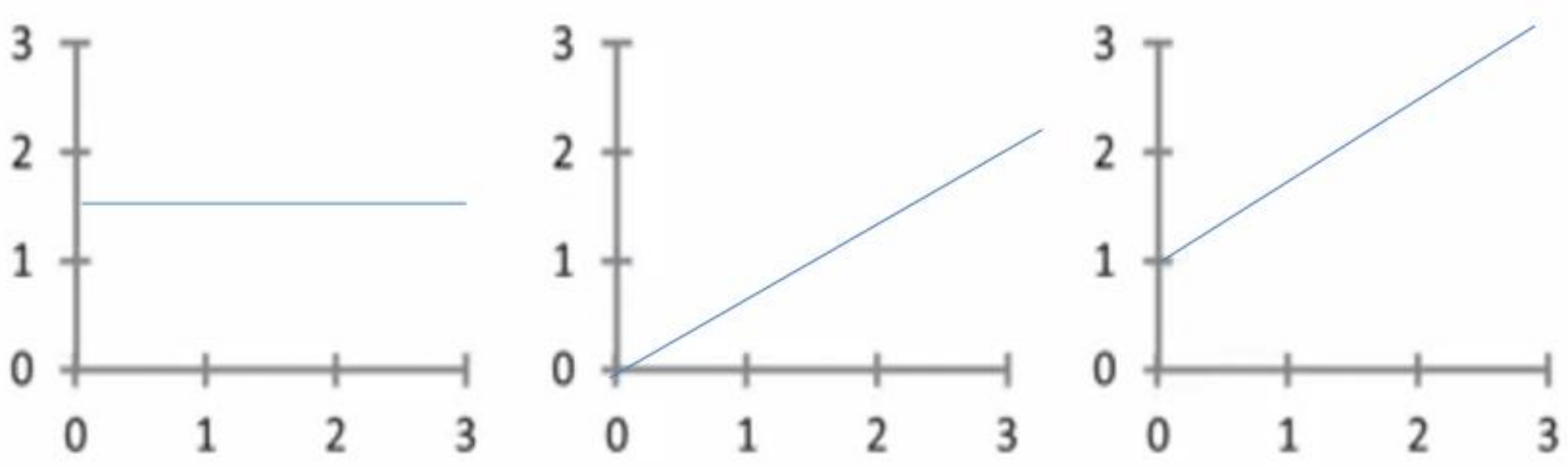


- What do we need to define a line?
  - General line equation is **Y = M X + C**
  - Slope
  - Intercept

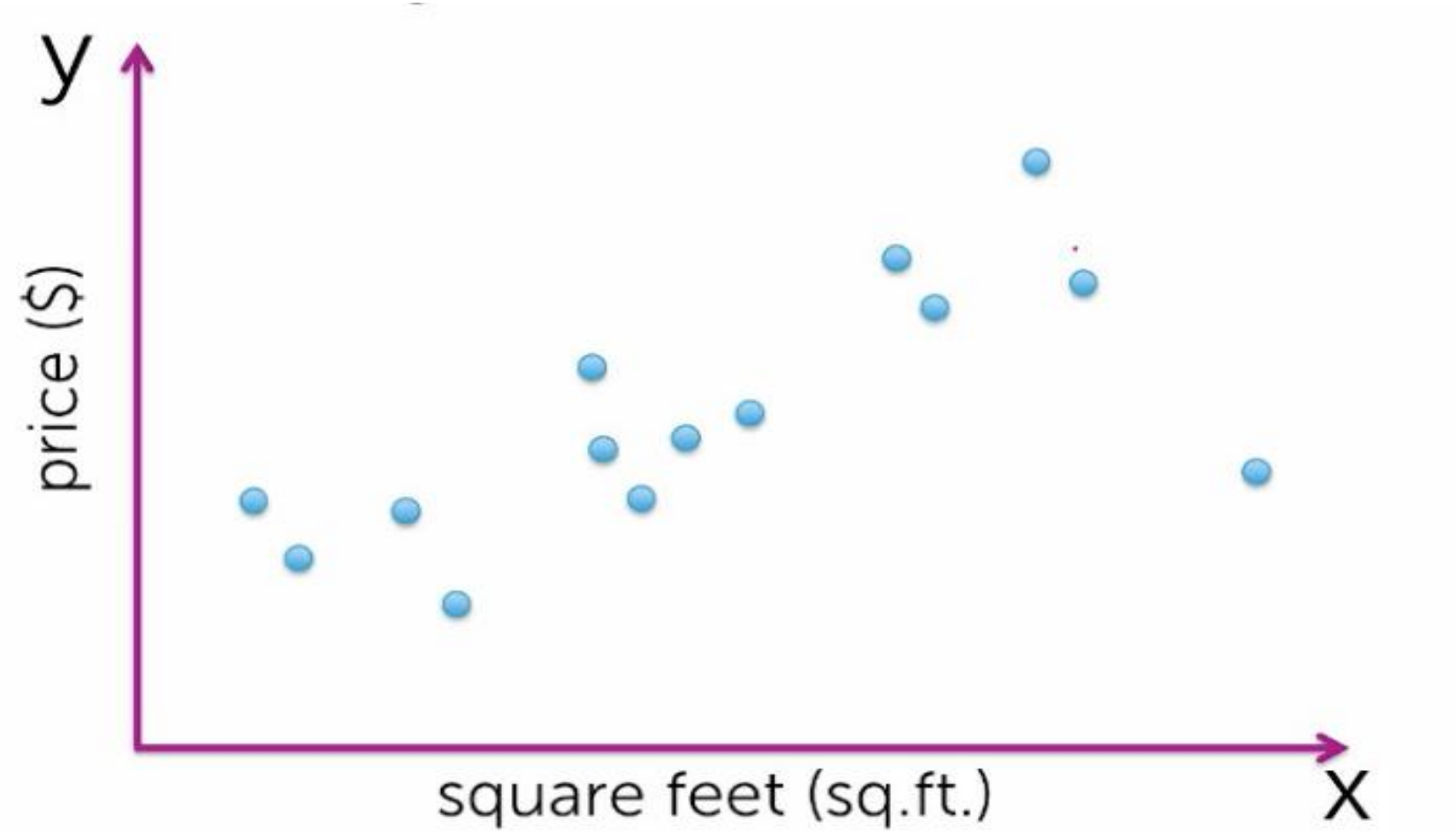
- Let's denote our slope using $\boldsymbol{w}$ and intercept using $\boldsymbol{b}$. So our linear regression model can be defined as $\boldsymbol{f_{w,b}(x) = wx + b}$ where $\boldsymbol{x}$ is the input variable or feature and $\boldsymbol{f(x)}$ or $\boldsymbol{y}$ is the target variable that we want to predict.
- w and b are known as parameters of the model which we need to learn during training of the model.

Simple Linear Regression: Model Representation

- What will be the values of $\boldsymbol{w}$ and $\boldsymbol{b}$ for following models?



DRAW THE LINE WHICH FITS THE FOLLOWING DATA BEST?



How do we define the notion of the best fit?

Linear Regression: Cost Function

- We say a line fits the data best, if it minimizes the error between predicted value and actual value also known as the cost of that line.
- Most common type of error used in regression is mean squared error which is defined as following -

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

- m – number of training examples
- $(x^{(i)}, y^{(i)})$ – i^{th} training example
- $\hat{y}^{(i)} = f_{w,b}(x^{(i)}) = w\,x^{(i)} + b$ is the predicted value for feature $x^{(i)}$
- $y^{(i)}$ is the actual value for feature $x^{(i)}$ in the training set
- $J(w,b)$ – Squared error cost function

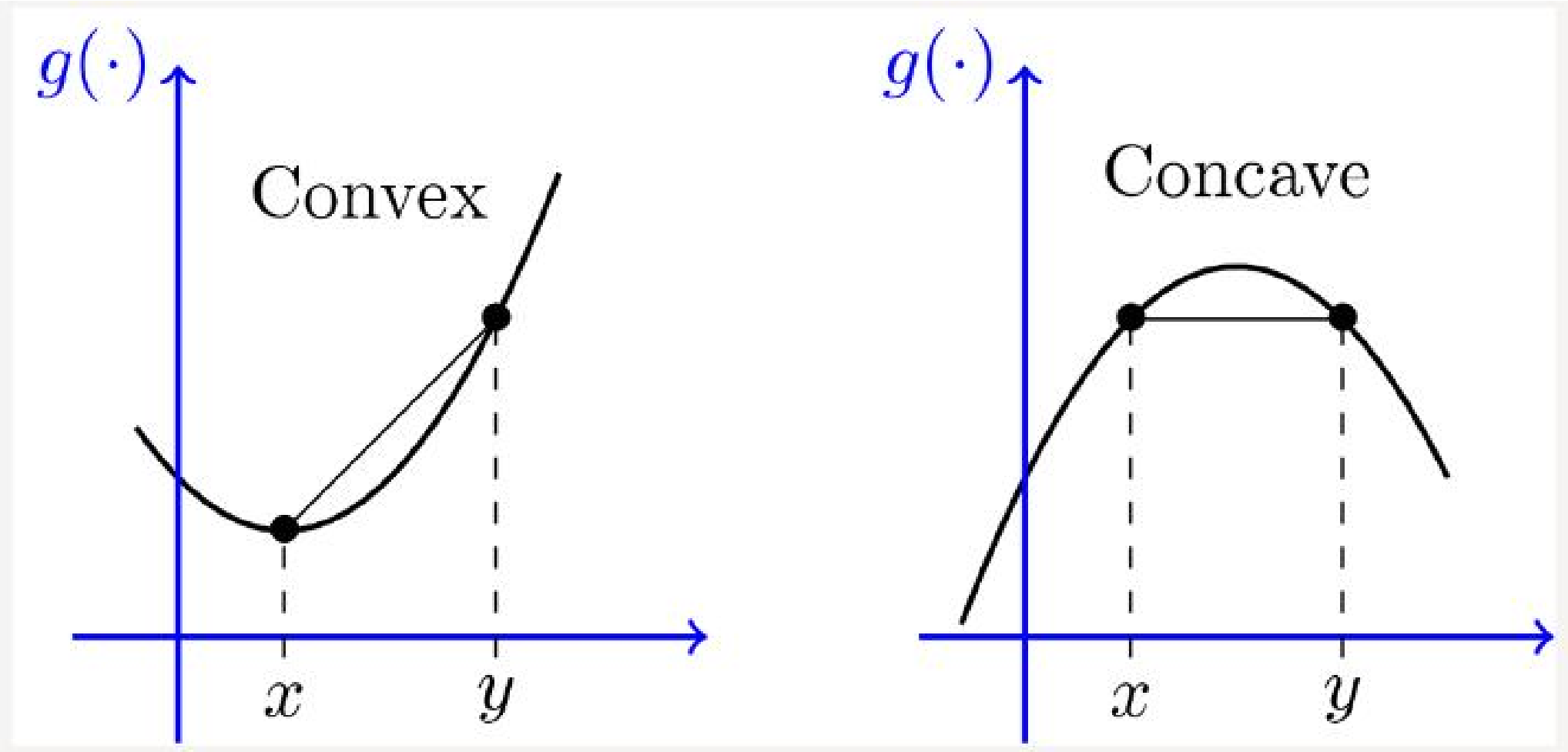
Our goal is to find parameters *w* and *b* such that cost *J(w, b)* is minimized.

Line represented using these parameters will be the best fit model for our data.

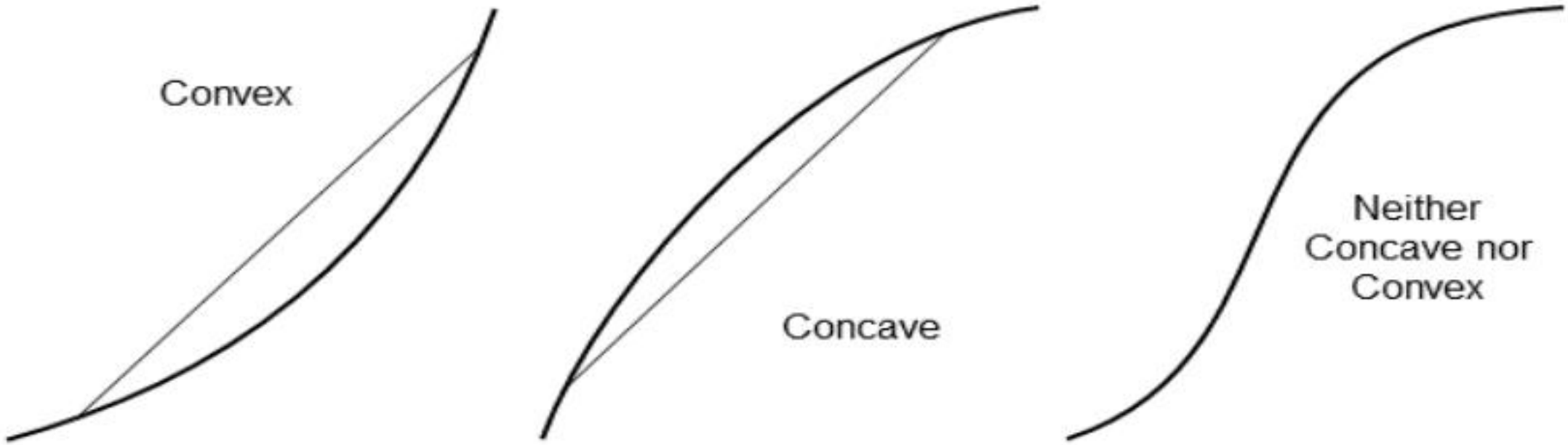
How do we minimize Cost function?

- Before going into the minimization of cost function. Let's talk about how to minimize a polynomial function.
- Given a polynomial $y = 2x^3 - 3x^2 + 6$. Find the values of x for which $y = f(x)$ is minimized and find the minimum value also.

Convex and Concave functions

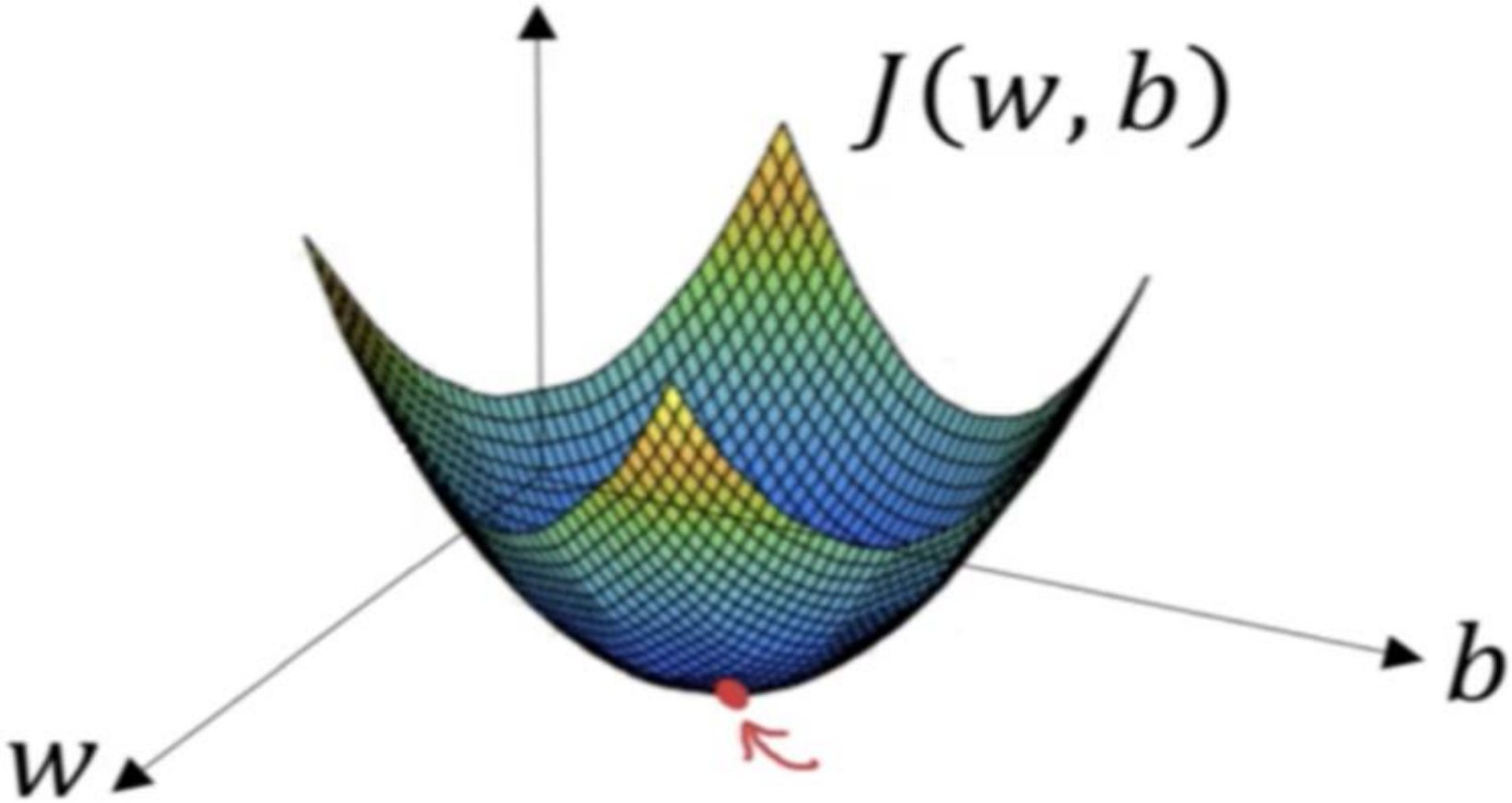


Convex and Concave Functions



- **J(w, b) is also a convex function therefore will have only one minima. At the point of minima, we will find our optimal parameters *w* and *b*.**

Cost function is also convex



MINIMIZE COST FUNCTION J(W, B): CLOSED FORM SOLUTIONS

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$= J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b) - y^{(i)})^2$$

- Compute the derivative of ***J(w, b)*** and set it to **0**.
- We have two variables ***w*** and ***b*** so we need to compute partial derivatives of *J(w, b)* with respect to *w* and *b* and set them to 0.
- Above step will give us, two equations which we can solve to find values of two variables *w* and *b*.

Minimize Cost Function J(w, b): Closed Form Solutions

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$= J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}((wx^{(i)} + b) - y^{(i)})^2$$

Closed form solution

$$w = \frac{m\sum_{i=1}^{m} x^{(i)}y^{(i)} - \sum_{i=1}^{m} x^{(i)}\sum_{i=1}^{m} y^{(i)}}{m\sum_{i=1}^{m}(x^{(i)})^2 - (\sum_{i=1}^{m} x^{(i)})^2}$$

$$b = \frac{1}{m}(\sum_{i=1}^{m} y^{(i)} - w\sum_{i=1}^{m} x^{(i)})$$

Homework -

Try out the maths behind how we came up with these values.

Linear Regression cont...

- The calculation of b and a is as follows:

$$b = \frac{n\Sigma XY - \Sigma X \Sigma Y}{n\Sigma X^2 - (\Sigma X)^2} \qquad a = \frac{\Sigma Y}{n} - b\,\frac{\Sigma X}{n}$$

- If b > 0, then x(predictor) and y(target) have a positive relationship. That is increase in x will increase y.
- If b < 0, then x(predictor) and y(target) have a negative relationship. That is increase in x will decrease y.

- If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.

$$\text{Error} = \sum_{i=1}^{n} (actual\_output - predicted\_output) ** 2$$

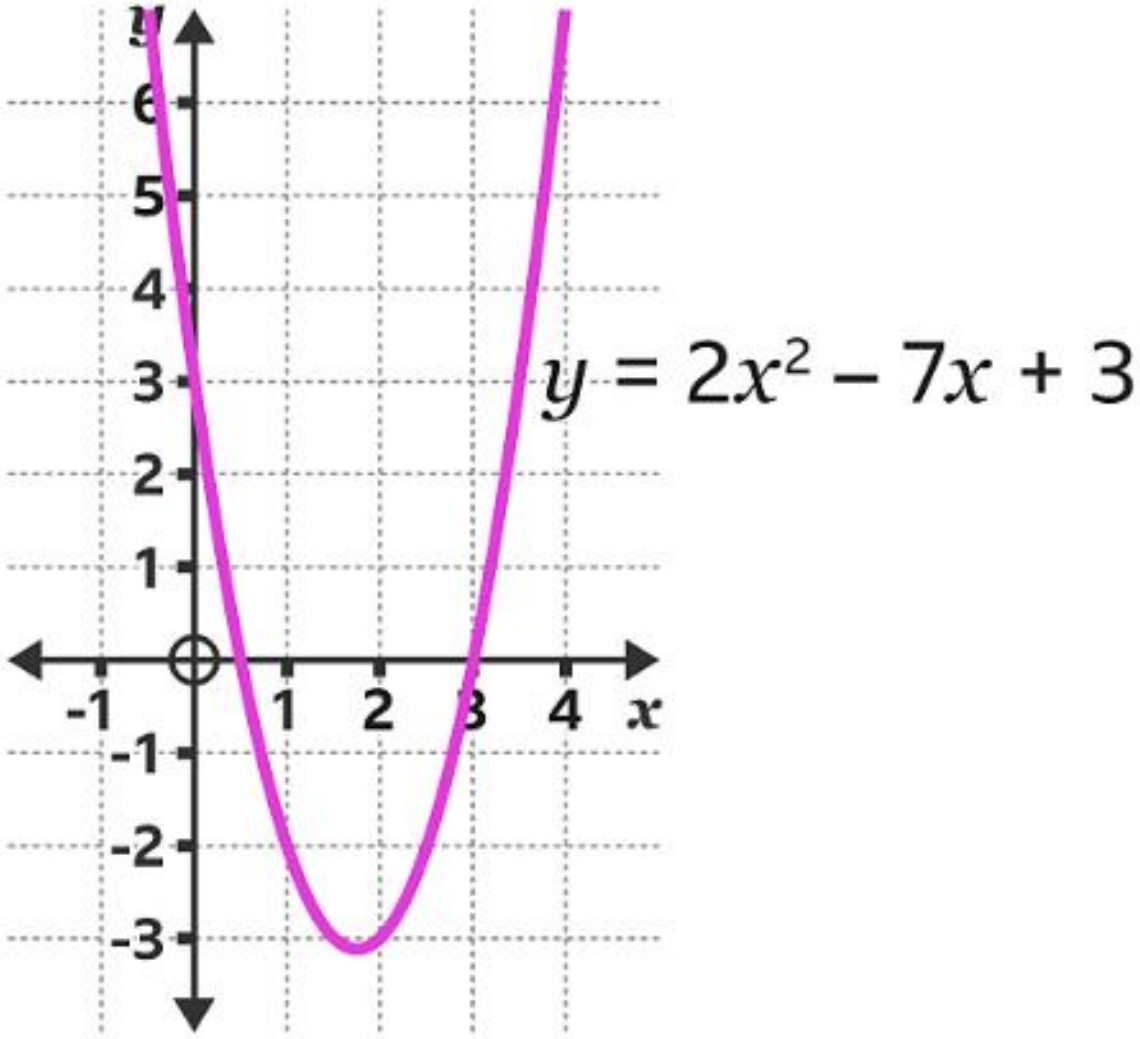
Linear Regression cont...

Company	Sales in 1000s (Y)	Number of agents in 100s (X)
A	25	8
B	35	12
C	29	11
D	24	5
E	38	14
F	12	3
G	18	6
H	27	8
I	17	4
J	30	9

$$b = \frac{10 \times 2289 - (80 \times 255)}{[10 \times 756 - (80)^2]} = 2.1466; \qquad a = \frac{255}{10} - 2.1466\frac{80}{10} = 8.3272$$

Minimize Cost Function: Gradient Descent

- How do we find the minimum of given function with out setting derivative equal to 0?
  - Start with some value of x
  - Keep changing x to reduce $f(x)$ until we reach the minimum or close to minimum

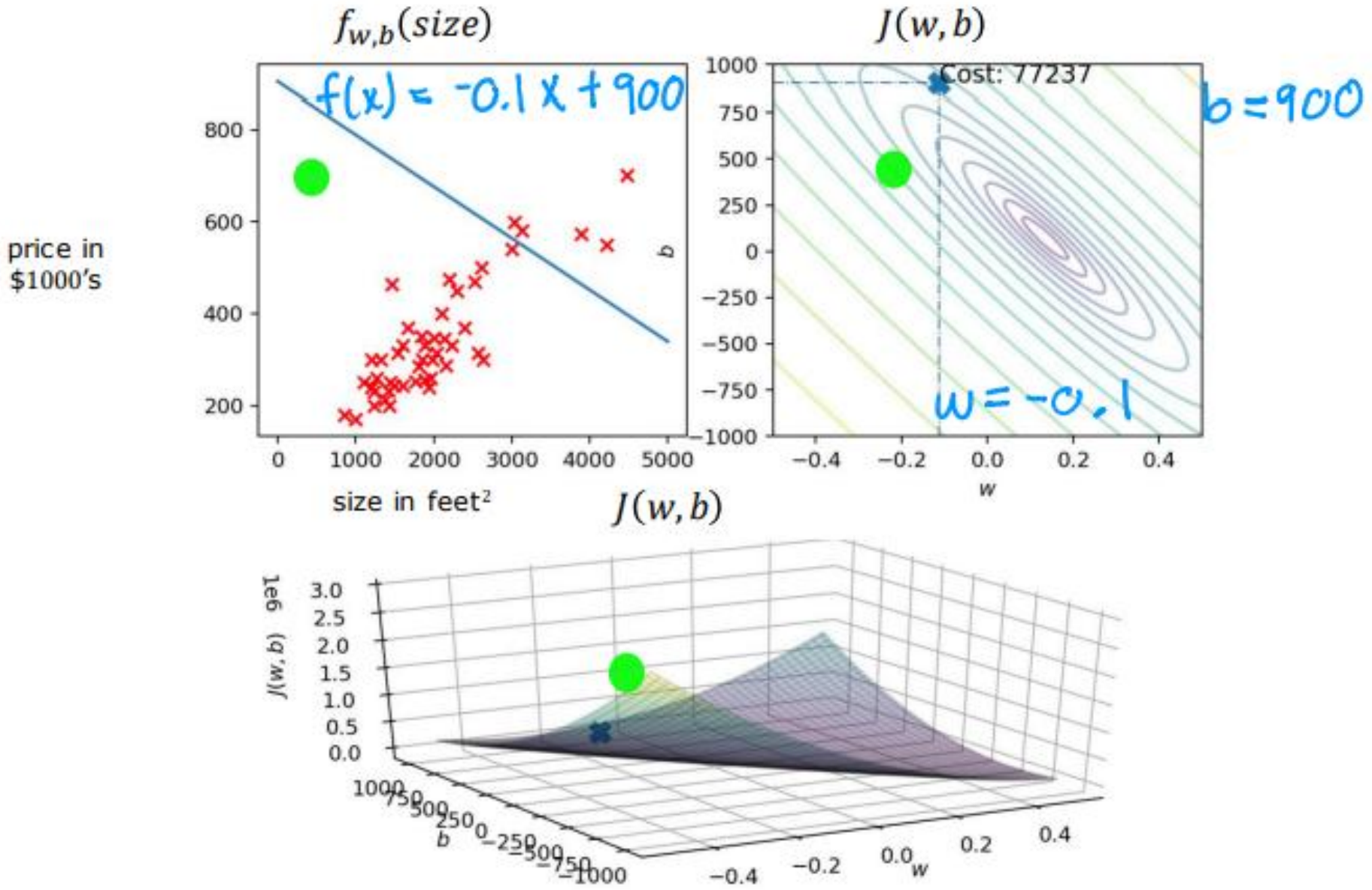


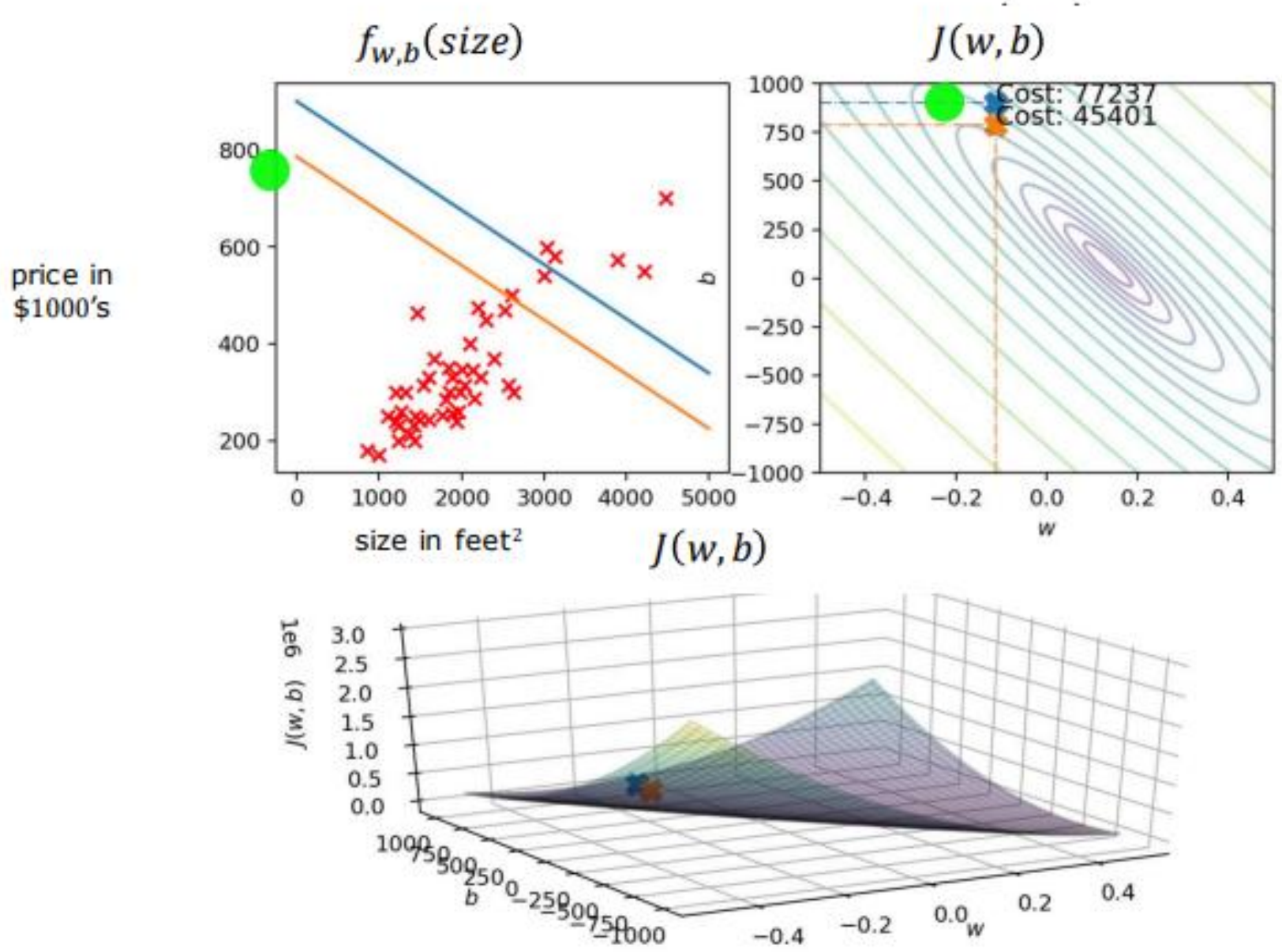
Minimize Cost Function: Gradient Descent

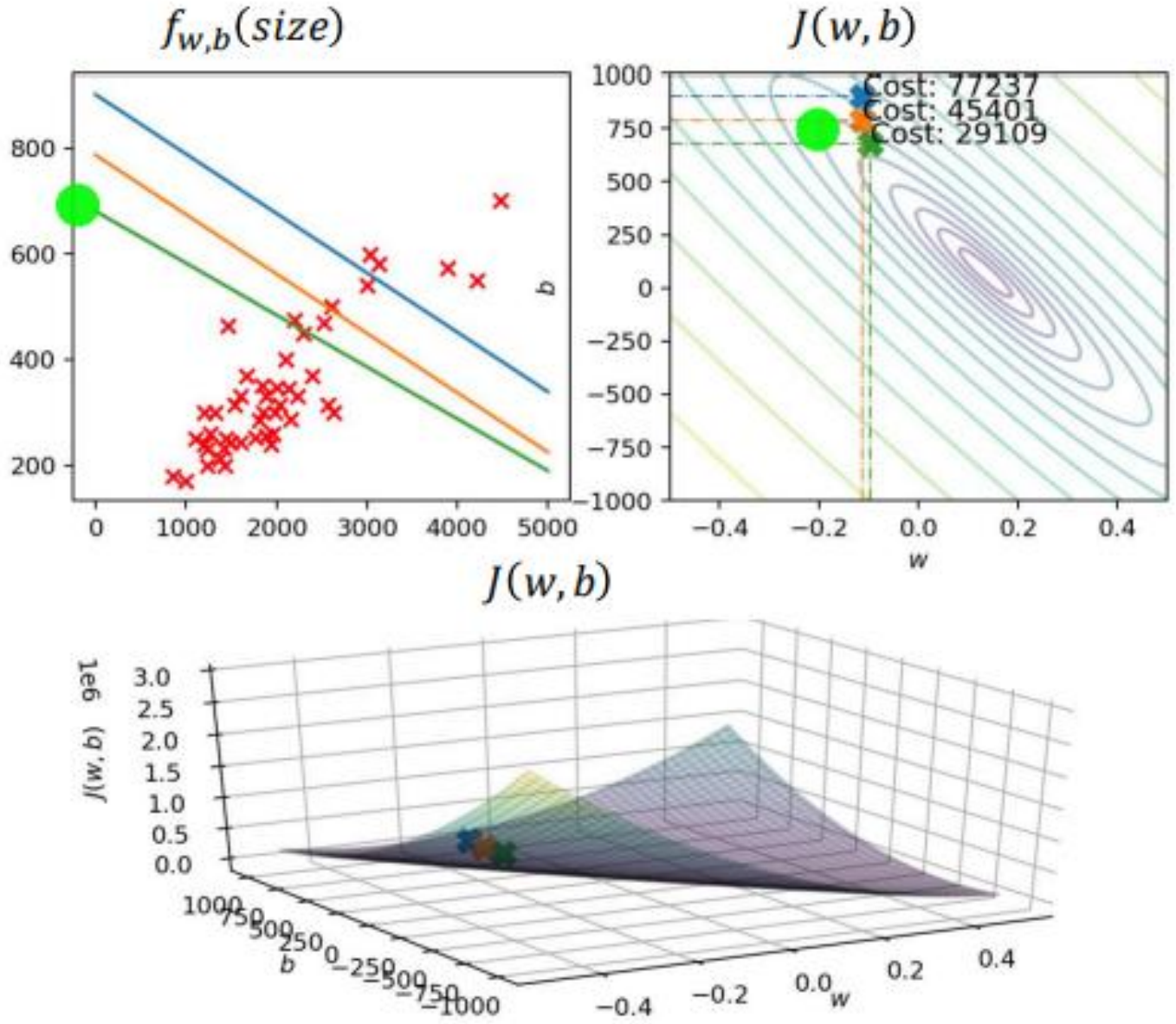
- alpha is also known as ***step size*** or ***learning rate***.
- Usually alpha is a constant between (0, 1).

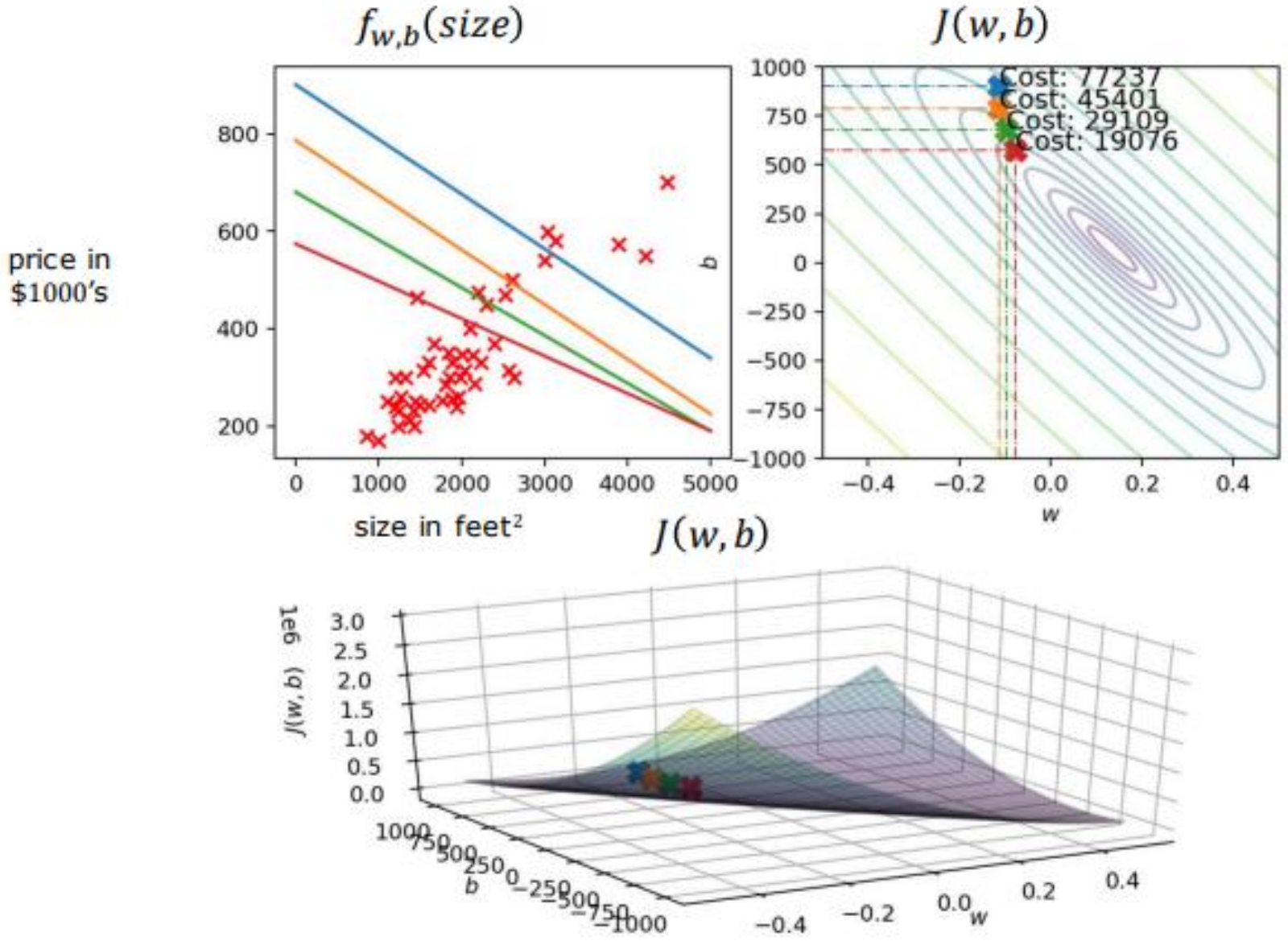
Repeat until convergence:

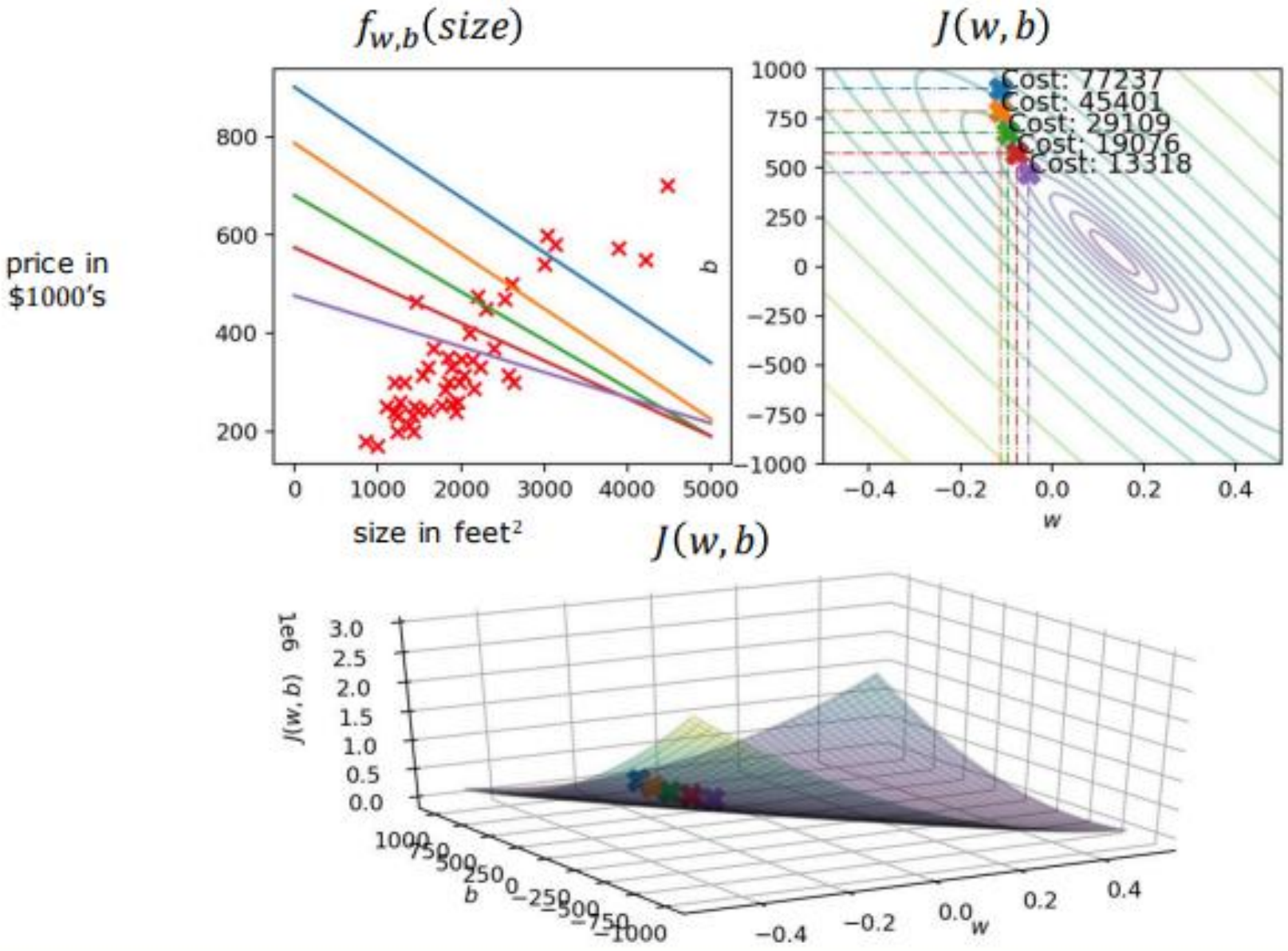
- $tmp\_w = w - \alpha \frac{\delta J(w,b)}{\delta w}$
- $tmp\_b = b - \alpha \frac{\delta J(w,b)}{\delta b}$
- $w = tmp\_w$
- $b = tmp\_b$

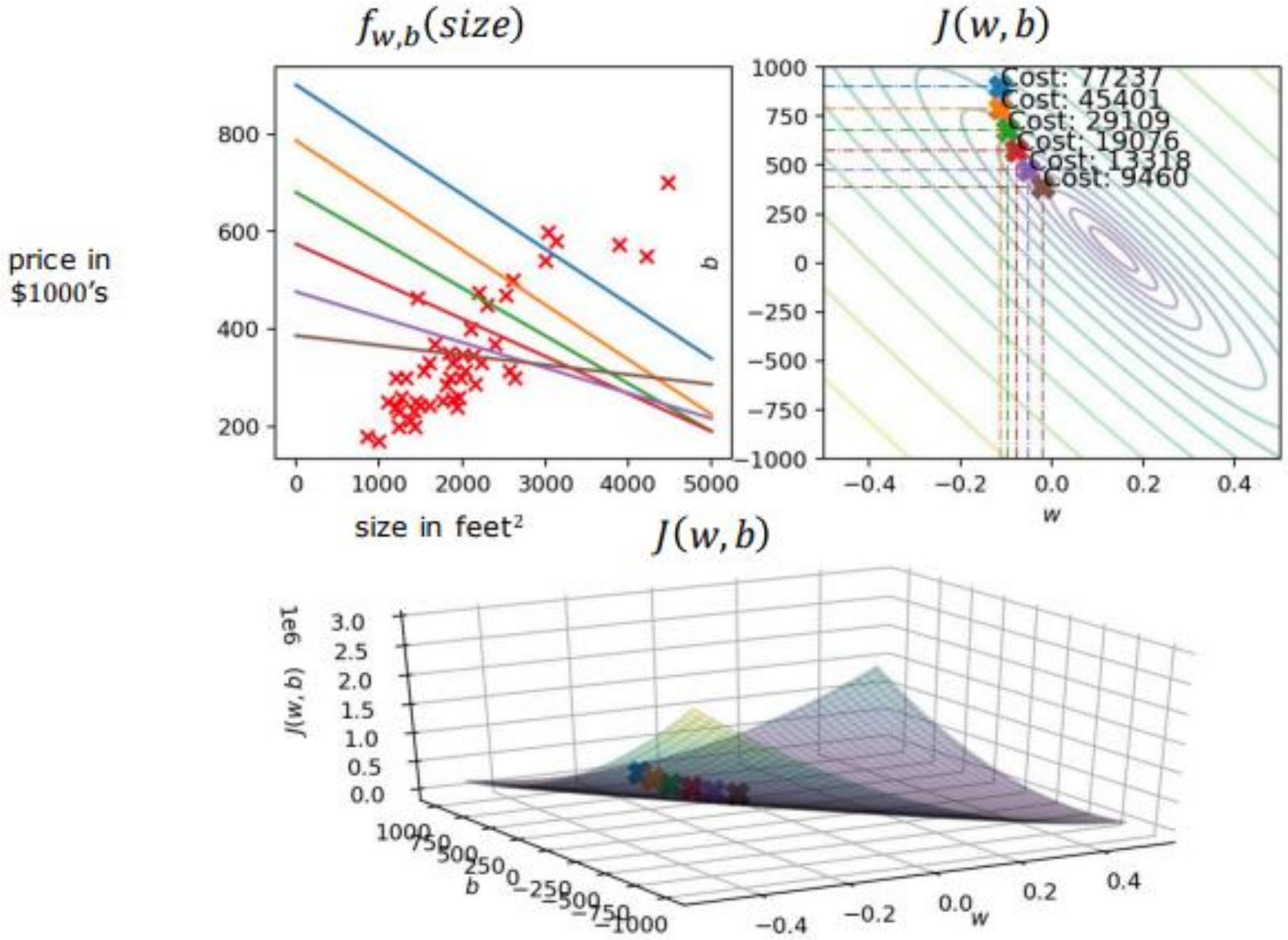

$f_{w,b}(size)$
$J(w,b)$
f(x) = -0.1 x + 900
Cost: 77237
b=900
w=-0.1
price in $1000's
size in feet²
$J(w,b)$
$J(w,b)$
b
w

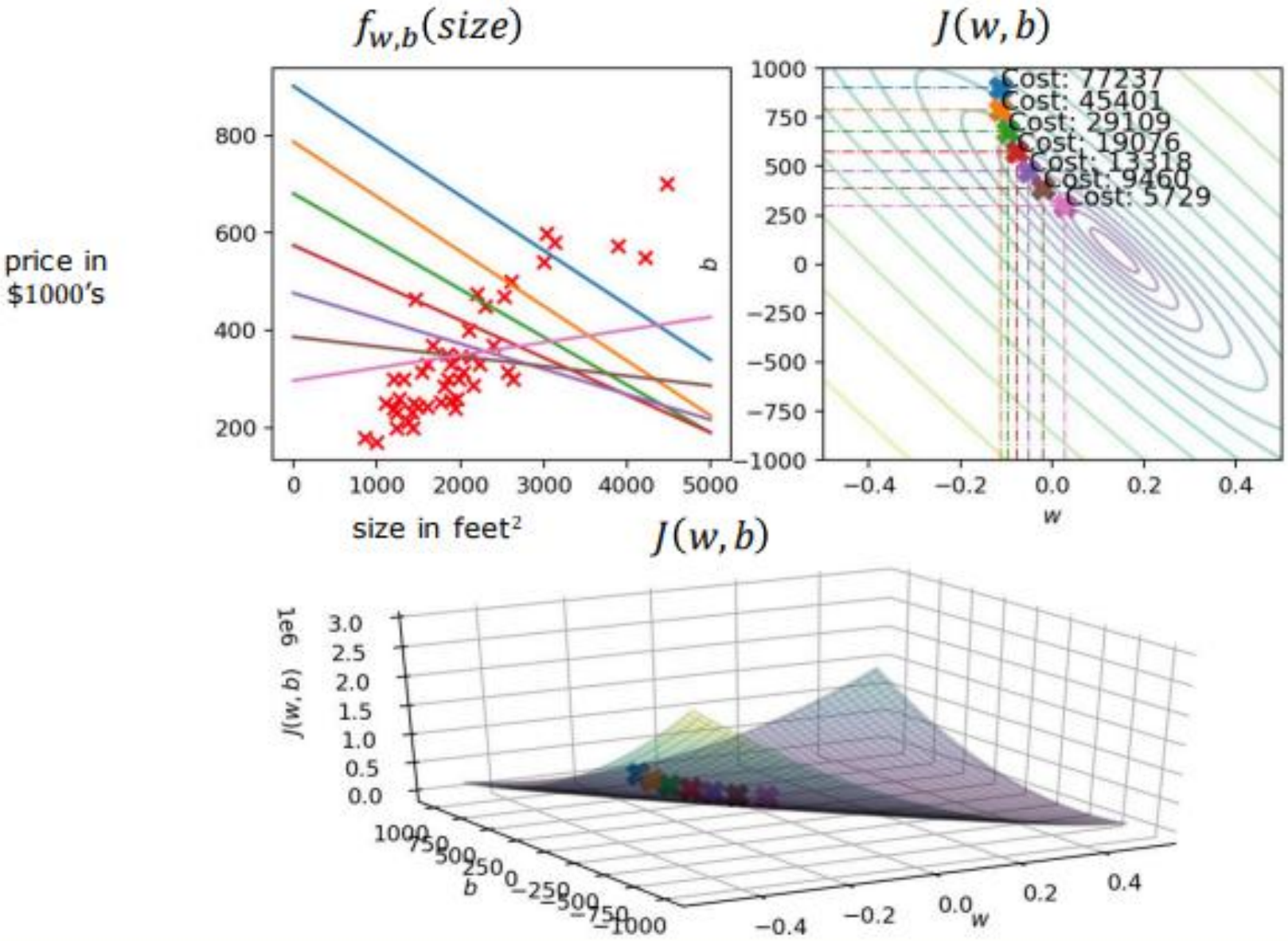

$f_{w,b}(size)$
price in $1000's
800
600
400
200
0
1000
2000
3000
4000
5000
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
1000
750
500
250
0
−250
−500
−750
−1000
−0.4
−0.2
0.0
0.2
0.4
w
b
$J(w,b)$
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$J(w,b)$

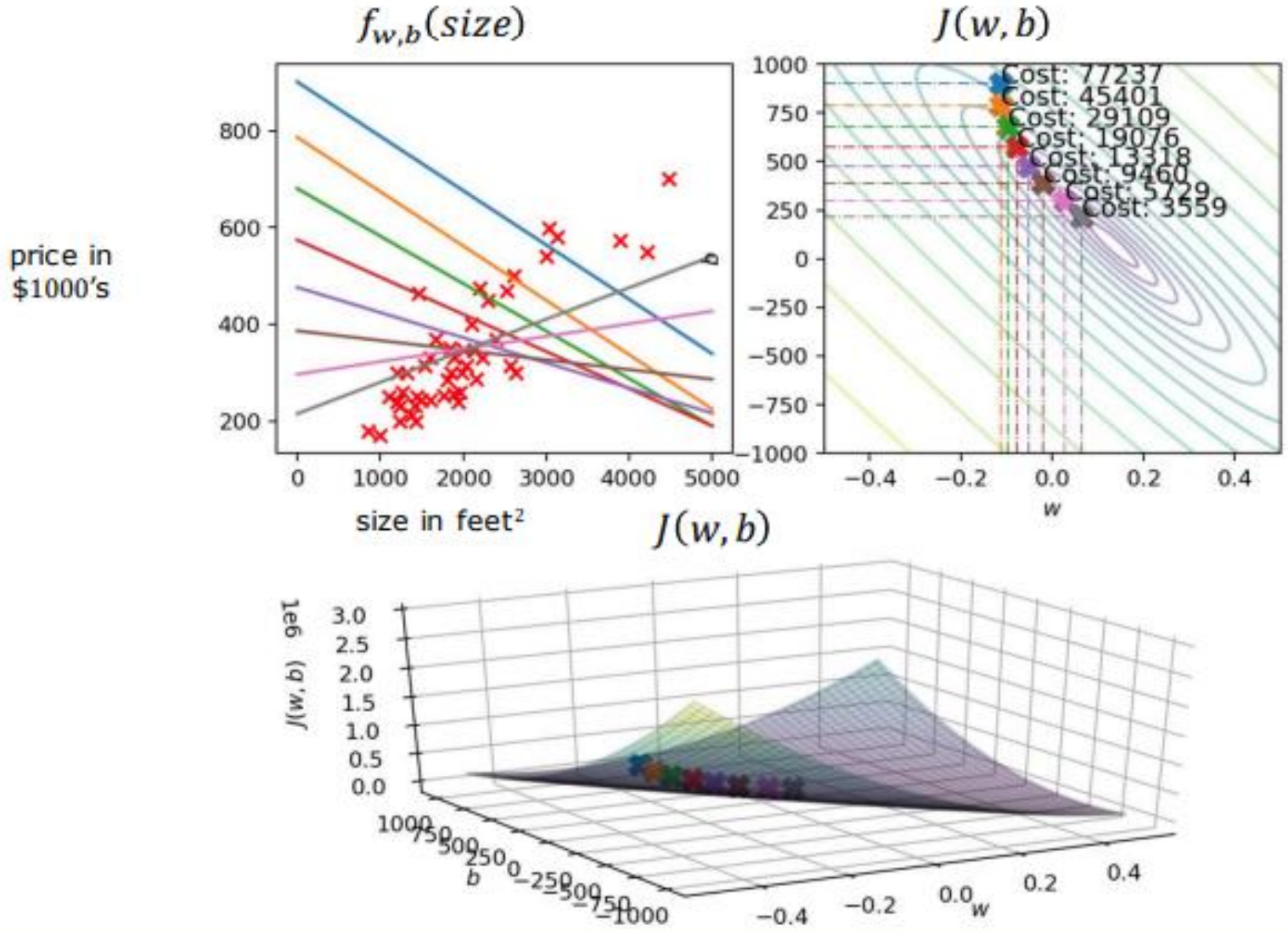

$f_{w,b}(size)$
800
600
400
200
0
1000
2000
3000
4000
5000
b
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
1000
750
500
250
0
−250
−500
−750
−1000
−0.4
−0.2
0.0
0.2
0.4
w
$J(w,b)$
1e6
$J(w,b)$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
b
w

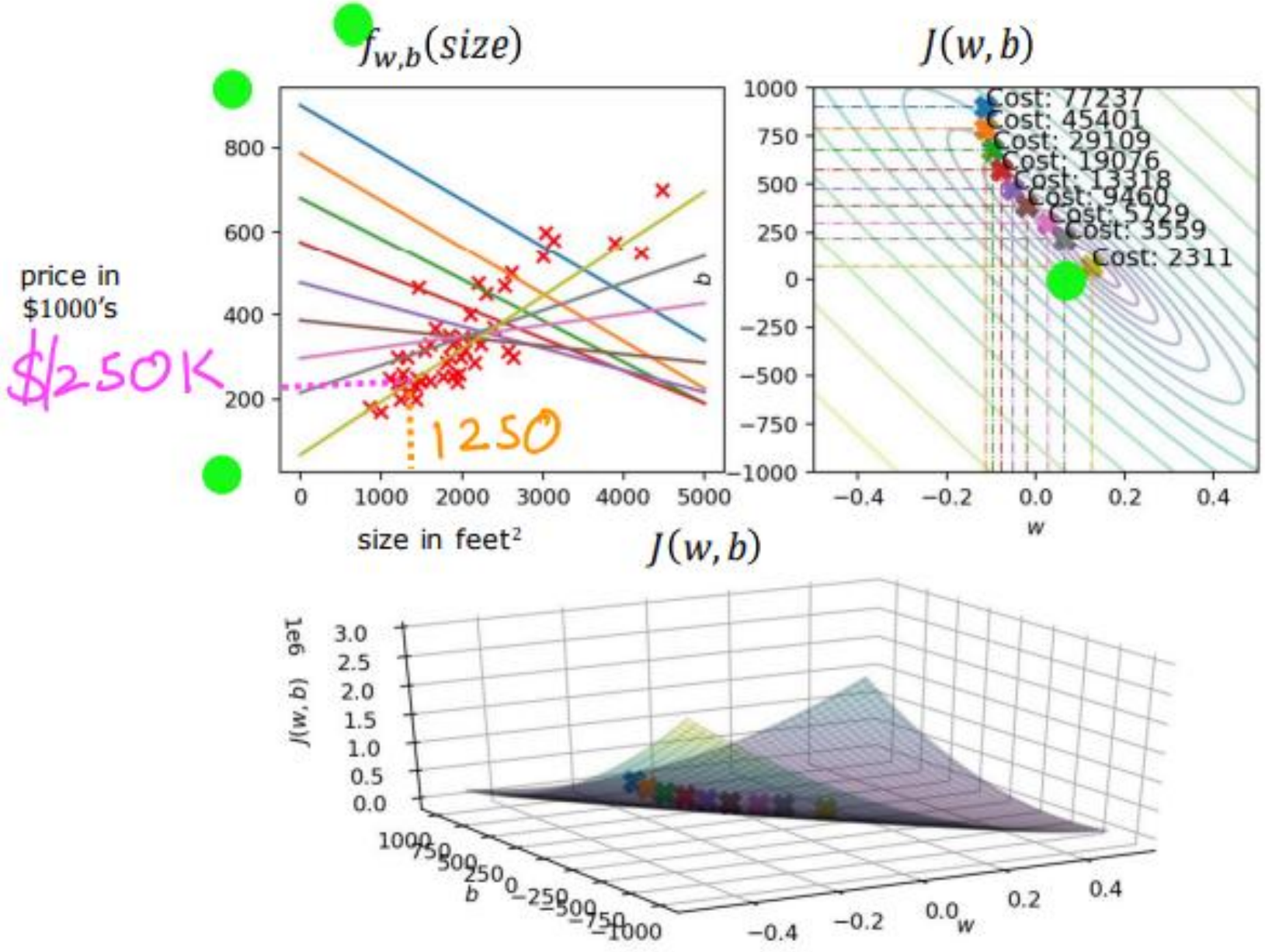

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
w
b
$J(w,b)$
1e6
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
w
b
$J(w,b)$
1e6
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
$J(w,b)$
1e6


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
$J(w,b)$
1e6


$f_{w,b}(size)$
price in $1000's
800
600
400
200
0
1000
2000
3000
4000
5000
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
1000
750
500
250
0
−250
−500
−750
−1000
−0.4
−0.2
0.0
0.2
0.4
w
$J(w,b)$
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
J(w, b)
b
w


$f_{w,b}(size)$
price in $1000's
$250K
1250
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
Cost: 2311
w
b
$J(w,b)$
1e6
J(w, b)

Stanford
ONLINE

DeepLearning.AI

Linear Regression with Multiple Variables

Multiple Features

Multiple features (variables)

one feature →

Size in feet² (x)	Price ($) in 1000's ($y$) ←
2104	400
1416	232
1534	315
852	178
...	...

$$f_{w,b}(x) = wx + b$$

Multiple features (variables)

Size in feet²	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
x_1	x_2	x_3	x_4	
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$j = 1 \ldots 4$

$n = 4$

$i = 2$

x_j = j^{th} feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \ 3 \ 2 \ 40]$

$x_3^{(2)} = 2$

Notation Recap:

- m: The number of training examples.
- n: The number of features.
- $x_j^{(i)}$: The value of the j-th feature for the i-th training example.
- $y^{(i)}$: The true value of the target variable for the i-th training example.
- $w_1, w_2, \ldots, w_n$: The coefficients (slopes) for each feature.
- b: The intercept.
- $\mathbf{X}$: The $m \times (n+1)$ feature matrix, where the first column consists of ones (to account for the intercept).
- $\mathbf{w}$: The vector of coefficients, including the intercept.

Linear Regression Model

The hypothesis (prediction) for the i-th training example is:

$$\hat{y}^{(i)} = b + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)}$$

In matrix notation, this can be written as:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

Where:

- $\hat{\mathbf{y}}$ is the $m \times 1$ vector of predictions.
- $\mathbf{X}$ is the $m \times (n+1)$ matrix, where the first column is all ones (to account for b), and the remaining columns contain the feature values.
- $\mathbf{w} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$ is the $(n+1) \times 1$ vector of coefficients, including the intercept.

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$$

Where:

- m is the number of training examples.
- $y^{(i)}$ is the true target value for the i-th training example.
- $\hat{y}^{(i)} = b + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)}$ is the predicted value using the model.
- $\mathbf{w} = [b, w_1, w_2, \ldots, w_n]^T$ is the vector of parameters (intercept and weights).

$$J(\mathbf{w}) = \frac{1}{2m} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Solving for $\mathbf{w}$, we get the closed-form solution:

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

For m training examples and n features, the matrix $\mathbf{X}$ is:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$

Where each row corresponds to a training example, and each column corresponds to a feature (plus a column of ones for the intercept).

Gradient Descent Steps

Step 1: Initialize parameters

Initialize $w_1, w_2, \ldots, w_n$ and b to small random values or zeros.

Step 2: Update parameters iteratively

For each iteration:

1. Compute the predicted value for all training examples:

$$\hat{y}^{(i)} = b + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)}$$

2. Compute the gradients:

$$\frac{\partial J(\mathbf{w})}{\partial b} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right)$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

Step 2: Update parameters iteratively

For each iteration:

1. Compute the predicted value for all training examples:

$$\hat{y}^{(i)} = b + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)}$$

2. Compute the gradients:

$$\frac{\partial J(\mathbf{w})}{\partial b} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right)$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

3. Update the parameters:

$$b := b - \alpha \cdot \frac{\partial J(\mathbf{w})}{\partial b}$$

$$w_j := w_j - \alpha \cdot \frac{\partial J(\mathbf{w})}{\partial w_j}$$

Full Algorithm Summary

1. **Initialize** parameters $\mathbf{w} = [b, w_1, w_2, \ldots, w_n]^T$ to small values (e.g., zeros).
2. **Repeat** until convergence:
    - Compute predictions: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$.
    - Compute the error: $\text{error} = \hat{\mathbf{y}} - \mathbf{y}$.
    - Compute the gradient: $\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{m}\mathbf{X}^T\left(\hat{\mathbf{y}} - \mathbf{y}\right)$.
    - Update the parameters: $\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w})$.
3. **Stop** when the change in cost function is below a threshold or after a maximum number of iterations.

Some Practical Tips

- Always normalize input features so that values in features roughly lies in same range.
  - Use normalization techniques like min-max normalization, z-score normalization, etc.
- Try various values of learning rate to check for faster convergence.
  - Try values like 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, ...