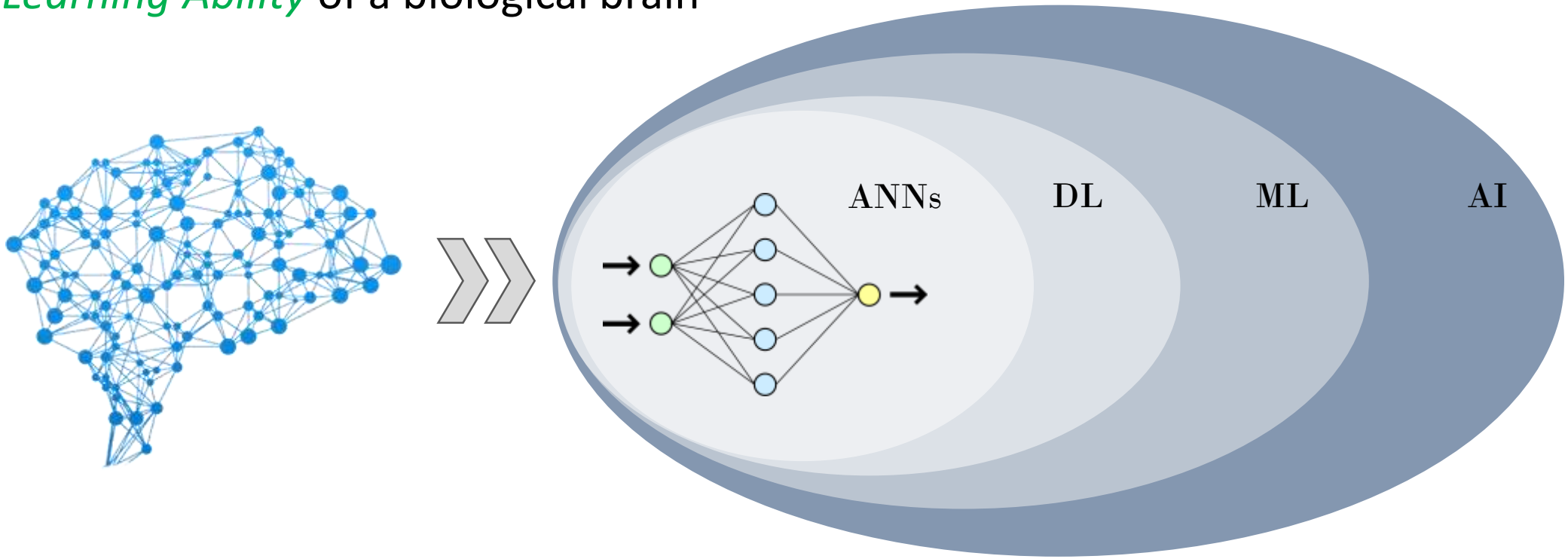
Learning from Examples

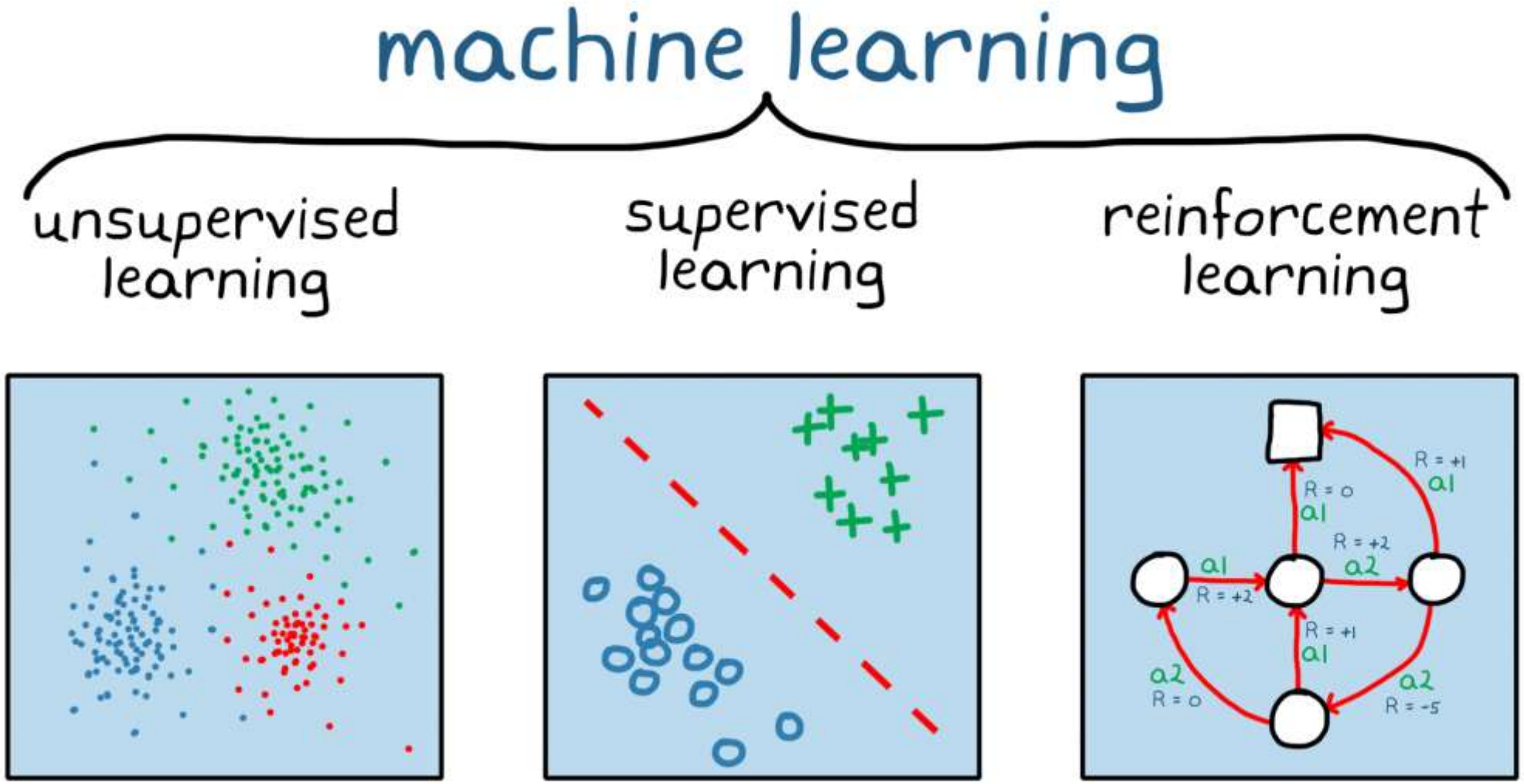
Module 6

Artificial Intelligence

- Compute: Solves an algorithm
- Computational system inspired by:
  - *Structure*
  - *Processing Method*
  - *Learning Ability* of a biological brain



Concept of Learning



Concept of Learning

- **Supervised Learning**

Dog

Cat

Concept of Learning … (contd.)

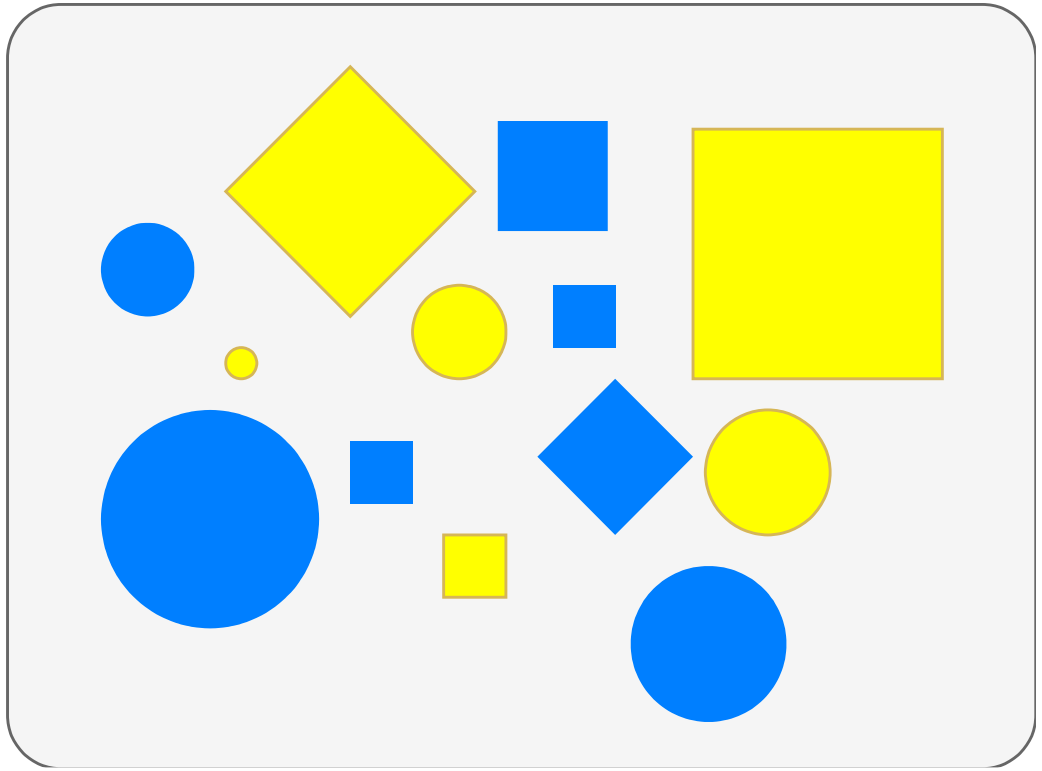
- **Supervised Learning**

Not Dog & Cat!

Concept of Learning … (contd.)

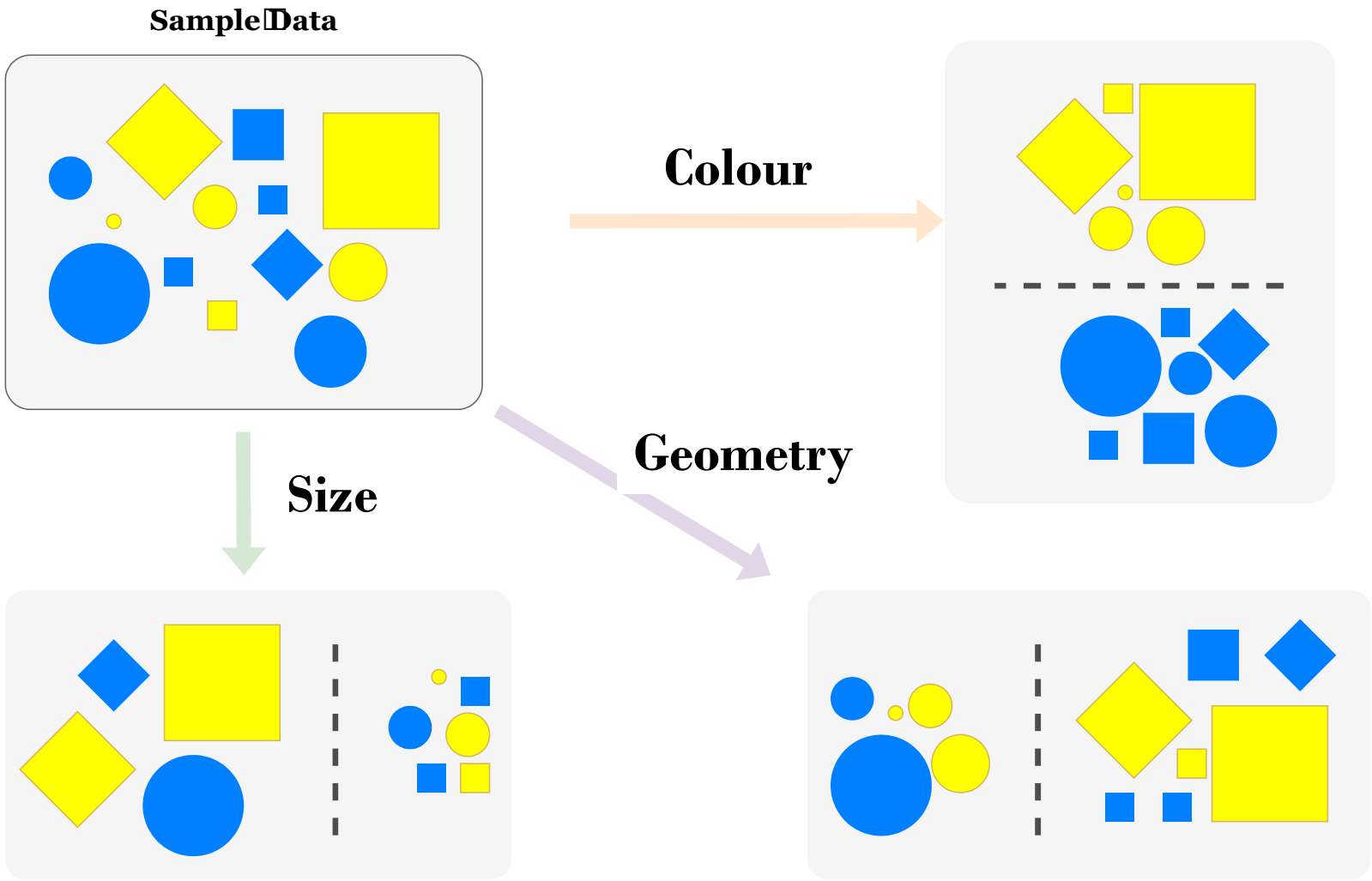
- Unsupervised Learning

Sample Data



Concept of Learning … (contd.)

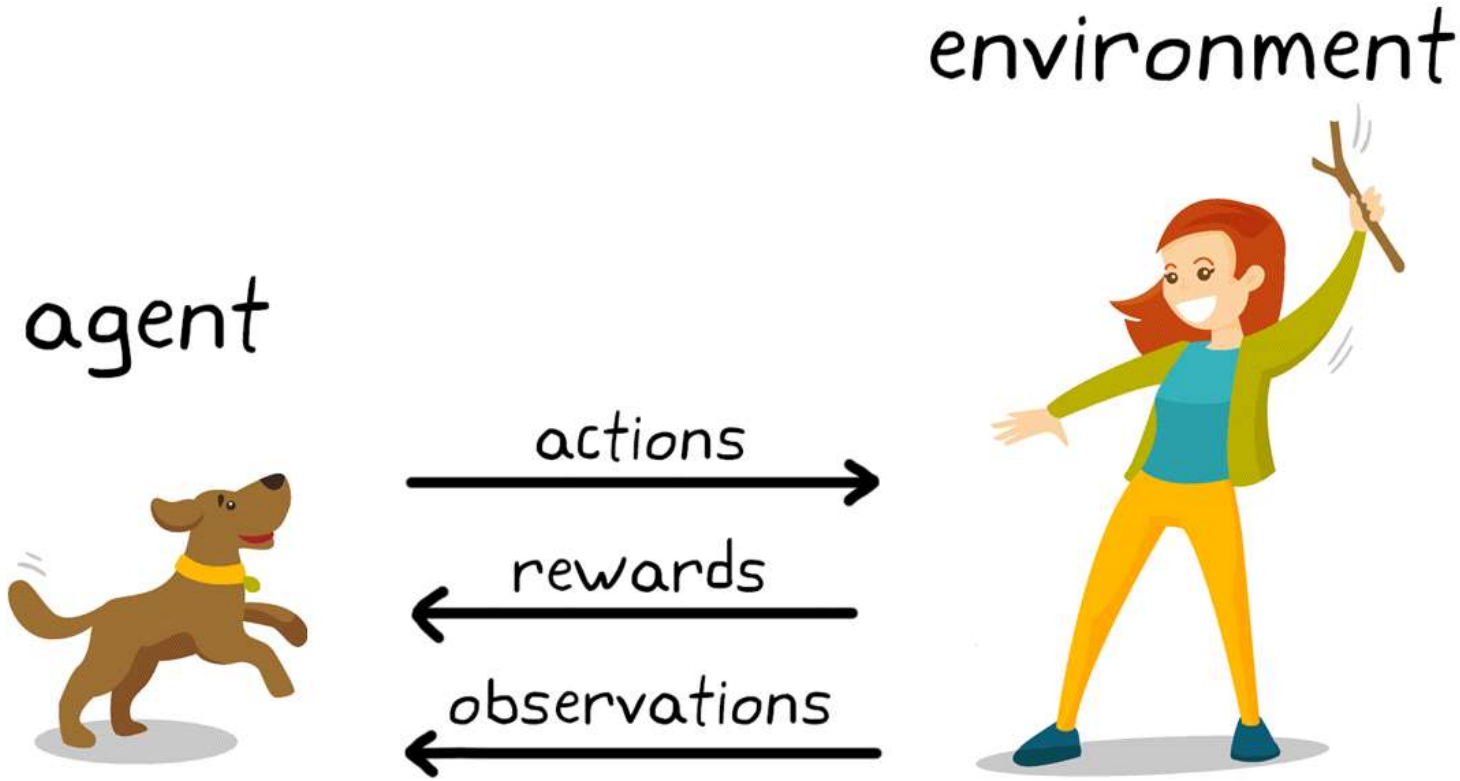
- Unsupervised Learning



Concept of Learning … (contd.)

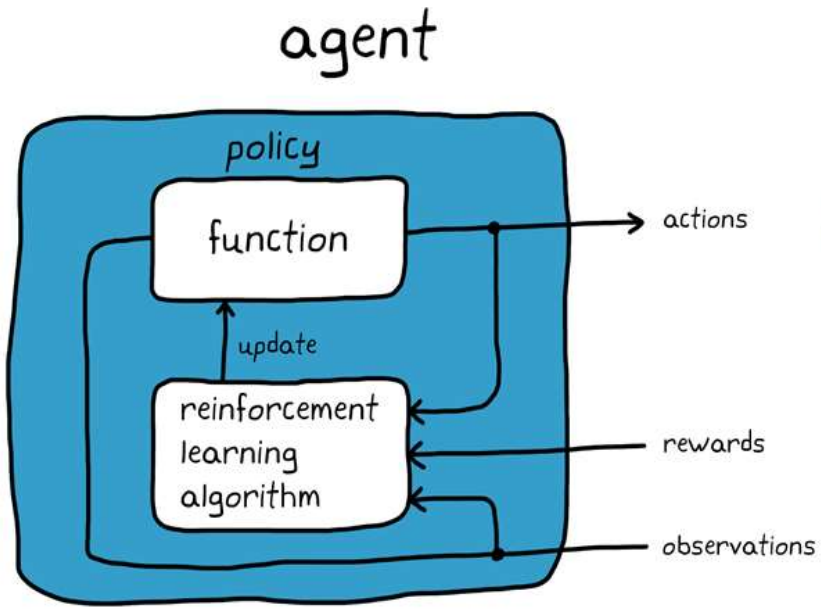
- Reinforcement Learning

- Learning to control the behavior of a system.
- It is neither supervised nor unsupervised.

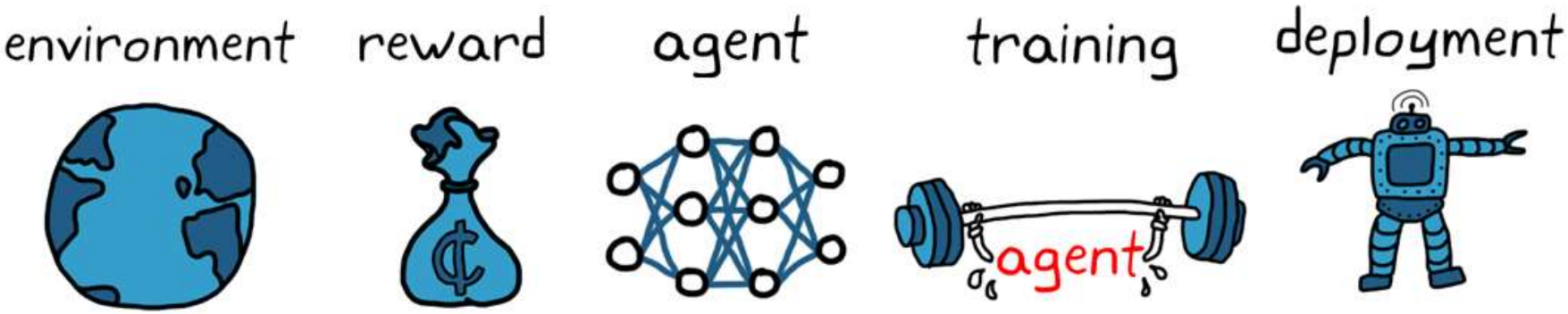


How Reinforcement Learning Works

- RL agent is able to ***perceive and interpret its environment***
- Take ***actions*** and ***learn through trial and error***
- Trials-false till it can ***maximize its reward*** and continue to develop further

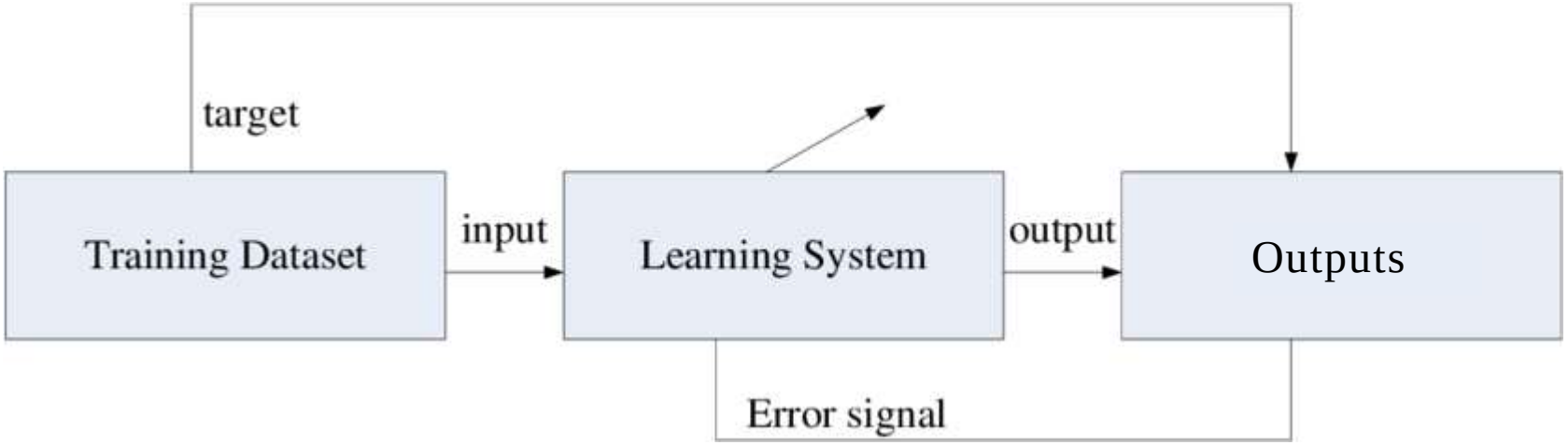


WORKFLOW:



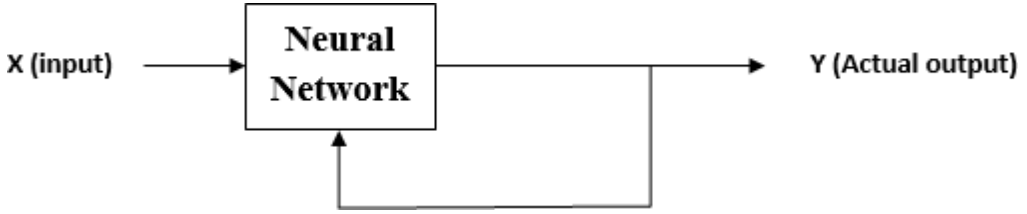
Leaning Process / Adjustment of Weights

- Learning in ANN,
  - The *method of modifying the weights of connections* between the neurons of a specified network.
- Supervised learning

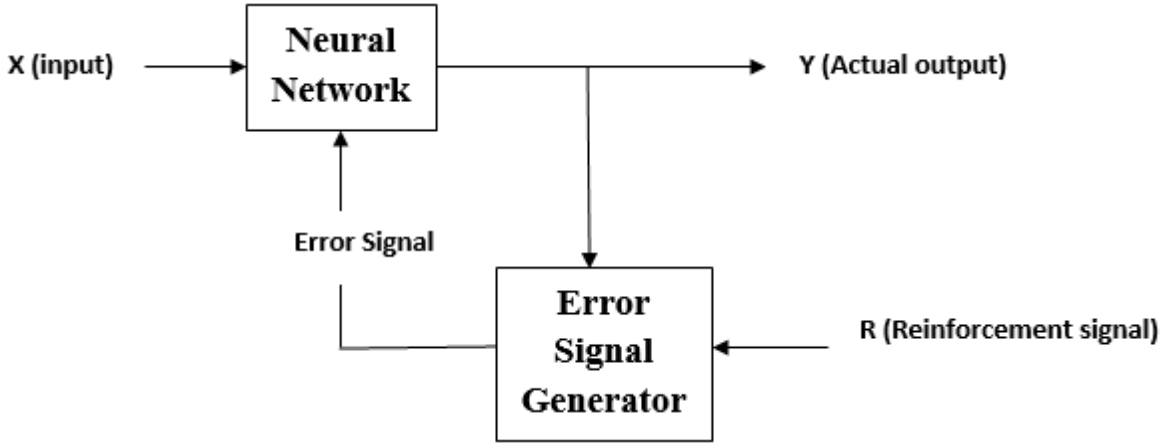


Leaning Process / Adjustment of Weights

- Unsupervised learning, and



- Reinforcement learning.



When to use Reinforcement Learning?

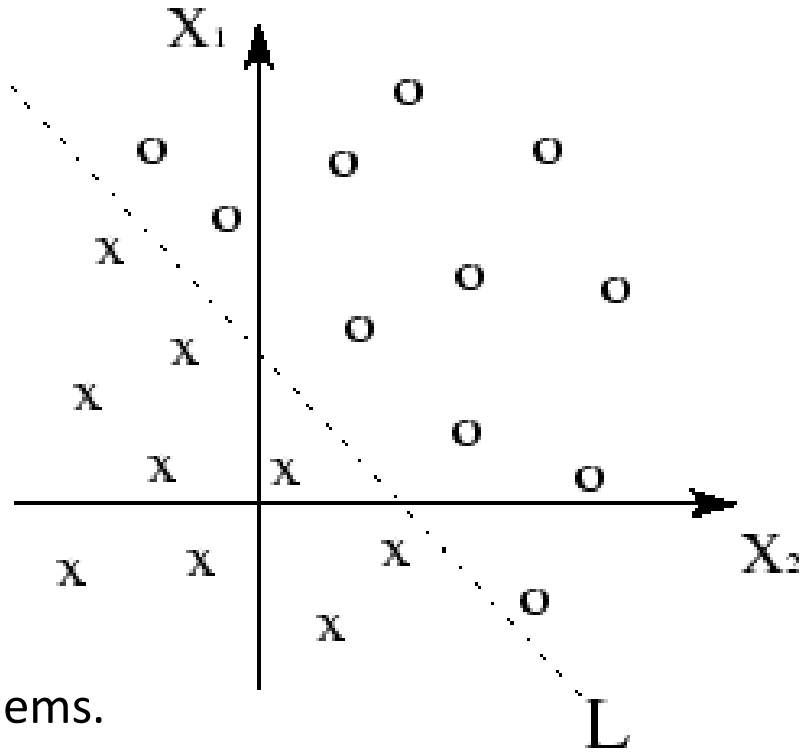
1. A model of the **environment is known**, but an **analytic solution is not available**;
2. Only a simulation model of the environment is given (the subject of simulation-based optimization)
3. The only way to collect information about the environment is to interact with it.

Linear Separability

Linear Separability

- Consider two-input patterns
  - X1 and X2 being classified into two classes
- Net input calculated to the output unit is given as

$$Net = b + \sum_{i=1}^{n} x_i w_i$$



- ANN **does not give an exact solution** for a nonlinear problem.
- However, it **provides an approximate solution** to nonlinear problems.

Linear separability is the concept wherein the separation of input space into regions is based on whether the network response is positive or negative.

7 Stages of Building a Machine Learning Model

Step 1: Problem Understanding

Business feasibility	Data feasibility	Implementation feasibility
Is there a clear problem definition? .... Is the organization willing to invest and change? .... Is there sufficient ROI or impact?	Do you have the required data that measures what you care about? .... Is there a sufficient quantity of data needed to train systems and do you have access to that data? .... Is the data of sufficient quality?	Do you have the required technology and skills? .... Can you execute the model as required in a timely manner? .... Does it make sense to use the model where you plan to use it?

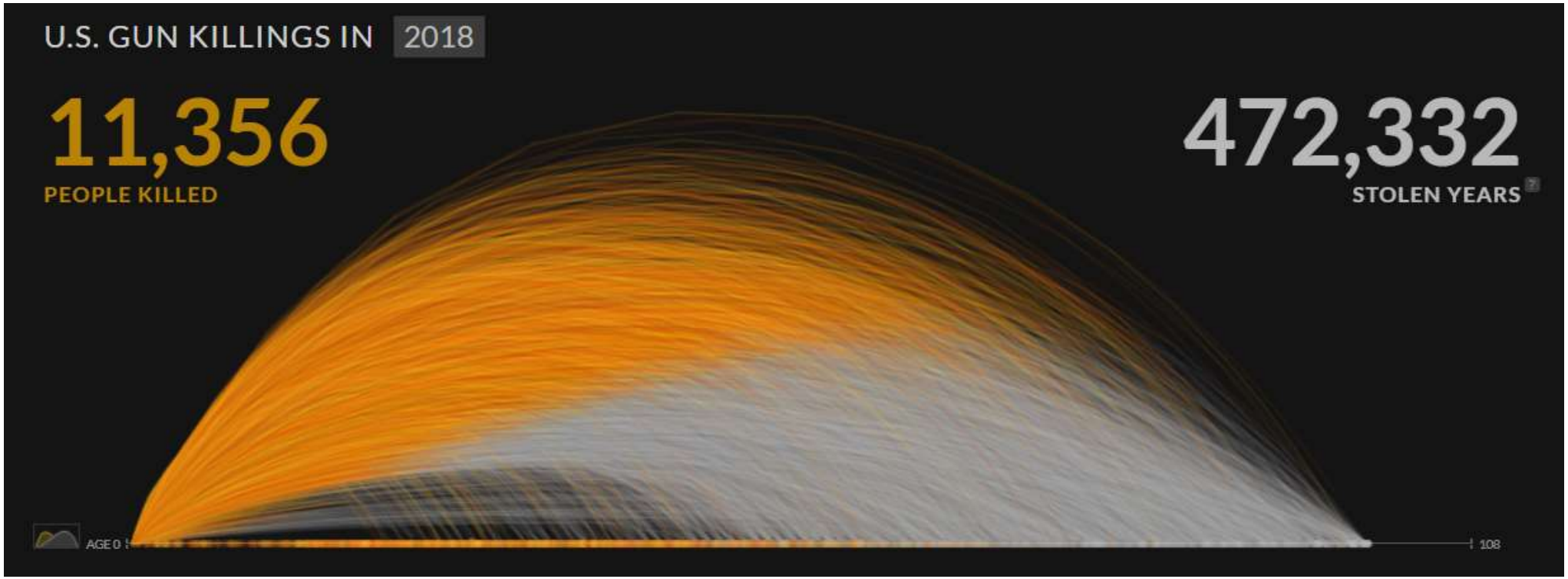
Step 2: Understand & Identify Data

Sources of training data	Data structure types	Data locations
■ Images ■ Videos ■ Emails ■ Documents, PDFs ■ Quantitative data from databases ■ Spreadsheets ■ Text content ■ Voice and audio	■ Structured data □ Queryable with a well-defined schema ■ Unstructured data □ No well-defined schema, not queryable ■ Semi-structured data □ Limited structure and queryability	■ On-premises databases ■ Data warehouses ■ Data Lakes ■ Cloud sources ■ Edge devices ■ Hybrid ■ Sensors/IoT

Step 2: Understand & Identify Data

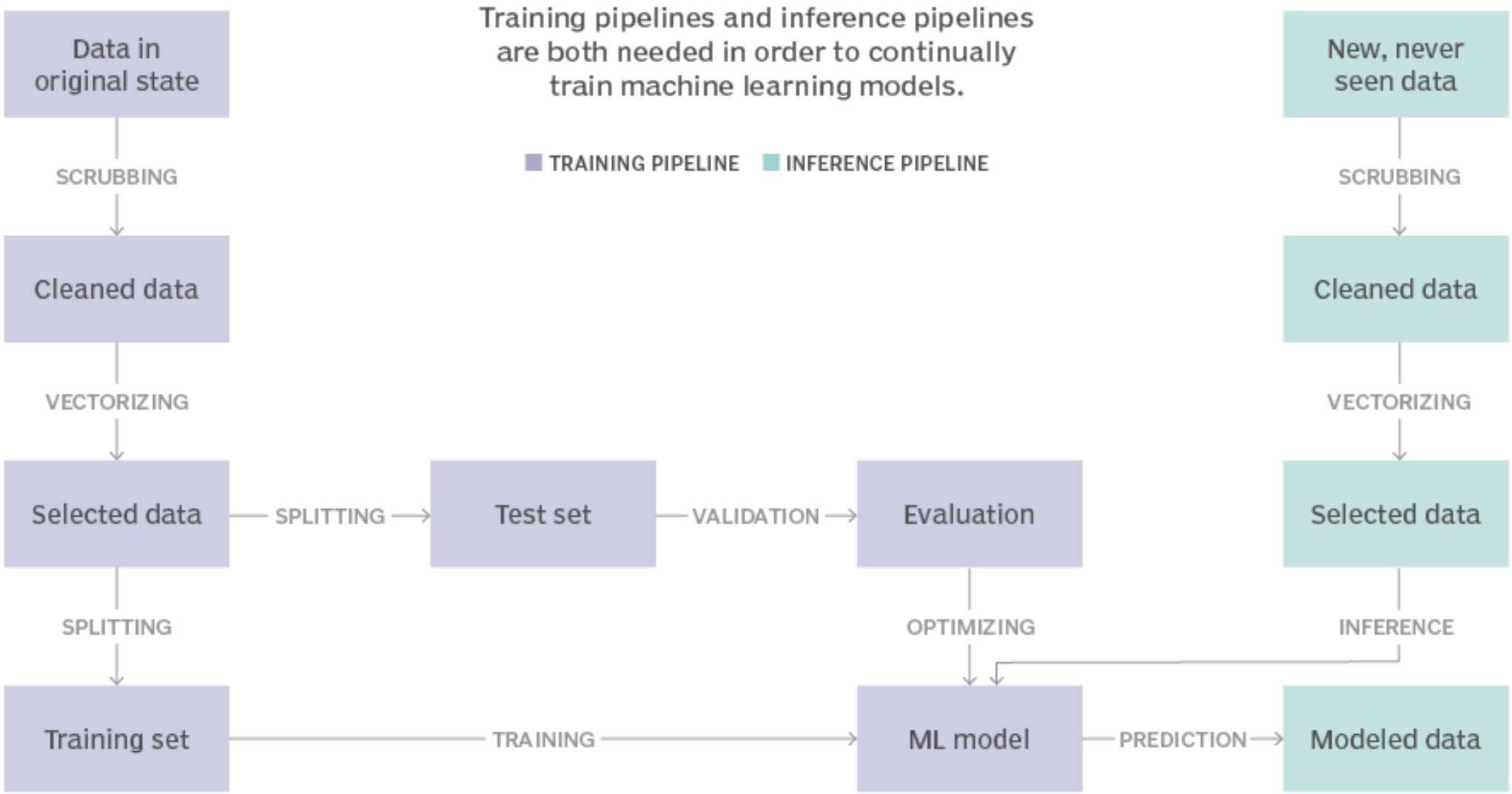
States with the Five *Lowest* Overall Gun Death Rates				States with the Five *Highest* Overall Gun Death Rates			
Rank	**State**	**Household Gun Ownership**	**Gun Death Rate per 100,000**	**Rank**	**State**	**Household Gun Ownership**	**Gun Death Rate per 100,000**
50	Massachusetts	12.5 percent	3.58	1	Alaska	54.6 percent	24.47
49	New York	20.5 percent	4.13	2	Mississippi	50.9 percent	23.86
48	New Jersey	15.2 percent	4.14	3	Wyoming	68.8 percent	22.98
47	Hawaii	10.4 percent	4.38	4	New Mexico	37.0 percent	22.46
46	Rhode Island	15.0 percent	4.53	5	Alabama	48.3 percent	21.94

Step 2: Understand & Identify Data



https://guns.periscopic.com/

Step 3: Collect & Prepare Data

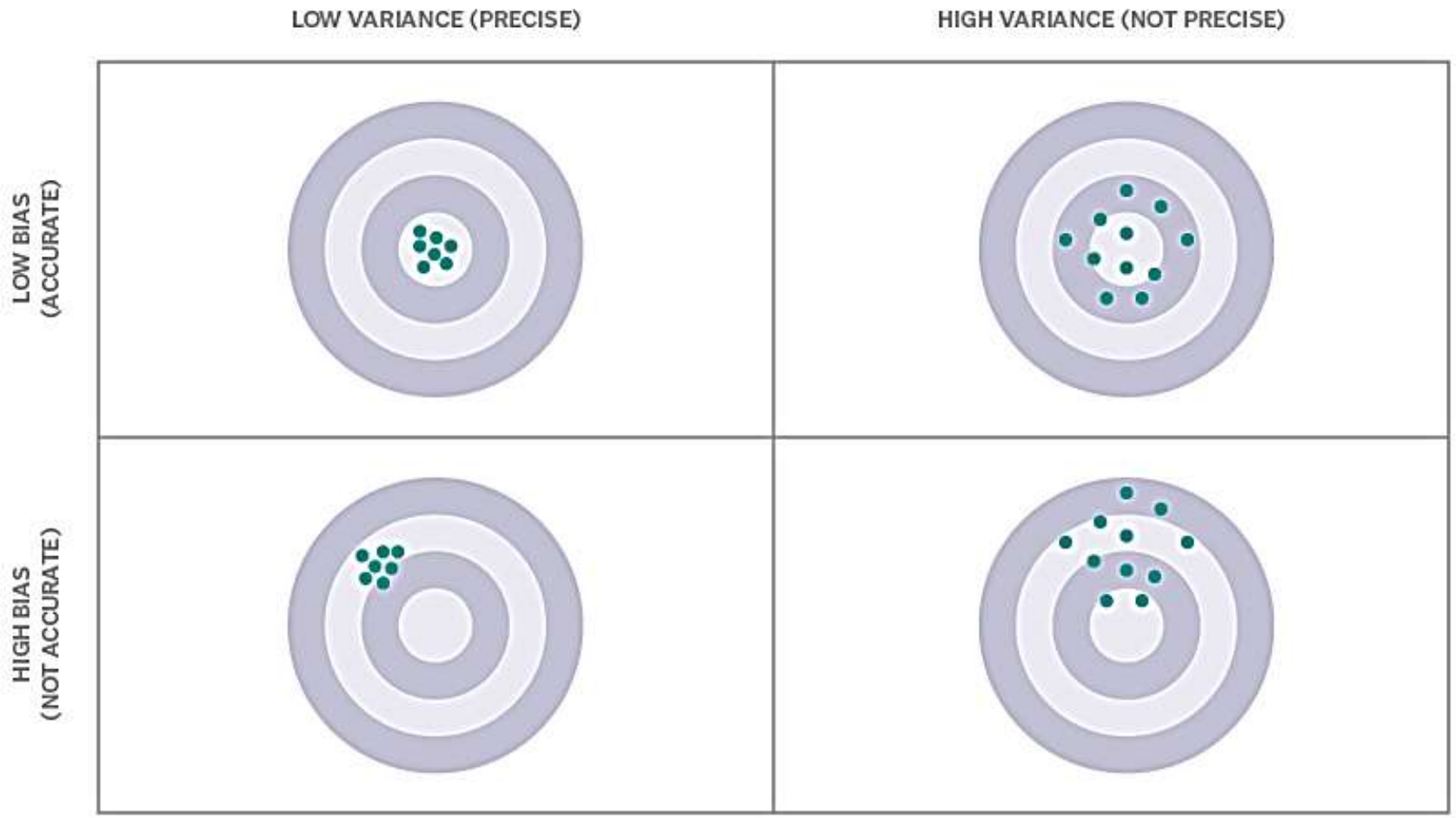


Step 4: Determine the model's features and train it

Machine learning			
Supervised learning		Unsupervised learning	Reinforcement learning
Classification	**Regression**	**Clustering**	Q-learning
Naive Bayes	Generalized linear models	K-means, fuzzy means	Policy gradient
Support vector machines	Logistic regression	Gaussian mixture	Trust region policy optimization
K-nearest neighborhood	Support vector regression, Gaussian process regression	Hidden Markov model	Proximal policy optimization
Decision trees, random forest	Ensemble methods	Spectral clustering	Hindsight experience replay
Neural network	Neural network	Neural network	Deep Q neural network

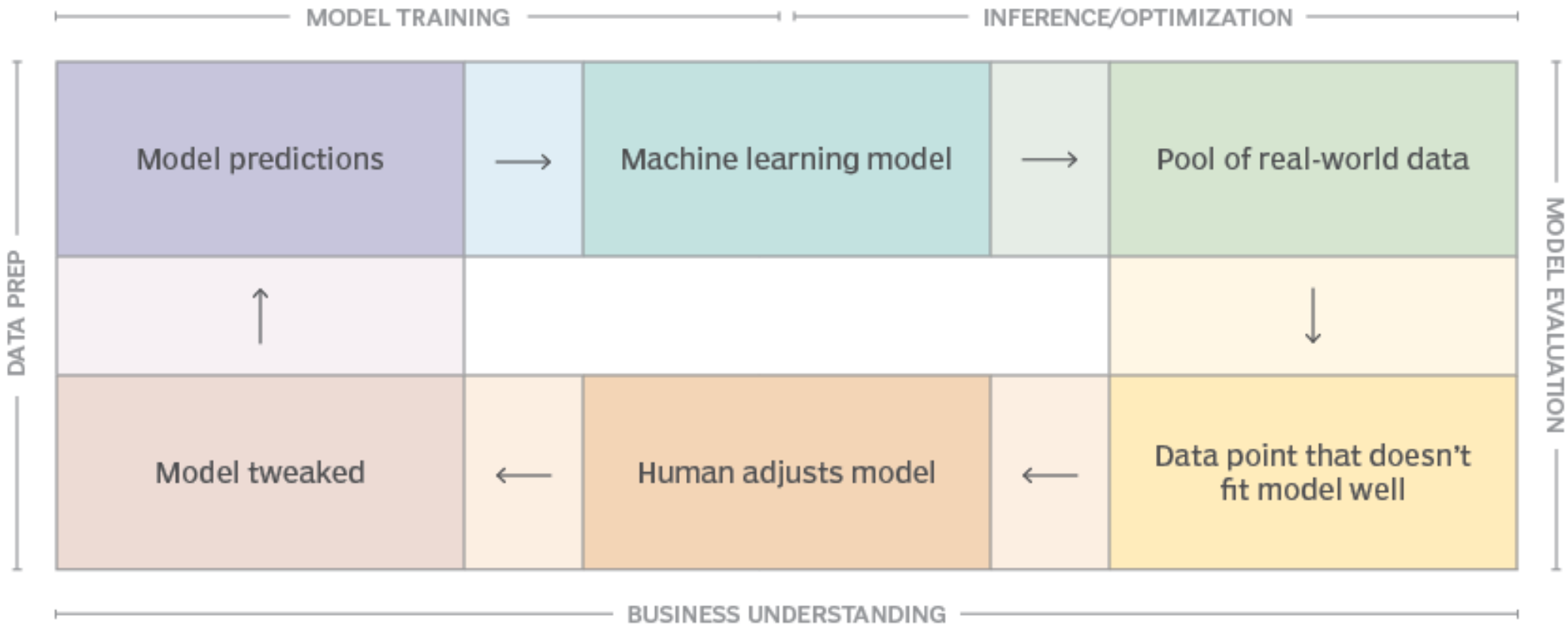
Step 5: Evaluate the model's performance & establish benchmarks

Bias refers to situations where the models make erroneous assumptions. Variance refers to the specificity the system has to trained data and its ability to properly classify new data.



Step 6: Determine the model's features and train it
Step 7: Iterate and adjust the model

Many real-world environments will require continuous model iteration and updates. This image illustrates a predictable model iteration pathway that guarantees that the models continue to provide valuable, reliable and desirable results.

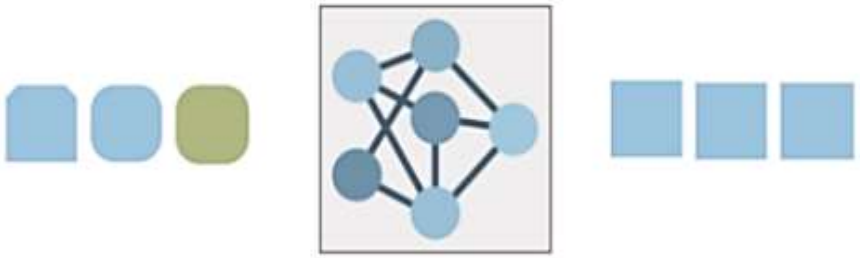


Performance Metrics

Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

Classifier that always predicts label blue



yields prediction accuracy of 90%

Accuracy doesn't always give the correct insight about your trained model			
Accuracy:	%age correct prediction	Correct prediction over total predictions	One value for entire network
Precision:	Exactness of model	From the detected cats, how many were actually cats	Each class/label has a value
Recall:	Completeness of model	Correctly detected cats over total cats	Each class/label has a value
F1 Score:	Combines Precision/Recall	Harmonic mean of Precision and Recall	Each class/label has a value

Confusion Matrix

- **Confusion Matrix** is a tool to determine the performance of classifier.
- It contains information about **actual and predicted classifications**.

		Actual Labels	
		1	0
Predicted Labels	1	True Positive	False Positive
	0	False Negative	True Negative

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F1 score} = 2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{Recall, Sensitivity, True +ve rate} = \frac{TP}{TP + FN}$$

$$\frac{TN}{TN + FN} = \text{Specificity}$$

$$\frac{FN}{TN + FN} = \text{False +ve rate}$$

(Is your prediction correct?) (What did you predict)

True Negative

(Your prediction is correct) (You predicted 0)

- **TP**- Positive class correctly identified as positive
- **FN**-Positive class incorrectly identified as negative
- **FP**-Negative class incorrectly identified as positive
- **TN**-Negative class correctly identified as negative

Confusion Matrix

- **Accuracy:**
  Proportion of the total number of predictions that are correct.

$$Accuracy = \frac{45 + 30}{45 + 20 + 5 + 30} = 75\%$$

of examples are correctly classified by the classifier.

- **Sensitivity / Recall:**
  Measure of Positive examples labeled as positive by classifier.

$$Sensitivity = \frac{45}{45 + 20} = 69.23\%$$

of spam emails are correctly classified and excluded from all non-spam emails

		Predicted Class	
		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Confusion Matrix

- **Specificity:**
  Measure of negative examples labeled as negative by classifier.

$$Specificity = \frac{30}{30+5} = 85.71\%$$

of non-spam emails are accurately classified

- **Precision**
  Correctness achieved in positive prediction-Total number of correctly classified positive examples and the total number of predicted positive examples.

$$Precision = \frac{45}{45+5} = 90\%$$

of examples are classified as spam are actually spam.

Predicted Class

Actual Class	Spam	Non-Spam
Spam	TP=45	FN=20
Non-Spam	FP=5	TN=30

Confusion Matrix

- **F1 Score :**
  Weighted average of the recall (sensitivity) and precision

$$F1 - score \;=\; 2 * \frac{90 * 69.23}{90 + 69.23} \;=\; 78.26\%$$

of harmonic mean of precision and recall.

		Predicted Class	
		Spam	**Non-Spam**
Actual Class	**Spam**	TP=45	FN=20
	Non-Spam	FP=5	TN=30

Confusion Matrix

Actual Labels

$$\text{Accuracy} = \frac{30+20+1}{30+2+1+1+20+3+4+4+1} = 77.2\%$$

Predicted Labels

30	2	1
1	20	3
4	4	1

$$\text{Precision for Panda} = \frac{30}{30+2+1} = 0.91$$

$$\text{Precision for Polar Bear} = \frac{20}{20+3+1} = 0.83$$

$$\text{Precision for Bear} = \frac{1}{1+4+2} = 0.14$$

$$\text{Recall for Panda} = \frac{30}{30+1+4} = 0.85$$

$$\text{Recall for Polar B} = \frac{20}{20+4+2} = 0.77$$

$$\text{Recall for Bear} = \frac{1}{1+3+1} = 0.2$$

$$f1 = 2 * \frac{0.91 * 0.85}{0.91 + 0.85} = 0.88$$

$$f1 = 2 * \frac{0.83 * 0.77}{0.83 + 0.77} = 0.8$$

$$f1 = 2 * \frac{0.14 * 0.2}{0.14 + 0.2} = 0.16$$

Even though the accuracy is 77% but the model will perform very poorly on test bears

Solutions to improve model's accuracy

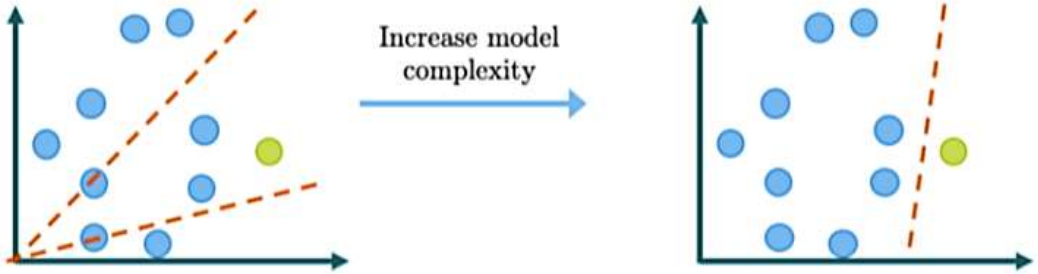
1. **Data Replication**
   Replicate the available data until the number of samples are comparable

2. **Synthetic Data**
   Images: Rotate, dilate, crop, add noise to existing input images and create new data

3. **Modified Loss**
   Modify the loss to reflect greater error when misclassifying smaller sample set

$$loss = a * loss_{green} + b * loss_{blue} \qquad a > b$$

4. **Change the algorithm**
   Increase the model/algorithm complexity so that the two classes are perfectly separable (Con: Overfitting)



No straight line (y=ax) passing through origin can perfectly separate data.
Best solution: line y=0, predict all labels blue

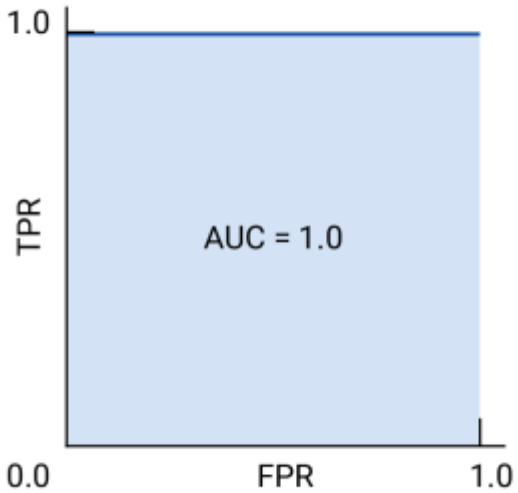
Straight line (y=ax+b) can perfectly separate data. Hence yielding an accuracy of 100%.
Green class will no longer be predicted as blue

ROC

- ROC curve is a **visual representation** of **model performance** across **all thresholds**.
- Graphical plot that illustrates the ***diagnostic ability*** of a ***binary classifier system*** as its ***discrimination threshold*** is varied.

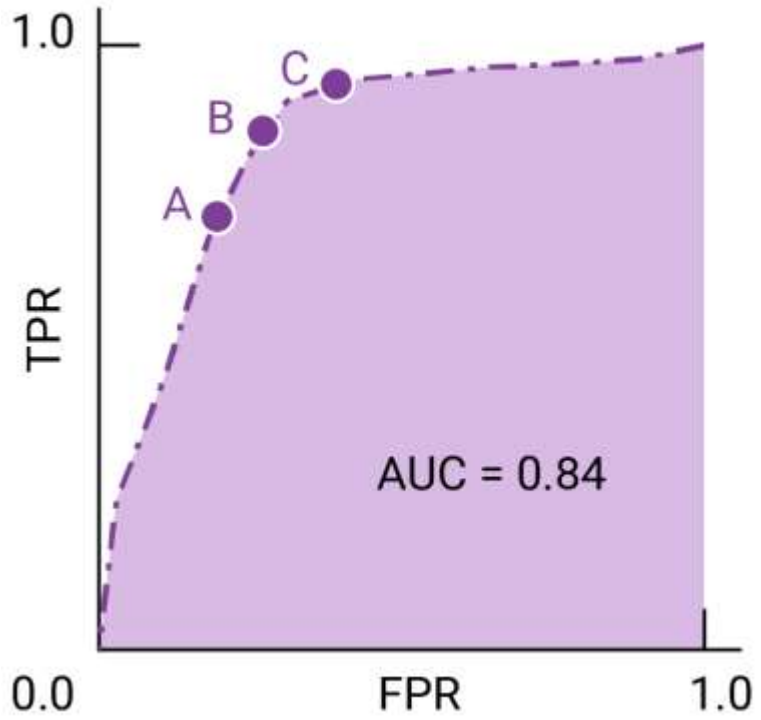
- ROC curve ***summarizes the performance*** by ***combining confusion matrices*** at all ***threshold values***.
  - *ROC curve is drawn by calculating the **true positive rate (TPR)** and **false positive rate (FPR)** at **every possible threshold** (in practice, at selected intervals), then **graphing TPR over FPR**.*

- **Perfect Model**, which at **some threshold** has a **TPR of 1.0** and a **FPR of 0.0**, can be represented by either a point at **(0, 1)** if all other thresholds are ignored, or by the following:

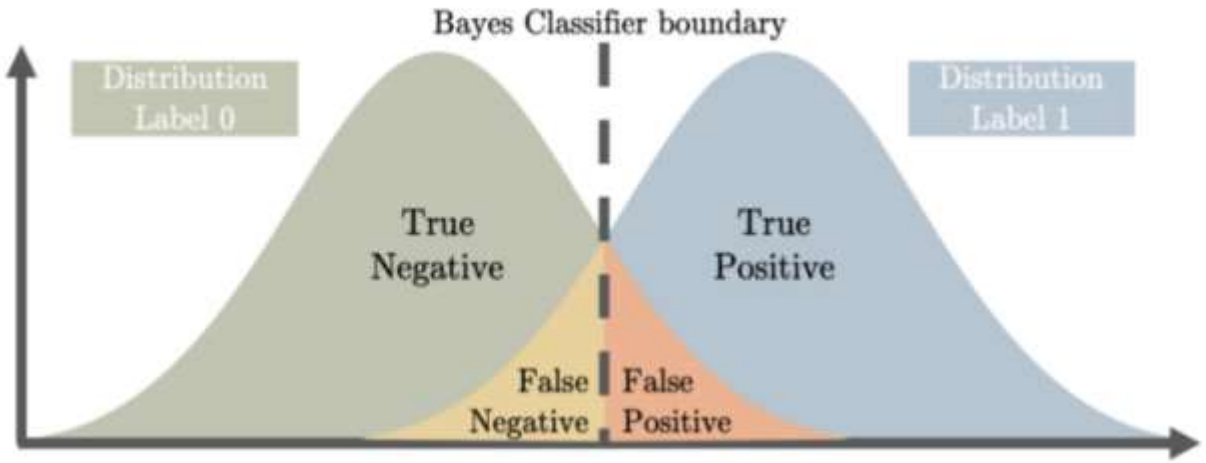


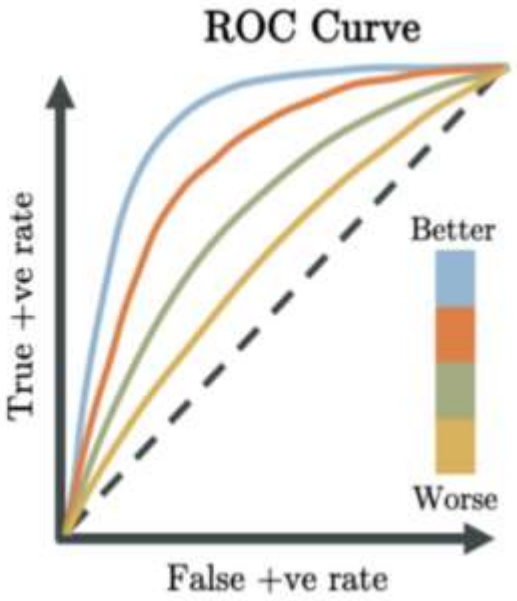
AUC

- AUC measures the entire **two-dimensional area underneath the entire ROC curve** from 0 to 1.
- Model whose **predictions** are **100% wrong** has an **AUC of 0.0**;
- One whose **predictions are 100% correct** has an **AUC of 1.0**.
- **AUC of a classifier is equal to the probability** that the *classifier will rank a randomly chosen positive example higher than a randomly chosen negative example*.
- **AUC** provides an **aggregate measure of performance across** all **possible classification thresholds**.

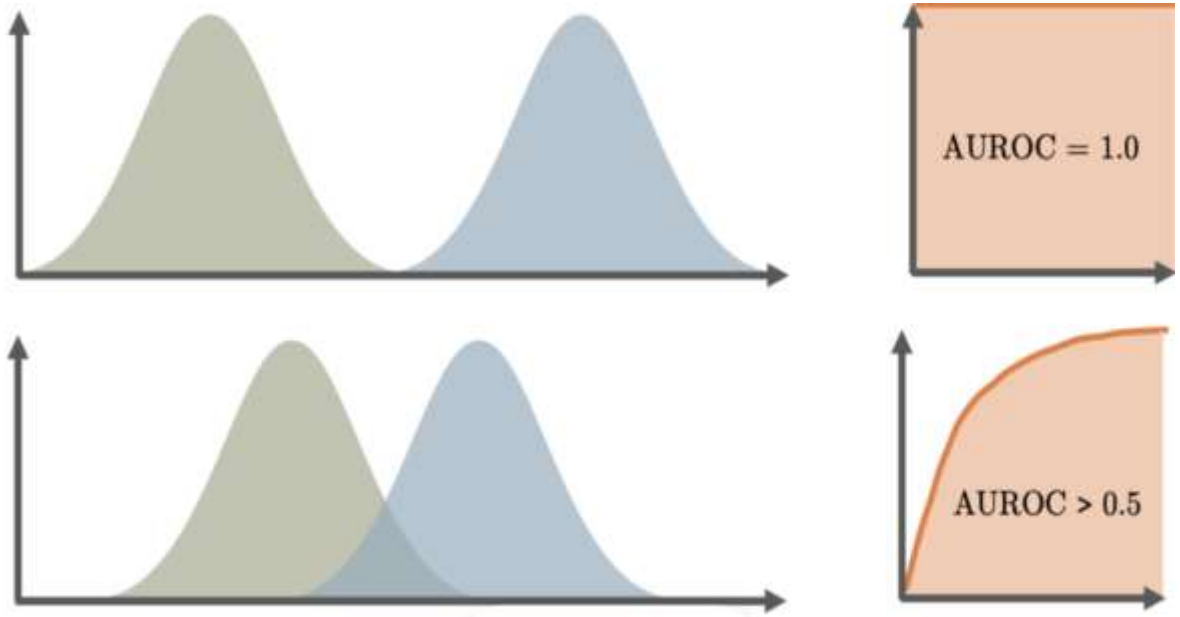


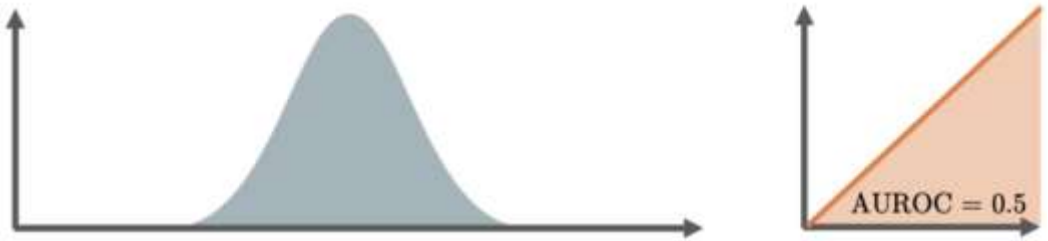
Performance Metrics





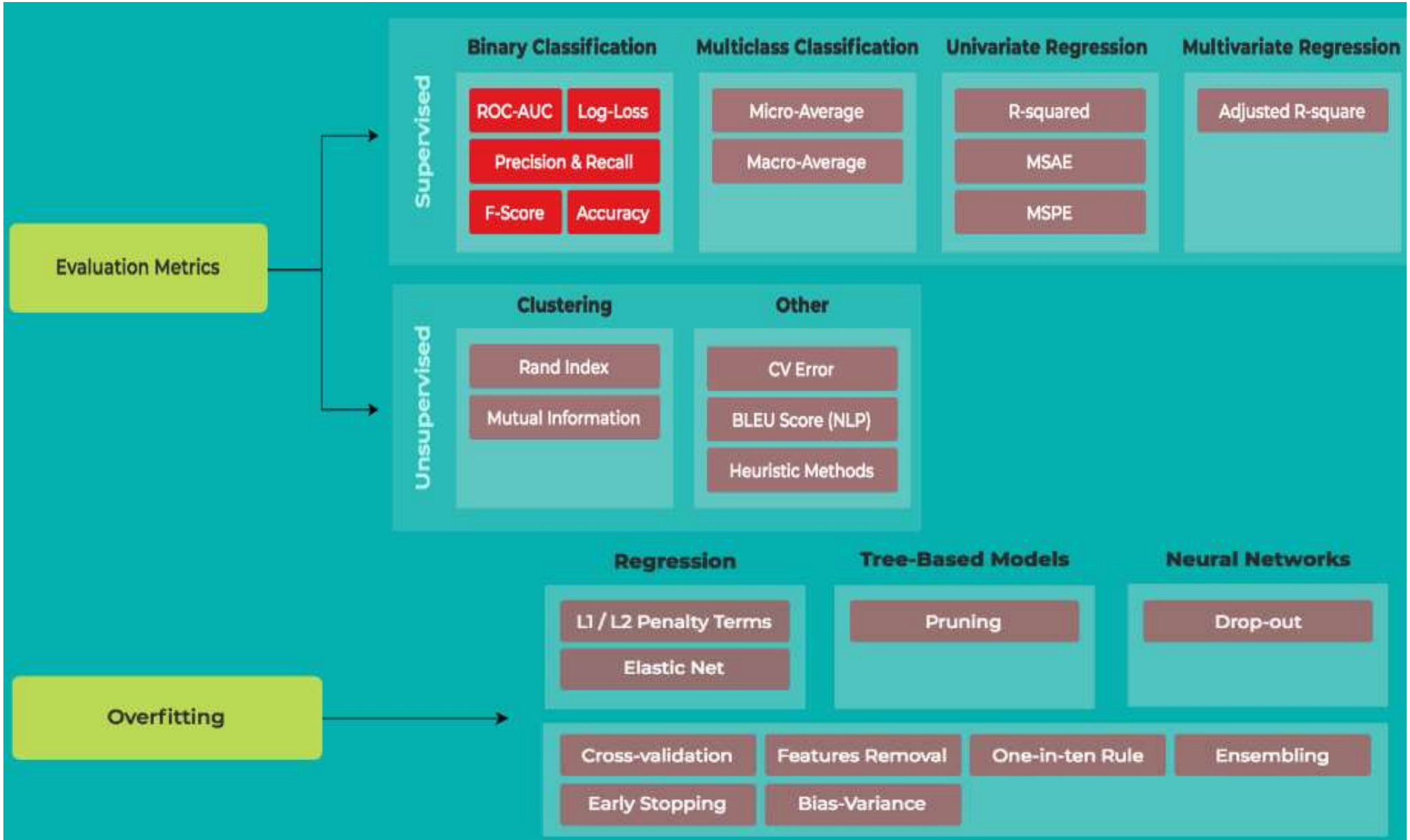
Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives





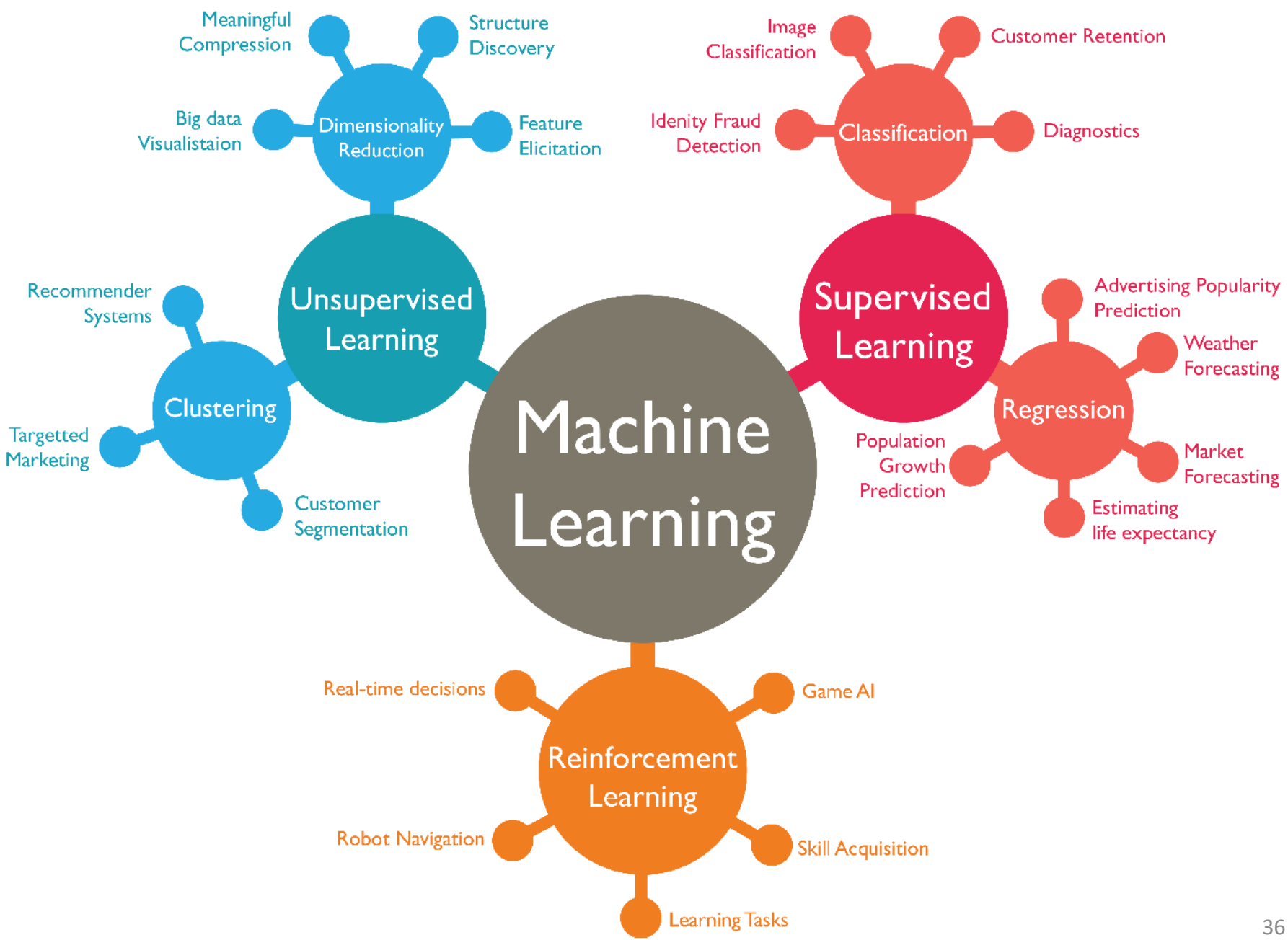
AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

Performance Metrics | Summary



Machine Learning Algorithms

Machine Learning Algorithms



Bayes' Theorem| Example

- *Given:*
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- *If a patient has stiff neck, what's the probability he/she has meningitis?*

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

Naïve Bayes Classification Model

- One of the *simple* and *most effective classification algorithms*
  - **Fast** machine learning models that can **make quick predictions**.
- **A Probabilistic Classifier: Predicts on the basis of the probability of an object**.
  - **Naïve:** It is called Naïve because it *assumes that the occurrence of a certain feature is* ***independent*** *of the occurrence of other features*.
    - Such as if the fruit is identified on the bases of color, shape, and taste
    - Then red, spherical, and sweet fruit is recognized as an apple.
    - Hence each feature individually contributes to identify that it is an apple without depending on each other.
  - **Bayes:** It is called Bayes because it depends on the principle of **Bayes' Theorem**

Naïve Bayes Classification Model

- ***Bayes' theorem*** is also known as ***Bayes' Rule*** or ***Bayes' law***, which is used to **determine the probability** of a **hypothesis** with **prior knowledge**.
- It depends on the **conditional probability**.
- *Formula for Bayes' theorem is given as:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  - Where,
    - ***P(A) is Prior Probability***: Probability of hypothesis before observing the evidence.
    - ***P(B) is Marginal Probability***: Probability of Evidence.
    - ***P(A|B) is Posterior probability***: Probability of hypothesis A on the observed event B.
    - ***P(B|A) is Likelihood probability***: Probability of the evidence given that the probability of a hypothesis is true.

Naïve Bayes Classification Model

The Bayes Naive classifier selects the most likely classification V_{nb} given the attribute values $a_1, a_2, \ldots a_n$. This results in:

$$V_{nb} = \text{argmax}_{v_j \in V} P(v_j) \prod P(a_i|v_j) \tag{1}$$

We generally estimate $P(a_i|v_j)$ using m-estimates:

$$P(a_i|v_j) = \frac{n_c + mp}{n + m} \tag{2}$$

where:

$n =$	the number of training examples for which $v = v_j$
$n_c =$	number of examples for which $v = v_j$ and $a = a_i$
$p =$	a priori estimate for $P(a_i|v_j)$
$m =$	the equivalent sample size

Naïve Bayes Classification Model : Example

- **If the weather is sunny, then the Player should play or not?**

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Naïve Bayes Classification Model : Example

- **Step 1 :**
- ***Frequency table of Weather Conditions***

Weather	YES	NO
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	**10**	**4**

- **Step 2 :**
- ***Likelihood table weather condition:***

Weather	NO	YES	
Overcast	0	5	**5/14 =0.35**
Rainy	2	2	**4/14 =0.29**
Sunny	2	3	**5/14 =0.35**
Total	**4/14 = 0.29**	**10/14=0.71**	

Naïve Bayes Classification Model : Example

- **Step 3 :** ***Applying Bayes Theorem***
  - **If the weather is sunny, then the Player should play or not?**

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Naïve Bayes Classification Model :

Example 2: Given dataset

Hypothesis :
$X = \{Color = Green, Legs = 2, Height = Tall, Smelly = No\}$

Sl. No.	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

Naïve Bayes

Example 2: Given dataset

Hypothesis :

$X = \{Color = Green, Legs = 2, Height = Tall, Smelly = No\}$

1st Step:

First calculate the probability of the species being M or H in total.

$P(Species = M) = 4/8 = 0.5$

$P(Species = H) = 4/8 = 0.5$

2nd Step

Calculate the conditional probability of each attribute value for each class label.

$P(Color = White \mid Species = M) = 2/4 = 0.5$

$P(Color = White \mid Species = H) = ¾ = 0.75$

$P(Color = Green \mid Species = M) = 2/4 = 0.5$

$P(Color = Green \mid Species = H) = ¼ = 0.25$

$P(Legs = 2 \mid Species = M) = 1/4 = 0.25$

$P(Legs = 2 \mid Species = H) = 4/4 = 1$

$P(Legs = 3 \mid Species = M) = 3/4 = 0.75$

$P(Legs = 3 \mid Species = H) = 0/4 = 0$

$P(Height = Tall \mid Species = M) = 3/4 = 0.75$

$P(Height = Tall \mid Species = H) = 2/4 = 0.5$

$P(Height = Short \mid Species = M) = 1/4 = 0.25$

$P(Height = Short \mid Species = H) = 2/4 = 0.5$

$P(Smelly = Yes \mid Species = M) = 3/4 = 0.75$

$P(Smelly = Yes \mid Species = H) = 1/4 = 0.25$

$P(Smelly = No \mid Species = M) = 1/4 = 0.25$

$P(Smelly = No \mid Species = H) = 3/4 = 0.75$

Naïve Bayes Example 2: Given dataset

We can tabulate the above calculations in the tables for better visualization.

Conditional Probabilities for **Color** Attribute

Color	**M**	**H**
White	0.5	0.75
Green	0.5	0.25

Conditional Probabilities for **Height** Attribute

Height	**M**	**H**
Tall	0.75	0.5
Short	0.25	0.5

Conditional Probabilities for **Legs** Attribute

Legs	**M**	**H**
2	0.25	1
3	0.75	0

Conditional Probabilities for **Smelly** Attribute

Smelly	**M**	**H**
Yes	0.75	0.25
No	0.25	0.75

Naïve Bayes Example 2: Given dataset

Hypothesis :

$X = \{Color = Green, Legs = 2, Height = Tall, Smelly = No\}$

Probability of X belonging to Species M will be as follows.

$$\boldsymbol{P(M|X) = P(Species = M) * P(Color = Green|Species = M) * P(Legs = 2|Species = M) * P(Height = Tall|Species = M) * P(Smelly = No|Species = M)}$$

= 0.5*0.5*0.25*0.75*0.25

= 0.0117

Probability of X belonging to Species H will be as follows.

$$\boldsymbol{P(H|X) = P(Species = H) * P(Color = Green|Species = H) * P(Legs = 2|Species = H) * P(Height = Tall|Species = H) * P(Smelly = No|Species = H)}$$

=0.5*0.25*1*0.5*0.75

=0.0468

Naïve Bayes Classification Model : Example

- **Advantages :**
  - One of the ***fast*** and ***easy ML algorithms*** to predict a class of datasets.
  - It can be used for ***Binary*** as well as ***Multi-class Classifications***.
  - It ***performs well in Multi-class predictions*** as compared to the other Algorithms.
  - It is the most popular choice for text classification problems.

- **Disadvantages :**
  - Assumes that ***all features are independent or unrelated***, so it ***cannot learn the relationship between features***.

- **Applications :**
  - Credit Scoring.
  - Medical data classification.
  - It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
  - Text classification such as Spam filtering and Sentiment analysis

Naïve Bayes Classification Model : Summary

- Naïve Bayes algorithm is a **supervised learning algorithm**, which is based on **Bayes theorem** and used for **solving classification problems**.
- Bayes' rule can be turned into a classifier
- Maximum A Posteriori (MAP) hypothesis estimation incorporates prior knowledge; Max Likelihood (ML) doesn't
- Naive Bayes Classifier is a **simple but effective Bayesian classifier** for vector data (i.e. data with several attributes) that assumes that attributes are independent given the class.
- Bayesian classification is a generative approach to classification
- It is a **probabilistic classifier**, which means it predicts on the basis of the probability of an object

K-Nearest Neighbor (KNN) Algorithm

- K-Nearest Neighbour based on **Supervised Learning** technique used for **Regression** as well as for **Classification**.
- It ***assumes*** the ***similarity between*** the ***new case/data*** and ***available cases*** and put the ***new case*** into the ***category*** that is ***most similar*** to the ***available categories***.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

KNN model will find the similar features of the **new data set to the cats and dogs' images** and based on the **most similar features it will put it in either cat or dog category.**

Need of KNN

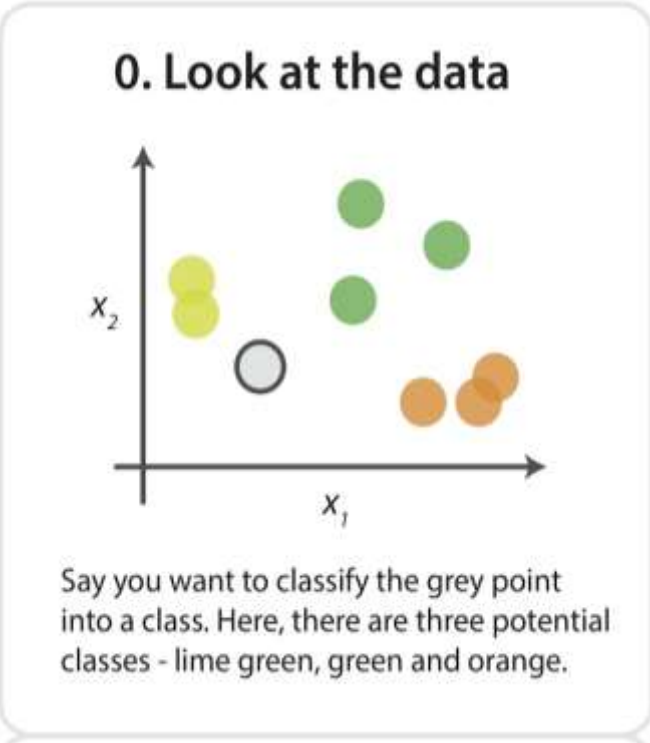
- KNN algorithm at the **training phase** just **stores the dataset** and,
- When it **gets new data**, then it **classifies** that **data** into a **category** that is **much similar to the new data**.
- It is also called a **Lazy Learner Algorithm**
  - As it **does not learn** from the **training set immediately**
  - Instead, it **stores the dataset** and at **time of classification**, it **performs** an **action on the dataset**.

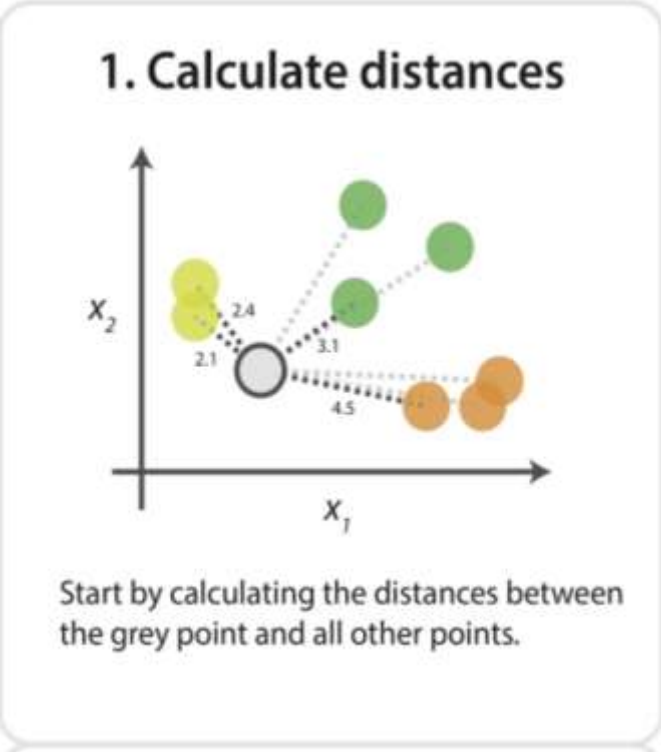
- **A feature space representation of the instance in the dataset and a measure of similarity between instances.**
- **The prediction is based on finding out what class the nearest instance belongs to.**

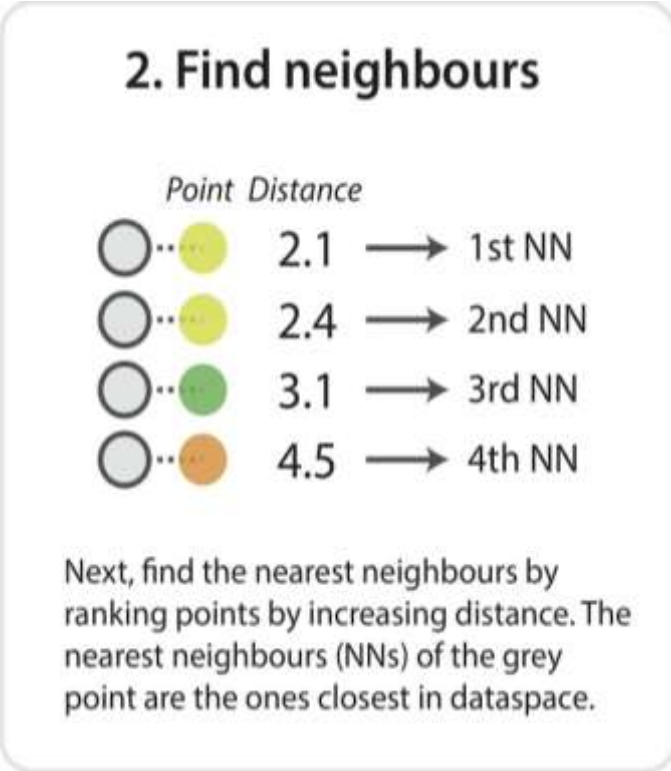
Need of KNN

- KNN algorithm at the **training phase** just **stores the dataset** and,
- When it **gets new data**, then it **classifies** that **data** into a **category** that is **much similar to the new data**.
- It is also called a **Lazy Learner Algorithm** A
  - As it **does not learn** from the **training set immediately**
  - Instead, it **stores the dataset** and at the **time of classification**, it **performs** an **action on the dataset**.

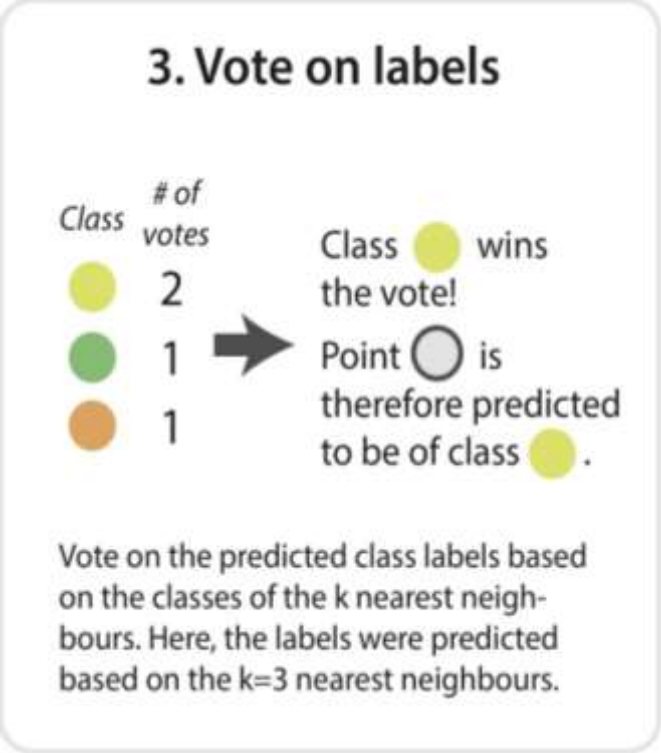
KNN Algorithm





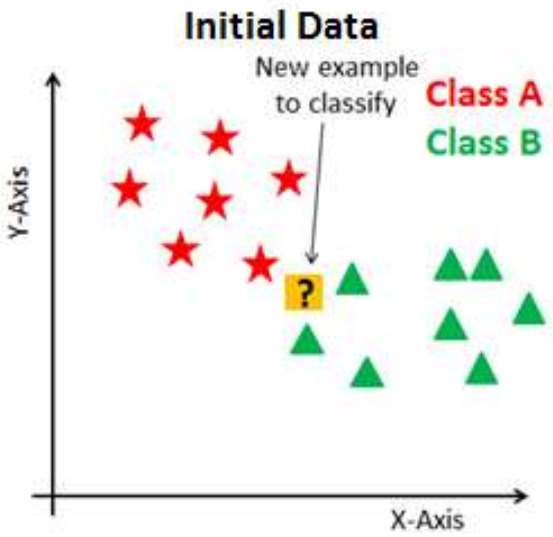


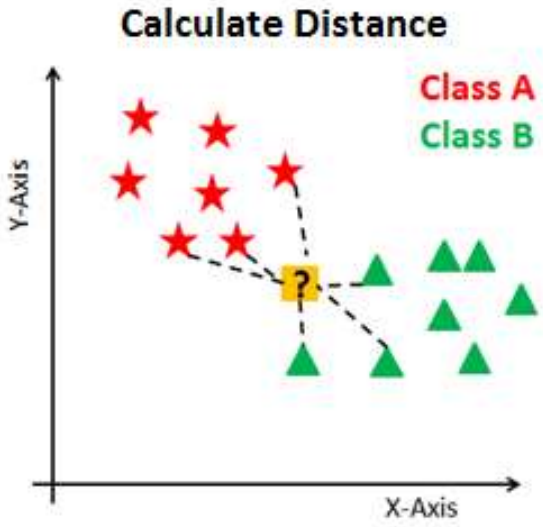
KNN Algorithm

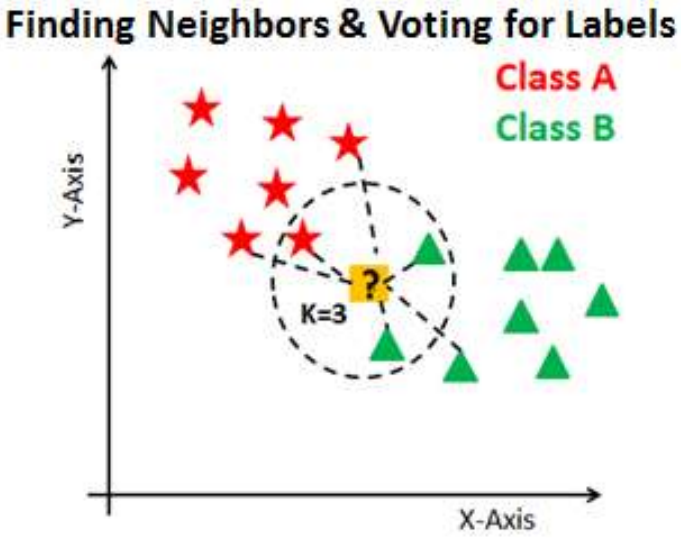


KNN Algorithm

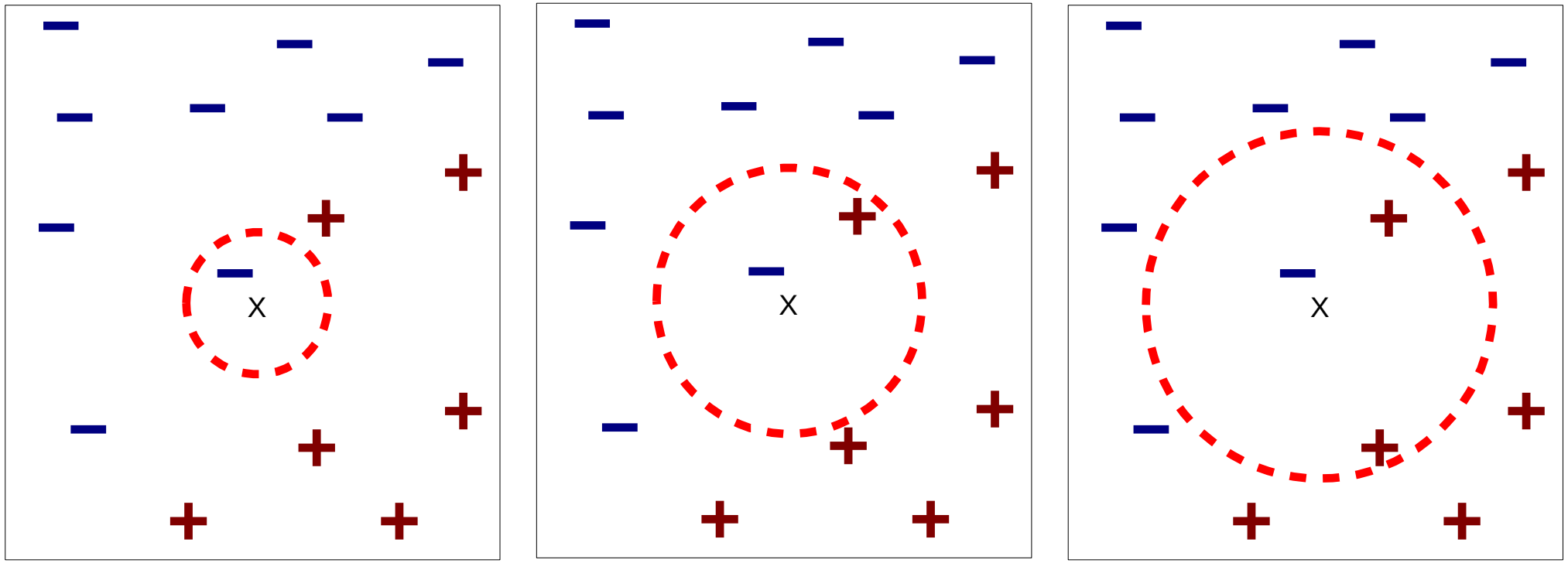
- **Pseudocode:**
  1. Load the data
  2. Choose K value
  3. For each data point in the data:
     A. Find the Euclidean distance to all training data samples
     B. Store the distances on an ordered list and sort it
     C. Choose the top K entries from the sorted list
     D. Label the test point based on the majority of classes present in the selected points
  4. End







KNN Algorithm

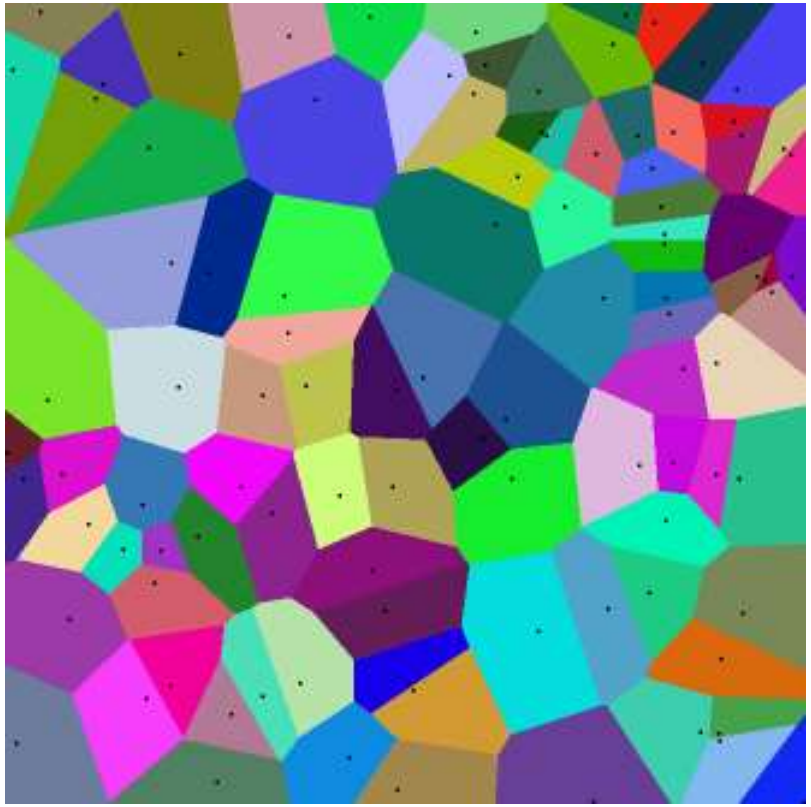


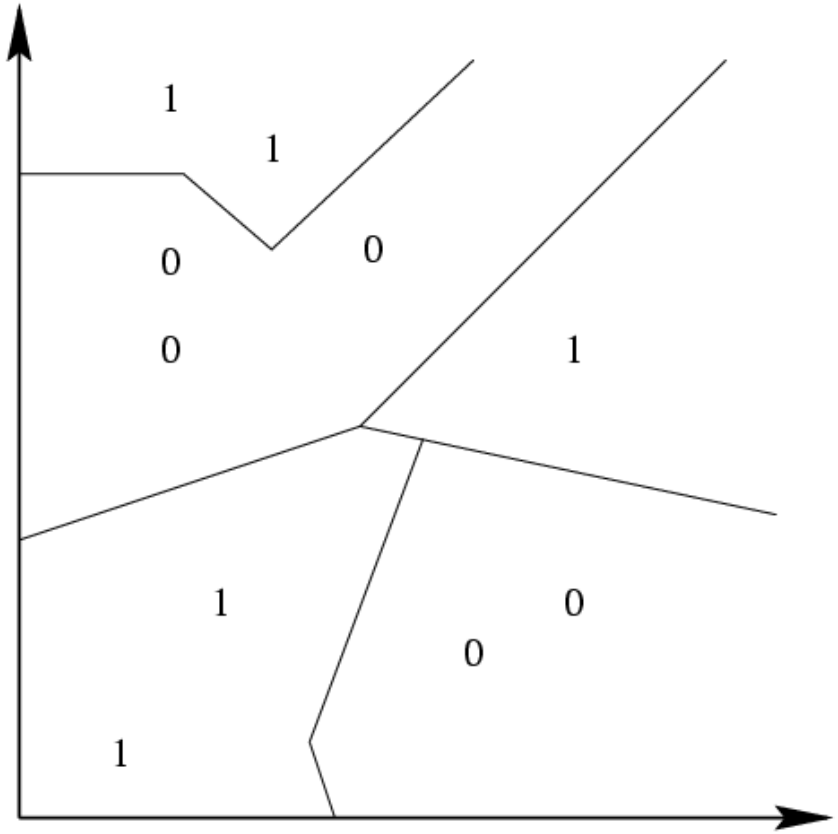
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

KNN Algorithm

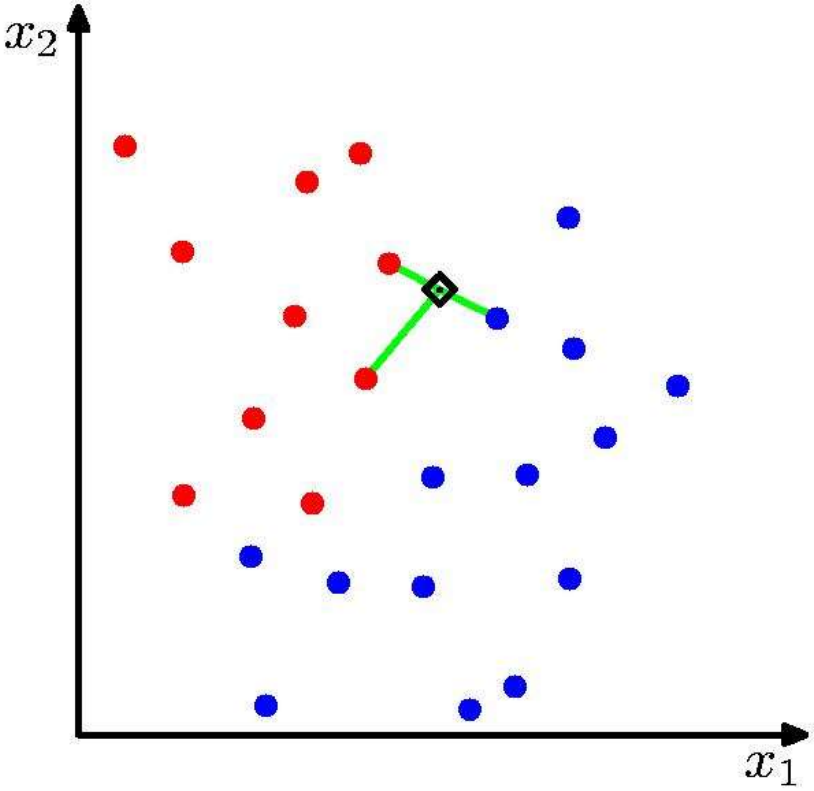
- **Voronoi Diagram**
  - Each line segment is equidistance between points in opposite classes.
  - The more points, the more complex the boundaries.



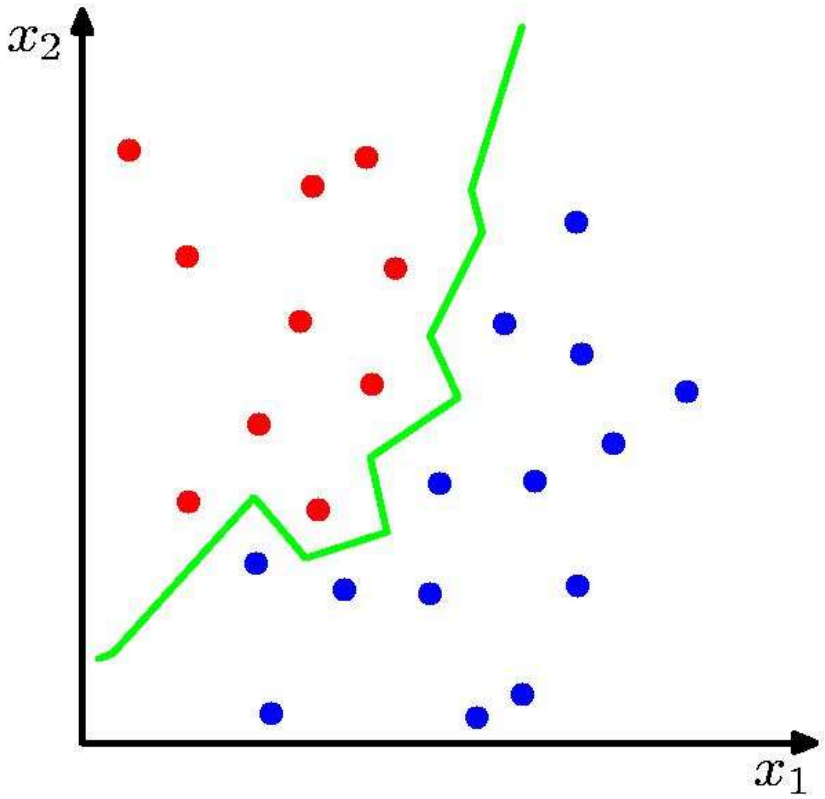


KNN Algorithm

- **Voronoi Diagram**
  - The boundary is always the perpendicular bisector of the line between two points (Voronoi tessellation)



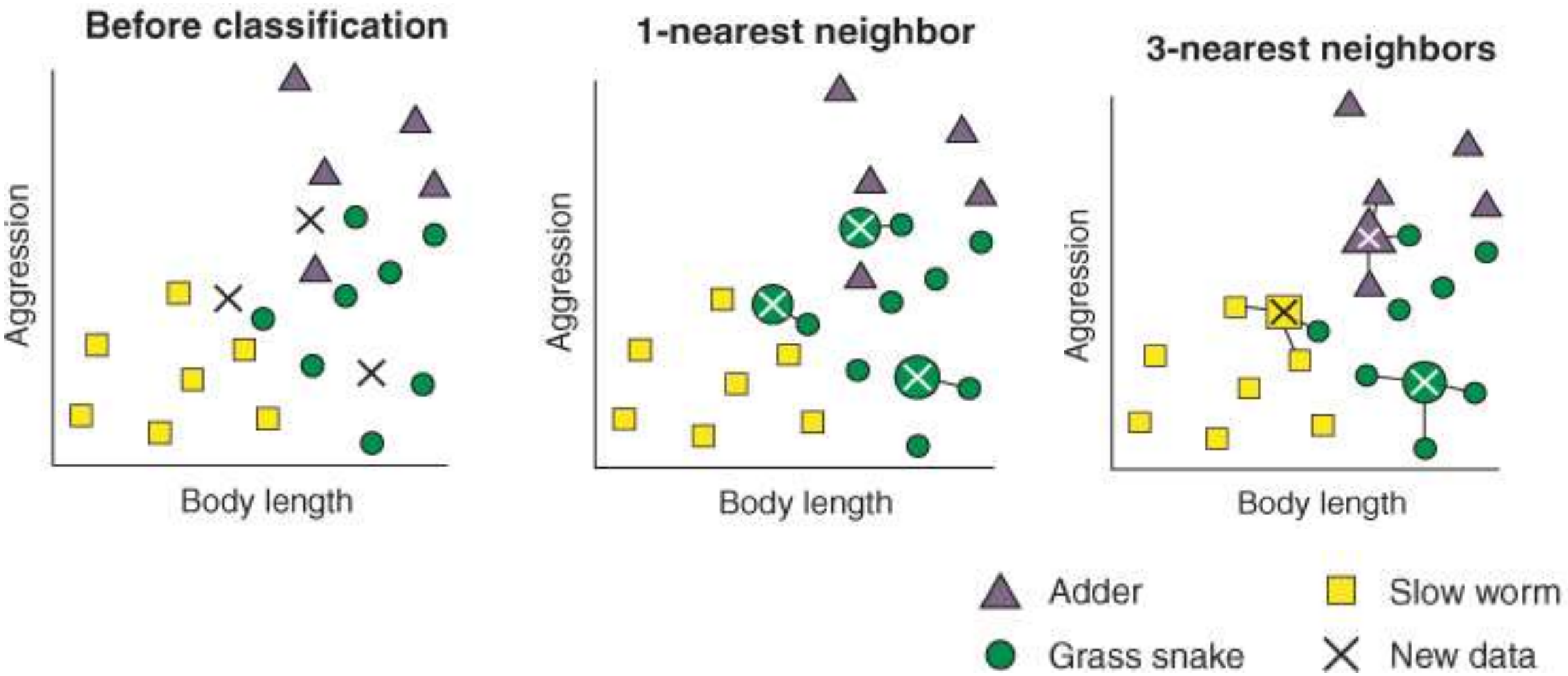
(a)



(b)

KNN Algorithm

- **Example**



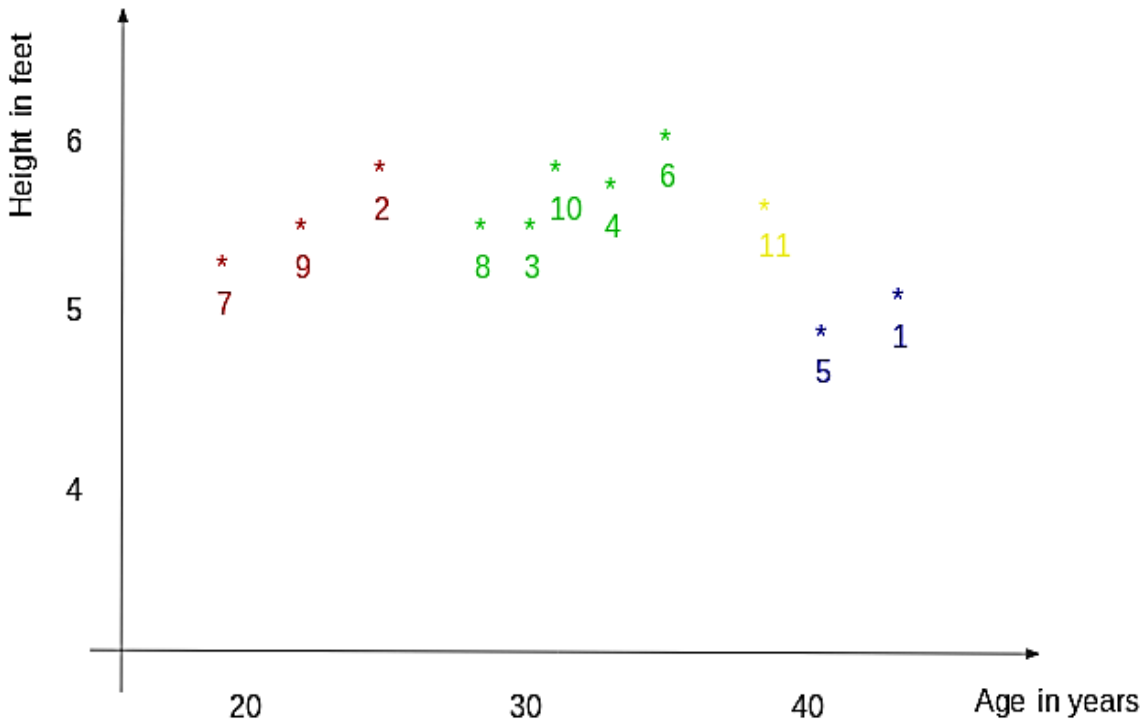
KNN Algorithm

- **Choice of k parameter**
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes
  - Higher values of k provide smoothing that reduces the risk of overfitting due to noise in the training data
  - Value of k can be chosen based on error rate measures
  - We should also avoid over-smoothing by choosing k=n, where n is the total number of tuples in the training data set

KNN Algorithm : Example

- **Let us consider the data given in table above consisting of 10 entries.**

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?



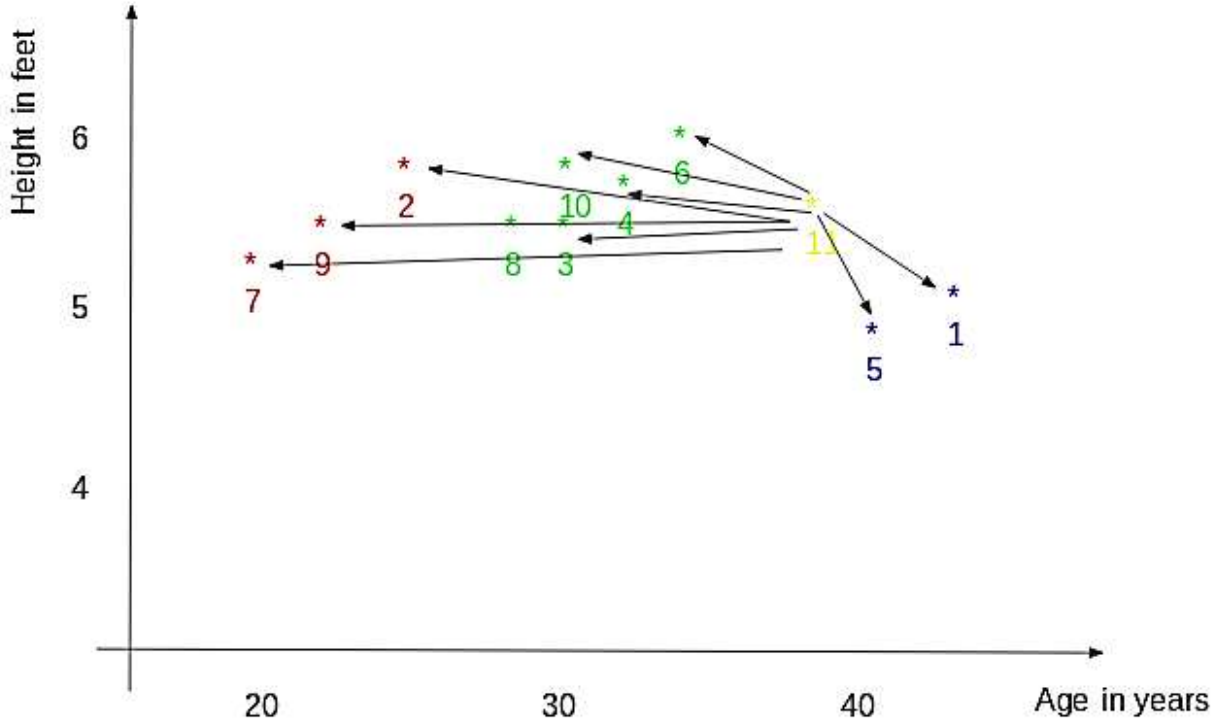
KNN Algorithm : Example

- **Let us consider the data given in table above consisting of 10 entries.**

1. **Euclidean Distance:** Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).
2. **Manhattan Distance**: This is the distance between real vectors using the sum of their absolute difference.

$$Euclidean = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

$$Manhattan = \sum_{i=1}^{k}|x_i - y_i|$$

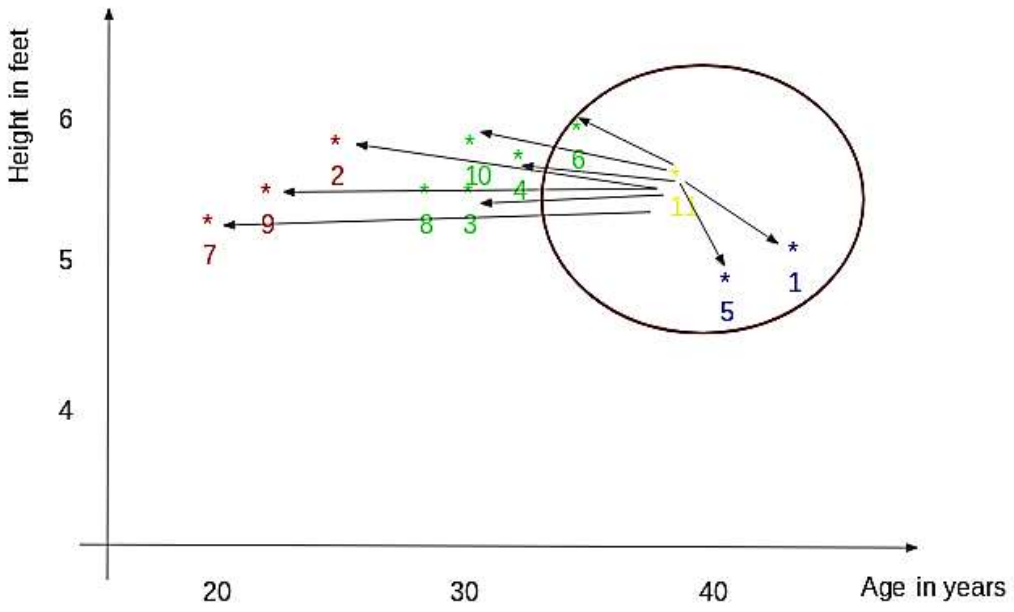


KNN Algorithm : Example

- Let us consider the data given in table above consisting of 10 entries.

- The closest k data points are selected (based on the distance).
- In this example, points 1, 5, 6 will be selected if the value of k is 3.

ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60



- Select the k value.
- This determines the number of neighbors we look at when we assign a value to any new observation.
- In our example, for a value k = 3, the closest points are ID1, ID5 and ID6.

The prediction of weight for ID11 will be:

```
ID11 = (77+72+60)/3

ID11 = 69.66 kg
```

KNN Algorithm : Example

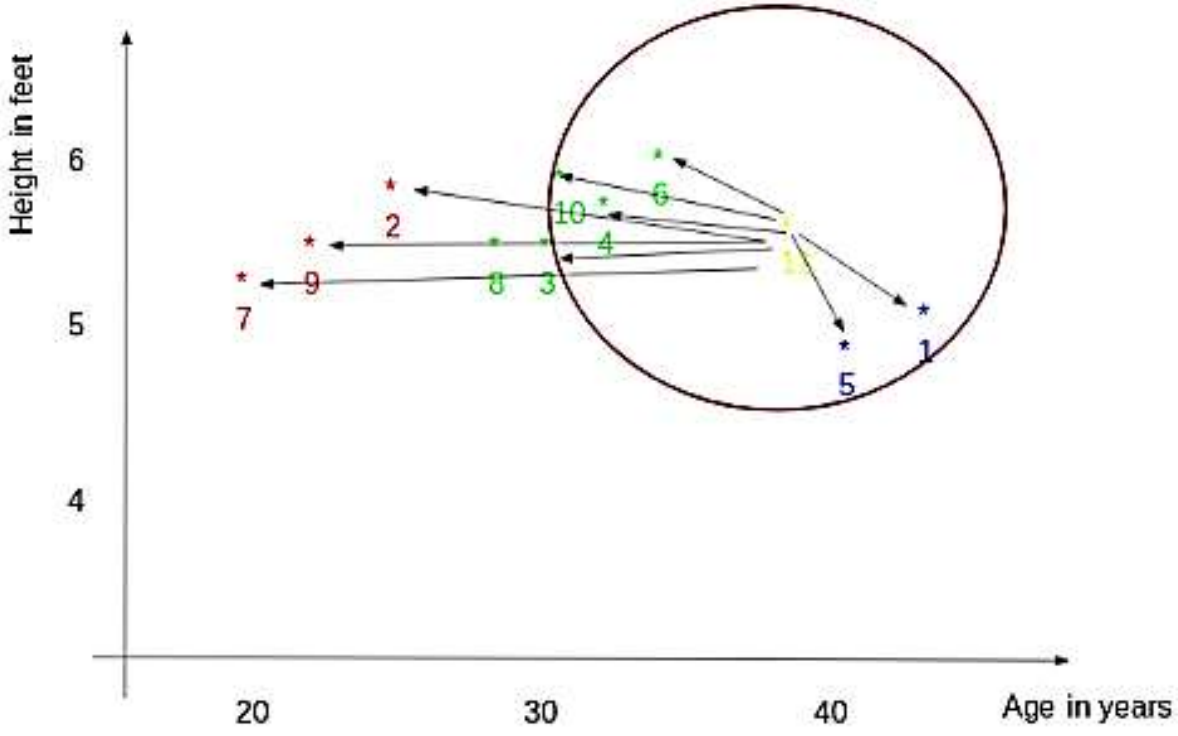
- In our example, for a value k = 5, the closest points are ID1, ID4, ID5, ID6 and ID10.

ID	Height	Age	Weight
1	5	45	77
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
10	5.6	32	58

The prediction for ID11 will be :

```
ID 11 =  (77+59+72+60+58)/5

ID 11 = 65.2 kg
```



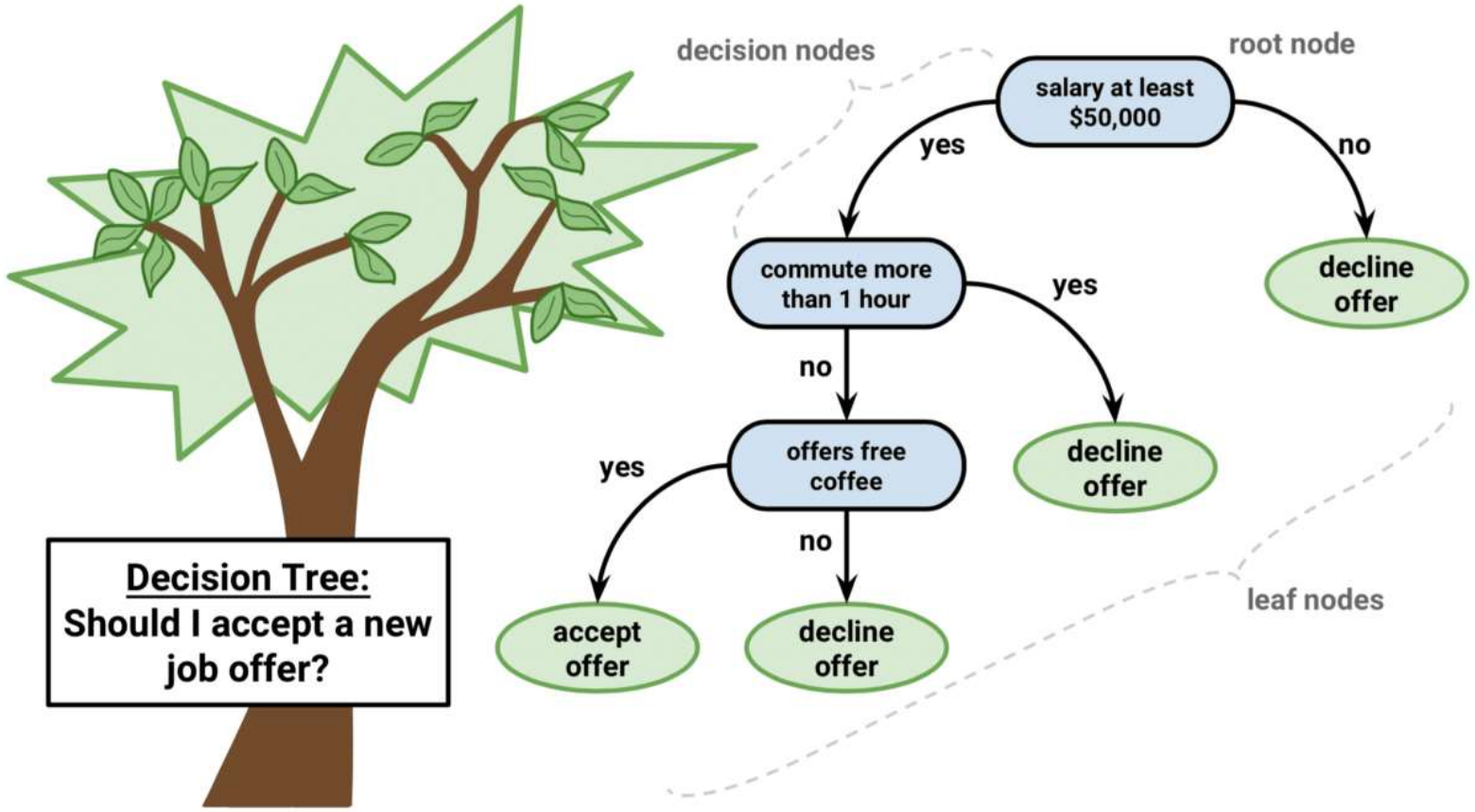
KNN Algorithm

- **Advantages**
  - It is simple to implement.
  - It is robust to the noisy training data
  - It can be more effective if the training data is large.

- **Disadvantages:**
  - Always needs to determine the value of K which may be complex some time.
  - The computation cost is high because of calculating the distance between the data points for all the training samples.

Decision Tree

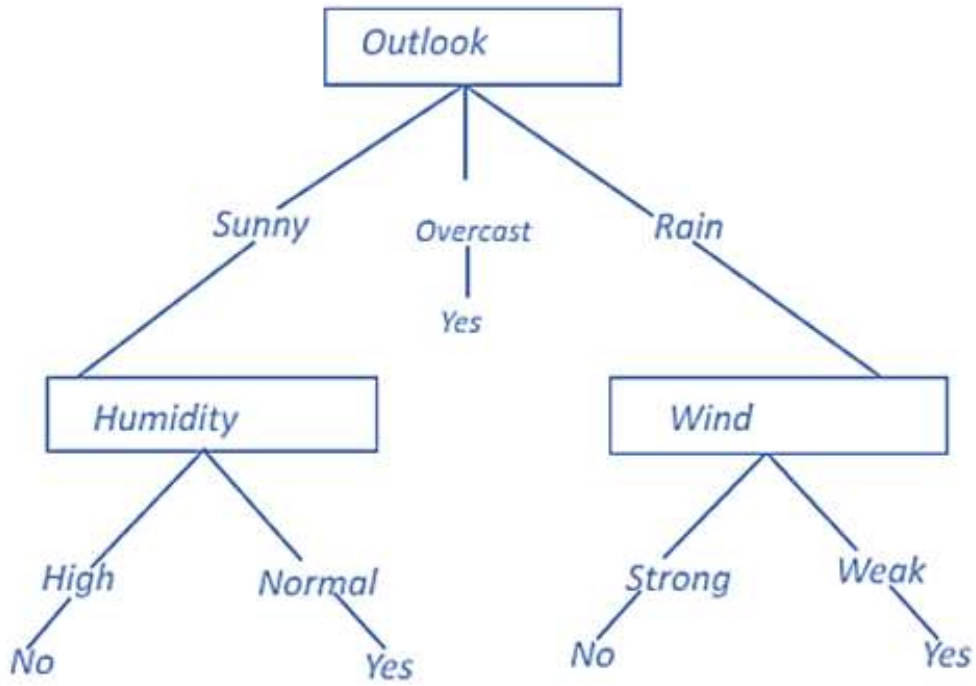
- Decision tree is a flowchart-like tree structure:
  - **Each internal node :** denotes a ***test*** on an attribute
  - **Each branch :** represents an ***outcome*** of the test, and
  - **Each leaf node :** (terminal node) holds a ***class label***.



Decision Tree

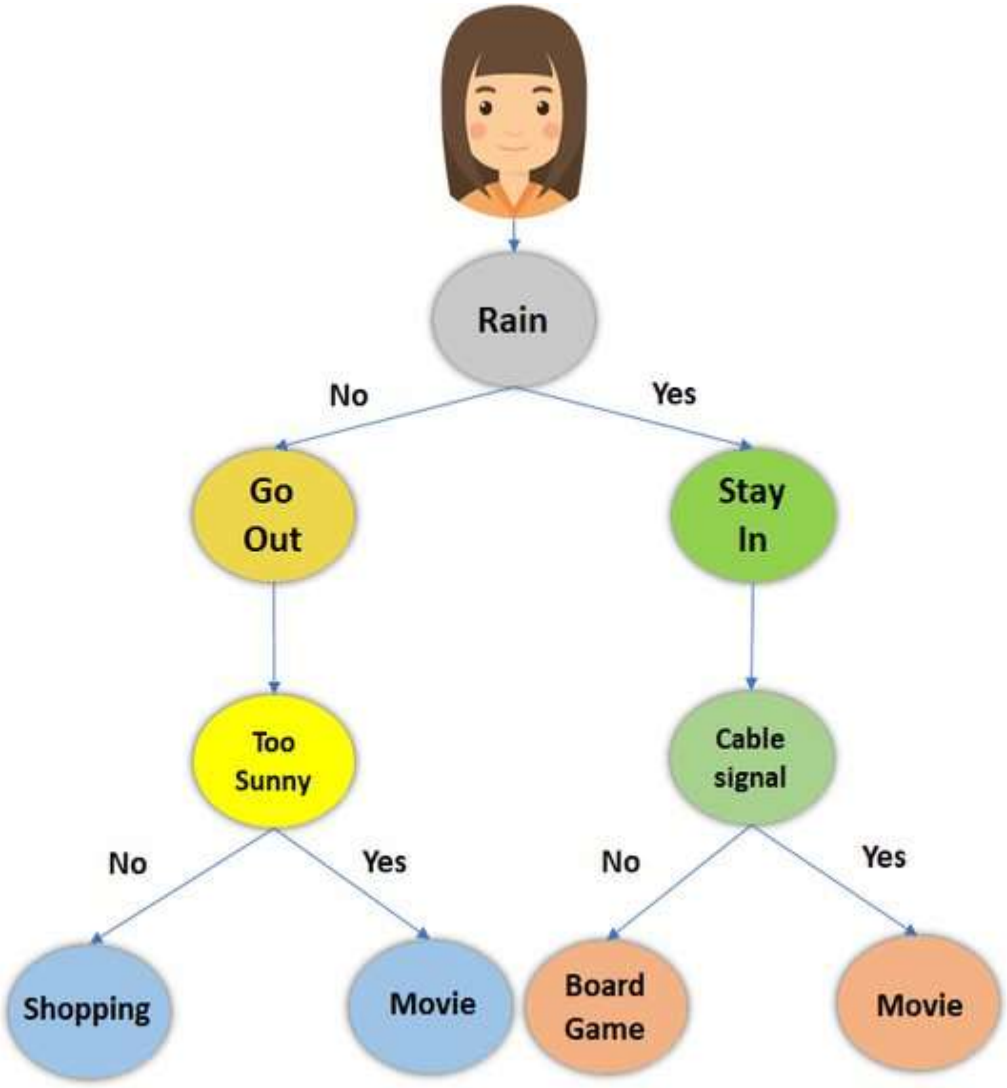
- A tree can be ***"learned" by splitting*** the ***source set*** into ***subsets*** based on an ***attribute value test***.
- ***Process is repeated*** on each ***derived subset*** in a ***recursive manner*** called recursive partitioning.
- ***Recursion is completed*** when the ***subset*** at a ***node*** all has the ***same value of the target variable***, or when splitting no longer adds value to the predictions.
- **Construction** of a decision tree classifier ***does not require any domain knowledge*** or ***parameter setting***, and therefore is appropriate for exploratory knowledge discovery.

Decision Tree for Playing Tennis



Decision Tree

- Decision trees can handle **high-dimensional data**. In general decision tree classifier has good accuracy.
- Decision tree induction is a typical **inductive approach** to **learn knowledge** on classification.
- Decision trees **classify instances** by **sorting them down the tree** from the **root** to **some leaf node**, which provides the classification of the instance.
- An **instance** is **classified** by ***starting*** at the ***root node*** of the tree, ***testing*** the ***attribute specified*** by this ***node***, then ***moving down*** the ***tree branch corresponding*** to the ***value*** of the ***attribute***.



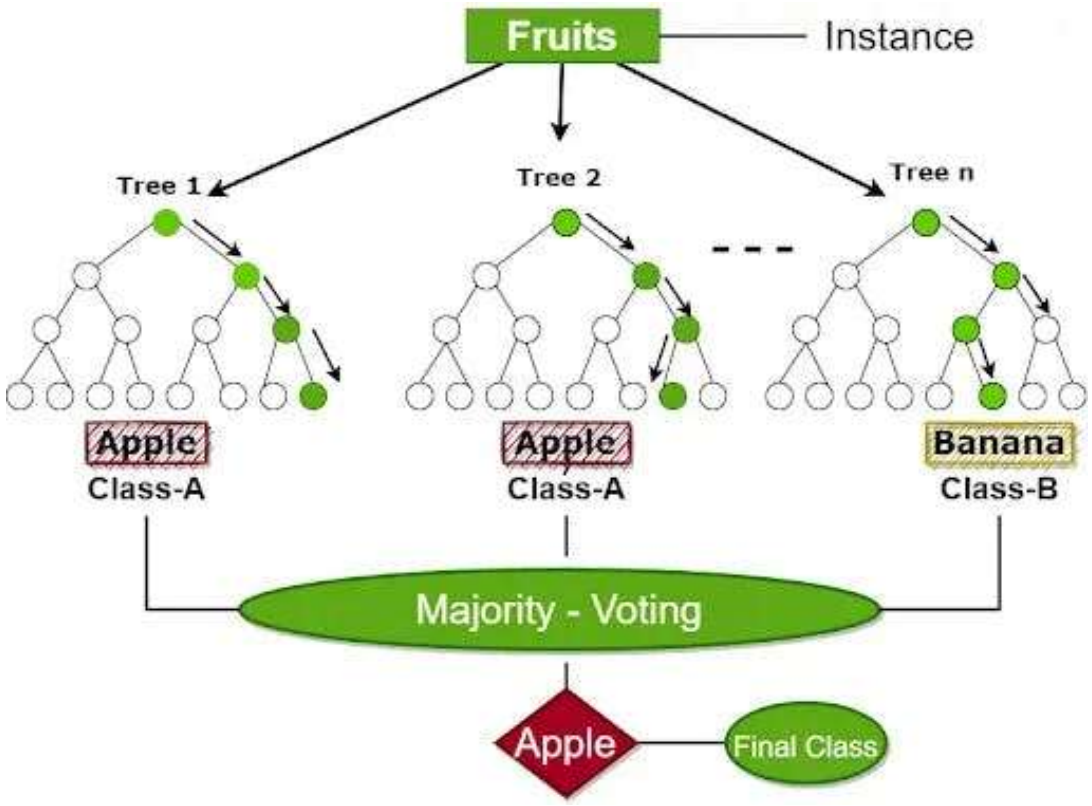
Jane's Weekend Plan

Decision Tree

- **Strength:**
  - Decision trees are able to **generate understandable rules**.
  - Decision trees **perform classification without requiring much computation**.
  - Decision trees are able to **handle both continuous and categorical variables**.
  - Decision trees provide a clear indication of which **fields are most important for prediction** or **classification**.
- **Disadvantage:**
  - Decision trees are **less appropriate for estimation tasks** where the goal is to predict the value of a continuous attribute.
  - Decision trees are **prone to errors** in classification problems with many classes and a relatively small number of training examples.
  - Decision tree can be **computationally expensive to train**. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

Random Forest

- Random forest is a commonly-used machine learning algorithm.
- Trademarked by Leo Breiman and Adele Cutler,
- RF combines the output of multiple decision trees to reach a single result.
  - The “forest” it builds is an ensemble of decision trees, usually trained with the bagging method
  - Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.
- It is a supervised learning algorithm.



Random Forest

- **Random forest** is identified as a **collection of decision trees**.
  - **Each tree estimates a classification**, and this is called a "**VOTE**".
  - Ideally, **consider each vote** from **every tree** and **chose the most voted classification** (Majority-Voting).

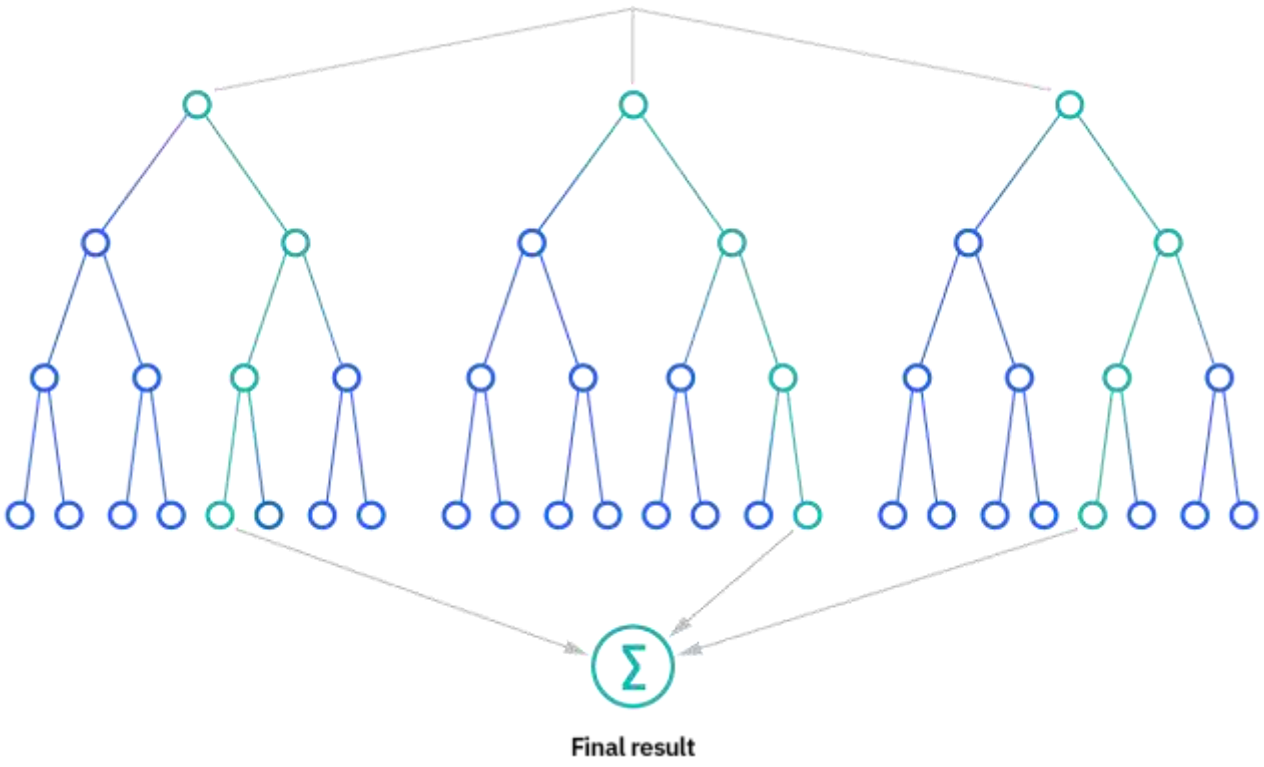
- **Random Forest** follow the same **bagging process** as the **decision trees**
  - Each time a *split is to be performed*, the search for the *split variable* is *limited* to a *random subset* of *m of the p attributes* (variables or features) aka Split-Attribute Randomization :
    - ***Classification trees***: $m = \sqrt{p}$
    - ***Regression trees***: $m = p/3$

- Random Forests produce many unique trees.

Random Forest

- Random forest algorithms have **three main hyperparameters**, which need to be set before training.
  - **Node size,**
  - **Number of trees,**
  - **Number of features sampled.**

- From there, the random forest classifier can be used to solve for ***regression*** or ***classification*** problems.



Random Forest

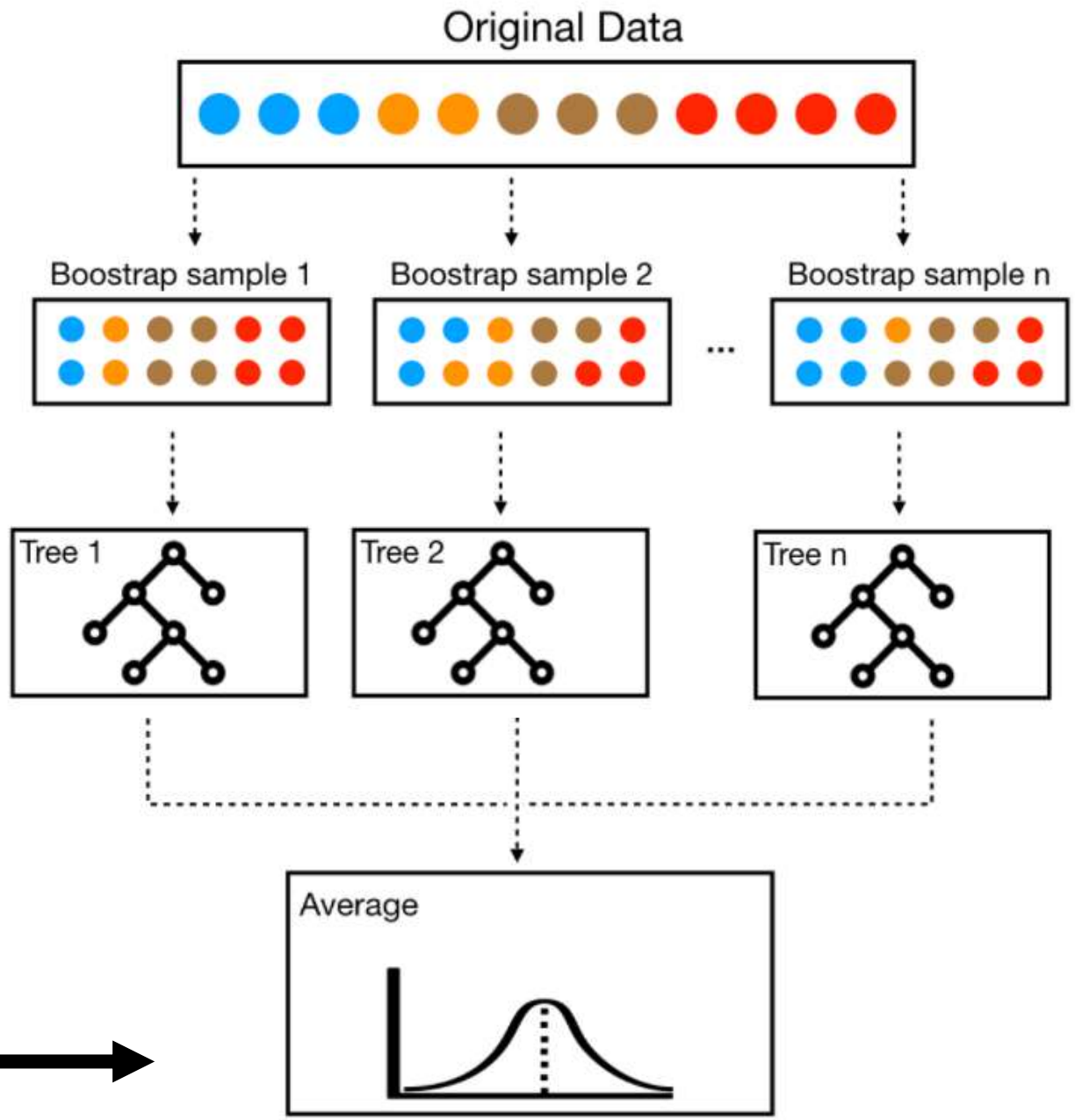
- Random forest algorithm is made up of a **collection of decision trees**.
- **Each tree in the ensemble:**
  - Comprised of a data sample drawn from a training set with replacement, called the **bootstrap** sample.
- **Of that training sample,**
  - **One-third** of it is set aside as **test data**, known as the ***out-of-bag (oob)*** sample.
  - **Another instance of randomness** is then injected through ***feature bagging***, adding more **diversity** to the **dataset** and reducing the correlation among decision trees.
  - Depending on the **type of problem**, the **determination** of the **prediction will vary**.
    - **For a regression task**, the **individual decision trees** will be **averaged**, and
    - **For a classification task**, a **majority vote**—i.e. the most frequent categorical variable—will yield the predicted class.
  - **Finally**, the **oob sample** is then used for **cross-validation**, finalizing that prediction.

Random Forest

- Random forest algorithm is **suitable** for both **classifications** and **regression task**.
- It **gives** a **higher accuracy** through **cross validation**.
  - **Standard decision trees often have high variance and low bias**
    High chance of **overfitting** (with 'deep trees', many nodes)
  - With a **Random Forest**, the **bias remains low** and the **variance is reduced**
    thus, we decrease the **chances of overfitting**
- Random forest classifier **can handle** the **missing values** and **maintain the accuracy** of a large proportion of data.
- If there are ***more trees***, it ***doesn't allow over-fitting trees*** in the model.
- It has the ***ability to work upon a large data set*** with ***higher dimensionality***.

Bagging :
Bootstrap Aggregating

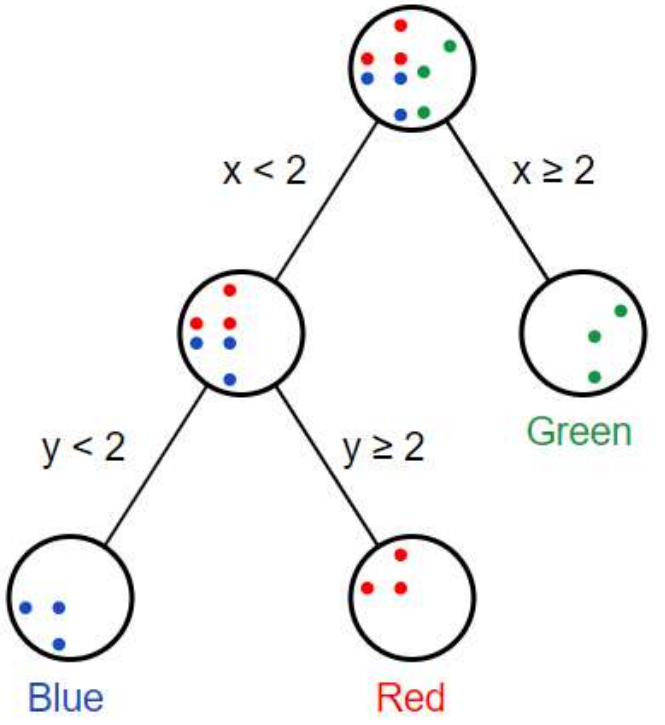
- Sample records with replacement
  "bootstrap" the training data
  - Sampling is the process of **selecting a subset of items** from a vast collection of items.
  - Bootstrap = Sampling with replacement. It **means a data point in a drawn sample can reappear in future** drawn samples as well.
- Fit an overgrown tree to each resampled data set
- Average predictions

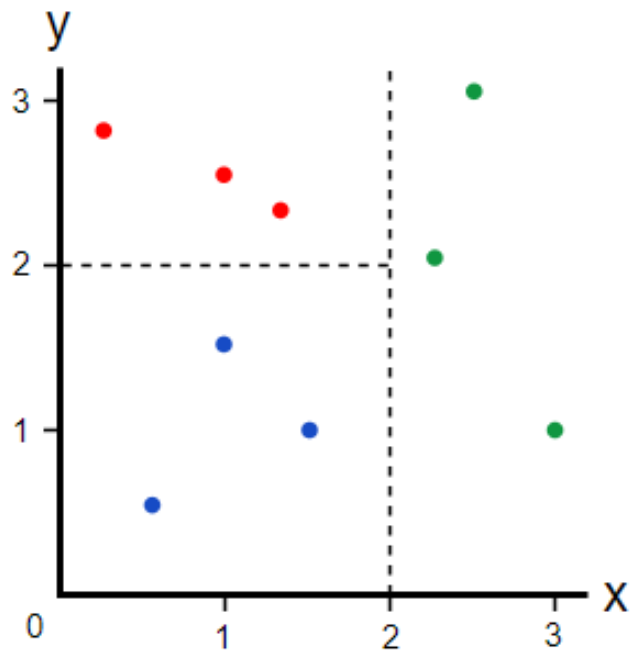


Random Forest

- **A random forest consists of decision trees.**
  - A decision tree consists of
    - **decision nodes** & the top decision node is called the ***root*** node
    - **terminal nodes** or leaf nodes

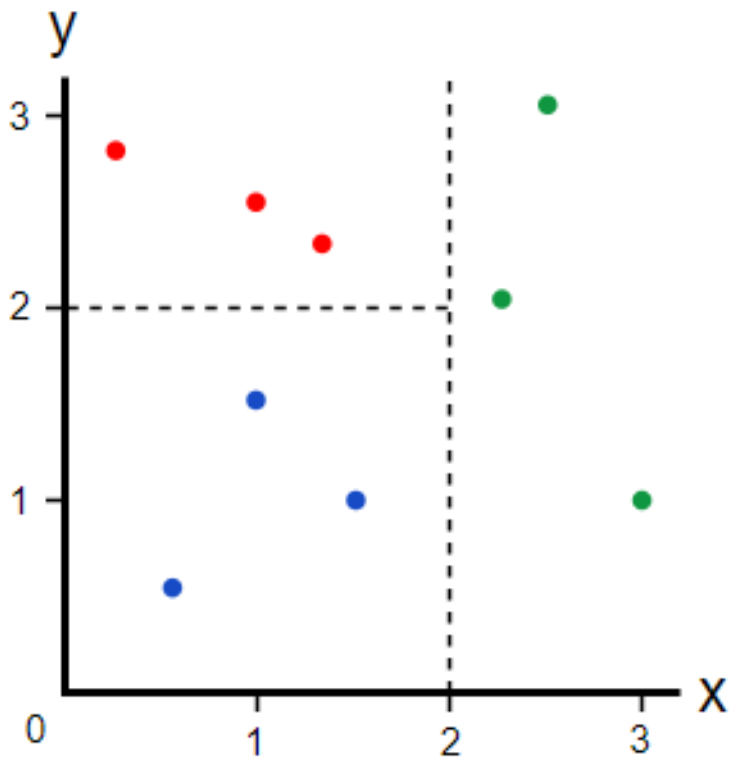
- **A selection of data and features is used for each tree**
  - For every decision tree
    - a sample of the training data is used
    - a sample of the features (√n_features up to 30 – 40%) is used





Random Forest

- For an individual decision tree:
  - Find the best split in the data.
    This is the **root node** (decision node)
  - Find in the first branch for that part of the data again the best split.
    That is the first **sub node** (decision node)
  - Continue creating decision nodes until splitting doesn't improve the situation
    The **average value of the target variable** is assigned to the leaf (terminal node)
  - Continue until there are only leaf nodes left or until a minimum value is reached
  - Now the decision tree can be used to do a prediction based on the input features

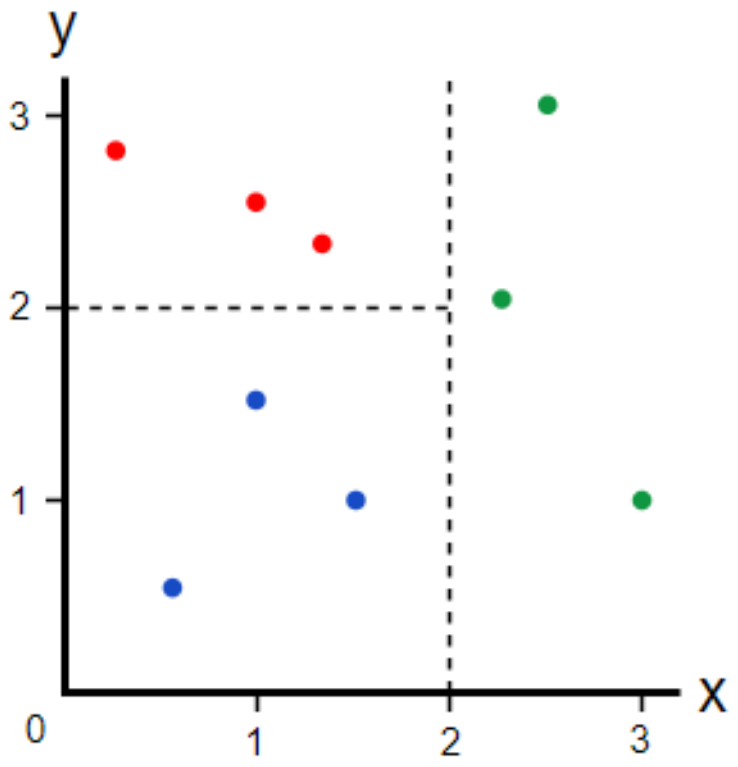


Random Forest

- **Deciding the Best Split:**
  - Determine the **mean** of all datapoints
  - Determine the **mean square error** of all datapoints

$$MSE = \frac{1}{n}\sum_{i}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

  - Apply a split.
  - Try at every datapoint
  - Calculate the **mean of all datapoints** on **each side of the split**
  - Calculate the **MSE on each side** and take a **weighted average of MSEs on all sides**
  - Lowest (weighted averaged) MSE is best split



Random Forest : Feature Importance

- **Feature importance is calculated as**
  - the **reduction in sum of squared errors** whenever a **variable is chosen to split** **weighted by the probability** of **reaching** that node.
- **Node probability**
  - Can be **calculated by the number of samples** that reach the node, **divided** by the **total number of samples**.
- **Higher the value** **the** **more important the feature.**
- However, the variable importance measures are ***not reliable in situations*** where ***potential predictor variables vary*** in their ***scale of measurement*** or their number of categories

Random Forest : Model Eval.

- **Split the data in a train set and test set**
  - Train the model using the trainset.
  - Test predictions using the test set.
  - Vary the train and test set contents to confirm results

- **Compare the test set and the train set**
  - Determine the **Coefficient of Determination** for both sets
    - The **proportion of the variance** in the **dependent variable** that is predictable from the independent variable(s)
    - **How well are observed outcomes replicated by the model**
    - If **R2 is different for both sets**, the **test or train set is probably biased**

- **Determine the accuracy of the predictions**
  - By comparing predicted results of the test set with the actual results of the test set

Random Forest : Tuning

Random forests provide good "out-of-the-📦" performance but there are a few parameters we can tune to increase performance.

- Number of trees
- mtry
- Node size
- Sampling scheme
- Split rule

- Typically have the largest impact on predictive accuracy.
- Tend to have marginal impact on predictive accuracy but still worth exploring. Can also increase computational efficiency.
- Generally used to increase computational efficiency

- Bagging introduces randomness into the rows of the data.
- Random forest introduces randomness into the rows and columns of the data
- Combined, this **provides a more diverse set of trees that almost always lowers our prediction error**.

Random Forest

Strengths

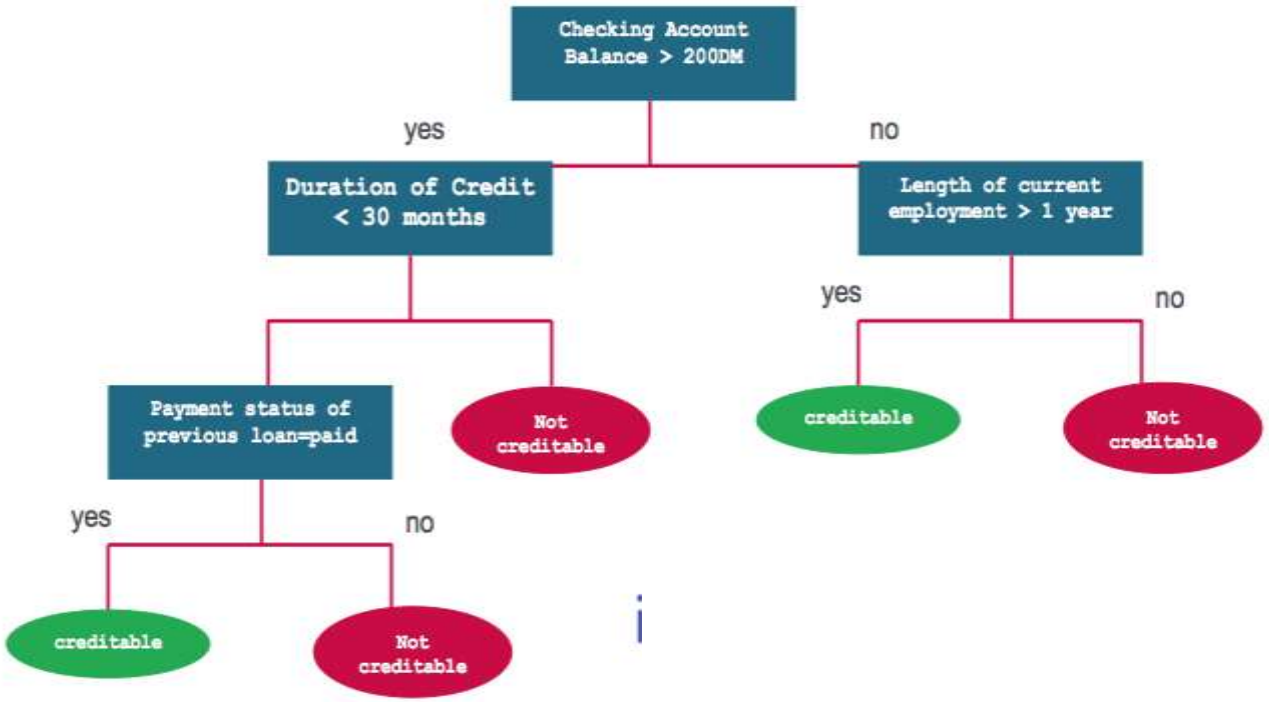
- Competitive performance.
- Remarkably good "out-of-the box" (very little tuning required).
- Built-in validation set (don't need to sacrifice data for extra validation).
- Typically does not overfit.
- Robust to outliers.
- Handles missing data (imputation not required).
- Provide automatic feature selection.
- Minimal preprocessing required.

Weaknesses

- Although accurate, often cannot compete with the accuracy of advanced boosting algorithms.
- Can become slow on large data sets.
- Less interpretable (although this is easily addressed with various tools such as variable importance, partial dependence plots, LIME, etc.).

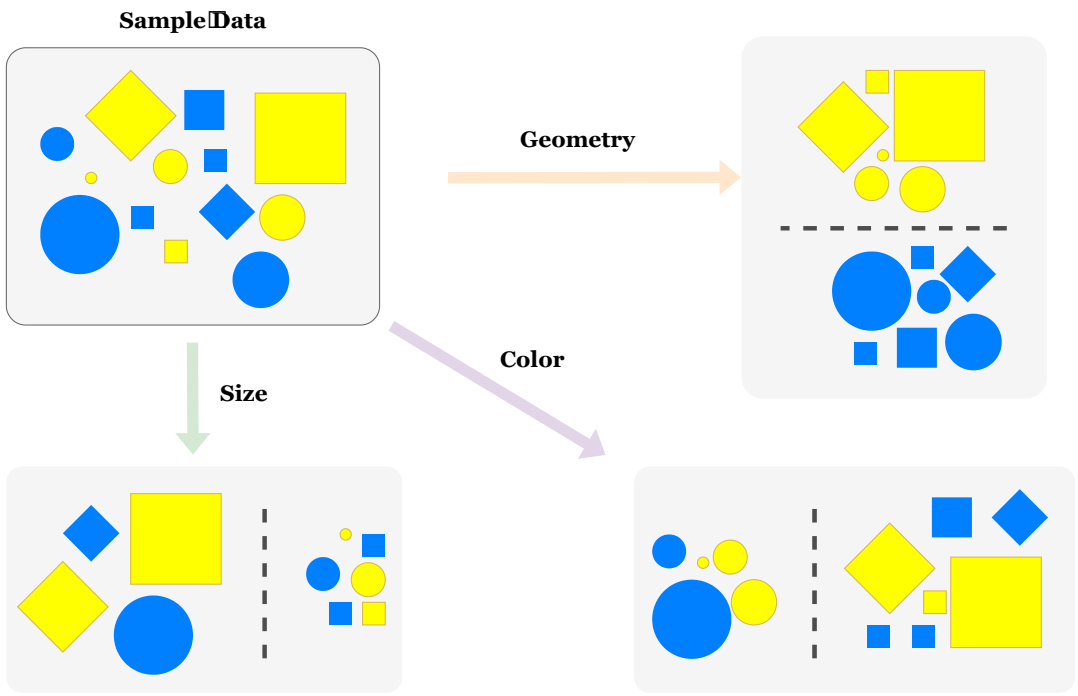
Random Forest

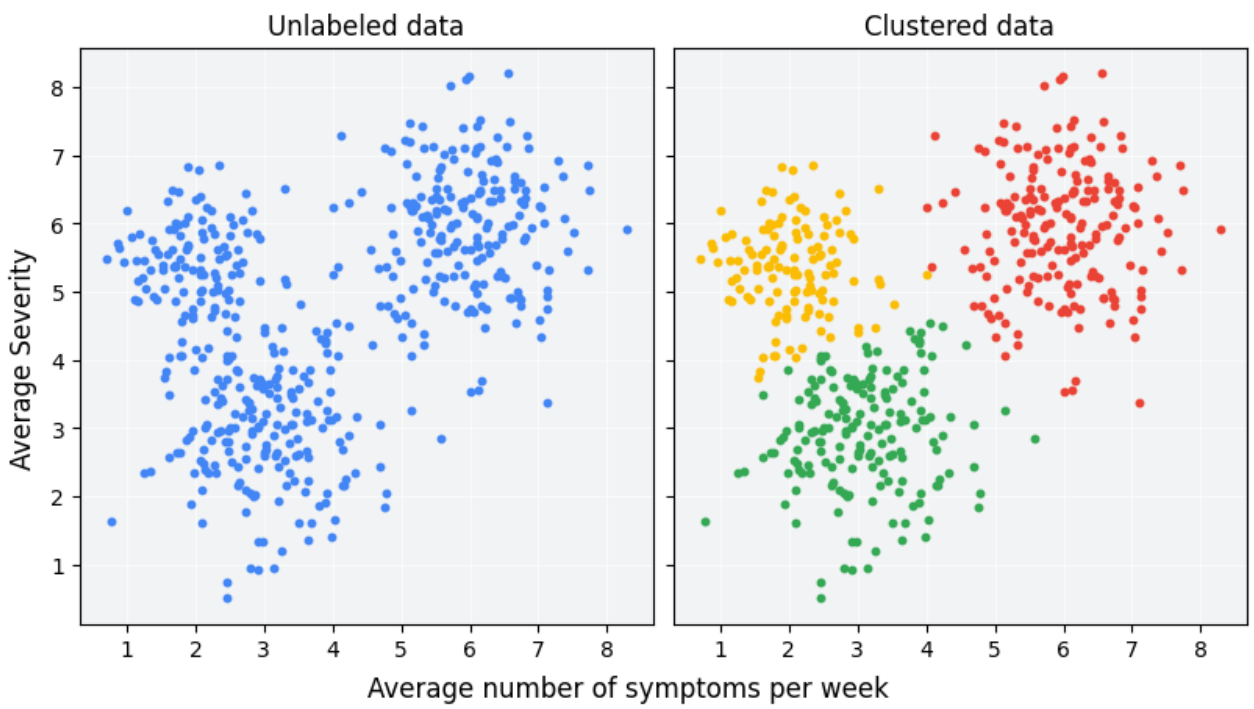
- Banking Industry
  - Credit Card Fraud Detection
  - Customer Segmentation
  - Predicting Loan Defaults
- Healthcare and Medicine
  - Cardiovascular Disease Prediction
  - Diabetes Prediction
  - Breast Cancer Prediction
- Stock Market
  - Stock Market Prediction
  - Stock Market Sentiment Analysis
  - Bitcoin Price Detection
- E-Commerce
  - Product Recommendation
  - Price Optimization
  - Search Ranking



Clustering

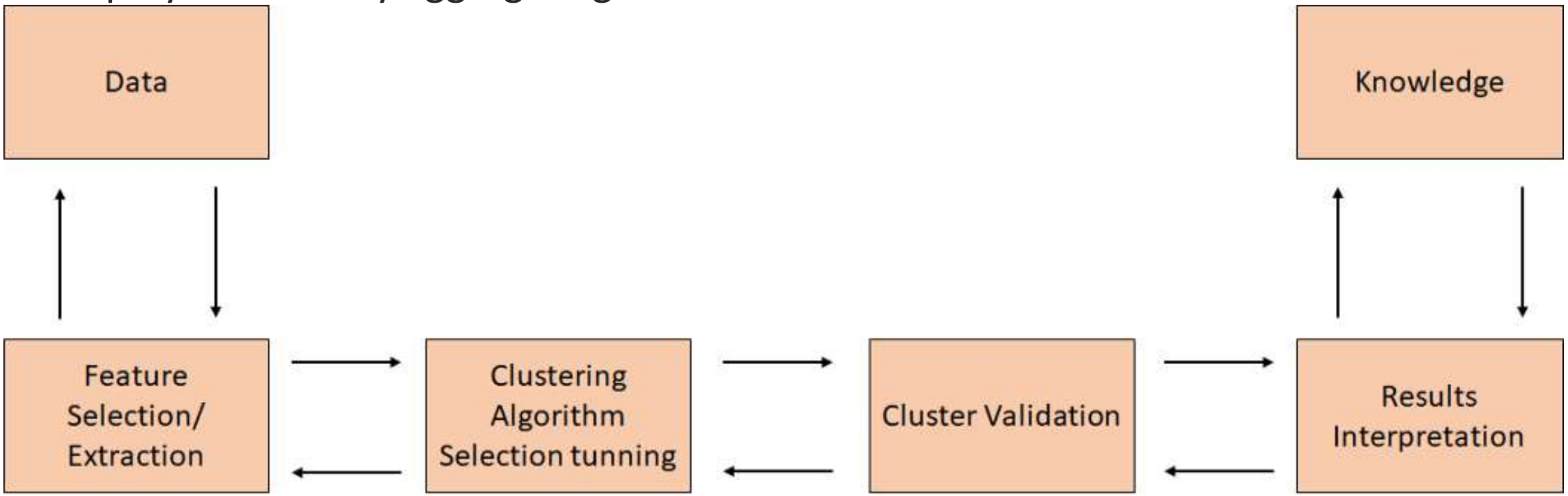
- **Segregate groups** with **similar traits** and **assign them into clusters**
- An **unsupervised learning method** of identifying similar groups of data in a dataset
- A way of grouping the data points into different clusters, consisting of similar data points.
- Objects with possible similarities remain in a group that has less or no similarities with another group



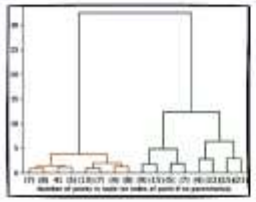
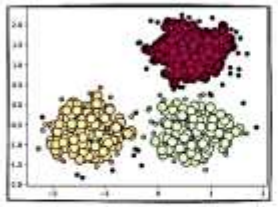
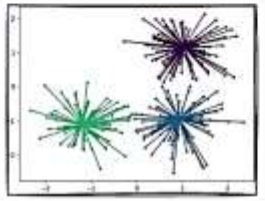
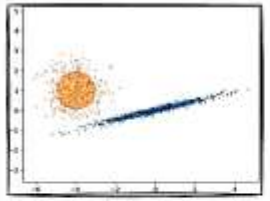
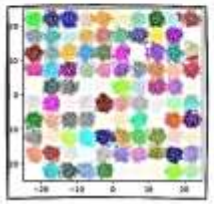


Clustering : Process

- Goal is to study the intrinsic (and commonly hidden) structure of the data
- Two approaches-Clustering and Dimensionality reduction.
- Segmenting datasets by some shared attributes.
- Detecting anomalies that do not fit to any group.
- Simplify datasets by aggregating variables with similar attributes.

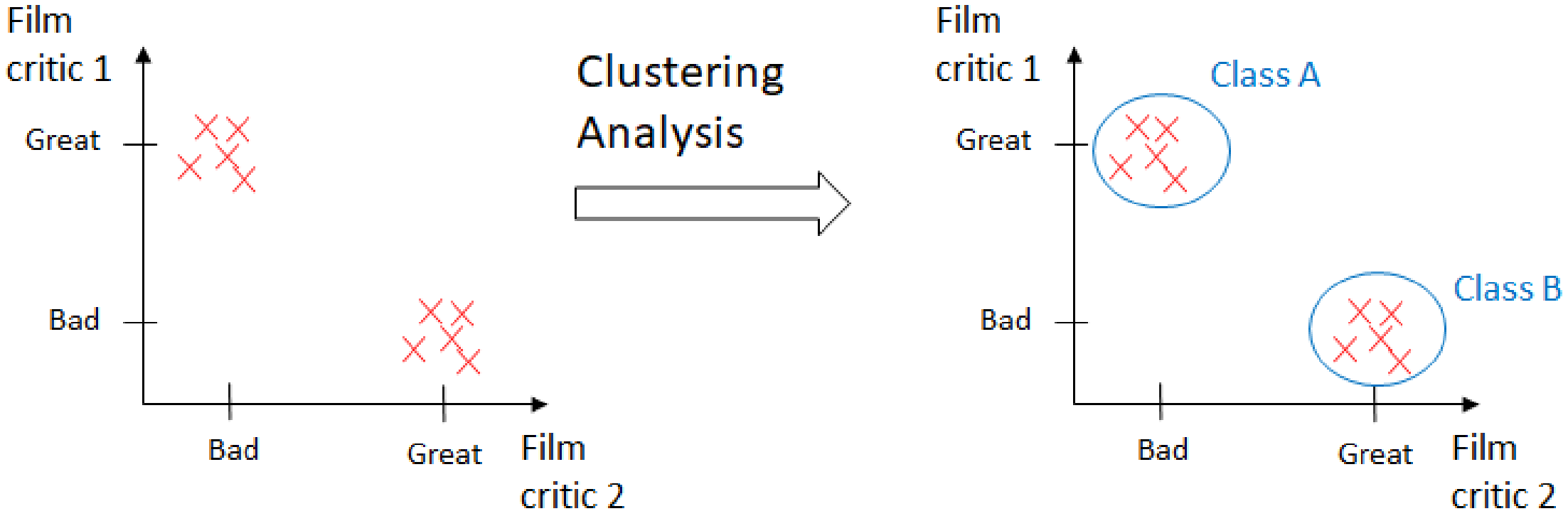


Clustering Types

Clustering Algorithm Type		Clustering Methodology	Algorithm(s)
	Centroid-based	Cluster points based on proximity to centroid	KMeans KMeans++ KMedoids
	Connectivity-based	Cluster points based on proximity between clusters	Hierarchical Clustering (Agglomerative and Divisive)
	Density-based	Cluster points based on their density instead of proximity	DBSCAN OPTICS HDBSCAN
	Graph-based	Cluster points based on graph distance	Affinitiy Propagation Spectral Clustering
	Distribution-based	Cluster points based on their likelihood of belonging to the same distribution.	Gaussian Mixture Models (GMMs)
	Compression-based	Transform data to a lower dimensional space and then perform clustering	BIRCH

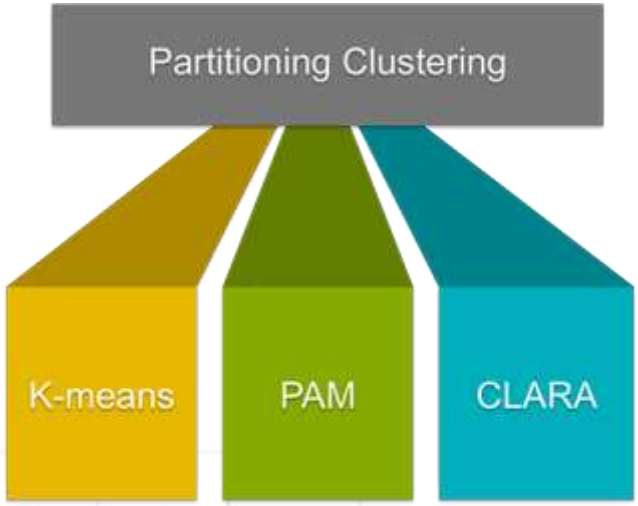
Clustering : Analysis

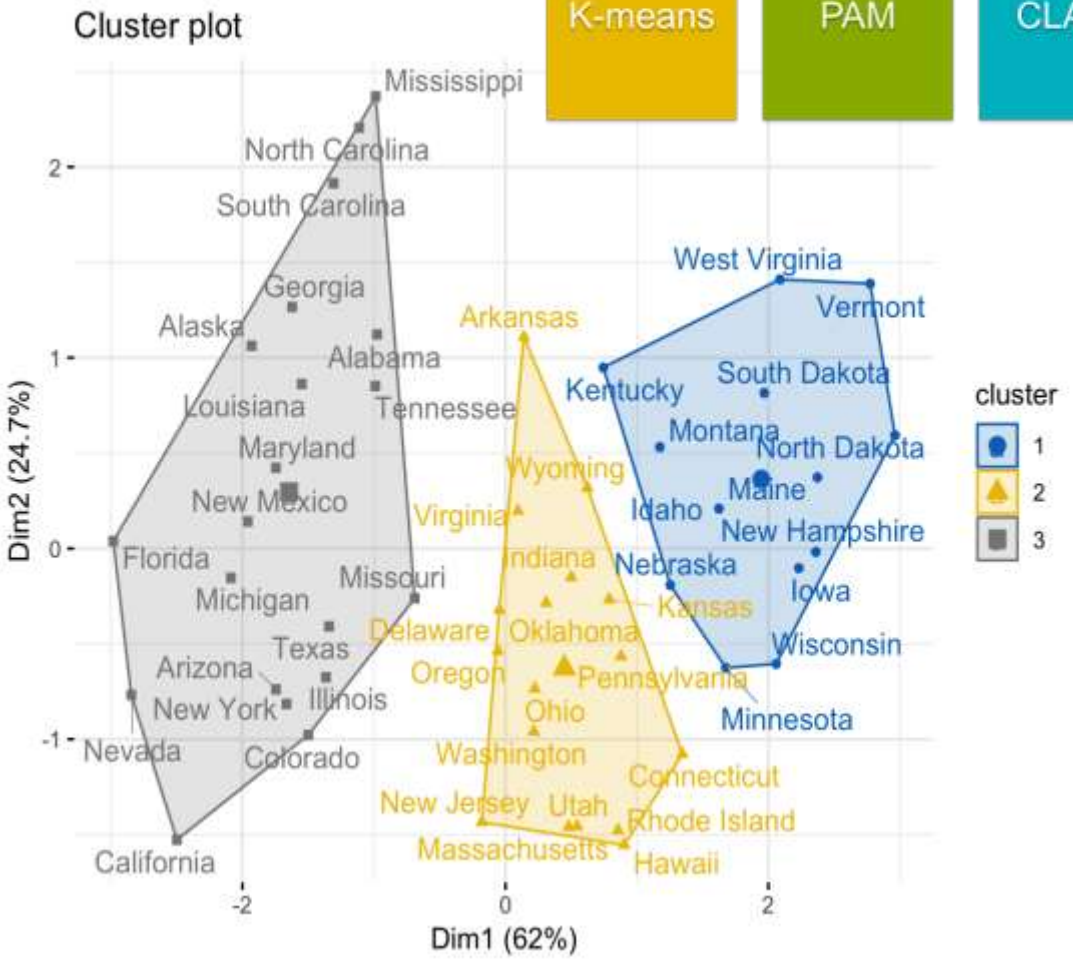
- Objective of clustering is to **find different groups** within **the elements in the data**.
- Clustering algorithms find **the structure in the data** so that **elements of the same cluster (or group) are more similar** to each other than to those from different clusters.
- The **machine learning model** will be **<u>able to infer</u>** that **there are two different classes without knowing anything else from the data**.



Clustering : Types

- ## Partitional Clustering
- Given a **database of n objects or data tuples**, a **partitioning method** constructs **k partitions of the data**, where ***each partition represents a cluster*** and $k \leq n$. That is, it classifies the data into k groups, which together satisfy the following requirements
  - **Each group** must contain at **least one object**,
  - **Each object** must **belong to exactly one group**.
  - **Given k**, the **number of partitions** to **construct**, a partitioning method **creates an initial partition**.
  - It uses **iterative relocation technique** that attempts to improve the **partitioning by moving objects** from **one group to another**.
  - The general criterion of a good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart" or very different.

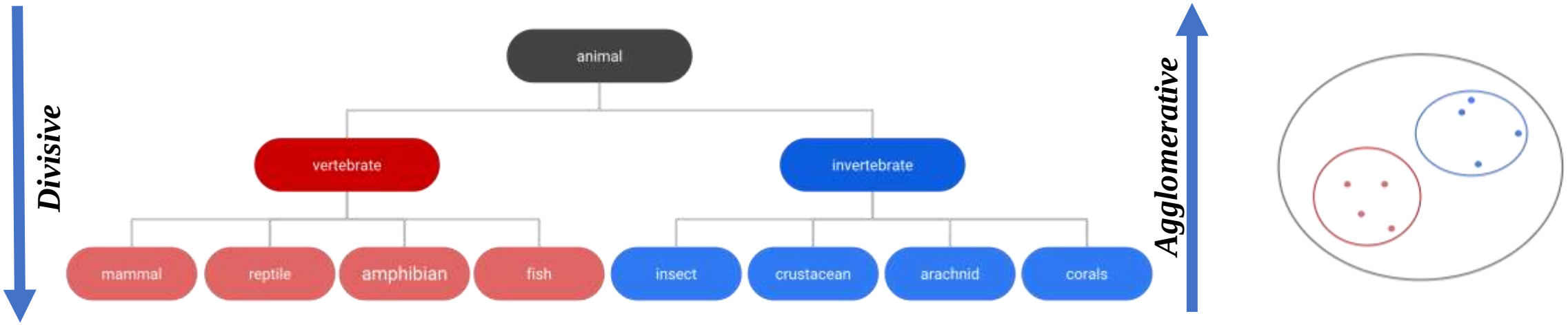




Clustering : Types

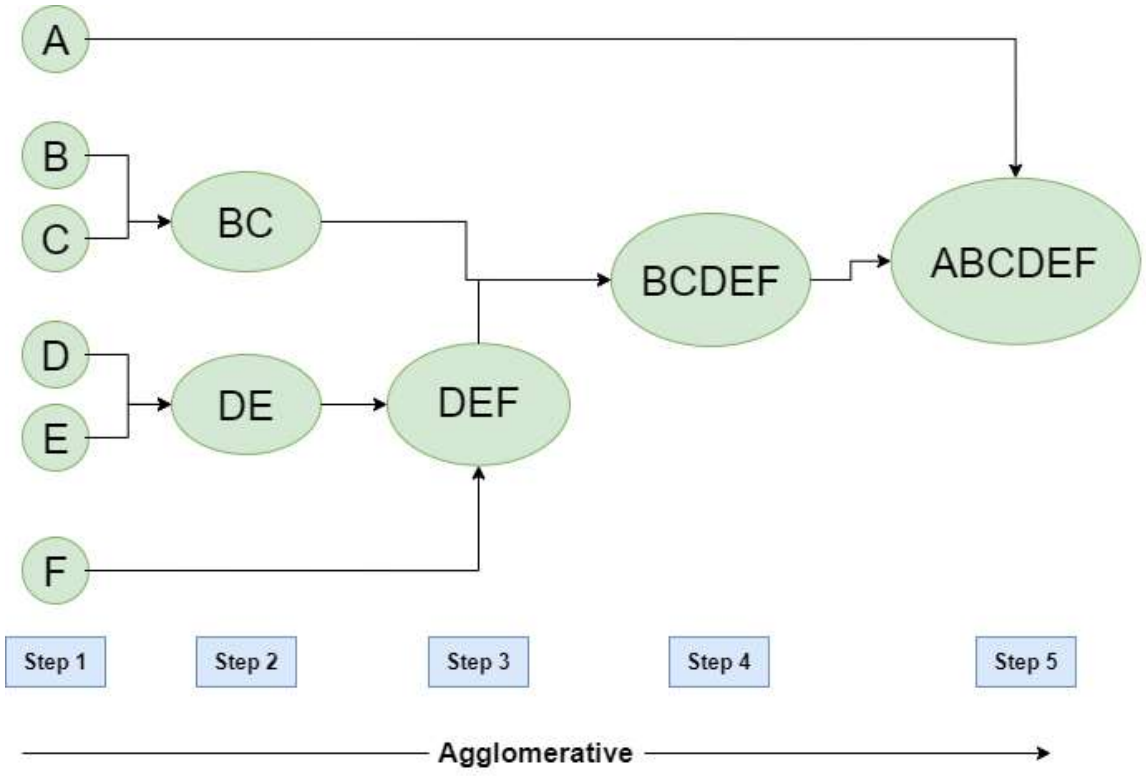
- **Hierarchical clustering**
  - Works via **grouping data into a tree of clusters**.
  - **Begins** by **treating every data point** as a **separate cluster**.
  - Then, it **repeatedly executes the subsequent steps**:
    - **Identify the 2 clusters** which can be **closest together**, and
    - **Merge/Split the 2 maximum comparable clusters**.
    - We need to continue these steps until all the clusters are merged/split together.

- In Hierarchical Clustering, the **aim is to produce a hierarchical series of nested clusters**.
  - A diagram called **Dendrogram** (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view).



Clustering : Types

- **Agglomerative Clustering:**
  - Also known as **bottom-up approach** or **hierarchical agglomerative clustering** (HAC).
  - A structure that is ***more informative*** than the ***unstructured set of clusters*** returned by ***flat clustering***.
  - This clustering algorithm **does not require** us to **prespecify the number of clusters**.
  - Bottom-up algorithms **treat each data as a singleton cluster** at the **outset** and then **successively agglomerates** pairs of clusters until all clusters have been merged into a single cluster that contains all data.

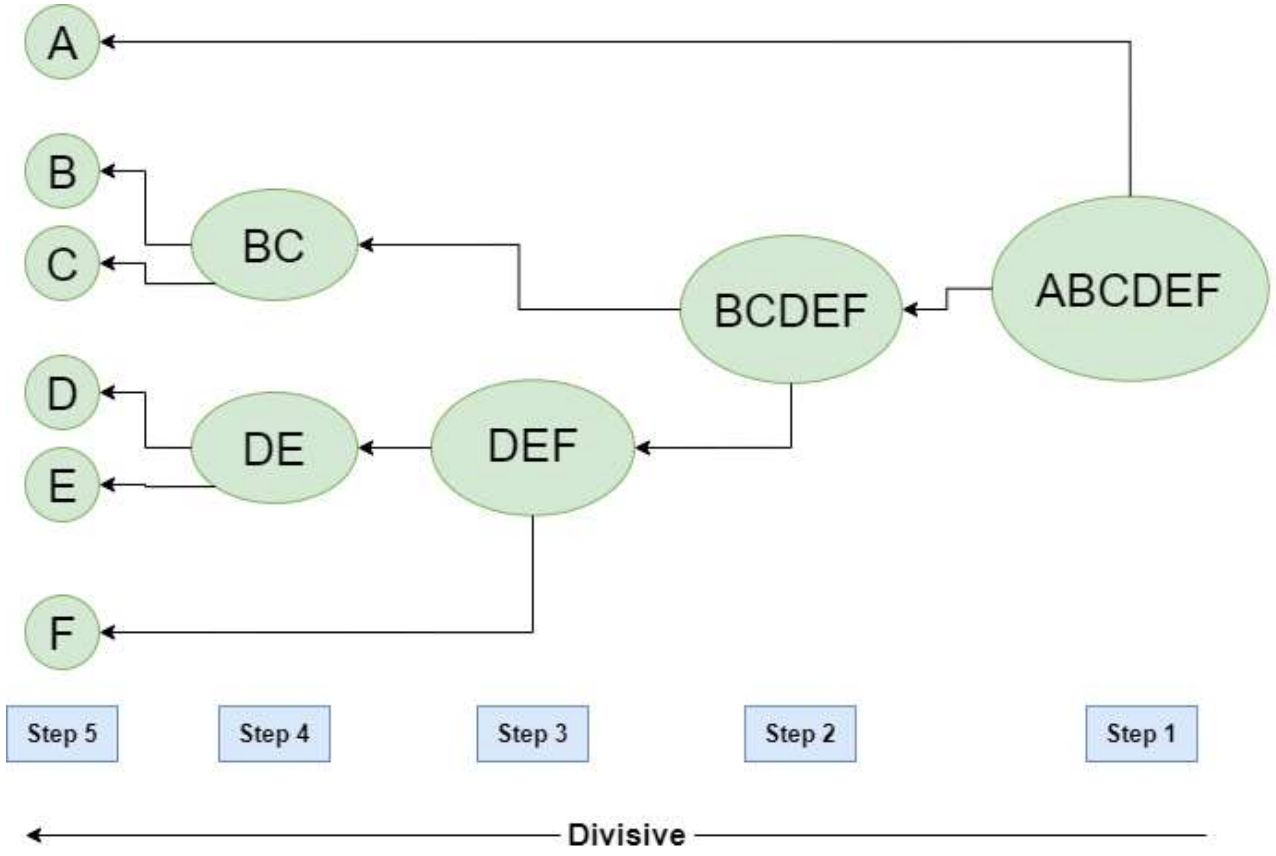


Algorithm

- Calculate the **similarity of one cluster** with **all the other clusters** (calculate proximity matrix)
- Consider every data point as an individual cluster
- Merge the clusters which are highly similar or close to each other.
- Recalculate the proximity matrix for each cluster
- Repeat Steps 3 and 4 until only a single cluster remains.

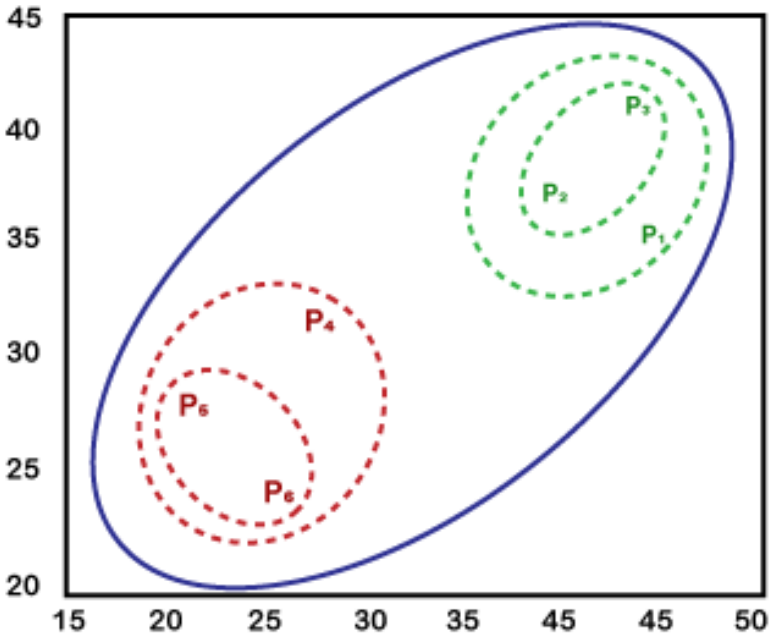
Clustering : Types

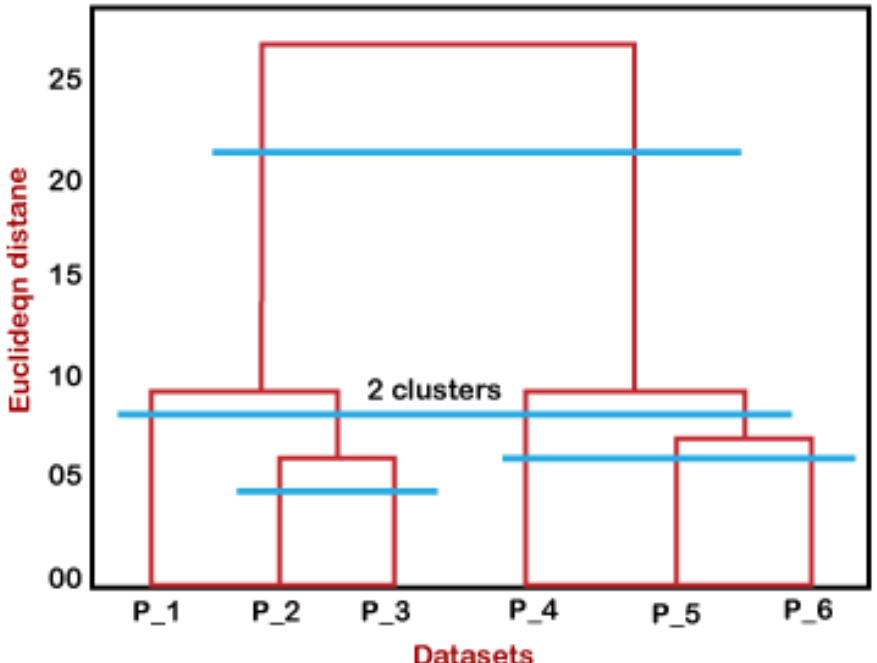
- **Divisive Clustering:**
  - Also known as a **top-down approach**.
  - This algorithm also does not require to prespecify the number of clusters.
  - Top-down clustering requires a method for splitting a cluster that contains the whole data and ***proceeds by splitting clusters recursively until individual data have been split into singleton clusters***.



Clustering : Types

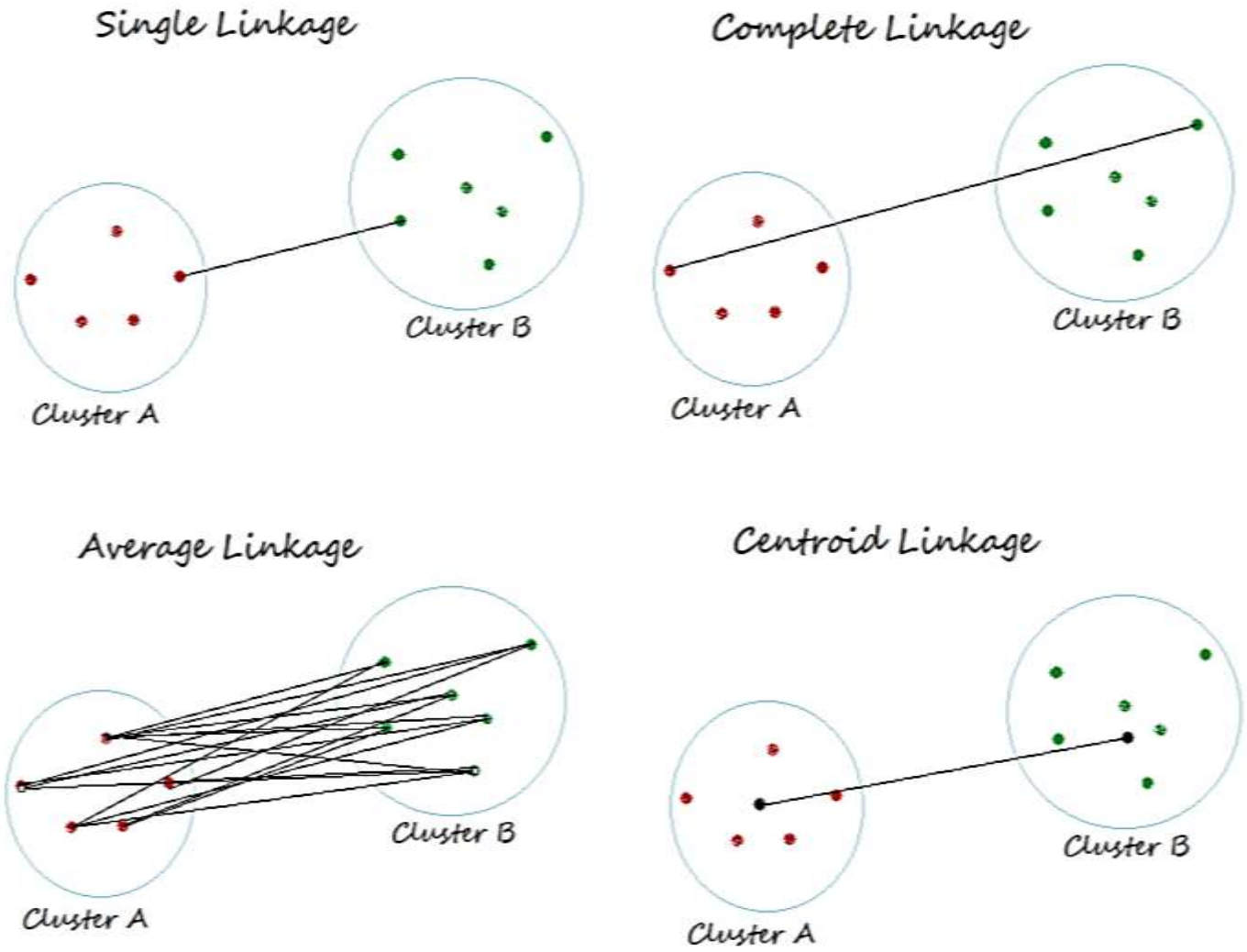
- **Hierarchical Clustering:**
- Process involves **either clustering sub-clusters** (data points in the first iteration) **into larger clusters** in a **bottom-up manner** or **dividing a larger cluster** into **smaller sub-clusters** in a **top-down manner**.
- **During both** types of **hierarchical clustering**, the **distance between two sub-clusters** needs to be **computed**.
- The **different types of linkages** describe the different approaches to measure the distance between two sub-clusters of data points.



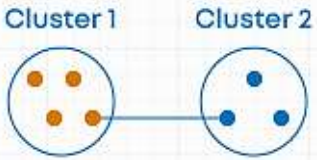


Clustering : Types

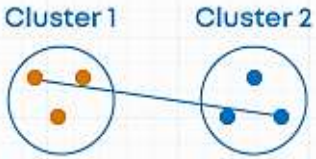
- **Hierarchical Clustering:** Linkage Types



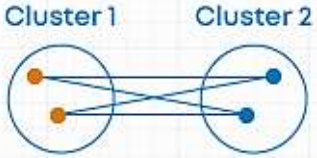
- **Single Linkage**
  $D(c_1,c_2) = \min D(x_1,x_2)$
  Minimum distance or distance between closest elements in clusters



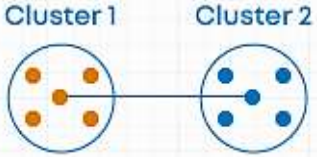
- **Complete Linkage**
  $D(c_1,c_2) = \max D(x_1,x_2)$
  Maximum distance between elements in clusters



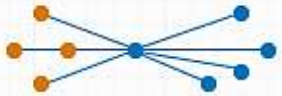
- **Average Linkage**
  $D(c_1,c_2) = \frac{1}{|c_1|}\frac{1}{|c_2|}\Sigma\Sigma D(x_1,x_2)$
  Average of the distances of all pairs



- **Centroid Method**
  Combining clusters with minimum distance between the centroids of the two clusters



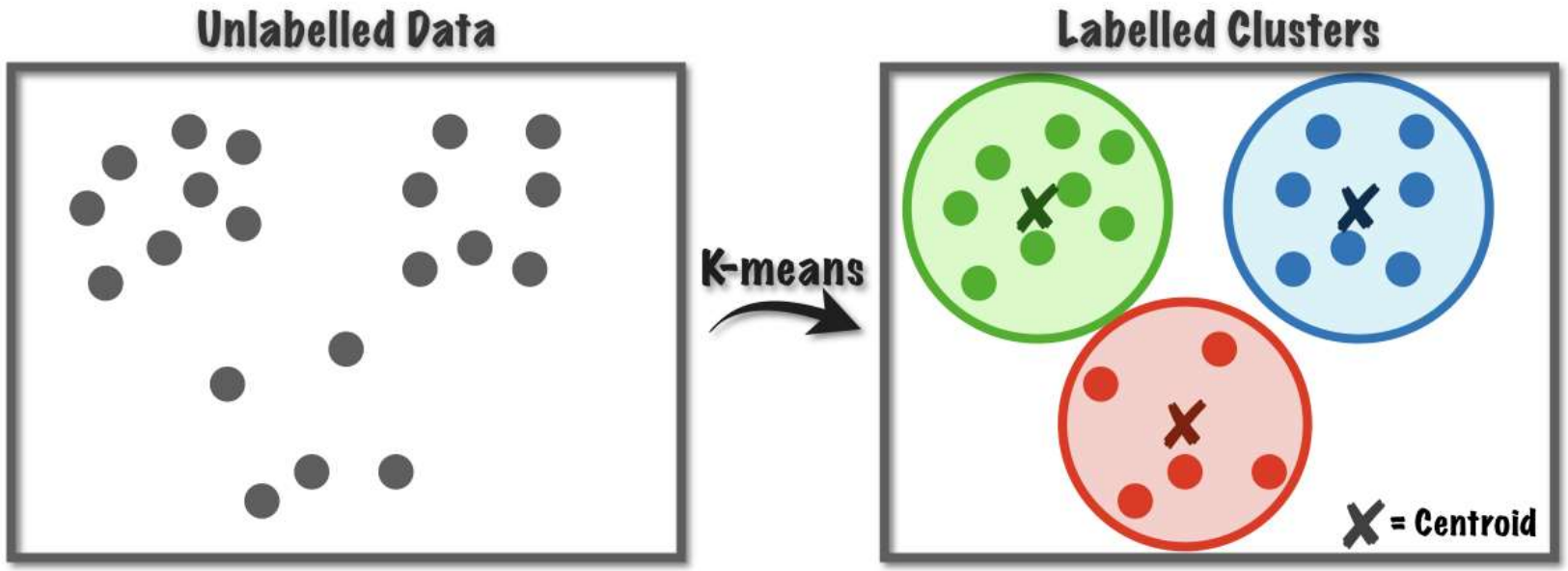
- **Ward's Method**
  - Combining clusters where increase in within cluster variance is to the smallest degree.



  - Objective is to minimize the total within cluster vairance

K Means Clustering

- K-Means algorithms aims to **find and group in classes** the **data points** that **have high similarity between them.**
- In the terms of the algorithm, this similarity is understood as the opposite of the distance between datapoints.
- The ***closer the data points are***, the ***more similar*** and ***more likely to belon***g to ***the same cluster*** they will be.



K Means Clustering

- **Squared Euclidean Distance:** Most commonly used distance in K-Means is the **squared Euclidean distance.** An example of this distance between two points x and y in m-dimensional space is:

$$d(\boldsymbol{x},\boldsymbol{y})^2 = \sum_{j=1}^{m}\left(x_j - y_j\right)^2 = \left\|\boldsymbol{x}-\boldsymbol{y}\right\|_2^2$$

j is the jth dimension (or feature column) of the sample points *x and y.*

- **Cluster Inertia:** Cluster inertia is the name given to the **Sum of Squared Errors** within the clustering context,

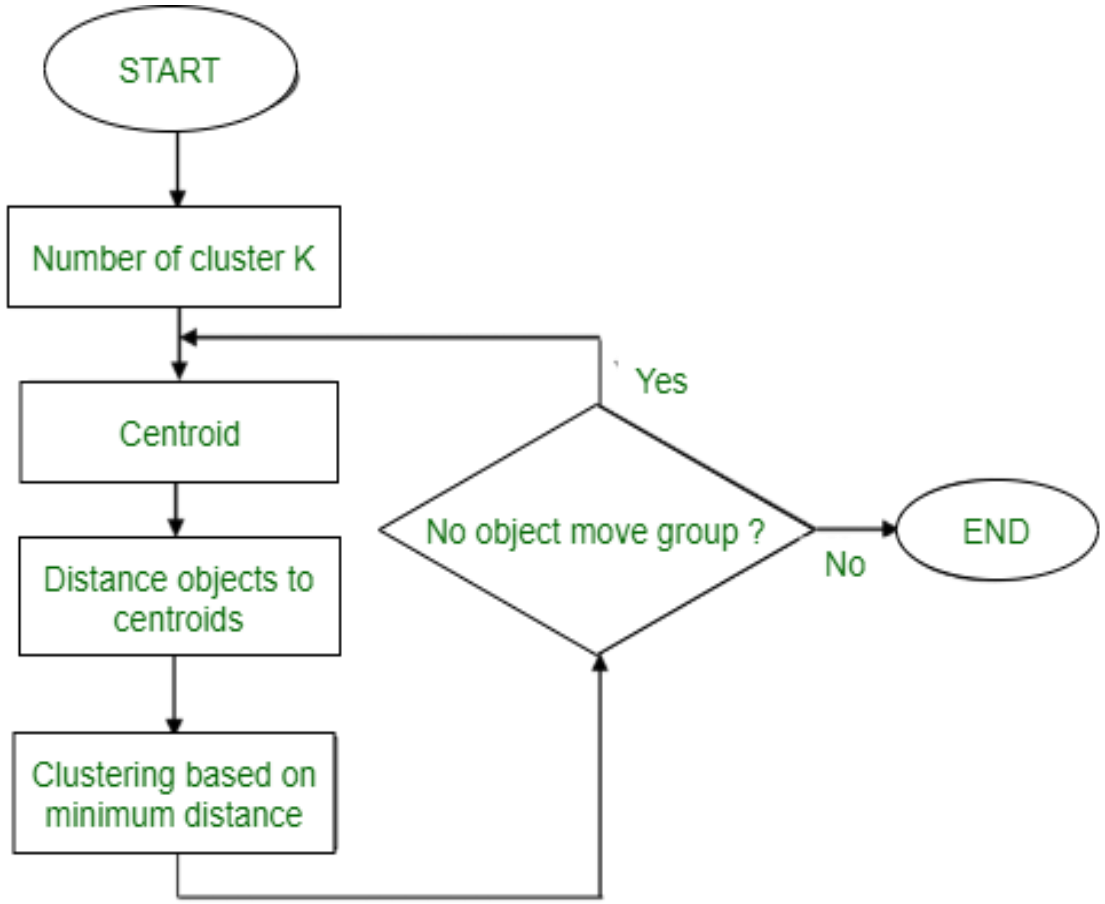
$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right\|_2^2$$

Where *μ(j)* is the centroid for cluster *j,* and *w(i,j)* is 1 if the sample *x(i)* is in cluster *j* and 0 otherwise.

- K-Means can be understood as an algorithm that will try to minimize the cluster inertia factor.

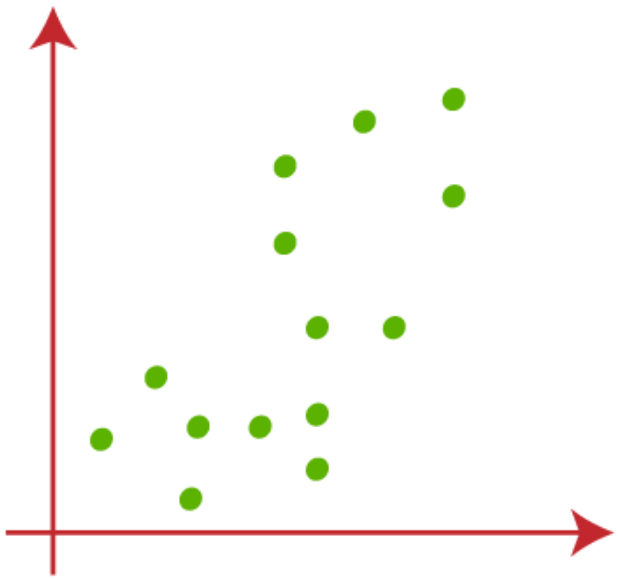
K Means Clustering

- First, we need to **choose k**, **the number of clusters** that we want to be find.
- Then, the algorithm will **select randomly** the **centroids** of **each cluster**.
- It will be **assigned each datapoint** to the **closest centroid** (using euclidean distance).
- It will be **computed** the **cluster inertia**.
- The ***new centroids*** will be ***calculated*** as the ***mean*** of the ***points*** that ***belong*** to the ***centroid*** of the ***previous step***.
- In other words, by **calculating the minimum quadratic error** of the datapoints to the **center of each cluster**, moving the center towards that point
- Back to step 3.

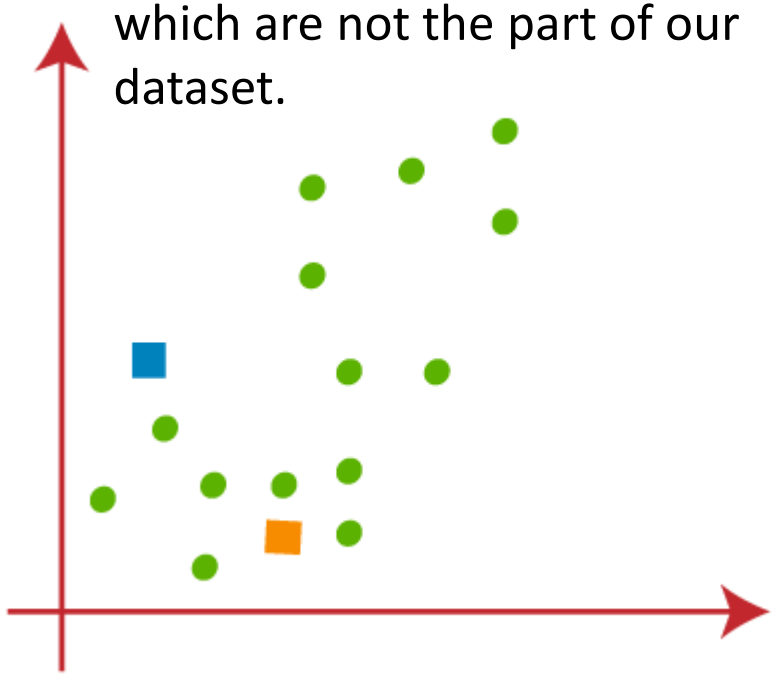


K Means Clustering

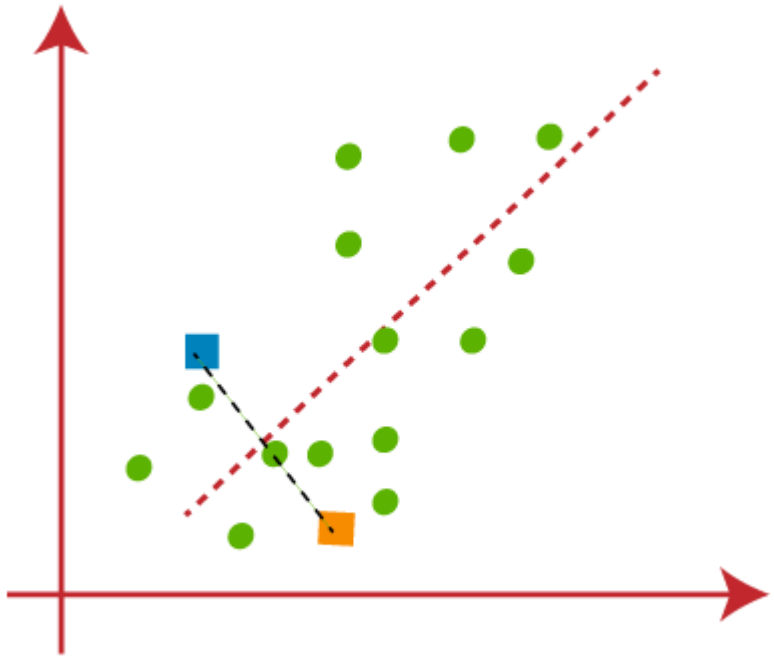
- Choose K=2, to identify the dataset and to put them into different clusters. It means to group these datasets into two different clusters.



- Choose **some random k points** or **centroid** to form the cluster.
- These points ***can be either the points from the dataset or any other point***.
- So, here we are selecting the below two points as k points, which are not the part of our dataset.



- Assign each data point of the scatter plot to its closest K-point or centroid.
- Calculate the distance between two points.
- So, we will draw a median between both the centroids.



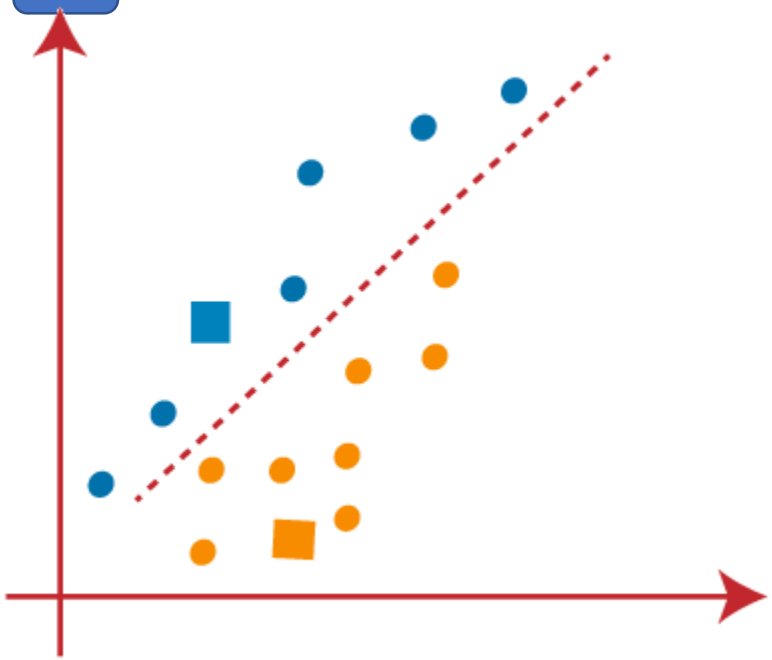
K Means Clustering

Points **left side of the line is near to the K1 or blue centroid**, and **points to the right of the line are close to the yellow centroid**

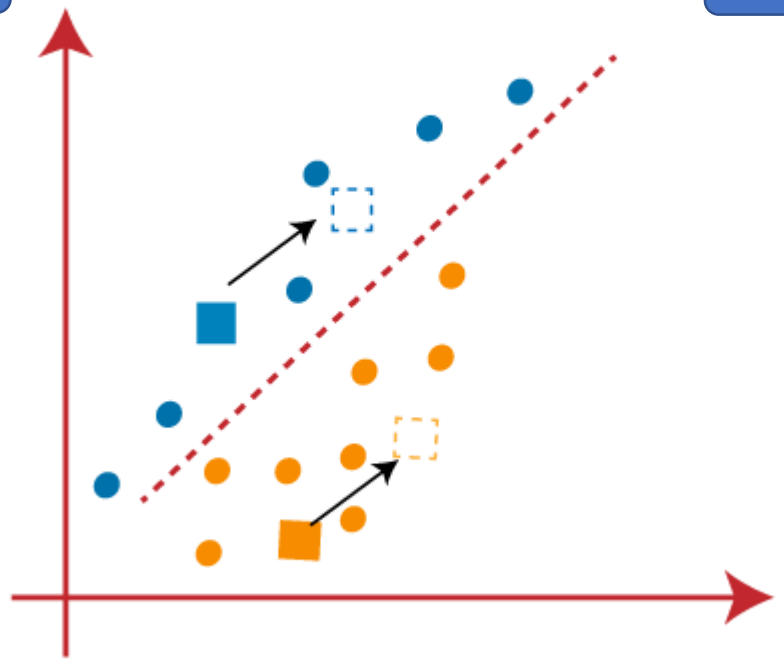
To find the closest cluster, **Repeat the process by choosing a new centroid.** To choose the new centroids, we will compute the center of gravity of these centroids.

Reassign **each datapoint to the new centroid**.
One yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

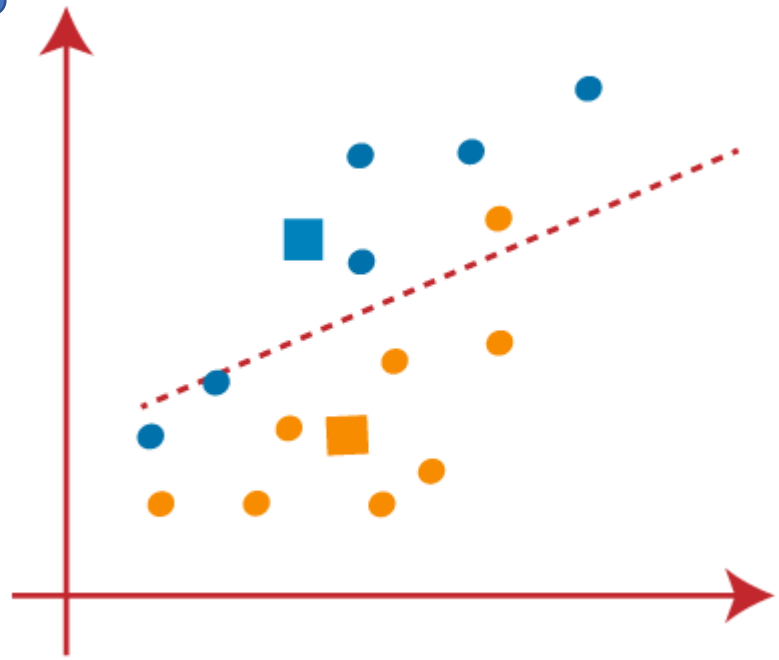
4



5



6



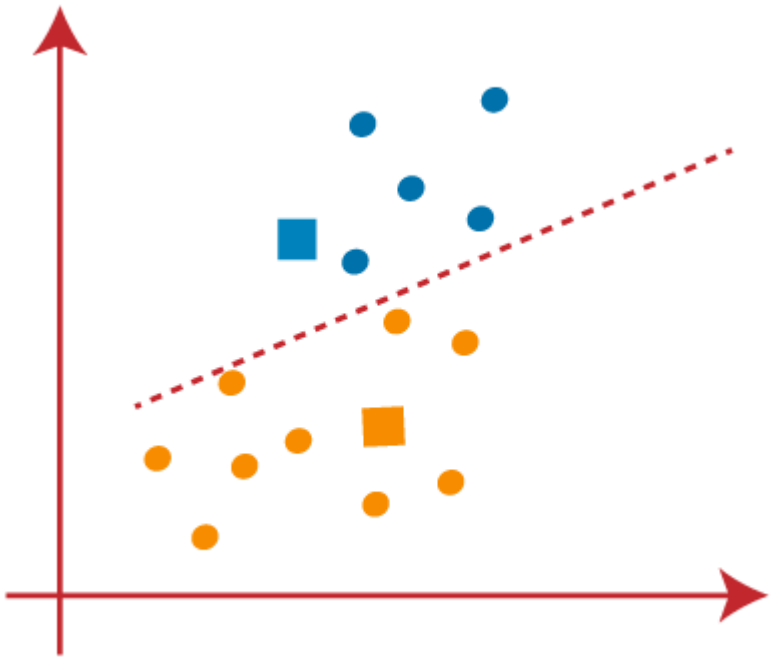
K Means Clustering

finding new **centroids or K-points**.

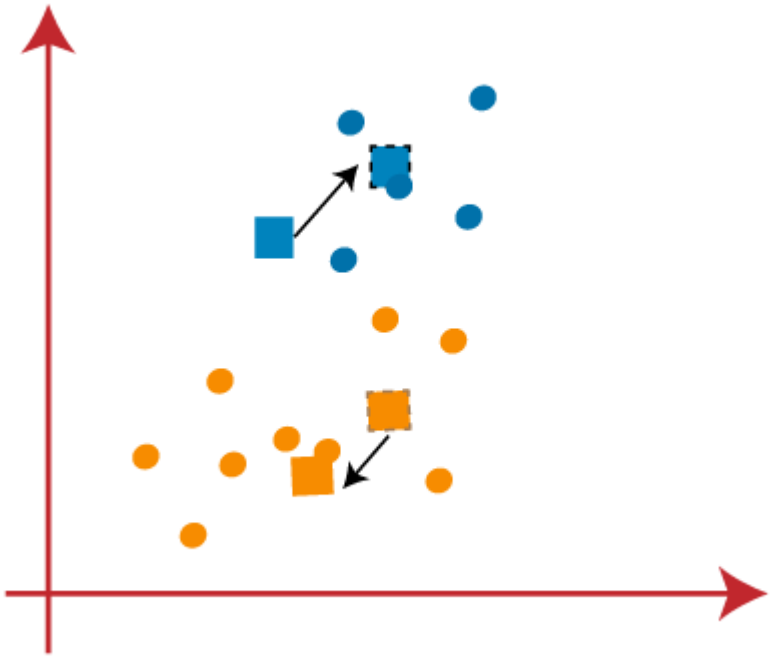
Compute the **center of gravity of these centroids** to find closest cluster.

New centroids so again will draw the median line and reassign the data points.

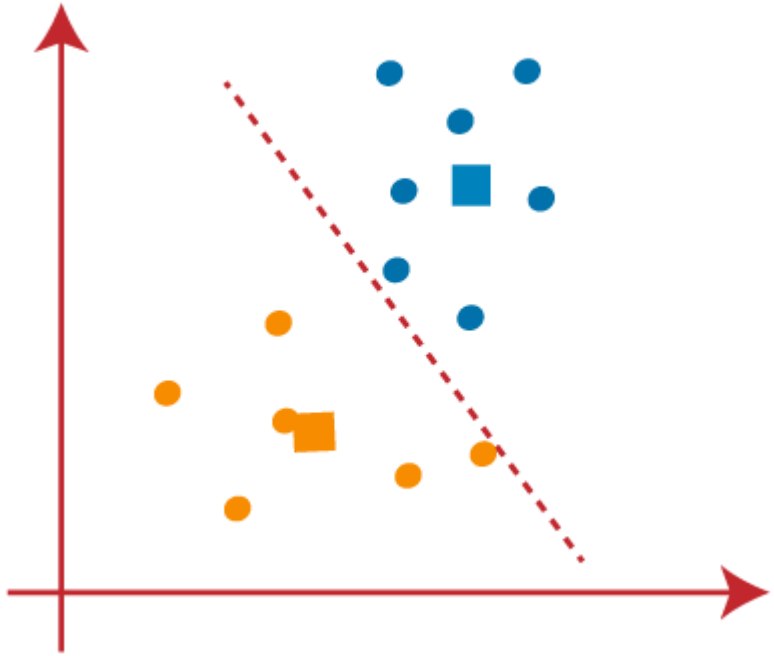
7



8



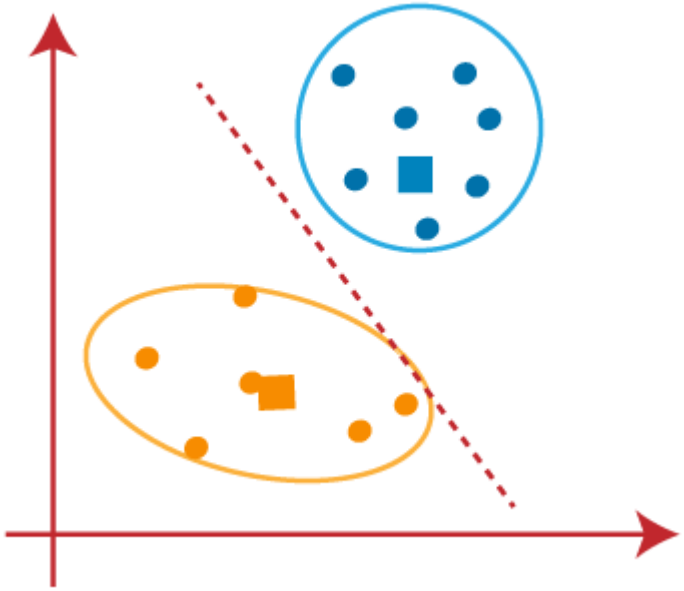
9



K Means Clustering

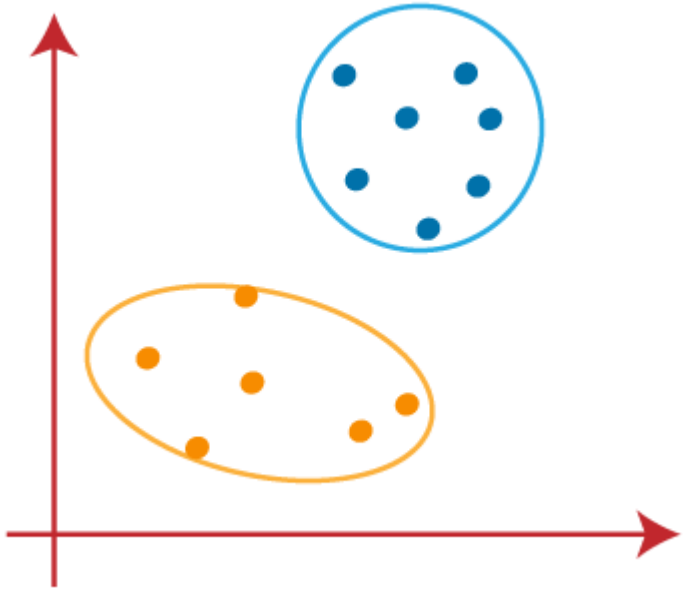
There are no dissimilar data points on either side of the line, which means our model is formed.

10



Remove the assumed centroids, and the two final clusters will be as shown in the below image:

11



Kmeans : Solved Example

Cluster the following eight points (with (x, y) representing locations) into three clusters:
$\boldsymbol{A1(2,10), A2(2,5), A3(8,4), A4(5,8), A5(7,5), A6(6,4), A7(1,2), A8(4,9)}$.
Initial cluster centers are:
$\boldsymbol{A1(2,10), A4(5,8)\ and\ A7(1,2)}$.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

Kmeans : Solved Example

Iteration-01: Distance of each point from each of the center of the three clusters.

The distance function between two points $a = (x1, y1)\ and\ b = (x2, y2)$ **using Manhattan Distance** is defined as-

$$\mathrm{P}(a, b) = |x2 - x1| + |y2 - y1|$$

- **For Cluster-01:**

  We have only one point A1(2, 10) in Cluster-01. So, cluster center remains the same.

- **For Cluster-02:**

  Center of Cluster-02
  = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
  = (6, 6)

- **For Cluster-03:**

  Center of Cluster-03
  = ((2 + 1)/2, (5 + 2)/2)
  = (1.5, 3.5)

C1={1}, C2={3,4,5,6,8}, C3={2,7}

Kmeans : Solved Example

Iteration-02: Distance of each point from each of the center of the three clusters using the given distance function.

Re-compute the new cluster clusters by the new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01: Center of Cluster-01
= ((2 + 4)/2, (10 + 9)/2)
= (3, 9.5)

For Cluster-02: Center of Cluster-02
= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)
= (6.5, 5.25)

For Cluster-03: Center of Cluster-03
= ((2 + 1)/2, (5 + 2)/2)
= (1.5, 3.5)

C1={1,8}, C2={3,4,5,6}, C3={2,7}

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

Kmeans

Advantages	Disadvantages
Relatively efficient and easy to implement.	Sensitive to Initialization.
Measurable and effective in larger data collection.	The selection of the initial centroids is random.
K-Means may produce tighter clusters than hierarchical clustering, especially if the clusters are globular.	Not suitable to discover clusters with non-convex shapes.
Terminates at a local optimum.	Limiting case of fixed data.
With a larger number of variables, K-Means may be computationally faster than hierarchical clustering.	Unable to handle noisy data or outliers.
Apply even large data sets.	Difficult to compare with different numbers of clusters.
The clusters are not hierarchical and they do not overlap.	Needs to specify the number of clusters in advance.

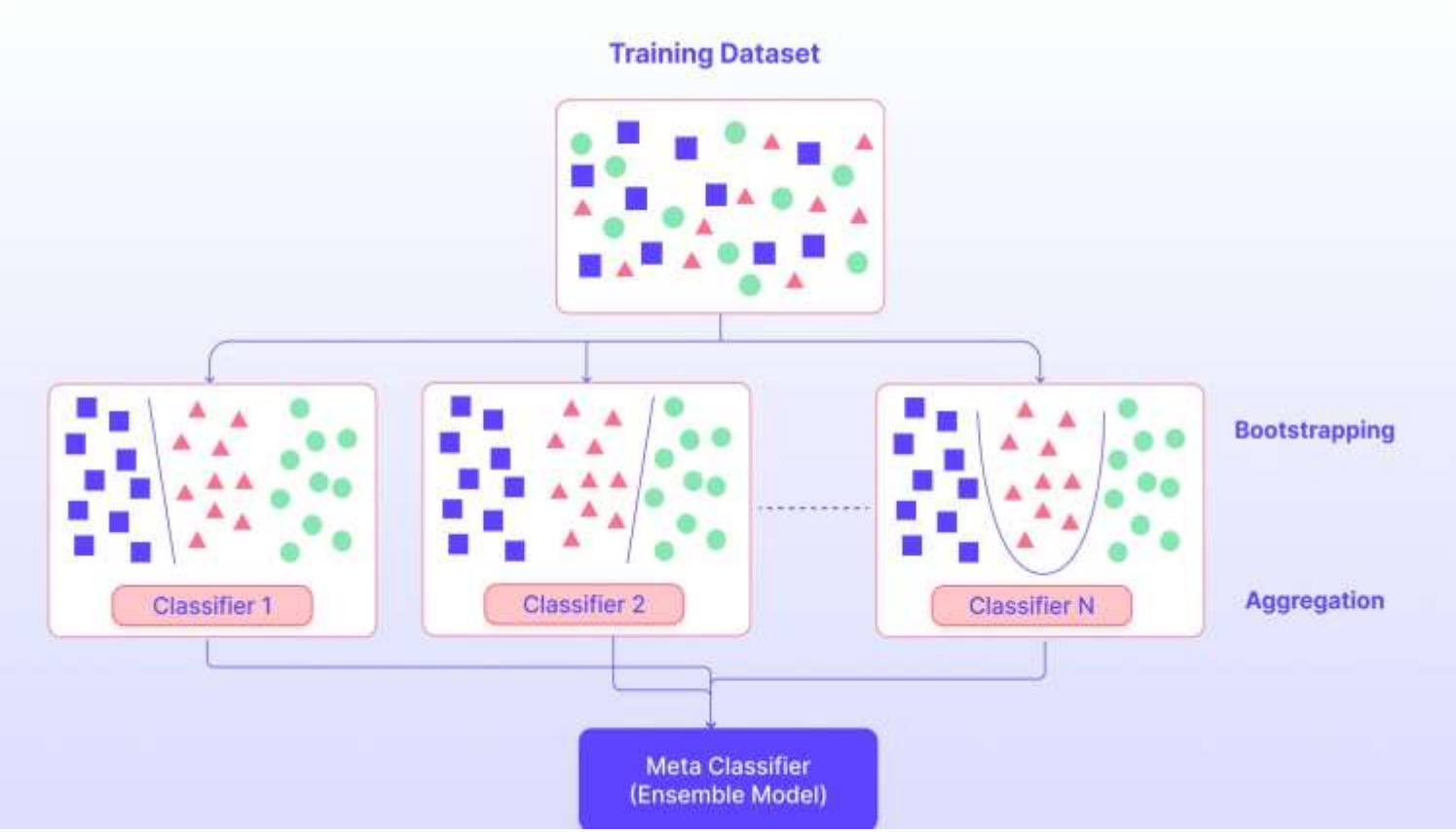
Applications

Ensemble learning

- Ensemble learning is '***a collection of things***'.
- A ML technique that **combines the predictions from multiple individual models** to obtain a **better predictive performance** than **any single model**.

- Basic idea behind ensemble learning is to **leverage the wisdom of the crowd** by **aggregating the predictions of multiple models**, each of which may have its own strengths and weaknesses.

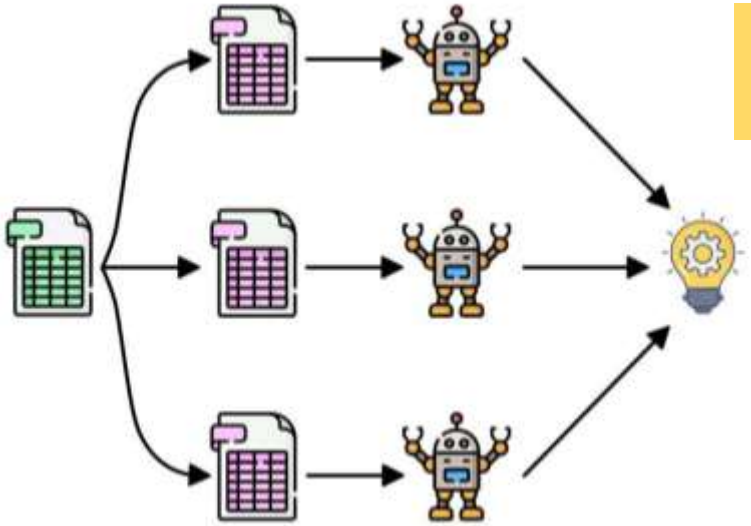
- This can lead to improved performance and generalization.



- **It implements an ensemble of fast algorithms (classifiers) such as decision trees for learning and allows them to vote.**

Ensemble learning Algorithms : Bagging

- **Bootstrap Sampling:**
  - Divides the **original training data into 'N' subsets** and **randomly selects a subset with replacement** in some rows from other subsets.
  - This step ensures that the **base models are trained on diverse subsets of the data** and there **is no class imbalance**.
- **Base Model Training:**
  - For each bootstrapped sample**, train a base model independently** on that subset of data. These **weak models are trained in parallel** to **increase computational efficiency** and reduce time consumption.
- **Prediction Aggregation:**
  - To **make a prediction** on **testing data combine the predictions of all base models**.
  - For classification tasks, it can include **majority voting** or **weighted majority while for regression**, it involves averaging the predictions.
- **Out-of-Bag (OOB) Evaluation:**
  - Some samples are excluded from the training subset of particular base models during the bootstrapping method.
  - These "***out-of-bag***" ***samples*** can be used to ***estimate*** the ***model's performance*** without the ***need for cross-validation***.
- **Final Prediction:**
  - After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

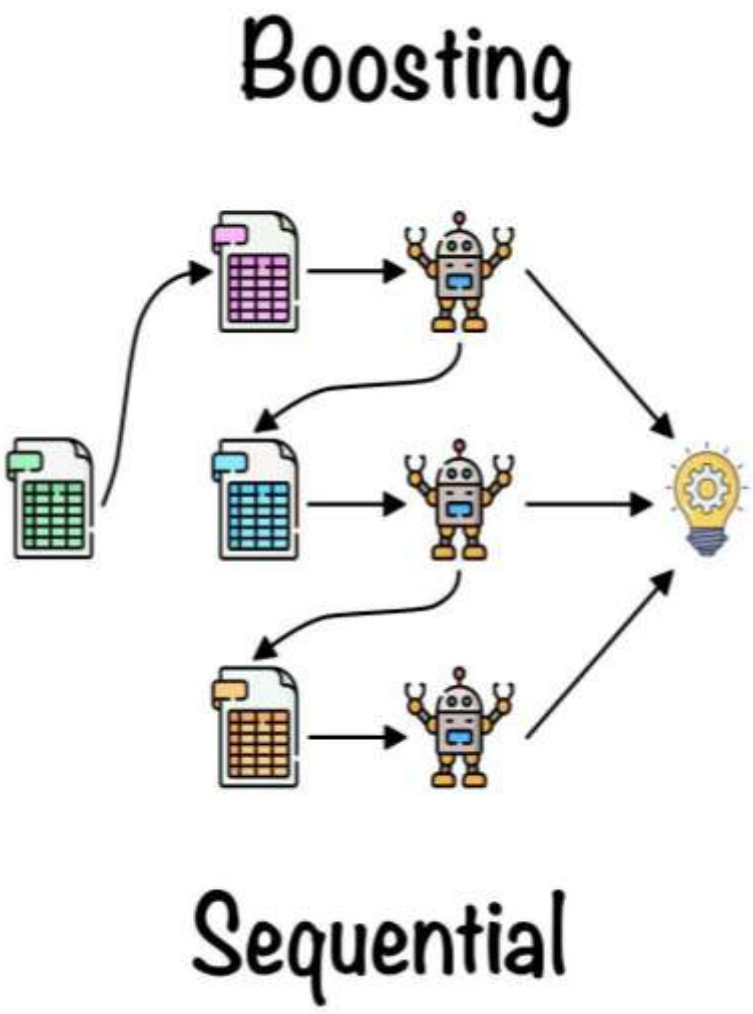


Bagging: Parallel

Ensemble learning Algorithms - Boosting

- Boosting is an ensemble technique that combines **multiple weak learners to create a strong learner.**
- The **ensemble of weak models** are **trained in series** such that ***each model that comes next***, ***tries to correct errors of the previous model*** until the entire training dataset is predicted correctly.

- **Boosting algorithm frameworks:**
- **AdaBoost (Adaptive Boosting):**
  - AdaBoost assigns **different weights to data points**, focusing on challenging examples in each iteration.
  - It combines weighted weak classifiers to make predictions.
- **Gradient Boosting:**
  - Gradient Boosting, including algorithms like Gradient Boosting Machines (GBM), XGBoost, and LightGBM, ***optimizes a loss function by training*** a ***sequence of weak learners*** to ***minimize the residuals between predictions and actual values***, producing strong predictive models.



How to Stack Estimators for a Classification Problem?

- **Diversify Your Choice of Base Models:**
  - Start by choosing a **diverse mix of base classifiers** for your classification task.
  - These may include ***decision trees, support vector machines, random forests, logistic regression, or other suitable algorithms***.
  - Varying your base models can often lead to a more robust stacking result.
- **Data partitioning:**
  - Split your labeled data set into at least two parts: a **training set** and a **separate validation** set.
  - You will use the training set to train your base models, and the ***validation set*** helps ***generate new features based on their predictions***.
- **Training base models :**
  - Train each base model in the training set. It is important to ***ensure that all base models are trained*** on the ***same set of features*** and ***labels*** for consistency.
- **Generate predictions :**
  - Make predictions on the validation set using the base models you have trained.
  - These **predictions become new features** for the next steps.
- **Developing a meta-learner**:
  - Select a **meta-learner**, e.g **logistic regression**.
- **Final Inferences:**
  - **Once the meta-learner is trained**, use it to **make final inferences from new and unseen data**.
  - Combine predictions from the base model to create input features for the meta-learner.
- **Evaluate and refine:**
  - Assess the performance of your stacked ensemble using various evaluation metrics, such as accuracy, precision, recall, or F1-score.
  - Fine-tune the group by adjusting the meta-teacher or considering additional foundation models as needed.

Uses of Ensemble Learning

- **Classification and Regression:**
  - Ensemble techniques make problems like classification and regression versatile in various domains, including finance, healthcare, marketing, and more.
- **Anomaly Detection:**
  - Ensembles can be used to detect anomalies in datasets by combining multiple anomaly detection algorithms, thus making it more robust.
- **Portfolio Optimization:**
  - Ensembles can be employed to optimize investment portfolios by collecting predictions from various models to make better investment decisions.
- **Customer Churn Prediction:**
  - In business and marketing analytics, by combining the results of various models capturing different aspects of customer behaviour, ensembles can be used to predict customer churn.
- **Medical Diagnostics:**
  - In healthcare, ensembles can be used to make more accurate predictions of diseases based on various medical data sources and diagnostic models.
- **Credit Scoring:**
  - Ensembles can be used to improve the accuracy of credit scoring models by combining the outputs of various credit risk assessment models.
- **Climate Prediction:**
  - Ensembles of climate models help in making more accurate and reliable predictions for weather forecasting, climate change projections, and related environmental studies.
- **Time Series Forecasting:**
  - Ensemble learning combines multiple time series forecasting models to enhance accuracy and reliability, adapting to changing temporal patterns.

Case-studies - Machine learning in Signal processing

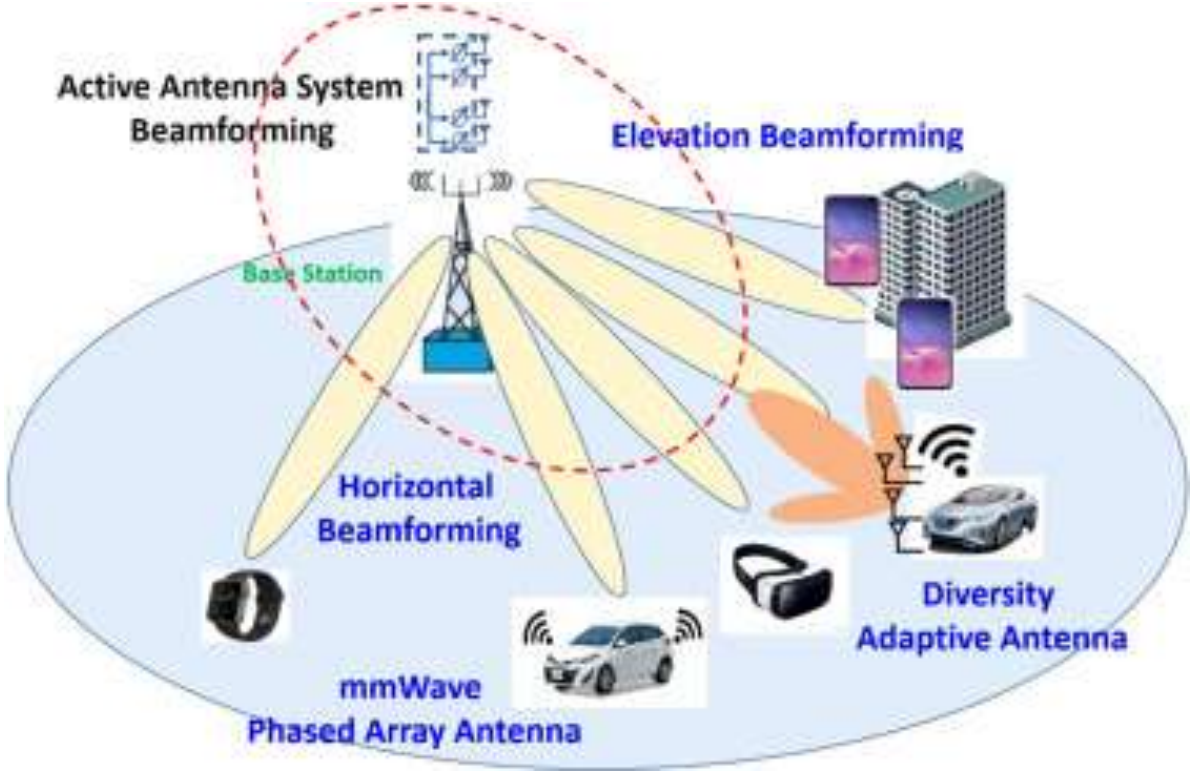
- **Speech Emotion Recognition and Music Genre Classification:** Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks- learn temporal dependencies in audio data, allowing them to effectively handle sequences of varying lengths.
  - Feature extraction techniques, such as Mel-frequency spectral coefficients (MFCCs), are used to convert raw audio signals into a format suitable for model training.
  - Modern applications, like Google Assistant and Apple's Siri, utilize these advanced models to provide highly accurate and context-aware speech recognition.
- **Heart Disease Diagnosis Using ECG Signals:** Machine learning classifiers like SVM and decision trees were trained on features extracted from ECG signals. A dataset containing normal and abnormal ECG signals was used.
- **Fault Detection in Industrial Equipment Using Vibration Signals:** Time-series analysis and machine learning algorithms, including Random Forests and LSTM networks, were utilized to analyze vibration signals from machinery.

Case-studies - Machine learning in Signal processing

- **Modulation and Phase Recovery:** In communication systems, recovering original signals from modulated versions often presents challenges due to unknown phase shifts. Machine learning algorithms can analyze modulated signals to estimate phase values effectively, facilitating accurate signal reconstruction
- **Radar and Sonar Signal Processing:** Machine learning models, particularly ensemble methods- to improve target detection and classification in military and maritime operations by processing high-dimensional data and distinguishing between legitimate targets and noise.
- **Real-Time Language Translation:** A combination of Automatic Speech Recognition (ASR) and machine translation models was used, leveraging sequence-to-sequence learning with attention mechanisms.
- **Time Series Forecasting:** Machine learning models, particularly Long Short-Term Memory (LSTM) networks and other recurrent architectures, are well-suited for time series forecasting due to their ability to learn long-term dependencies. These models can handle non-linear relationships and incorporate exogenous variables, improving forecast accuracy.

Case-studies : Intelligent Antenna

- **Smart Antennas for Wireless Communication:** Researchers implemented adaptive beamforming techniques using smart antennas that can dynamically adjust their radiation patterns based on the location of users. Techniques like Maximum Signal-to-Noise Ratio (MSNR) and Minimum Mean Square Error (MMSE) were utilized.
- **Intelligent Antennas in 5G Networks:** Intelligent antennas were integrated with Massive MIMO technology, enabling dynamic resource allocation through user scheduling and beam management algorithms.



- **Smart Antennas for Satellite Communication**: Adaptive antennas with beamforming capabilities were utilized to dynamically track satellites, optimizing signal reception from various angles, and machine learning algorithms were utilized for predictive modeling.

Case-studies–Intelligent Antenna

- **Intelligent Antennas in Automotive Applications:** Intelligent antennas were integrated into vehicle systems for improved V2V and V2I communications, using adaptive beamforming techniques for enhanced signal strength and reliability.
- **Smart Antennas for IoT Applications:** IoT networks have been equipped with intelligent antennas to enhance coverage, reduce power consumption, and minimize interference using techniques like spatial diversity and adaptive beamforming.
- **Intelligent Antennas in Healthcare Monitoring Systems:** Smart antennas are utilized in telemedicine applications to improve signal reception from wearable health monitoring devices, with adaptive algorithms prioritizing critical health data transmission.