ENSEMBLE, CLUSTERING & NEURAL NETWORK

4th Sem, DSE (A)

ENSEMBLE

- Ensemble learning: multiple models combined to improve performance of overall system.
- Create diverse set of models that can complement each other's strengths and weaknesses.
- Models can be trained on different subsets of training data or using different algorithms.
- Their outputs are combined to make final prediction.
- Ensemble methods improve accuracy and robustness of ML models.
- Commonly used in complex applications; image recognition, speech recognition, and natural language processing.

ENSEMBLE

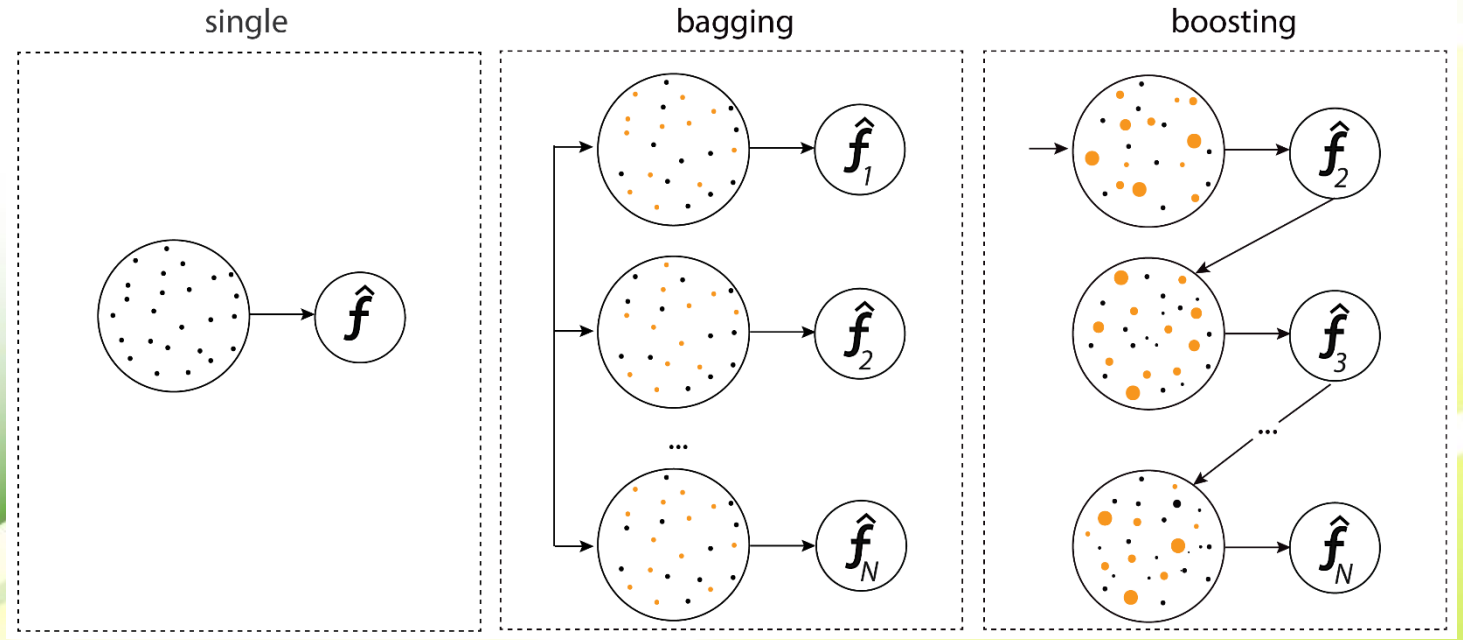
Types of ensemble methods:

- Bagging (Bootstrap Aggregating): involves creating multiple copies of original training data set.
  - each model trained on a different copy of training data set --> results in a collection of models that are less correlated with each other.
- Boosting: involves training a series of weak models sequentially.
  - each model learns from mistakes of previous model.
- Stacking: involves combining multiple models by training a meta-model that takes output of these models as inputs.
- Blending, Cascading

ENSEMBLE

Types of ensemble methods:

- Max Voting
- Averaging
- Weighted Averaging



- Parallel and Sequential
- Homogeneous and Heterogeneous
- Bagging:
  - Random forest
  - Bootstrap aggregation
- Boosting:
  - adaboost, gradientboost, xgboost, light GBM, catboost

ENSEMBLE

- Weak model in ensemble learning is a model that performs only slightly better than random guessing.
- Combining multiple weak models together creates a strong model that outperforms any individual model.
- Simple models with limited capacity.
- decision stumps (a decision tree with only one split) or linear models.
- models may suffer from high bias or low variance.
- each weak model is trained on a subset of training data, and final model is weighted sum of all weak models.
- weak models focus on different aspects of data and, when combined, produce a more accurate model.

ENSEMBLE

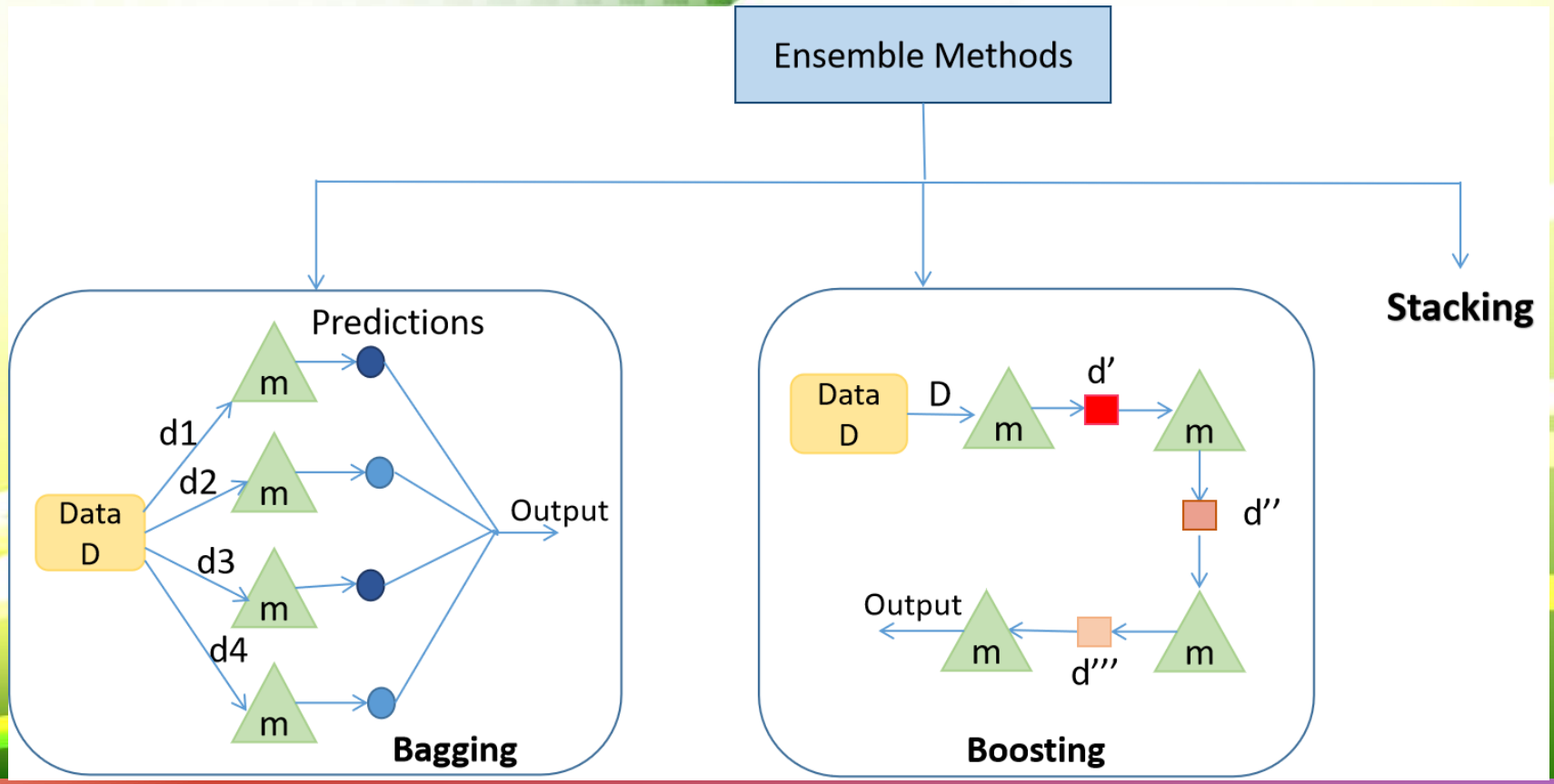
- weak learner is not very accurate on its own.
- model may not capture all complexities of underlying data.
- provides some useful information for an ensemble model.
- Examples of weak learners include decision stumps and simple linear models.
- linear model with a low number of features (linear regression model with only one or two predictor variables).
- decision stumps is a decision tree with only one split.
- binary classification problem to predict whether a person will purchase a particular product based on their age.
- A decision stump could be a model that predicts that all people below age of 30 will purchase the product, and all people above age of 30 will not purchase the product.
- This model is not very accurate on its own, but better than random guessing.

ENSEMBLE

- Epoch (In ML) refers to a single iteration through entire training dataset during model training.
- During each epoch, model is presented with each training example once and makes predictions based on current set of model parameters.
- During an epoch, model computes error on each training example.
- model then uses an optimization algorithm (example, stochastic gradient descent) to update model parameters to reduce error.
- This process of presenting training data and updating model parameters is repeated for a fixed number of epochs or until some convergence criteria are met.
- number of epochs used to train a model is a hyperparameter that can be tuned to find best balance between underfitting and overfitting.
- Underfitting occurs when model is too simple and fails to capture complexity of data.
- overfitting occurs when model is too complex and fits training data too closely, leading to poor performance on new, unseen data.

ENSEMBLE

- Base learners: first level of ensemble learning architecture,
  - each one of them is trained to make individual predictions.
- Meta learners: second level, and they are trained on output of base learners.



ENSEMBLE

Bagging:

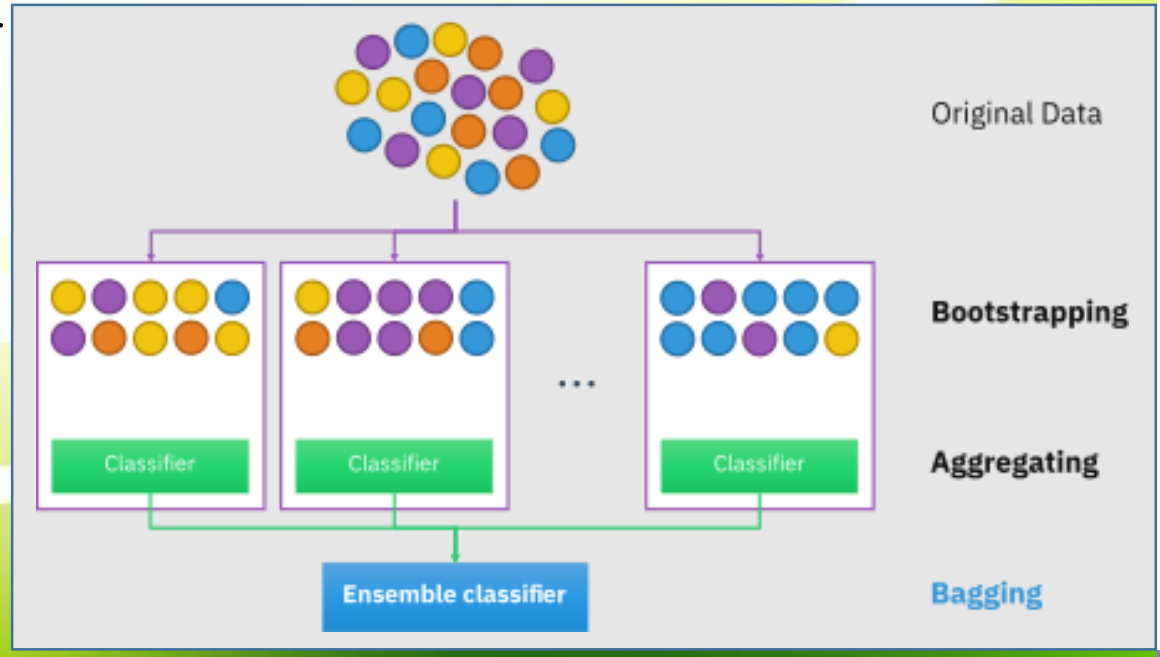
- random sample of data from training set is selected with replacement,
- This enables duplication of sample instances in a set.

Bagging steps:

- Generation of multiple bootstrap resamples.
- Running an algorithm on each resample to make predictions.
- Combining predictions by taking average of predictions (regression) or majority vote (classification).

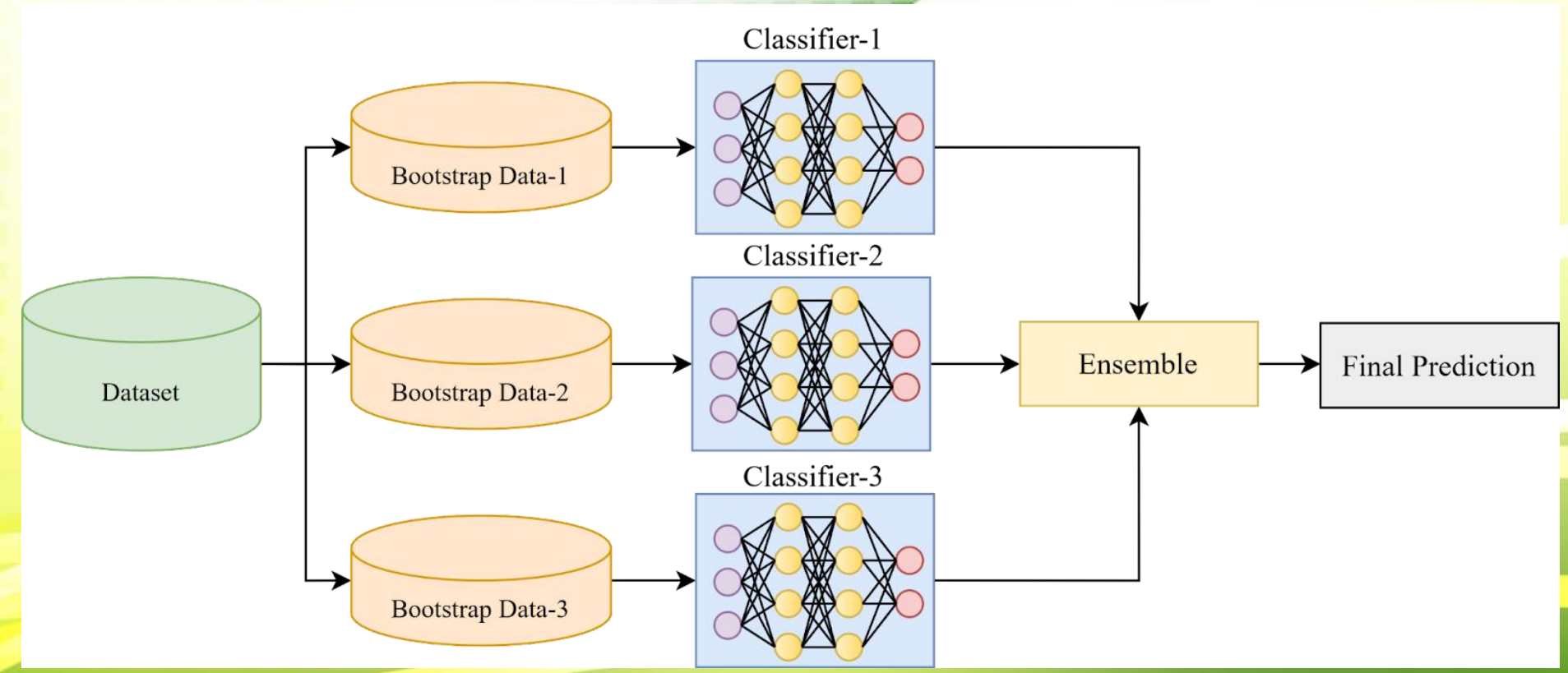
Random Forest

- Bootstrap Sampling is a method that involves drawing of sample data repeatedly with replacement from a data source to estimate a population parameter.
- Bagging (***Bootstrap Aggregation)*** is the ensemble technique used by random forest.
  - Bagging chooses a random sample from the data set.
  - Each model is generated from samples (Bootstrap Samples) provided by Original Data with replacement (***row sampling)***.
  - This step of row sampling with replacement is called ***bootstrap***.
  - Now each model is trained independently which generates results.
  - Final output is based on majority voting after combining all results.
  - This step which involves combining all results and generating output based on majority voting is known as ***aggregation***.



ENSEMBLE

- Bagging / Bootstrap Aggregation



ENSEMBLE

Advantages

- straightforward and does not require any deep mathematical concepts,
- can handle missing values.
- scikit-learn package makes it easy to implement underlying logic.
- significantly effects on reducing variance on high-variance classifiers.
- Incorporates new training data.

Disadvantages

- computationally expensive due to use of several models.
- averaging involved across predictions makes it difficult to interpret final result.

ENSEMBLE

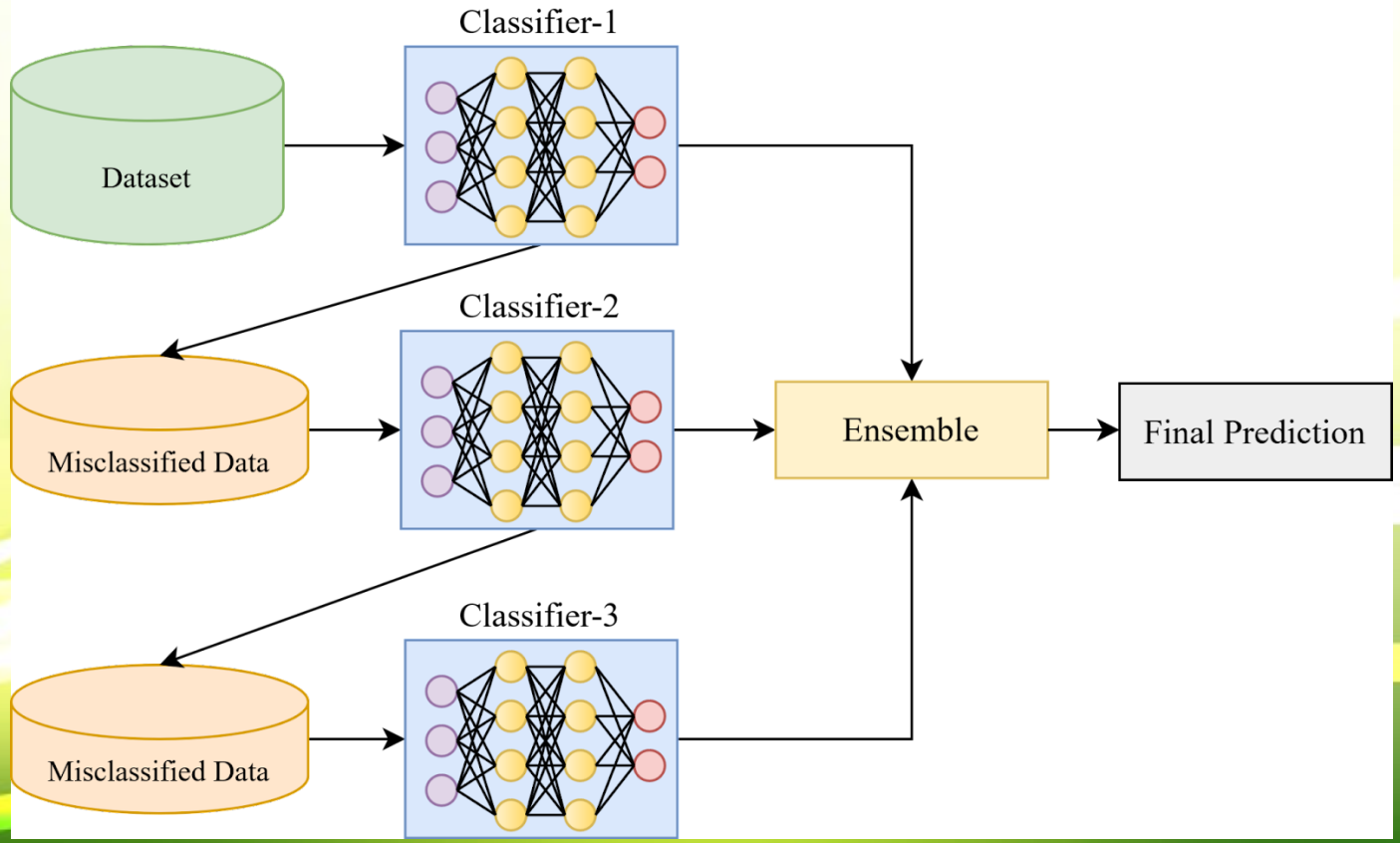
- Boosting is a sequential process.
- Each subsequent model attempts to correct errors of the previous model. The succeeding models are dependent on the previous model.

Boosting steps:

- A subset is created from the original dataset.
- Initially, all data points are given equal weights.
- A base model is created on this subset.
- This model is used to make predictions on the whole dataset.
- Errors are calculated using the actual values and predicted values.
- The observations which are incorrectly predicted, are given higher weights.
- Another model is created and predictions are made on the dataset. This model tries to correct the errors from the previous model.
- Similarly, multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).

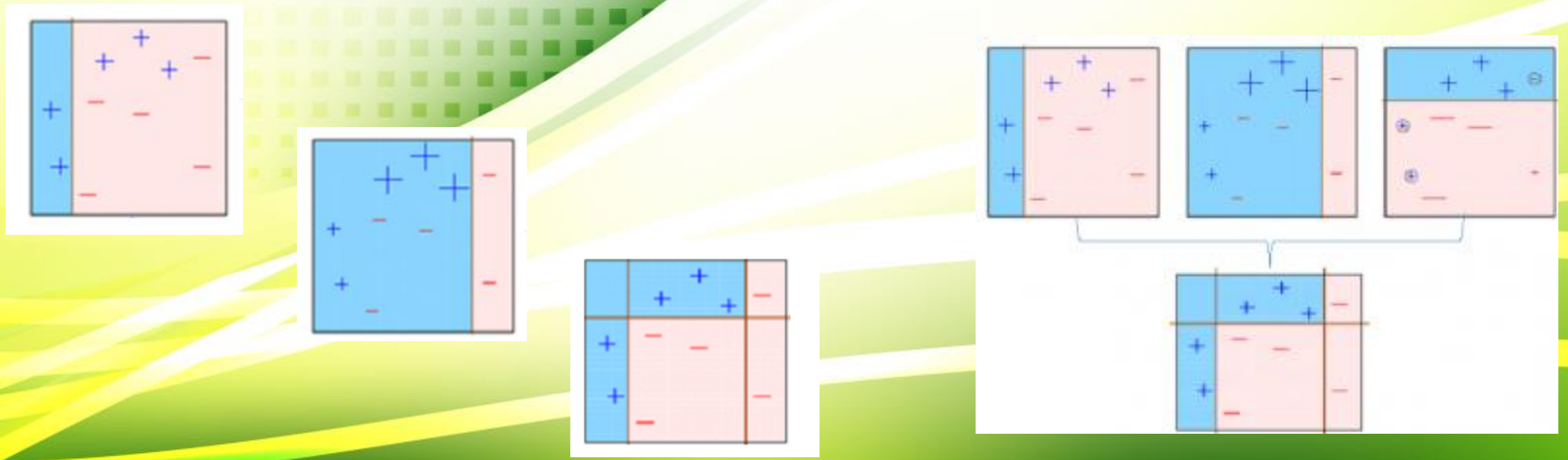
ENSEMBLE

- Boosting adopts sequential approach, where prediction of current model is transferred to next one.
- Each model iteratively focuses attention on observations that are misclassified by its predecessors.



ENSEMBLE

- Boosting algorithm combines a number of weak learners to form a strong learner.
- Individual models would not perform well on entire dataset, but they work well for some part of the dataset.
- Each model actually boosts the performance of the ensemble.



ENSEMBLE

Advantages

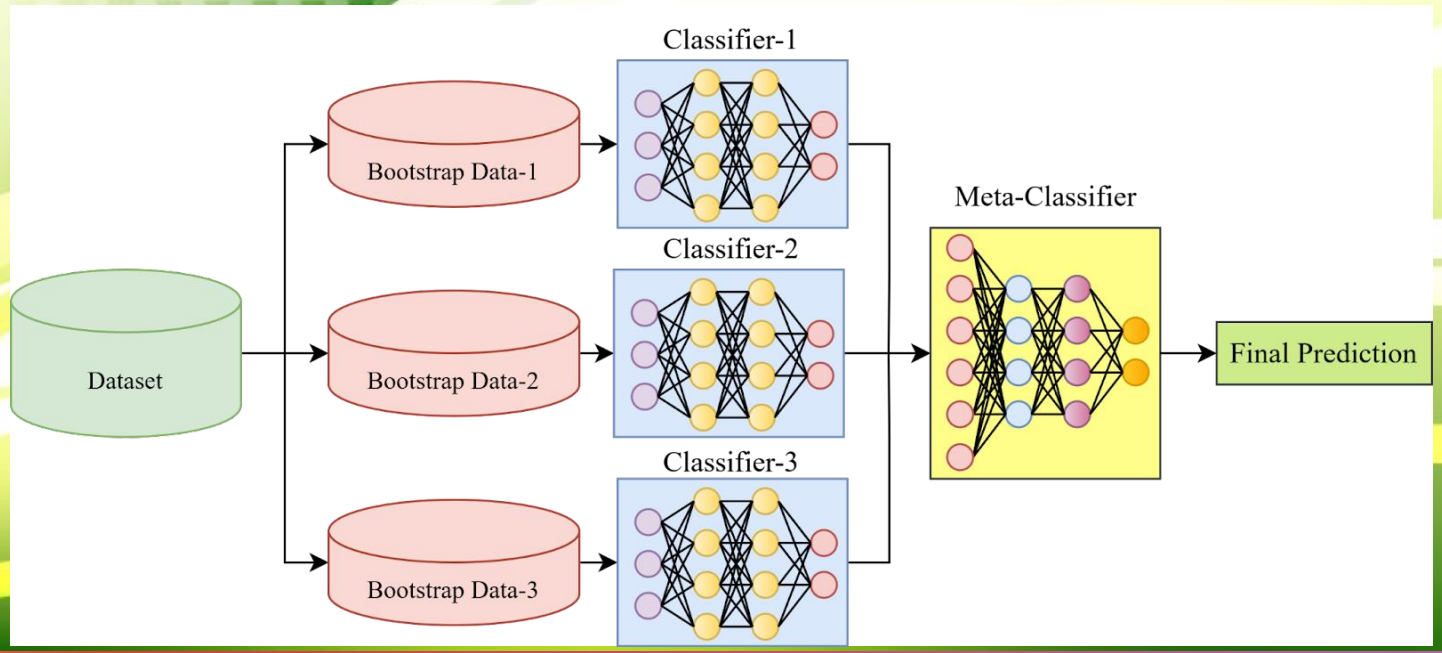
- easy to understand and implement.
- does not require any major preprocessing.
- can handle missing values in the data.
- efficiently reduces bias.
- prioritize features that increase overall accuracy during training.
    - reduces dimensionality of data, hence reducing computation time.

Disadvantages

- sequential approach of boosting makes next models correct mistakes of its predecessor.
    - makes overall model vulnerable to outliers.
    - not easily scalable.

ENSEMBLE

- Stacking is similar to boosting.
- Predictions from base learners are stacked together and are used as input to train meta learner to produce more robust predictions.
- Meta learner is then used to make final predictions.
- Bagging and boosting typically use homogeneous base learners, whereas stacking tends to include heterogeneous ones.



ENSEMBLE

Blending: similar to Stacking.

- Structure of data is made of training, hold-out (validation), and test data.
- Base learners are trained on training data. Then their predictions are combined with hold-out data to build final meta model, which uses test data to make final predictions.

Cascading: stacking approach but with only one model in each layer.

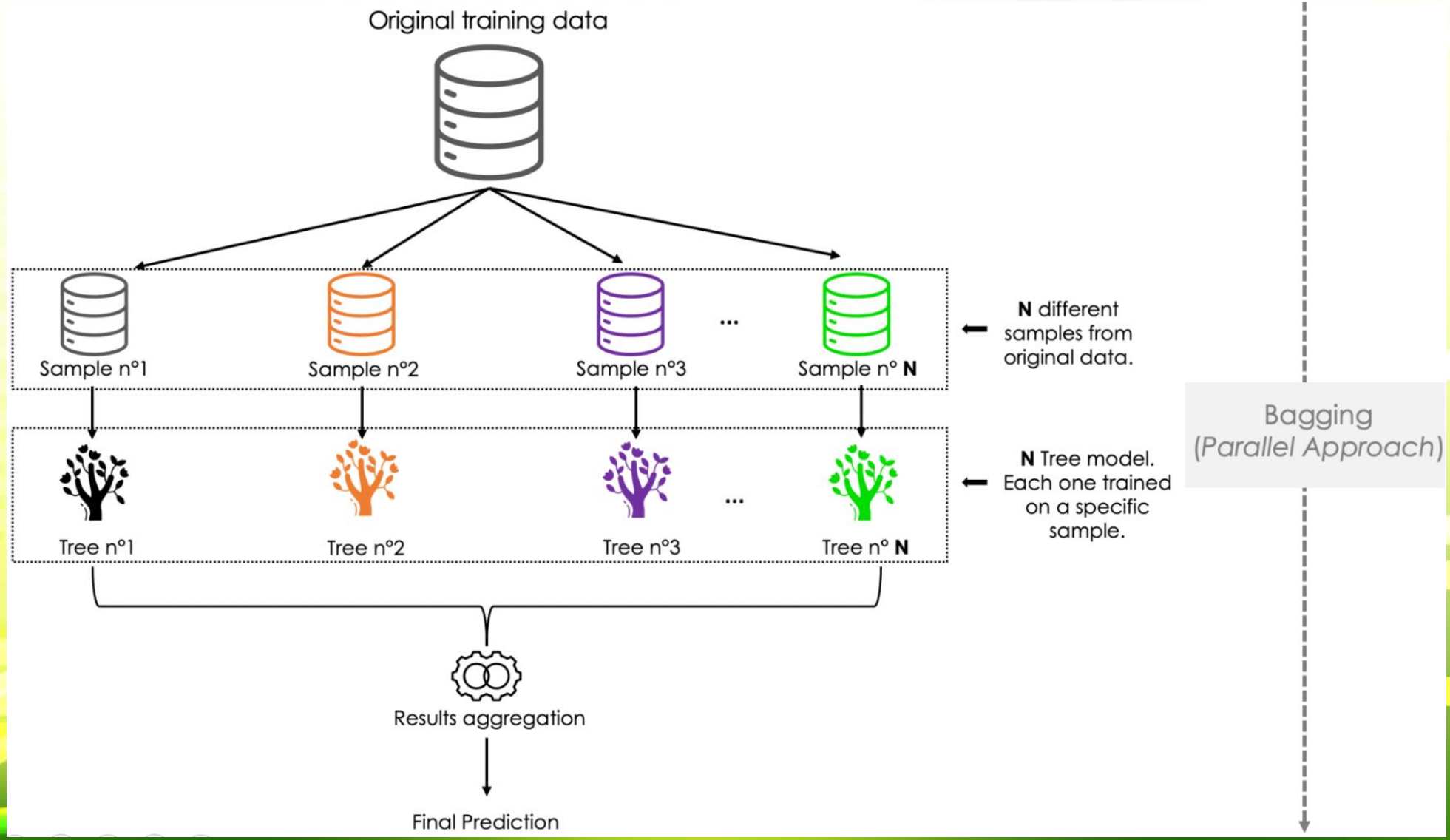
- first model is trained on whole training data,
- next model is trained on output of earlier model.
- helps to learn complex patterns from data, hence allow the model to make better predictions.

ENSEMBLE

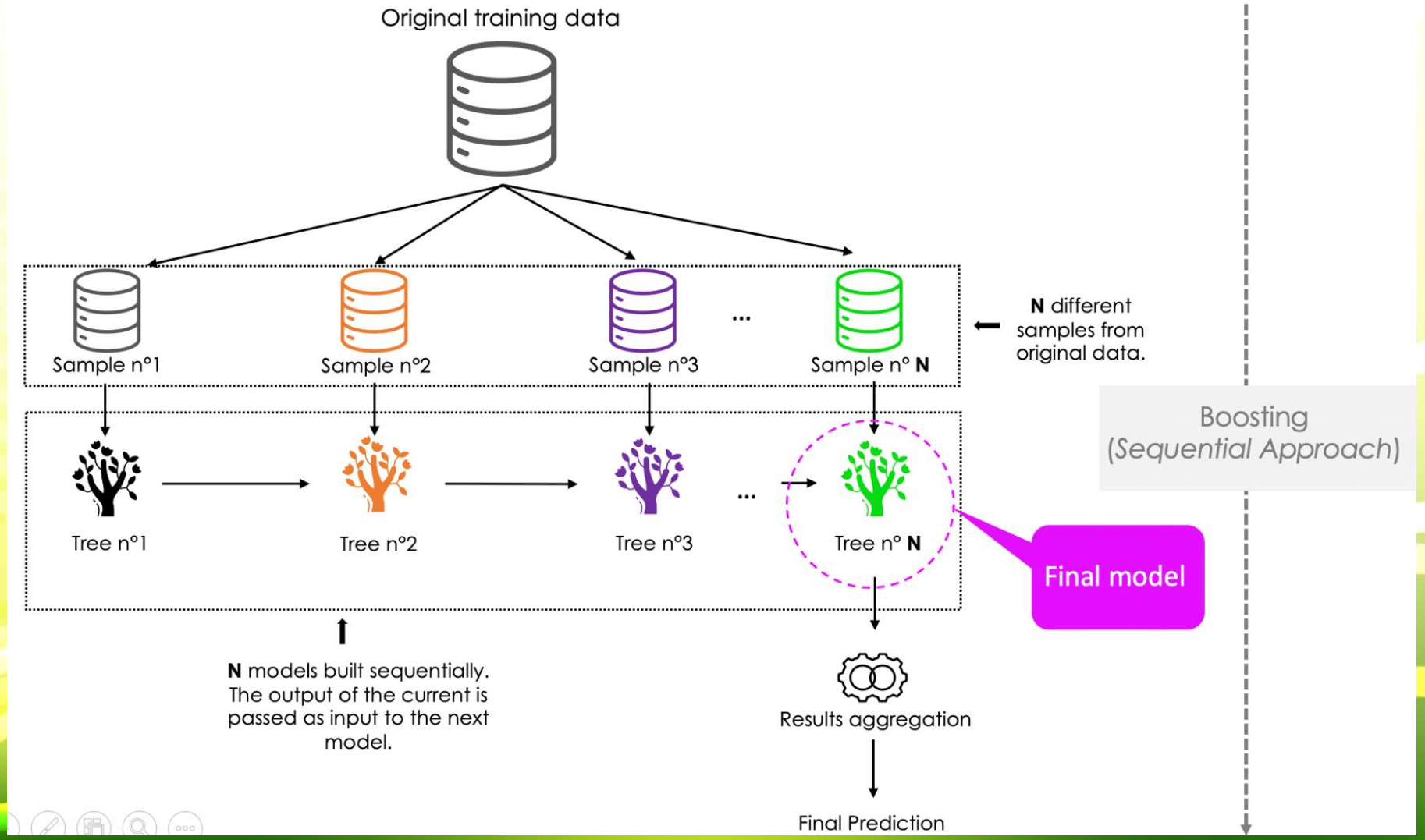
Voting: multiple models are trained independently, and their predictions are combined to make a final prediction.

- Hard voting: final prediction is most common prediction from all models.
- Soft voting: each model generates a probability distribution instead of a binary prediction.
    - class with highest probability is predicted.
- Weighted voting: some models are assigned with weights as they have more contribution when making predictions.

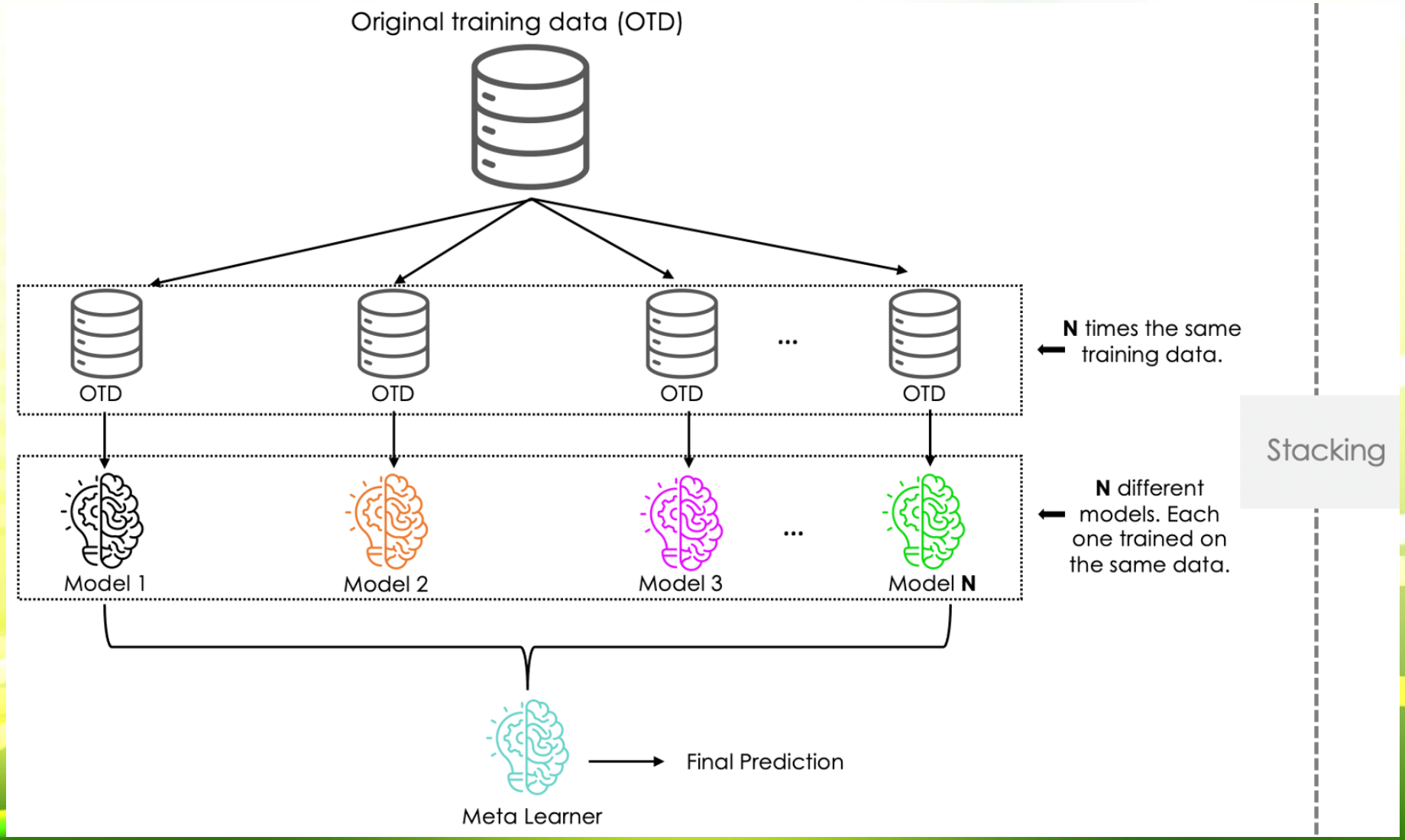
ENSEMBLE


Original training data
Sample n°1
Sample n°2
Sample n°3
...
Sample n° N
N different samples from original data.
Tree n°1
Tree n°2
Tree n°3
...
Tree n° N
N Tree model. Each one trained on a specific sample.
Bagging (Parallel Approach)
Results aggregation
Final Prediction

ENSEMBLE


Original training data
Sample n°1
Sample n°2
Sample n°3
...
Sample n° N
N different samples from original data.
Tree n°1
Tree n°2
Tree n°3
...
Tree n° N
Final model
N models built sequentially. The output of the current is passed as input to the next model.
Results aggregation
Final Prediction
Boosting
(Sequential Approach)

ENSEMBLE


Original training data (OTD)
OTD
OTD
OTD
...
OTD
N times the same training data.
Model 1
Model 2
Model 3
...
Model N
N different models. Each one trained on the same data.
Stacking
Meta Learner
Final Prediction

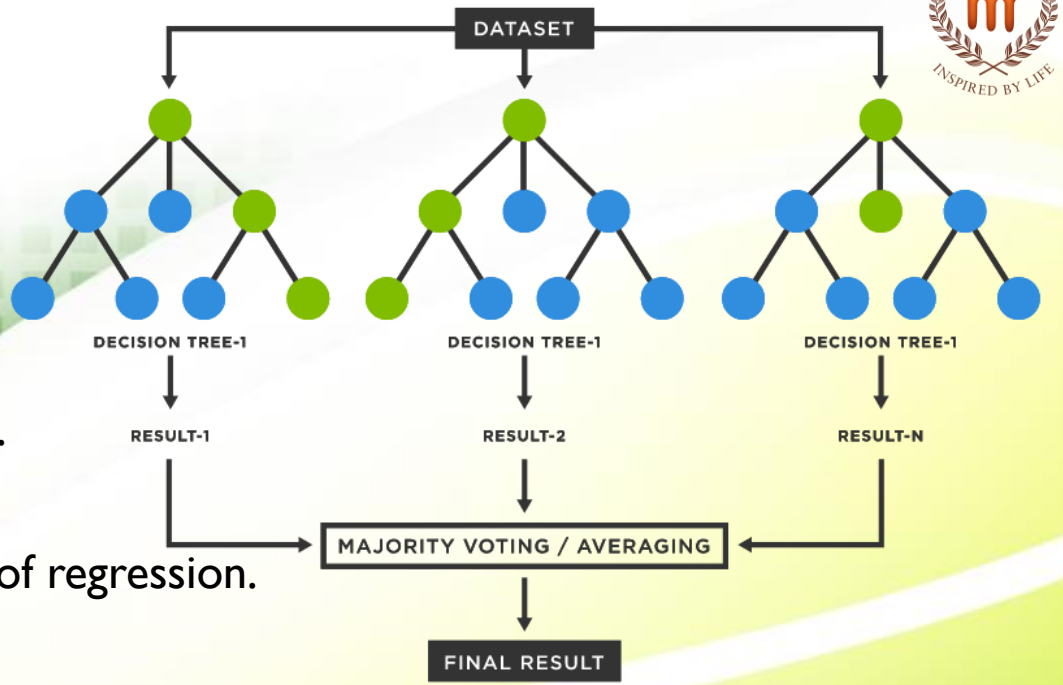
ENSEMBLE

- Random Forest is another ensemble machine learning algorithm that follows bagging technique.
- Extension of bagging estimator algorithm.
- The base estimators in random forest are decision trees.
- Random forest randomly selects a set of features which are used to decide best split at each node of decision tree.

Random forest model steps:

- Random subsets are created from the original dataset (bootstrapping).
- At each node in decision tree, only a random set of features are considered to decide best split.
- A decision tree model is fitted on each subsets.
- The final prediction is calculated by averaging the predictions from all decision trees.

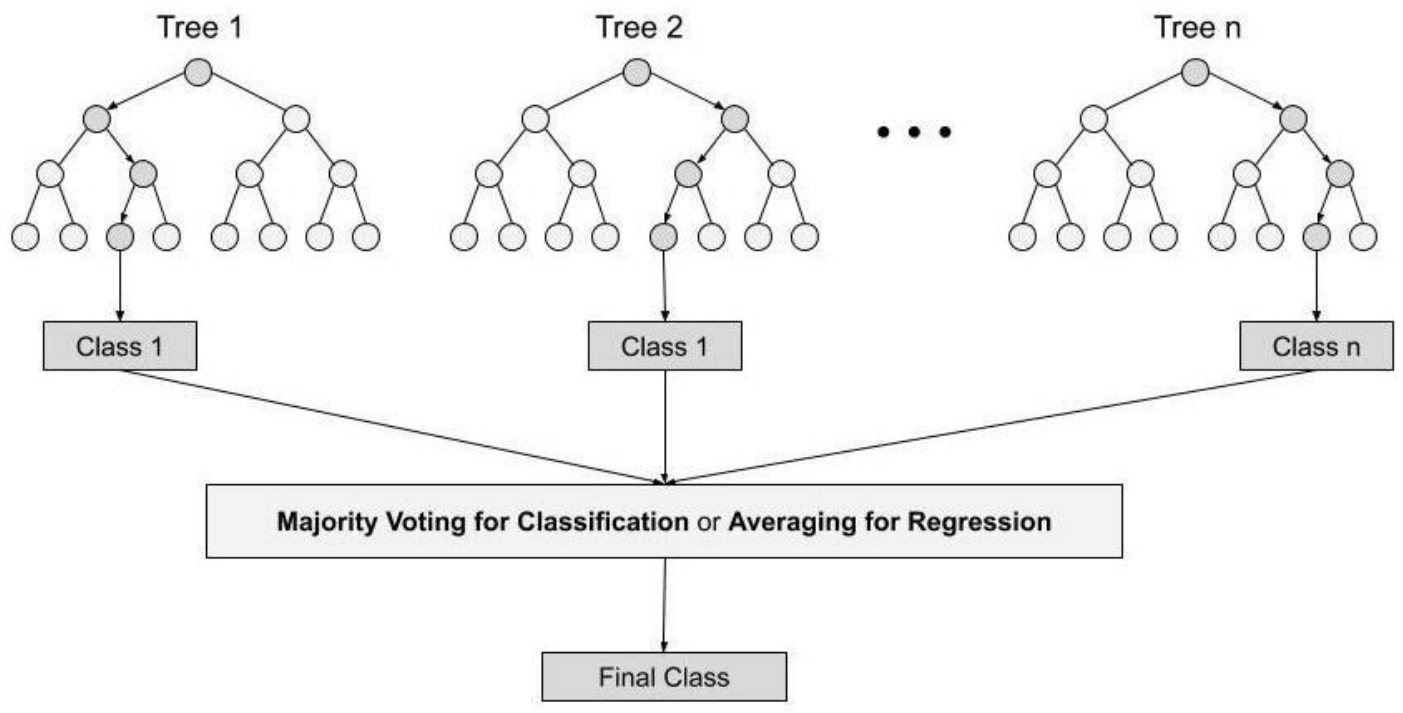
Random Forest



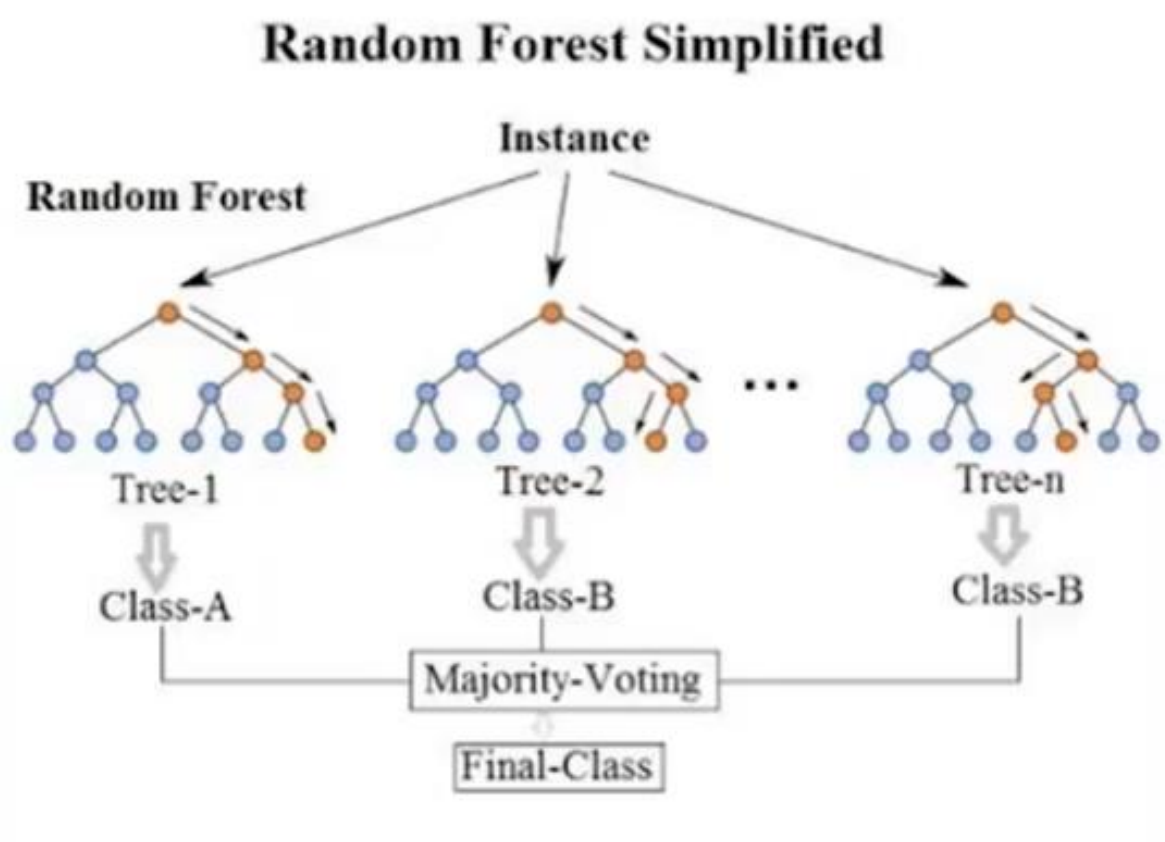
- *Supervised Learning Algorithm.*
- *used for Classification and Regression problems.*
- Random Forest has multiple decision trees as base learning models.
- builds decision trees on different samples
  - takes their majority vote for classification and average in case of regression.
- One important features of Random Forest Algorithm is that it can handle data set containing *continuous variables* as in case of regression and *categorical variables* as in case of classification.
  - performs better results for classification problems.
- basic idea behind this is to combine multiple decision trees in determining final output rather than relying on individual decision trees.

Random Forest

- Every decision tree has high variance, but when combine all of them together in parallel then resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence output doesn't depend on one decision tree but multiple decision trees.
  - In case of a classification problem, final output is taken by using the majority voting classifier.
  - In case of a regression problem, final output is the mean of all outputs. (Aggregation)
  - Randomly perform row sampling & feature sampling from the dataset forming sample datasets for every model (Bootstrap)

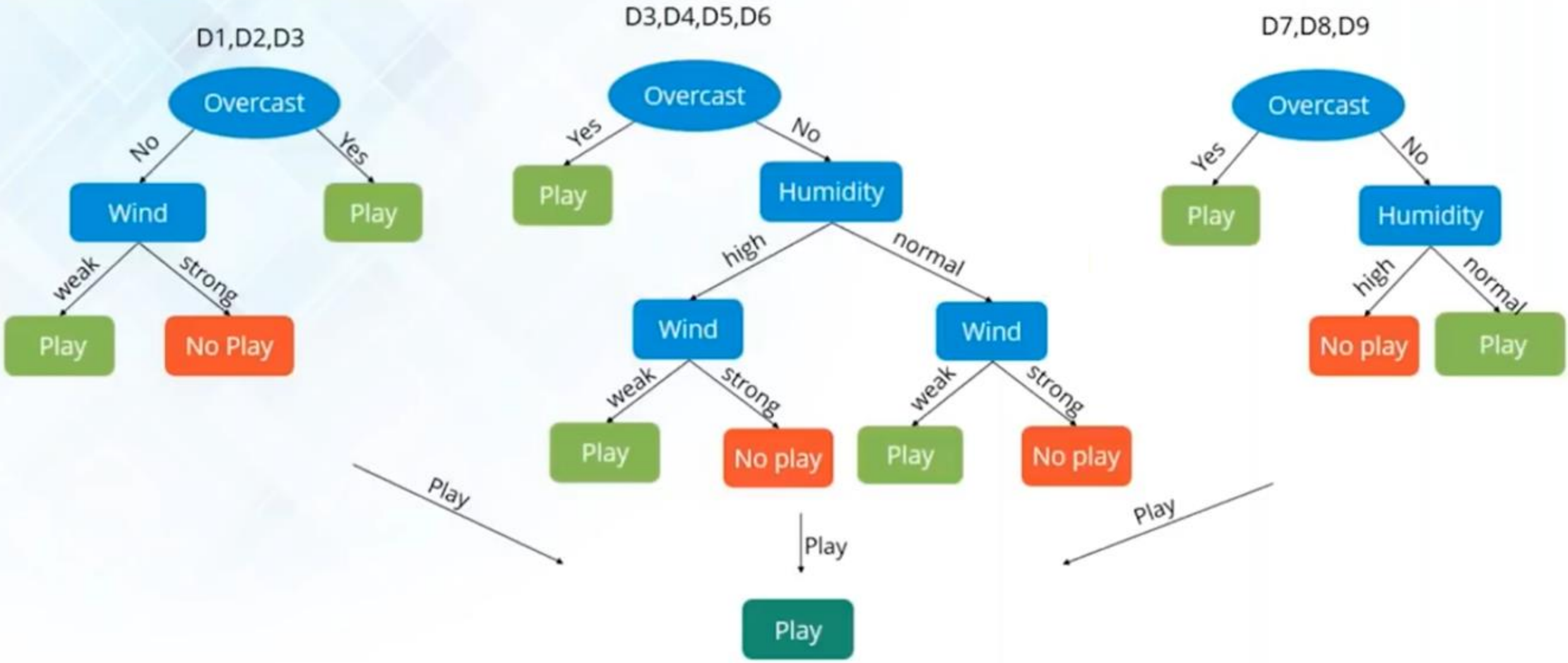


Random Forest



Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Random Forest



Random Forest

Approach for Base model in Random Forest technique is like any other machine learning technique;

- Design a specific question and get the source to determine the required data.
- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model.
- Set the baseline model that is to be achieved.
- Train the data on decision tree learning model.
- Provide an insight into the model with test data.
- Compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy expectations, try improving model accordingly or use another data modeling technique.

ENSEMBLE

- Adaptive boosting or AdaBoost is one of the simplest boosting algorithms.
- Usually, decision trees are used for modelling.
- Multiple sequential models are created, each correcting the errors from last model.
- AdaBoost assigns weights to observations which are incorrectly predicted and subsequent model works to predict these values correctly.

AdaBoost algorithm steps :

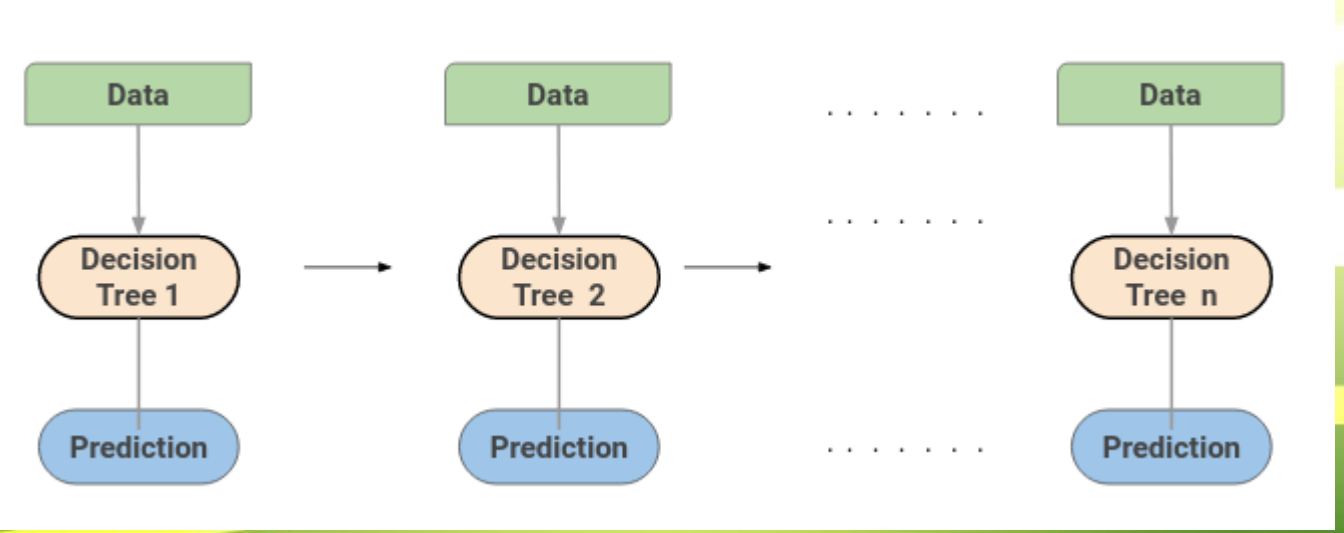
- Initially, all observations in the dataset are given equal weights.
- A model is built on a subset of data.
- Using this model, predictions are made on the whole dataset.
- Errors are calculated by comparing the predictions and actual values.
- While creating the next model, higher weights are given to the data points which were predicted incorrectly.
- Weights can be determined using the error value. Higher the error more weight assigned to the observation.
- This process is repeated until error function does not change, or max limit of number of estimators is reached.

ENSEMBLE

- Gradient boosting machine (GBM) is used to define loss function and reduce it.
- Solve problems of classification using prediction models.
- Use of loss function depends on type of problem.
- Usually, decision trees are used as a weak learner.
- A regression tree is used to give true values, which can be combined together to create correct predictions.
- Like in AdaBoost algorithm, small trees with a single split are used, i.e. decision stump. Larger trees are used for large levels i.e. 4-8 levels.
- Its an Additive Model. Trees are added one at a time. During the addition of trees, gradient descent is used to minimize the loss function

ENSEMBLE

- Nodes in every decision tree take a different subset of features for selecting the best split.
  - Individual trees aren't all the same and hence they are able to capture different signals from the data.
- Each new tree takes into account the errors made by previous trees.
  - Every successive decision tree is built on errors of previous trees.



ENSEMBLE

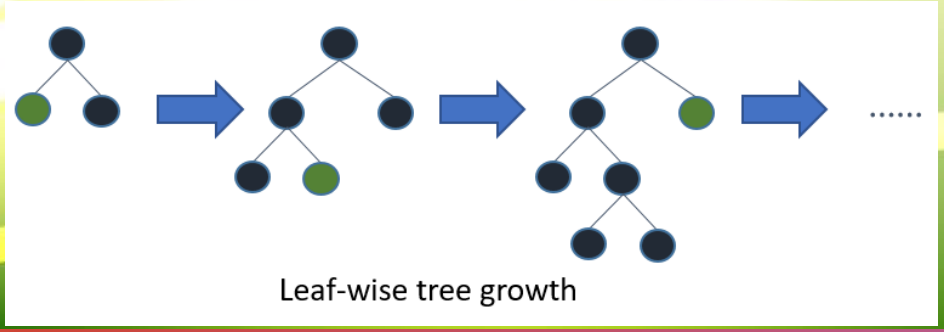
- Extreme Gradient Boosting Machine (XG Boost or XGBM) is an upgraded implementation of Gradient Boosting Algorithm, which is developed for high computational speed, scalability, and better performance.

XG Boost features:

- XG Boost provides Parallel Processing for tree construction which uses CPU cores while training.
- XG Boost enables users to run cross-validation of the boosting process at each iteration, making it easy to get the exact optimum number of boosting iterations in one run.
- It provides Cache Optimization of the algorithms for higher execution speed.
- For training large models, XG Boost allows Distributed Computing.
- XGBM model can handle missing values on its own.

ENSEMBLE

- LightGBM boosting algorithm shows better speed and efficiency.
- Handles huge amounts of data with ease. But, does not perform well with a small number of data points.
- Trees in LightGBM have a leaf-wise growth, rather than a level-wise growth.
- After first split, the next split is done only on leaf node that has a higher delta loss.
- After first split, the left node had a higher loss and is selected for next split.
- Leaf-wise split of LightGBM algorithm enables it to work with large datasets.
- To speed up training process, LightGBM uses a histogram-based method for selecting the best split.
- For any continuous variable, instead of using individual values, these are divided into bins or buckets. This makes training process faster and lowers memory usage.



Leaf-wise tree growth

ENSEMBLE

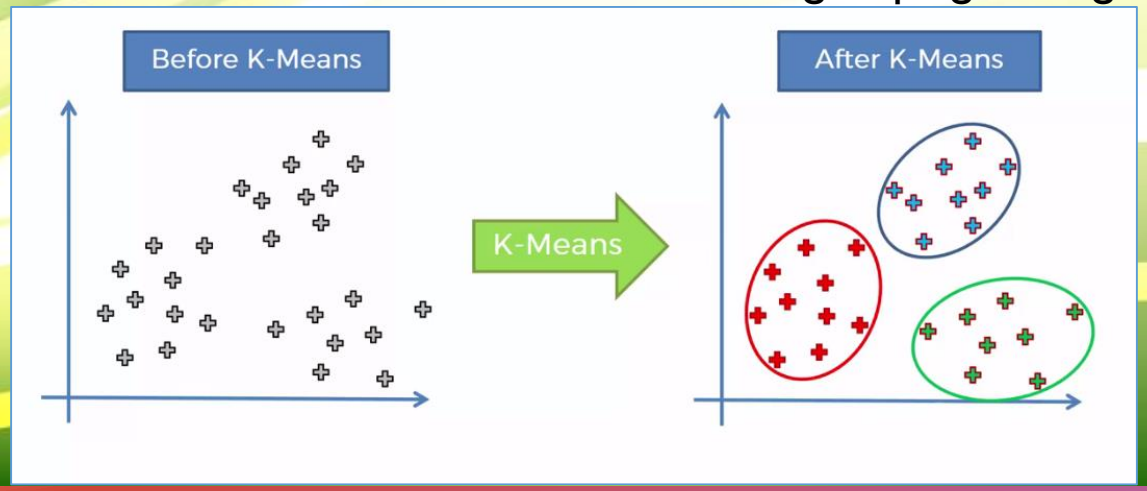
- CatBoost is a boosting algorithm that can handle categorical variables in data.
- Most machine learning algorithms cannot work with strings or categories in data. Thus, converting categorical variables into numerical values is an essential preprocessing step.
- CatBoost can internally handle categorical variables in data → These variables are transformed to numerical ones using various statistics on combinations of features.
- CatBoost works well with default set of hyperparameters → user need not have to spend lot of time tuning hyperparameters.

GROUPING

- Grouping and classification techniques are very important methods in predictive data analysis.
- Grouping Analysis methods helps to determine natural groupings in data.
- Useful to decompose data set into simpler subsets → helps to make sense of entire collection of observations.
- For each group summary statistics, variety of graphs may help in better analysis
- Different ways to visualize and group observations,
  - *Clustering:* based on similarities of overall set of variables of interest.
  - *Association rule:* identify groups based on interesting combinations of predefined categories
  - *Decision tree / Random forest:* groups observation based on combination of ranges of continuous variables or of specific categories.

CLUSTERING

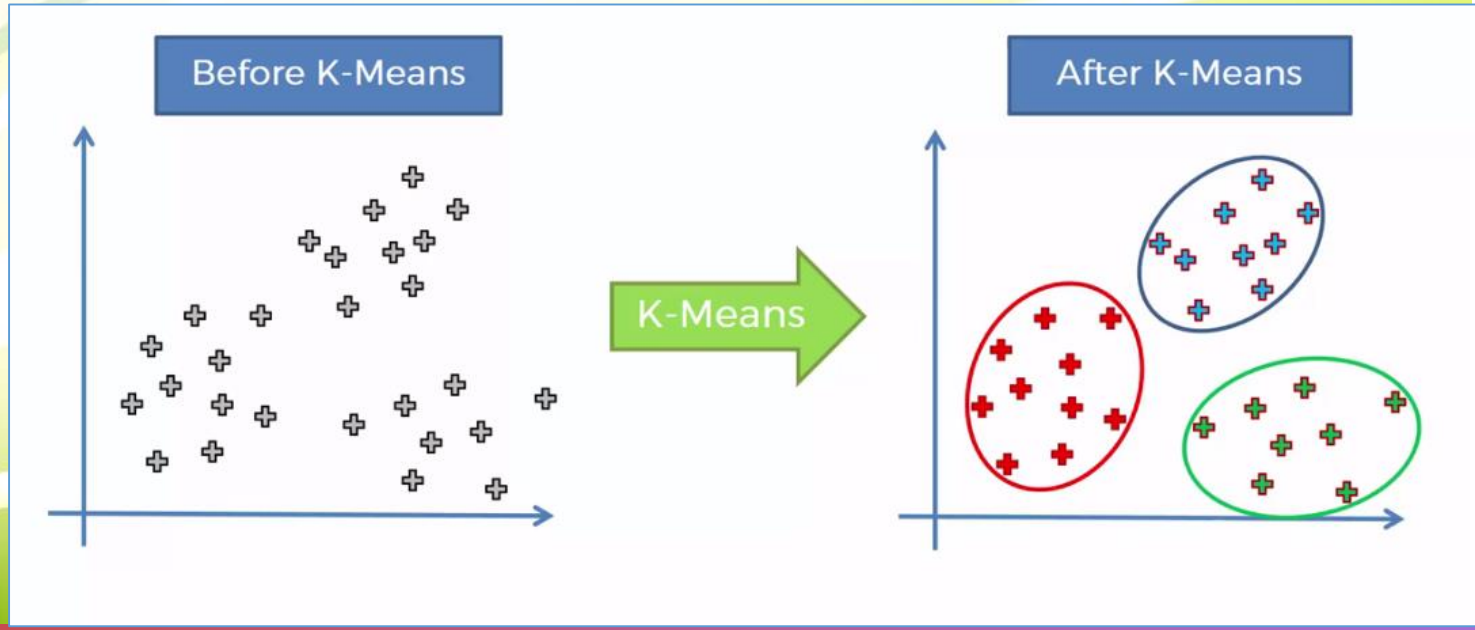
- Cluster: group of (similar) objects that belongs to same class.
- Clustering: process of making a group of abstract objects into classes of similar objects.
- Given a data set of items, with certain features, and values for these features; the task is to categorize those items into groups.
  - Used to find similarity as well as relationship patterns among data samples and then cluster those samples into groups having similarity based on features.
  - Clustering is important because it determines the intrinsic grouping among the present unlabeled data.



CLUSTERING

Clustering methods −

- Partitioning Method
- Hierarchical Method; Agglomerative Approach, Divisive Approach
- Constraint-based Method
- Density-based Method
- Grid-Based Method
- Distribution Model-Based Method
- Fuzzy Method



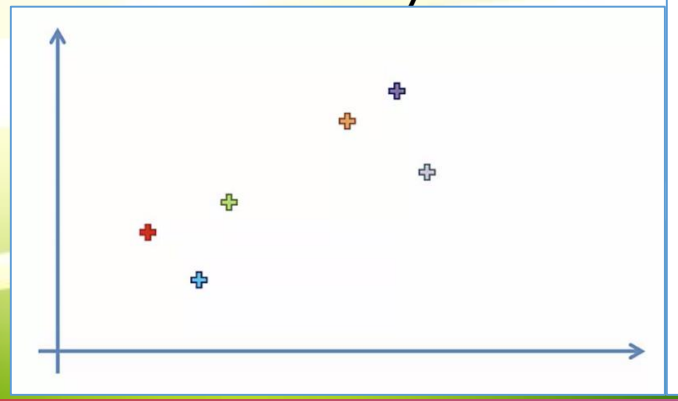
Clustering

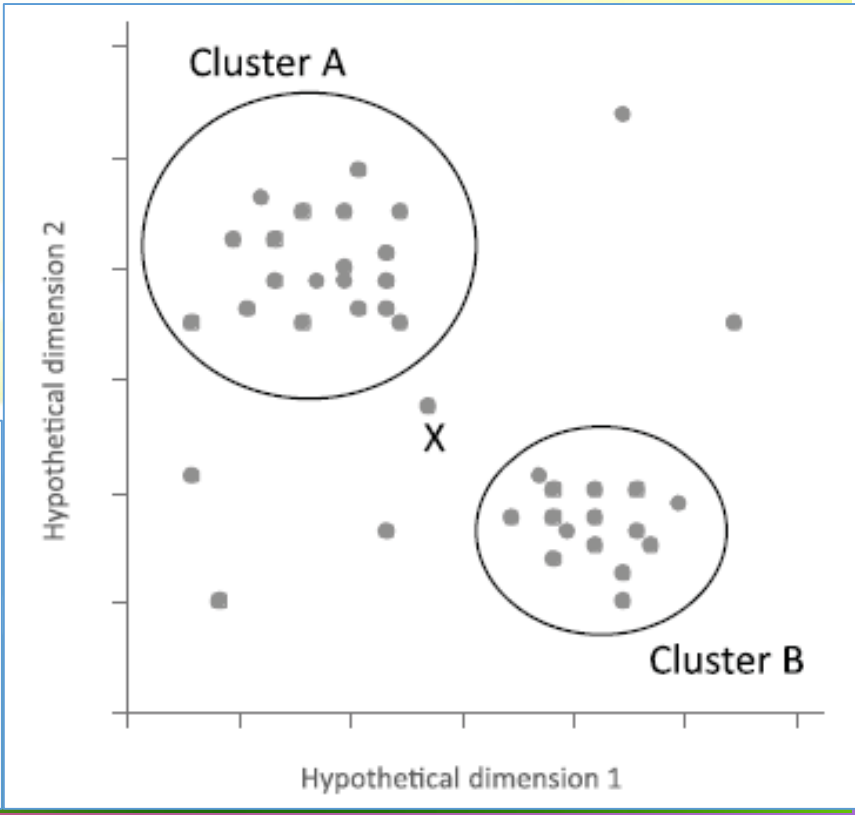
Applications of Cluster Analysis

- Market Segmentation: help marketers discover distinct groups in their customer base → characterize customer groups based on purchasing patterns.
- Anomaly detection, Outlier detection applications; example detection of credit card fraud.
- *Biological data analysis:* used to derive plant and animal taxonomies, categorize genes with similar functionalities and gain insight into structures inherent to populations.
- Social network analysis
- Image segmentation

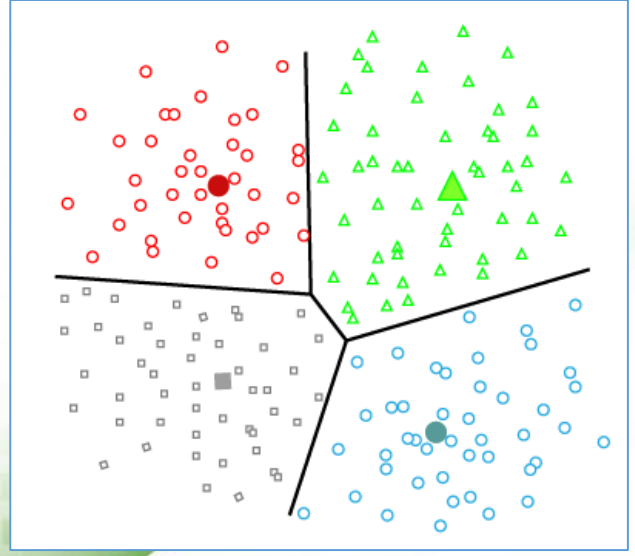
CLUSTERING

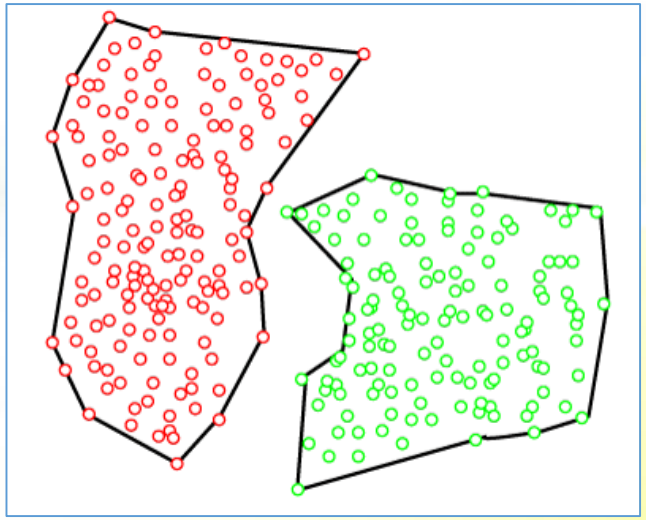
- Clustering is an *unsupervised* method for grouping.
- Unsupervised: groups are not known in advance.
- Clustering method chosen to subdivide data into groups applies automated procedure to discover groups based on some criteria.
- Many clustering methods.
- Each method will group data differently based on criteria it uses.
- For clustering, there is no way to measure accuracy (usefulness matters).
- Distance between two observations defines how similar they are to be in same cluster or not.





CLUSTERING





Partitioning Clustering

- Also known as centroid-based method.
- Divides the data into non-hierarchical groups.
- Example: K-Means Clustering algorithm.
- The dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups.
- The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.

Density-Based Clustering

- Connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected.
- Identify different clusters in the dataset and connects the areas of high densities into clusters.
- The dense areas in data space are divided from each other by sparser areas.
- Face difficulty if the dataset has varying densities and high dimensions.

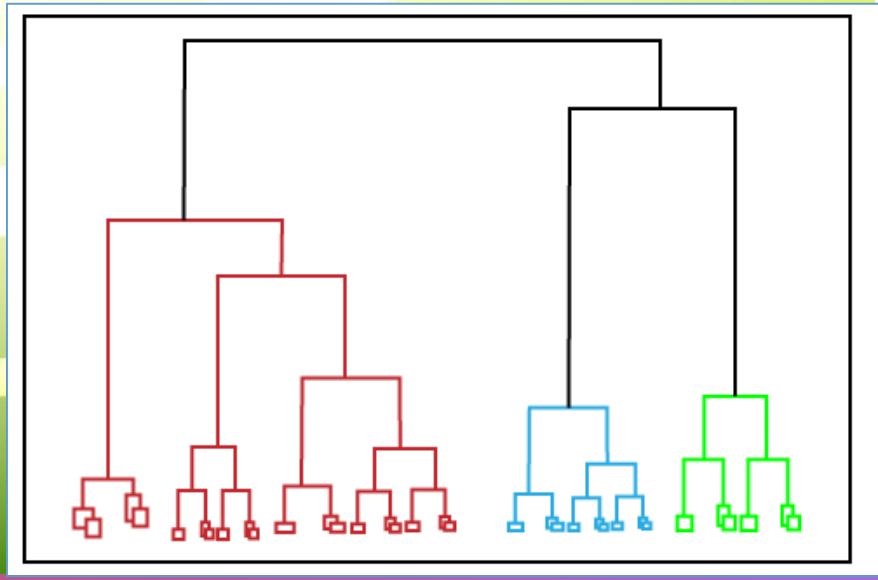
CLUSTERING

Hierarchical Clustering

- Used as an alternative for partitioned clustering, as there is no requirement of pre-specifying the number of clusters (k) to be created.
- Dataset is divided into clusters to create a tree-like structure (dendrogram).
- The observations or any number of clusters can be selected by cutting the tree at the correct level.
- Example: Agglomerative Hierarchical algorithm.

Fuzzy Clustering

- Data object may belong to more than one group or cluster.
- Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster.
- Example: Fuzzy k-means algorithm.



CLUSTERING

Clustering Algorithms

- **K-Means algorithm:** It classifies dataset by dividing samples into different clusters of equal variances. Number of clusters must be specified. It is fast with lower computation time required.

- **Agglomerative Hierarchical algorithm:** Performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.

- **Affinity Propagation:** It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has higher time complexity, which is the main drawback of this algorithm.

CLUSTERING

Clustering Algorithms

- **Mean-shift algorithm:** Tries to find dense areas in smooth density of data points. It is example of centroid-based model, that works on updating candidates for centroid to be center of points within given region.

- **DBSCAN Algorithm: Density-Based Spatial Clustering of Applications with Noise**. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.

- **Expectation-Maximization Clustering using GMM:** This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.

HIERARCHICAL CLUSTERING

- Hierarchical Clustering: creates hierarchical decomposition of given set of data objects.

Two approaches;

- Agglomerative Approach: (bottom-up approach) "*AGNES*" (*Agglomerative Nesting*)
  - Start with each object forming a separate/singleton group/cluster.
  - Keeps on merging objects or groups that are close/similar to one another. *(Euclidian distance)*
  - Keep on doing so until all of groups are merged into one or until termination condition holds.
  - normally limited to data sets with fewer (< 10,000 observations) → computational cost to generate hierarchical tree can be high for larger numbers of observations
  - result is a tree-based representation of the objects, named *dendrogram*.
- Divisive Approach: (top-down approach) "DIANA" (*Divise Analysis*)
  - Start with all of objects in same cluster.
  - In continuous iteration, a cluster is split up into smaller clusters.
  - Keep doing until each object in one cluster or termination condition holds.

Hierarchical clustering

Step-1: Create each data point as a single cluster (for N data points, number of clusters will also be N).

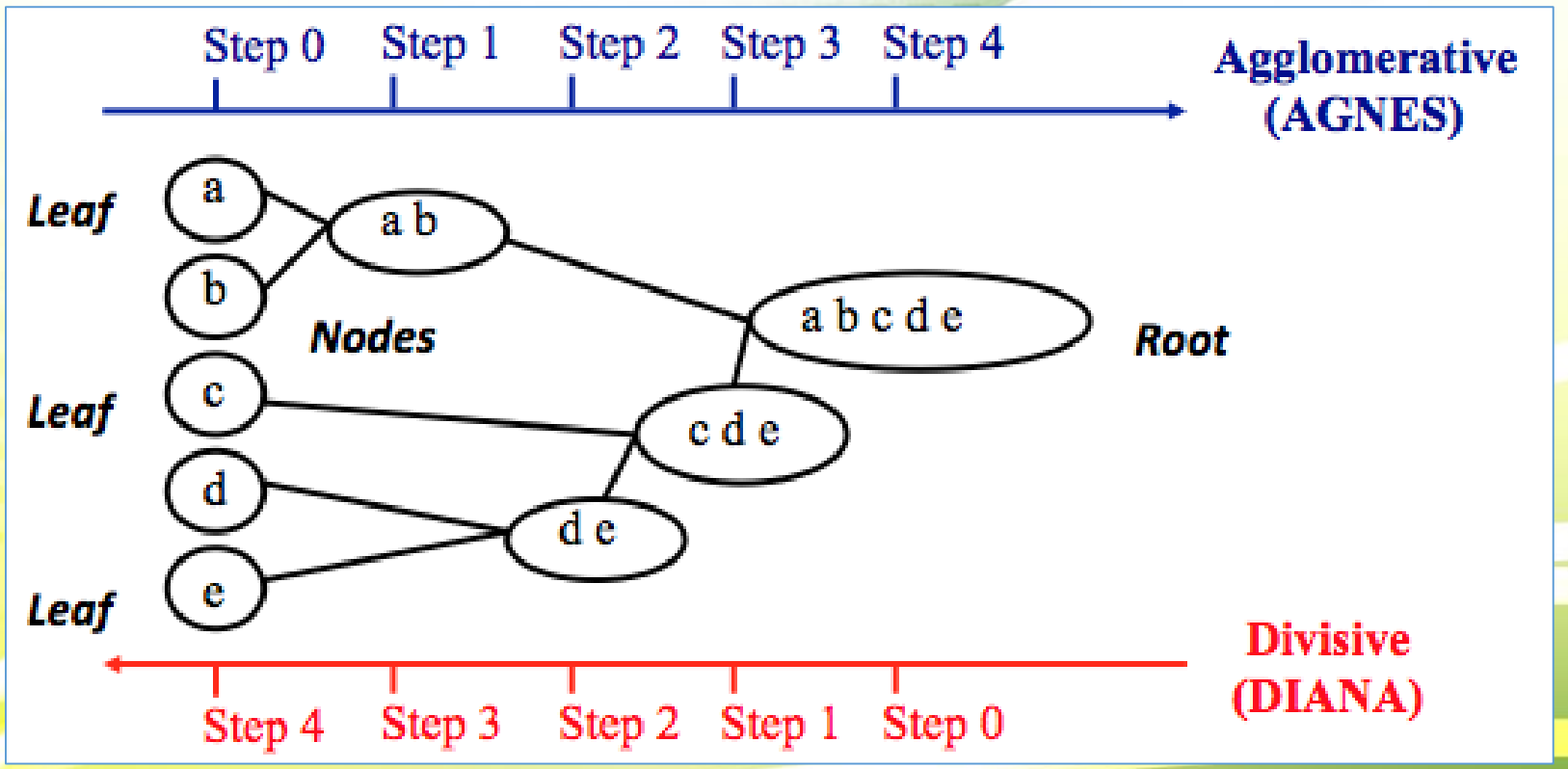
Step-2: Take two closest data points/clusters and merge them to form one cluster (N-1 clusters remains).

Step-3: Again, take two closest clusters and merge them together to form one cluster (remaining N-2 clusters).

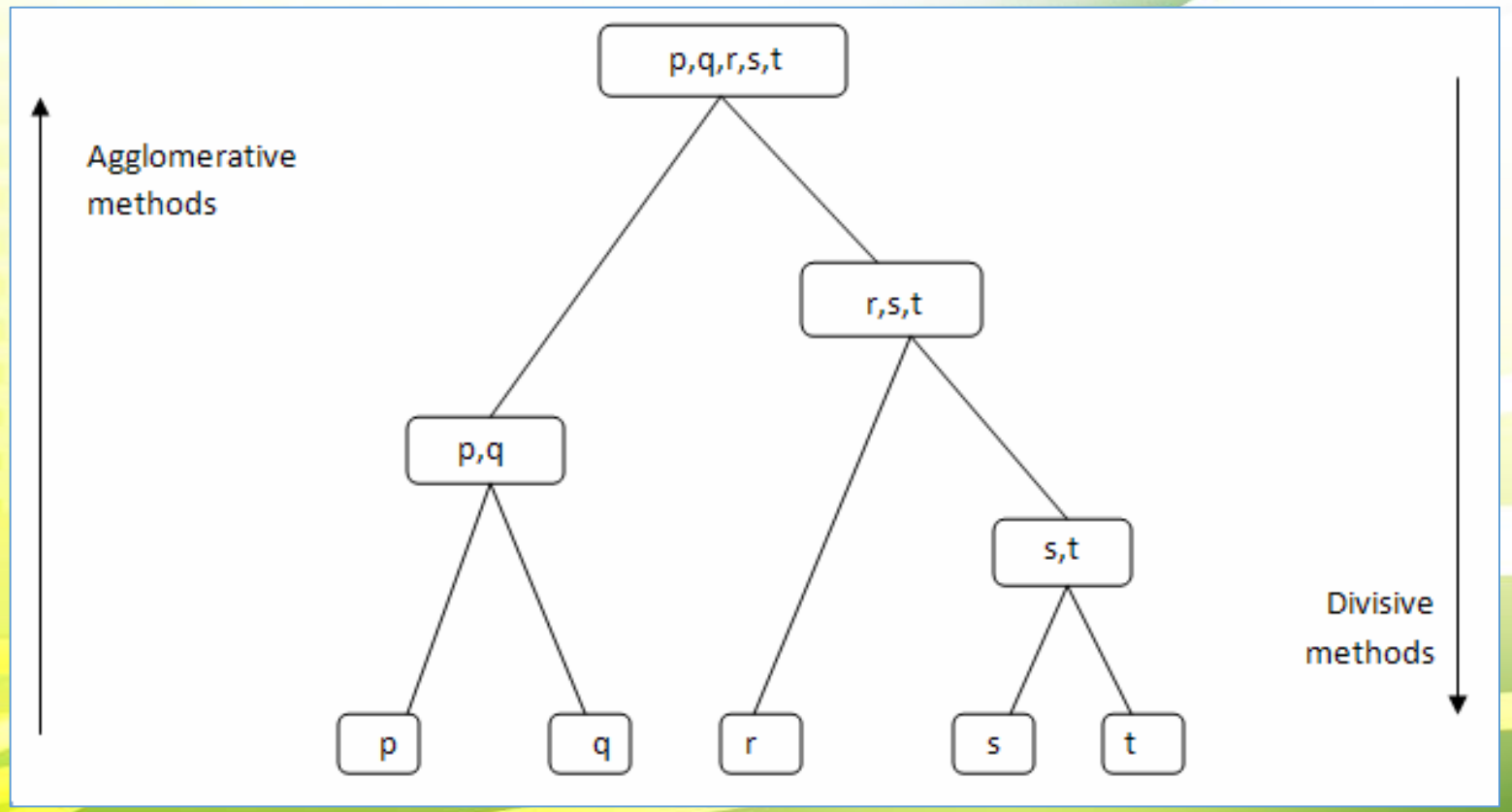
Step-4: Repeat Step 3 until only one cluster left *(or termination condition meets).*

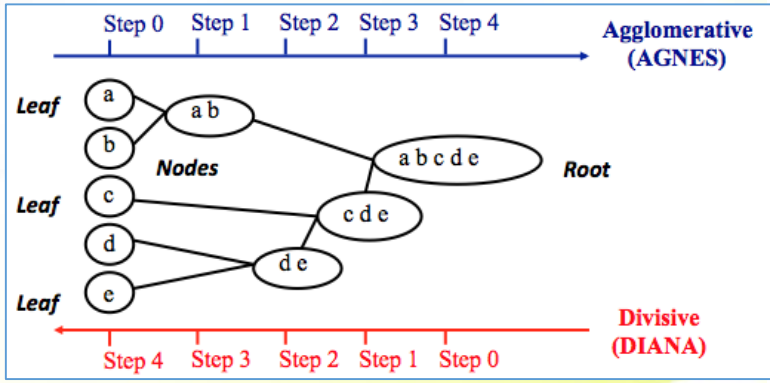
Step-5: Once all clusters are combined into one big cluster, develop the dendrogram to find required clusters.

HIERARCHICAL CLUSTERING

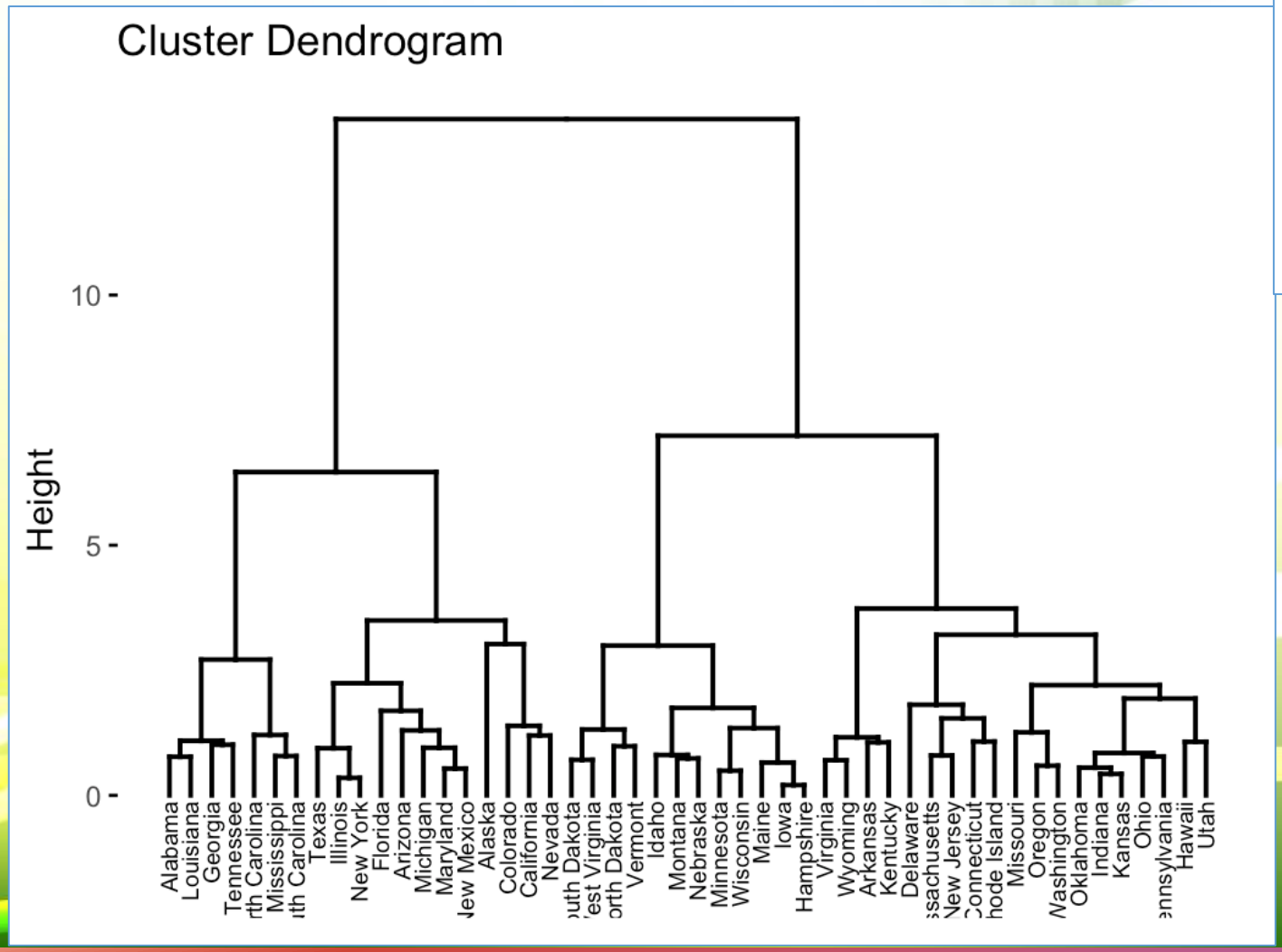


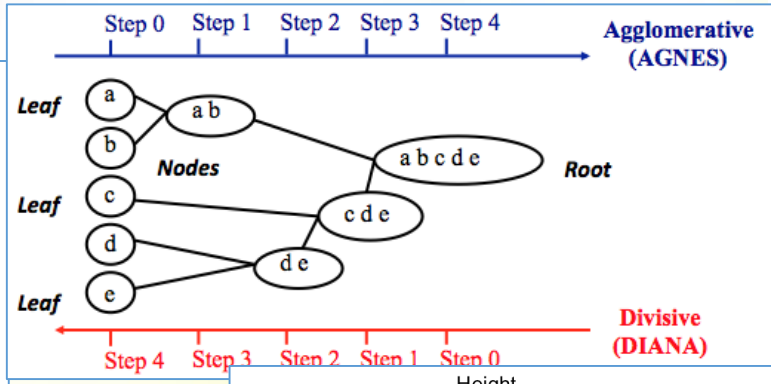
HIERARCHICAL CLUSTERING

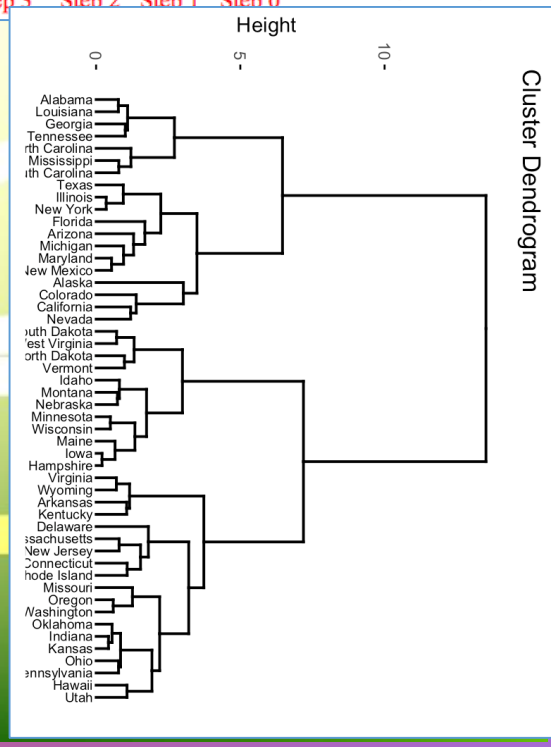




HIERARCHICAL CLUSTERING







HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

Data

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Proximity/distance matrix

Iteration - 2

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

Data

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Proximity/distance matrix

HIERARCHICAL CLUSTERING

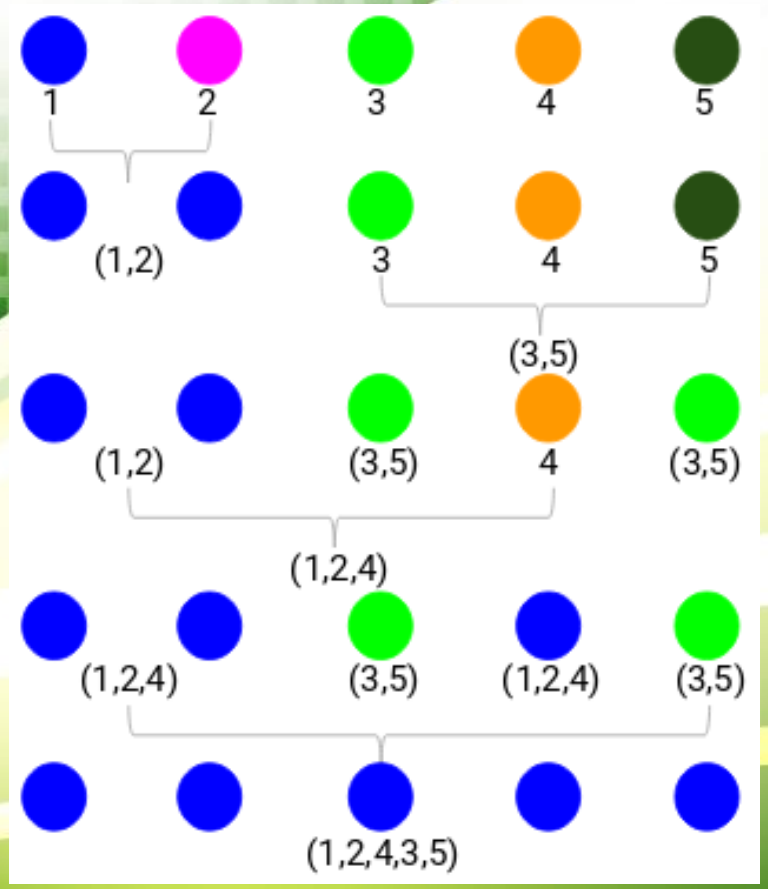
Agglomerative Hierarchical Clustering

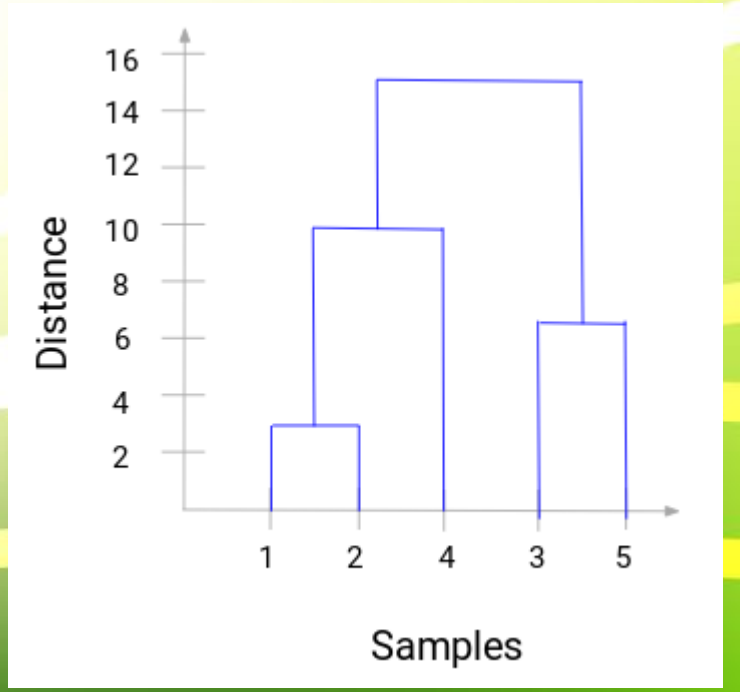
Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0





HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

- distances between all combinations of observations are summarized in **distance matrix**.
- diagonal values are excluded (pairs of same observation).
- distance matrix is usually symmetrical about diagonal *(distance between A & B is same as distance between B & A)*.
- The two closest observations are identified and are merged into a single cluster.
- Next iteration starts considering these two observations as single group (n-1 of observations).

Name	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
A	7.9	8.6	4.4	5.0	2.5
B	6.8	8.2	5.2	4.2	2.2
C	8.7	9.6	7.5	8.9	9.8
D	6.1	7.3	7.9	7.3	8.3
E	1.5	2.0	5.1	3.6	4.2
F	3.7	4.3	5.4	3.3	5.8
G	7.2	8.5	8.6	6.7	6.1
H	8.5	9.7	6.3	5.2	5.0
I	2.0	3.4	5.8	6.1	5.6
J	1.3	2.6	4.2	4.5	2.1
K	3.4	2.9	6.5	5.9	7.4
L	2.3	5.3	6.2	8.3	9.9
M	3.8	5.5	4.6	6.7	3.3
N	3.2	5.9	5.2	6.2	3.7

	A	B	C	D	...
A		$d_{A,B}$	$d_{A,C}$	$d_{A,D}$	...
B	$d_{B,A}$		$d_{B,C}$	$d_{B,D}$	...
C	$d_{C,A}$	$d_{C,B}$		$d_{C,D}$	...
D	$d_{D,A}$	$d_{D,B}$	$d_{D,C}$		...
...	...	...	...	...	

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

Name	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
A	7.9	8.6	4.4	5.0	2.5
B	6.8	8.2	5.2	4.2	2.2
C	8.7	9.6	7.5	8.9	9.8
D	6.1	7.3	7.9	7.3	8.3
E	1.5	2.0	5.1	3.6	4.2
F	3.7	4.3	5.4	3.3	5.8
G	7.2	8.5	8.6	6.7	6.1
H	8.5	9.7	6.3	5.2	5.0
I	2.0	3.4	5.8	6.1	5.6
J	1.3	2.6	4.2	4.5	2.1
K	3.4	2.9	6.5	5.9	7.4
L	2.3	5.3	6.2	8.3	9.9
M	3.8	5.5	4.6	6.7	3.3
N	3.2	5.9	5.2	6.2	3.7

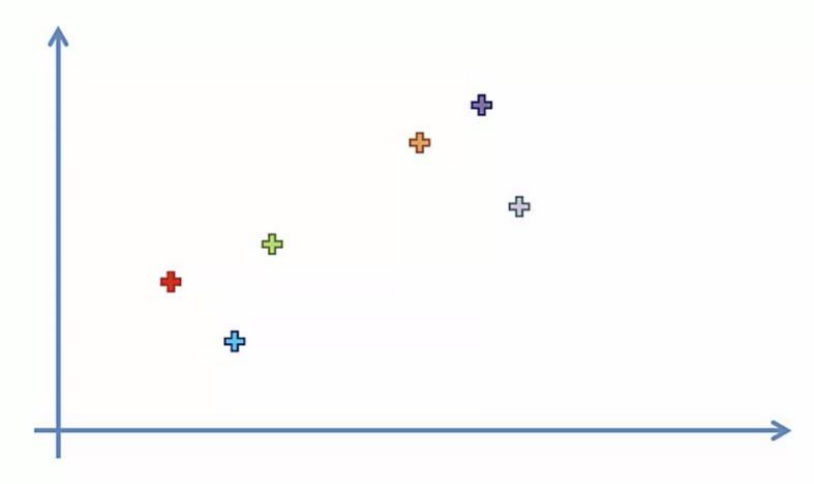
ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

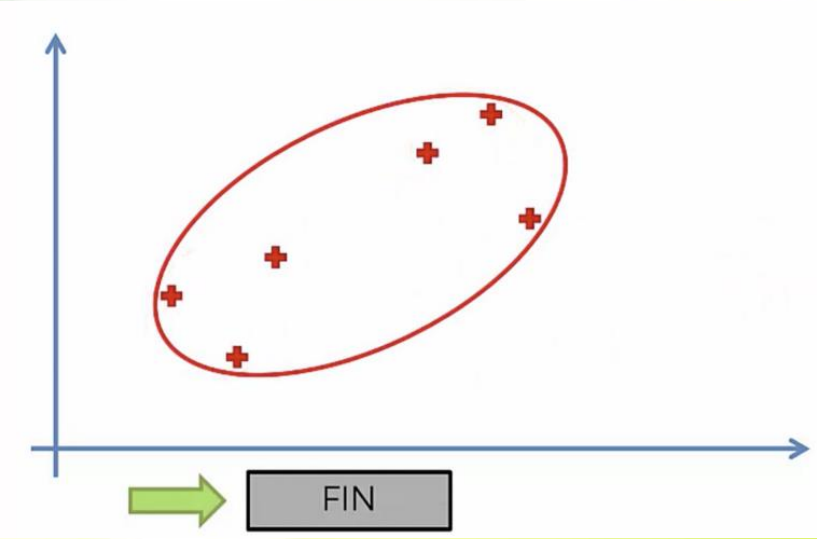
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A		0.282	1.373	1.2	1.272	0.978	1.106	0.563	1.178	1.189	1.251	1.473	0.757	0.793
B	0.282		1.423	1.147	1.113	0.82	1.025	0.56	1.064	1.065	1.144	1.44	0.724	0.7
C	1.373	1.423		0.582	1.905	1.555	0.709	0.943	1.468	1.995	1.305	1.076	1.416	1.378
D	1.2	1.147	0.582		1.406	1.092	0.403	0.808	0.978	1.543	0.797	0.744	1.065	0.974
E	1.272	1.113	1.905	1.406		0.476	1.518	1.435	0.542	0.383	0.719	1.223	0.797	0.727
F	0.978	0.82	1.555	1.092	0.476		1.191	1.039	0.57	0.706	0.595	1.076	0.727	0.624
G	1.106	1.025	0.709	0.403	1.518	1.191		0.648	1.163	1.624	1.033	1.108	1.148	1.051
H	0.563	0.56	0.943	0.808	1.435	1.039	0.648		1.218	1.475	1.169	1.315	0.984	0.937
I	1.178	1.064	1.468	0.978	0.542	0.57	1.163	1.218		0.659	0.346	0.727	0.553	0.458
J	1.189	1.065	1.995	1.543	0.383	0.706	1.624	1.475	0.659		0.937	1.344	0.665	0.659
K	1.251	1.144	1.305	0.797	0.719	0.595	1.033	1.169	0.346	0.937		0.64	0.774	0.683
L	1.473	1.44	1.076	0.744	1.223	1.076	1.108	1.315	0.727	1.344	0.64		0.985	0.919
M	0.757	0.724	1.416	1.065	0.797	0.727	1.148	0.984	0.553	0.665	0.774	0.985		0.196
N	0.793	0.7	1.378	0.974	0.727	0.624	1.051	0.937	0.458	0.659	0.683	0.919	0.196	

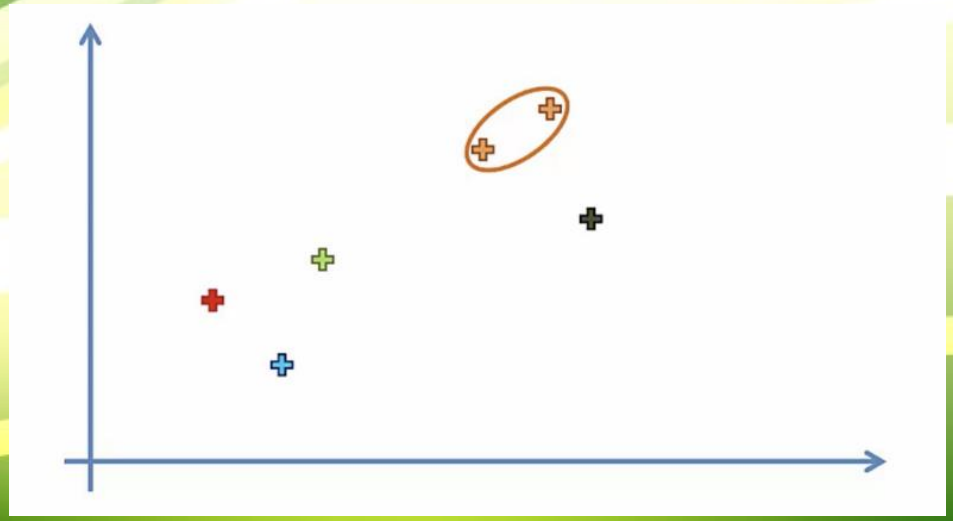
HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering



- Euclidean distance for distance between 2 observations.
- For distance between individual observations and clusters, a joining or linkage rule is used

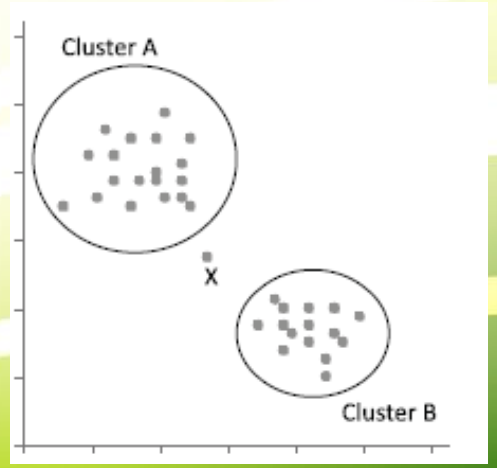




HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering Linkage Rule between single observation & a cluster:

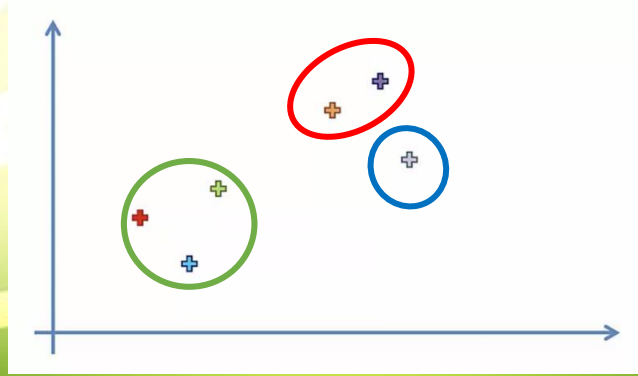
- Average linkage: distance between all members of cluster and the observation under consideration are calculated → average is used for overall distance.
- Single linkage: distance between all members of cluster and the observation under consideration are calculated → smallest is selected.
- Complete linkage: distance between all members of cluster and the observation under consideration are calculated → highest is selected.



HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering Linkage Rule between two clusters:

- Average linkage: distance between average/centroid values of both clusters.
- Single linkage: distance between all members of both clusters are calculated → smallest is selected.
- Complete linkage: distance between all members of both clusters are calculated → highest is selected.



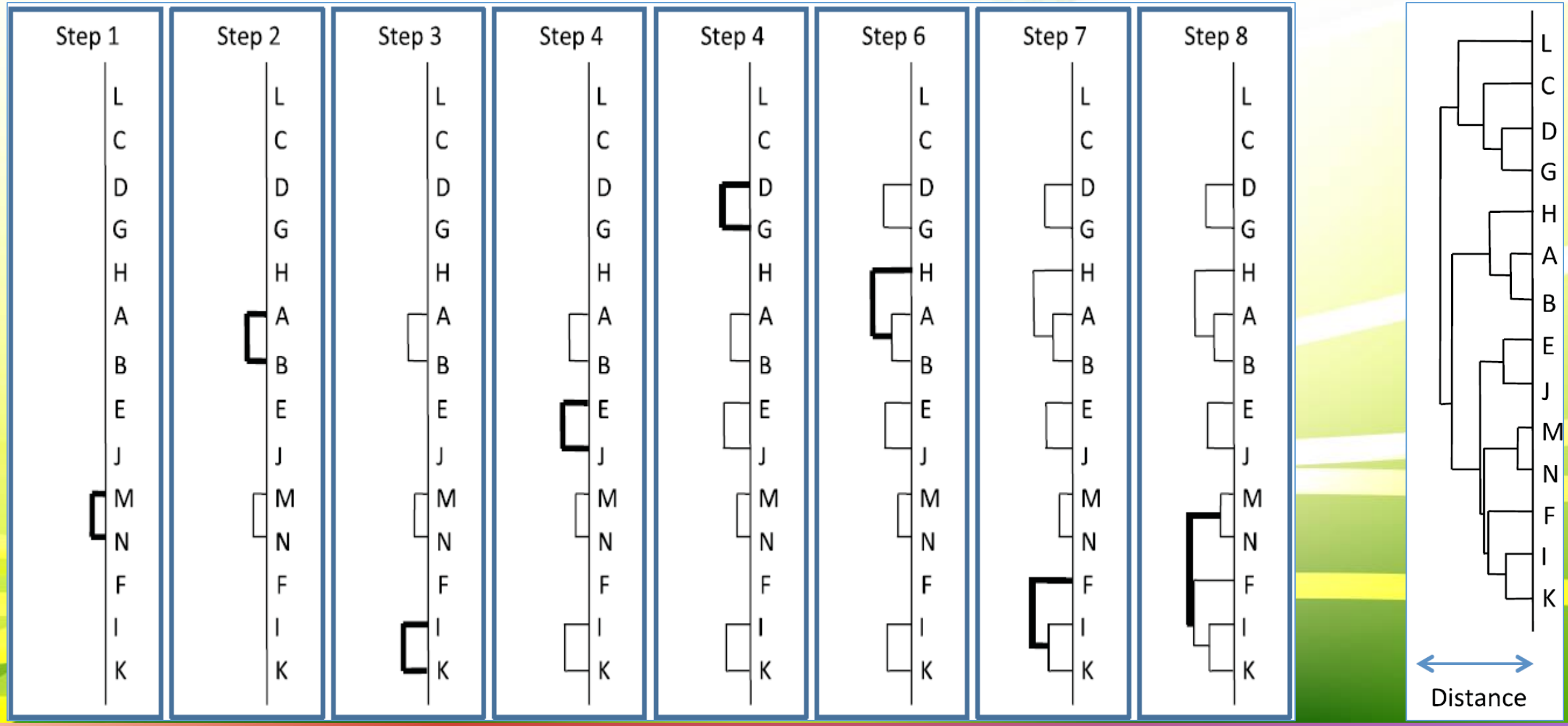
HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

Name	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
A	7.9	8.6	4.4	5.0	2.5
B	6.8	8.2	5.2	4.2	2.2
C	8.7	9.6	7.5	8.9	9.8
D	6.1	7.3	7.9	7.3	8.3
E	1.5	2.0	5.1	3.6	4.2
F	3.7	4.3	5.4	3.3	5.8
G	7.2	8.5	8.6	6.7	6.1
H	8.5	9.7	6.3	5.2	5.0
I	2.0	3.4	5.8	6.1	5.6
J	1.3	2.6	4.2	4.5	2.1
K	3.4	2.9	6.5	5.9	7.4
L	2.3	5.3	6.2	8.3	9.9
M	3.8	5.5	4.6	6.7	3.3
N	3.2	5.9	5.2	6.2	3.7

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A		0.282	1.373	1.2	1.272	0.978	1.106	0.563	1.178	1.189	1.251	1.473	0.757	0.793
B	0.282		1.423	1.147	1.113	0.82	1.025	0.56	1.064	1.065	1.144	1.44	0.724	0.7
C	1.373	1.423		0.582	1.905	1.555	0.709	0.943	1.468	1.995	1.305	1.076	1.416	1.378
D	1.2	1.147	0.582		1.406	1.092	0.403	0.808	0.978	1.543	0.797	0.744	1.065	0.974
E	1.272	1.113	1.905	1.406		0.476	1.518	1.435	0.542	0.383	0.719	1.223	0.797	0.727
F	0.978	0.82	1.555	1.092	0.476		1.191	1.039	0.57	0.706	0.595	1.076	0.727	0.624
G	1.106	1.025	0.709	0.403	1.518	1.191		0.648	1.163	1.624	1.033	1.108	1.148	1.051
H	0.563	0.56	0.943	0.808	1.435	1.039	0.648		1.218	1.475	1.169	1.315	0.984	0.937
I	1.178	1.064	1.468	0.978	0.542	0.57	1.163	1.218		0.659	0.346	0.727	0.553	0.458
J	1.189	1.065	1.995	1.543	0.383	0.706	1.624	1.475	0.659		0.937	1.344	0.665	0.659
K	1.251	1.144	1.305	0.797	0.719	0.595	1.033	1.169	0.346	0.937		0.64	0.774	0.683
L	1.473	1.44	1.076	0.744	1.223	1.076	1.108	1.315	0.727	1.344	0.64		0.985	0.919
M	0.757	0.724	1.416	1.065	0.797	0.727	1.148	0.984	0.553	0.665	0.774	0.985		0.196
N	0.793	0.7	1.378	0.974	0.727	0.624	1.051	0.937	0.458	0.659	0.683	0.919	0.196	

HIERARCHICAL CLUSTERING

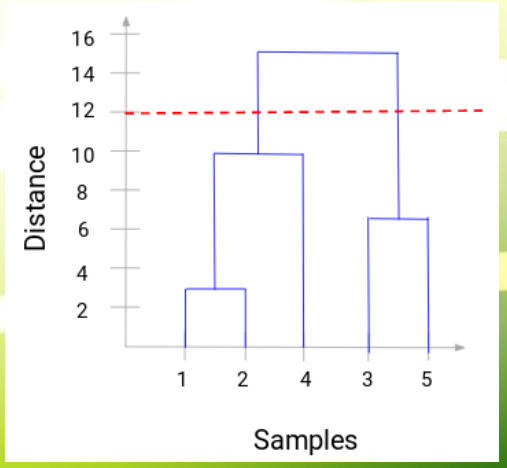


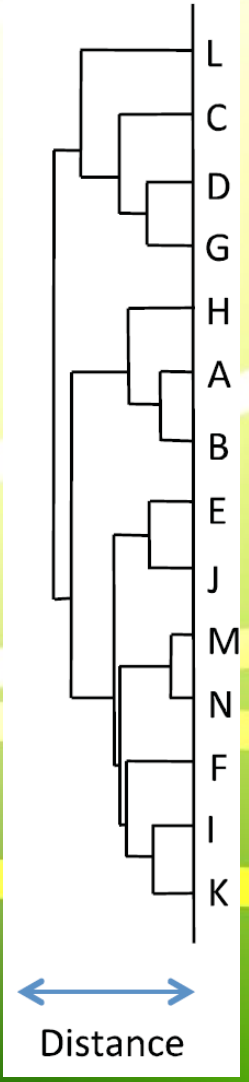
HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

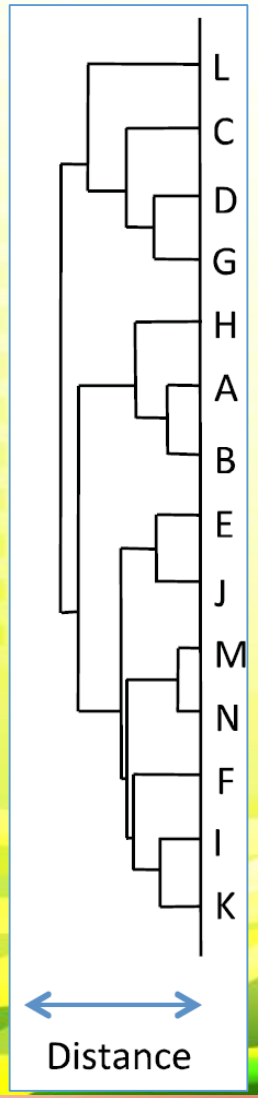
- When clustering completes, a tree called a dendrogram is generated showing similarity between observations and clusters.
- Horizontal length of lines reflects distance at which the cluster was formed.
- To divide dataset into series of distinct clusters, threshold distance (cut-off) is selected.
  - Where this distance intersects with a horizontal line on tree, a separate cluster is formed.

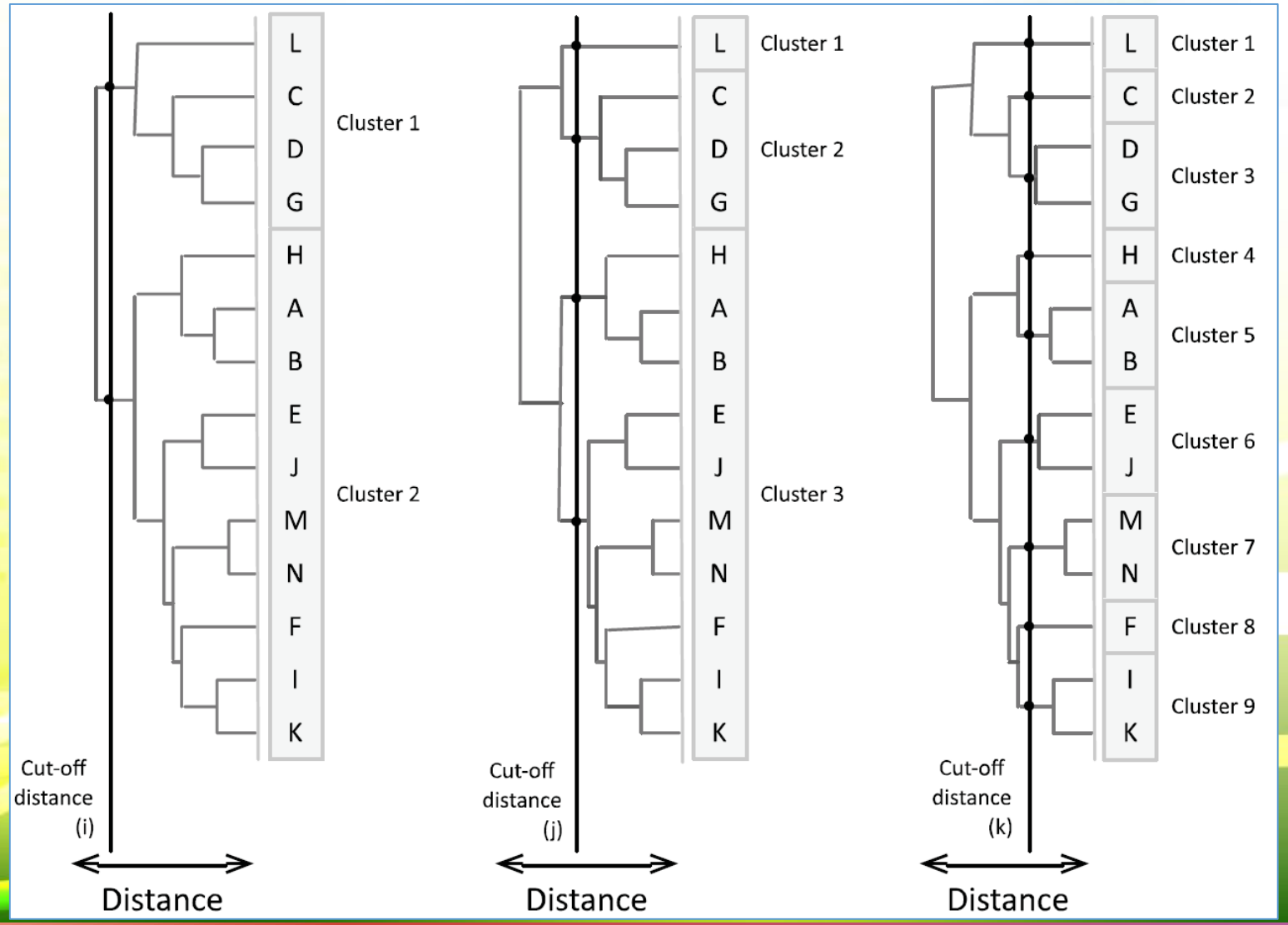
Student_ID	Marks
1	10
2	7
3	28
4	20
5	35





HIERARCHICAL CLUSTERING





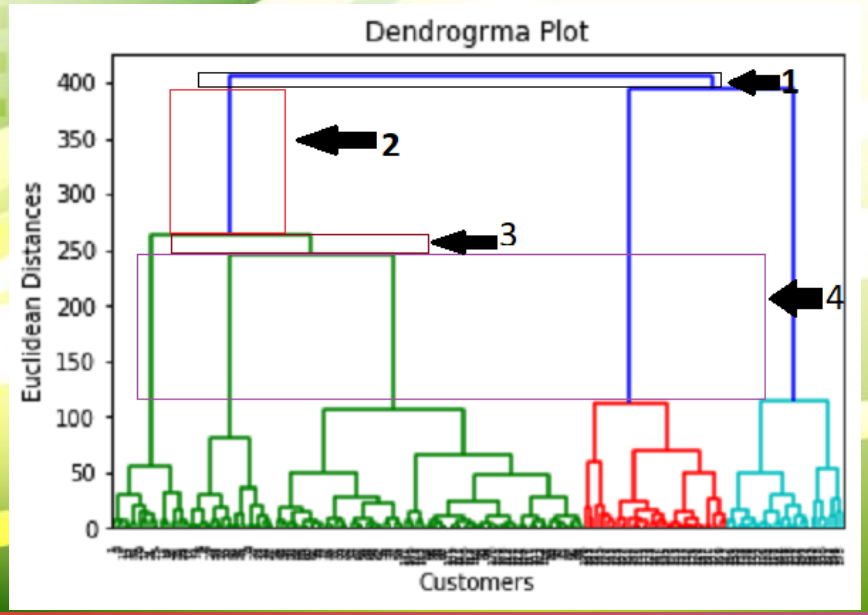
Distance cut-offs toward left result in fewer clusters with more diverse observations within each cluster.

Cut-offs toward right result in greater number of clusters with more similar observations within each cluster.

HIERARCHICAL CLUSTERING

Finding optimal number of clusters using Dendrogram:

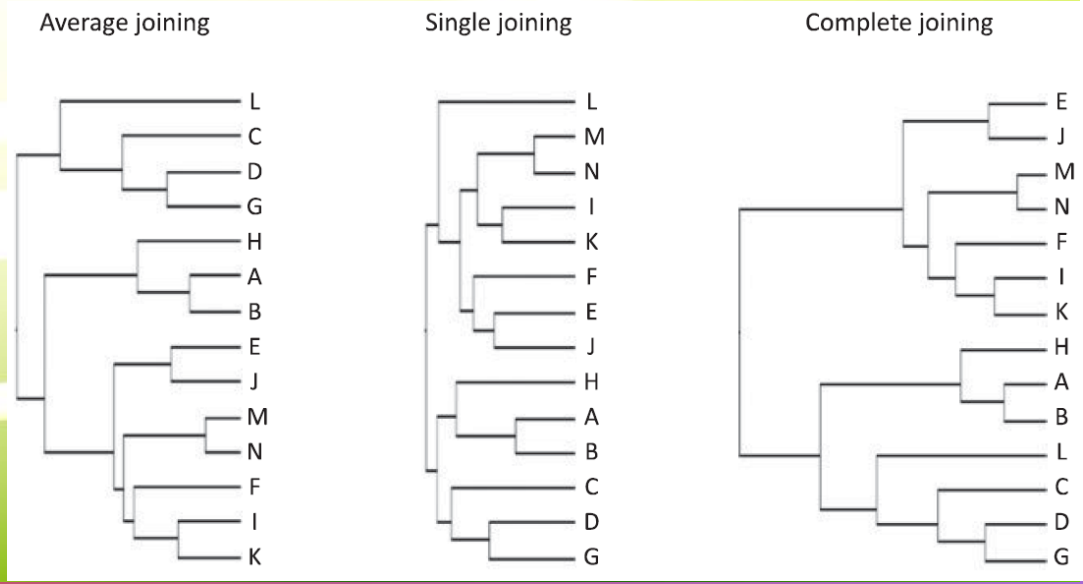
- Find maximum vertical distance that does not cut any horizontal bar.
- The 4th distance looks maximum; so, number of clusters will be 5 (vertical lines in this range).
- The 2nd number also approximately equals the 4th distance *(but K-means algorithm also gives 5 clusters as result)*.
- So, optimal number of clusters will be 5.



HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering

- Different joining/linkage rules change final hierarchical clustering.
- Hierarchical clustering of same set of observations using average, single, and complete linkage rules would be completely different.
  - barrier for merging observations and clusters is lowest with single linkage approach → clustering dendrogram may contain chains of clusters as well as clusters that are spread out.
  - barrier to joining clusters is highest with complete linkage → possible that an observation is closer to observations in other clusters than the cluster to which it has been assigned.
  - average linkage approach moderates the tendencies of single or complete linkage approaches

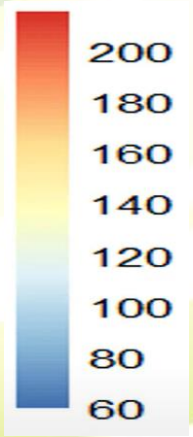


HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering Heatmap

- Heatmap is a graphical representation of data where values are depicted by color.
- Cluster heatmaps is a heatmap where rows and columns of data matrix have been ordered according to the output from clustering.

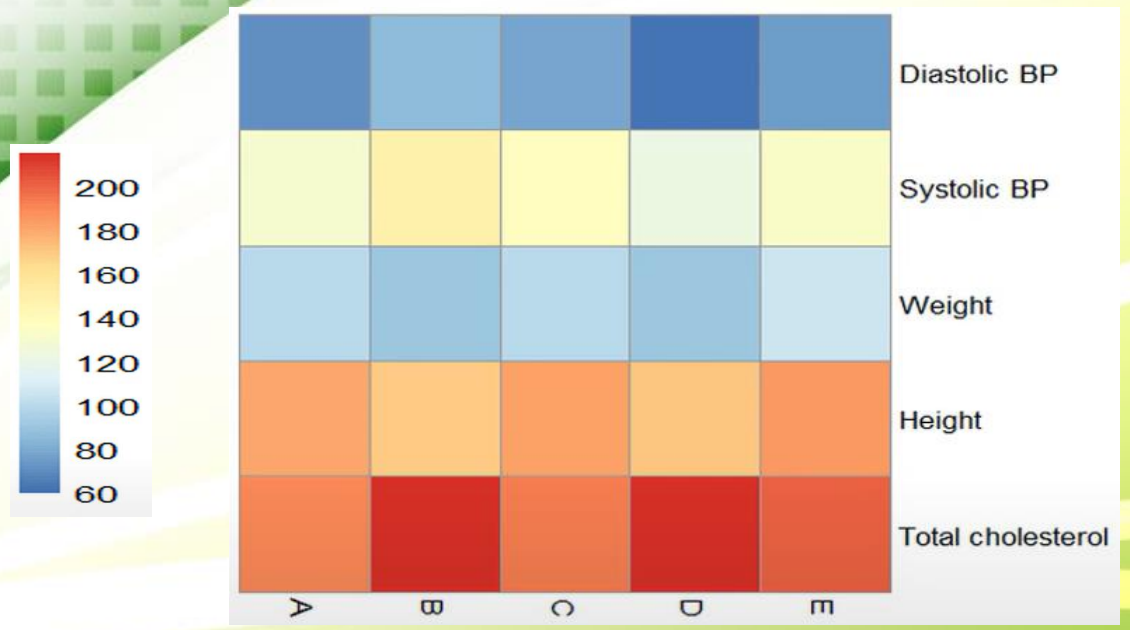
	A	B	C	D	E
Diastolic BP (mmHg)	70	86	78	60	75
Systolic BP (mmHg)	130	147	137	121	134
Weight (kg)	100	90	100	90	106
Height (cm)	180	170	181	171	185
Total cholesterol (mg/dl)	190	215	192	215	200



	A	B	C	D	E
Diastolic BP (mmHg)	70	86	78	60	75
Systolic BP (mmHg)	130	147	137	121	134
Weight (kg)	100	90	100	90	106
Height (cm)	180	170	181	171	185
Total cholesterol (mg/dl)	190	215	192	215	200

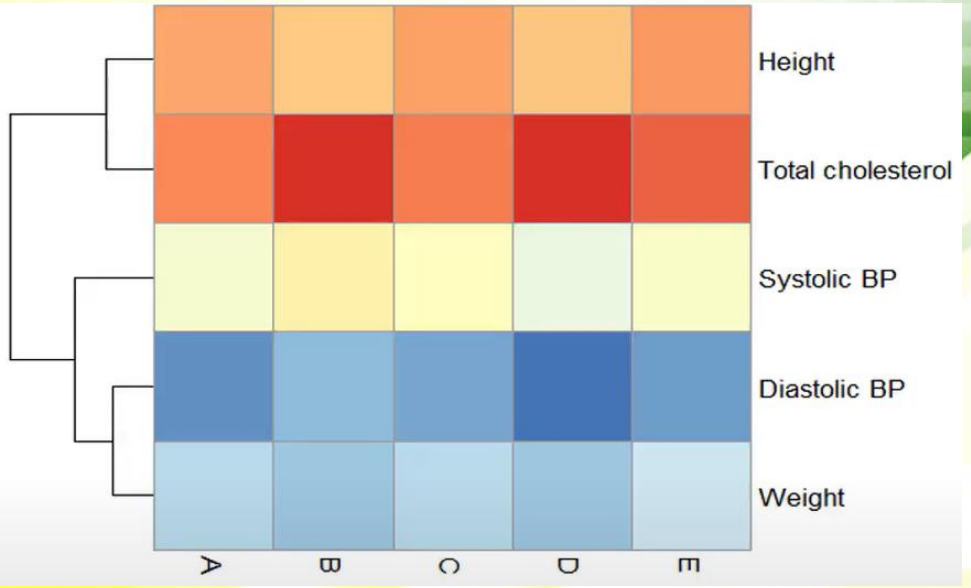
HIERARCHICAL CLUSTERING

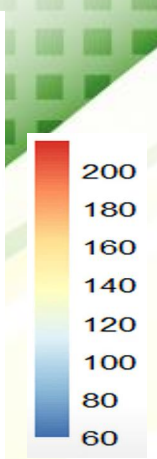
	A	B	C	D	E
Diastolic BP (mmHg)	70	86	78	60	75
Systolic BP (mmHg)	130	147	137	121	134
Weight (kg)	100	90	100	90	106
Height (cm)	180	170	181	171	185
Total cholesterol (mg/dl)	190	215	192	215	200

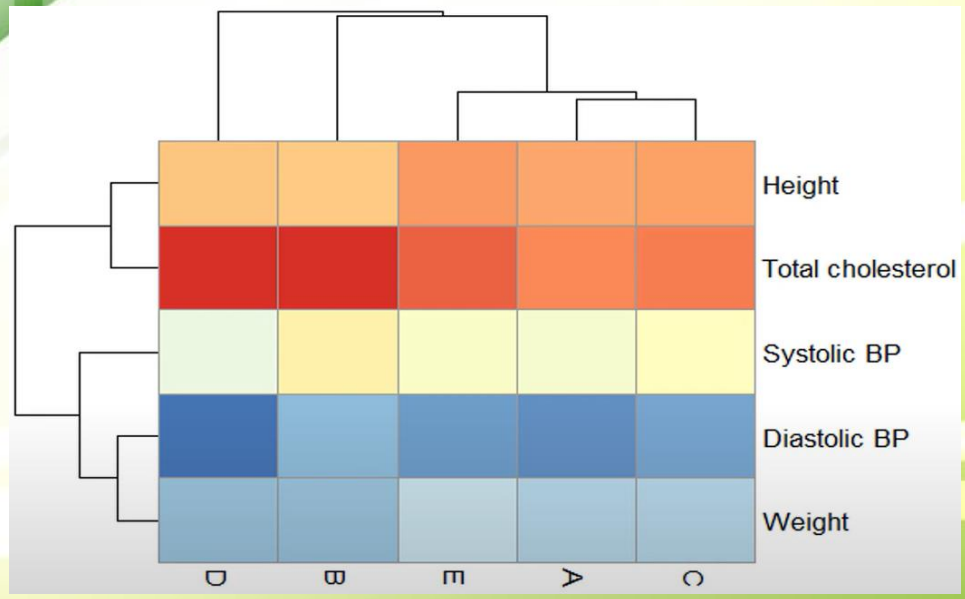


HIERARCHICAL CLUSTERING

Agglomerative Hierarchical Clustering Heatmap

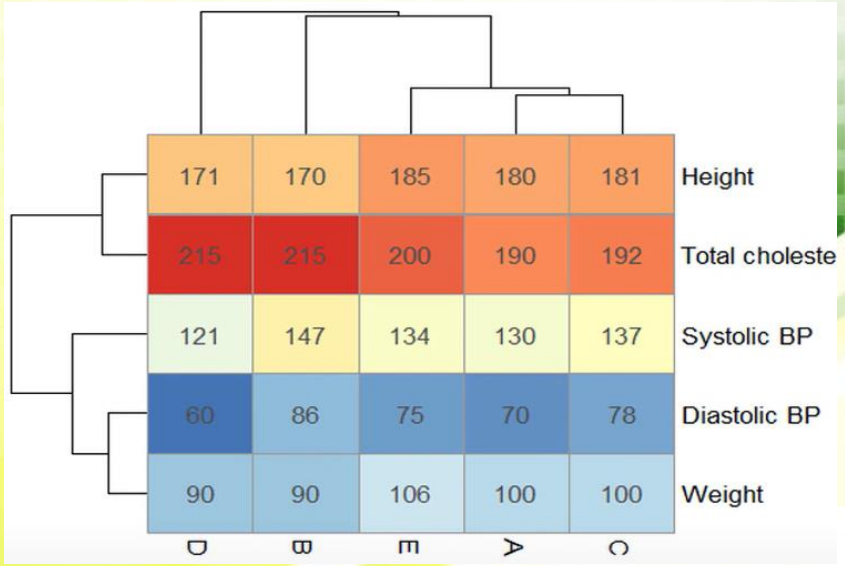




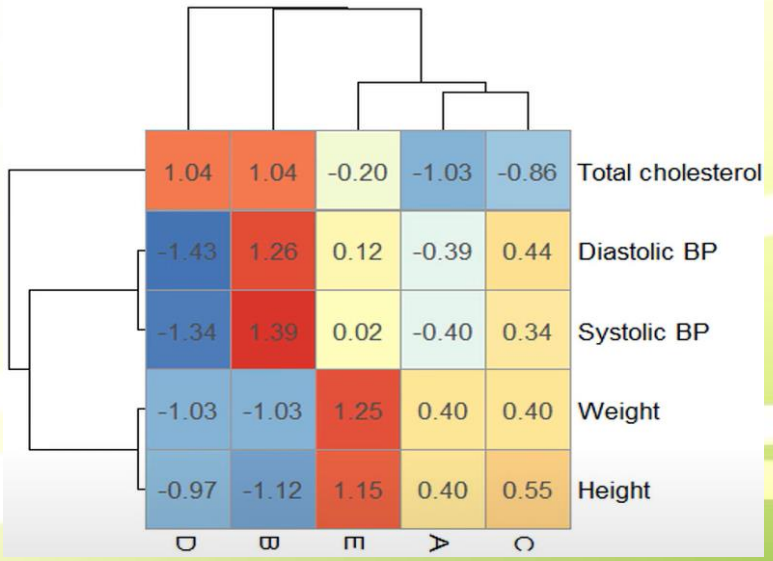


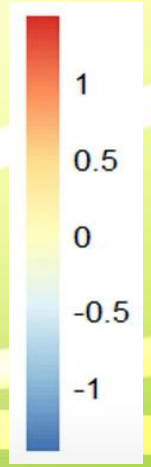
HIERARCHICAL CLUSTERING

Normalization



$$Z = \frac{x_i - \bar{x}}{SD}$$





HIERARCHICAL CLUSTERING

Apply Agglomerative Hierarchical Clustering to group the following data into 2 clusters. Prepare a dendogram for the same.

	x1	x2
A	3	8
B	9	3
C	3	1
D	6	9
E	8	2

K-MEANS CLUSTERING

- K-Means Clustering is an unsupervised machine learning algorithm.
- Non-hierarchical method for grouping a data set.
- K-Means attempts to classify data without having first been trained with labeled data.
  - Once algorithm has been run and groups are defined, any new data can be easily assigned to most relevant group.
- *Top-down* approach: starts with predefined number of clusters and assigns all observations to each of them.
- Computationally faster and can handle greater numbers of observations than agglomerative hierarchical clustering.

<u>Disadvantages:</u>

- Number of groups (K) must be specified before creating clusters and this number is not guaranteed to provide best partitioning of observations.
- when dataset contains many outliers, k-means may not create an optimal grouping; because reassignment of observations is based on closeness to cluster center and outliers pull cluster center in their direction *(WRONG)*.
- No hierarchical organization is generated using k-means clustering and hence there is no ordering of individual observations.

K-MEANS CLUSTERING

- K-means clustering is a partition-based clustering algorithm.
- Aims to group similar data points into k clusters based on their distance from k randomly selected centroids.
- Iteratively assigns data points to the nearest centroid and updates the centroids until convergence is reached.
- K-means clustering is sensitive to the initial random selection of centroids and may converge to a local optimum instead of a global optimum.

How to initialize centers in K-means

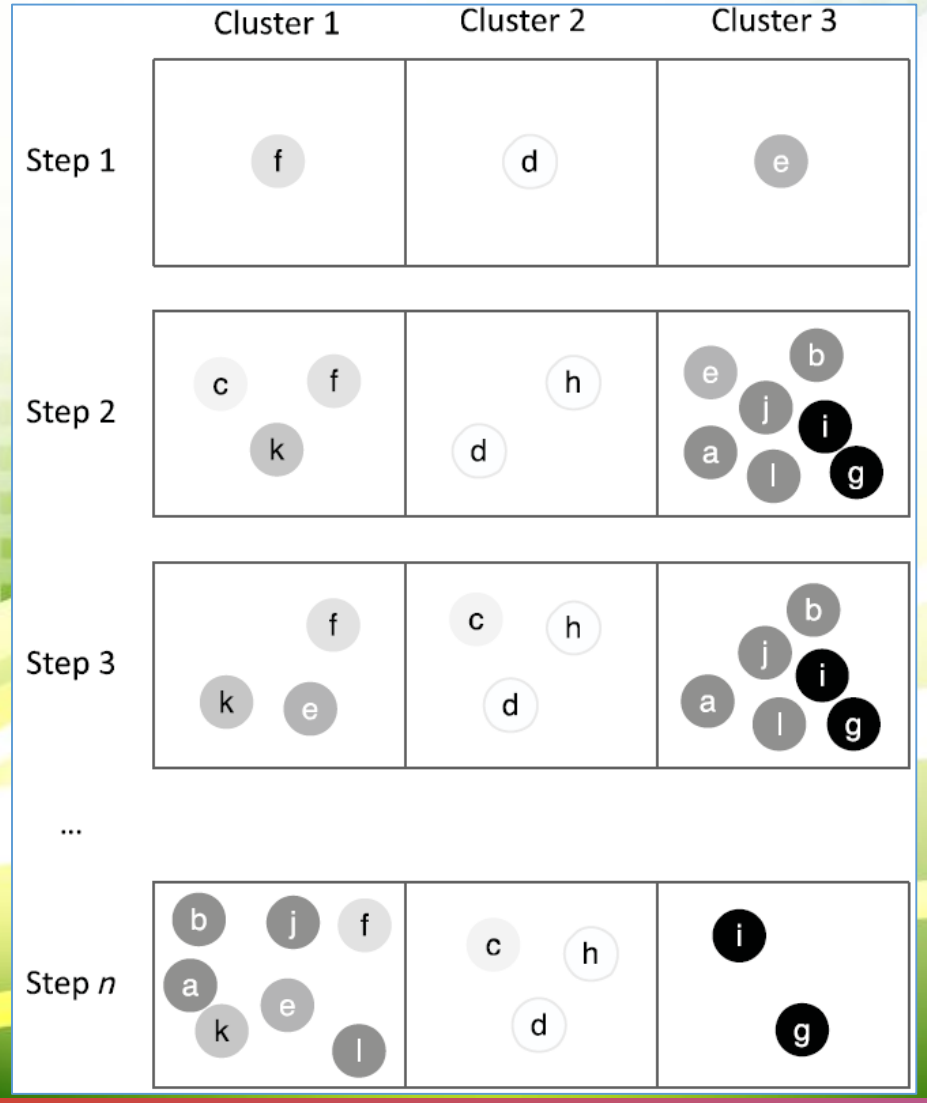
- Random Points From Data Set
- Look For Dense Regions of Space
- Space them uniformly around the feature space

K-MEANS CLUSTERING

1. Select **K** (= 2) random points as cluster centers called centroids.
2. Assign each data point to the closest cluster by calculating its distance with respect to each centroid.
3. Determine the new cluster center by computing the average of the assigned points.
4. Repeat steps 2 and 3 until none of the cluster assignments change.

K-MEANS CLUSTERING

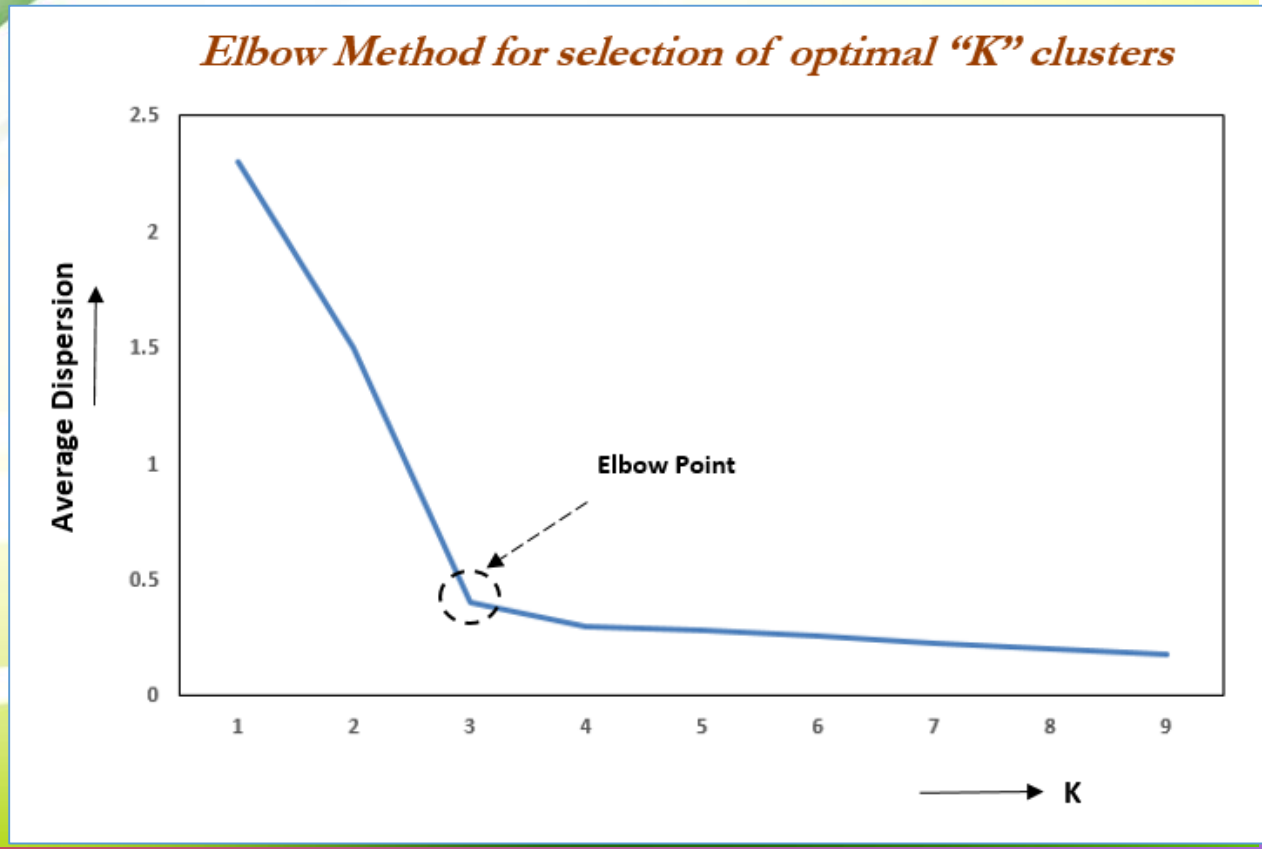
Example



K-MEANS CLUSTERING

Choosing right number of clusters

- Most common technique is **elbow method**.
- Elbow method is used to determine optimal number of clusters in K-means clustering.
- Elbow method plots the value of cost function produced by different values of K.
- Value of K at which improvement in distortion declines the most is called the elbow.
- At this point, STOP division of data into further clusters.



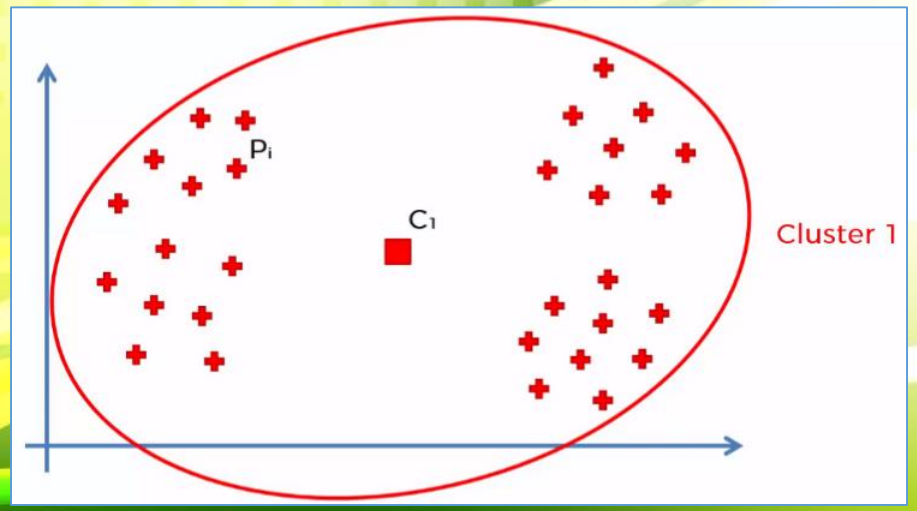
K-MEANS CLUSTERING

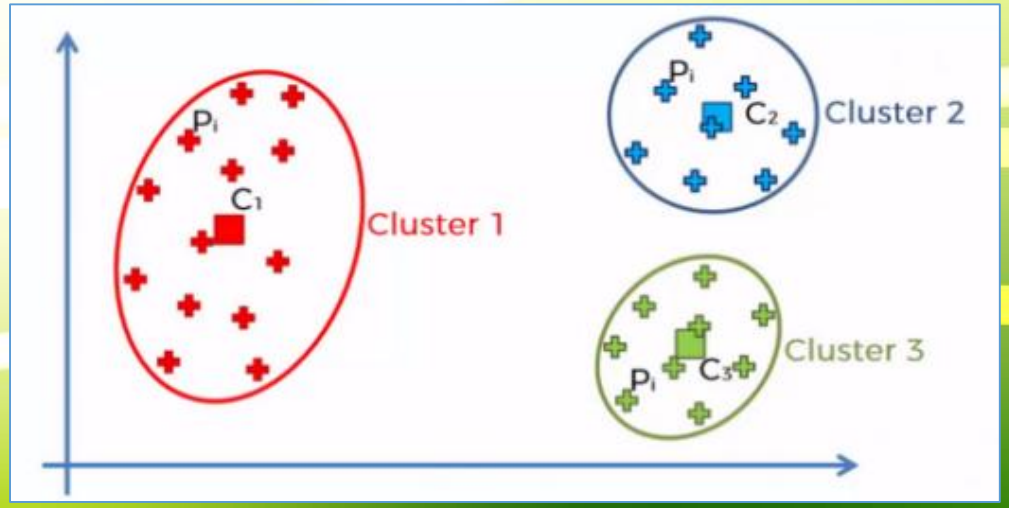
Choosing the right number of clusters

- Correct idea of number of cluster is very important for model performance.
- Choosing right number of cluster is very difficult, at first.
- Within Cluster Sum of Squares (WCSS) helps finding 'K' value.
  - WCSS: sum of squares of distances of data points in each and every cluster from its centroid.
  - Main idea is to minimize distance between data points and centroid of clusters.
- **Example**, computed WCSS for K=1 is greater than WCSS calculated for *K=3*.

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where x_i = data point and c_i = closest point to centroid





K-MEANS CLUSTERING

Cluster the following eight points (with (x, y) representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Consider the 3 initial cluster centers as: A1(2, 10), A4(5, 8) and A7(1, 2) to perform 2 iteration.

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

K-MEANS CLUSTERING

Cluster the following eight points (with (x, y) representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Consider the 3 initial cluster centers as: A1(2, 10), A4(5, 8) and A7(1, 2) to perform 2 iteration.

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

Revised Cluster centers:

C1 (2,10)

C2 (6,6)

C3 (1.5,3.5)

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

K-MEANS CLUSTERING

- In K-means clustering, datapoints are assigned to clusters based on their similarity.
- This is done such that the objective function (Sum of squares,..) is minimized.
- The mathematical condition for the K clusters C_k and the K centroids \mu_k can be expressed as:

$$\text{Minimize} \sum_{k=1}^{K} \sum_{x_n \in C_k} ||x_n - \mu_k||^2 \text{ with respect to } C_k, \mu_k$$

Lloyd's k-Means Algorithm

- K-means takes a dataset X of N points as input, together with a parameter K (number of clusters to create).
- Output is a set of K cluster centroids and a labeling of X that assigns each points in X to a unique cluster.
- All points within a cluster are closer in distance to their centroid than they are to any other centroid.
- All datapoints are assigned to their closest centroid following the recalculation of centroids as mean over all their assigned datapoints.
- Lloyd algorithm loops over these last steps until no point changes its assignment.

K-MEANS CLUSTERING

Lloyd's k-Means Algorithm

Algorithm 1 Lloyd

1: choose k as the number of centroids
2: randomly assign all points to a centroid
3: calculate the centroids as mean of their assigned points
4: **repeat**
5: reassign each datapoint to its closest centroid
6: recalculate centroids as mean over their assigned datapoints
7: **until** convergence

- Lloyd suffers greatly as forming of clusters is heavily influenced by the choosing of initial centroids.
- Lloyd might get stuck in a local optima, depending on its initialization.

K-MEANS CLUSTERING

MacQueen's k-Means Algorithm

- When Lloyd updates the assignments of datapoints, it does not update the centroids.
- This is problematic, as with each new assignment, the centroid changes its position.
- It can happen that a datapoint is wrongfully assigned to a centroid simply because said centroid was not updated.
- MacQueen addresses this, by updating centroids with each new assignment. Leads in more computation time.
- MacQueen initializes the same way as Lloyd. But, while iterating over datapoints, and reassigning them, we recalculate the affected centroids. This is done in a loop until no point changes its assignment.

Algorithm 1 MacQueen

```
choose k as the number of clusters
randomly choose k datapoints as centroids
repeat
    for each datapoint do
        assign point to closest centroid
        recalculate centroid as mean over all points assigned
    end for
until convergence
```

K-MEANS CLUSTERING

MacQueen's k-Means Algorithm

- MacQueen's k-means algorithm and Lloyd's algorithm are both iterative algorithms used for clustering data into k clusters. However, they differ in a few ways:
- Initialization: In Lloyd's algorithm, the initial k cluster centers are randomly selected from the data points. In MacQueen's algorithm, the initial k cluster centers are selected from a random subset of the data points.
- Updating cluster centers: In Lloyd's algorithm, the cluster centers are updated by computing the mean of all data points assigned to that cluster. In MacQueen's algorithm, the cluster centers are updated by computing the mean of only the data points that were newly assigned to that cluster in the current iteration.
- Memory usage: Lloyd's algorithm requires storing all data points and their cluster assignments in memory, which can be prohibitive for large datasets. MacQueen's algorithm only needs to store the current subset of data points and their assignments, making it more memory-efficient.
- Overall, MacQueen's algorithm can be faster and more memory-efficient than Lloyd's algorithm, especially for large datasets, but it may converge to a suboptimal solution due to its random initialization.

K-MEANS CLUSTERING

Hartingan-Wong k-Means Algorithm

- Lloyd faces initialization Problem. MacQueen improves on it, but still somewhat suffers also from this.
- Hartigan-Wong k-Means uses a deterministic initialization procedure, which iteratively selects new cluster centers based on the distance between data points and the current centers.
- Less prone to converge to a local optima than Lloyd.
- Applies more complex criterion to assign data points to clusters in each iteration, considering both the distance between the point and current cluster center, as well as the distances between the data point and the other cluster centers.
- Cluster centers are updated by iteratively searching for the optimal new center within the set of data points that are currently assigned to the cluster.

Algorithm 1 Hartigan-Wong

```
choose k as the number of centroids
randomly assign all points to a centroid
calculate the centroids as mean of their assigned points
repeat
    for each datapoint d do
        for each centroid c do
            assign datapoint d to centroid c
            compute the sum of squared distances from each point to its
            centroid
        end for
        assign d to the centroid which resulted in the smallest sum
        recalculate centroids as mean over all points assigned
    end for
until convergence
```

K-MEDIAN CLUSTERING

- k-median clustering algorithms aims to partition a given dataset into K clusters.
- Differs from k-means on how to define cluster center and how to compute distance between data points and center.
- In k-median, cluster center is the median of the data points within that cluster.
- Algorithm assigns each data point to the closest cluster center and then updates the center of each cluster to the median of the points assigned to that cluster.
- This process is repeated until convergence, similar to k-means.
- K-median is more robust to outliers in dataset since it uses median instead of mean. Median is less sensitive to outliers than the mean since it is not affected by extreme values.
- K-median is computationally more expensive than k-means; computing median of data points in each cluster can be more time-consuming than computing the mean.

K-MEAN++ CLUSTERING

- k-means++ is clustering algorithms that aim to partition a given dataset into K clusters.
- Differs from K-means on how to initialize the cluster centers.
- K-means initializes cluster centers randomly, which can lead to suboptimal results, especially when the initial centers are far from the true centers.
- k-means++ uses a more sophisticated initialization strategy that aims to choose initial cluster centers in a way that improves convergence of the algorithm and leads to better results.
- First center is chosen uniformly at random from the data points.
- Each subsequent center is chosen with a probability proportional to the square of its distance from the closest existing center. This process is repeated until K centers are selected.
- Such initialization strategy ensures that initial centers are well spread out and avoids the problem of having centers that are too close to each other.
- After initialization, both k-means and k-means++ algorithms proceed in similar manner, iteratively assigning each data point to nearest cluster center and then updating the center of each cluster based on the newly assigned data points.
- k-means++ converge to better solution faster, with fewer iterations & fewer chances of getting stuck in local minimum.
- K-means++ initialization is often used in practice to improve the performance of k-means, especially for large datasets and high-dimensional data.

K-MEDOIDS CLUSTERING

- k-medoids is a clustering algorithms that aims to partition a given dataset into K clusters.
- Differs on how to define cluster center and how to compute distance between data points and the center.
- In k-medoids, cluster center is defined as one of the actual data points in the cluster, called a "medoid."
- The medoid is the data point that minimizes the sum of the distances between it and all other points in the cluster.
- Algorithm iteratively assigns each data point to the closest medoid and then updates the medoid to data point that minimizes sum of distances to all other points in the cluster. This process is repeated until convergence.
- k-medoids is more robust to outliers in the dataset, since it uses actual data points as cluster centers instead of the mean, which can be affected by extreme values.
- Finding medoids for each cluster can be computationally expensive, especially for large datasets and high-dimensional data.
- K-medoids+ algorithm attempts to improve algorithm efficiency by using similar initialization strategy to k-means++.
- In k-medoids+, initial medoids are chosen randomly and then refined using probability-based approach.

K-MEDOIDS CLUSTERING

Three types of algorithms for K-Medoids Clustering:

- PAM (Partitioning Around Clustering)
- CLARA (Clustering Large Applications)
- CLARANS (Randomized Clustering Large Applications)

- PAM is most powerful algorithm of the three algorithms, but has disadvantage of time complexity.
- CLARA is an extension to PAM to support Medoid clustering for large data sets.
  - Selects data samples from data set, applies PAM on each sample, and outputs the best Clustering out of these samples.
  - More effective than PAM.
  - Need to ensure that selected samples aren't biased as they affect the Clustering of whole data.
- CLARANS algorithm selects a sample of neighbors to examine instead of selecting samples from the data set.
  - In every step, it examines the neighbors of every node.
  - Time complexity of this algorithm is better than both PAM and CLARA.

K-MEANS CLUSTERING

Apply K-means Clustering to group the following data into 2 clusters. Consider the 2 initial cluster centers as: B(9, 2) and E(8, 9).

	x1	x2
A	3	5
B	9	2
C	3	6
D	6	5
E	8	9

NEAREST NEIGHBOUR CLUSTERING

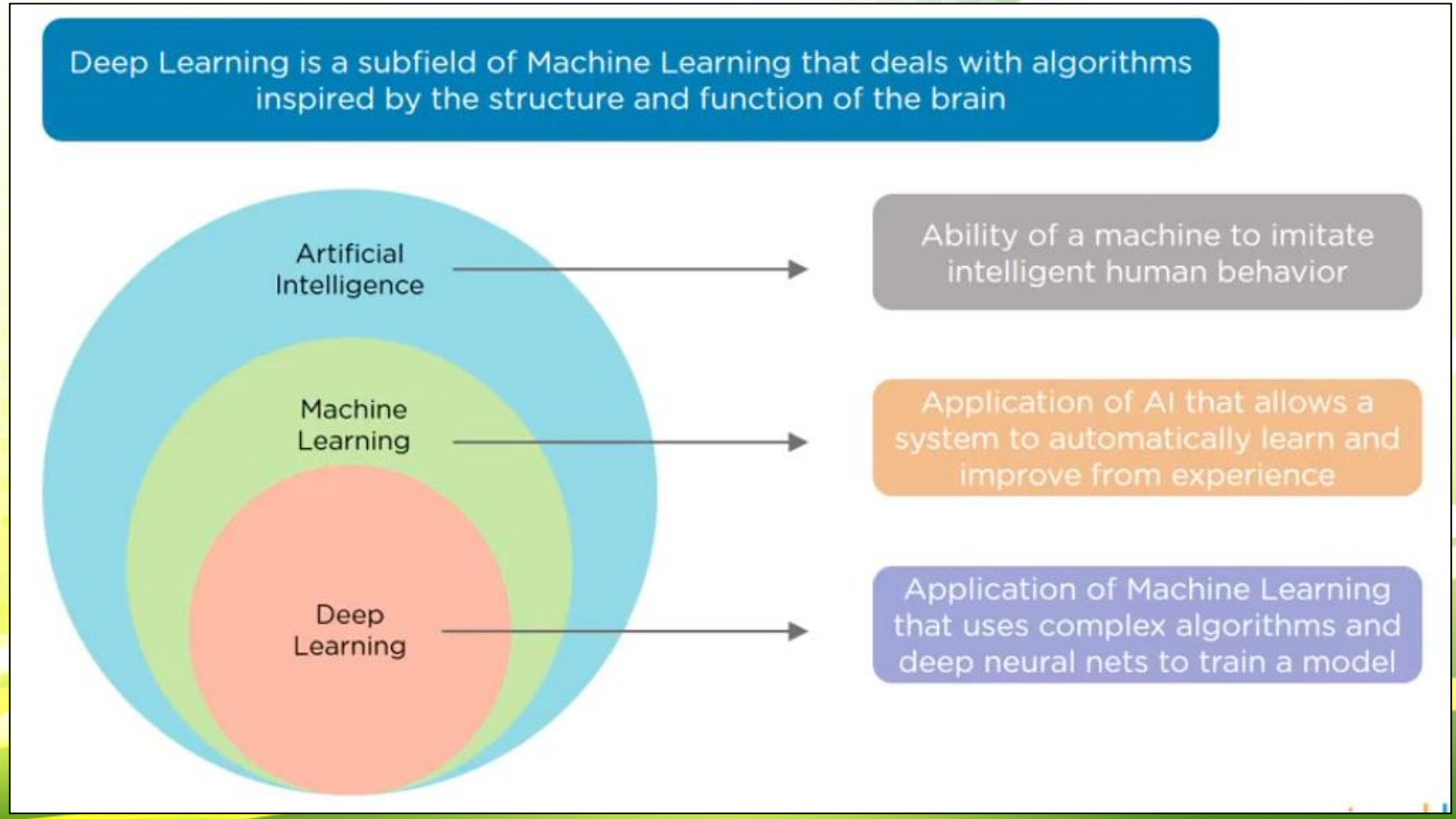
- Nearest neighbor clustering is a clustering technique that groups data points based on their proximity to their nearest neighbors.
- It is a type of unsupervised machine learning algorithm that is commonly used in pattern recognition, image processing, and data mining.
- Nearest neighbor clustering is a density-based clustering algorithm.
- Groups together data points that are close to each other in terms of their pairwise distances.
- The distance can be measured using various distance metrics, such as Euclidean distance or Manhattan distance.
- Does not require a pre-specified number of clusters and can automatically adjust number of clusters based on density of data points.
- Robust to outliers and can handle irregularly shaped clusters.

NEAREST NEIGHBOUR CLUSTERING

Steps in nearest neighbor clustering algorithm:

- Define a distance metric that measures pairwise distances between data points.
- Determine neighborhood of each data point, by identifying all other data points that are within a specified distance (radius is a hyperparameter that needs to be tuned based on characteristics of the data).
- Assign data points to clusters based on its connectivity to other data points. A data point is assigned to a cluster if it has at least a specified number of neighbors within its radius. (This number is called minimum number of points /minPts; another hyperparameter that needs to be tuned.
- Expand clusters by adding all reachable data points to each cluster. A datapoint is reachable if it can be reached from another data point by following a path of neighboring datapoints (unreachable datapoint is considered outlier).
- Repeat steps 3 and 4 until all data points have been assigned to clusters or marked as outliers.
- Evaluate quality of clusters based on some clustering criteria (silhouette score or Davies-Bouldin index).

Deep Learning



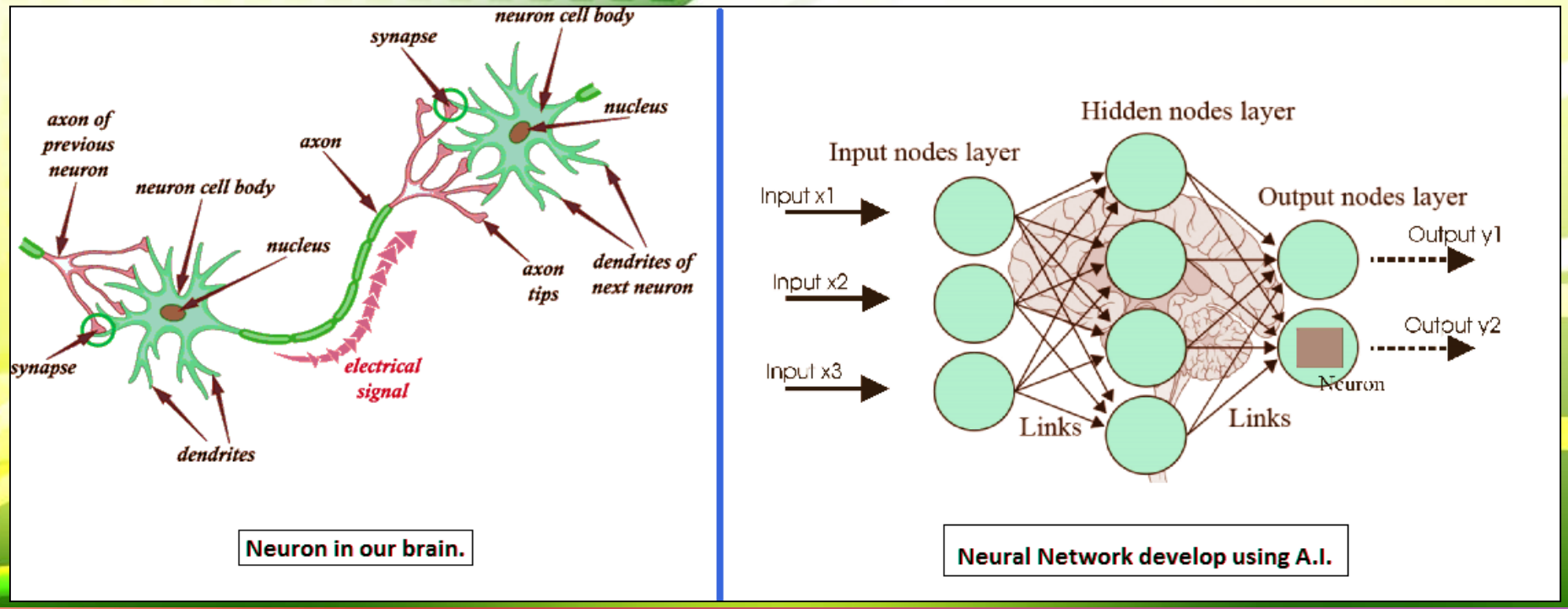
Neural Network

- DL uses the NN algorithm to do their work.
- NN is specific group of algorithms used for DL that models the data using graphs of Artificial Neurons.
- Neurons are mathematical models that *"mimics approximately how a neuron in human brain works"*.
- Series of algorithms that help to recognize underlying relationships in a set of data through a process that mimics the way human brain operates.

Neural Network

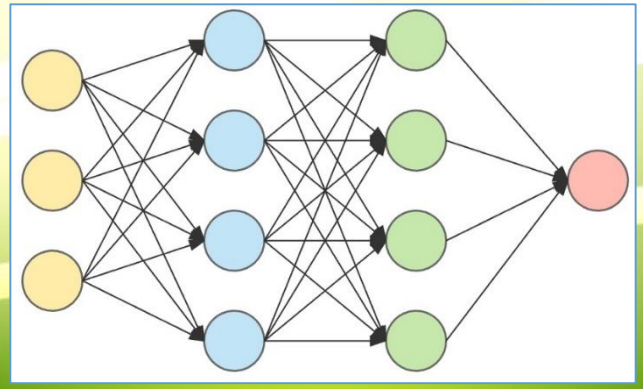
- Input Layer
- Hidden layer
- Output layer

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

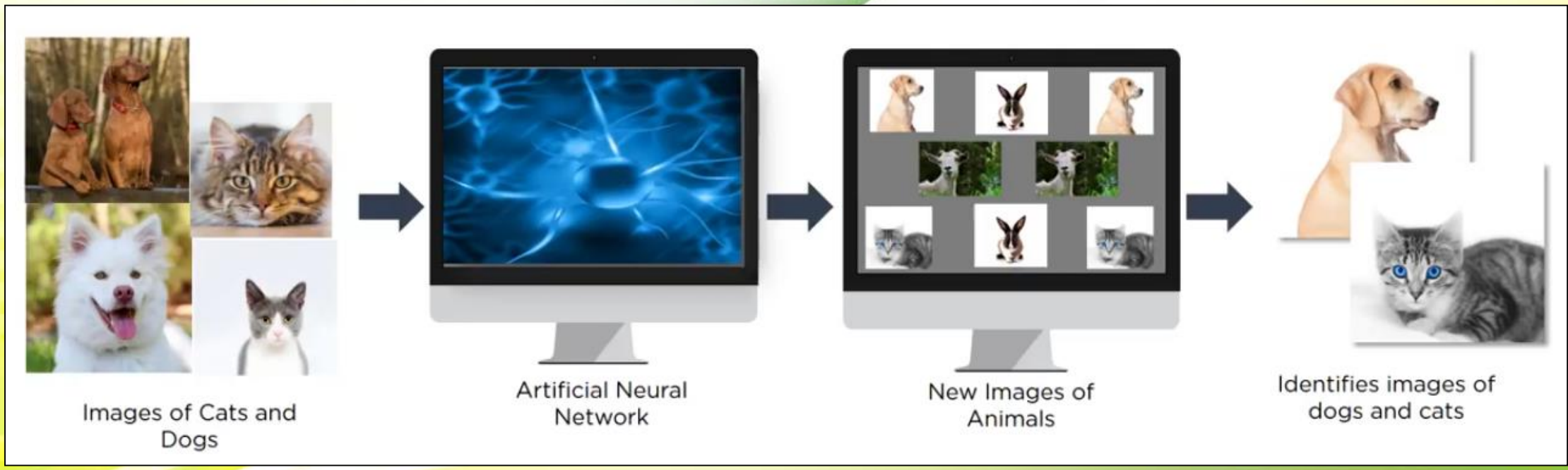


Neural Network

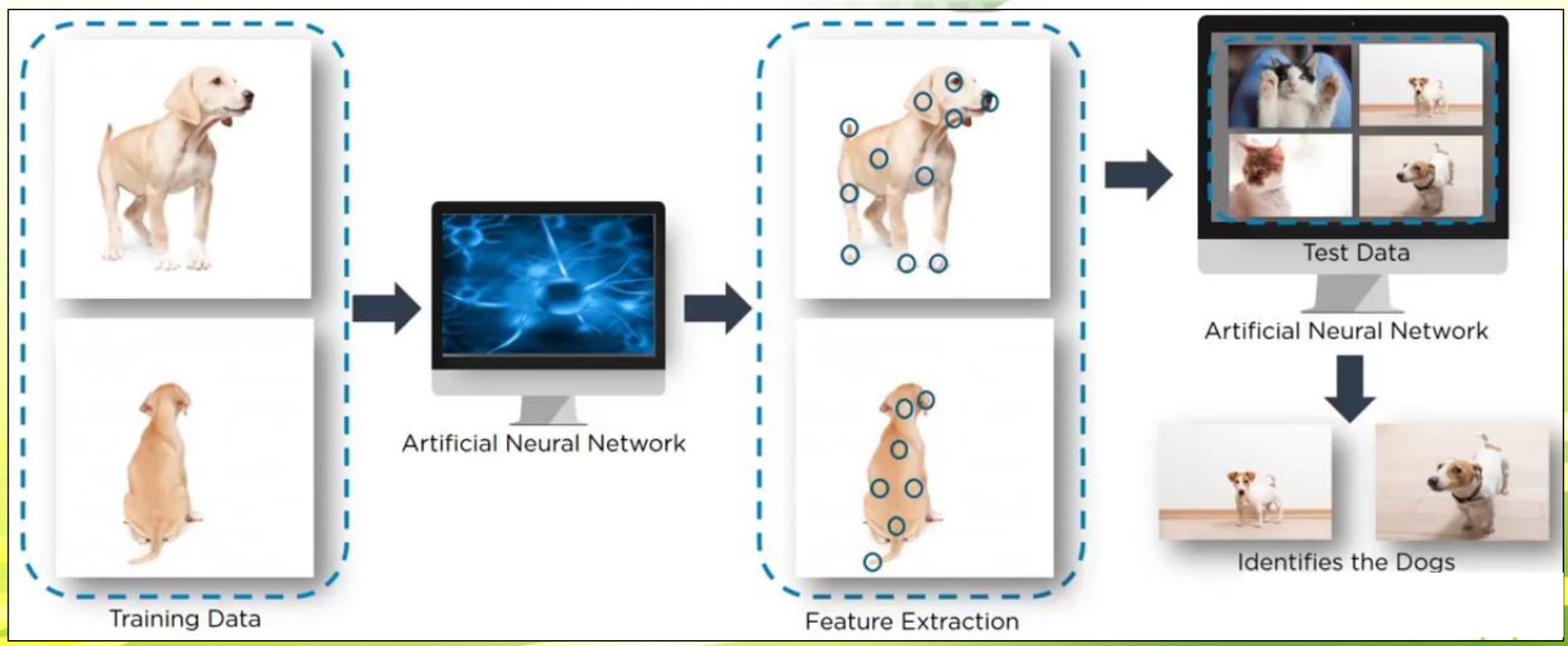
- "Neuron" in neural network is a mathematical function that collects and classifies information according to a specific architecture.
- NN is a strong statistical tool for classification and regression analysis.
- NN contains layers of interconnected nodes.
- Each node is a perceptron and is similar to a multiple linear regression.
- Perceptron feeds the signal produced by a linear model into an activation function that may be nonlinear.



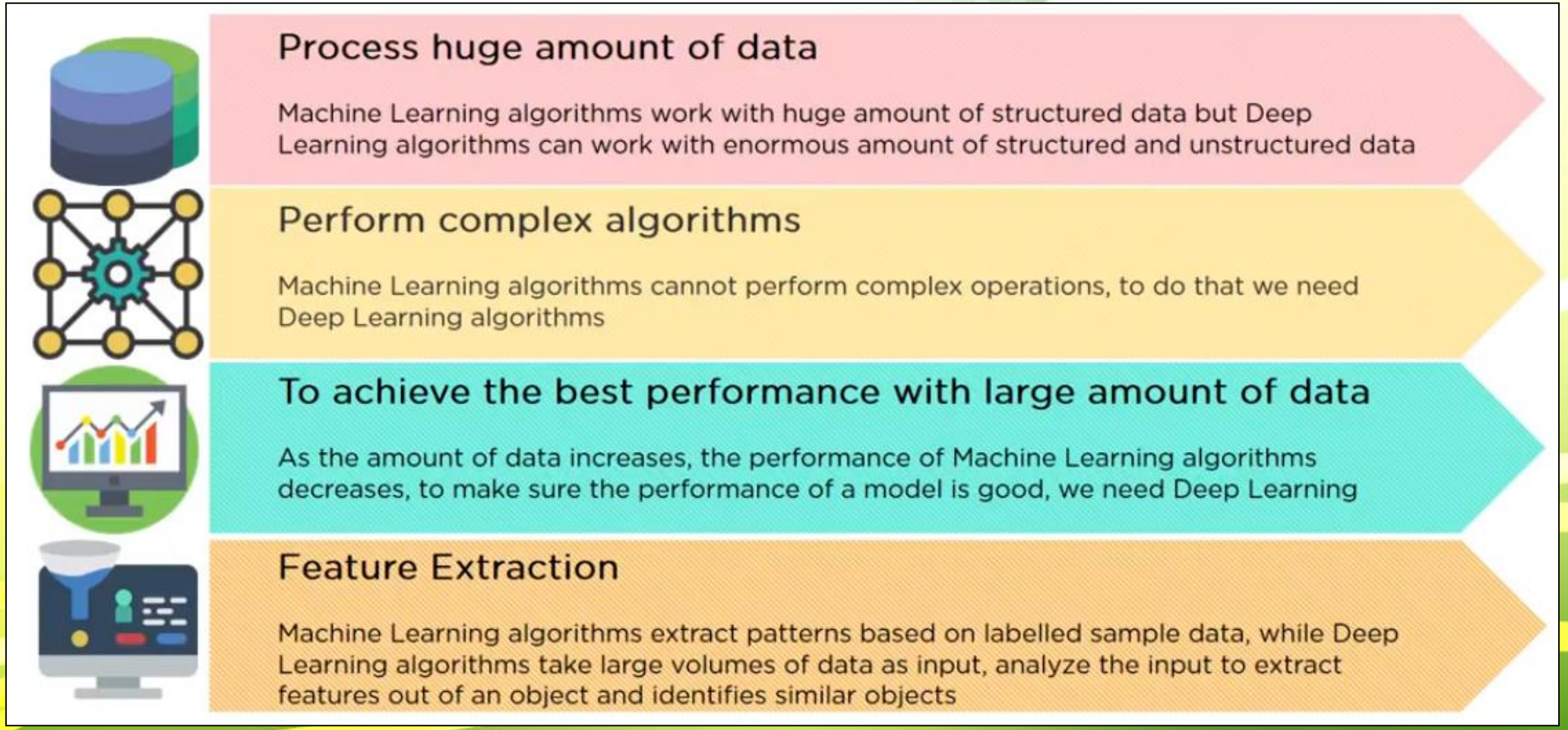
Deep Learning Application ___ Image Recognition



Deep Learning Application ___ Image Recognition

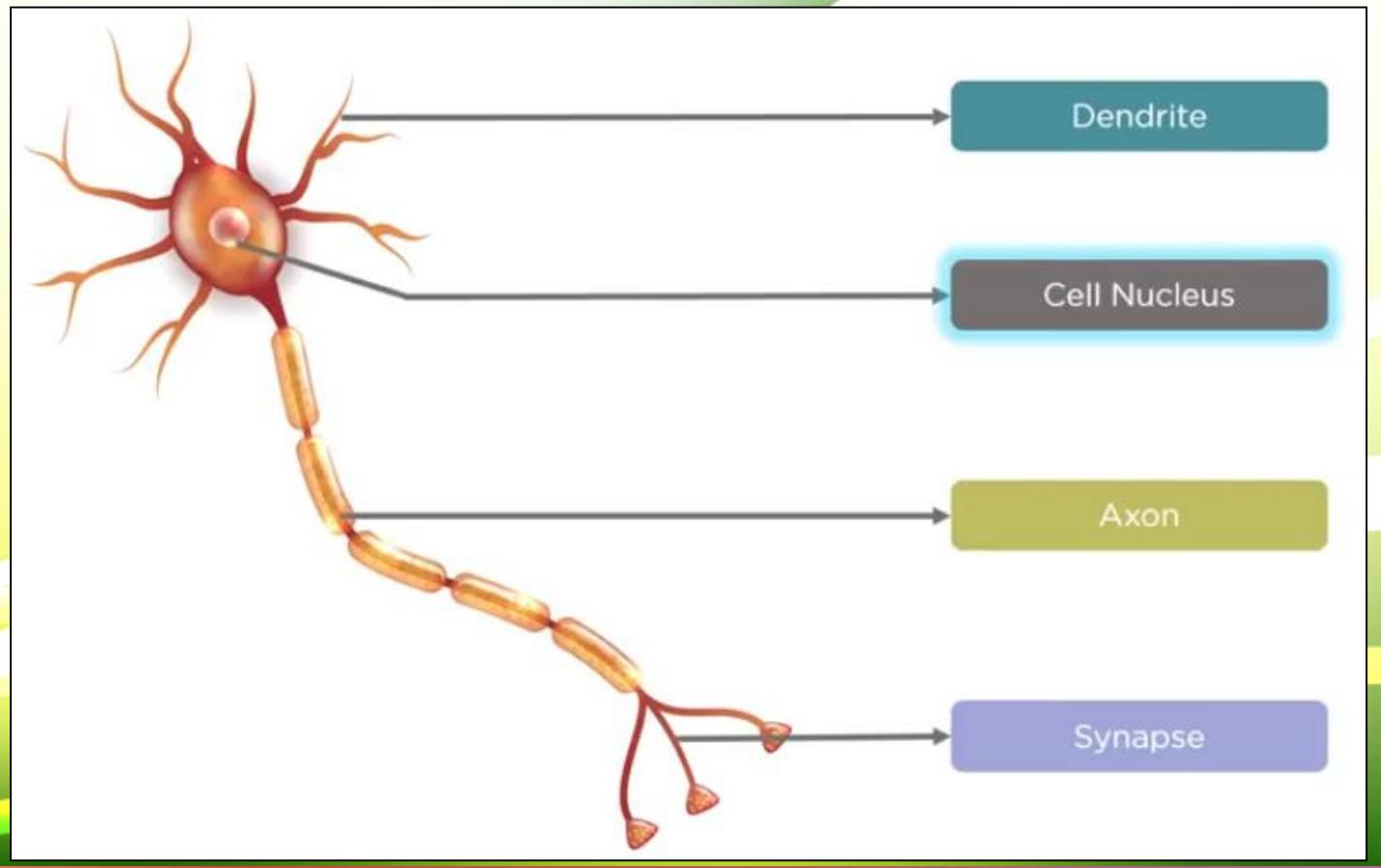


Deep Learning __ Advantages

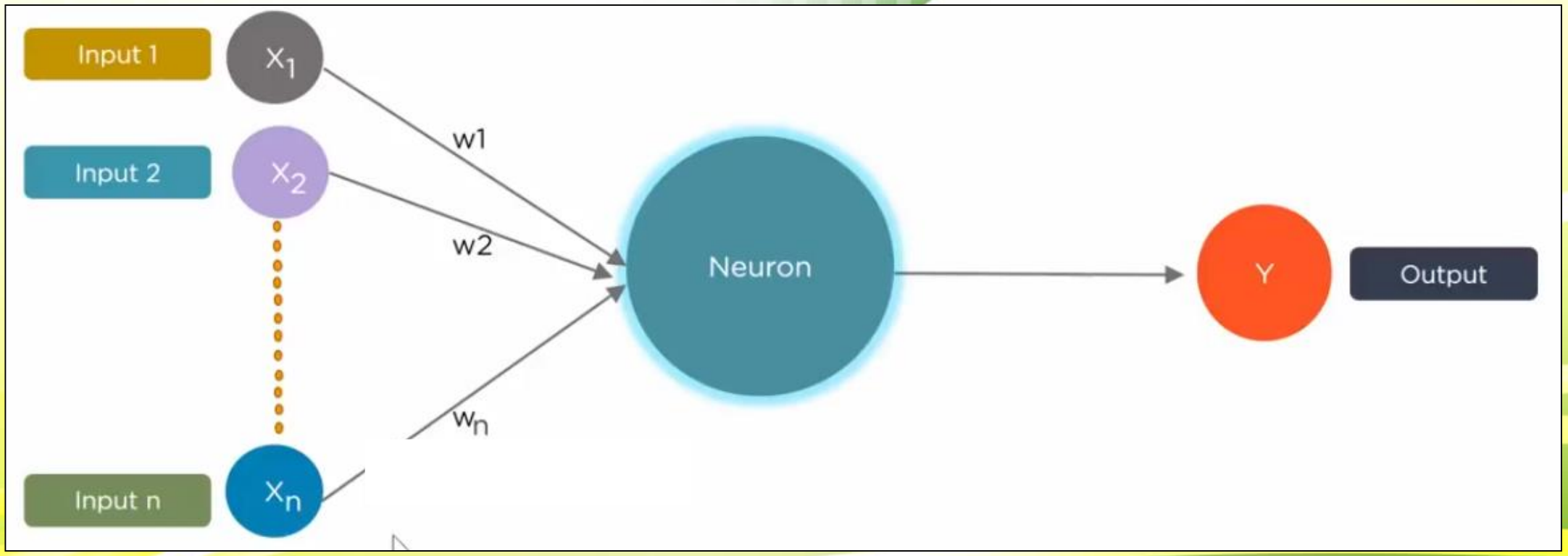


Deep Learning __ Neural Network

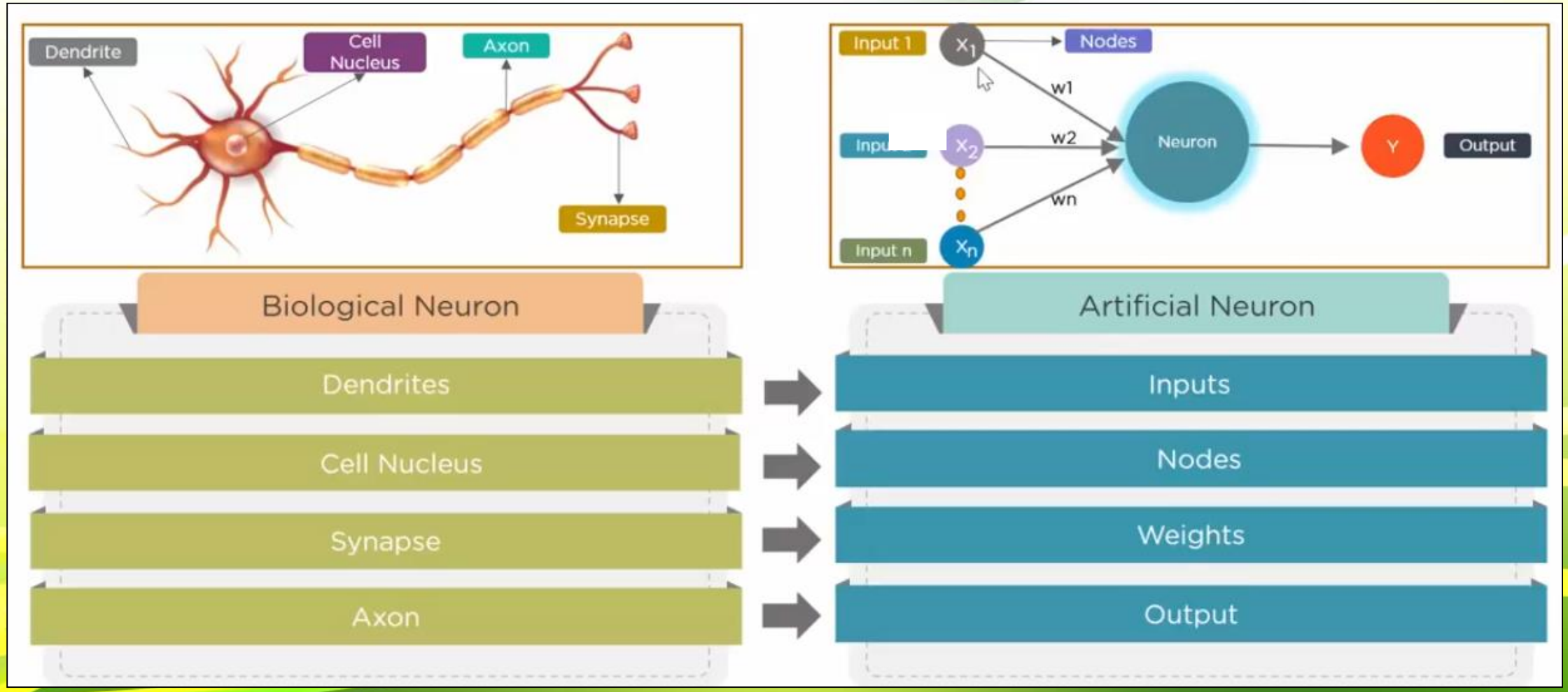
- DL is based on functioning of a Human brain.
- Biological Neural Network is as follows;



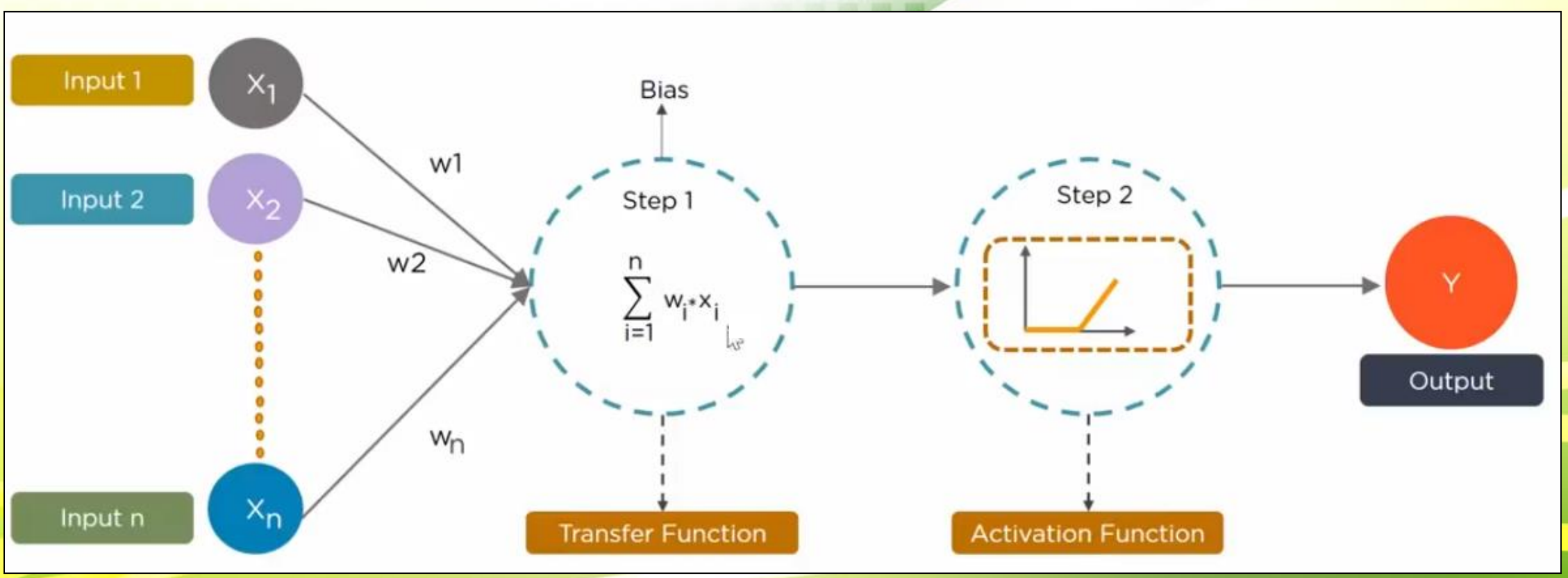
Deep Learning___ Artificial Neural Network



Deep Learning __ NN vs ANN

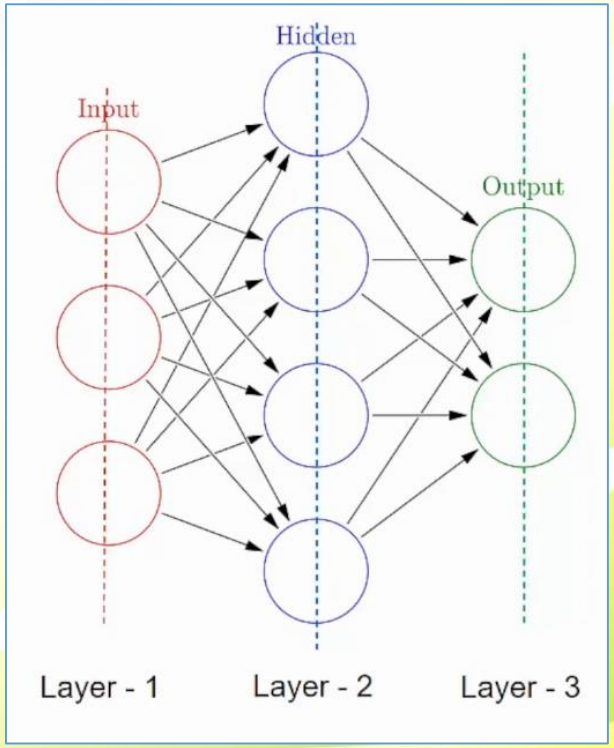


Deep Learning __ ANN



Deep Learning __ ANN

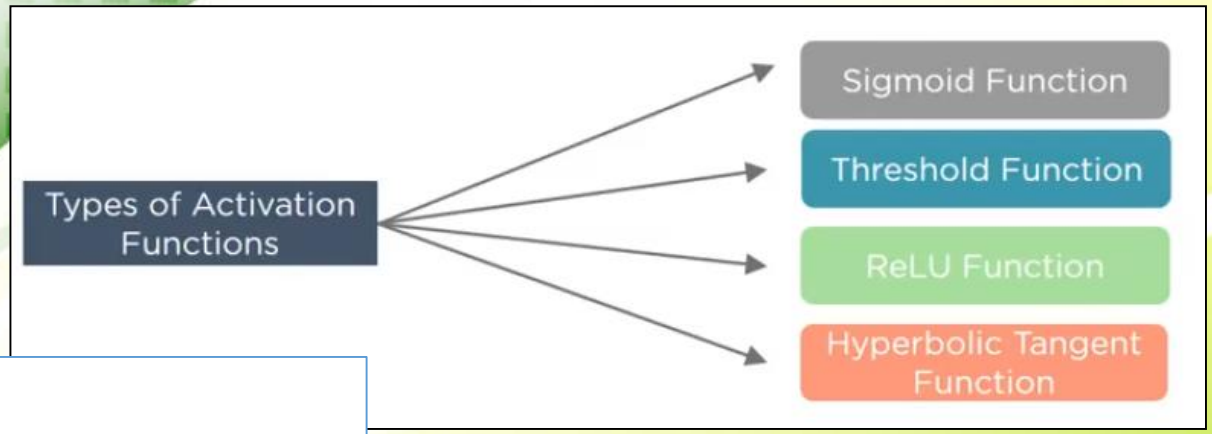
- NN is made up of vertically stacked components called Layers.
- 3 types of layers in a NN; Input, Hidden, Output.

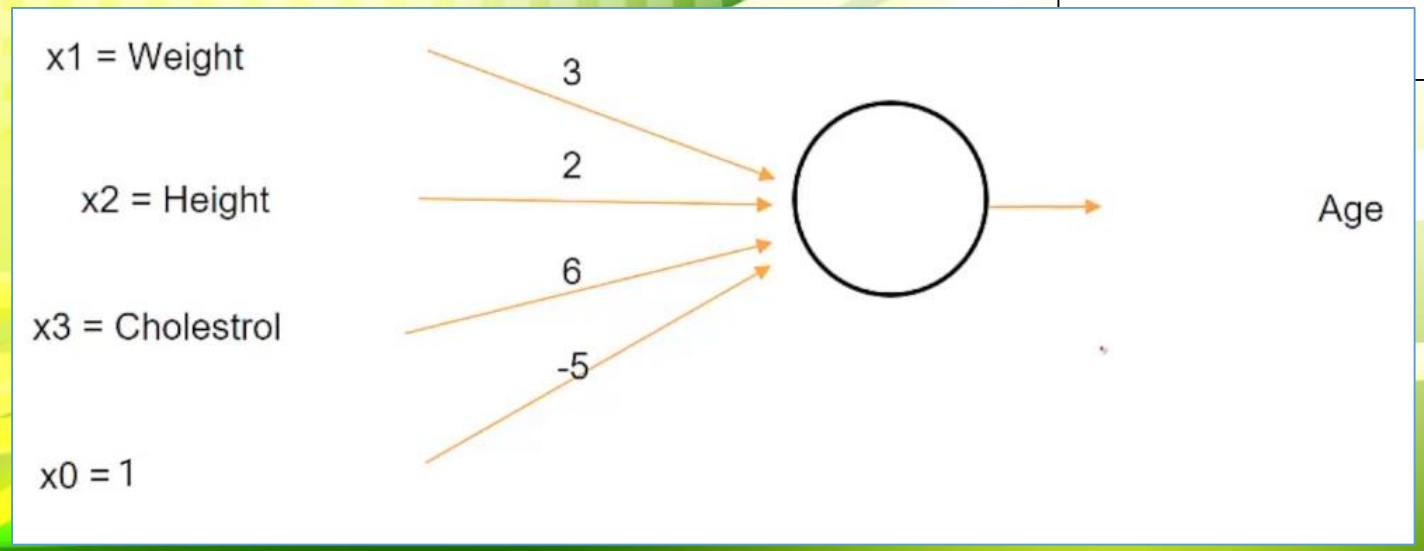


- Input Layer: First layer. Accept data and pass it to rest of the network.
- Hidden Layer: Second level of layer. One or more in number for a NN.
  - Responsible for excellent performance and complexity of neural networks.
  - Perform multiple functions (data transformation, automatic feature creation, etc.
- Output layer: Last type of layer. Holds result or output of the problem.

Deep Learning __ ANN

- Activation function takes the "weighted sum of inputs and the bias" as the input to the function and decides whether it should be fired or not.

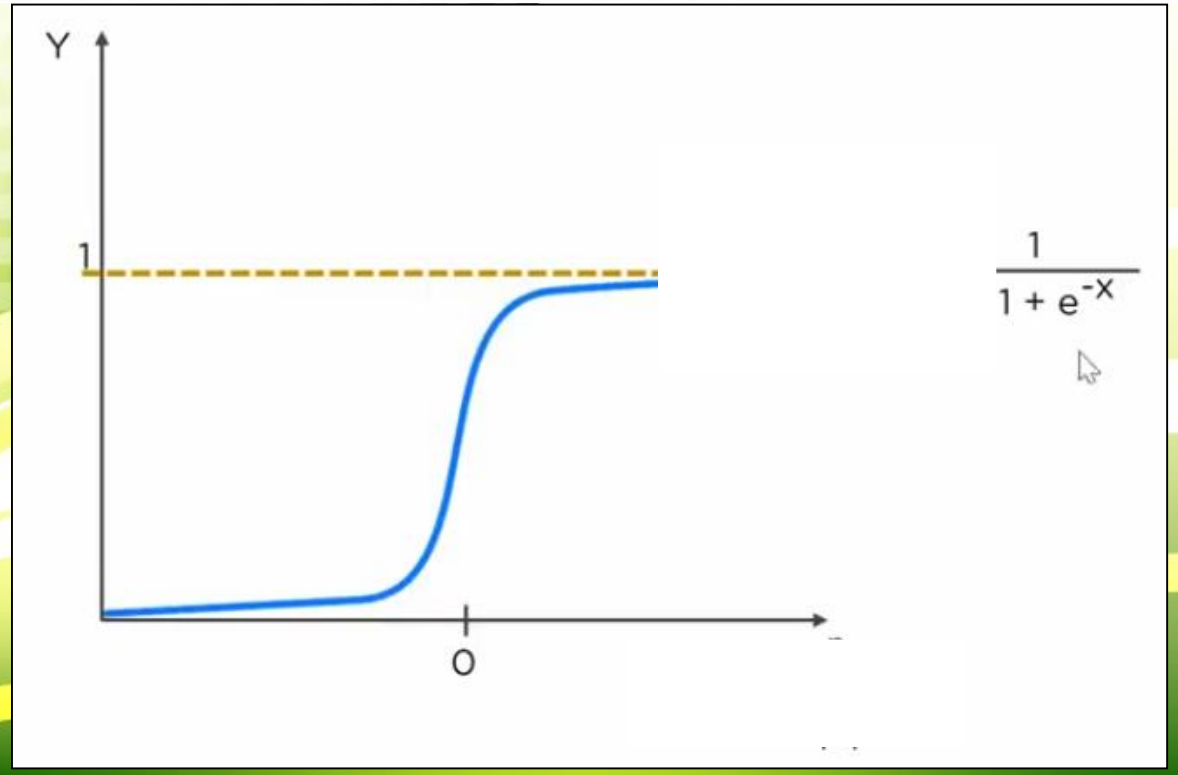




Deep Learning ___ ANN __ Activation function

Sigmoid function

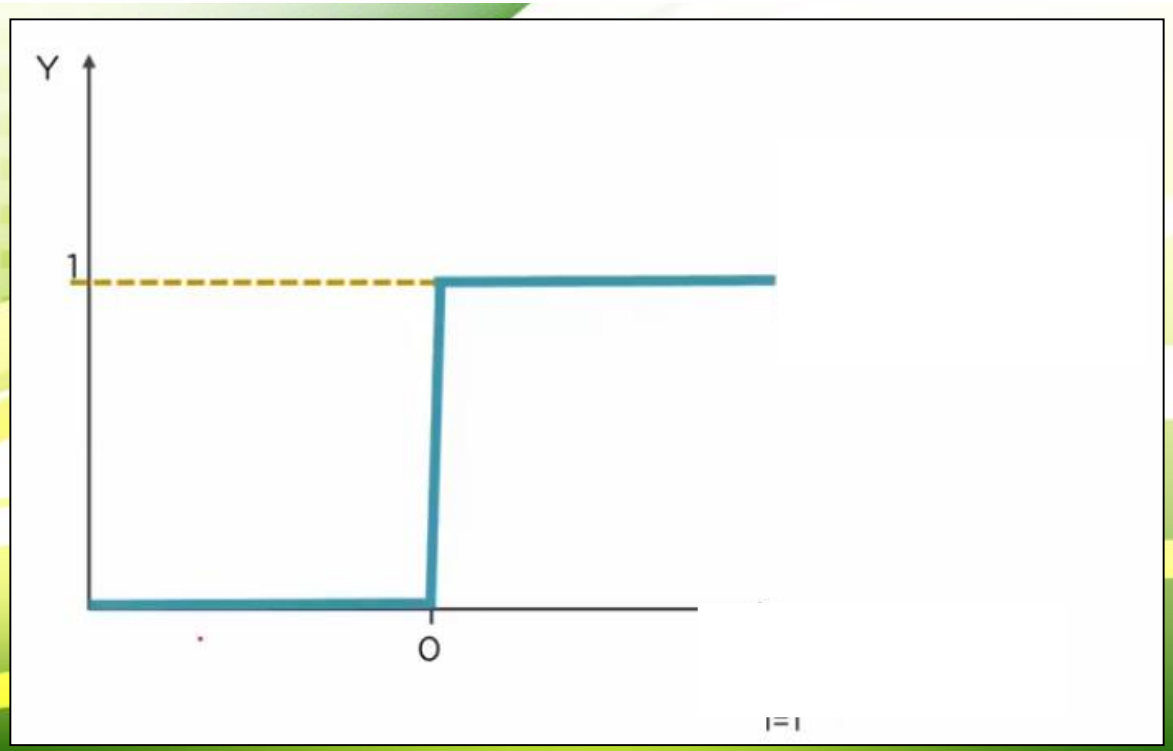
Used for models where we have to predict the probability as an output. It exists between 0 and 1.



Deep Learning ___ ANN __ Activation function

Threshold function

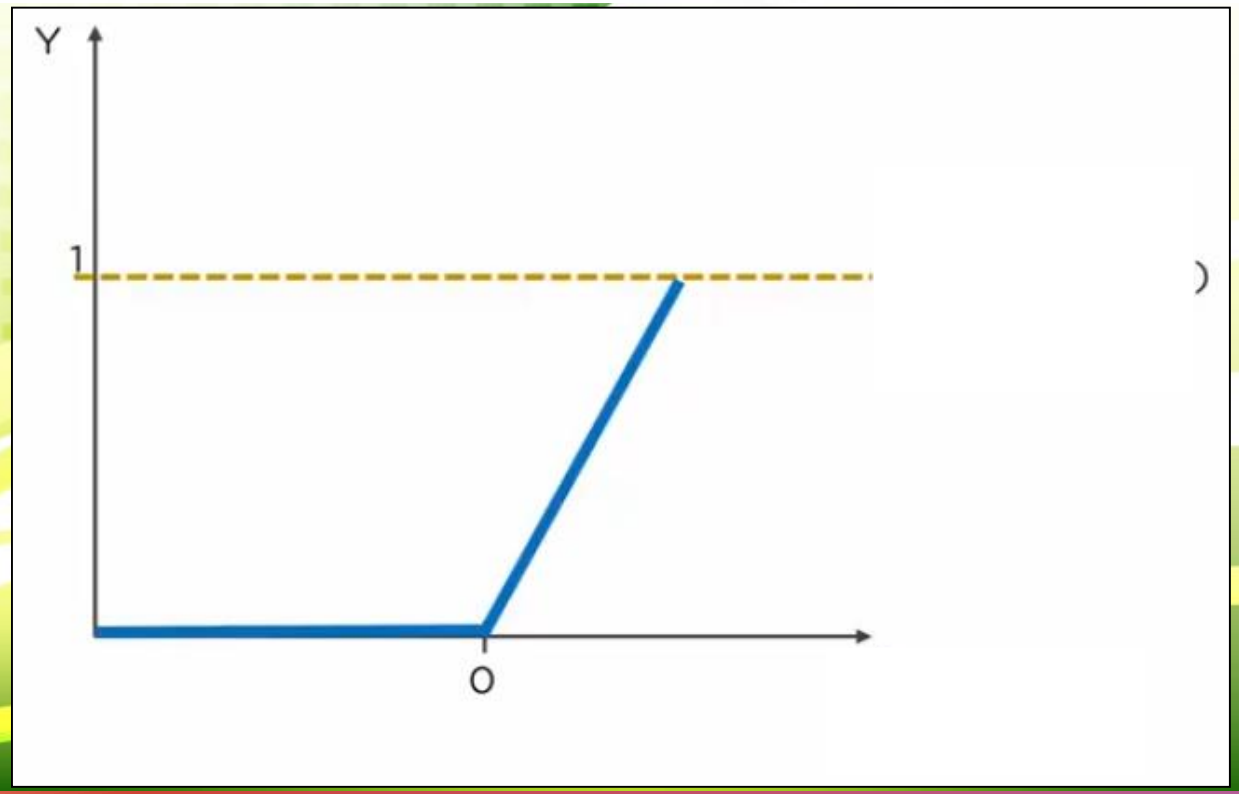
It is a threshold based activation function. If Y value is greater than a certain value, the function is activated and fired else not.



Deep Learning ___ ANN __ Activation function

ReLU function

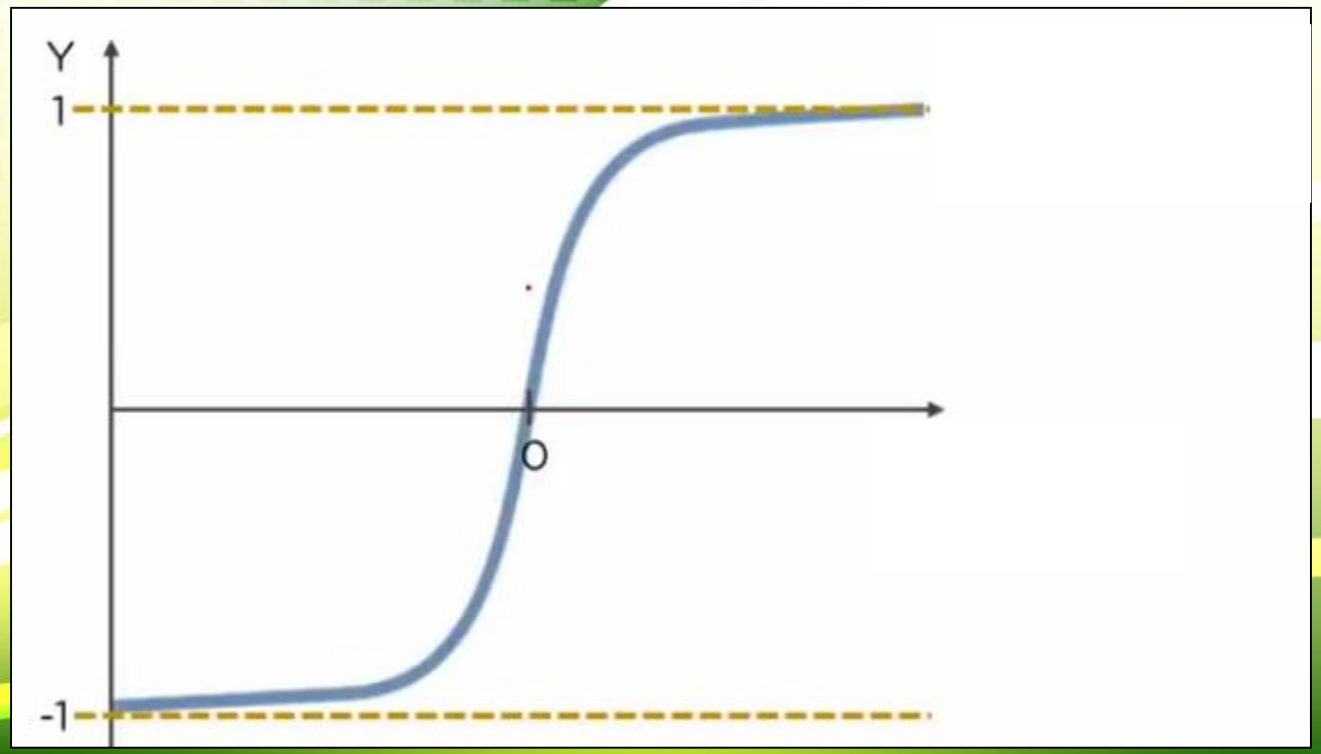
It is the most widely used Activation function and gives an output of X if X is positive and 0 otherwise



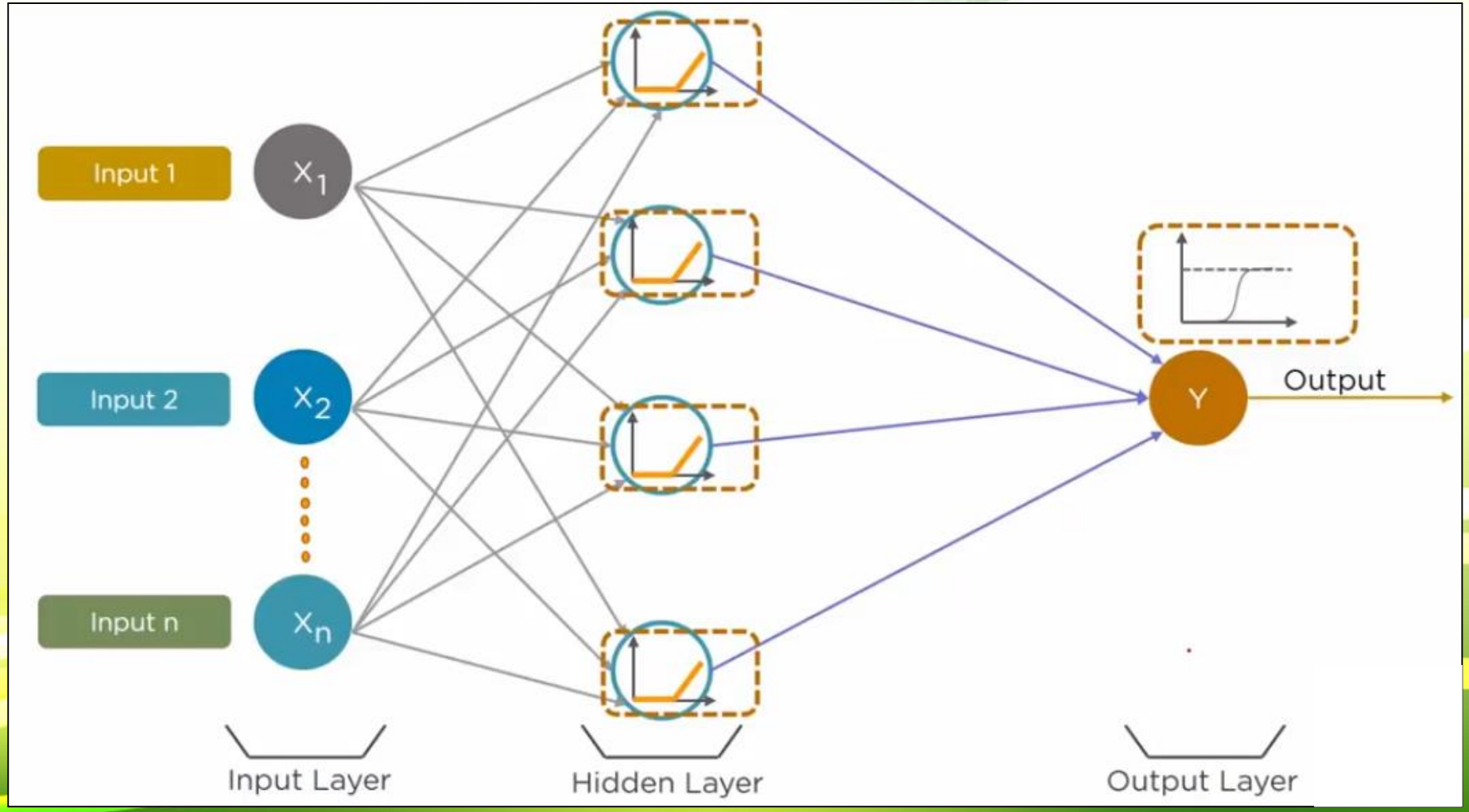
Deep Learning ___ ANN __ Activation function

Hyperbolic Tangent function

This function is similar to Sigmoid function and is bound to range (-1, 1)



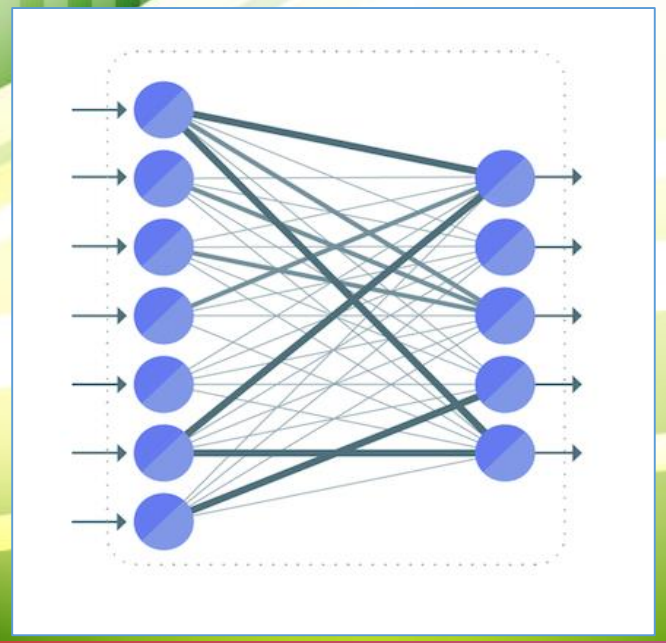
Deep Learning ___ ANN __ Activation function



Deep Learning __ ANN

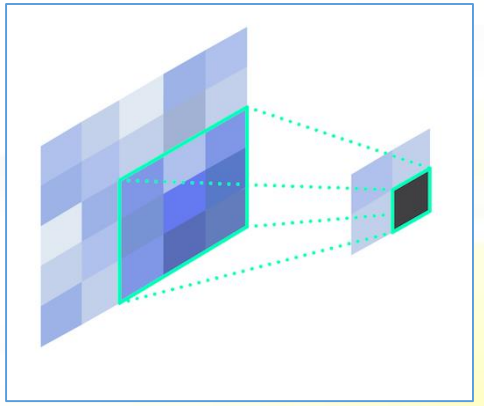
4 common types of NN layers: Fully connected, Convolution, Deconvolution, and Recurrent.

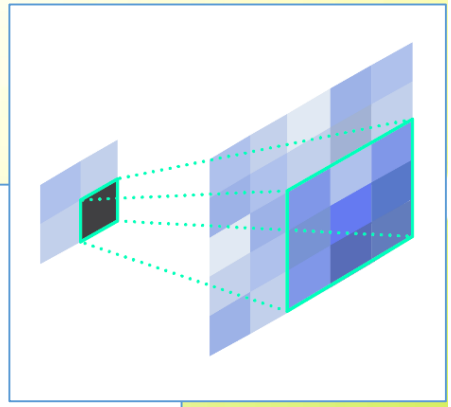
- **Fully connected layers** connect every neuron in one layer to every neuron in the next layer.
- Fully connected layers can become computationally expensive as their input grows, resulting in a combinatorial explosion of vector operations to be performed, and potentially poor scalability.

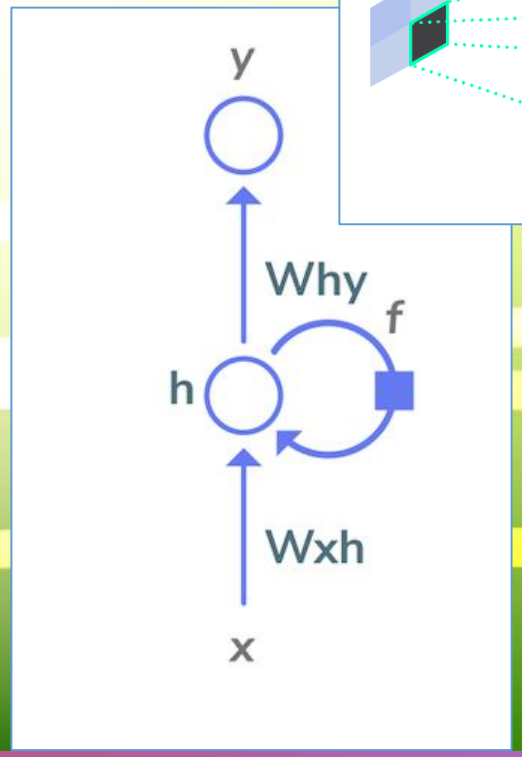


Deep Learning __ ANN

- Convolution Layer is an important type of layer in a CNN.
  - Its most common use is for detecting features in images, in which it uses a filter to scan an image, a few pixels at a time, and outputs a feature map that classifies each feature found.
- Deconvolution Layer is a transposed convolution process that effectively upsamples data to a higher resolution.
- Recurrent Layer includes a "looping" capability such that its input consists of both the data to analyze as well as the output from a previous calculation performed by that layer.
  - Recurrent layers form basis of recurrent neural networks (RNNs).





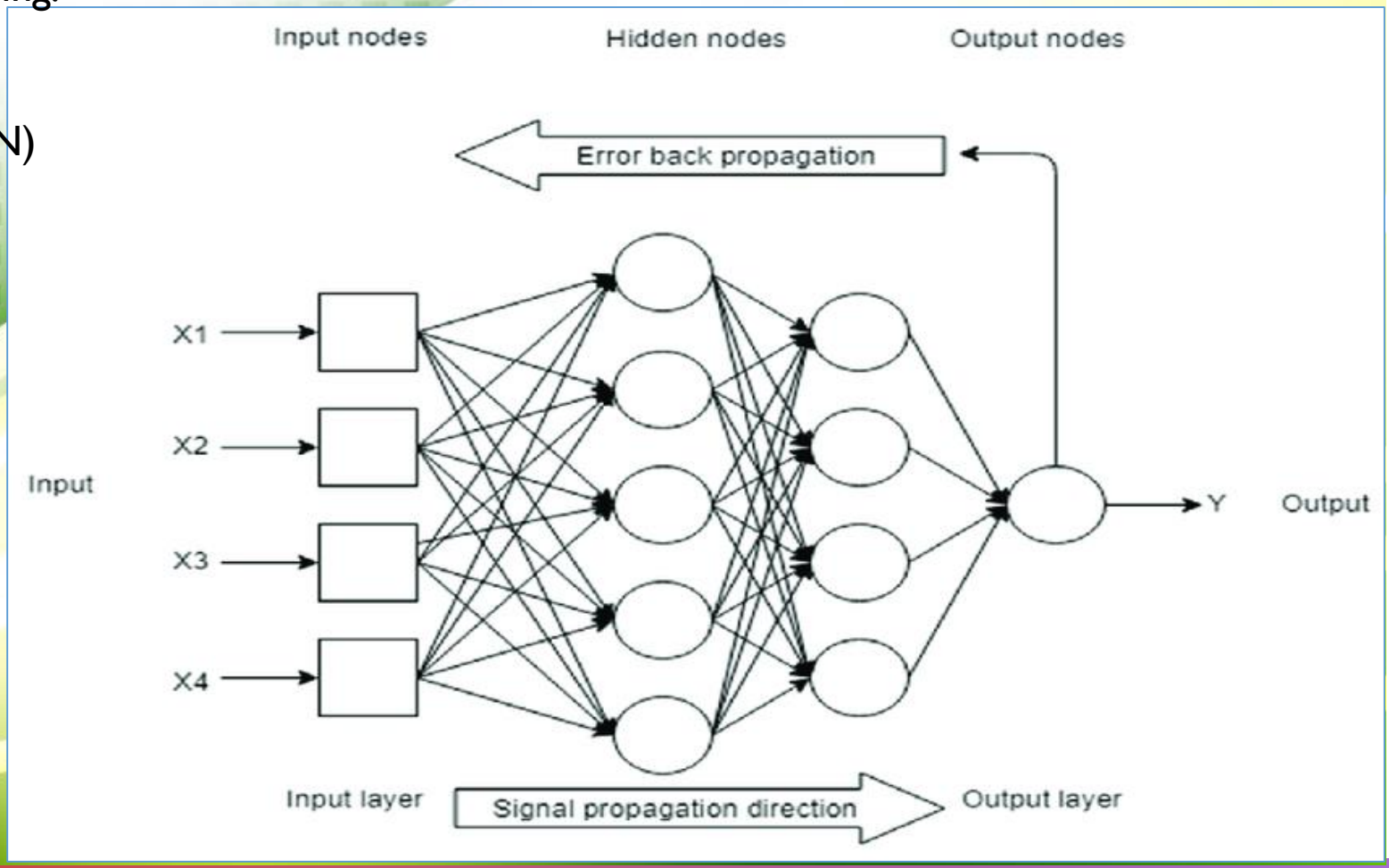


Deep Learning __ ANN

Types of Neural Networks in Deep Learning:

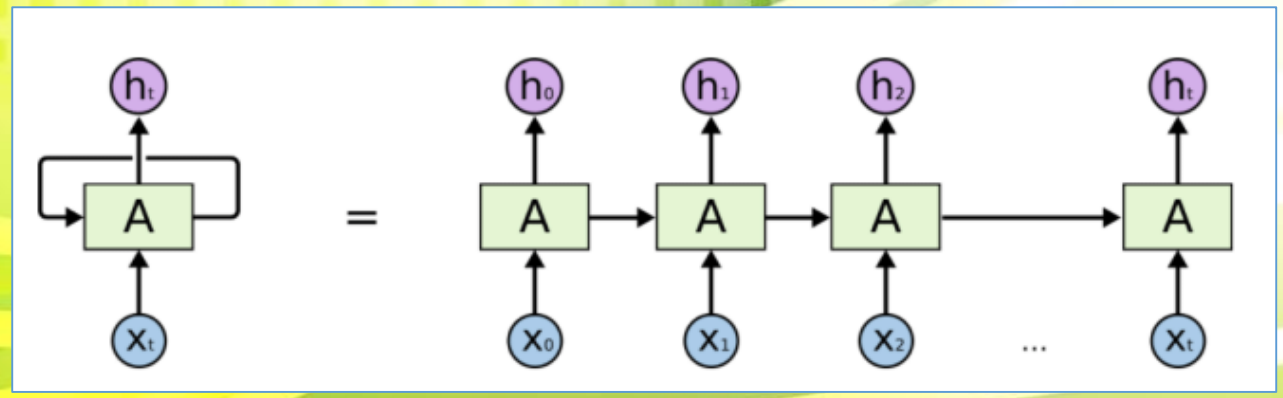
- Artificial Neural Networks (ANN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)

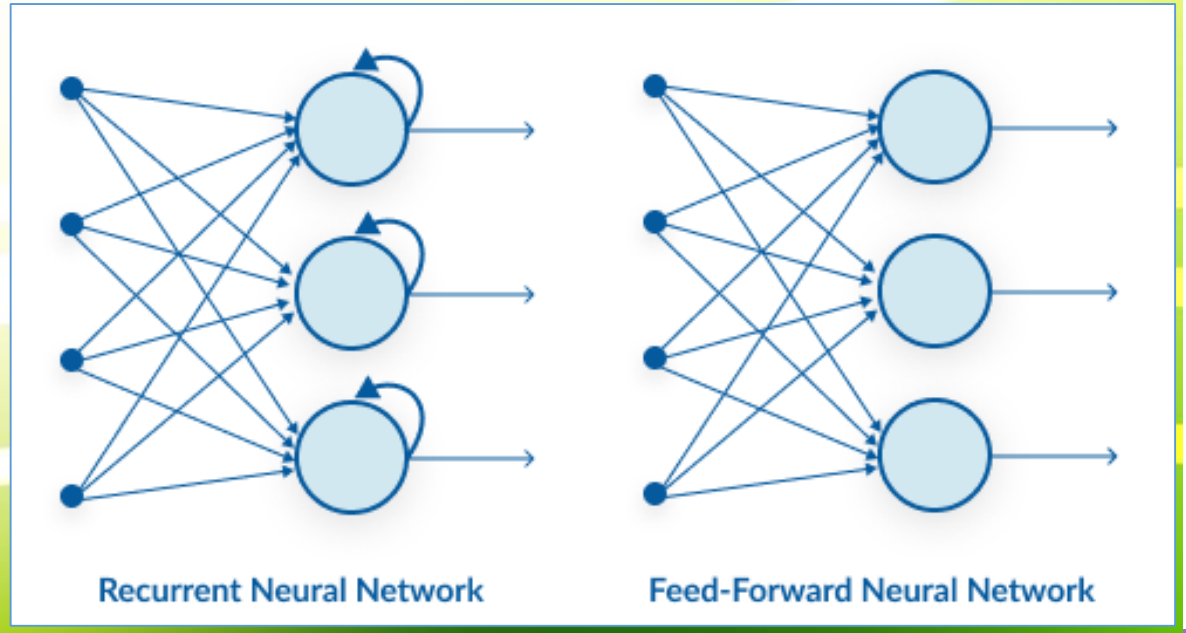
ANN is also known as a Feed-Forward Neural network because inputs are processed only in the forward direction.



Deep Learning __ RNN

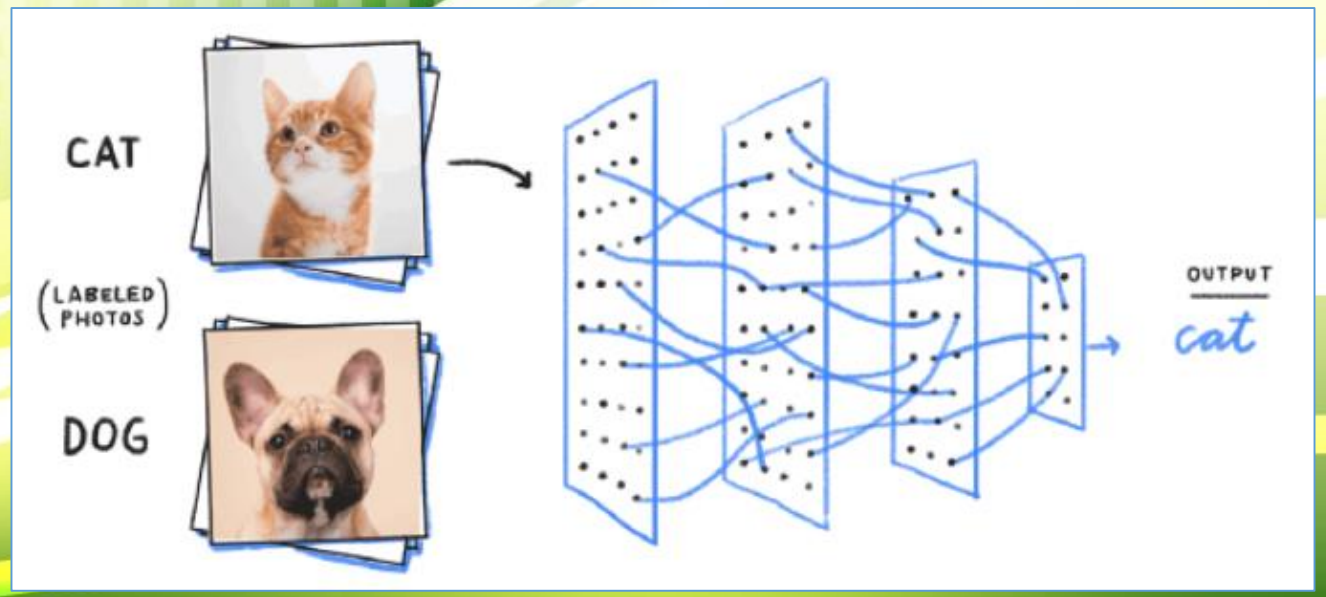
- RNN has a recurrent connection on hidden state. This looping constraint ensures that sequential information is captured in input data.
- RNN is widely used to solve the problems related to: Time Series data, Text data, Audio data, etc.
- RNN captures the sequential information present in input data i.e. dependency between words in the text while making predictions.



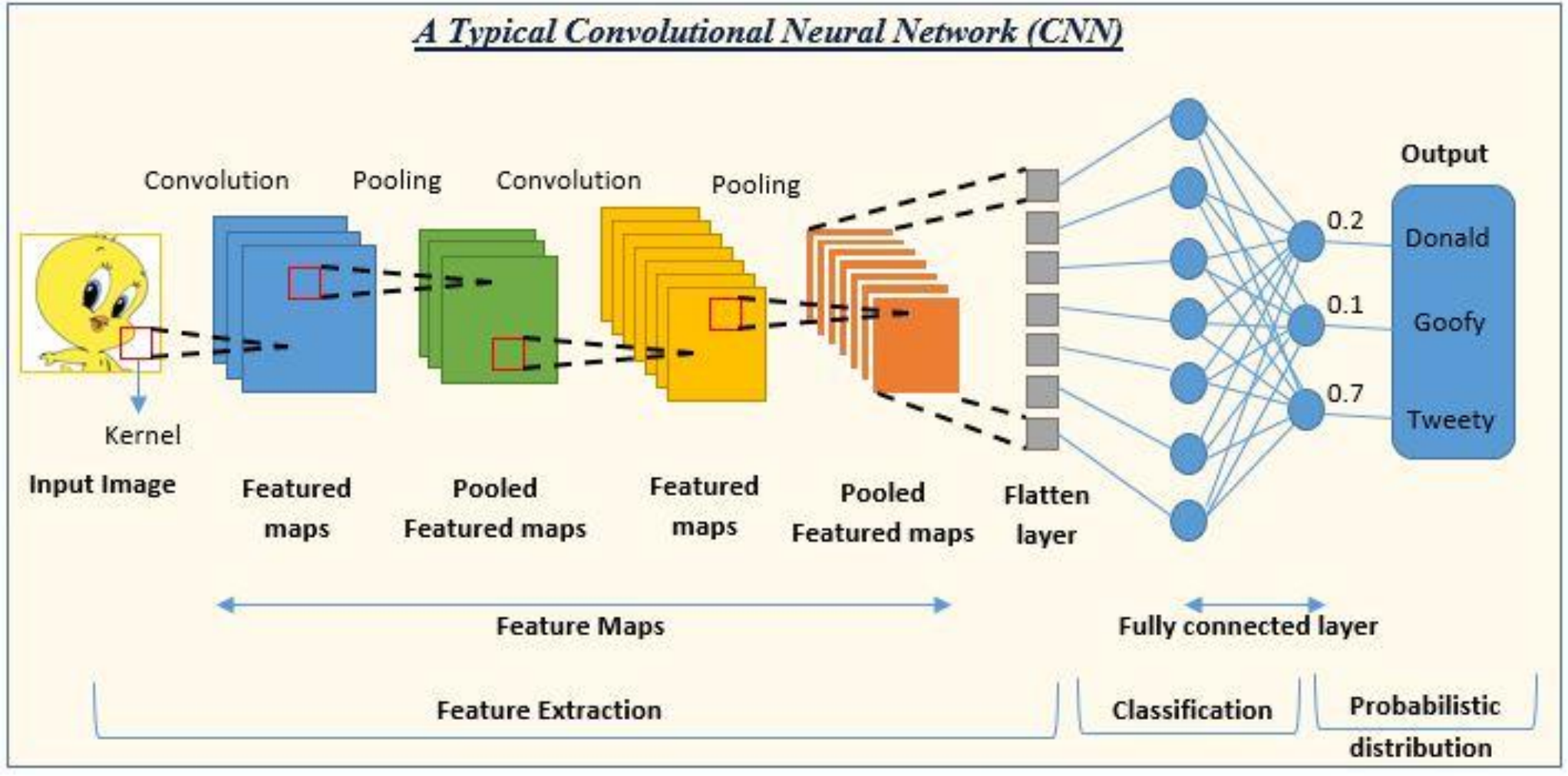


Deep Learning __ CNN

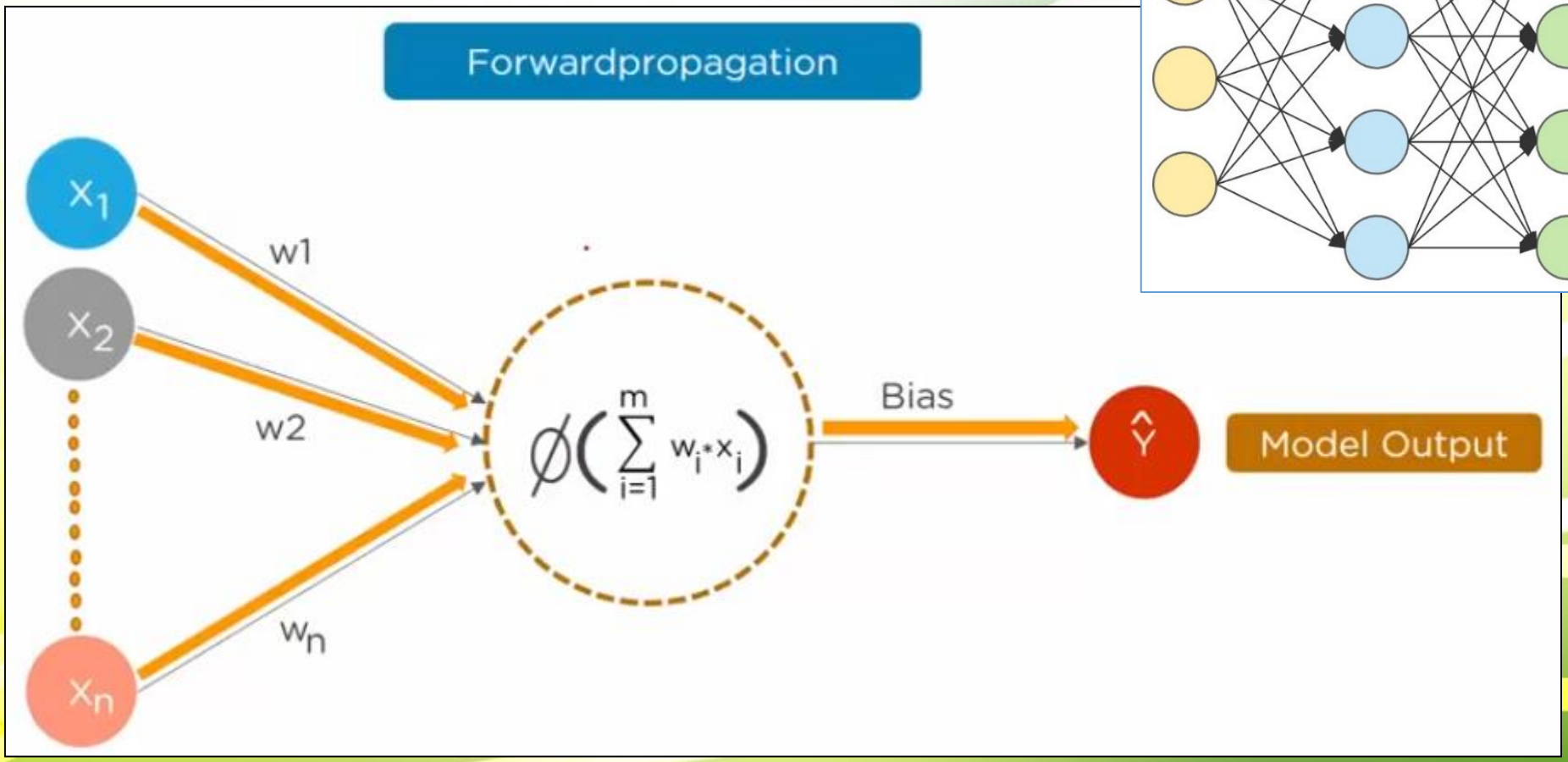
- Convolutional neural networks (CNN) models are being used across different applications and domains, especially in image and video processing projects.
- The building blocks of CNNs are filters (kernels).
- Kernels are used to extract relevant features from input using convolution operation.

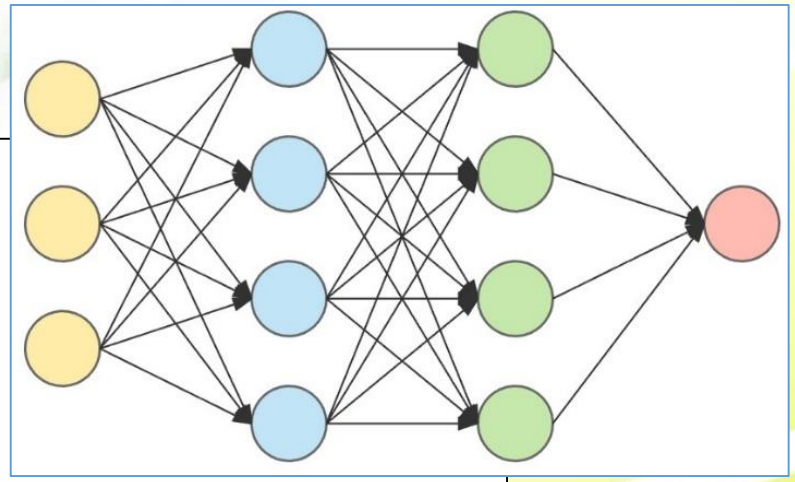


Deep Learning __ CNN

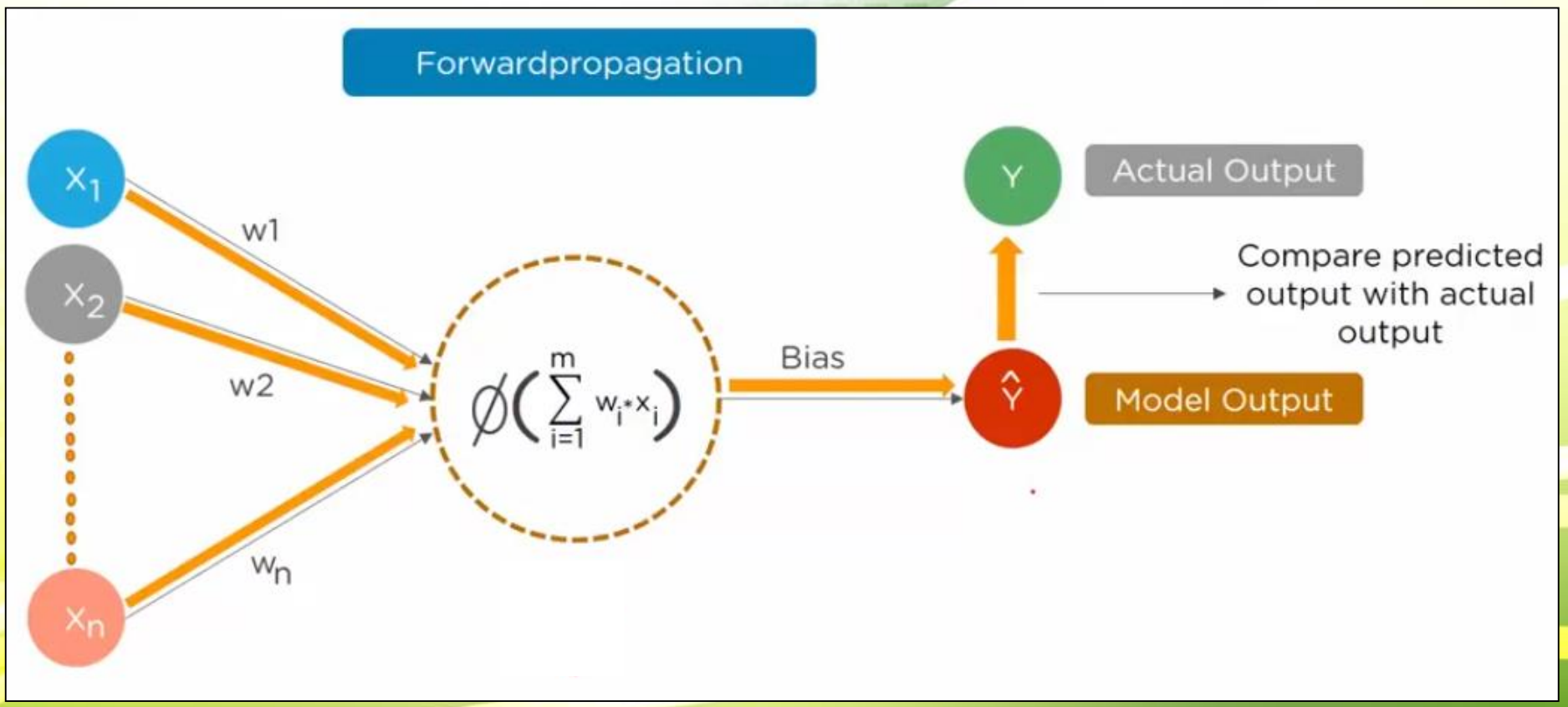


Deep Learning __ ANN



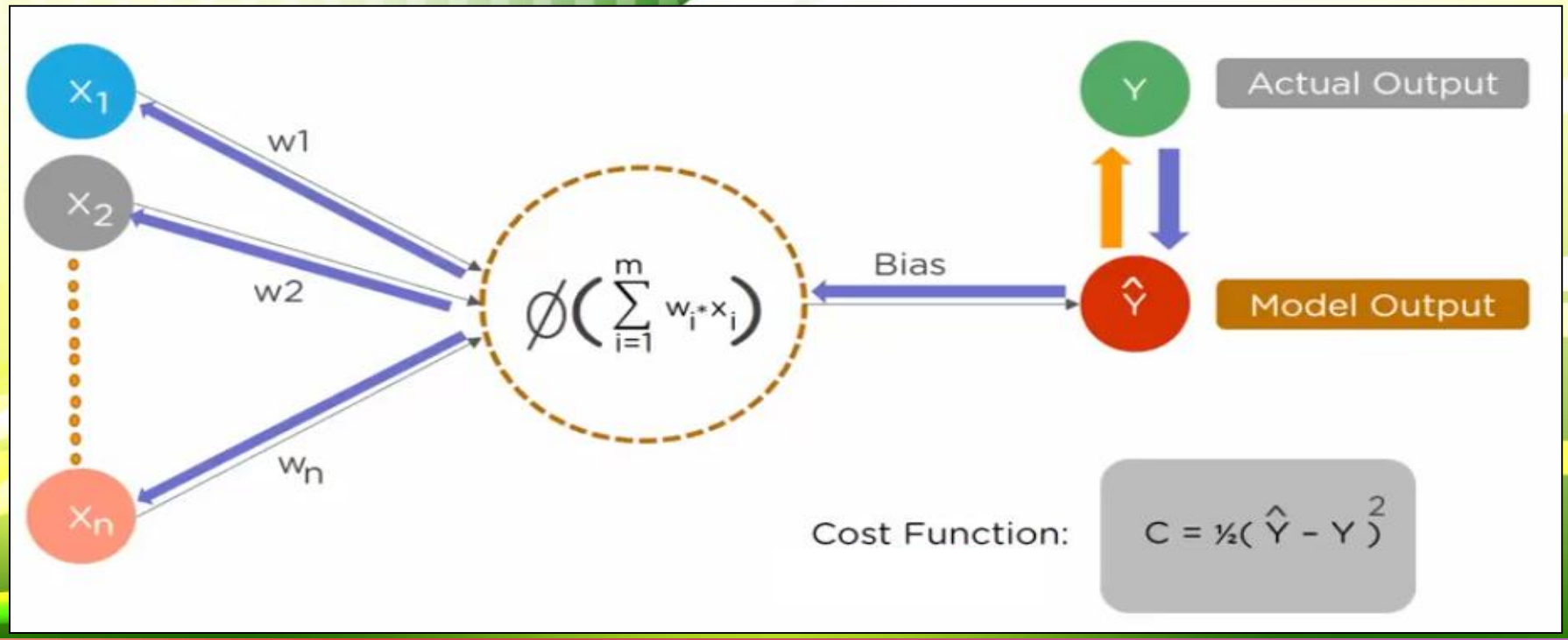


Deep Learning __ ANN



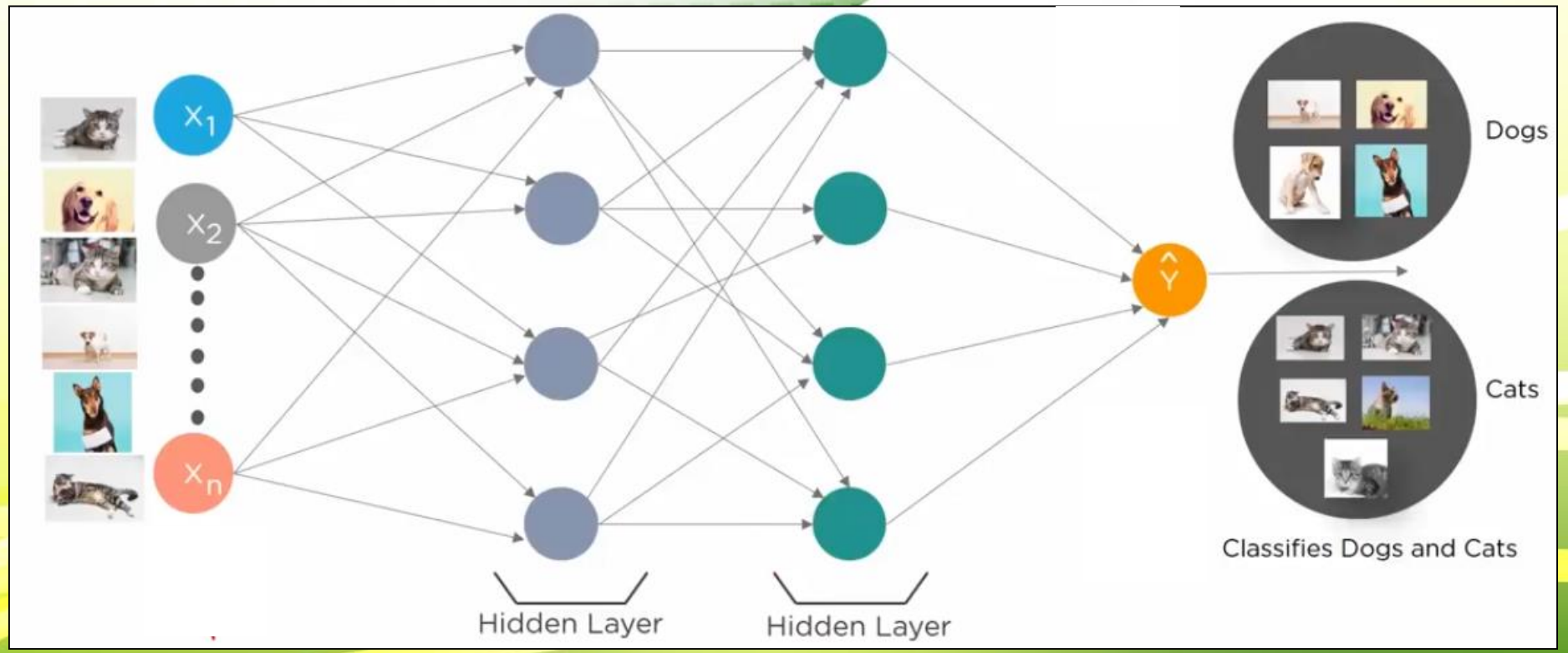
Deep Learning __ ANN

- After Training ANN, *BackPropagation* method helps to improve performance of network.
- *Cost function* helps to reduce the error rate.
- Cost is difference between ANN predicted output and actual output (for Training Dataset).
- Cost is reduced by adjusting weights & biases iteratively throughout the Training process.

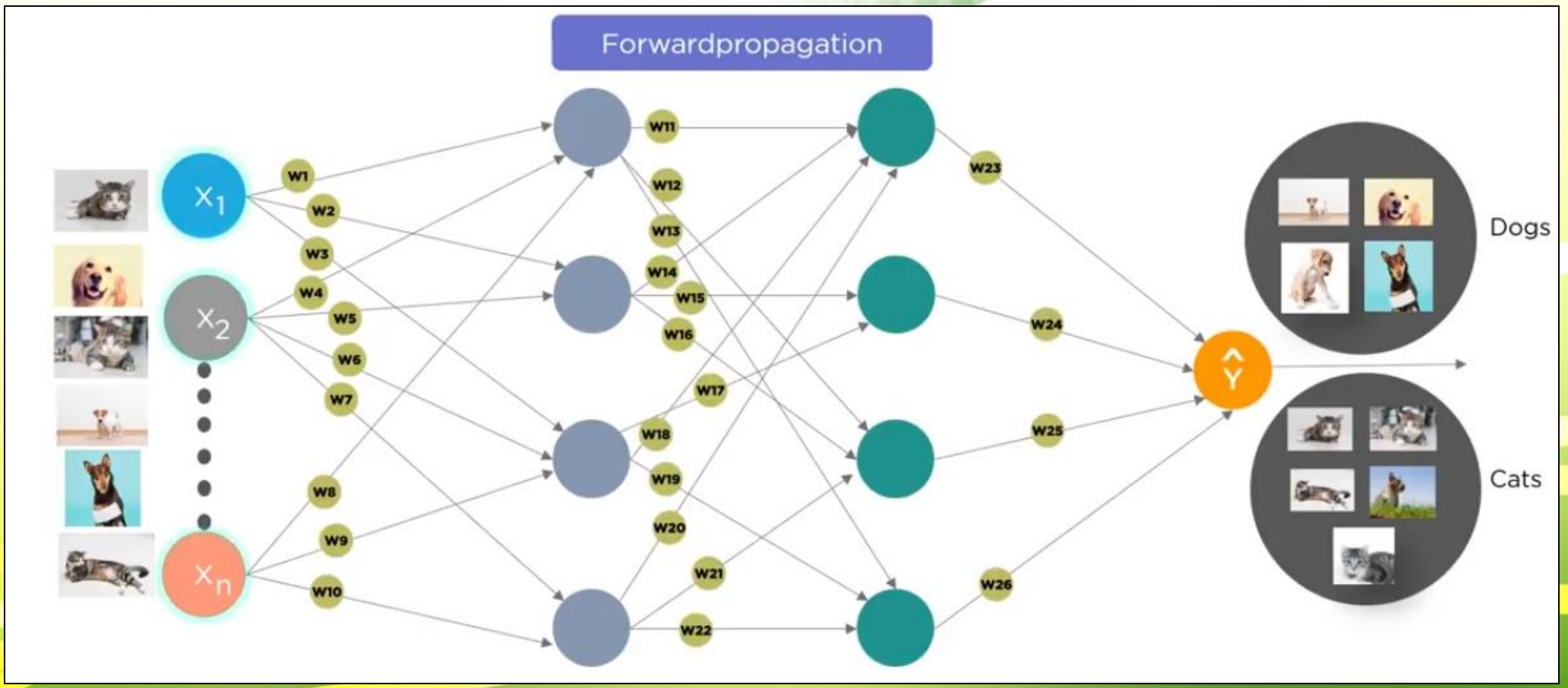


Deep Learning __ ANN __ How it works

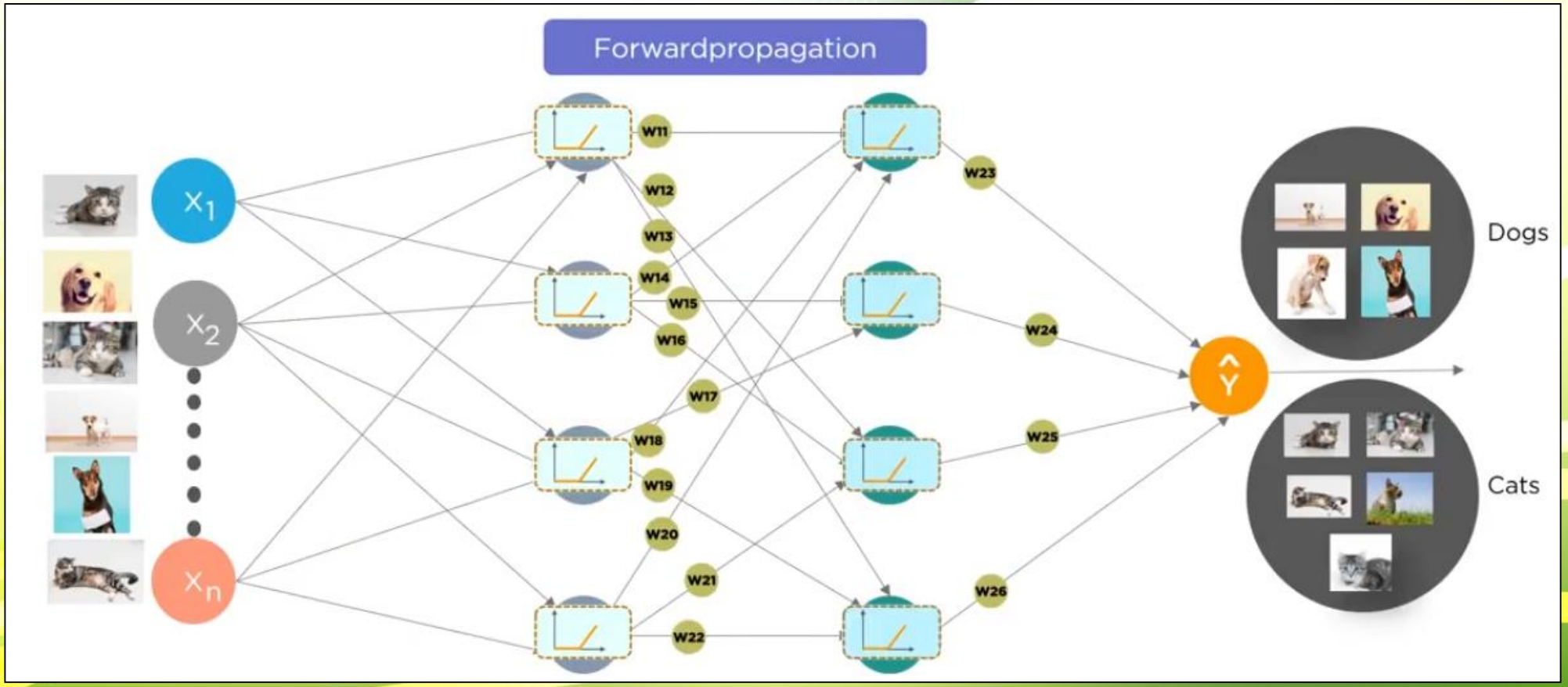
- Identify Dogs and Cats.



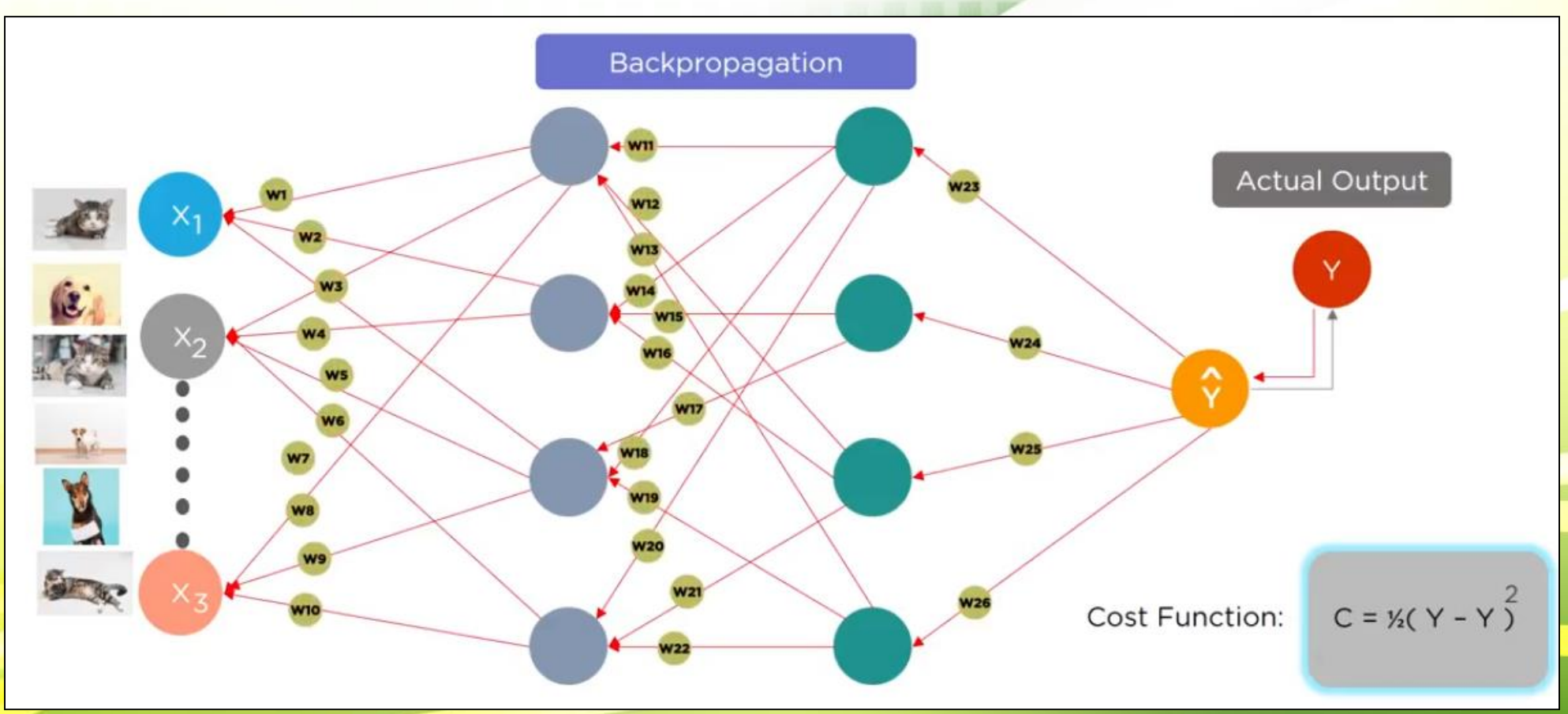
Deep Learning __ ANN



Deep Learning __ ANN



Deep Learning __ ANN



Hyperparameters

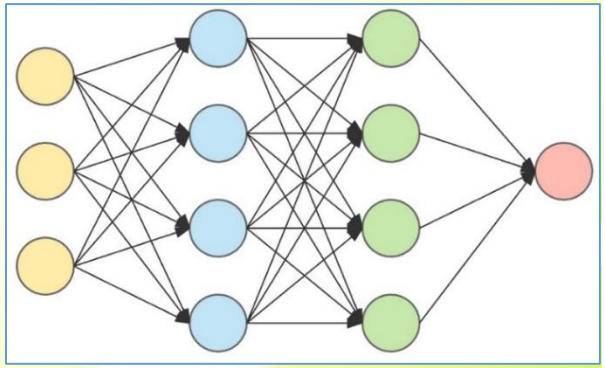
- Hyperparameters are used to improve model learning, and their values are set before starting the learning process of the model.
  - These parameters are explicitly defined by user to control the learning process.
  - These are external to the model, and their values cannot be changed during the training process.
- Some examples of Hyperparameters in Machine Learning
  - k in kNN or K-Nearest Neighbour algorithm
  - Train-test split ratio
  - Branches in Decision Tree
  - Number of clusters in Clustering Algorithm
  - Learning rate for training a neural network
  - Batch Size
  - Number of Epochs

Hyperparameters

- Broadly hyperparameters can be divided into two categories, which are given below:
  - Hyperparameter for Optimization:
    - Process of selecting the best hyperparameters to use is known as hyperparameter tuning/optimization.
    - *Example; Learning Rate, Batch size.*
  - Hyperparameter for Specific Models:
    - Hyperparameters that are involved in the structure of model are known as hyperparameters for specific models.
    - *Example; Number of Hidden Units, number of layers.*

Hyperparameters

- **Number of Layers**: A neural network is made up of vertically arranged components, called layers (input layers, hidden layers, and output layers).
  - A higher-layered neural network gives a better performance than a lower-layered network.
  - Model complexity may increase with higher-layered NN.
  - For some specific Neural network, a lower number of layers make a better model.



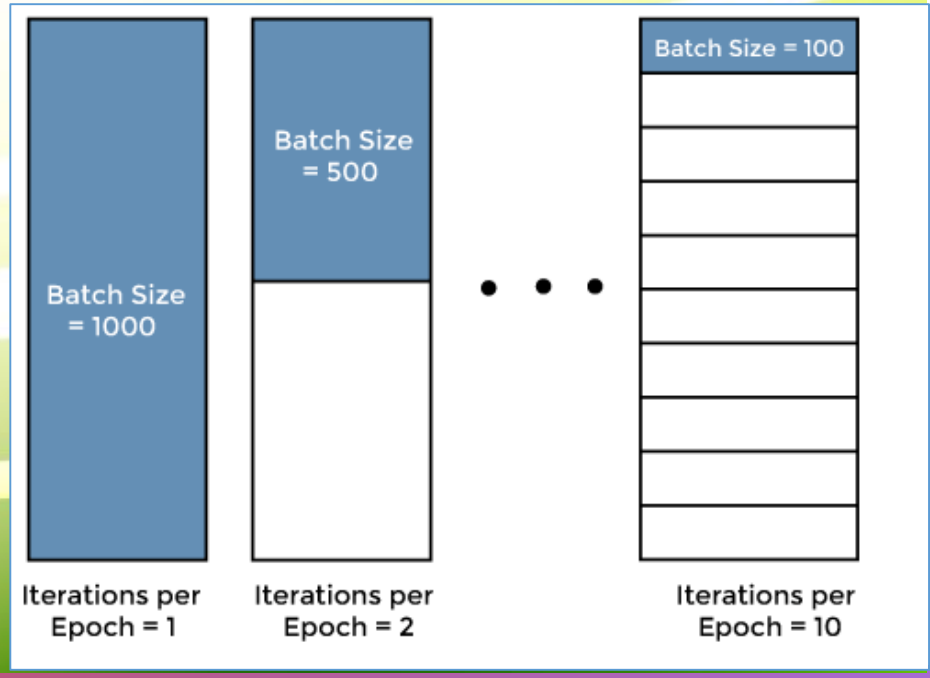
- **Number of Hidden Units**: Hidden units refer to components comprising layers of processors between input and output units in a neural network.
  - It is important to specify the number of hidden units hyperparameter for neural network.
  - It should be between the size of the input layer and the size of the output layer.
  - More specifically, the number of hidden units should be 2/3 of the size of the input layer, plus the size of the output layer.

Hyperparameters

- **Learning Rate**: Controls how much model needs to change in response to estimated error for each time when model's weights are updated.
  - Selecting the optimized learning rate is a challenging task.
  - Low learning rate may slow down training process; and high learning rate may not optimize model properly.
- **Batch Size**: To enhance speed of learning process, training set is divided into different subsets (batch).
- **Number of Epochs**: An epoch can be defined as the complete cycle for training the machine learning model.
  - Epoch represents an iterative learning process.
  - Number of epochs varies from model to model, and most models are created with more than one epoch.

Hyperparameters - EPOCH

- **Epoch** refers to one complete pass of the training dataset through the algorithm.
  - It takes a few epochs while training a machine learning model.
  - It will face an issue while feeding a bunch of training data in the model; due to limitations of computer storage.
    - To overcome this issue, training data is broken into small batches according to the computer memory or storage capacity.
    - *When all batches are fed exactly once to train the model, then this entire procedure is known as Epoch in Machine Learning.*
- *Example;*
  - *Total number of training examples = 3000;*
  - *Assume each batch size = 500;*
  - *Total number of Iterations = Total number of training examples/Individual batch size*
  - *Total number of iterations = 3000/500 = 6;*
  - ***1 Epoch = 6 Iterations.***



Hyperparameters - EPOCH

- Model with a single epoch may lead to overfitting in the model.
- Training a model typically requires multiple numbers of Epochs.
- Better generalization can be achieved with new inputs by using more epochs in the training of machine learning model.
- Given the complexity and variety of data in real-world applications, hundreds to thousands of epochs may be required to achieve reasonable test data correctness.
    - To determine the right number of epochs, a validation error is taken into account.
    - The number of epochs is increased until there is a reduction in a validation error.
    - If there is no improvement in reduction error for the consecutive epochs, then it indicates to stop increasing the number of epochs.

Deep Learning __ ANN