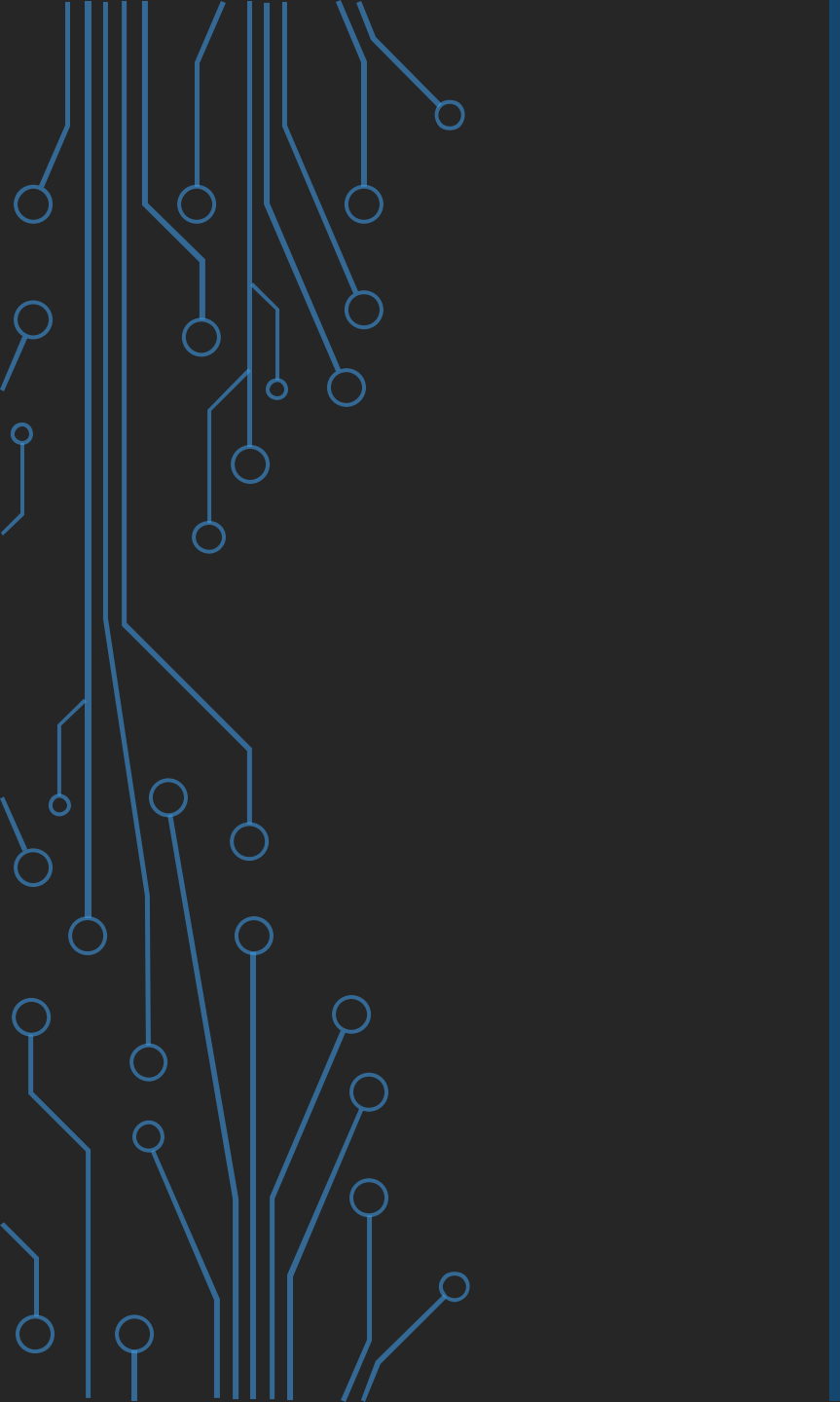
ADVANCED MACHINE LEARNING & INTRODUCTION TO DEEP LEARNING

Machine Learning – Yousef Elbaroudy

GUIDELINES

- Try to focus on the important information mentioned through the session
- Apply what you take on the practical section
- Do not try to memorize everything you got, just learn
- Don’t mind to ask about anything you want to know

Enjoy the Session ☺



ADVANCED MACHINE LEARNING

ADVANCED MACHINE LEARNING

- **Advanced machine learning** refers to the utilization of more sophisticated and complex techniques, algorithms, and models to tackle intricate problems and extract insights from data.

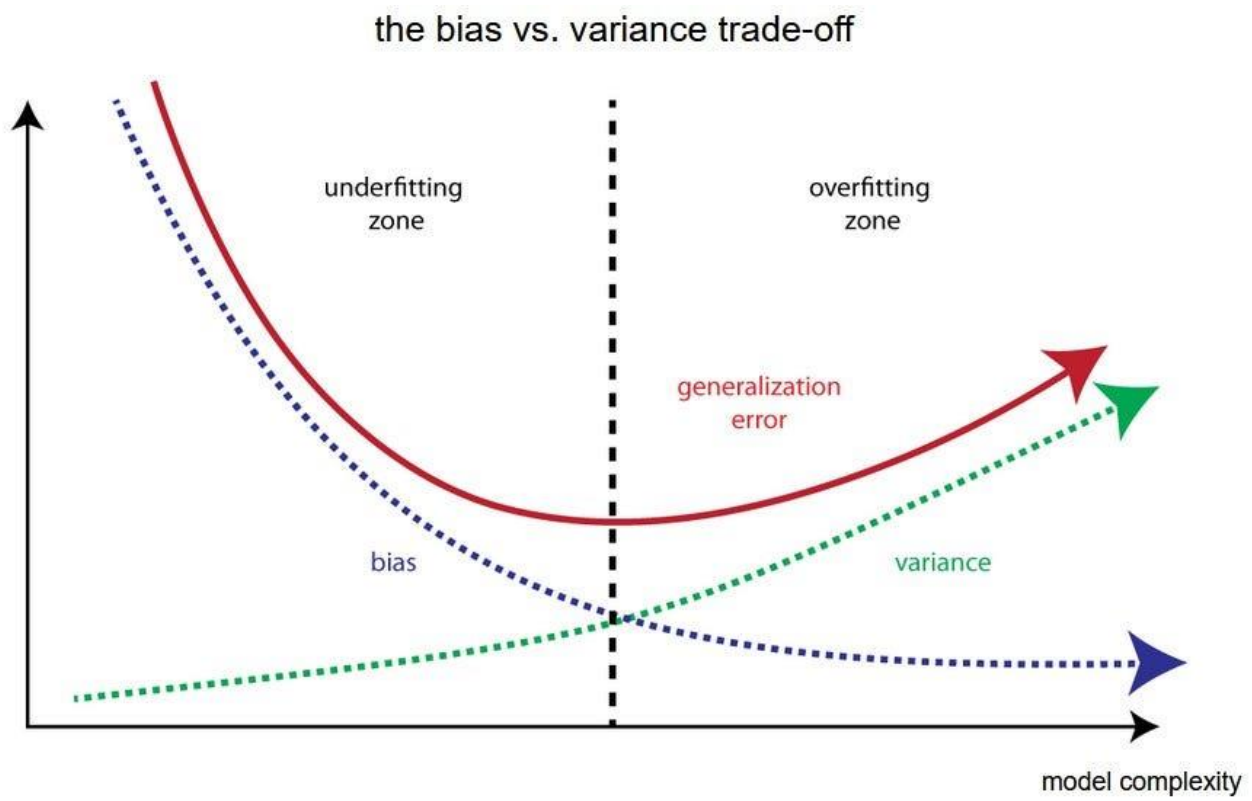
- It builds upon the foundation of traditional machine learning methods and delves into more intricate and specialized areas of study.

SOME CHARACTERISTICS

- Complex Algorithms
- Deep Learning
- Unsupervised Learning
- Natural Language Processing (NLP) & Computer Vision
- Reinforcement Learning
- Transfer Learning
- Hyperparameter Tuning

MODEL GENERALIZATION

BIAS – VARIANCE TRADEOFF



WHAT IS BIAS ?

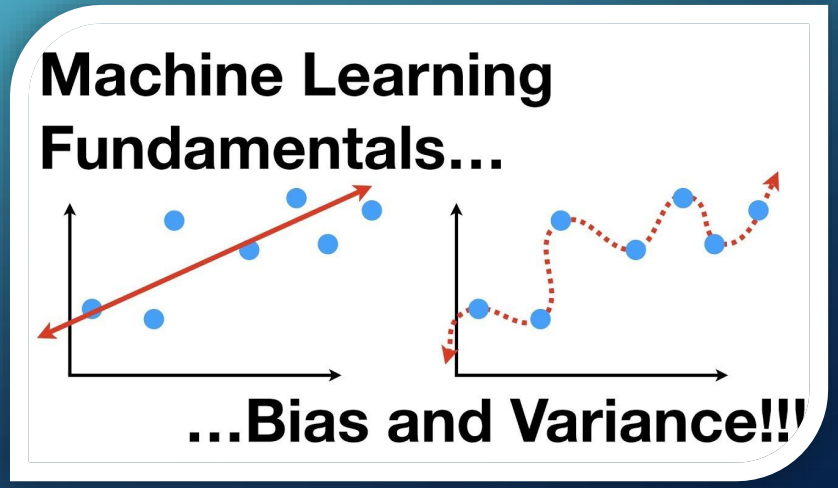
- **Bias** refers to a systematic and consistent deviation from the true value or an accurate representation of a particular concept, situation, data, or population. In various contexts, bias can introduce inaccuracies, unfairness, or errors in decision-making, analysis, or interpretation.

- In other words, The error caused by simplifying assumptions in the model, which causes certain test instances to have consistent errors across different choices of training sets.

BIAS

- Even if the model has access to an infinite training data, **the bias** cannot be removed.

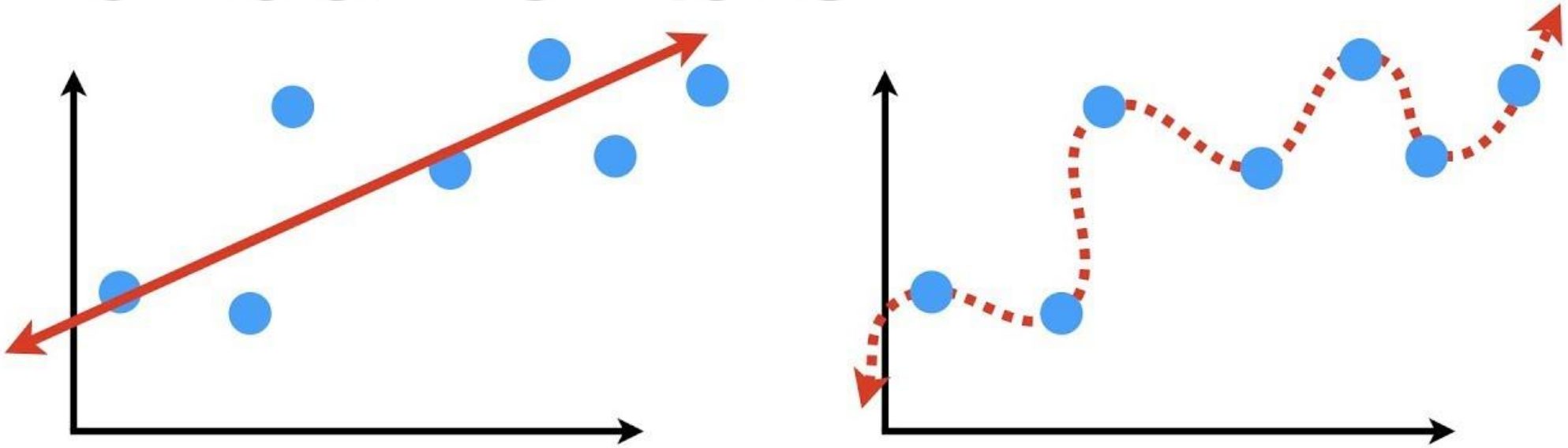
- The linear model has a higher bias than the polynomial model, as it can never fit the data distribution exactly (regardless of data size).



VARIANCE

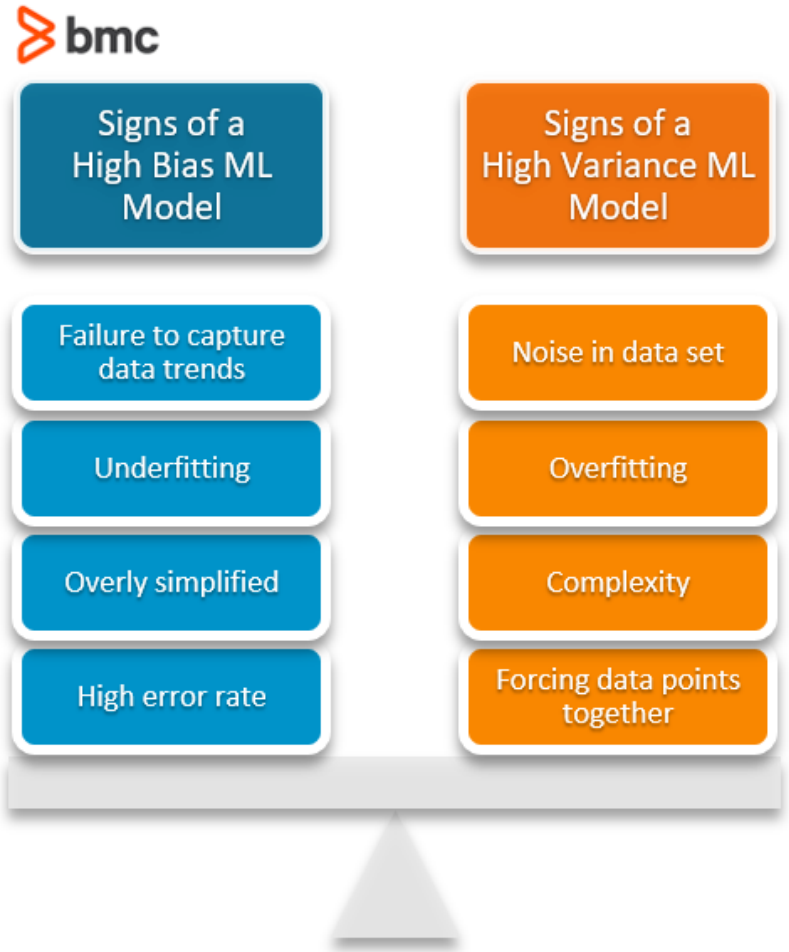
- **Variance** is caused by the inability to learn correctly all the model parameters, especially when **THE DATA IS LIMITED** and the model has many parameters.
- In machine learning, high variance can be a sign of **overfitting**. Overfitting occurs when a model learns the noise in the training data rather than the underlying patterns, resulting in poor generalization to new data.
- Therefore, if different training sets are used, different predictions will be provided for the same test instance.

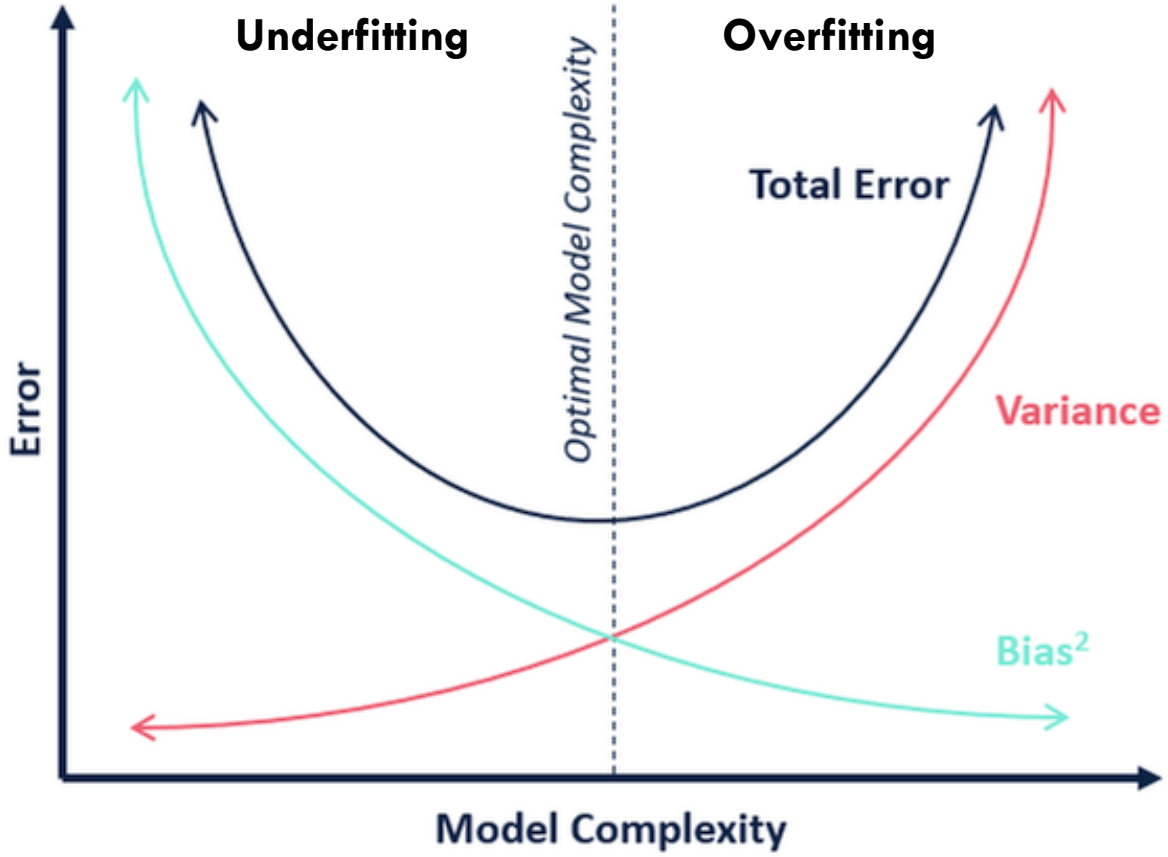
Machine Learning Fundamentals…



…Bias and Variance!!!

SIGNS OF HIGH BIAS OR HIGH VARIANCE





HOW TO DETECT UNDERFITTING OR OVERFITTING ?

1. **More Complex Model**: If your current model is too simple and doesn't fit the data well, consider using a more complex model that can learn more intricate relationships. However, be cautious not to make the model overly complex, as it could lead to overfitting.
2. **Feature Engineering**: Carefully select or create relevant features that help the model capture important patterns in the data. Domain knowledge and understanding of the problem can guide you in creating informative features.
3. **Increase Model Capacity**: For machine learning models like neural networks, increasing the number of hidden layers and neurons can increase the model's capacity to learn complex patterns. However, this should be balanced with regularization techniques to avoid overfitting.
4. **Decrease Regularization**: If you're using regularization techniques like L1 or L2 regularization, consider reducing the strength of the regularization term. This can allow the model to fit the training data more closely.

HOW TO PREVENT UNDERFITTING ?

5. **Hyperparameter Tuning**: Experiment with different hyperparameters to find the right balance between model complexity and generalization. This includes parameters like learning rate, batch size, and regularization strength.
6. **Ensemble Methods**: Combine predictions from multiple models to improve performance. Ensemble methods like bagging and boosting can help mitigate underfitting by leveraging the strengths of multiple models.
7. **Data Augmentation**: In certain cases, data augmentation techniques can help increase the variety of training examples and improve the model's ability to learn.
8. **More Training Iterations**: For iterative training algorithms, such as gradient descent in neural networks, increasing the number of training iterations can help the model learn more complex relationships.
9. **Better Initialization**: Properly initializing the model's weights can help speed up convergence and prevent getting stuck in suboptimal solutions.
10. **Fine-Tuning Pre-trained Models**: If using pre-trained models (transfer learning), consider fine-tuning them on your specific task. This can help leverage the model's existing knowledge while adapting it to your data.
11. **Check Data Quality**: Ensure that your training data is clean and representative of the problem you're trying to solve. Low-quality or skewed data can lead to underfitting.
12. **Change Algorithm**: If your current algorithm is not performing well, consider trying different algorithms that might be better suited to your problem.

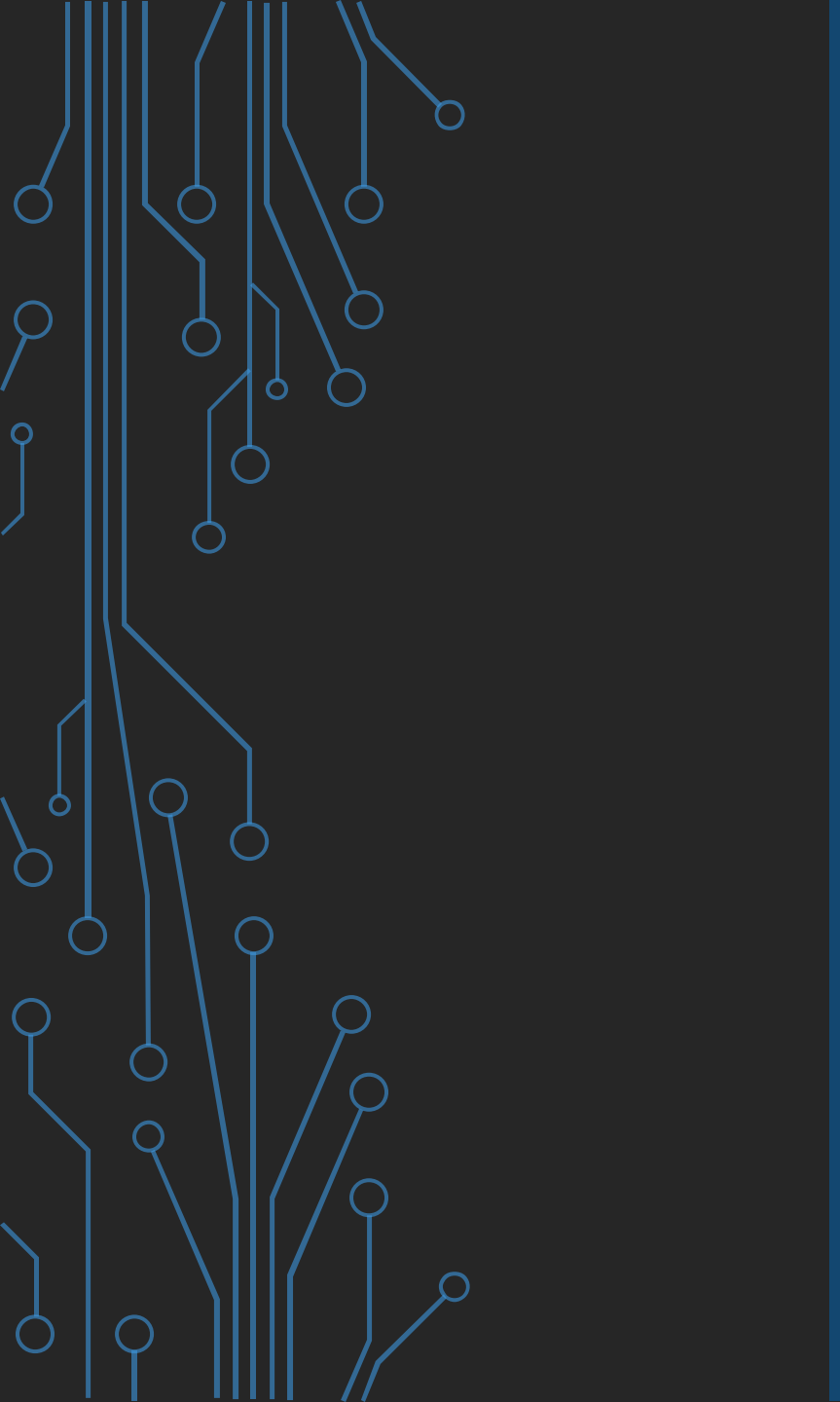
HOW TO PREVENT UNDERFITTING ? (CONT.)

1. **More Data**: Increasing the size of your training dataset can help the model generalize better. More diverse data can help the model learn the underlying patterns rather than focusing on noise.
2. **Cross-Validation**: Use techniques like k-fold cross-validation to assess the model's performance on multiple subsets of the data. This helps you understand how well the model generalizes and whether overfitting is occurring.
3. **Feature Selection**: Choose relevant and informative features. Avoid adding too many features, as this can lead to overfitting. Feature selection techniques like mutual information or regularization can help identify important features.
4. **Regularization**: Introduce regularization terms into the model's objective function. L1 regularization (Lasso) and L2 regularization (Ridge) are common techniques that penalize large coefficients, encouraging the model to focus on the most important features.
5. **Simpler Models**: Choose simpler model architectures or algorithms that have fewer parameters. Complex models are more prone to overfitting because they can fit the noise in the data.

HOW TO PREVENT OVERFITTING ?

6. **Early Stopping**: Monitor the model's performance on a validation set during training. Stop training when the validation performance starts to degrade, indicating that the model is beginning to overfit.
7. **Ensemble Methods**: Combine predictions from multiple models to improve generalization. Techniques like bagging (Bootstrap Aggregating) and boosting create an ensemble of models that collectively perform better than individual models.
8. **Dropout**: In neural networks, dropout is a regularization technique where randomly selected neurons are ignored during training. This helps prevent the network from relying too heavily on specific neurons.
9. **Data Augmentation**: Increase the effective size of your training data by applying various transformations like rotation, flipping, cropping, and adding noise. Data augmentation can help the model learn more robust features.
10. **Hyperparameter Tuning**: Experiment with different hyperparameters like learning rate, batch size, and regularization strength. Tuning these hyperparameters can help strike a balance between model complexity and generalization.
11. **Remove Outliers**: Outliers can have a disproportionate impact on the model, leading to overfitting. Consider removing or transforming outliers in your data.
12. **Validation Set**: Split your data into training, validation, and test sets. Use the validation set to tune hyperparameters and monitor for overfitting, while keeping the test set completely separate until final evaluation.

HOW TO PREVENT OVERFITTING ? (CONT.)



CROSS-VALIDATION

CROSS-VALIDATION

- Cross-validation is a statistical technique used to assess the performance of a machine learning model and to evaluate its generalization capabilities.
- It involves dividing the dataset into multiple subsets or "folds" to train and test the model multiple times, providing a more reliable estimate of the model's performance compared to a single train-test split.
- **The primary goal** of cross-validation is to understand how well a model will perform on unseen data, helping to identify potential issues such as overfitting or underfitting.

K-FOLD TECHNIQUE

1. **K-Fold Cross-Validation**:
   - The dataset is divided into k equally sized folds (partitions).
   - The model is trained on k-1 folds and tested on the remaining fold.
   - This process is repeated k times, each time using a different fold as the test set.
   - The final performance metric is the average of the performance scores from all k iterations.

K-FOLD TECHNIQUE (CONT.)

- **K-fold** cross-validation provides a more comprehensive assessment of the model's performance by using different subsets of data for training and testing.
- It helps in reducing the dependency on a particular train-test split, which might inadvertently result in overly optimistic or pessimistic performance estimates.

HYPERPARAMETERS TUNING

HYPERPARAMETERS TUNING

- **Hyperparameter** tuning, also known as hyperparameter optimization or hyperparameter search, refers to the process of finding the optimal set of hyperparameters for a machine learning model.
- **Hyperparameters** are parameters that are set before the learning process begins and are not learned from the data during training.
- They determine the behavior and characteristics of the model during training and can significantly impact its performance and generalization.

HYPERPARAMETERS TUNING FREQUENT STEPS

- Selection of Hyperparameters
- Choose a search space
- Search Method
- Iterative process with cross-validation
- Automated Tools

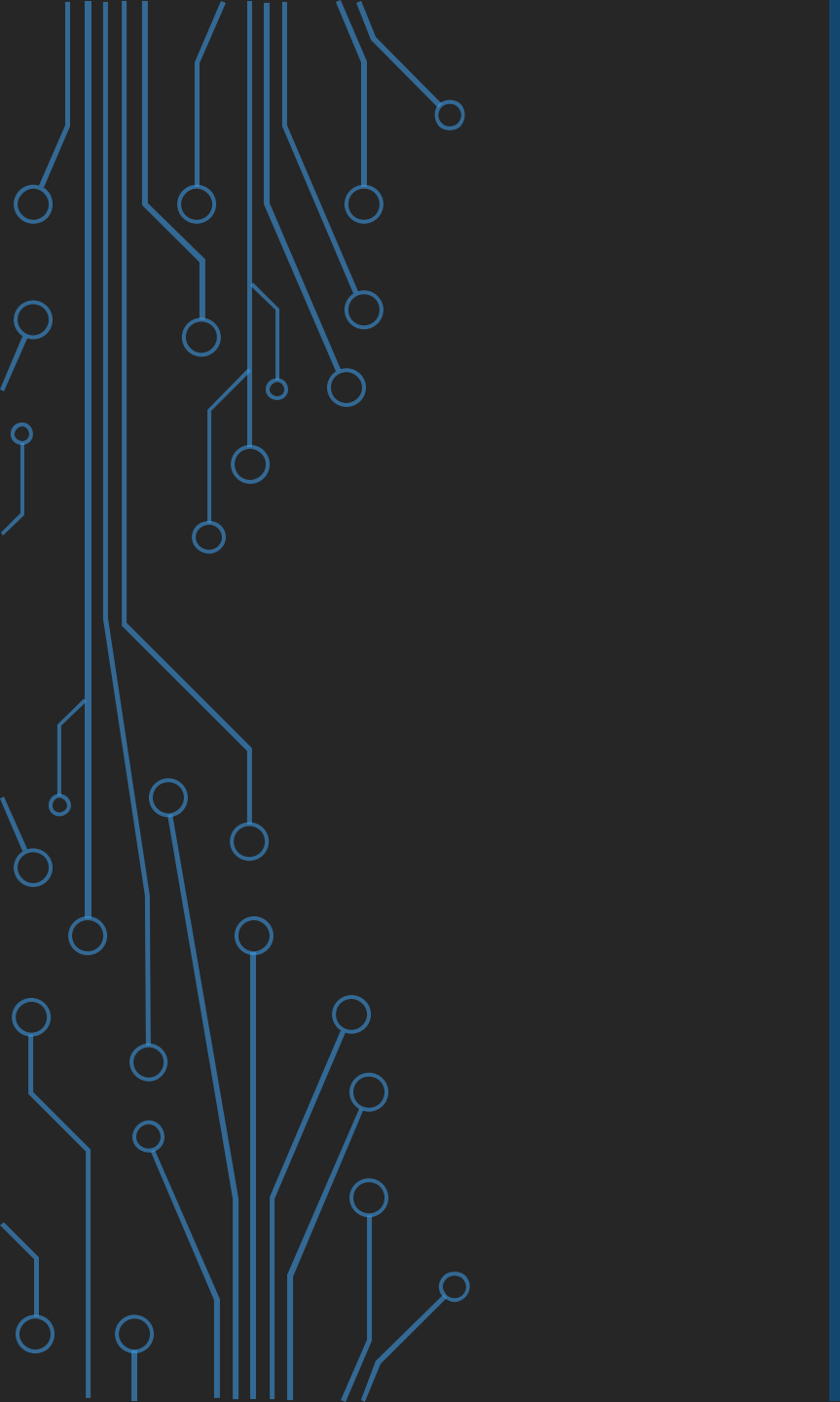
AUTOMATED HYPERPARAMETER TUNING TOOLS

- There are several libraries and tools that automate the hyperparameter tuning process.

- Popular examples include ***GridSearchCV*** and ***RandomizedSearchCV*** in scikit-learn, and platforms like Keras Tuner and Optuna.

TIME FOR PRACTICALITY (15 MINUTES)

BREAK (10 MINUTES)

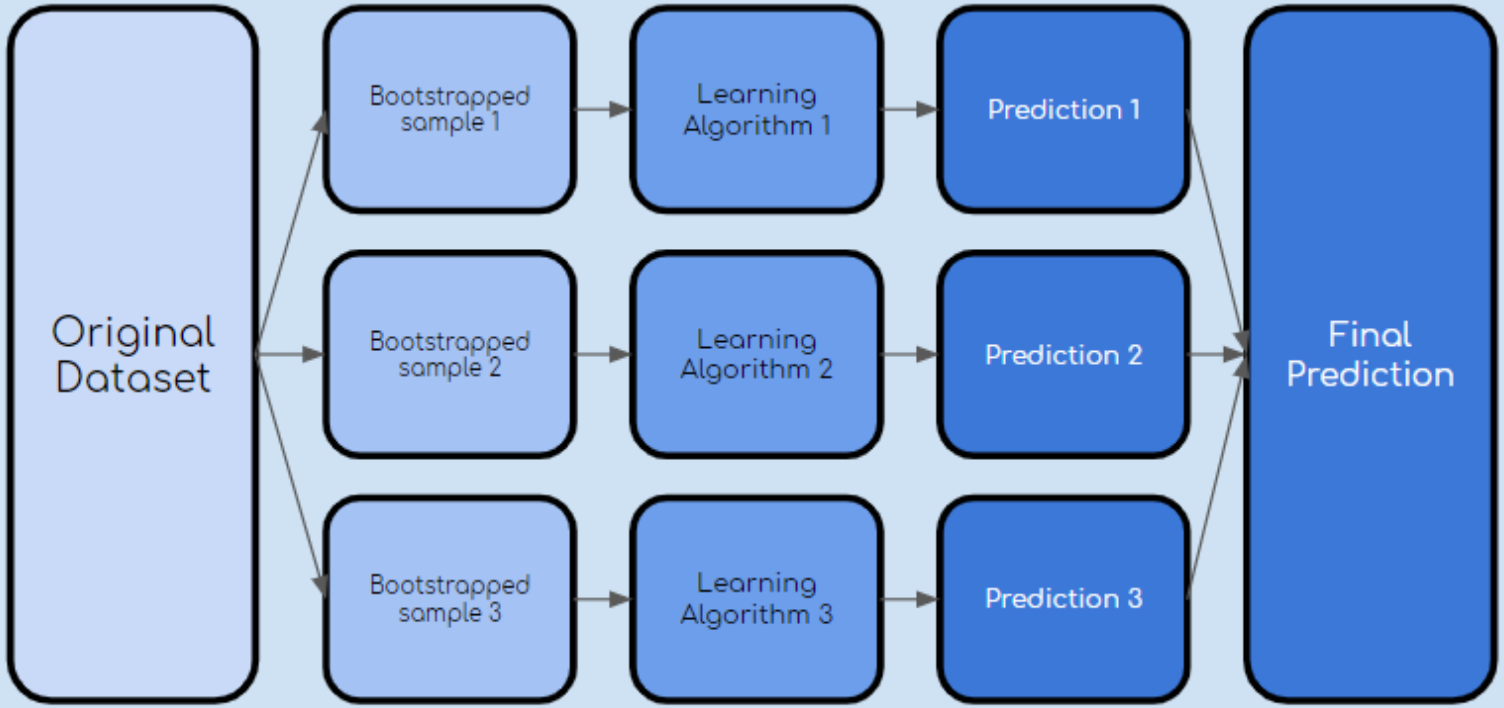


ENSEMBLE METHODS

ENSEMBLE METHODS (LEARNING)

- **Ensemble methods** are techniques in machine learning where multiple models are combined to improve overall predictive performance and robustness.
- Instead of relying on a single model's predictions, ensemble methods leverage the collective wisdom of multiple models to make more accurate and reliable predictions
- Ensemble methods are particularly useful when individual models have different strengths and weaknesses, as combining them can help mitigate their individual limitations.

Ensemble Learning, Bagging, and Boosting

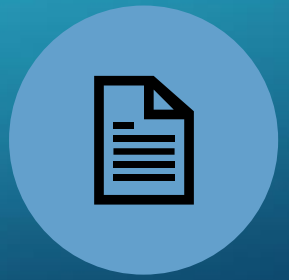


ENSEMBLE LEARNING TECHNIQUES

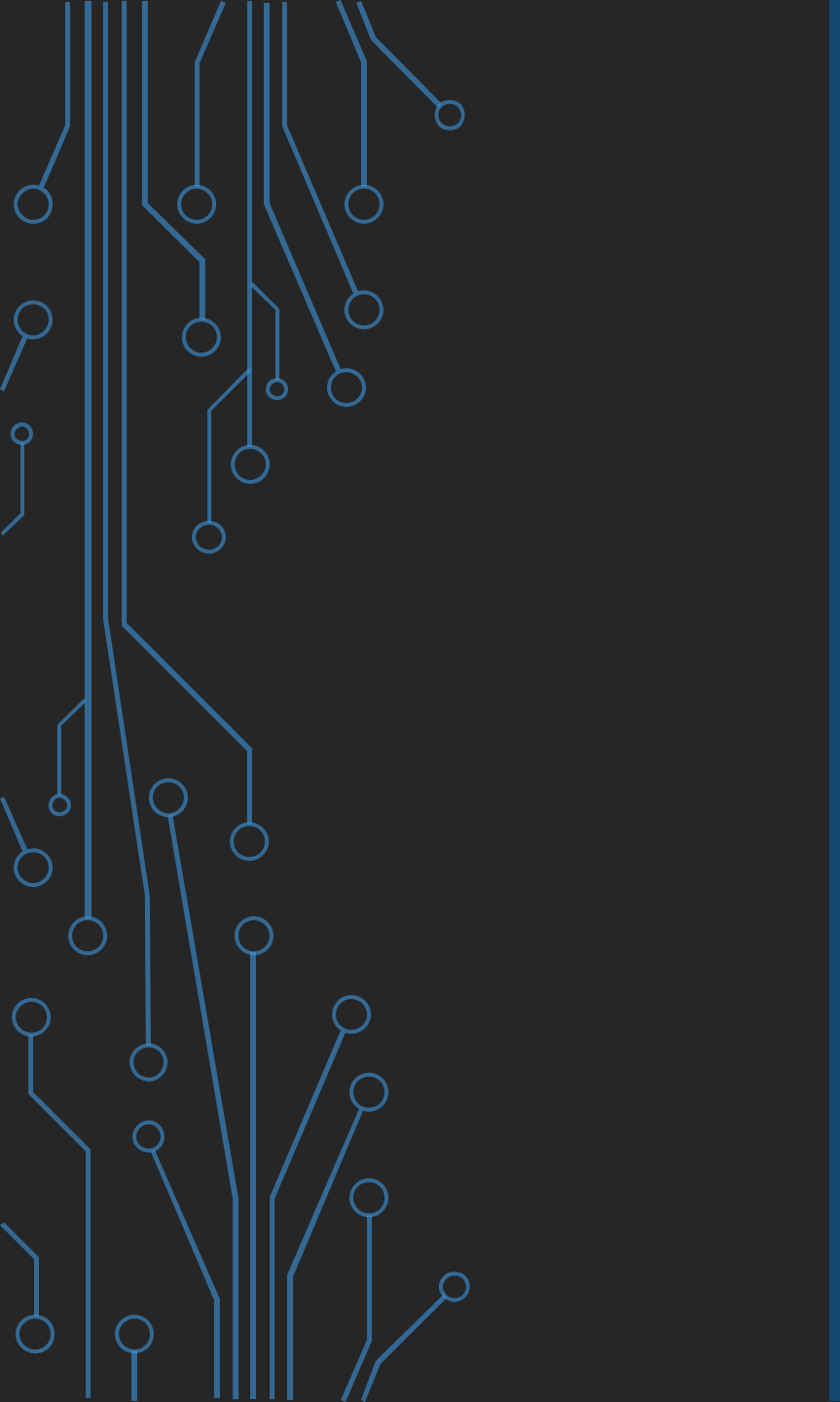
Bootstrap Aggregating (Bagging)

Boosting

Stacking



Random subspace method



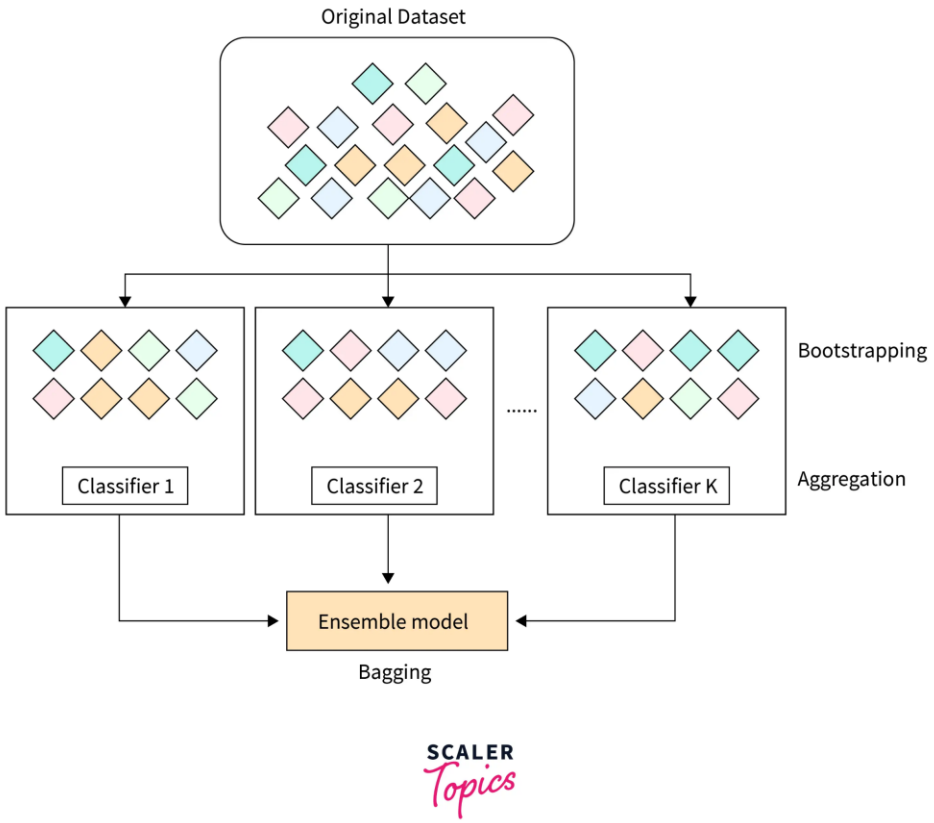
BAGGING

BAGGING

- **Bagging**, short for Bootstrap Aggregating, is an ensemble machine learning technique used to improve the accuracy and robustness of models.
- It involves training multiple instances of the same base model on different subsets of the training data and then aggregating their predictions to make the final prediction.
- Bagging is a method for **variance reduction** (Highly Overfitting models).

Example: Random Forest

BAGGING (CONT.)



BAGGING STEPS

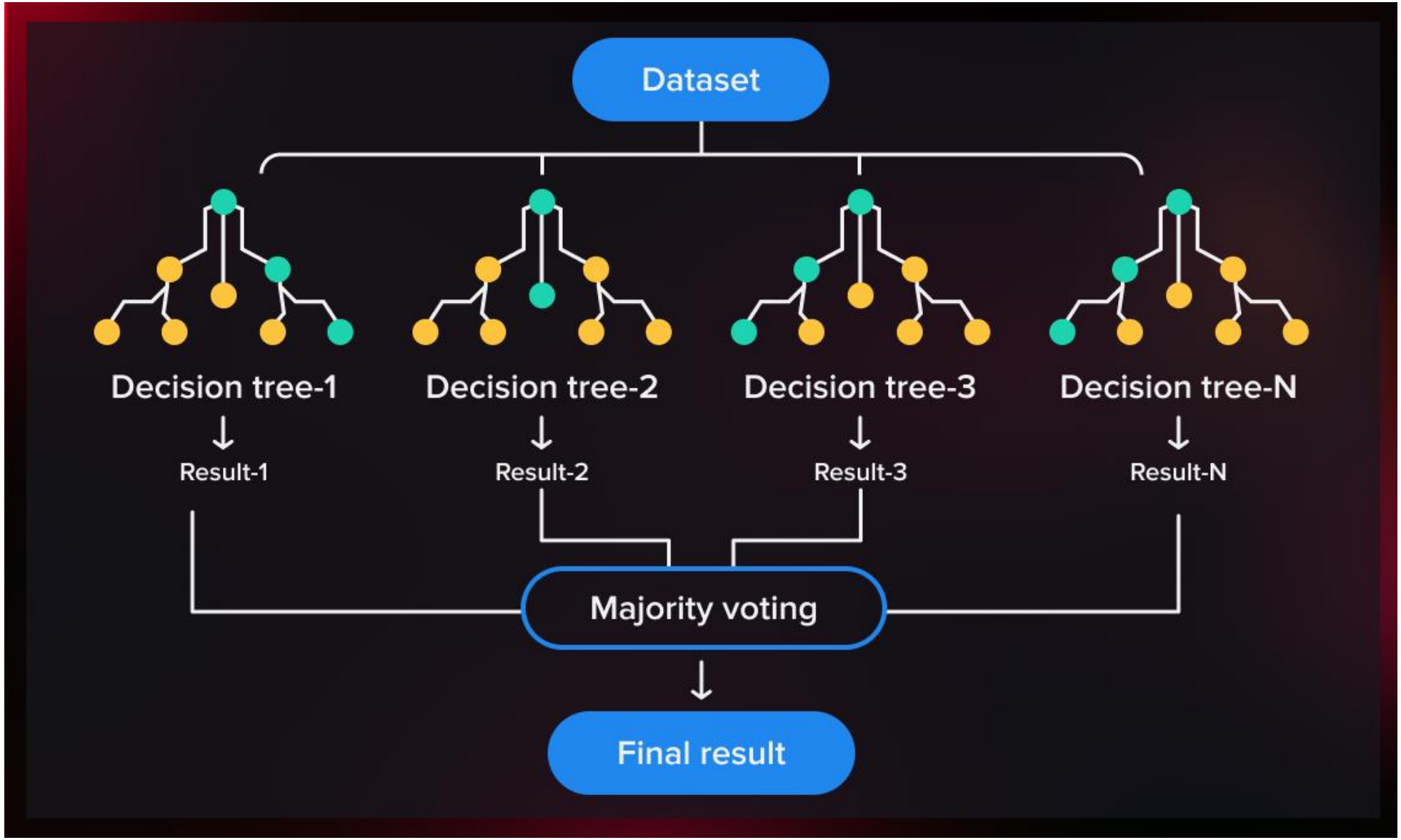
The key steps of the bagging technique are as follows:

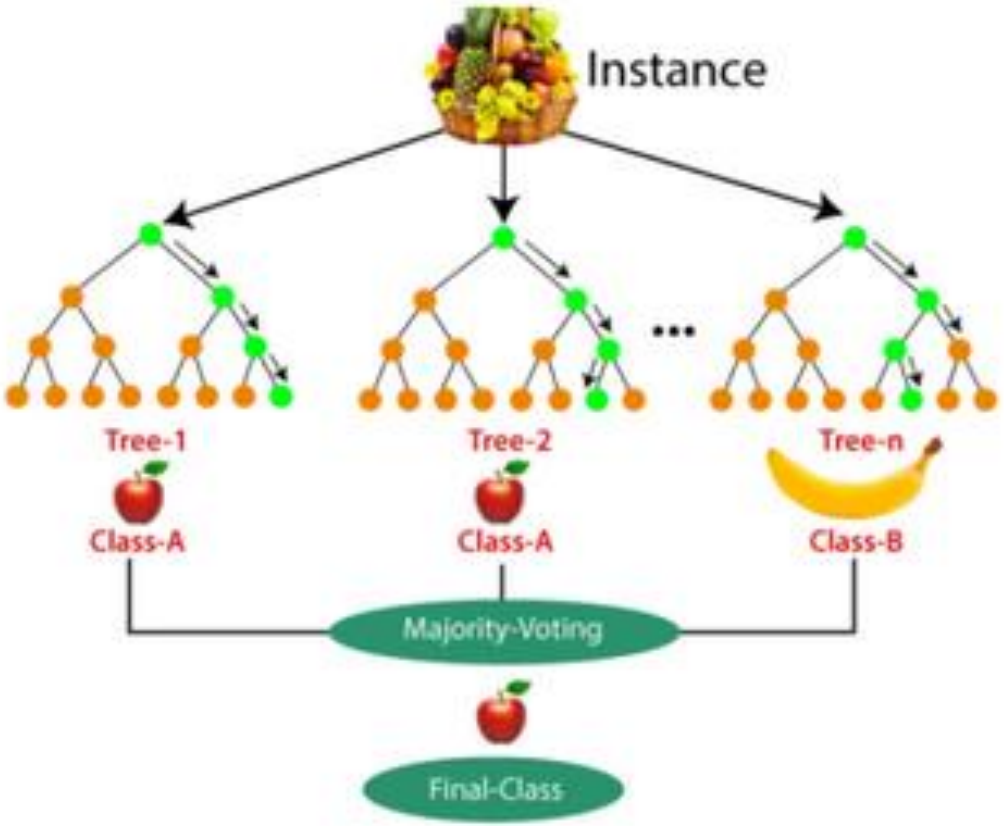
1. **Bootstrapping**: Bootstrapping involves creating multiple random subsets (samples) of the training data with replacement. Each subset is the same size as the original dataset, but some data points may be duplicated, and others may be left out. This process is done independently for each base model.
2. **Training Base Models**: For each of these bootstrapped subsets, a separate instance of the base model is trained. This results in multiple base models, each with slightly different training data.
3. **Aggregating Predictions**: When making predictions, the outputs of all the base models are combined. The most common aggregation methods are:
   - **Classification**: The final class prediction is determined by majority voting (for example, the most frequently predicted class among the base models).
   - **Regression**: The final prediction is often the average or median of the predictions made by the base models.

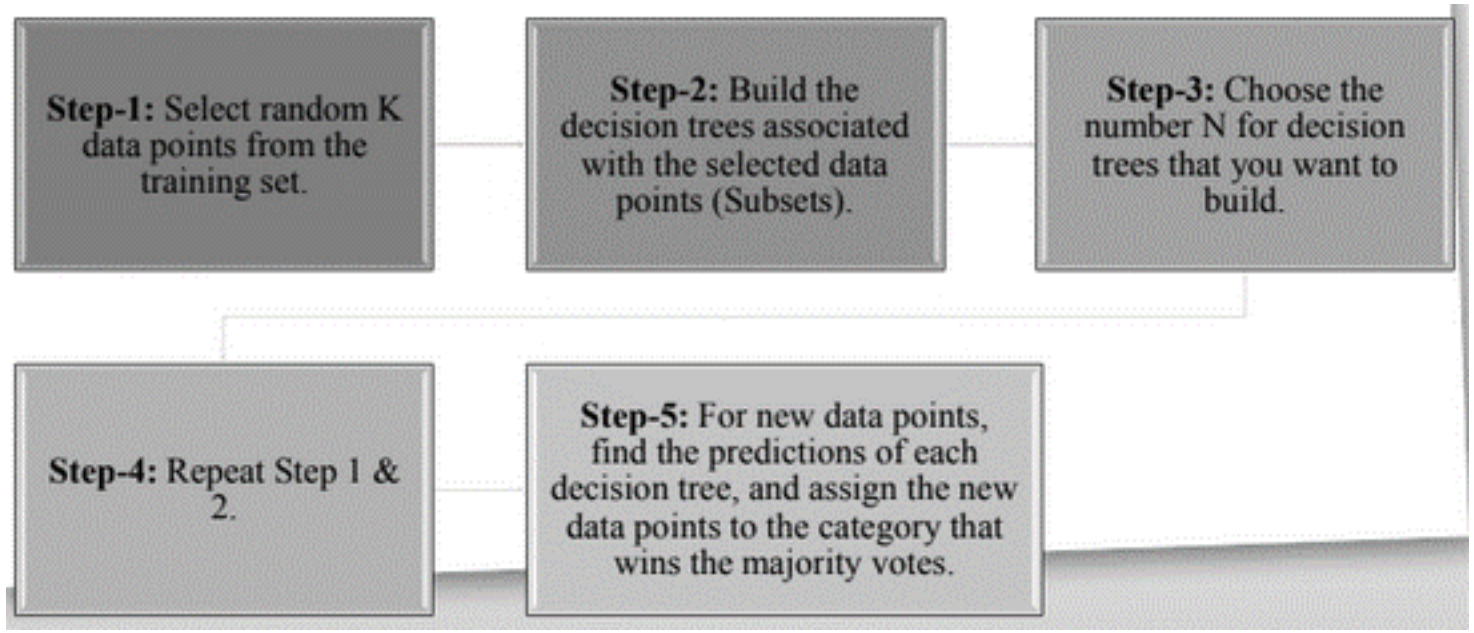
RANDOM FOREST

- A **Random Forest** is a popular ensemble learning algorithm that combines the concepts of bagging and random feature selection to create a more robust and accurate predictive model.

- It is particularly effective for both classification and regression tasks. The algorithm is based on using ***MULTIPLE DECISION TREES***, where each tree is trained on a different subset of the data and a random subset of features.


Dataset
Decision tree-1
Decision tree-2
Decision tree-3
Decision tree-N
Result-1
Result-2
Result-3
Result-N
Majority voting
Final result


Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class



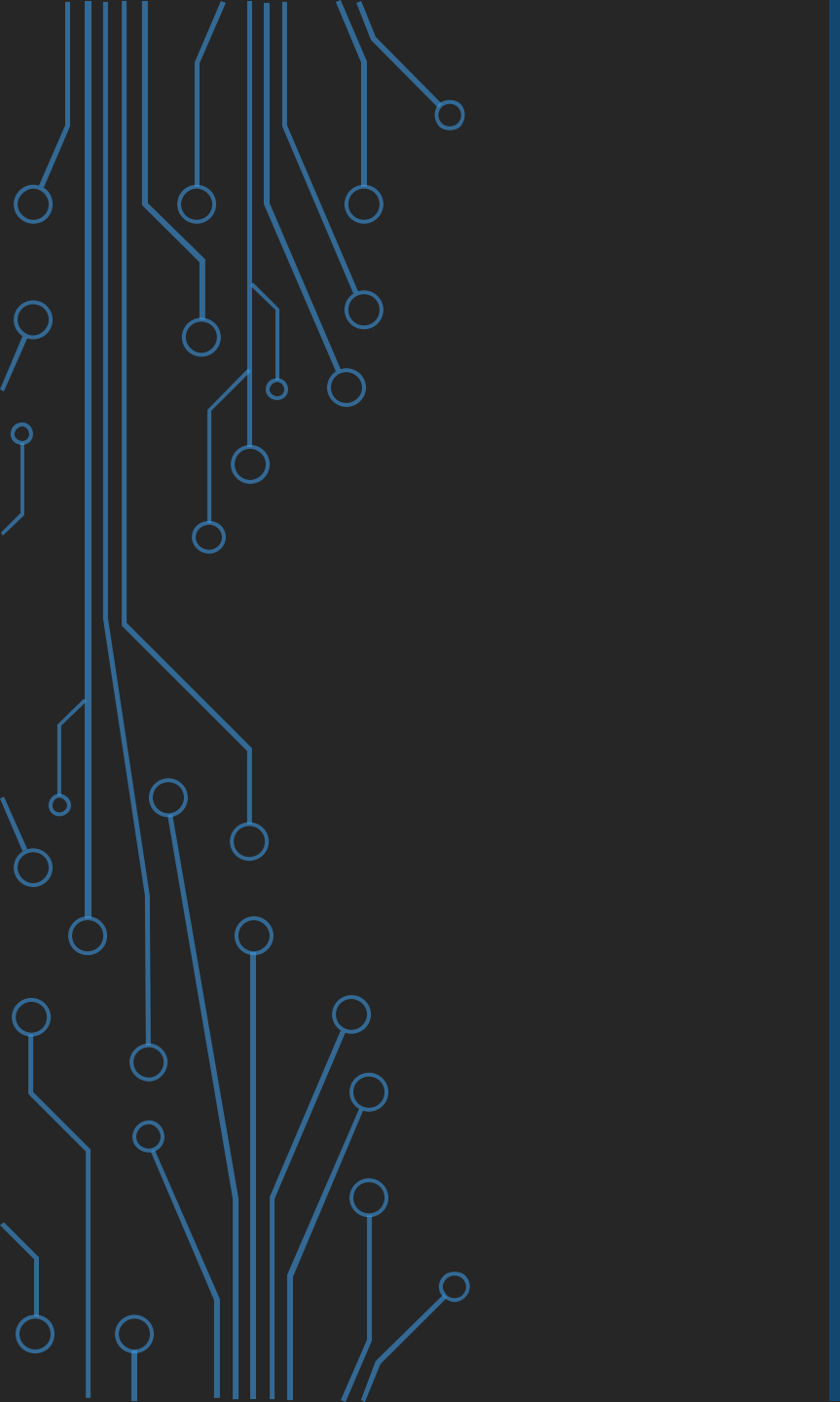
STEPS OF RANDOM FOREST

ADVANTAGES OF RANDOM FOREST

Key advantages of Random Forests:

- **Reduced Overfitting**: By training multiple trees on different subsets of the data, Random Forests mitigate overfitting and improve generalization compared to a single decision tree.
- **Robustness to Outliers and Noise**: The ensemble nature of Random Forests makes them more robust to noisy or outlier data points.
- **Feature Importance**: Random Forests can provide insights into feature importance by analyzing how much each feature contributes to reducing the impurity or variance in the predictions.
- **Parallelism**: The training of individual trees in a Random Forest can be parallelized, leading to faster training times on multicore processors.

TIME FOR PRACTICALITY (10 MINUTES)



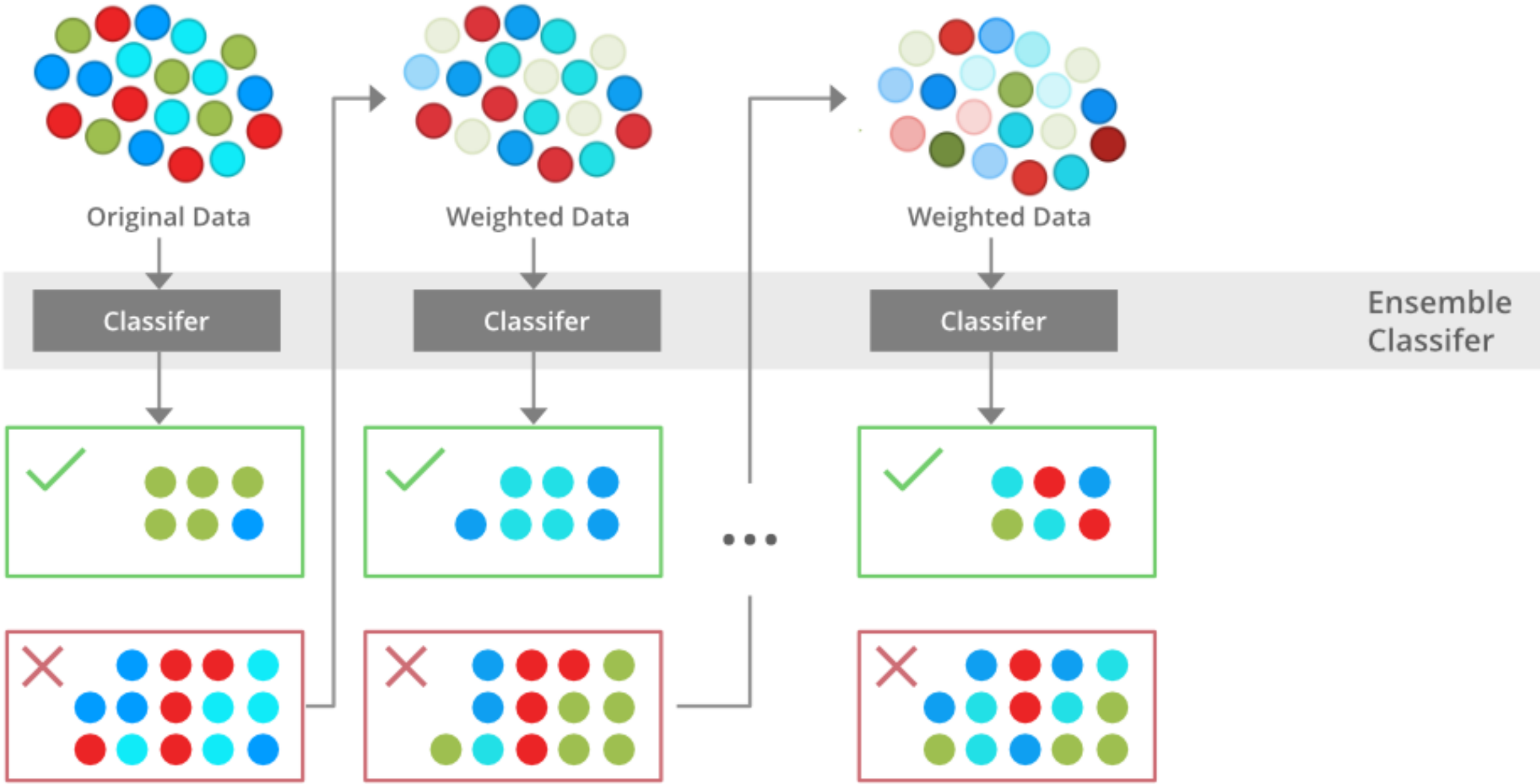
BOOSTING

BOOSTING

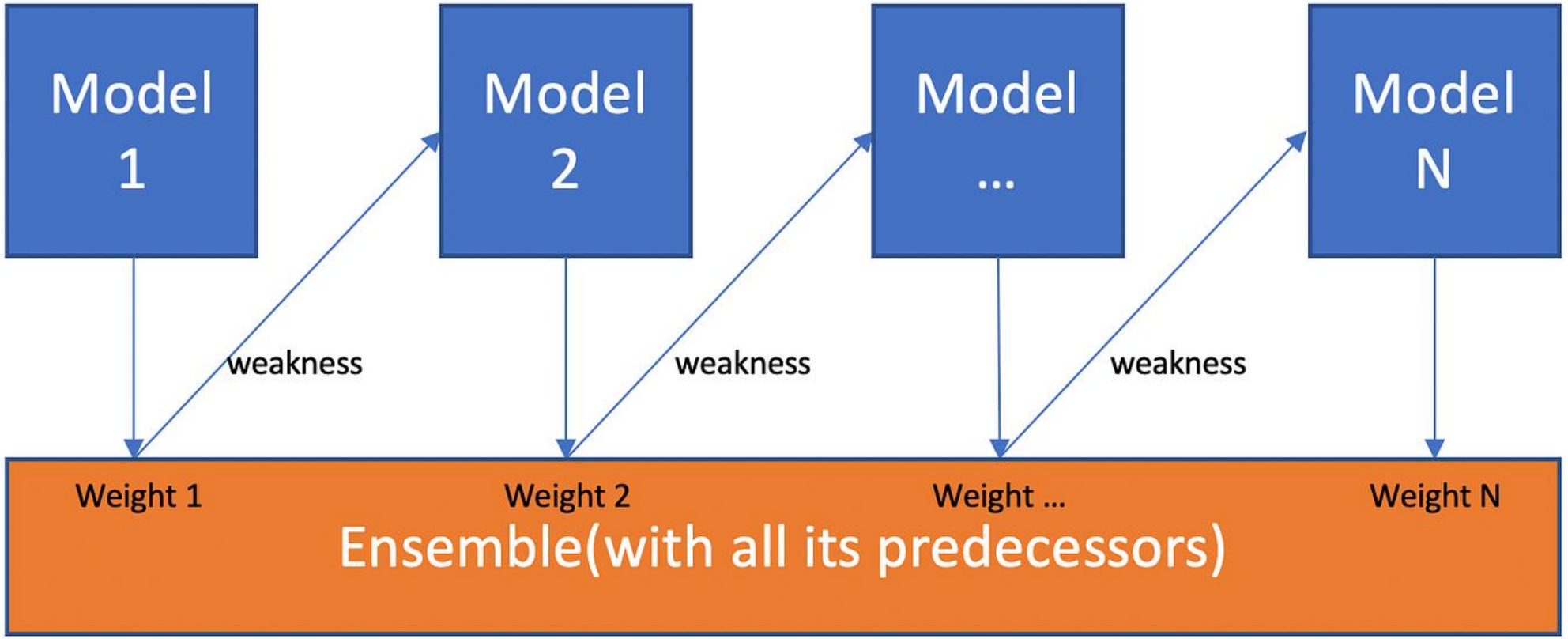
- **Boosting** is an ensemble learning technique that aims to improve the performance of a machine learning model by combining the predictions of **MULTIPLE WEAK LEARNERS** (often simple models) into a single strong learner.

- Unlike bagging, which focuses on training parallel models independently, **boosting trains models sequentially, with each subsequent model attempting to correct the errors made by the previous ones**.

- **Boosting** is a method for **bias reduction** (Highly Underfitting models).


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble
Classifer

Model 1,2,…, N are individual models (e.g. decision tree)



BOOSTING STEPS

1. **Training Initial Model (Weak Learner)**: The first model (weak learner) is trained on the entire dataset. This model might not perform well initially but is better than random guessing.
2. **Weighted Error Calculation**: After the first model is trained, the instances that it misclassified are assigned higher weights. This emphasizes the importance of these misclassified instances in the subsequent model's learning process.
3. **Training Subsequent Models**: Subsequent models are trained with a focus on the instances that the previous models struggled with. These models aim to correct the errors made by the previous ones.
4. **Weighted Voting/Averaging**: When making predictions, the predictions of all models are combined, usually with weighted voting or averaging. More accurate models contribute more to the final prediction.
5. **Stopping Criterion**: Boosting continues until a predefined stopping criterion is met, such as a maximum number of iterations or when the performance on the training set stops improving.

SOME POPULAR BOOSTING ALGORITHMS

AdaBoost (Adaptive Boosting)

Gradient Boosting Machines (GBM)

XGBoost

LightGBM

CatBoost

ADABOOST

- **AdaBoost**, short for Adaptive Boosting, is a popular ensemble learning technique that belongs to the family of boosting algorithms.
- It's specifically designed to enhance the performance of **WEAK LEARNERS** (models that perform slightly better than random guessing) by combining them into a strong, accurate ensemble model.
- AdaBoost is particularly effective for binary classification tasks.

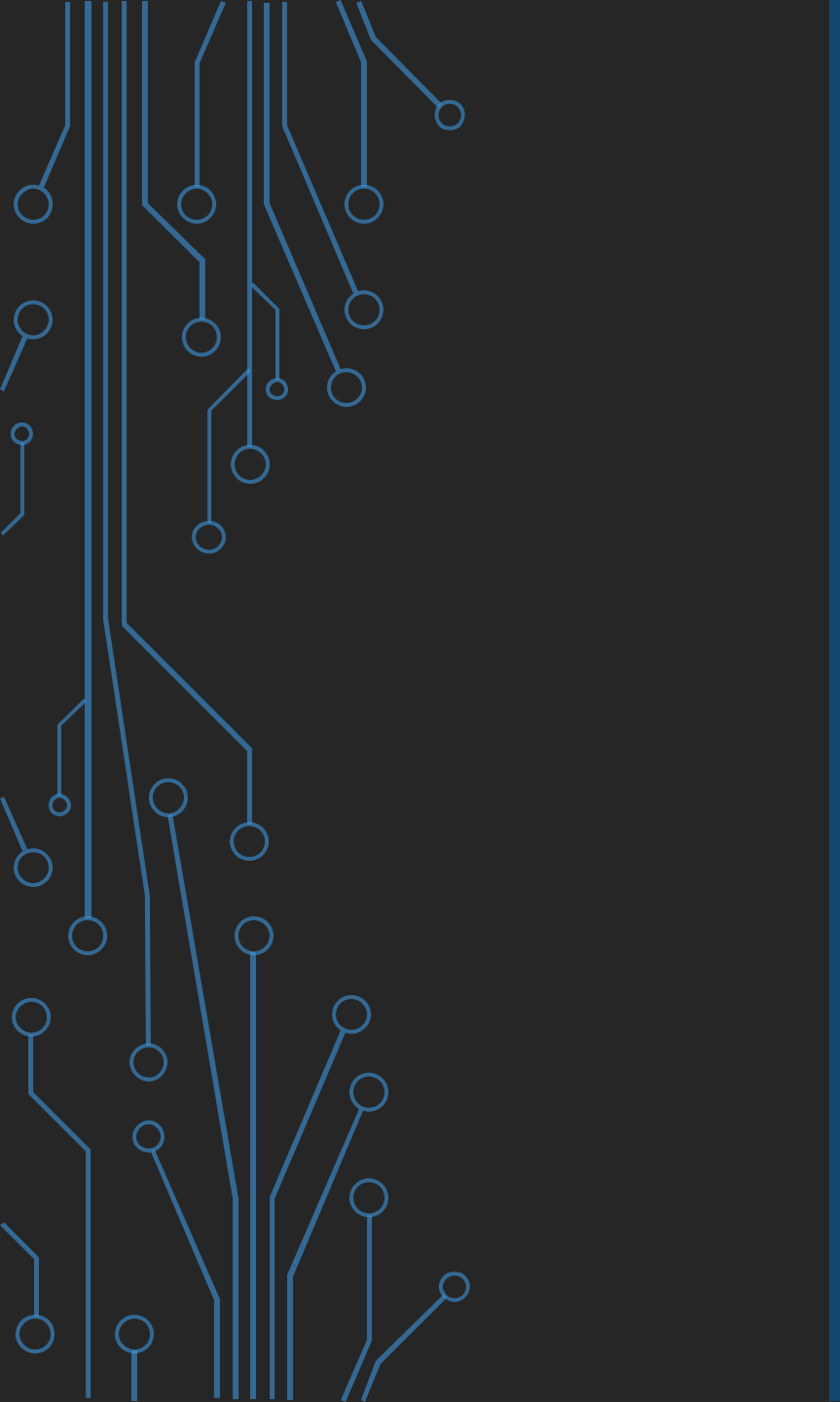
HOW ADABOOST WORKING ?

Here's how AdaBoost works:

1. **Initialization**: Each instance in the training dataset is initially assigned equal weight.
2. **Training Weak Learner**: In each iteration, a weak learner (often a decision tree with limited depth) is trained on the weighted dataset. The weak learner's performance is slightly better than random guessing.
3. **Weighted Error**: The weighted error of the weak learner is calculated as the sum of the weights of misclassified instances. The weaker the learner, the higher its weighted error will be.
4. **Model Weight**: The weight assigned to the weak learner's prediction is calculated based on its accuracy. More accurate models are given higher weight.
5. **Updating Weights**: The weights of instances are updated, increasing the weights of misclassified instances. This ensures that the next weak learner focuses more on these instances.
6. **Final Prediction**: The ensemble prediction is formed by combining the predictions of all weak learners, weighted by their individual model weights.

TIME FOR PRACTICALITY (10 MINUTES)

BREAK (10 MINUTES)



STACKING

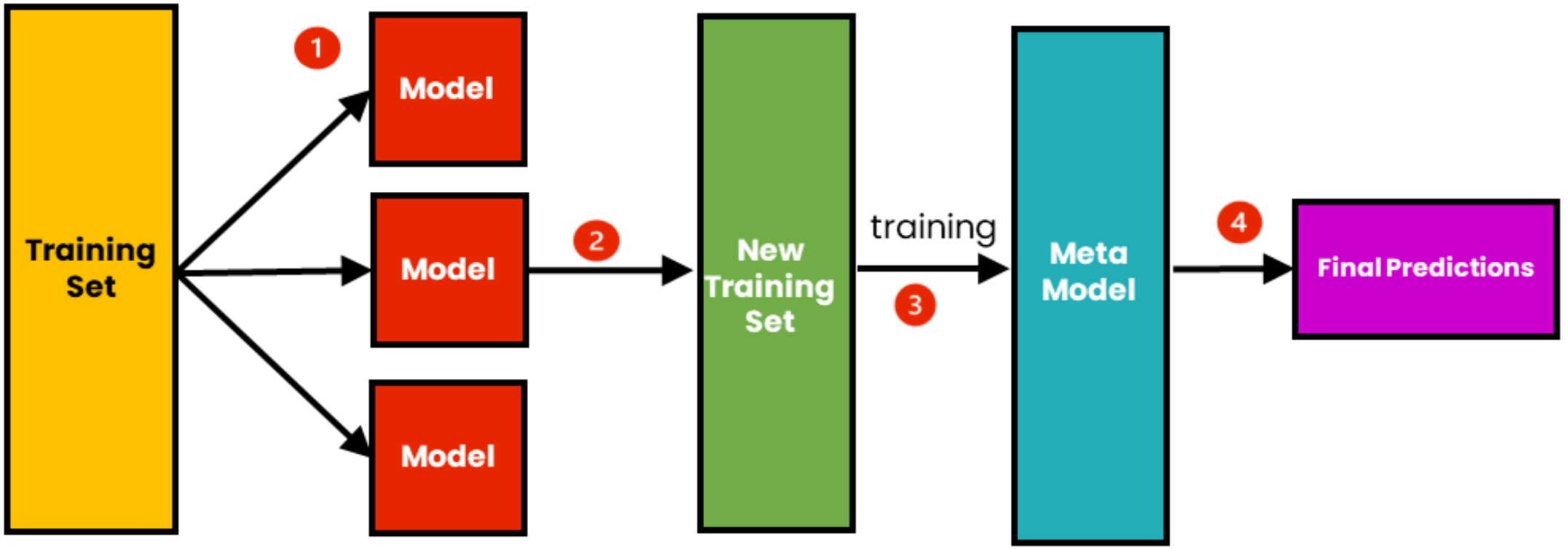
STACKING

- **Stacking**, also known as stacked generalization, is an ensemble learning technique that combines the predictions of multiple base models by training a **higher-level** model (meta-learner) to make the final prediction.

- Unlike traditional ensembles like bagging and boosting, where individual models' predictions are simply averaged or combined, stacking aims to learn how to best combine the predictions of the base models through another model.

- Stacking is used to improve performance of **STRONG LEARNERS**.

The Process of Stacking



STACKING STEPS

The process of stacking involves the following steps:

1. **Base Models**: Train multiple diverse base models on the same training data. These models can be of different types or trained with different algorithms.
2. **Predictions**: Make predictions on the validation (or test) dataset using each of the trained base models.
3. **Constructing Meta-Features**: Use the predictions from the base models as features to create a new dataset, often called the "meta-features" or "second-level features." Each base model's predictions become a new feature in this dataset.
4. **Meta-Learner**: Train a higher-level model (meta-learner) using the meta-features as input and the true target values as output. The meta-learner learns how to combine the base models' predictions effectively to make the final prediction.
5. **Final Prediction**: When making predictions on new, unseen data, first obtain the predictions from the base models, use these predictions to create meta-features, and then pass the meta-features through the trained meta-learner to generate the final prediction.

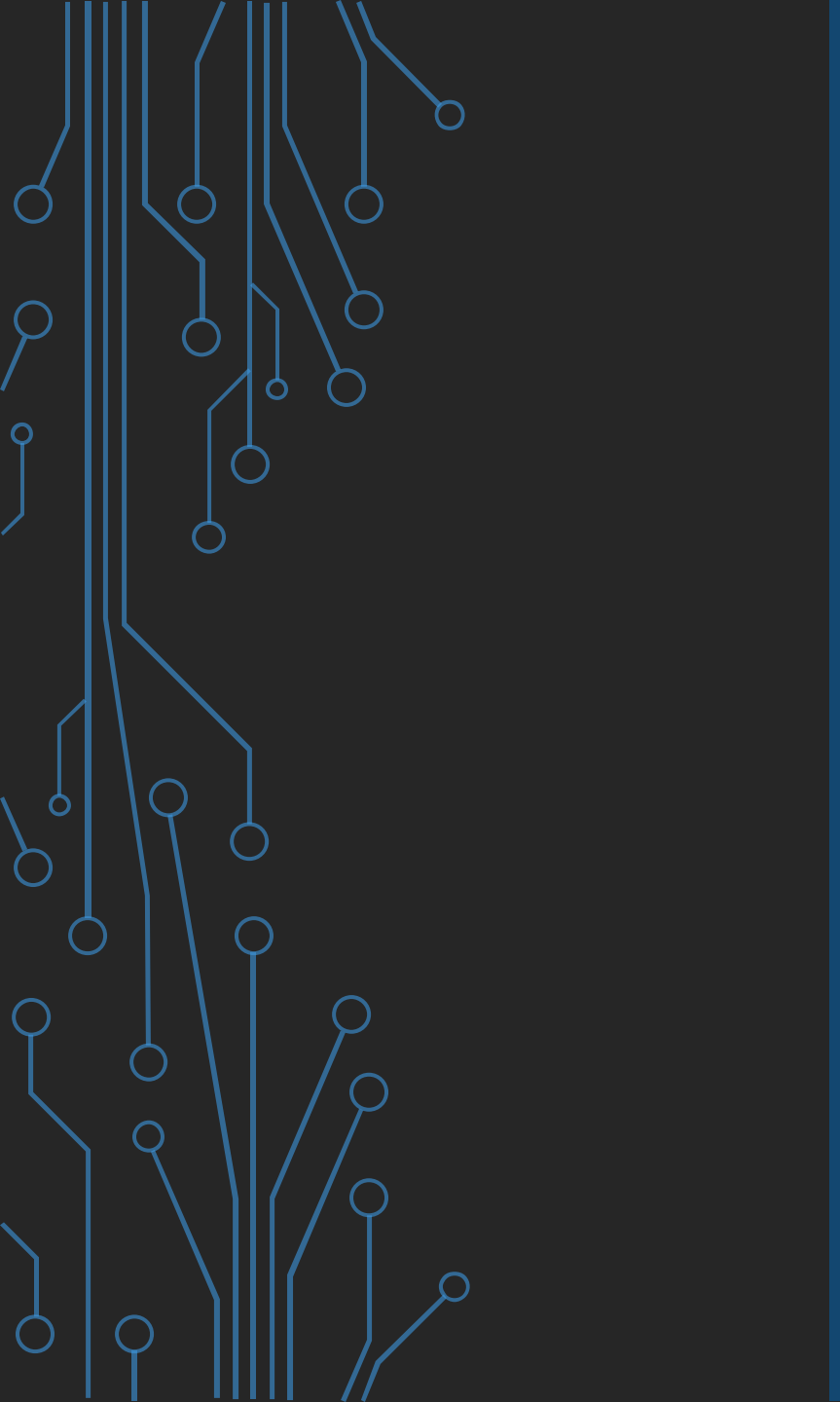
ADVANTAGES OF STACKING TECHNIQUE

Key advantages of stacking:

- **Model Combination**: Stacking allows different base models to compensate for each other's weaknesses. If one model performs well in certain situations but poorly in others, stacking can combine their strengths.
- **Optimal Combination**: The meta-learner learns how to optimally weigh and combine the predictions of the base models, potentially leading to better overall performance.
- **Flexibility**: Stacking is flexible in terms of the types of base models that can be used. Different algorithms, architectures, or even models with different preprocessing pipelines can be combined.

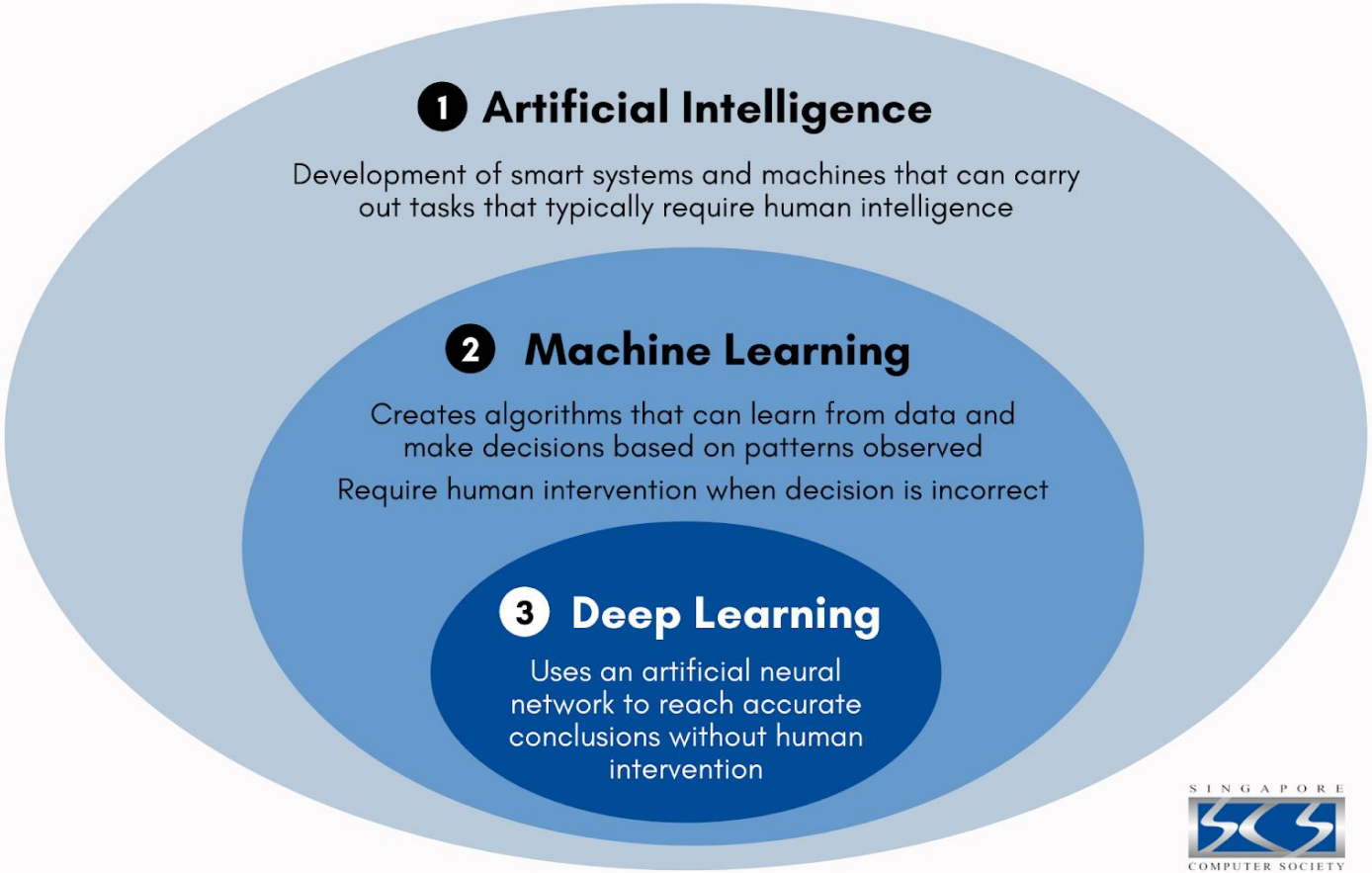
	Bagging	Boosting	Stacking
Purpose	Reduce Variance	Reduce Bias	Improve Accuracy
Base Learner Types	Homogeneous	Homogeneous	Heterogeneous
Base Learner Training	Parallel	Sequential	Meta Model
Aggregation	Max Voting, Averaging	Weighted Averaging	Weighted Averaging

TIME FOR PRACTICALITY (10 MINUTES)



INTRODUCTION TO DEEP LEARNING

ARTIFICIAL INTELLIGENCE VS MACHINE LEARNING VS DEEP LEARNING

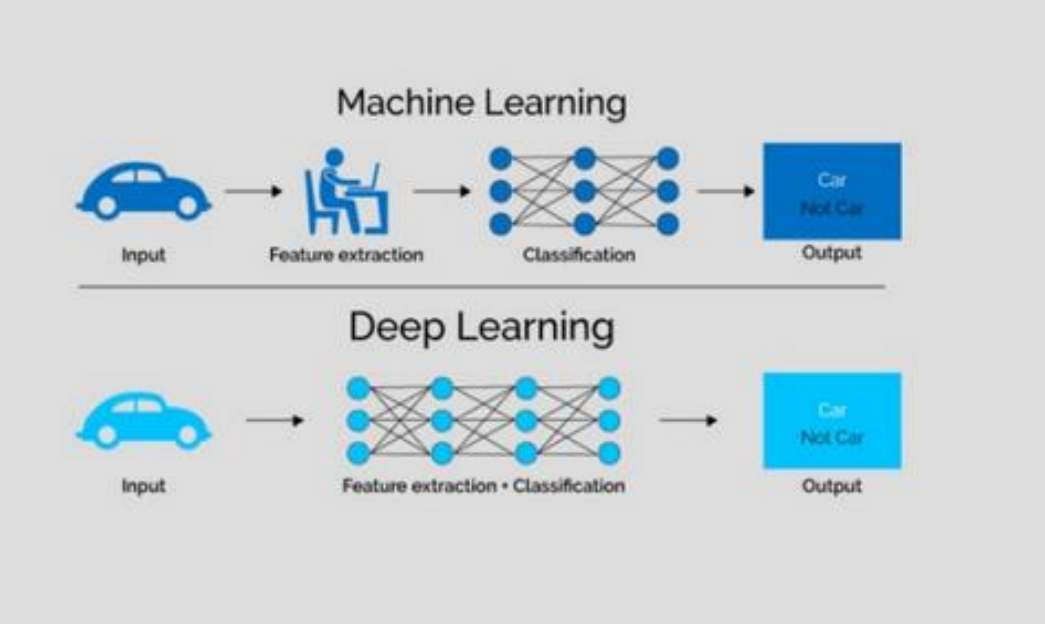


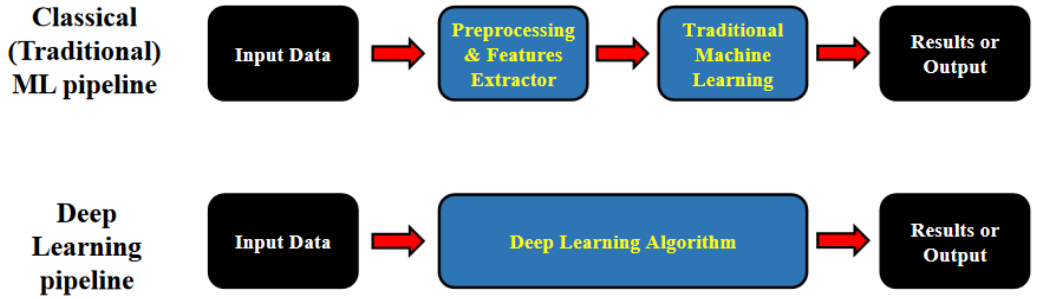
SINGAPORE COMPUTER SOCIETY

WHAT IS DEEP LEARNING ?

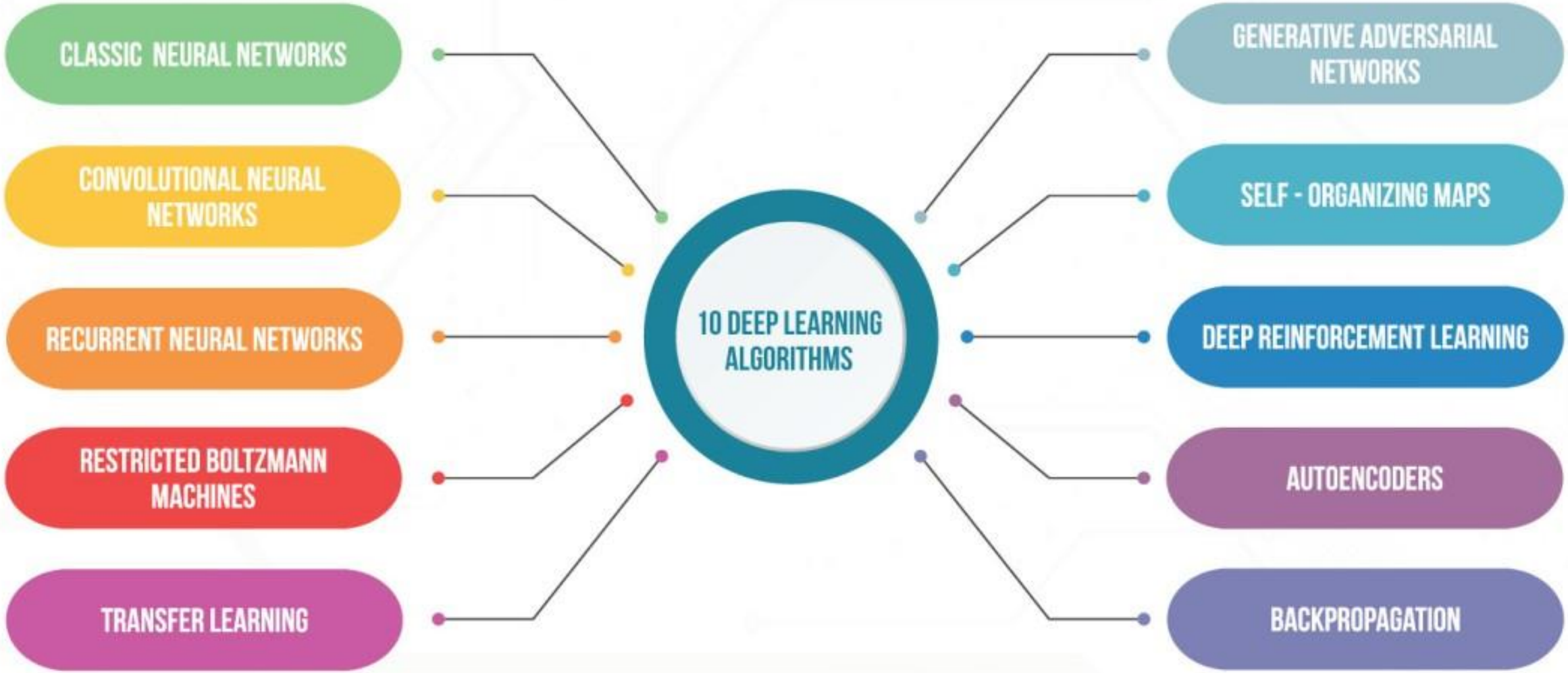
- **Deep learning** is a subset of machine learning that focuses on using ARTIFICIAL NEURAL NETWORKS **(ANN)** to model and solve complex patterns and representations in data.

- It involves training neural networks with multiple layers (hence the term "deep") to automatically learn hierarchical features from raw data.



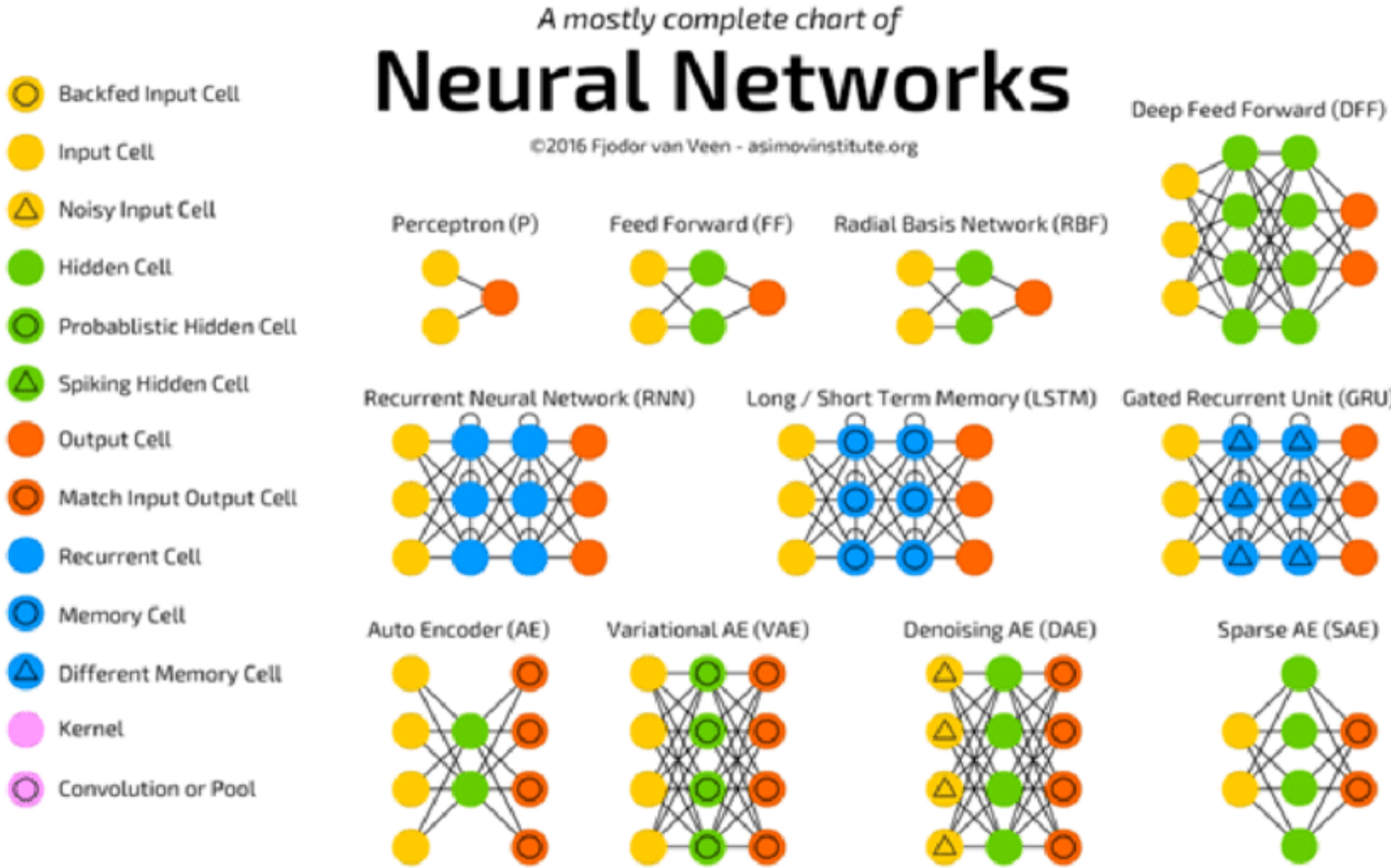


CLASSICAL ML VS. DL

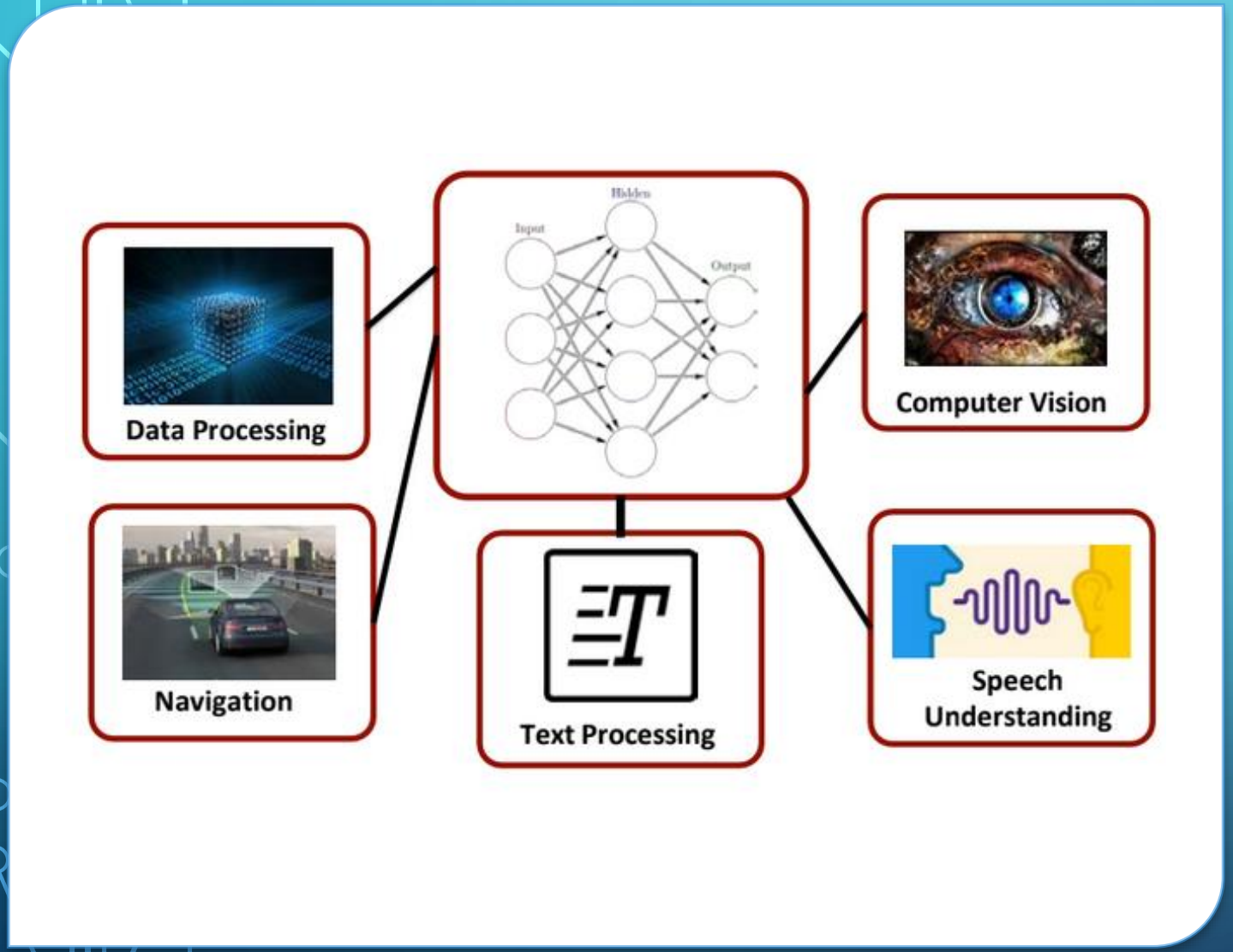


SOURCE: SHUTTERSTOCK

NEIL SAHOTA


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)

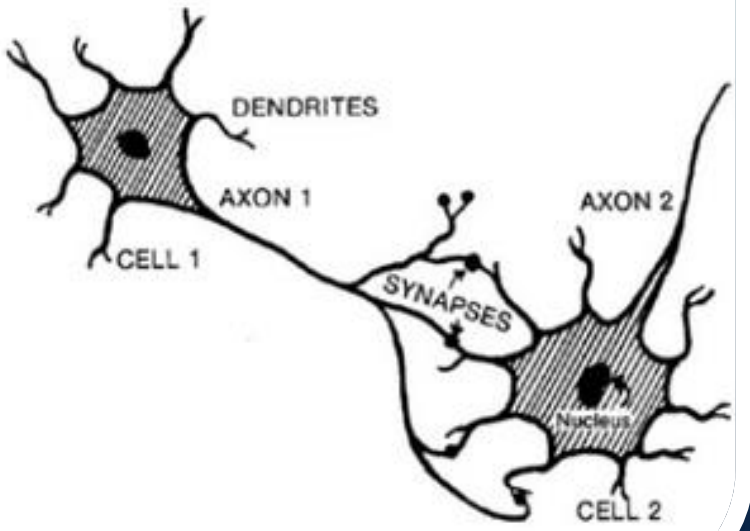
FIELDS OF DEEP LEARNING



INSPIRATION

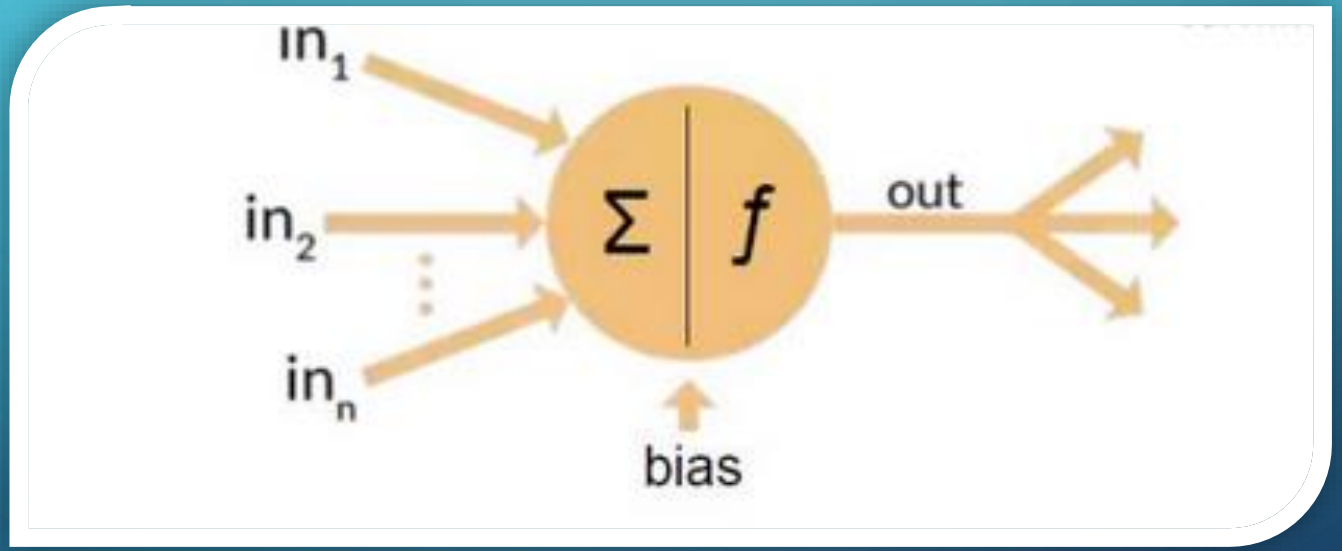
Biological Neural Network

- Dendrites accept inputs from other neurons
- Axon transmist impulses to other neurons
- Synapses are structures where impulses are transfered from one neuron to another

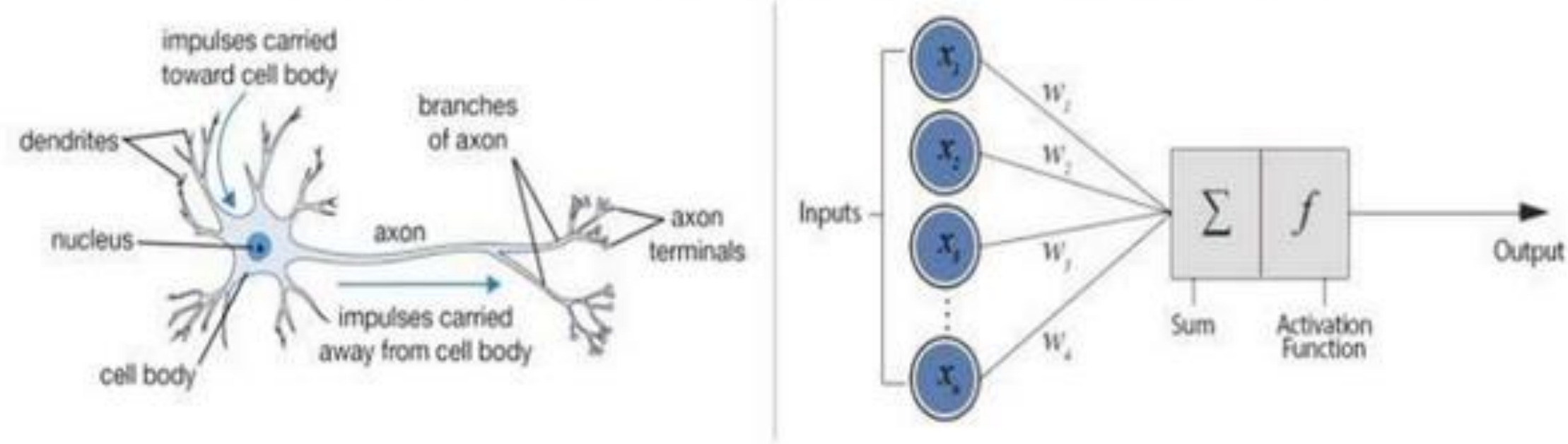


INSPIRATION (CONT.)

Artificial Neural Network

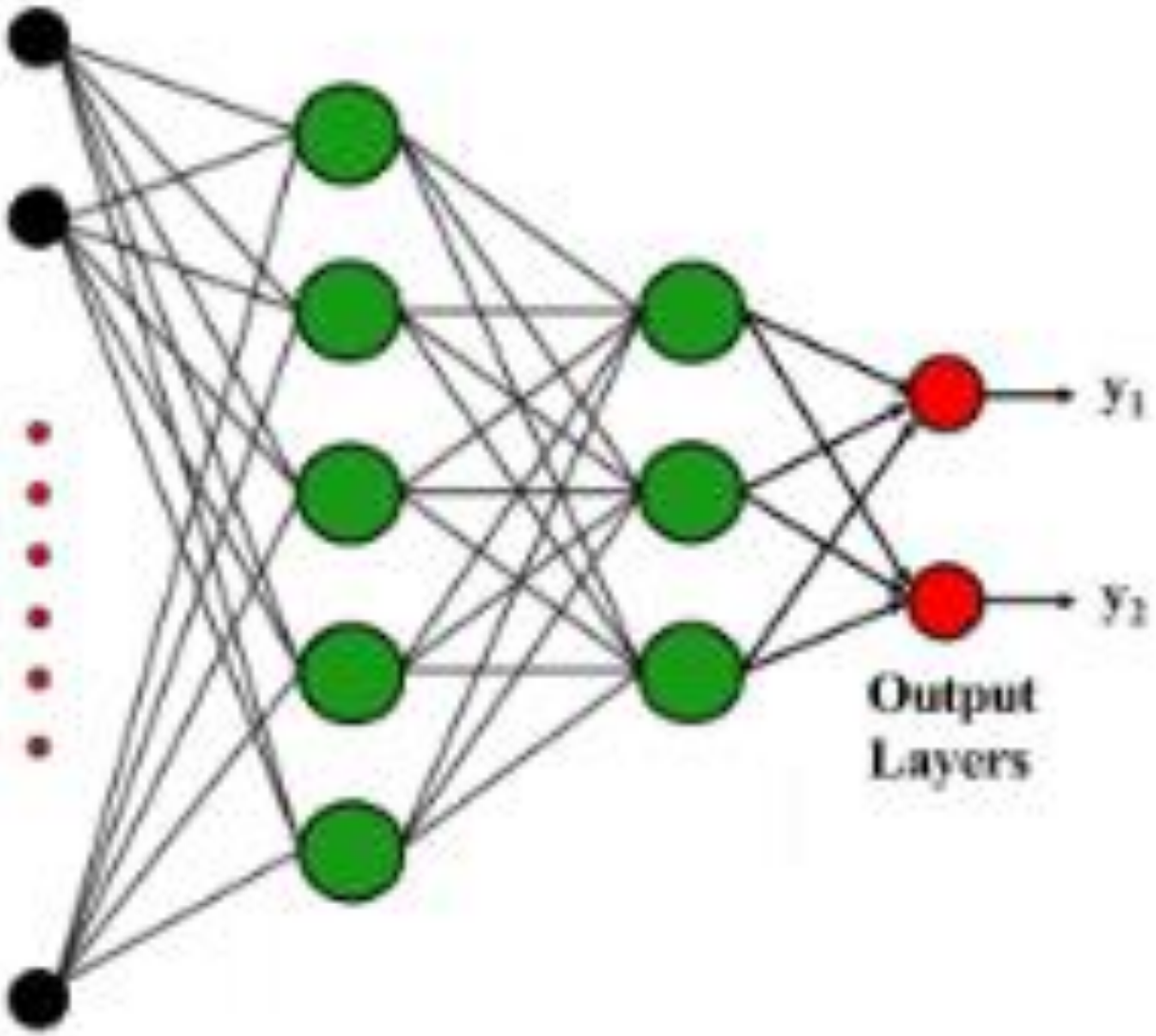


Biological Neuron versus Artificial Neural Network



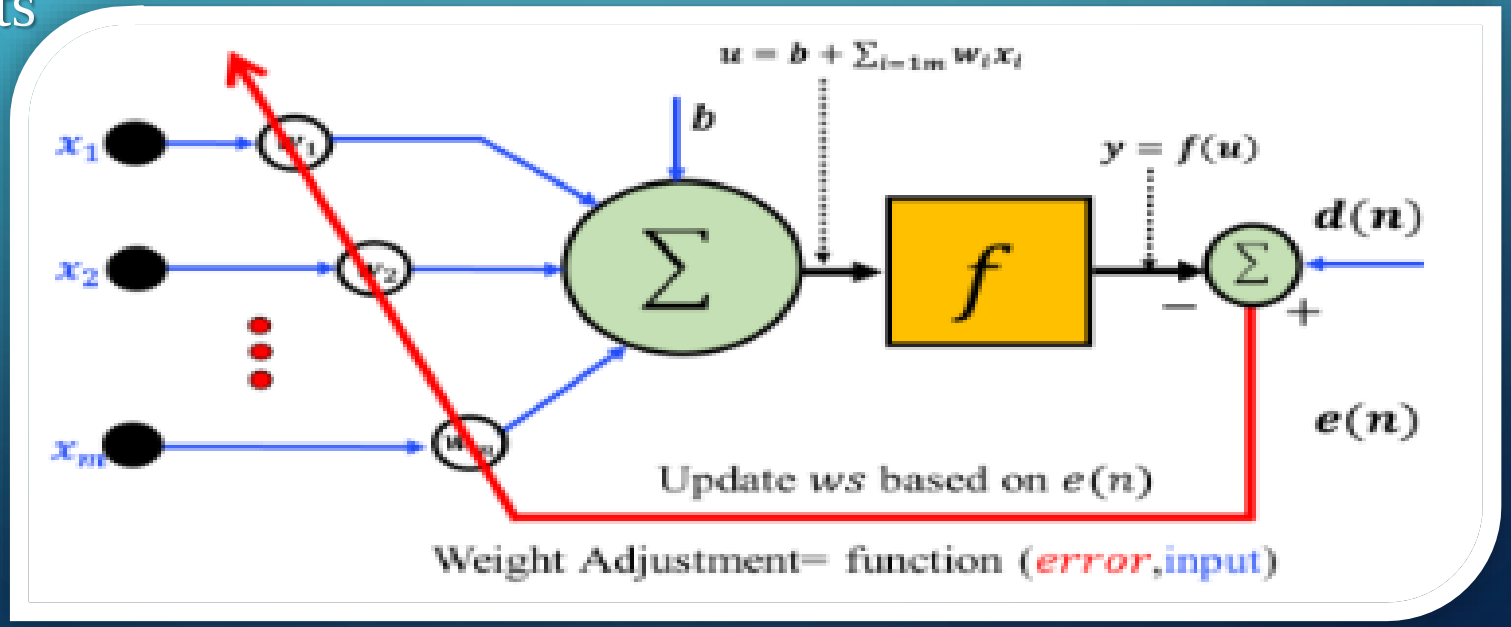
THE BEGINNING: ANN

- **Neural networks** are collections of thousands (or millions) of these simple processing units (neurons) that together perform useful computations.
- ANNs can be represented using oriented graphs
- A graph has two kinds of edges:
  - ❑ Synaptic edge representing linear input-output relation (multiplication by weight)
  - ❑ Activation edge representing a non linear input- output relation of activation function


y_1
y_2
Output
Layers

THE BEGINNING: ANN (CONT.)

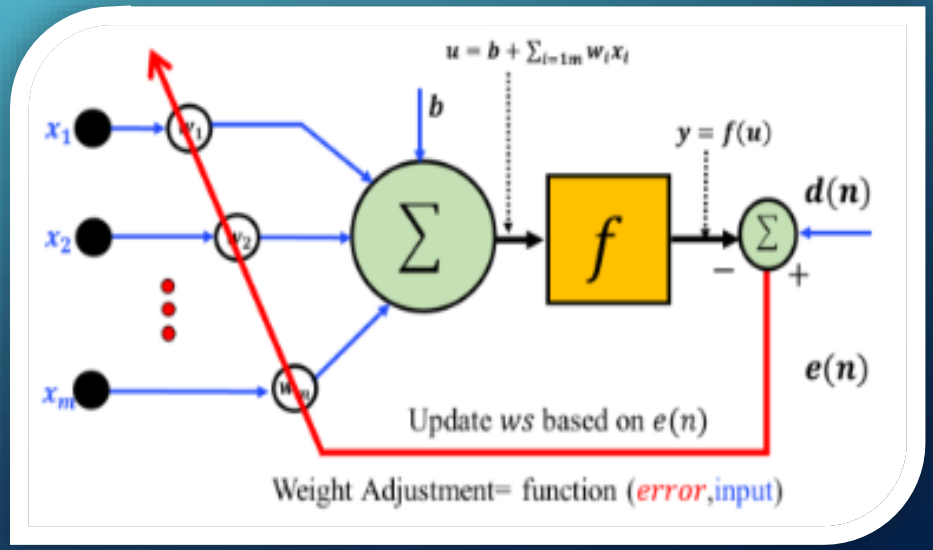
- **Neural network** are parallel processing using highly nonlinear prediction functions
- Neurons are nodes and they are **activated** based on the **weighted** sum of their inputs



THE BEGINNING: ANN (CONT.)

- **Learning** is a recursive process for optimizing a **LOSS FUNCTION (Error Rate)** using data samples (or batches) for a certain number of iterations **(or epochs).**
- **Backpropagation** (or error propagation) is an efficient way to compute activation function gradients

Gradient Descent



WHAT IS “ACTIVATION FUNCTION” ?

- An **activation function** is a crucial component in artificial neural networks and deep learning models. It introduces **NON-LINEARITY** into the network, allowing it to learn and approximate complex relationships in data.

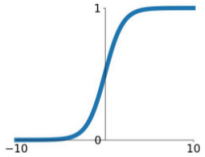
- Activation functions determine the output of a neuron (or node) in a neural network based on its weighted inputs.

TYPES OF ACTIVATION FUNCTION

Activation Functions

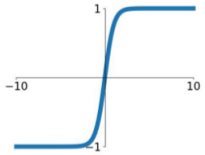
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



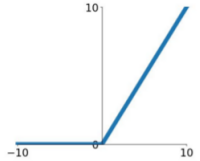
tanh

$$\tanh(x)$$



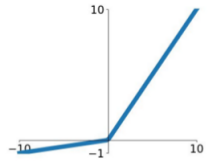
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

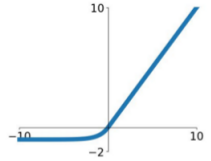


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



OUTPUT ACTIVATION FUNCTIONS

1. **Binary Classification**:
   - **Sigmoid Activation**: The sigmoid activation function maps the network's output to a range between 0 and 1. It's suitable for binary classification problems where you want the network to predict the probability of belonging to the positive class.
2. **Multi-Class Classification**:
   - **Softmax Activation**: The softmax activation function is used in the output layer for multi-class classification problems. It converts the network's raw scores into a probability distribution over multiple classes. Each class probability is between 0 and 1, and they all sum up to 1.
3. **Regression**:
   - **Linear Activation**: For regression tasks, where you're predicting continuous numeric values, a linear activation is often used in the output layer. The network's raw output values are used as predictions.

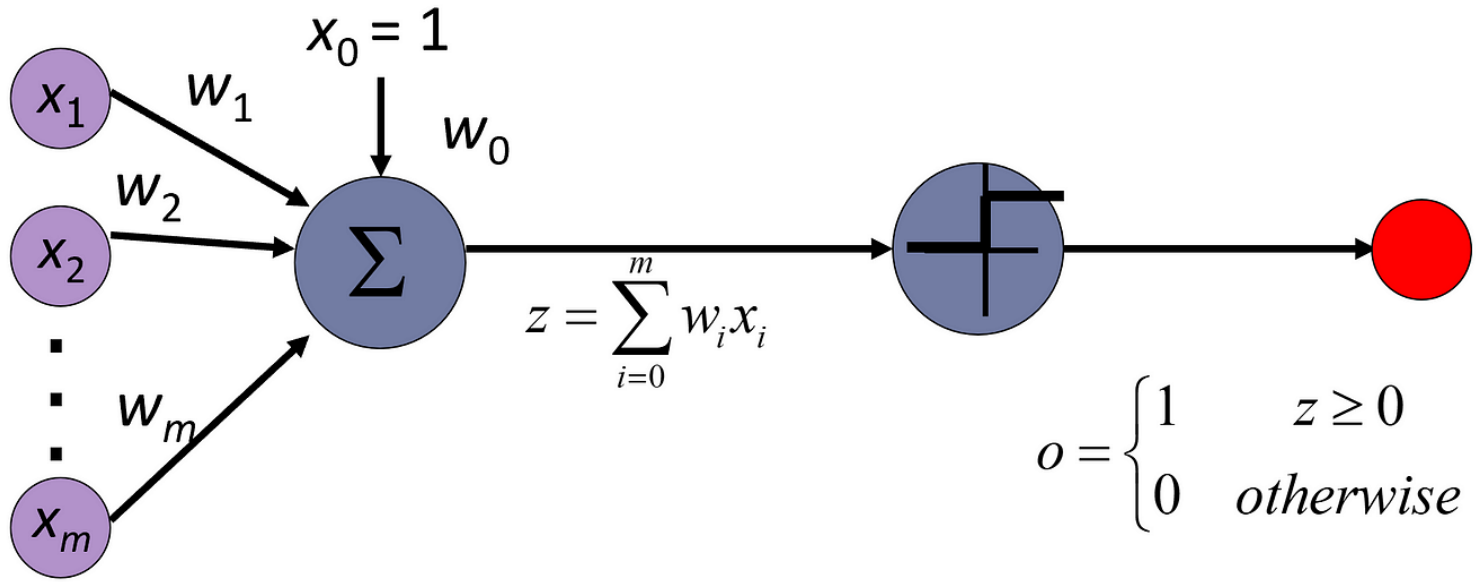
COST FUNCTION

The choice of objective function depends on the type of task the neural network is designed to solve. Different tasks, such as classification, regression, and more, require different types of loss functions. Here are some common objective functions for different types of tasks:

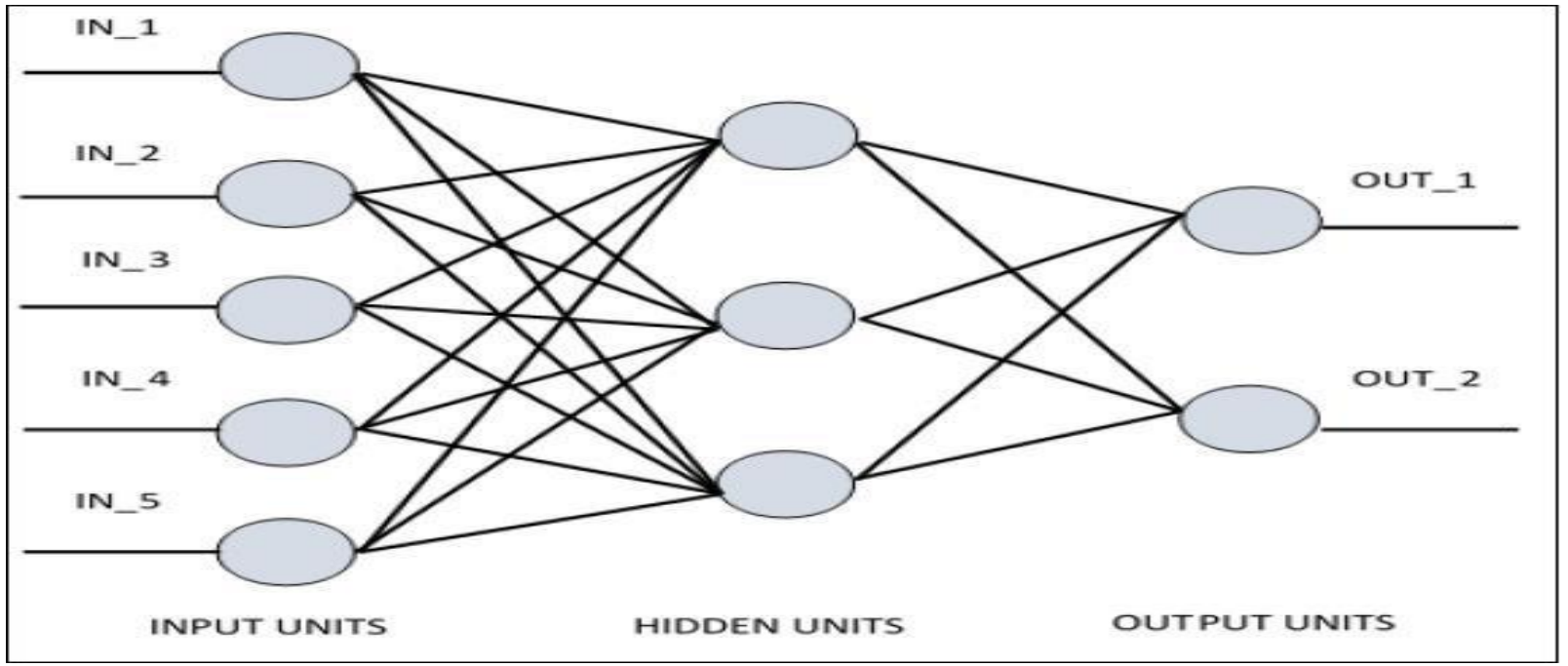
1. **Classification Tasks**:
   - **Binary Cross-Entropy Loss (Log Loss)**: Used for binary classification tasks. It measures the dissimilarity between the predicted probabilities and the actual binary labels.
   - **Categorical Cross-Entropy Loss**: Used for multi-class classification tasks. It quantifies the difference between the predicted class probabilities and the true one-hot encoded labels.
2. **Regression Tasks**:
   - **Mean Squared Error (MSE)**: Used for regression tasks. It calculates the average of the squared differences between predicted and true continuous values.
   - **Mean Absolute Error (MAE)**: Similar to MSE, but it calculates the average of the absolute differences.
   - **Huber Loss**: A combination of MSE and MAE that is less sensitive to outliers.

OPTIMIZATION METHODS (OPTIMIZERS)

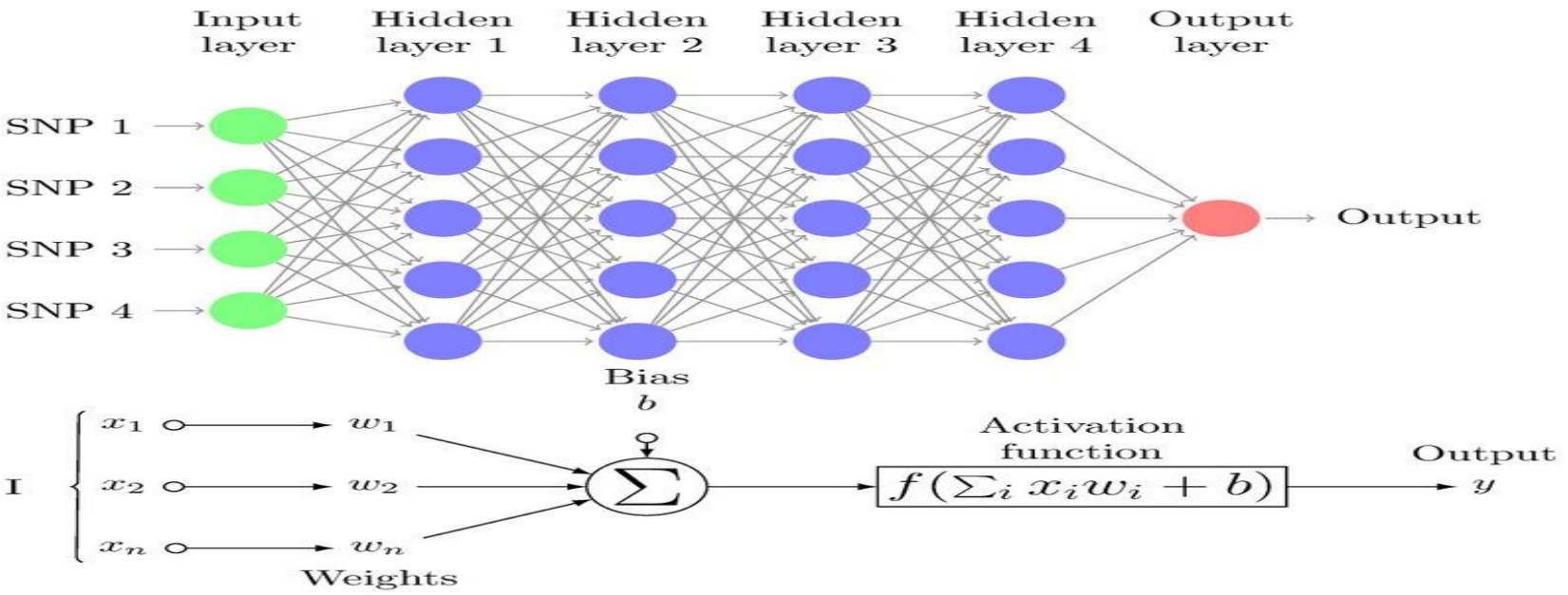
Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓


$x_0 = 1$
x_1
w_1
w_0
w_2
x_2
w_m
x_m
Σ
$z = \sum_{i=0}^{m} w_i x_i$
$o = \begin{cases} 1 & z \geq 0 \\ 0 & otherwise \end{cases}$

PERCEPTRON



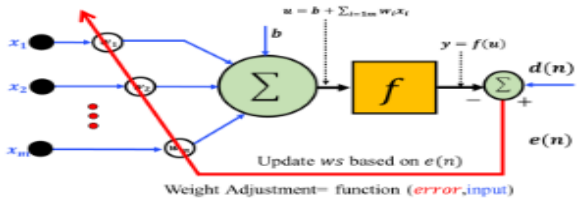
SINGLE LAYER PERCEPTRON

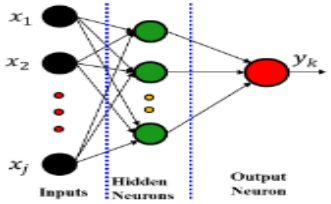


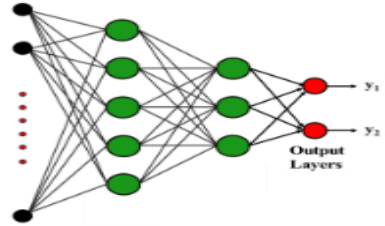
MULTI LAYER PERCEPTRON

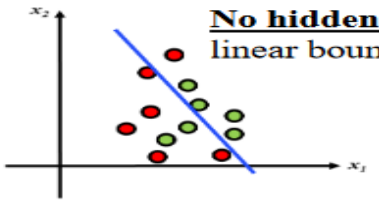
HOW TO DESIGN A NETWORK ?

- To design an ANN for a specific task, there are many design decisions:
  - Depth of the network (**i.e., the # of hidden layers**)
  - Width of the hidden layers (**i.e., the # of units per hidden layer**)
  - Type of *activation function* (nonlinearity)
  - Form of *objective function*

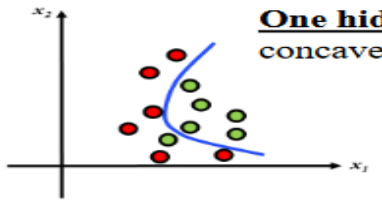








No hidden layer: linear boundary

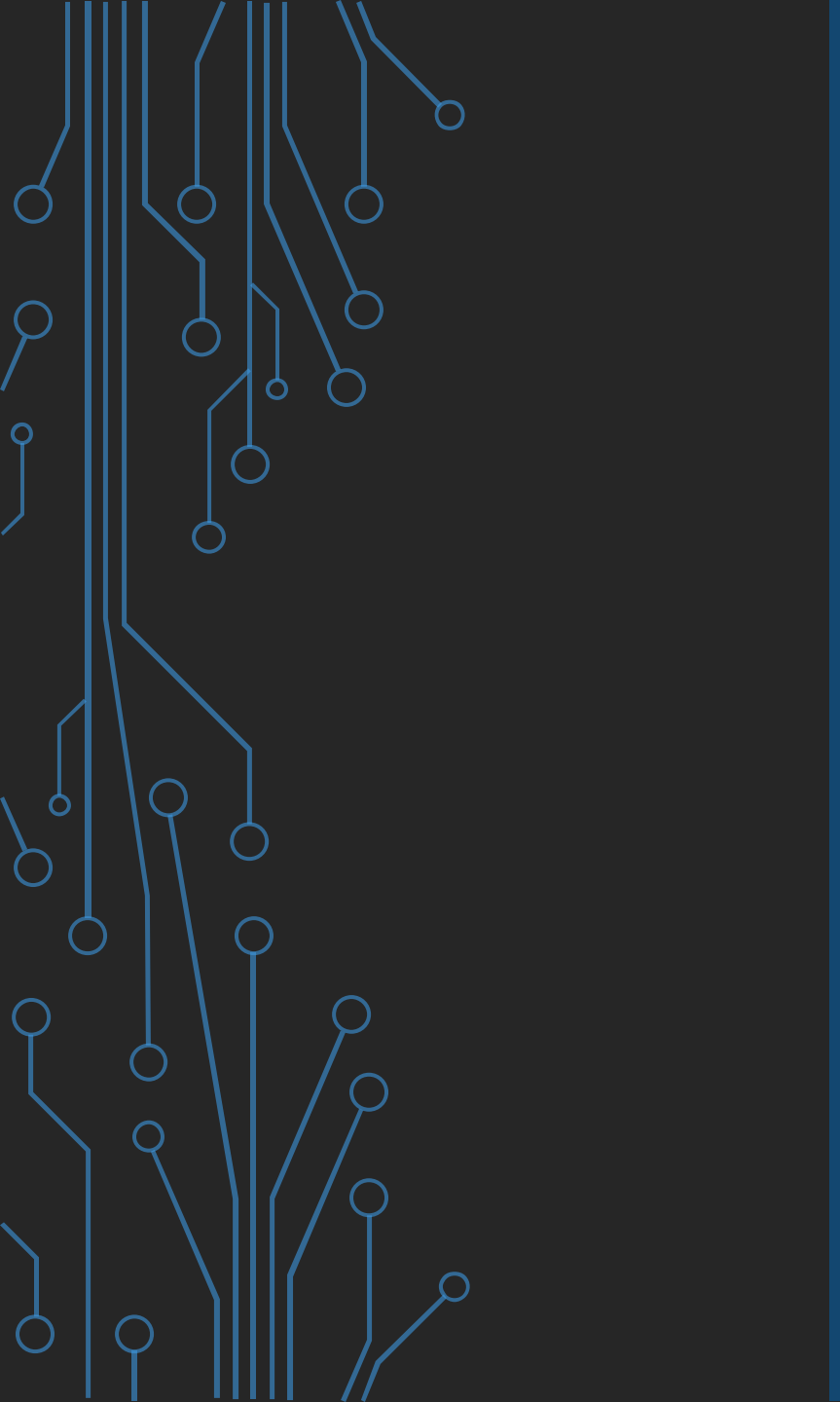


One hidden layer: concave boundary

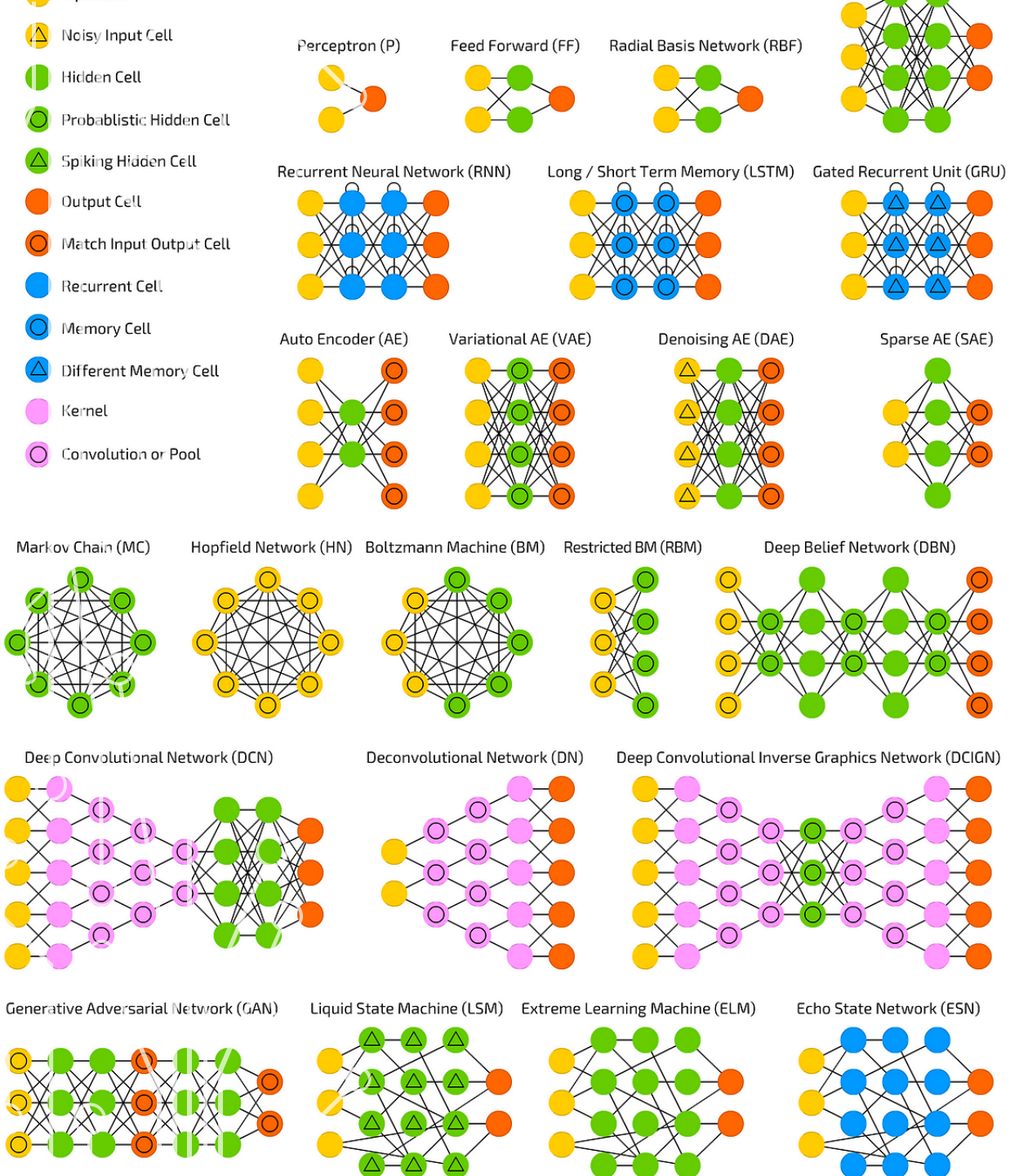
More hidden layer: More convex boundary

TIME FOR PRACTICALITY (10 MINUTES)

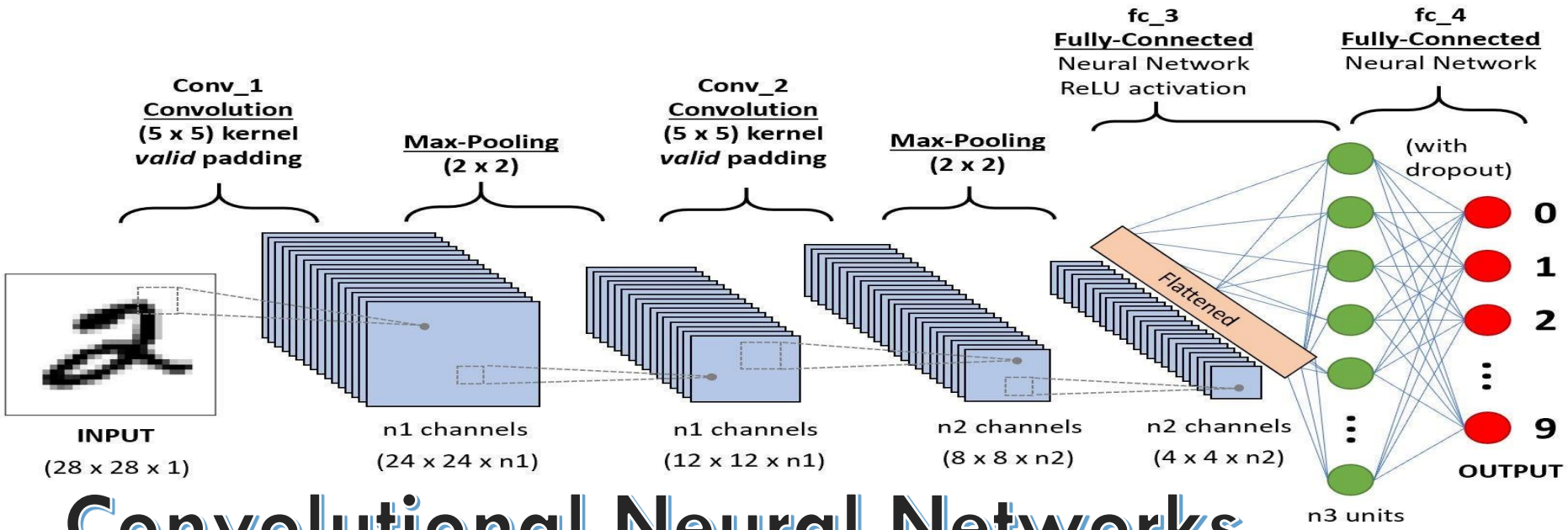
BREAK (10 MINUTES)



ARCHITECTURES



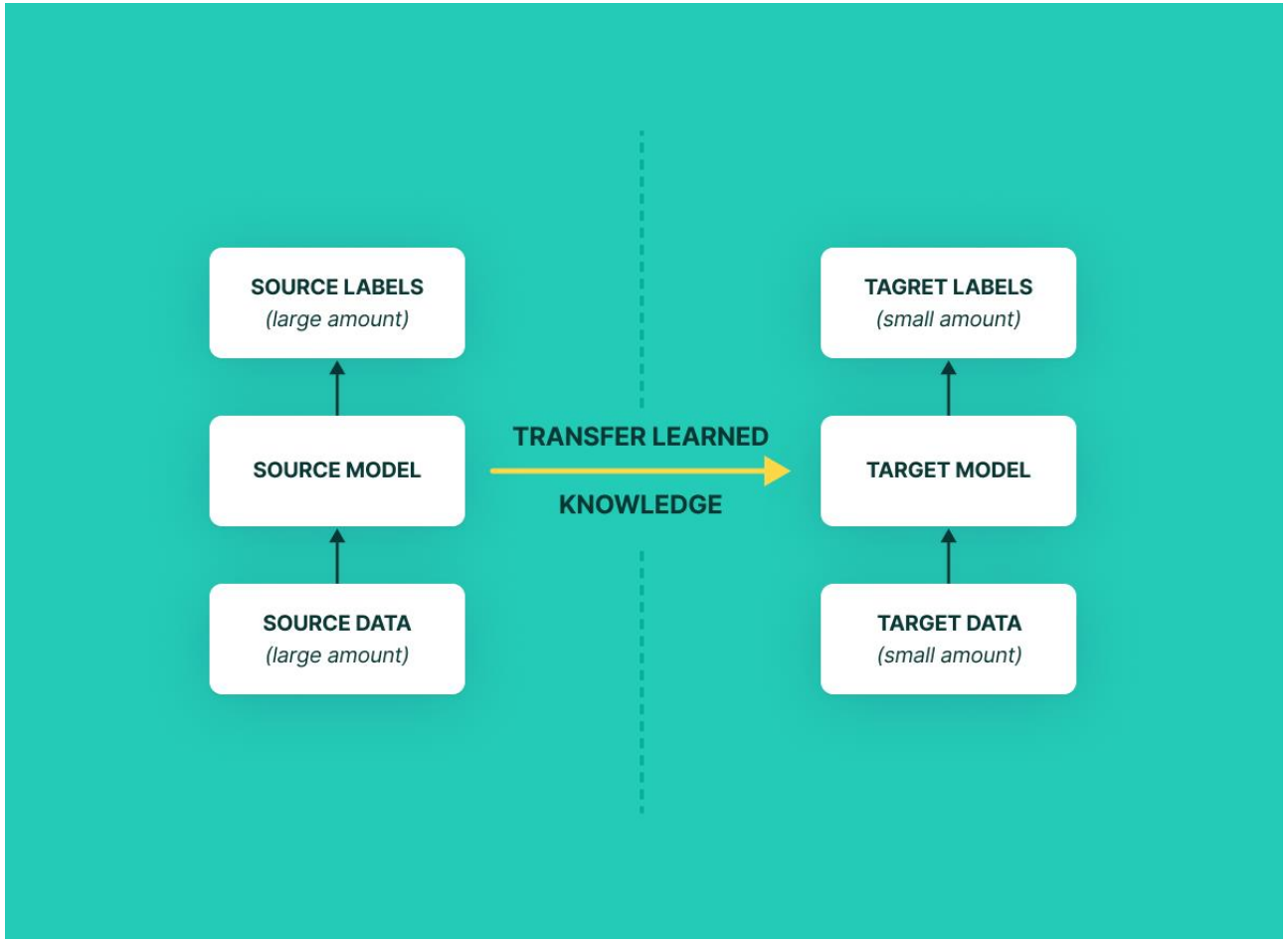
MANY ARCHITECTURES



Convolutional Neural Networks

BIGGER STRUCTURES

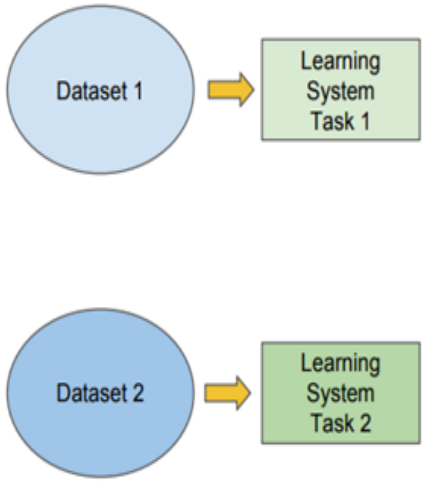
TRANSFER LEARNING



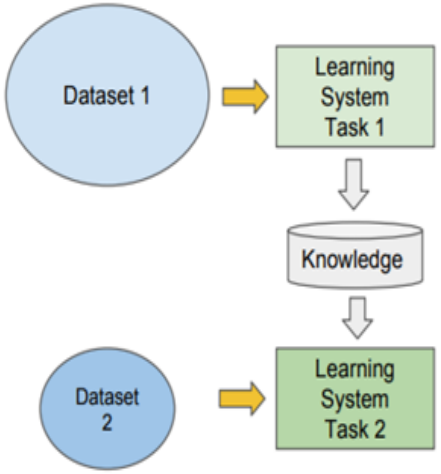
TRADITIONAL LEARNING VS. TRANSFER LEARNING

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data

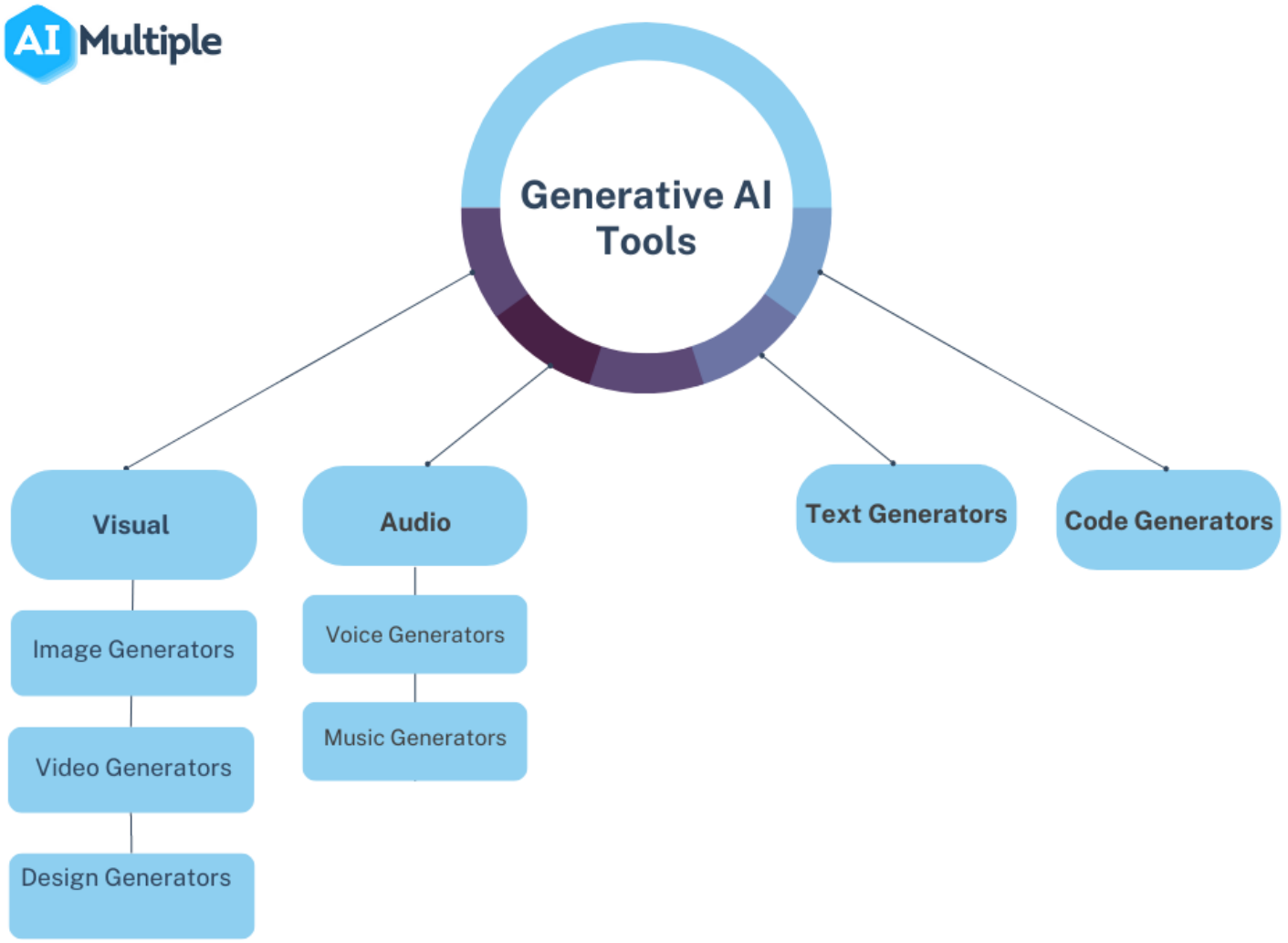


GENERATIVE AI

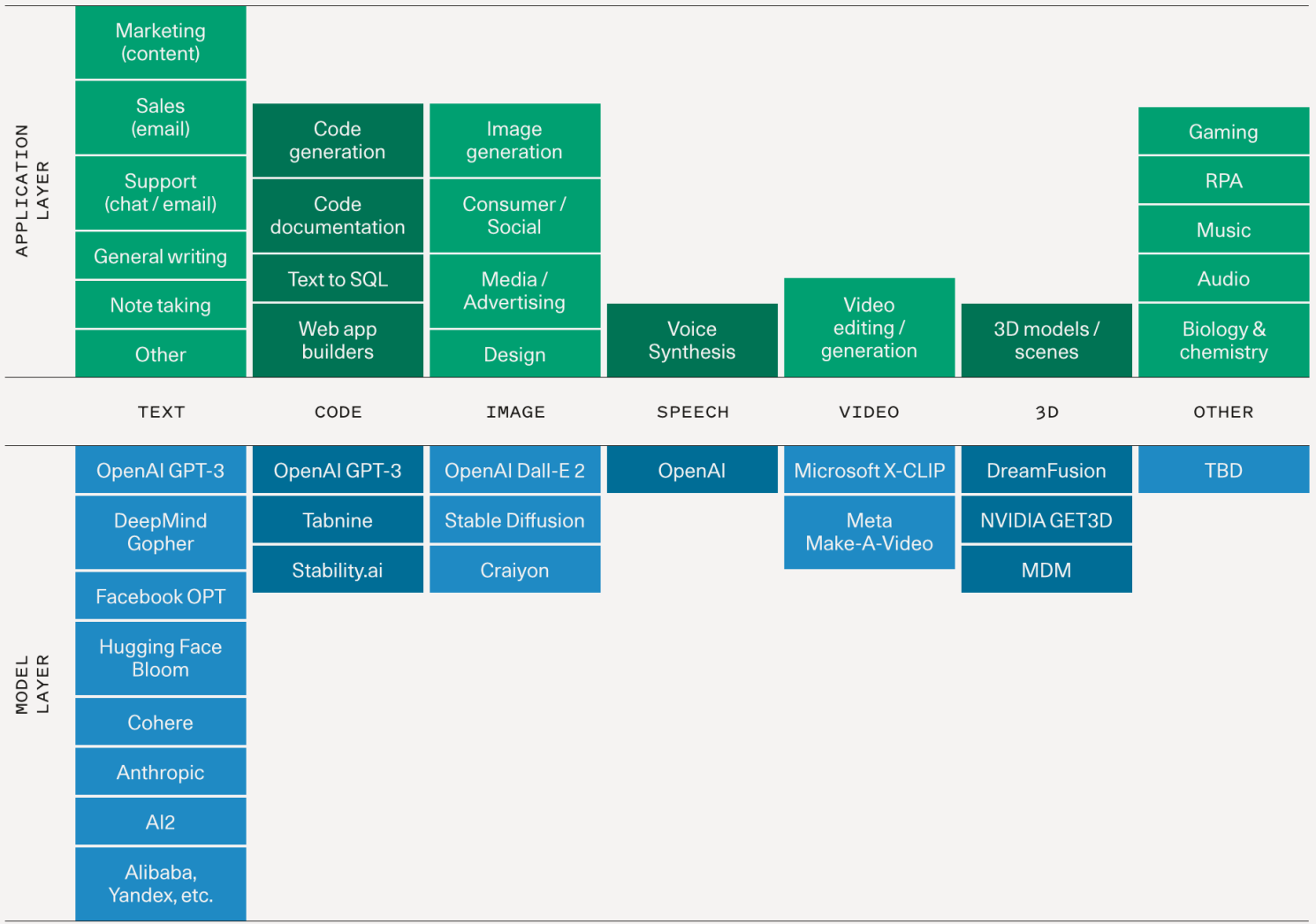
WHAT IS GENERATIVE AI ?

- **Generative AI**, short for **Generative Artificial Intelligence**, refers to a field of artificial intelligence that focuses on creating models and algorithms capable of generating new, original data that resembles a given input dataset.

- These models aim to capture and replicate patterns, styles, and characteristics present in the training data to produce new data samples that are similar in nature.


AI Multiple
Generative AI Tools
Visual
Image Generators
Video Generators
Design Generators
Audio
Voice Generators
Music Generators
Text Generators
Code Generators

The Generative AI Application Landscape



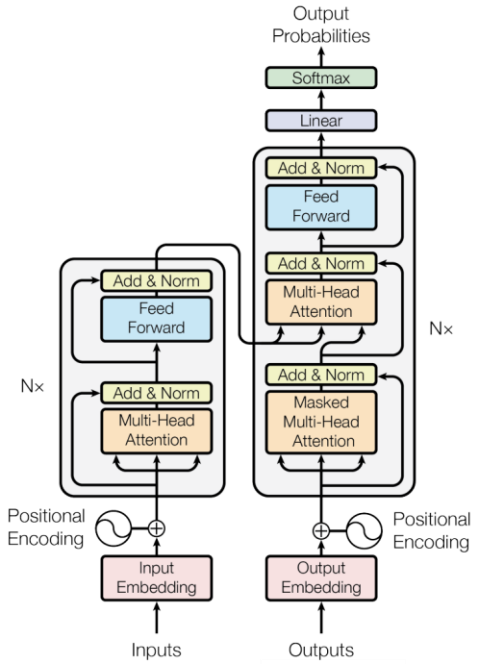
DEEP LEARNING MODELS USED IN GENERATIVE AI

1. **Generative Adversarial Networks (GANs)**: GANs consist of two neural networks, a generator and a discriminator, that compete against each other. The generator creates synthetic data, while the discriminator tries to distinguish between real and fake data. Over time, the generator becomes more skilled at generating realistic data as it learns from the feedback provided by the discriminator.
2. **Variational Autoencoders (VAEs)**: VAEs are neural networks designed for encoding and decoding data. They learn a probabilistic model of the data's distribution and can generate new samples by sampling from this distribution. VAEs are commonly used for generating images and other complex data types.
3. **Autoregressive Models**: These models generate data one element at a time, often conditioned on previous elements. They are commonly used in generating sequences, such as text and music.
4. **Transformer Models**: Transformers, initially designed for language processing tasks, have also been adapted for generative purposes. They can generate text, music, and even images by conditioning on a given prompt.

TRANSFORMER

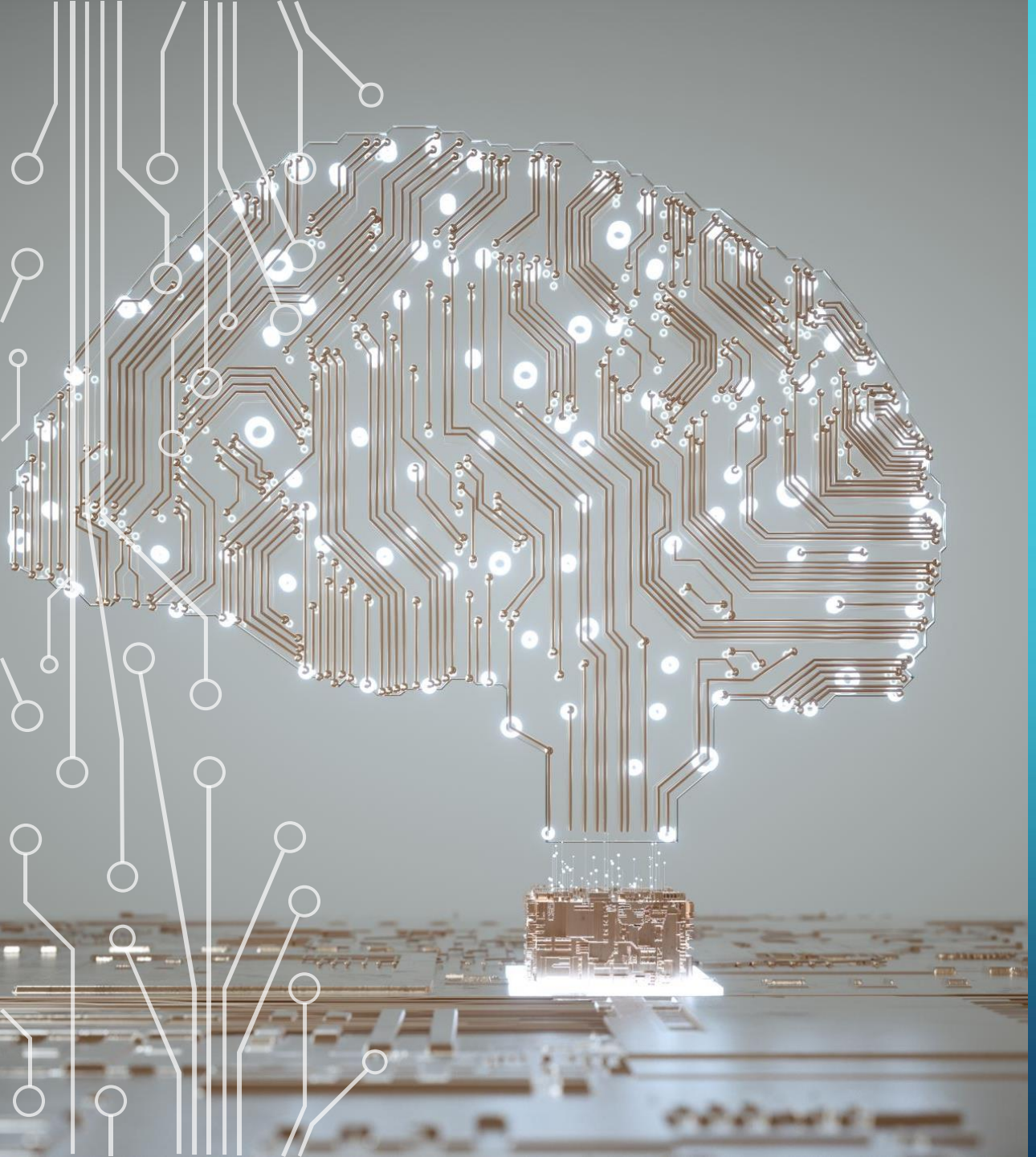
BERT

Encoder



GPT

Decoder



AI IS THE FUTURE

QUESTIONS

Artificial Intelligence

Machine Learning

Deep Learning

MLP

CNN

GAN

RBFN

RNN

LSTM

Autoencoders

Rule Based Systems

Support Vector Machines

Gaussian Process Regression

Random Forest

Linear Regression

Logistic Regression

K-Means Clustering

Cognitive Modeling

Planning

Game Playing

Search Algorithms

Knowledge Representation and Reasoning

Propositional Calculus


THE END