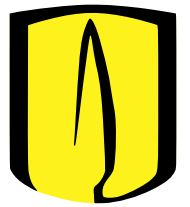
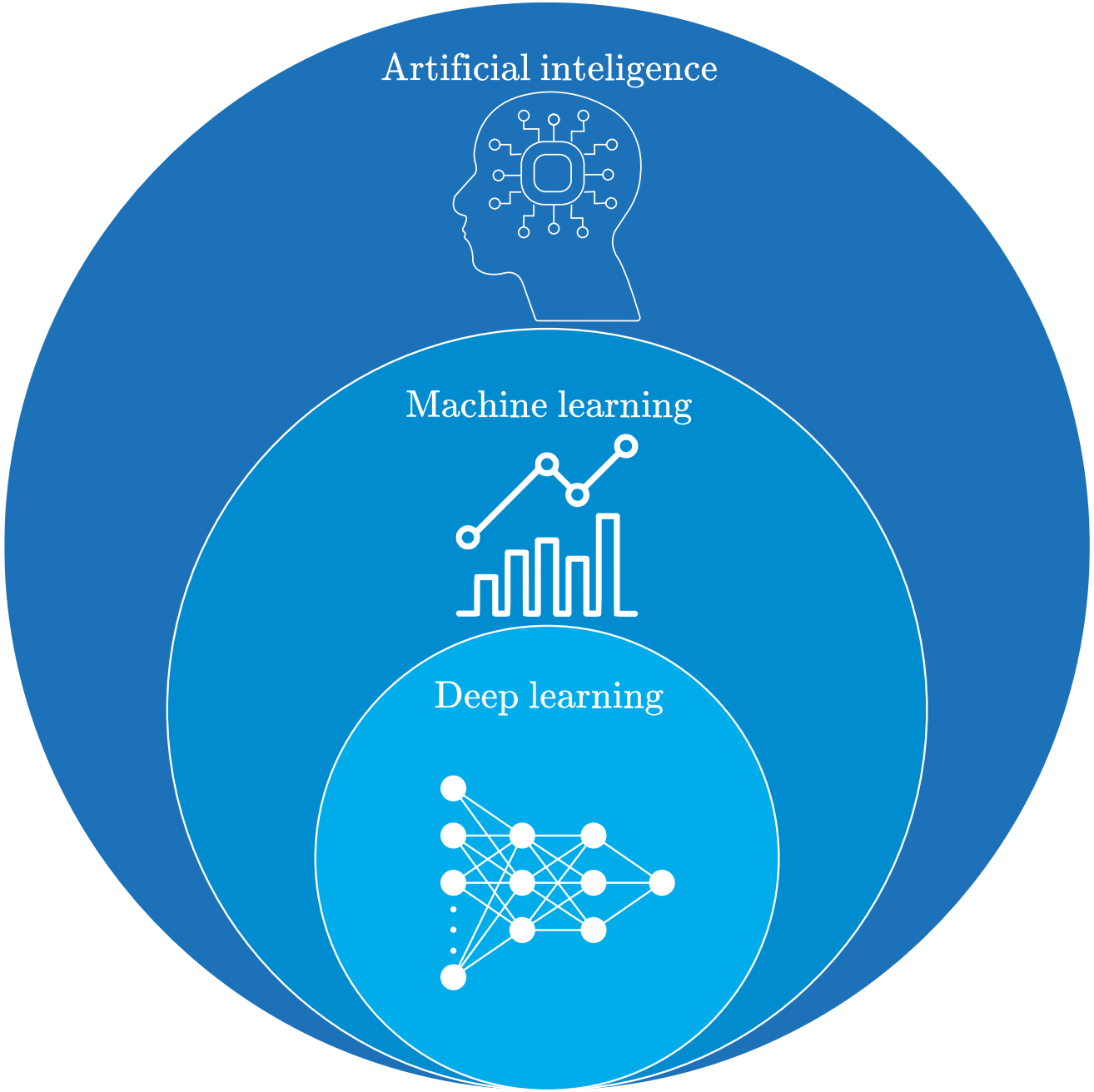


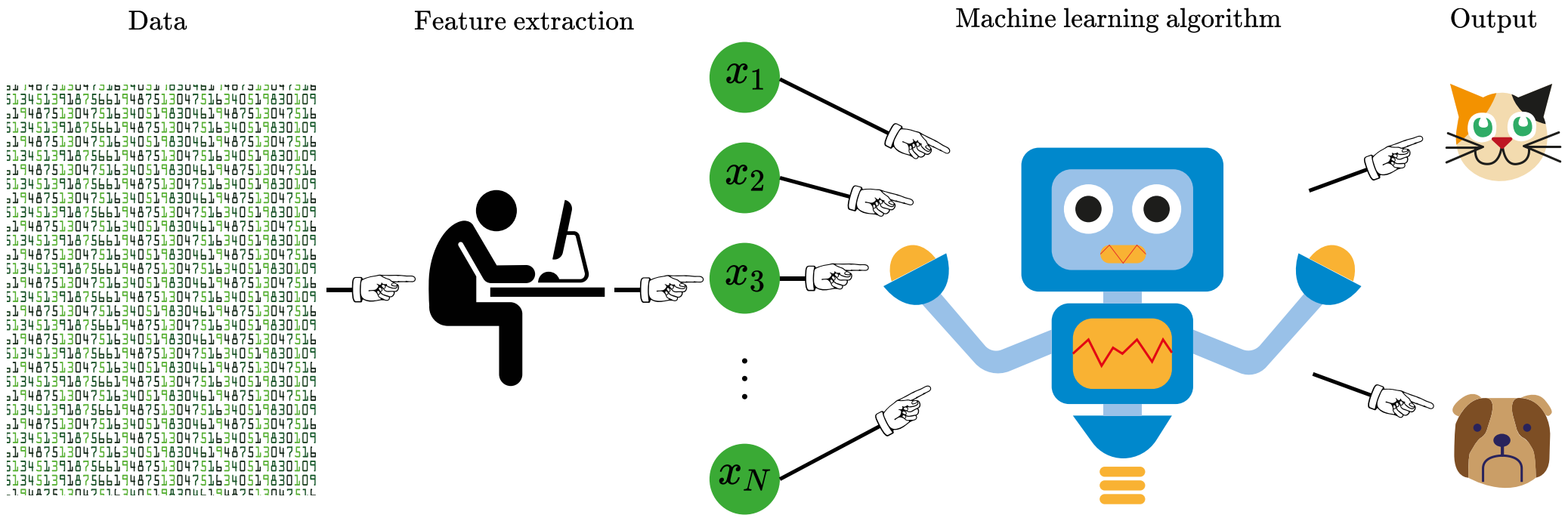
Machine Learning Methods for High Energy Physics

Manuel Fernando Sánchez Alarcón

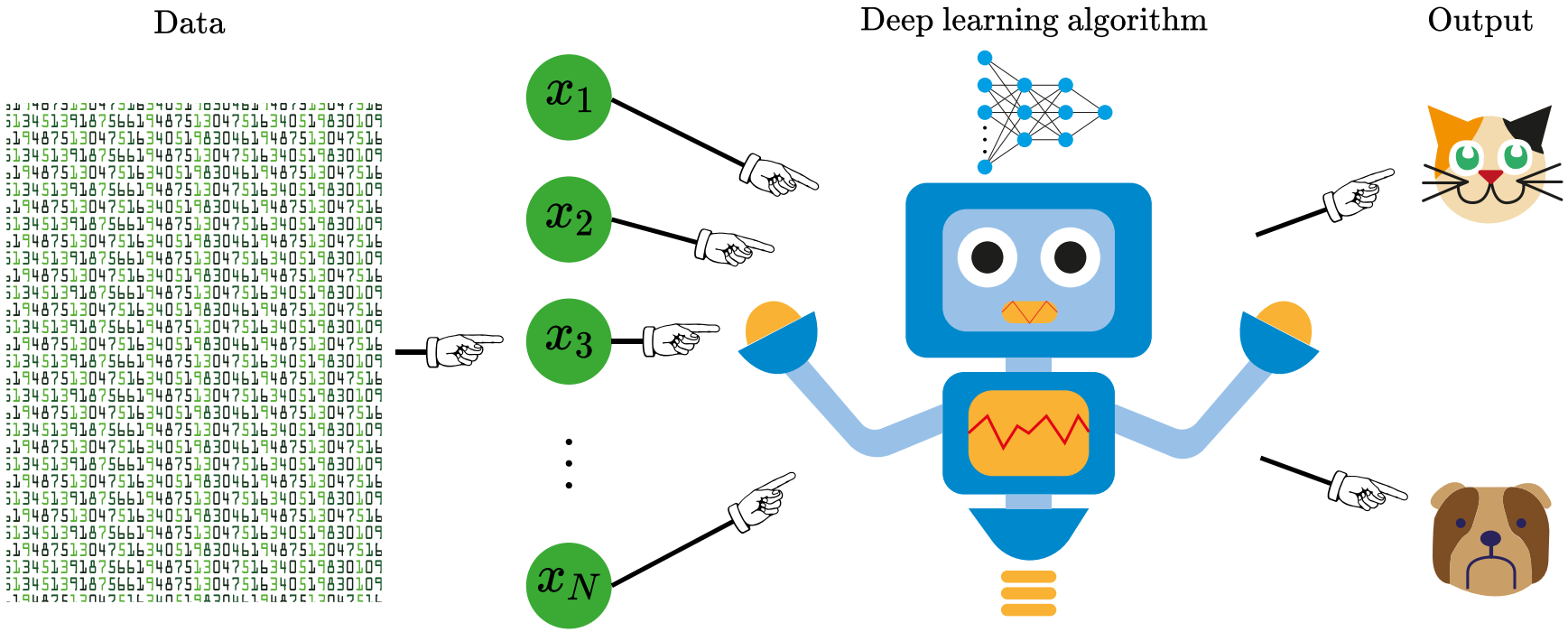
This is just a brief introduction!


Artificial inteligence
Machine learning
Deep learning

Machine learning vs Deep learning

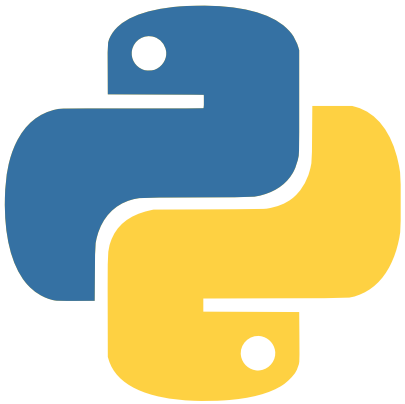


Machine learning vs Deep learning



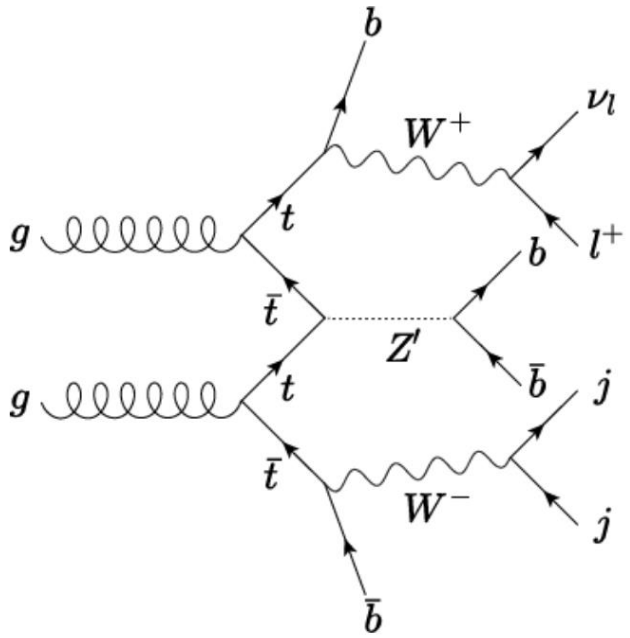
Choose Your Weapons

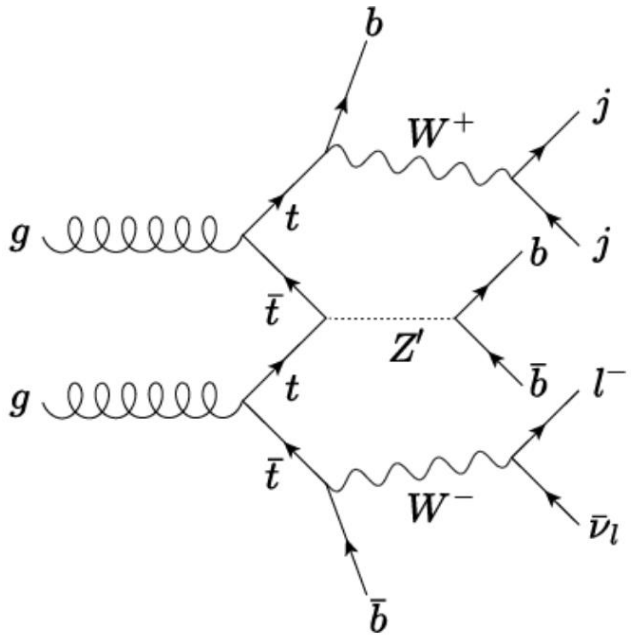
	x_1	x_2	x_3	x_4	Class/Output
1	355.4	344.7	432.5	344.7	1
2	342.1	543.2	355.4	543.2	1
3	344.7	355.4	543.2	342.1	1
4	543.2	432.5	344.7	543.2	0
5	432.5	543.2	342.1	355.4	0
6	543.2	342.1	543.2	432.5	0



Context of Our Dataset

Production of Z' boson through top-fusion (semileptonic channel)

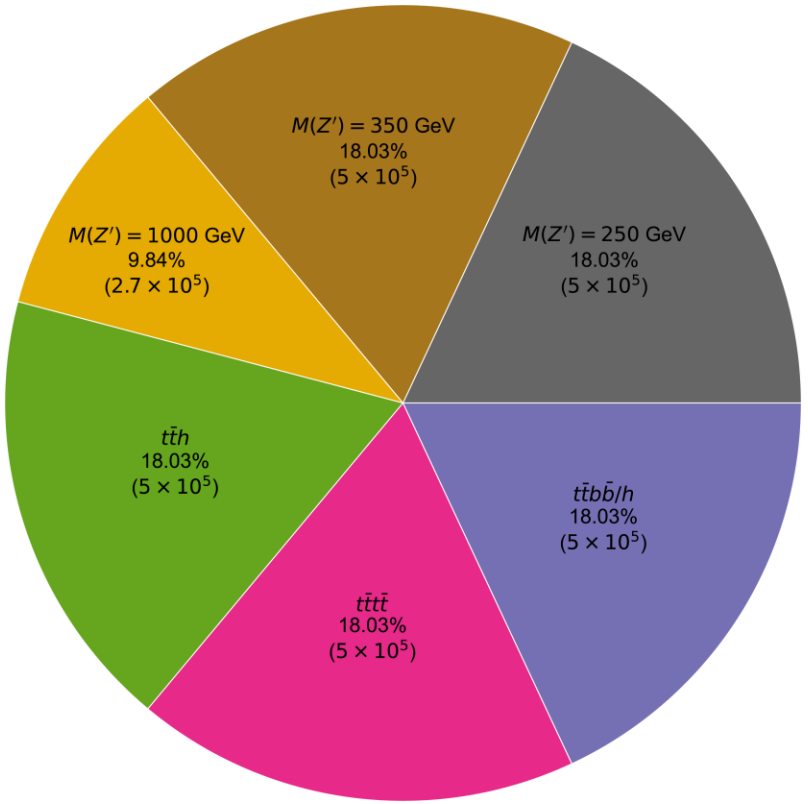




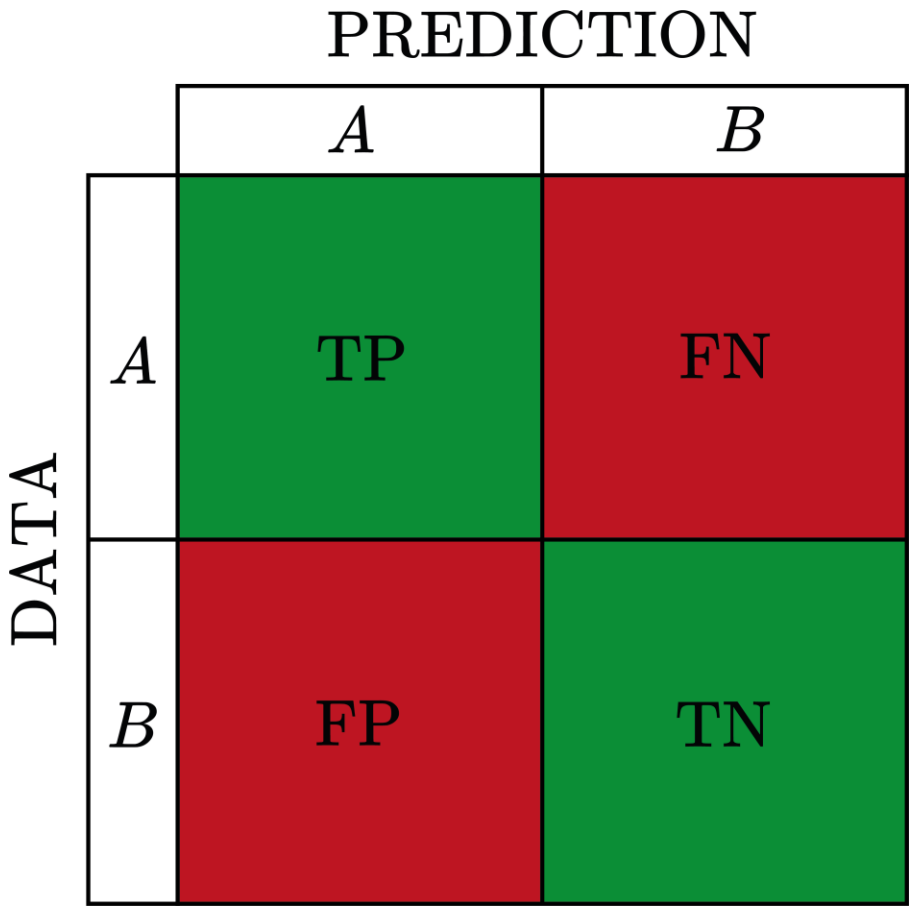
Context of Our Dataset

Representative Variables	
$p_T(b_i)$	$\Delta\eta(b_i, b_j)$
$p_T(j_k)$	$\Delta\phi(b_i, b_j)$
$p_T(l)$	$\Delta\phi(b_i, l)$
$M(b_i, b_j)$	$\Delta R(b_i, b_j)$
$M(j_1, j_2)$	$\Delta R(j_1, j_2)$
$M_T(b_i + l + \text{MET})$	$\Delta R(b_i, l)$
$M_T(l, \text{MET})$	MET

$i, j = 1, 2, 3, 4$ $k = 1, 2$



Model Evaluation (Confusion Matrix)

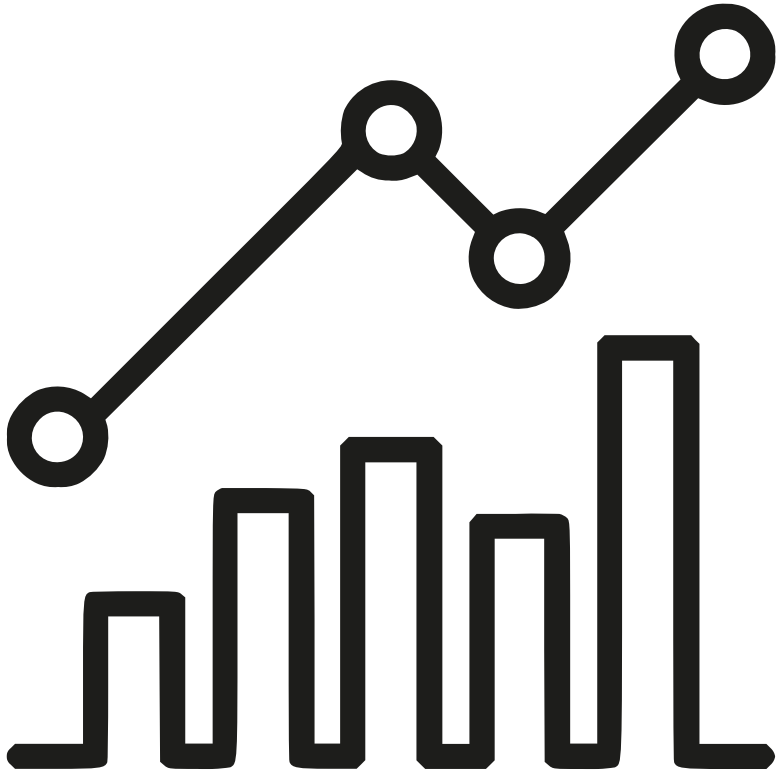


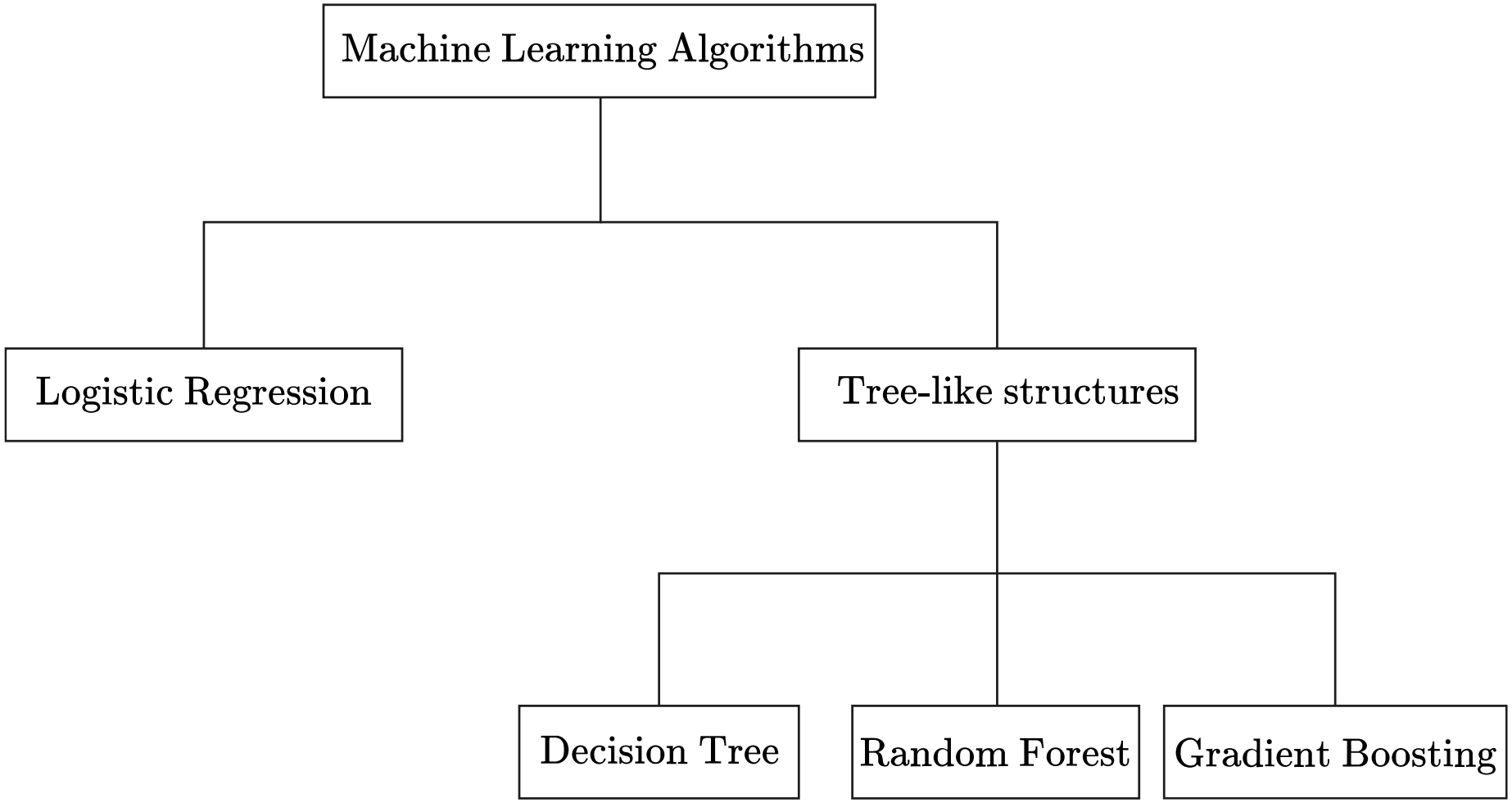
$$\text{accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of all predictions}}$$

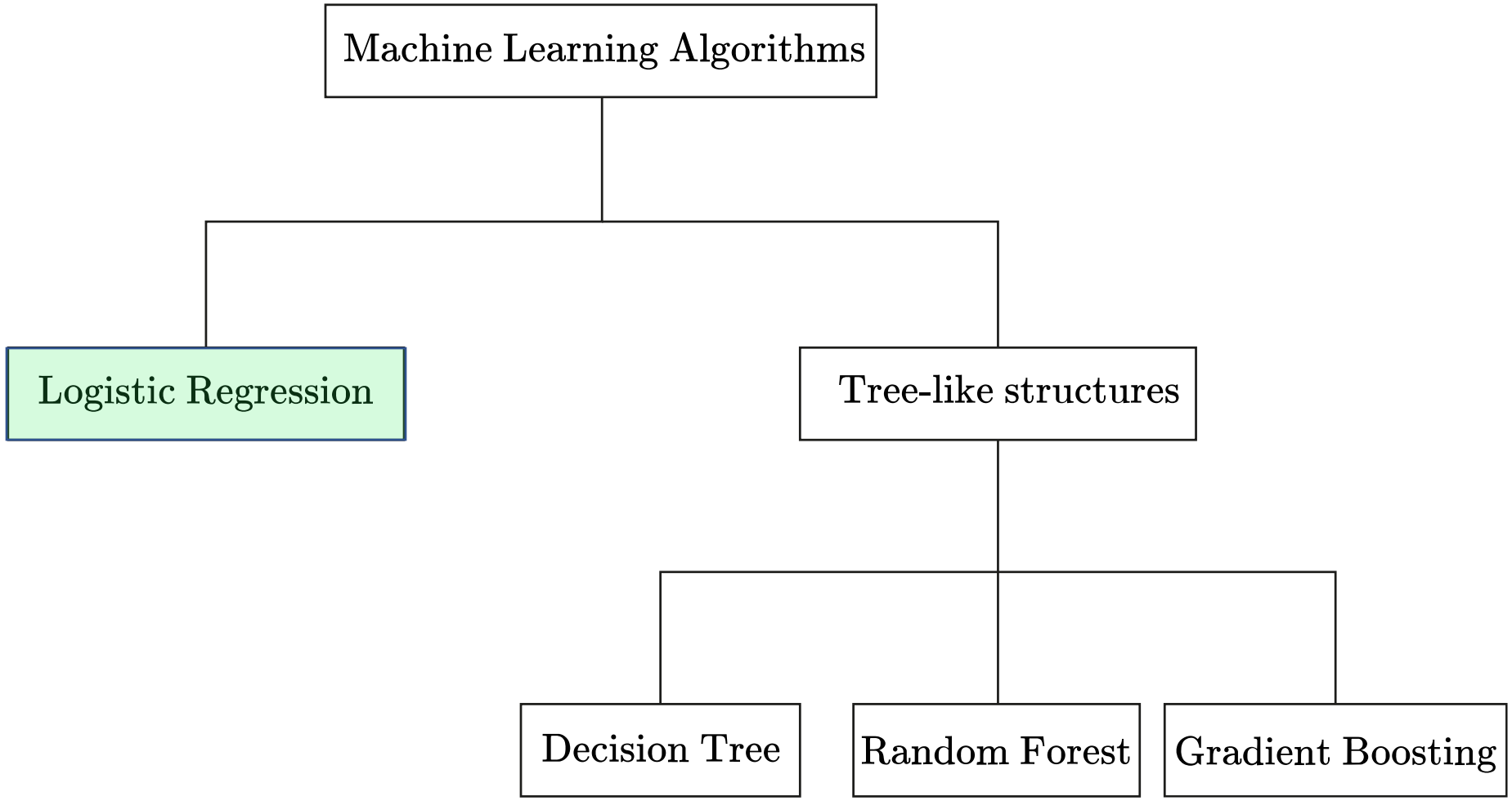
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

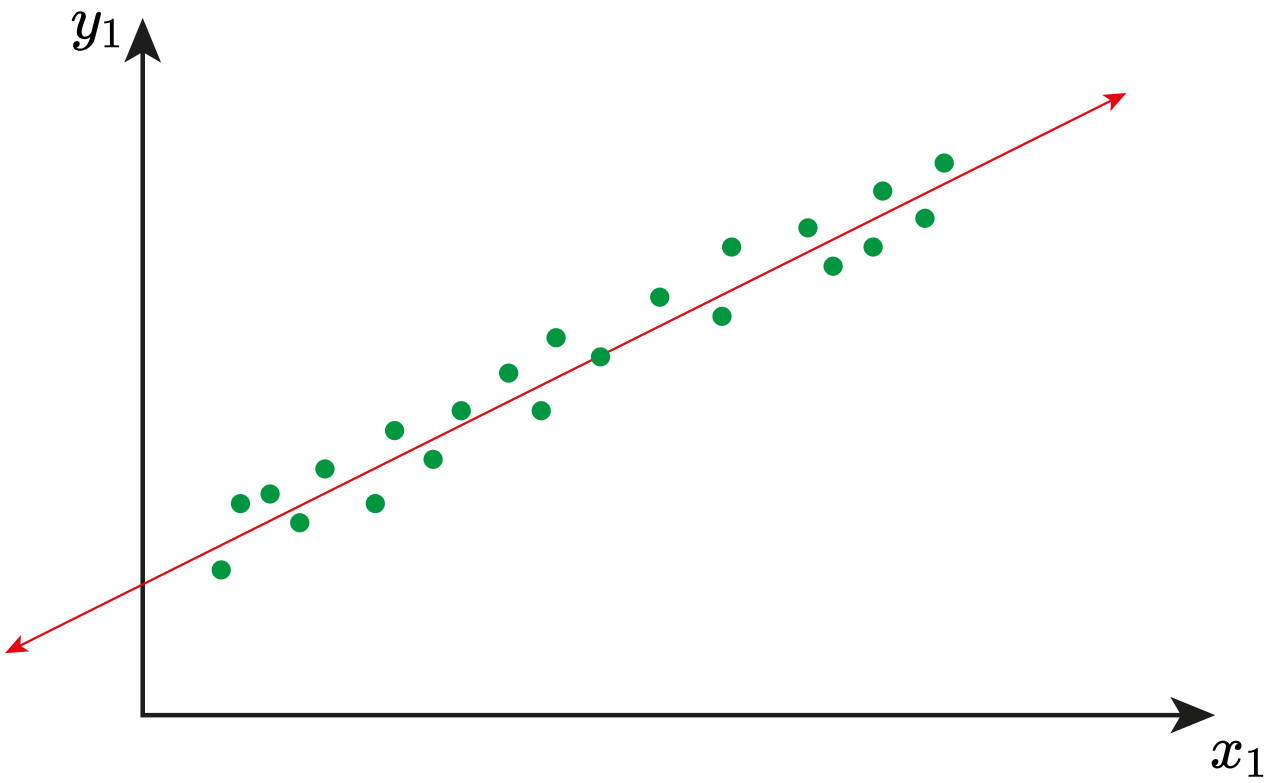
Machine Learning




Machine Learning Algorithms
Logistic Regression
Tree-like structures
Decision Tree
Random Forest
Gradient Boosting

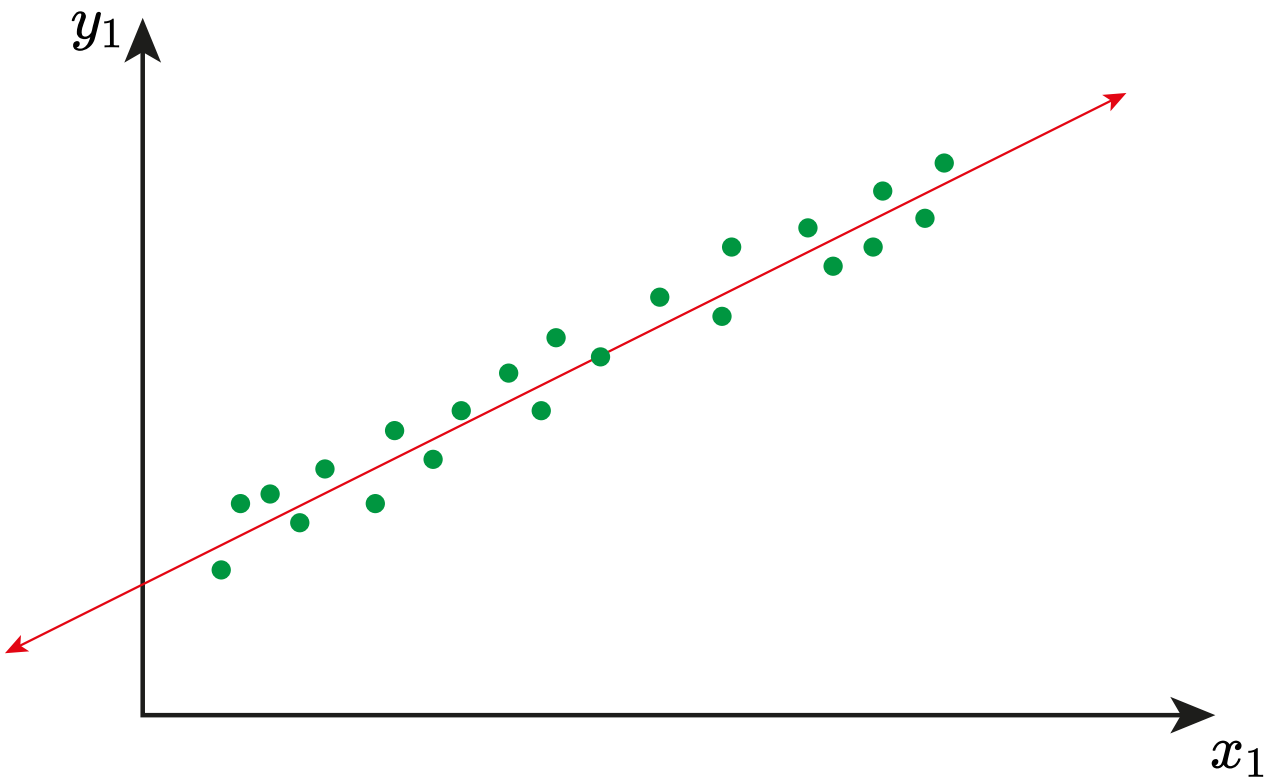

Machine Learning Algorithms
Logistic Regression
Tree-like structures
Decision Tree
Random Forest
Gradient Boosting

Linear Regression



Predict continuous dependent variable using a given set of independent variables

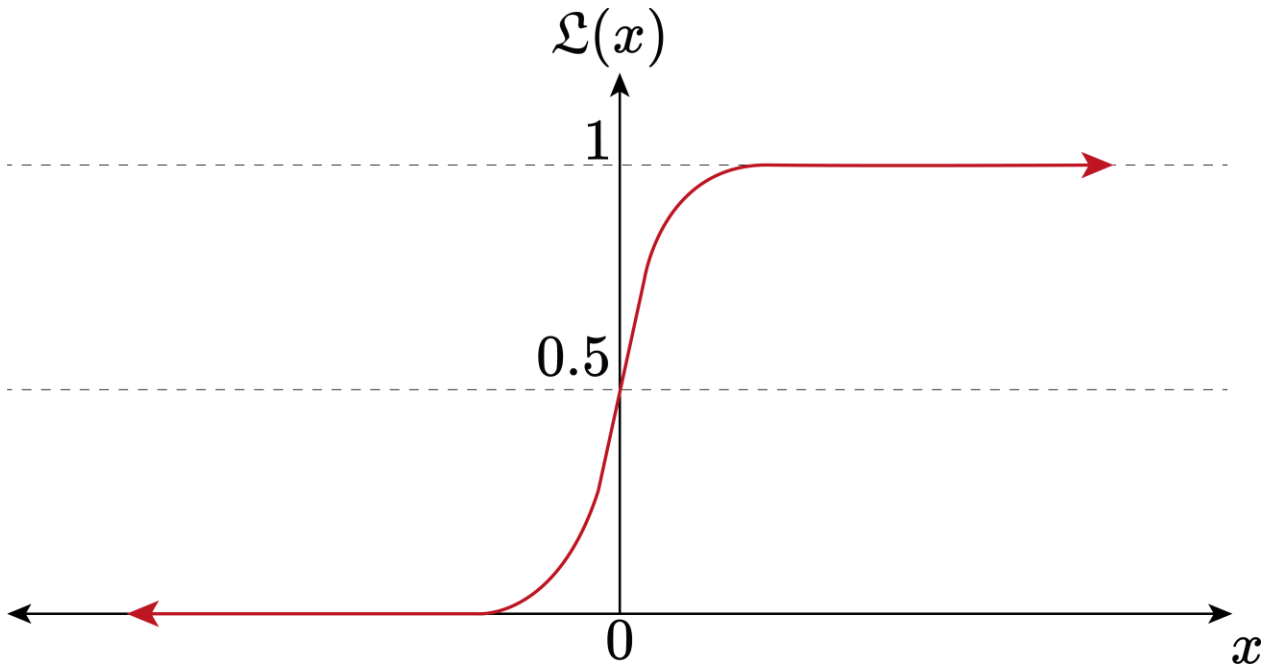
Linear Regression



Predict continuous dependent variable using a given set of independent variables

What we need for classification problems?

Logistic Regression

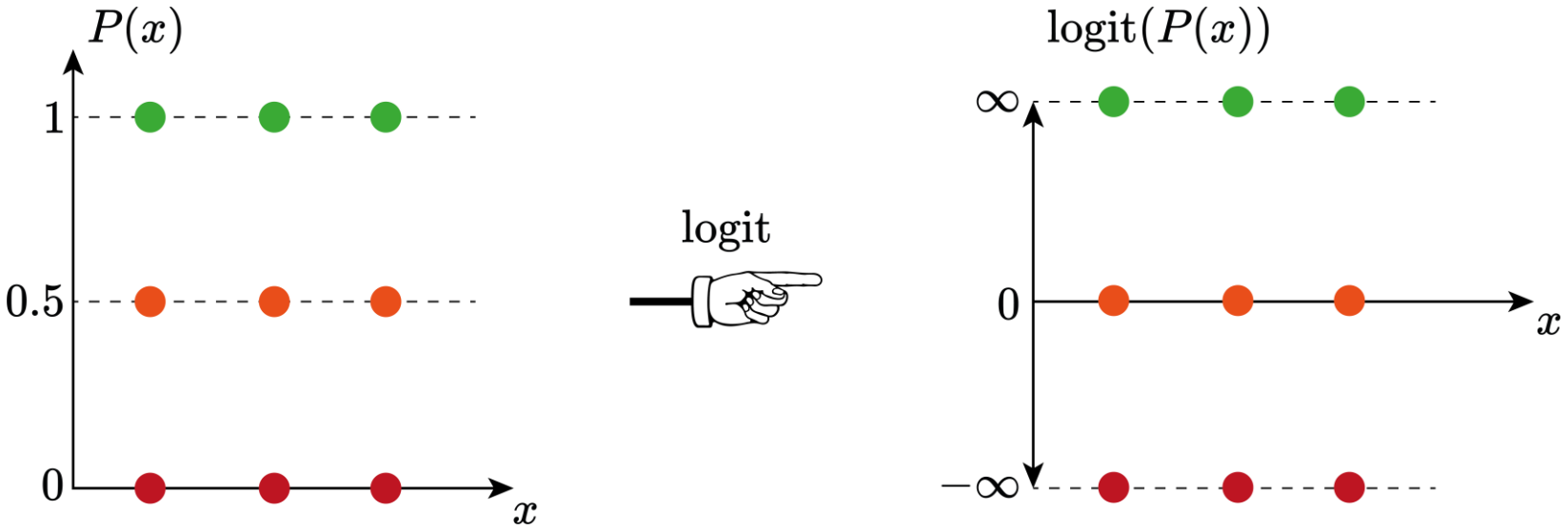


$$\mathfrak{L}(x) = \frac{1}{1 + e^{-x}}$$

$$\text{logit}(P) = \ln\left(\frac{P(x)}{1 - P(x)}\right)$$

$$\text{logit}(P) = mx + b$$

Logistic Regression

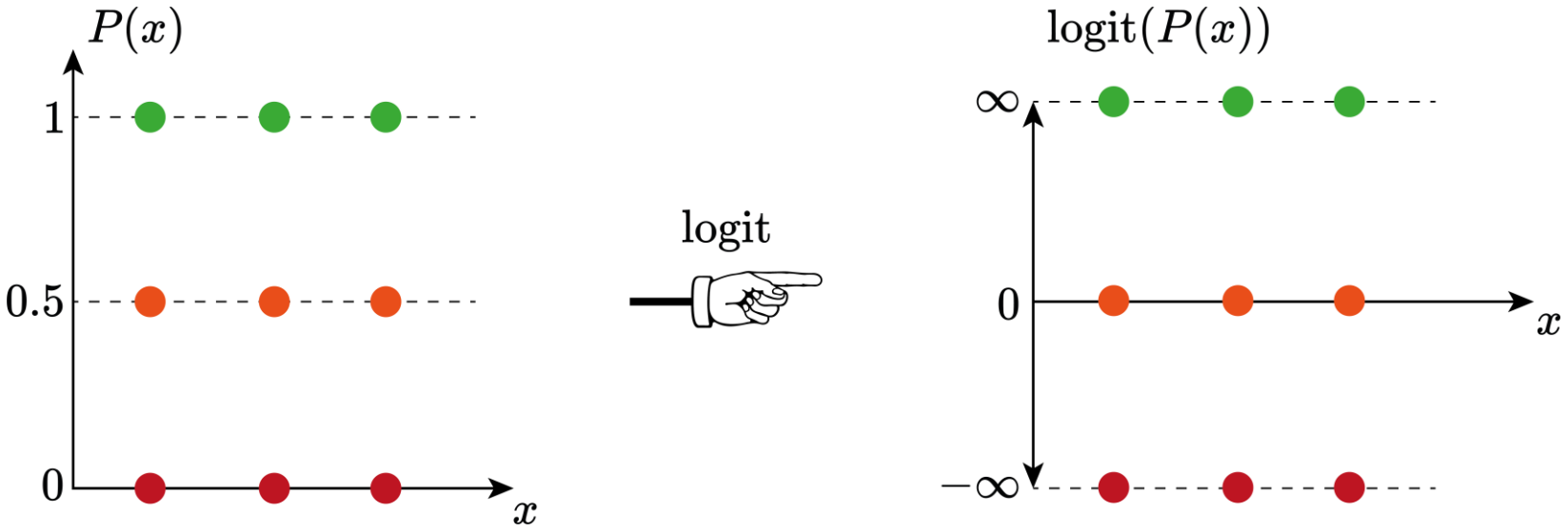


$$\mathfrak{L}(x) = \frac{1}{1 + e^{-x}}$$

$$\text{logit}(P) = \ln\left(\frac{P(x)}{1 - P(x)}\right)$$

$$\text{logit}(P) = mx + b$$

Logistic Regression



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

$$\text{logit}(P) = \ln\left(\frac{P(x)}{1 - P(x)}\right)$$

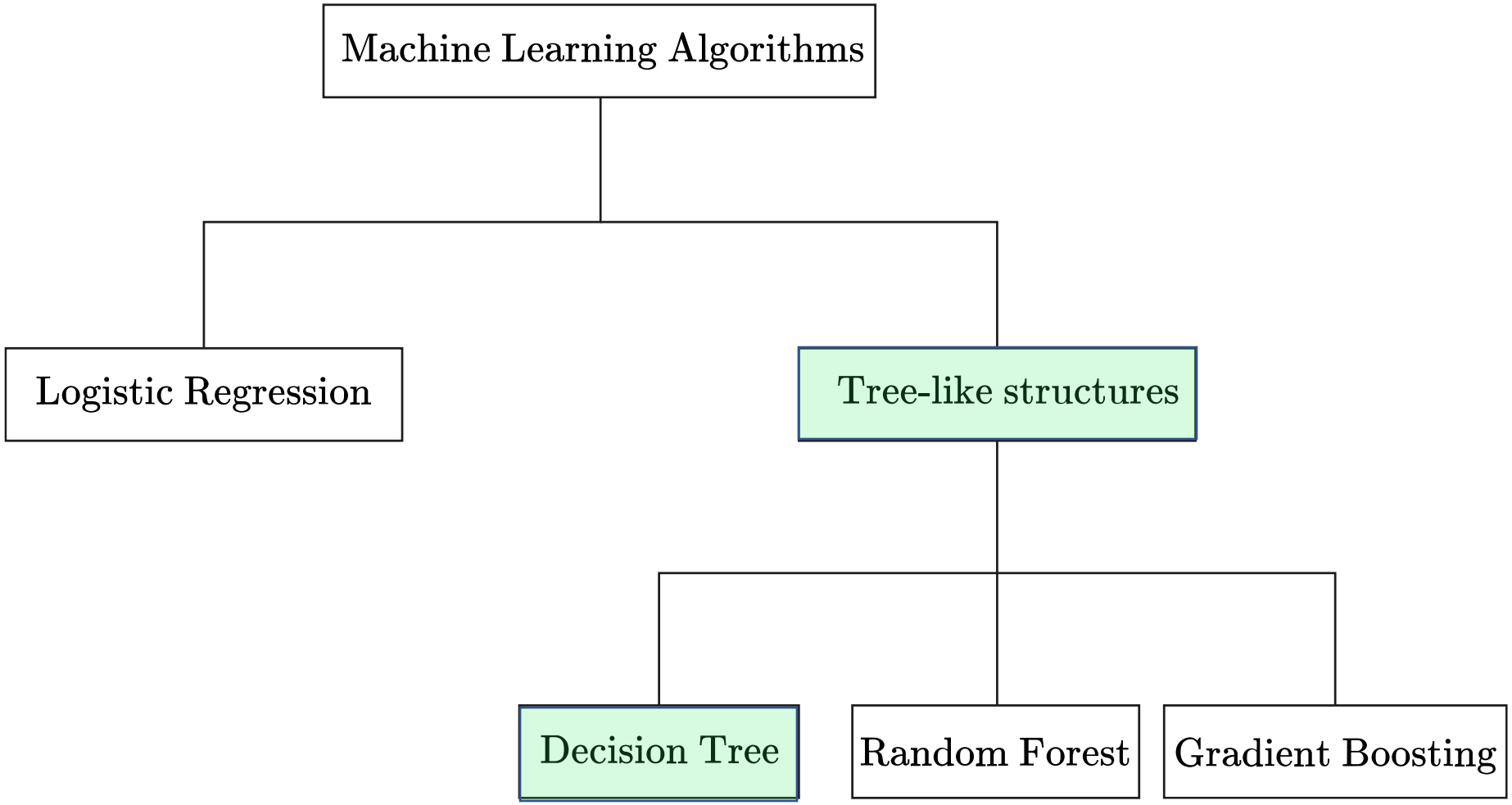
$$\text{logit}(P) = mx + b$$

It can be used for Classification as well as for Regression problems, but mainly used for Classification problems

Let’s see an example:

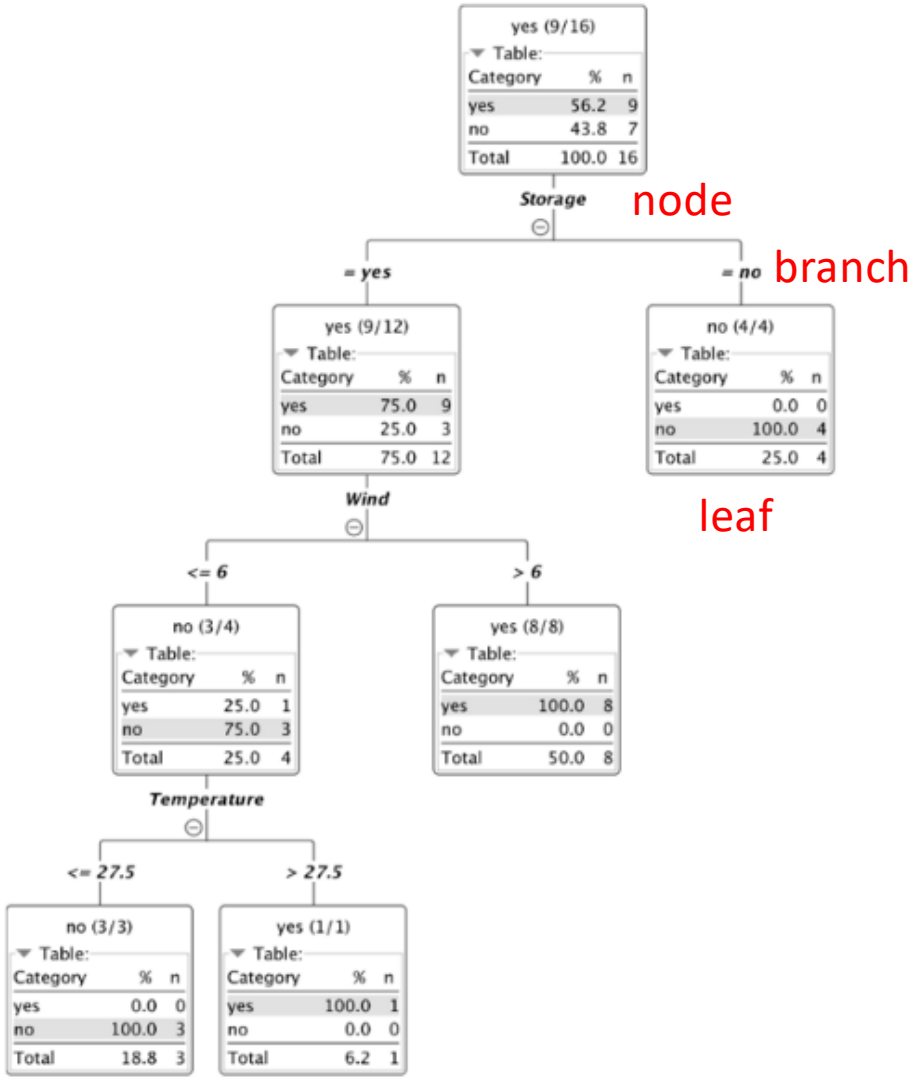
Code: https://github.com/Spoksonat/Particle_Physics_School/blob/main/logistic_regression.ipynb

Data: https://drive.google.com/drive/folders/1gsqEcoXQzrSyUrDCFkw3kkMzuYByp0jo?usp=sharing


Machine Learning Algorithms
Logistic Regression
Tree-like structures
Decision Tree
Random Forest
Gradient Boosting

Decision Trees

Outlook	Wind	Temp	Storage	Sailing
sunny	3	30	yes	yes
sunny	3	25	yes	no
rain	12	15	yes	yes
overcast	15	2	no	no
rain	16	25	yes	yes
sunny	14	18	yes	yes
rain	3	5	no	no
sunny	9	20	yes	yes
overcast	14	5	no	no
sunny	1	7	no	no
rain	4	25	yes	no
rain	14	24	yes	yes
sunny	11	20	yes	yes
sunny	2	18	yes	no
overcast	8	22	yes	yes
overcast	13	24	yes	yes



Flowchart-like structure made of nodes and branches

At each node, a split on the data is performed based on one of the input features, generating two or more branches as output

This continues until a node is generated where all or almost all of the data belong to the same class

Taken from: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

Quality Measures for Decision Trees

Metrics to measure the purity of a dataset

Entropy

Information gain

Gini index

Quality Measures for Decision Trees

Metrics to measure the purity of a dataset

Entropy

Information gain

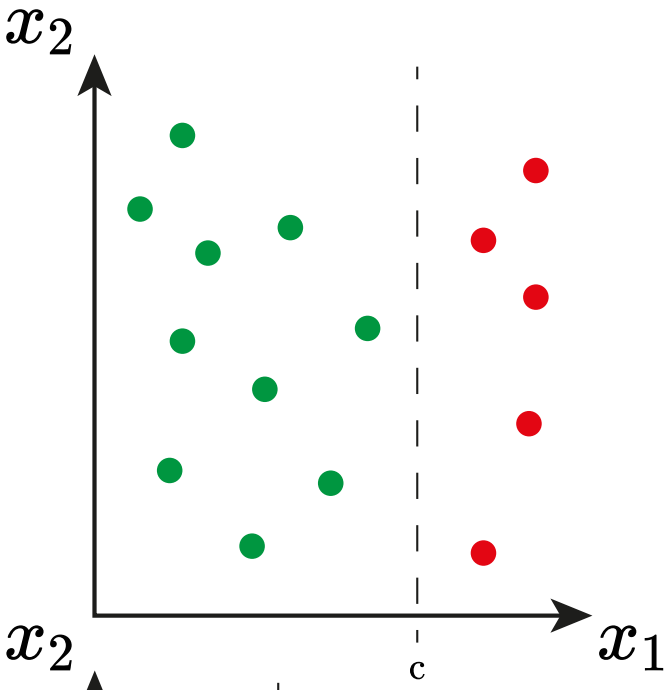
Gini index

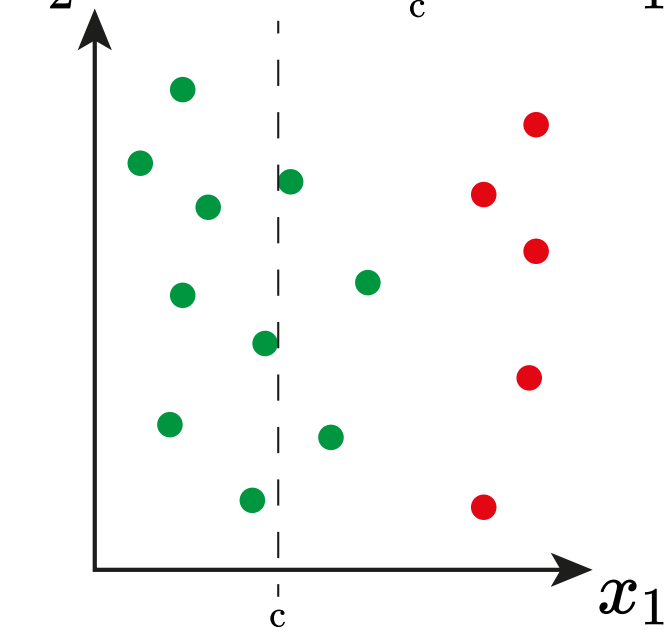
Entropy

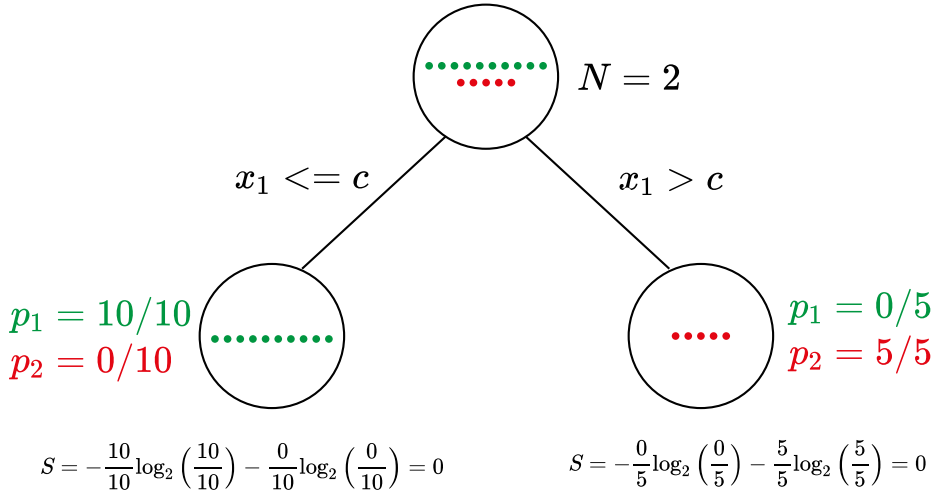
Measure of purity or information

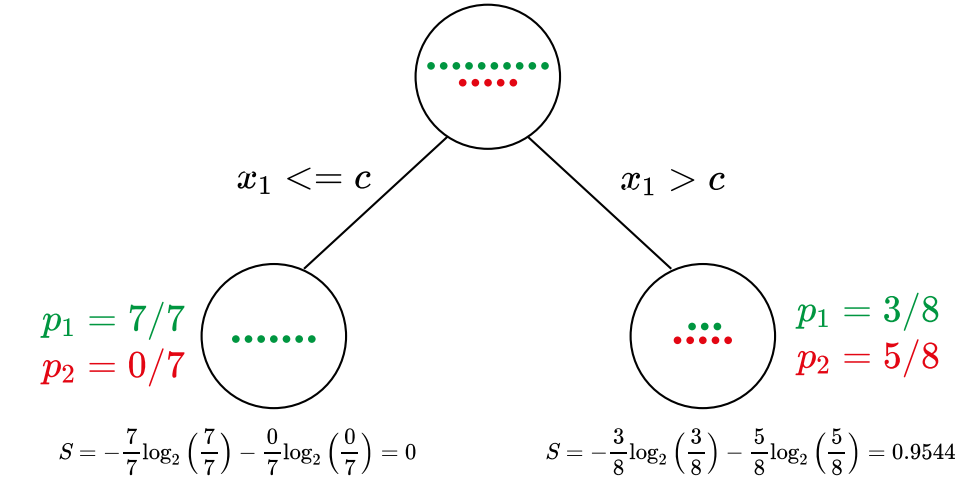
$$S(p) = -\sum_{i=0}^{N} p_i \log_2(p_i)$$

$S(p)$: Entropy of dataset p
N : Number of classes
p_i : Frequency of class i in dataset p


x_2
x_1
c


x_2
x_1
c


$N = 2$
$x_1 <= c$
$x_1 > c$
$p_1 = 10/10$
$p_2 = 0/10$
$p_1 = 0/5$
$p_2 = 5/5$
$S = -\frac{10}{10}\log_2\left(\frac{10}{10}\right) - \frac{0}{10}\log_2\left(\frac{0}{10}\right) = 0$
$S = -\frac{0}{5}\log_2\left(\frac{0}{5}\right) - \frac{5}{5}\log_2\left(\frac{5}{5}\right) = 0$


$x_1 <= c$
$x_1 > c$
$p_1 = 7/7$
$p_2 = 0/7$
$p_1 = 3/8$
$p_2 = 5/8$
$S = -\frac{7}{7}\log_2\left(\frac{7}{7}\right) - \frac{0}{7}\log_2\left(\frac{0}{7}\right) = 0$
$S = -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) = 0.9544$

Quality Measures for Decision Trees

Metrics to measure the purity of a dataset

Entropy

Information gain

Gini index

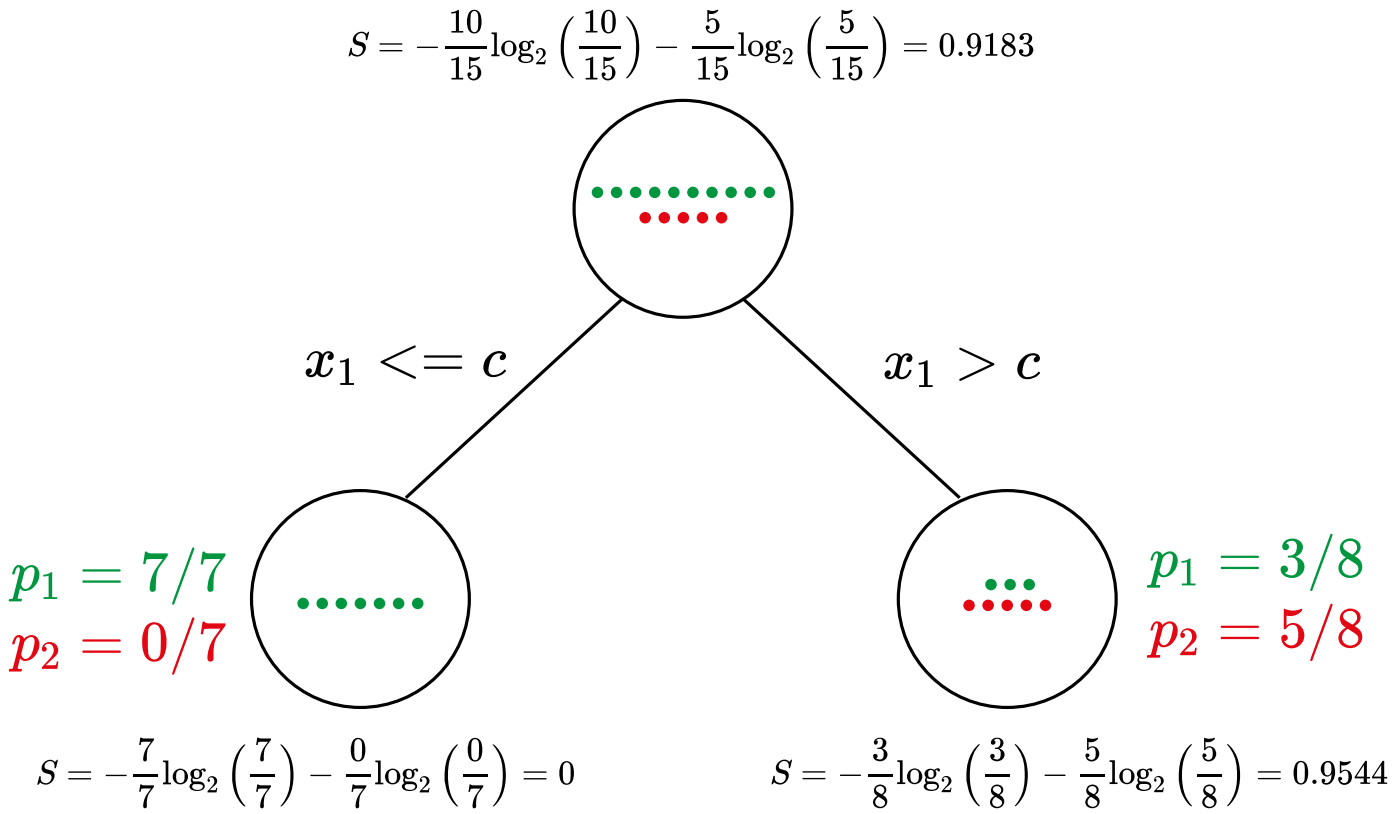
Information gain

Evaluate how good a feature is for splitting

$$\text{Information gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^{K} \text{Entropy}(j, \text{after})$$

$\text{Entropy}(\text{before})$: Entropy of data set before splitting
$\text{Entropy}(j, \text{after})$: Entropy of the j-th subset generated by the split
K = Number of subsets generated by the split

If the information gain is a positive number, this means that we move from a confused dataset to a number of purer subsets



$$\text{Information gain} = 0.9183 - (0 + 0.9544) = -0.0361$$

Quality Measures for Decision Trees

Metrics to measure the purity of a dataset

Entropy

Information gain

Gini index

Gini index

Another measure for purity (or actually impurity)

$$\text{Gini impurity} = 1 - \sum_{i=1}^{N} p_i^2$$

$$w_j = \frac{\text{Number of data points in subset } (j,\text{after})}{\text{Number of data points before the split}}$$

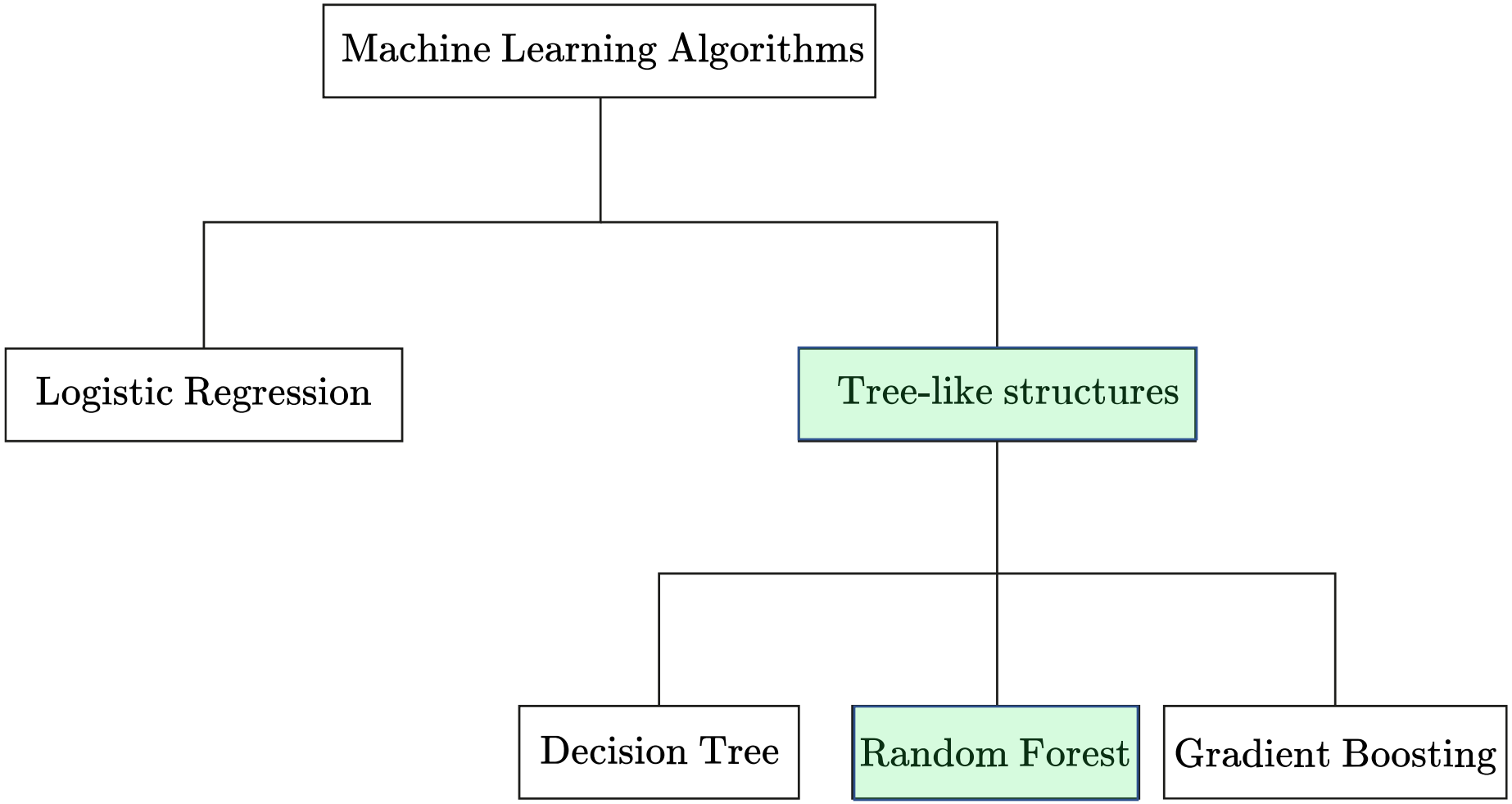
$$\text{Gini index} = \sum_{j=1}^{K} w_j \text{Gini impurity}(j,\text{after})$$

The feature with the lowest Gini index is used as the next splitting feature

Let’s see an example:

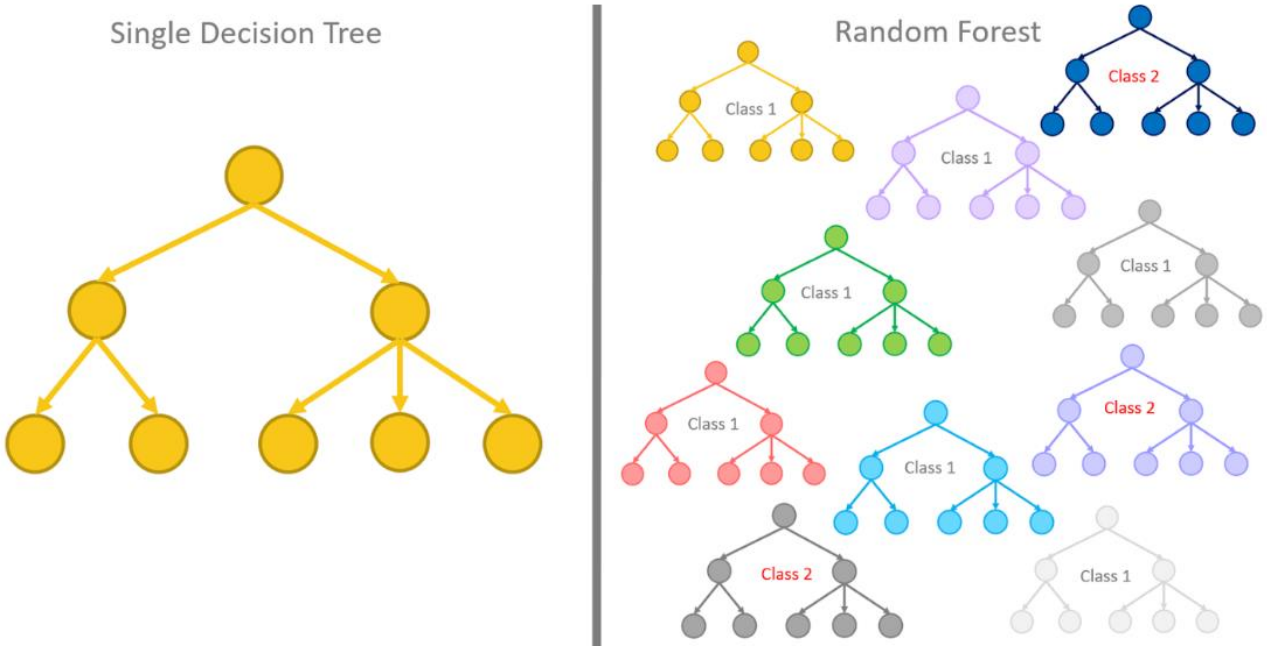
Code: https://github.com/Spoksonat/Particle_Physics_School/blob/main/decision_tree.ipynb

Data: https://drive.google.com/drive/folders/1gsqEcoXQzrSyUrDCFkw3kkMzuYByp0jo?usp=sharing


Machine Learning Algorithms
Logistic Regression
Tree-like structures
Decision Tree
Random Forest
Gradient Boosting

Random Forest

”Many is better than one”

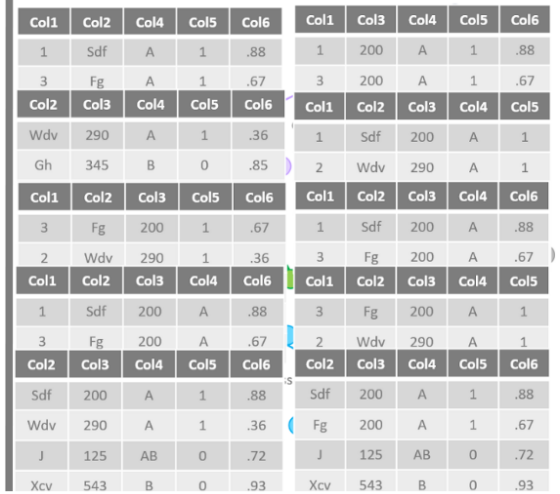


Taken from: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

Original Training Set

Col1	Col2	Col3	Col4	Col5	Col6
1	Sdf	200	A	1	.88
3	Fg	200	A	1	.67
2	Wdv	290	A	1	.36
4	Gh	345	B	0	.85
1	J	125	AB	0	.72
3	Xcv	543	B	0	.93
2	gbn	367	A	1	.18

Training Subsets via Bootstrapping



Generated by bootstrapping: random sampling with replacement

$\Rightarrow$ Trees are not statistically correlated

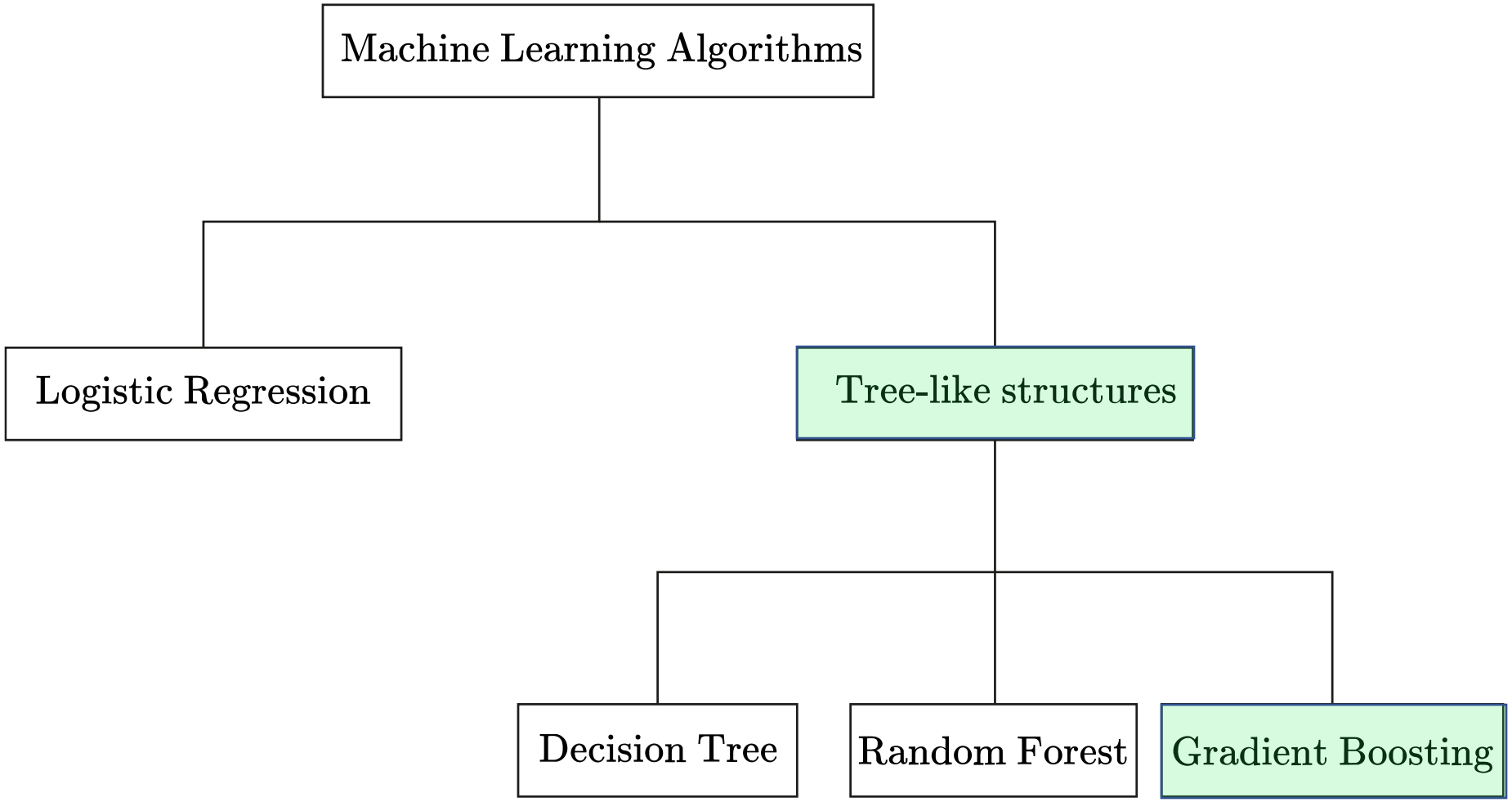
Typically, if there are m features, a subset of $\sqrt{m}$ is considered in each splitting of each decision tree

The majority rule: The prediction offered by the majority of the N trees is adopted as the final one

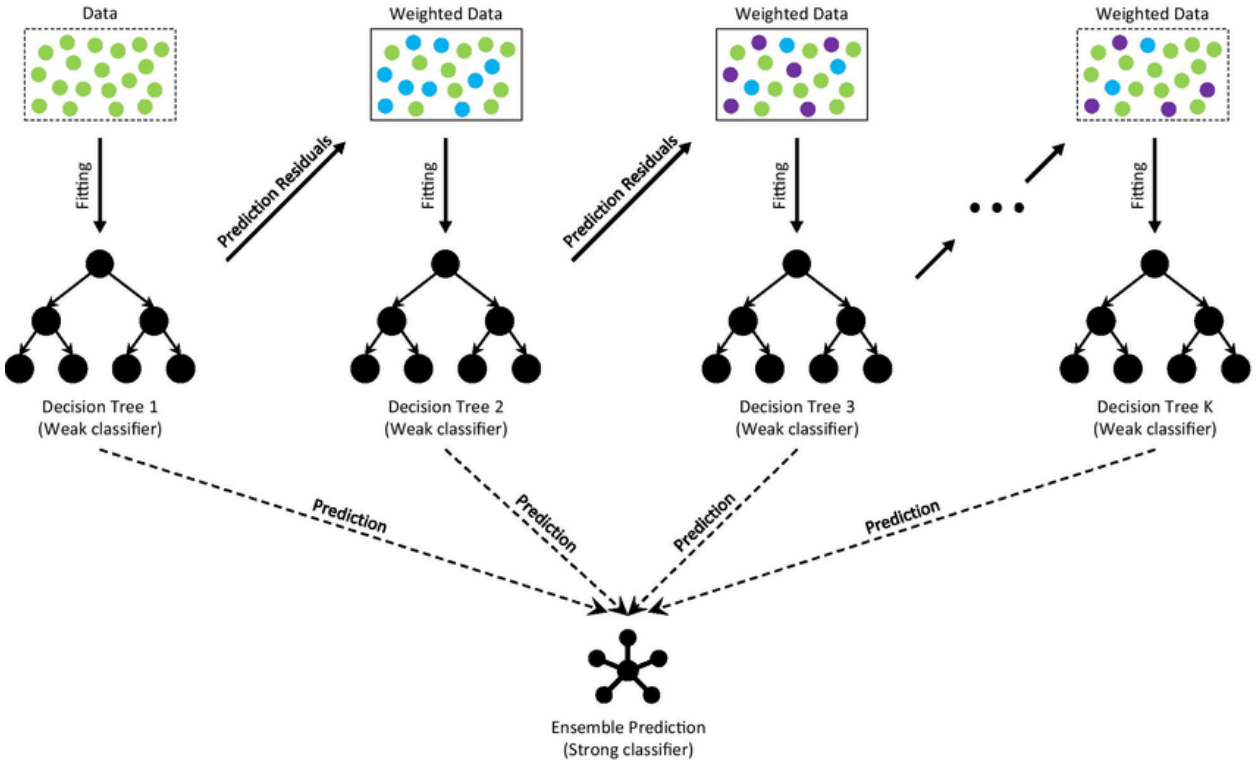
Let’s see an example:

Code: https://github.com/Spoksonat/Particle_Physics_School/blob/main/random_forest.ipynb

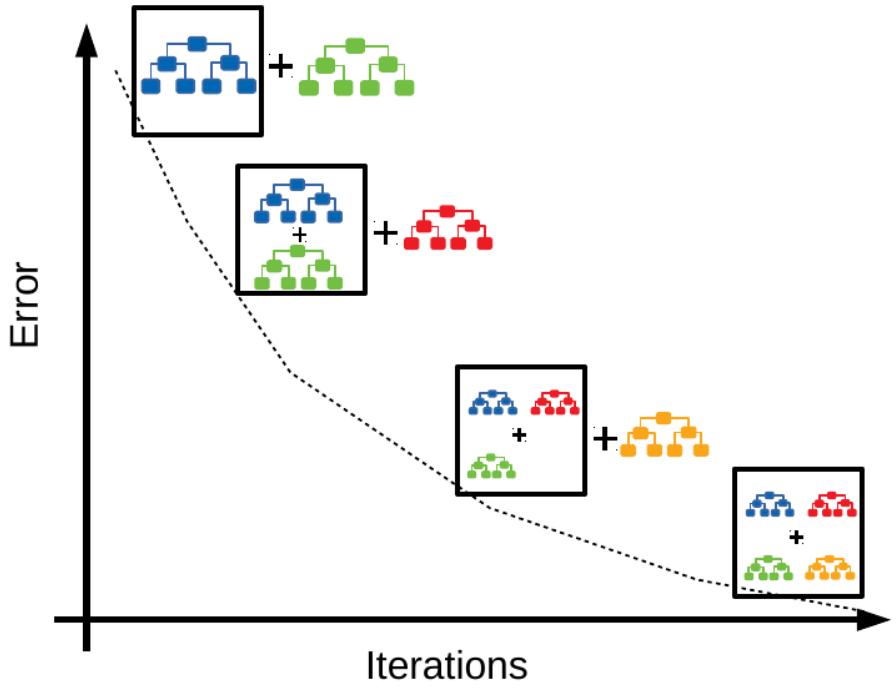
Data: https://drive.google.com/drive/folders/1gsqEcoXQzrSyUrDCFkw3kkMzuYByp0jo?usp=sharing


Machine Learning Algorithms
Logistic Regression
Tree-like structures
Decision Tree
Random Forest
Gradient Boosting

Gradient Boosting

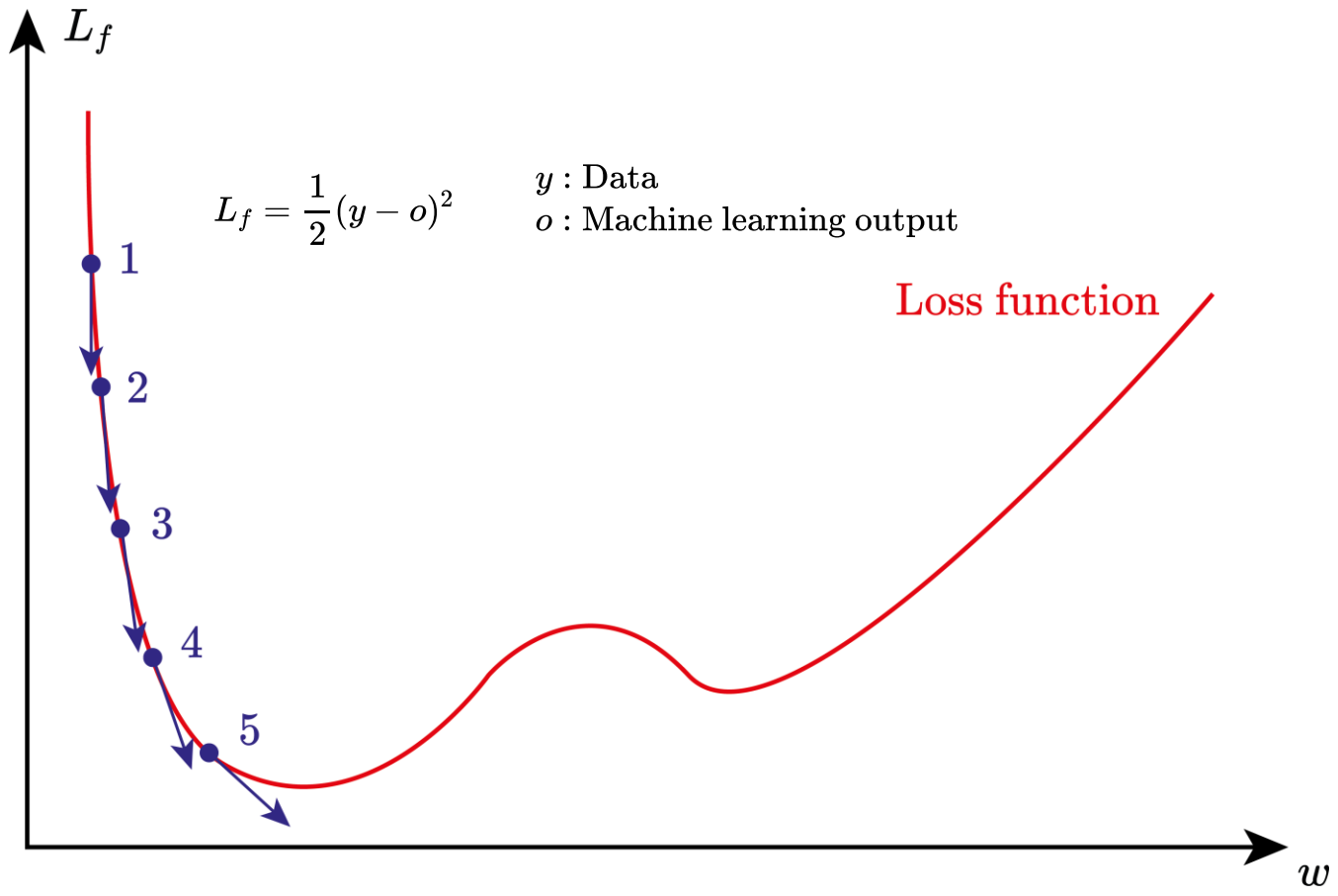


Taken from: https://www.researchgate.net/figure/The-architecture-of-Gradient-Boosting-Decision-Tree_fig2_356698772



Taken from: https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea

Gradient boosting algorithm is also based in the minimization of a loss function,
which tells us how far the predictions of the algorithm are from the true outputs given in training data.



Gradient descent method:

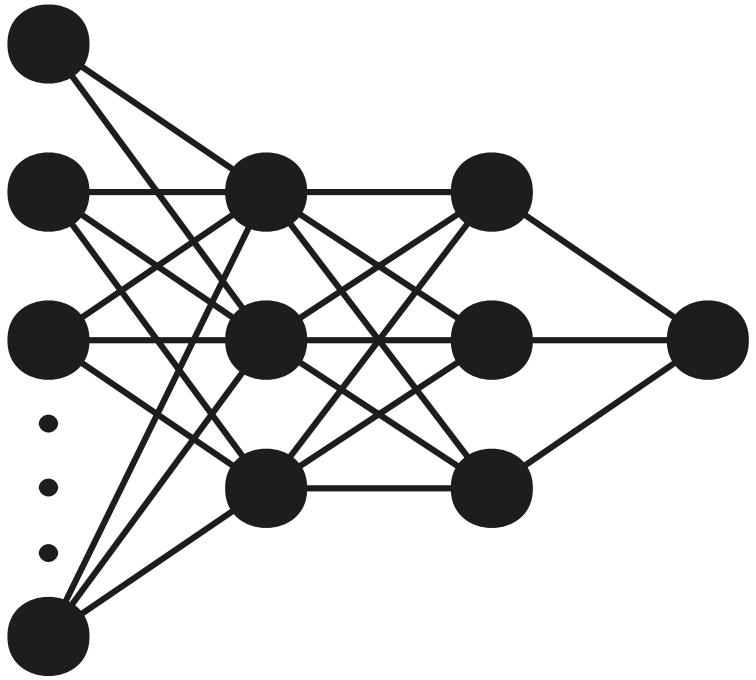
1. Select random values of weights w_i. We save these values in a vector $\vec{w}_1$ (represented by the point 1 in Figure).
2. Calculate the gradient of the loss function L, which is ∇L. The gradient vector is represented as a blue arrow in Figure
3. We update the weights $\vec{w}_1$ to $\vec{w}_1 \to \vec{w}_2 = \vec{w}_1 - \lambda \nabla L_f$. This new vector of weights is represented by the point 2 in Figure . The parameter λ is called **learning rate** and tells us at which velocity we pass from point 1 to point 2 in Figure . Typically the value of λ is chosen manually.
4. We repeat the updating procedure (steps 2 and 3) for the weights until we reach the minimum (until $\nabla L_f \approx 0$).
5. The values of the parameters w_i which makes $\nabla L_f \approx 0$ are the values that gradient boosting algorithm use to predict the final output.

Let’s see an example:

Code: https://github.com/Spoksonat/Particle_Physics_School/blob/main/gradient_boosting.ipynb

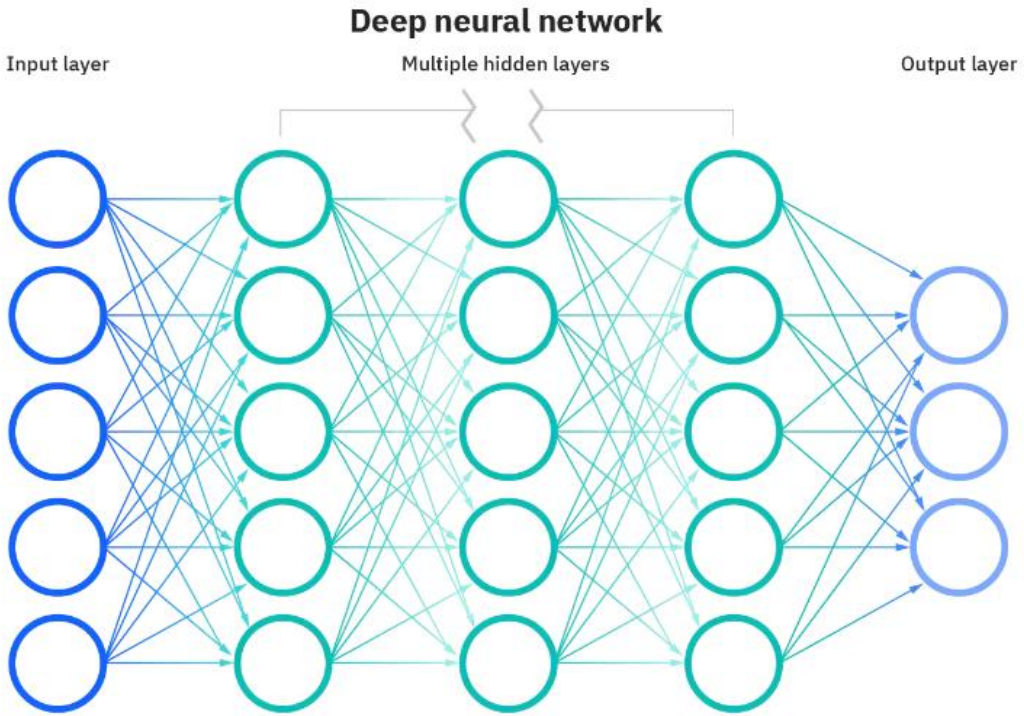
Data: https://drive.google.com/drive/folders/1gsqEcoXQzrSyUrDCFkw3kkMzuYByp0jo?usp=sharing

Deep Learning

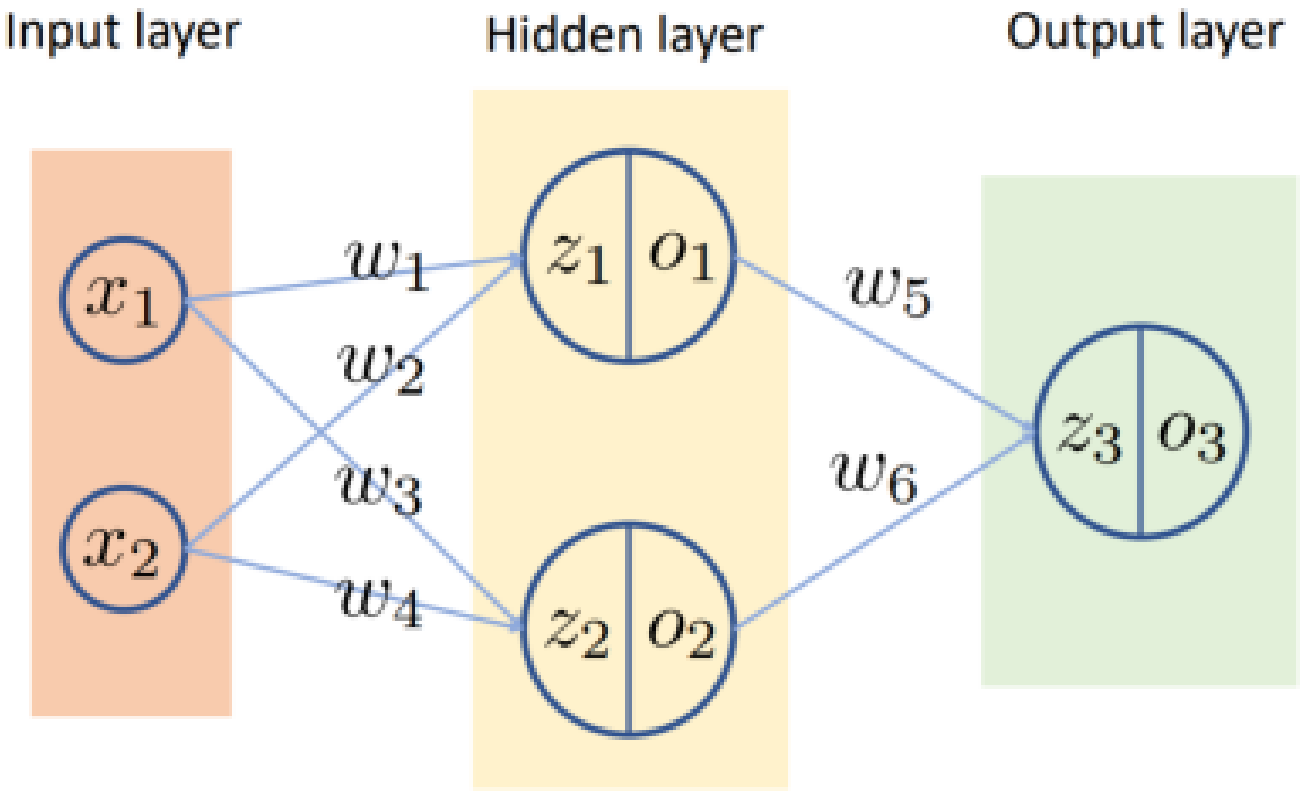


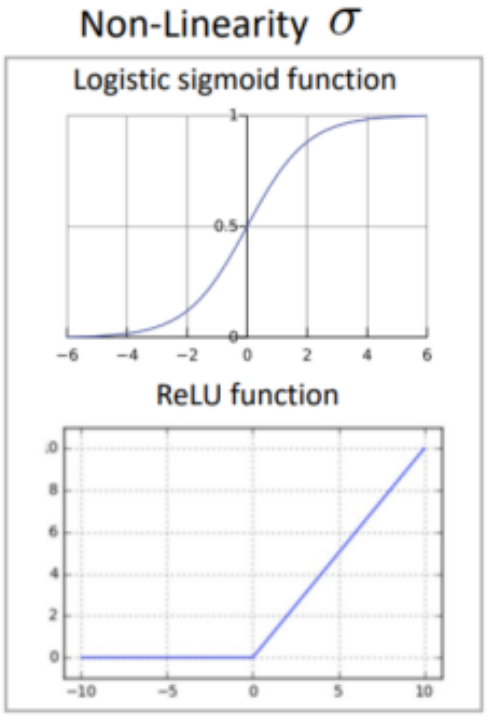
Deep Learning

Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another



Taken from: https://www.ibm.com/cloud/learn/neural-networks





$$z_1 = w_1 x_1 + w_2 x_2$$

$$o_1 = \sigma(z_1)$$

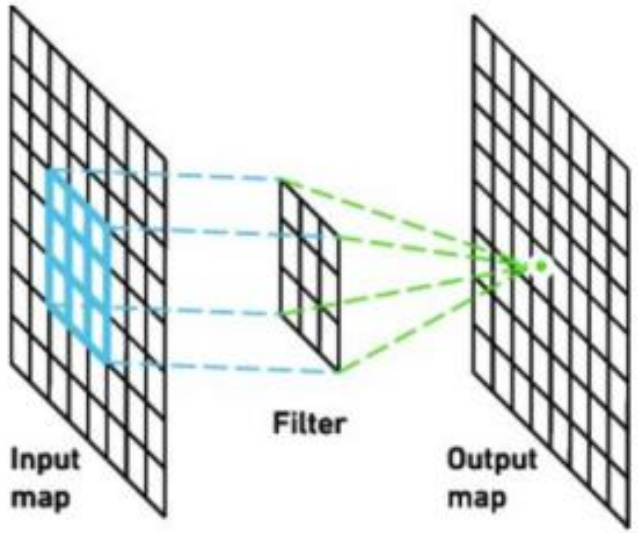
The idea of the neutral network is: Given input $x = x1, x2, \ldots$ and target y, find w_i that minimizes the loss function L

$\Rightarrow$ Gradient descent method

Convolutional layer

A convolutional layer is a layer we add to the neural network to simplify its complexity

We convolve a filter to an input map to create a feature map (output map) that summarizes the presence of detected features in the input



Input map

6 x 6 image

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

*

Filter

1	-1	-1
-1	1	-1
-1	-1	1

=

Output map

C_{11}	C_{12}	C_{13}	C_{14}
C_{21}	C_{22}	C_{23}	C_{24}
C_{31}	C_{32}	C_{33}	C_{34}
C_{41}	C_{42}	C_{43}	C_{44}

Input map

6 x 6 image

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

*

Filter

1	-1	-1
-1	1	-1
-1	-1	1

=

Output map

3	C_{12}	C_{13}	C_{14}
C_{21}	C_{22}	C_{23}	C_{24}
C_{31}	C_{32}	C_{33}	C_{34}
C_{41}	C_{42}	C_{43}	C_{44}

$$C_{11} = \sum \begin{matrix} 1\times1 & 0\times-1 & 0\times-1 \\ 0\times-1 & 1\times1 & 0\times-1 \\ 0\times-1 & 0\times-1 & 1\times1 \end{matrix} = \begin{matrix} (1)(1)+(0)(-1)+(0)(-1) \\ +(0)(-1)+(1)(1)+(0)(-1) \\ +(0)(-1)+(0)(-1)+(1)(1) \end{matrix} = 3$$

Input map

6 x 6 image

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

*

Filter

1	-1	-1
-1	1	-1
-1	-1	1

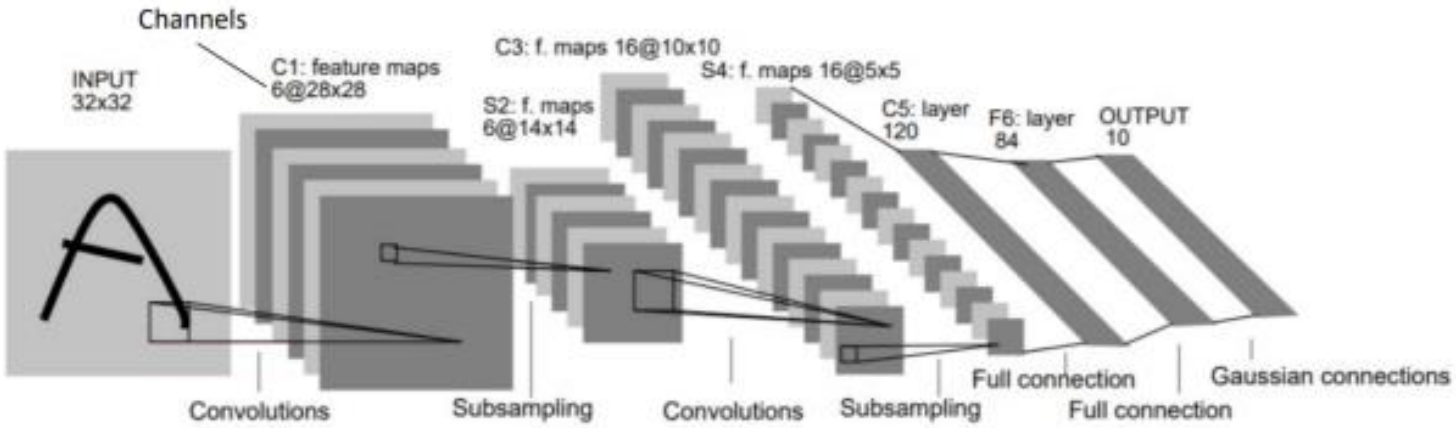
=

Output map

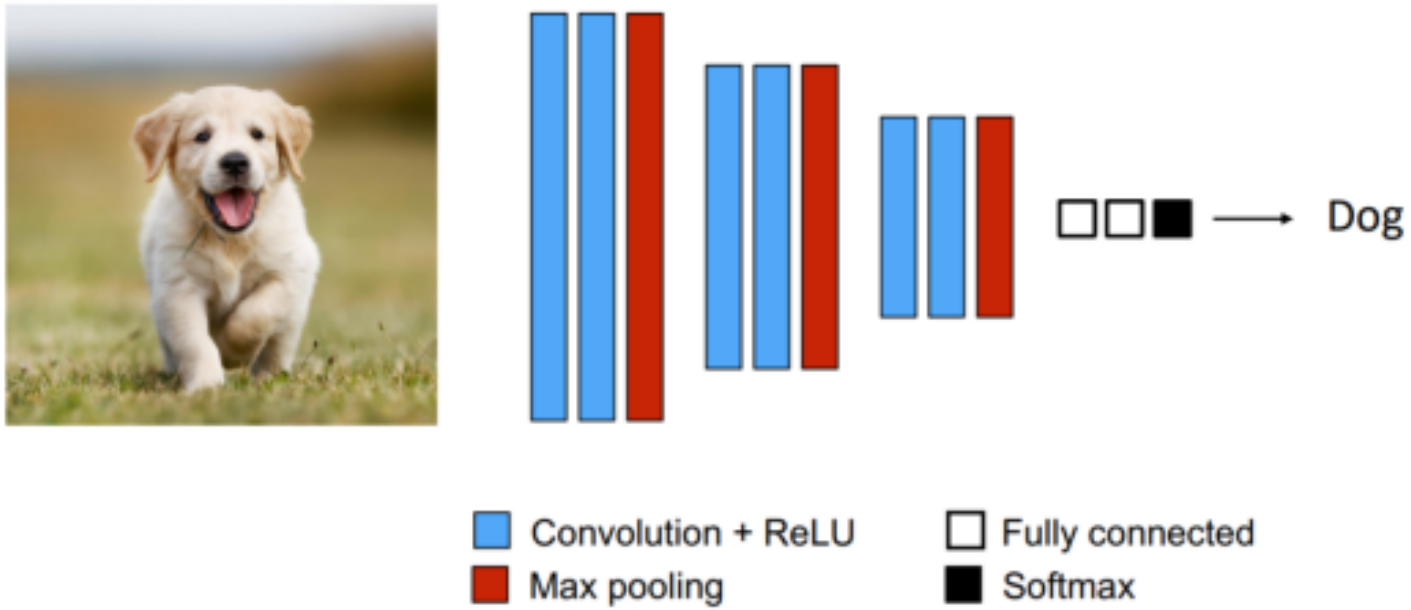
3	-1	C_{13}	C_{14}
C_{21}	C_{22}	C_{23}	C_{24}
C_{31}	C_{32}	C_{33}	C_{34}
C_{41}	C_{42}	C_{43}	C_{44}

$$C_{12} = \sum \begin{matrix} 0\times1 & 0\times-1 & 0\times-1 \\ 1\times-1 & 0\times1 & 0\times-1 \\ 0\times-1 & 1\times-1 & 1\times1 \end{matrix} = \begin{matrix} (0)(1)+(0)(-1)+(0)(-1) \\ +(1)(-1)+(0)(1)+(0)(-1) \\ +(0)(-1)+(1)(-1)+(1)(1) \end{matrix} = -1$$

Convolutional Neural Network



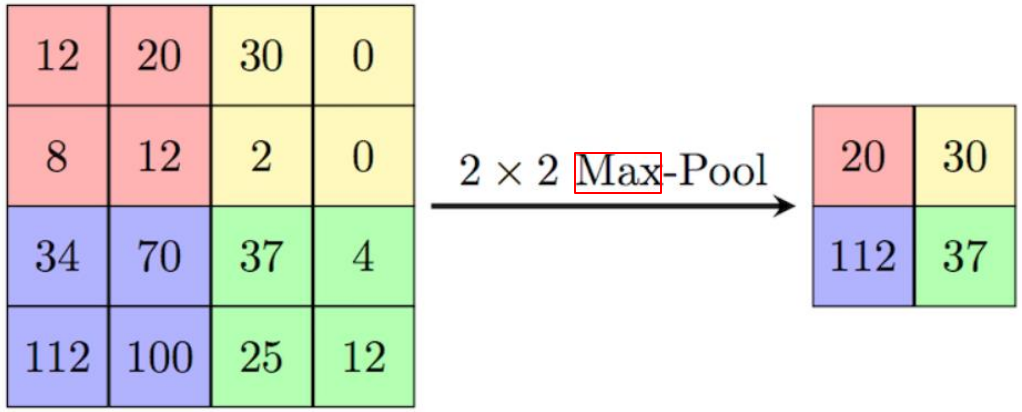
Classification Network

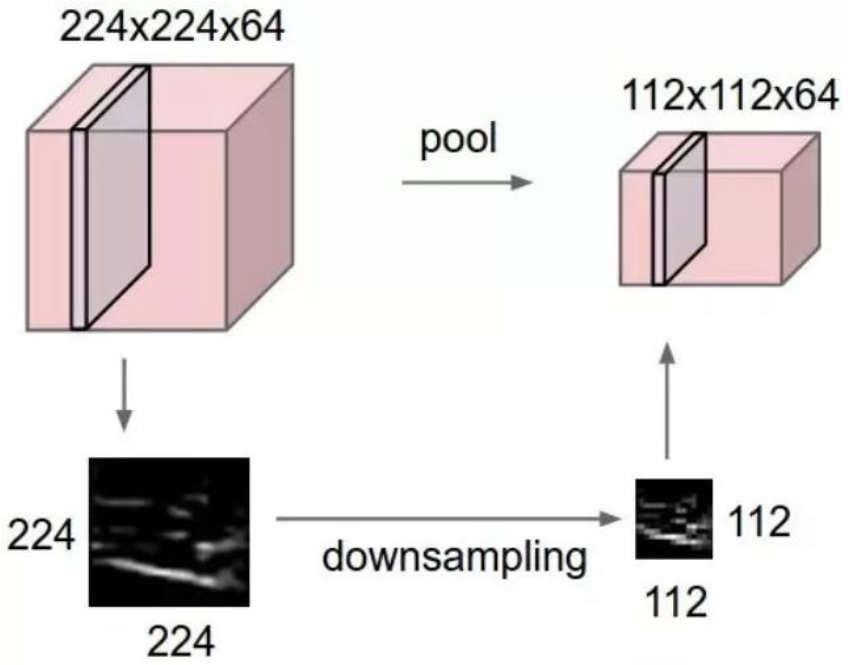


Max pooling: Help over-fitting by down-sampling of an input

Softmax: Map a vector of real numbers into a probability distribution

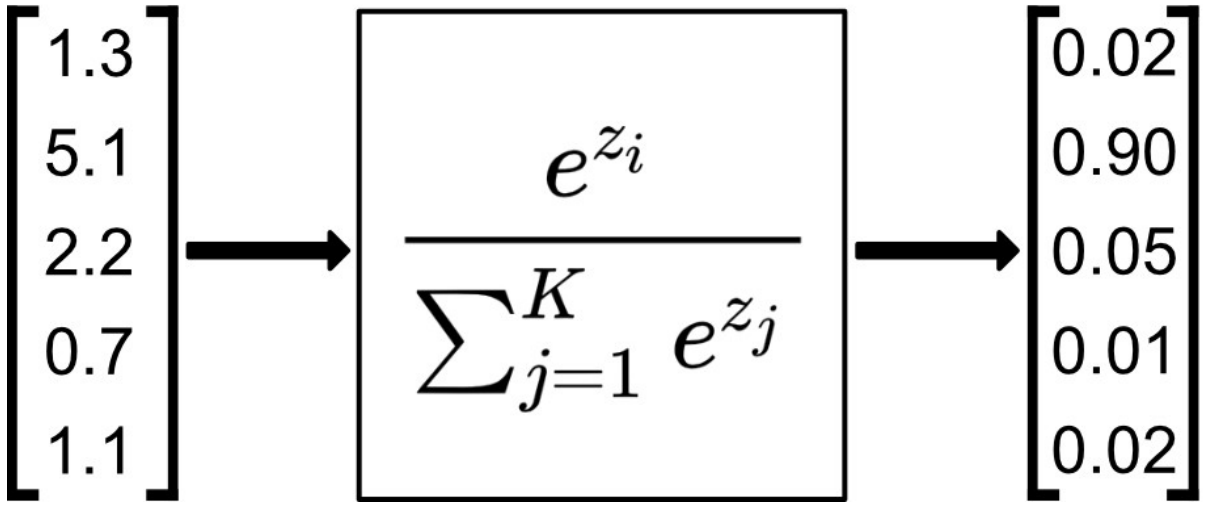
Max Pooling





Taken from: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Softmax Function



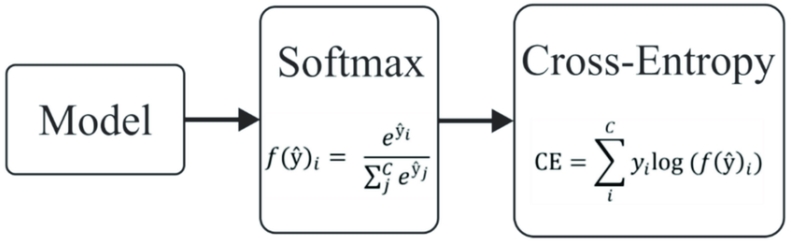
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=47),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(2, activation="softmax")
])

model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=30,
                  validation_data=(x_test, y_test))
```

Adapted from: https://thecleverprogrammer.com/2022/01/10/classification-with-neural-networks-using-python/

Epoch: one cycle through the full training dataset

SGD: Stochastic Gradient Descent

See: https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html



Thanks!