

# Data Analytics in Cyber Security

## CT115-3-M (Version E)

## **Machine Learning Algorithms in Practice**

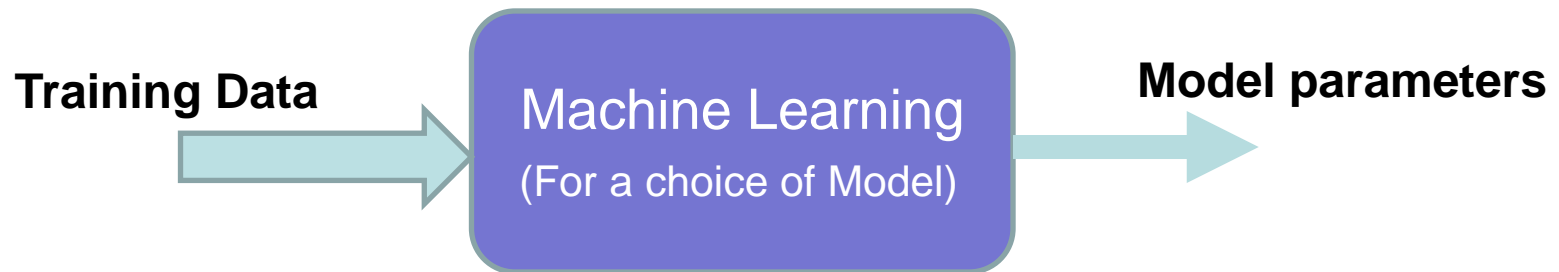
# TOPIC LEARNING OUTCOMES

At the end of this topic, you should be able to:

1. Understand different types of supervised learning algorithm.
2. Understand the operating details of each algorithm.

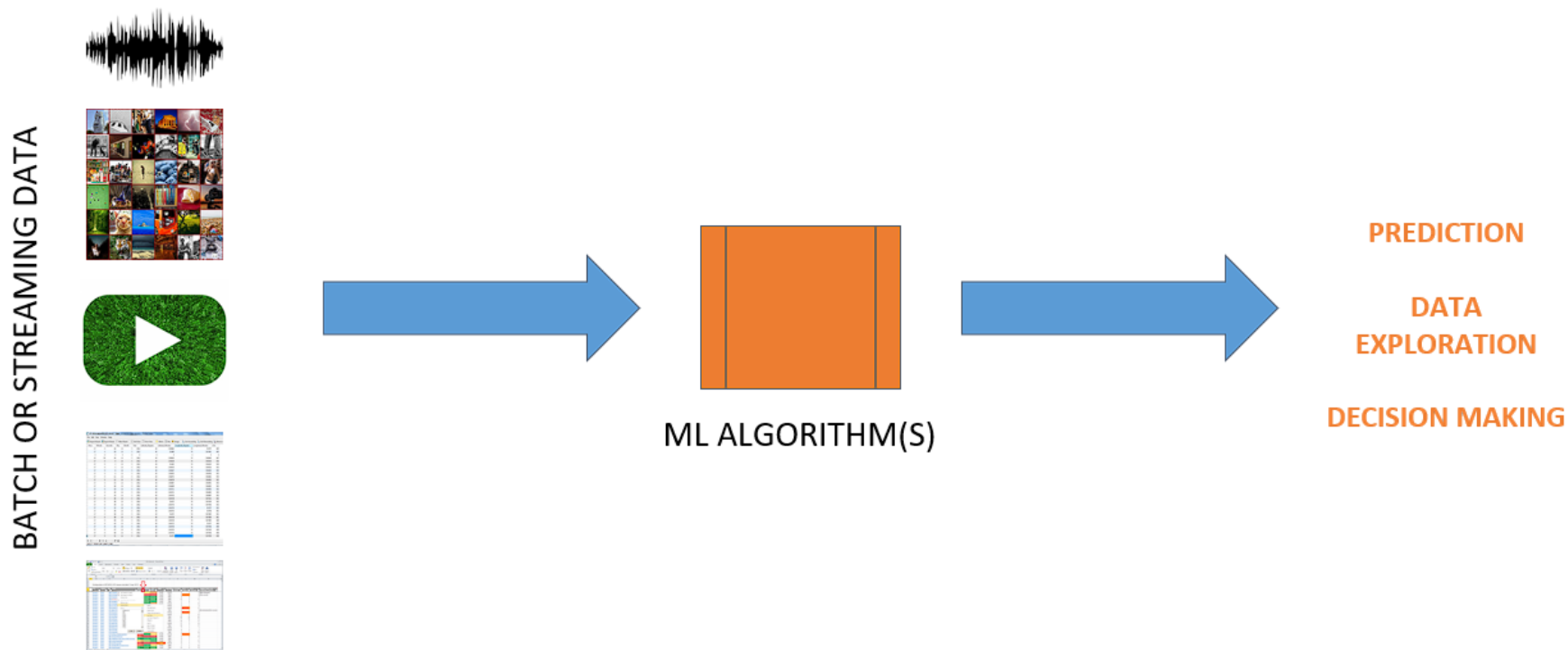
# Machine Learning

- To perform the tasks autonomously, machines need to learn.
- For this they **learn** a selected **model** using examples (**training data**)



- For a selected model, machines learn **model parameters**. This process is commonly known as “**model learning**”
- But what is a **model**?

# Machine Learning Models

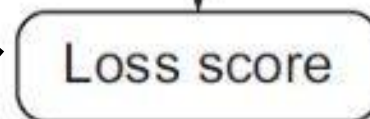
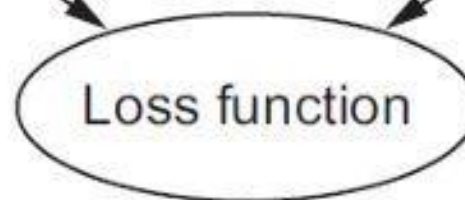
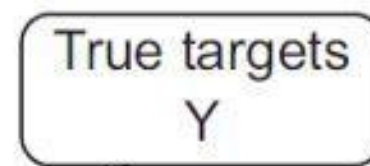
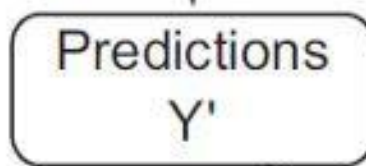
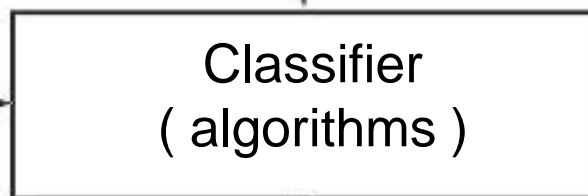
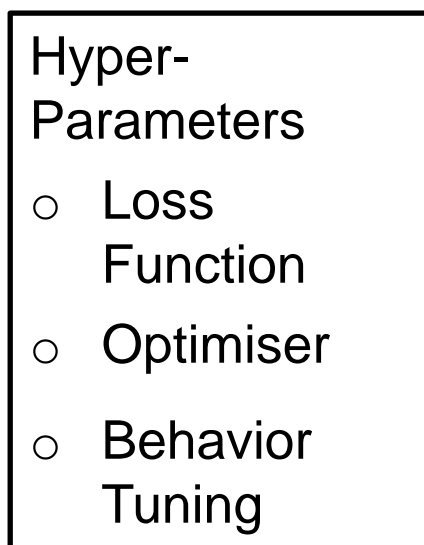


# Machine Learning Models

- Learning is the process of adapting **model parameters** of the classifier to produce a desired response.
- **Model parameters** are commonly known as the **weights**
- **Hyperparameters** are the knobs a data scientist gets to turn when setting up a classifier.
  - They are arguments to the function that affect the algorithm's behavior, such as number of iterations, or options between variants of how the algorithm behaves.

**Dataset**

Input Row



A **loss function** (or error function) measures how far an estimated value is from its true value.

Optimizer aims to **minimize** the **cost function**: the average loss over the entire training dataset.

Loss function is applied to a single training example

# Loss Function

- A **loss function** (or error function) measures how far an estimated value is from its true value.
- A loss function is for a single training example, and a **cost function** is the average loss over the entire training dataset.
  - Classifier optimization strategies aim to minimize the **cost function**
- In essence, the **loss function** and the **optimizer** work together to fit the algorithm to the data in the best way possible.

There are many different loss functions and optimisers

# Contents & Structure

- Algorithms for supervised learning
- Five general types



# Algorithms for Supervised Learning

## Linear Models

- Linear Regression
- Logistic Regression
- Linear Discriminant Analysis

## Support Vector

- (various kernels)

## Neural Network

- Perceptron & MLP

## Non-Linear Models

- Naïve Bayes
- k-Nearest Neighbours
- Decision Tree

## Ensemble

### Bagging

- Random Forest

### Boosting

- AdaBoost
- Gradient Tree Boosting

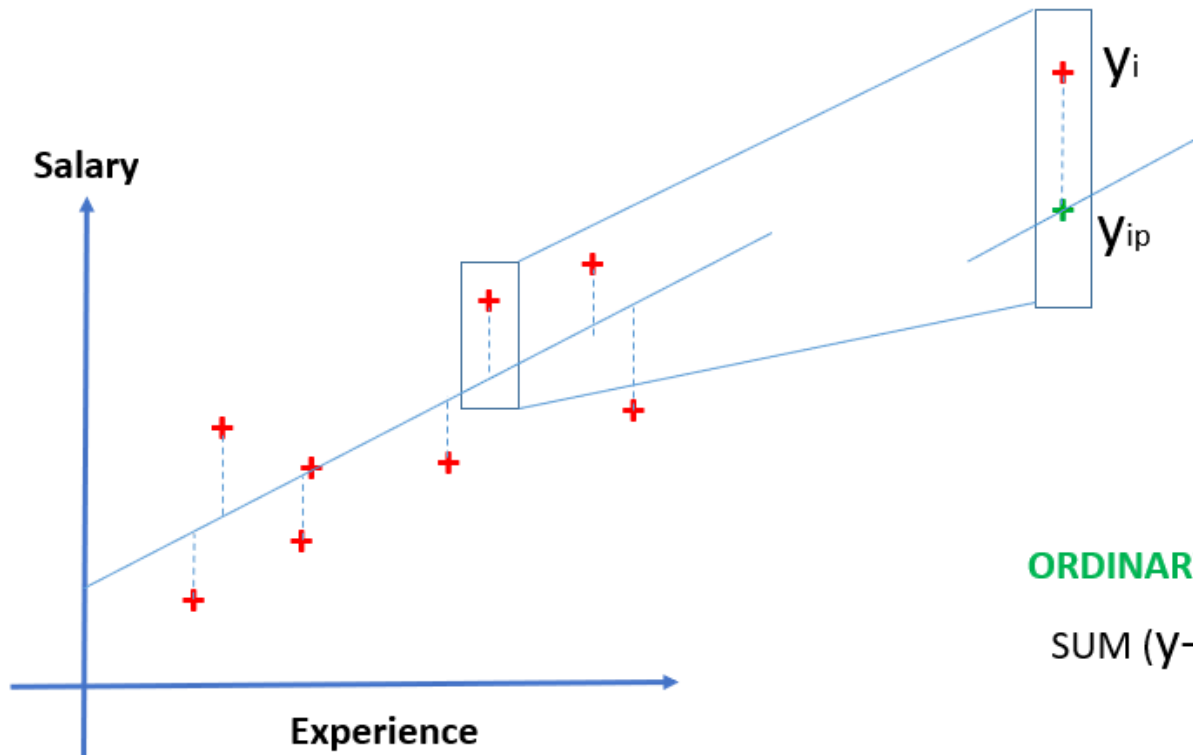
# Linear regression

- Linear Regression fits a linear model (  $y = a + bX$  ) that best fits the relationship between the input variables (X) and the variable we will predict (y)
- The *slope* of the line is **b**, and **a** is the *intercept* (the value of y when  $X = 0$ )
- Different techniques can be used to learn the linear regression model from data, such as ordinary least squares and gradient descent optimization (these are **hyperparameters** for the classifier)
- [Scikit Learn Documentation](#)

# Linear Regression – Examples



# Linear Regression – Examples



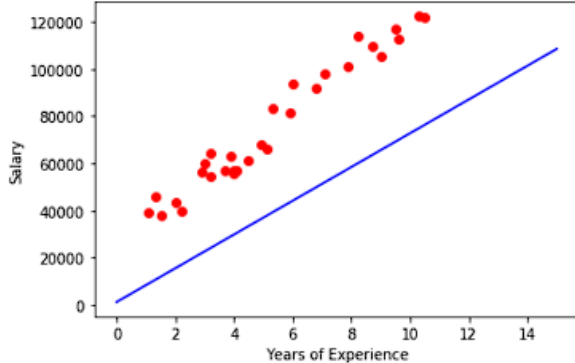
In most common case, the line is determined by minimising the sum of the squares of the distances (*vertical offset*) of all points to points on the line.

**ORDINARY LEAST SQUARE METHOD**

$\text{SUM } (y - y_p)^2 \longrightarrow \text{MINIMUM}$

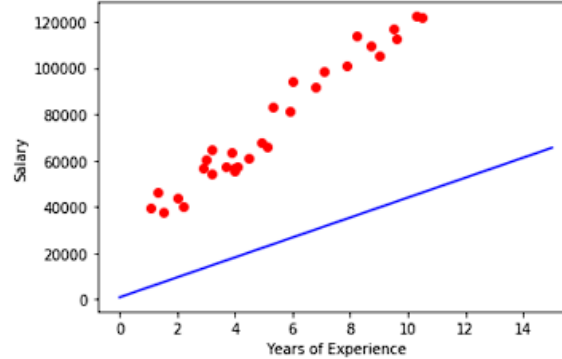
# Linear Regression – Examples

Salary vs Experience



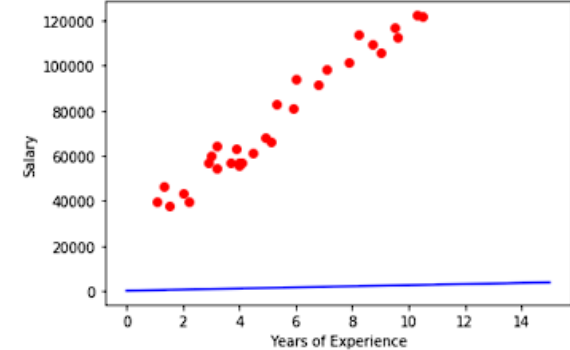
Loss is 1453919232.0

Salary vs Experience



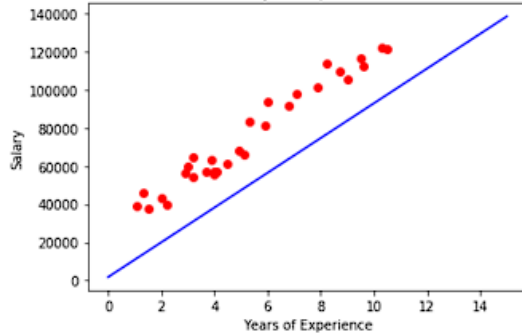
Loss is 3044867072.0

Salary vs Experience



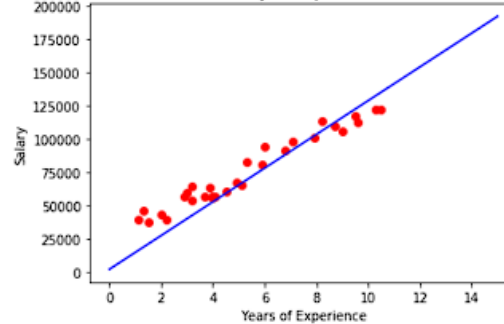
Loss is 6425976832.0

Salary vs Experience



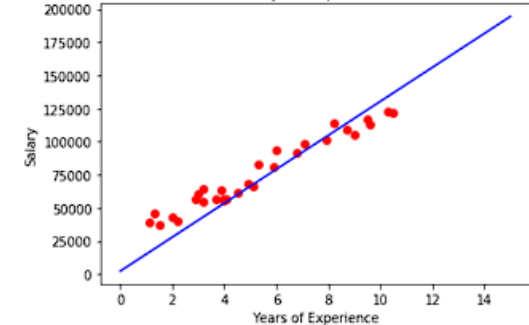
Loss is 712537856.0

Salary vs Experience



Loss is 152592432.0

Salary vs Experience



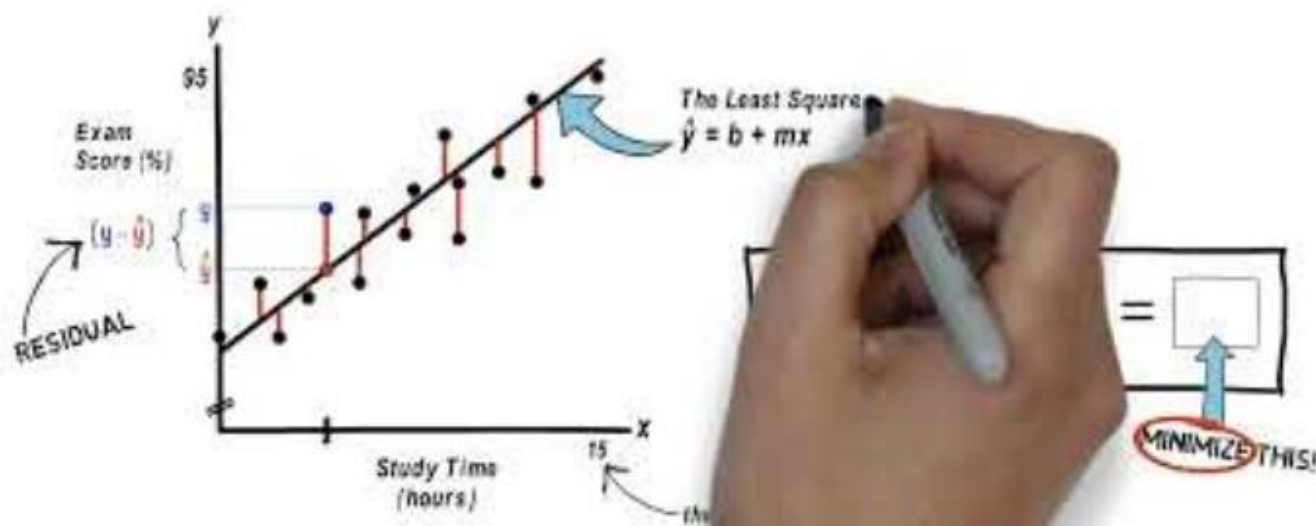
Loss is 149858032.0

# Linear Regression

## The Least Squares Regression Line

GOAL: Predict one variable ( $y$ ), given another variable ( $x$ ).

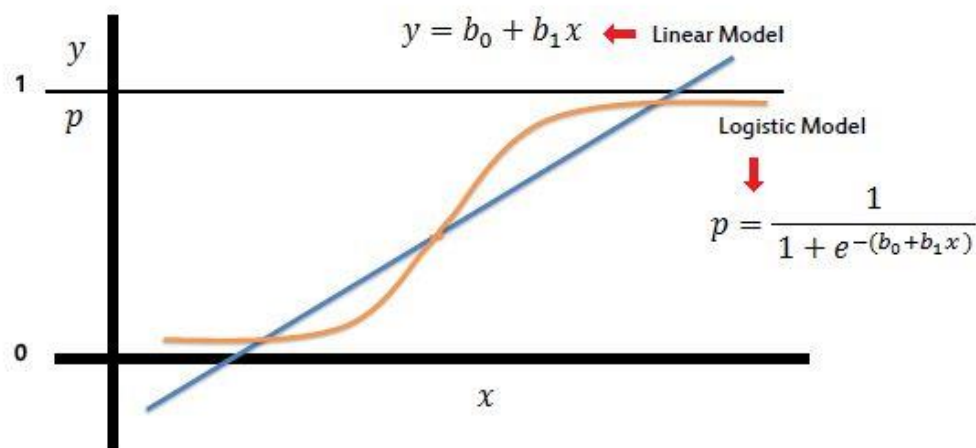
QUESTION: How do we define the best fitting line through a scatter plot?



<https://youtu.be/aPHg1kGeKsg>

# Logistic Regression

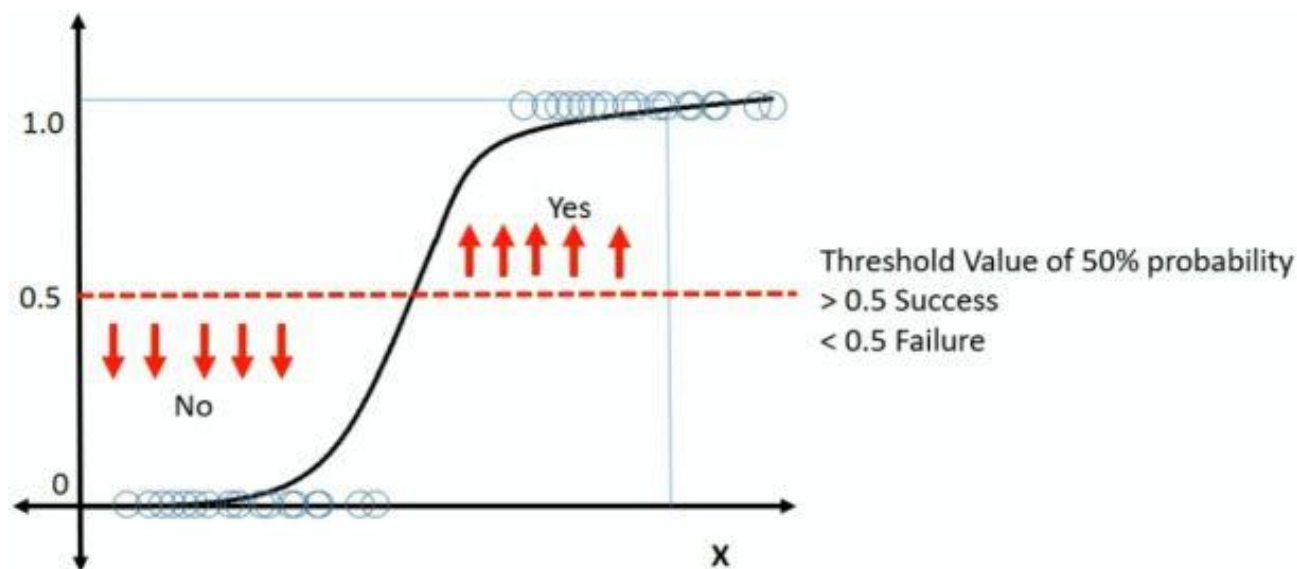
- Logistic regression is the go-to method for binary classification problems (two class values).
- Examples: Pass/Fail, Good/Bad, Defect/Not Defect, Mammal/Not Mammal.
- Does not assume a linear relationship between dependent and independent variables.



Unlike linear regression, the prediction is transformed using a logistic function

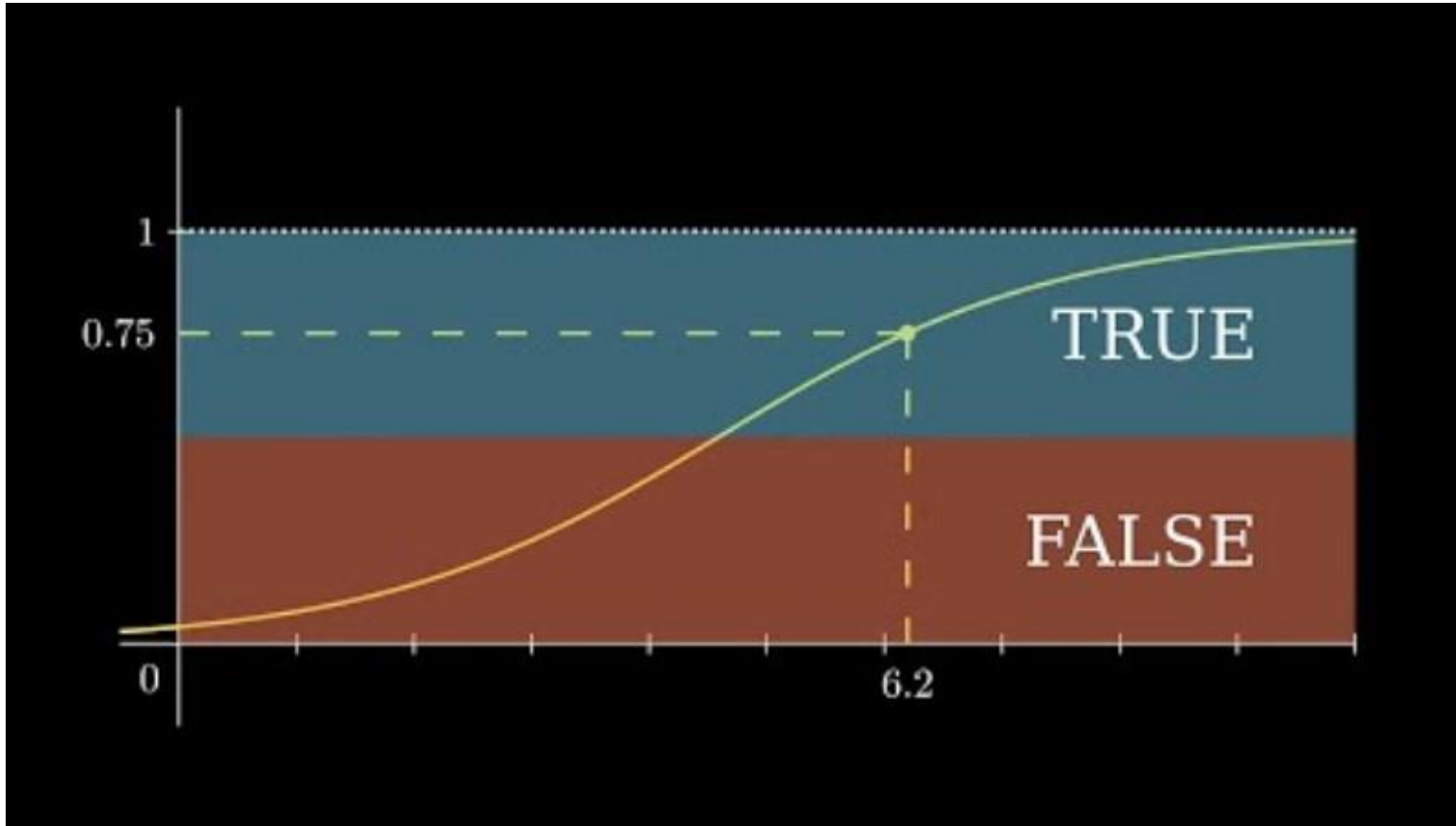
# Logistic Regression

The logistic function will transform any value into the range 0 to 1



- This is useful because we can apply a rule to the output of the logistic function to snap values to zero and one (e.g. IF less than 0.5 THEN output 0) and predict a class value.
- Predictions made by logistic regression can also be used as the probability of a given data instance belonging to class 0 or class 1.
- [Scikit Learn Documentation](#)

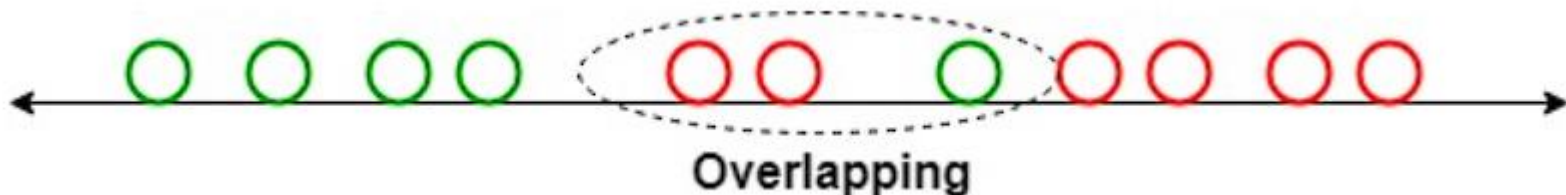
# Logistic Regression



<https://youtu.be/EKm0spFxFG4>

# Linear Discriminant Analysis

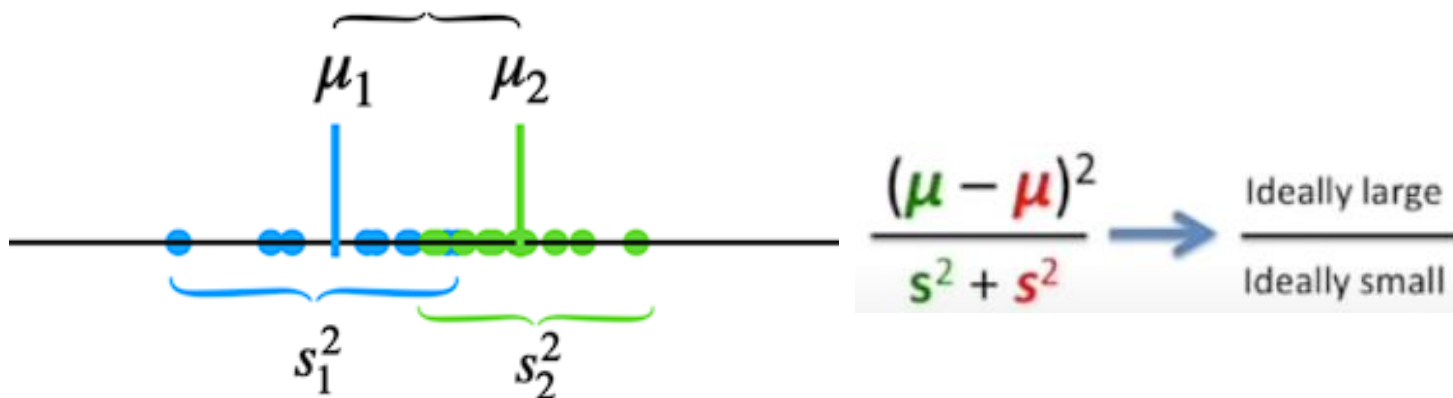
- LDA works by identifying a linear combination of features that separates or characterises two or more classes of objects or events.
- LDA does this by projecting data with two or more dimensions into one dimension so that it can be more easily classified. The technique is, therefore, sometimes referred to as dimensionality reduction.
- For example, we have two classes, and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping.



# Linear Discriminant Analysis

- LDA assumes that the data has a Gaussian distribution and that the covariance matrices of the different classes are equal. It also assumes that the data is linearly separable, meaning that a linear decision boundary can accurately classify the different classes. Two criteria are used by LDA to create a new axis:

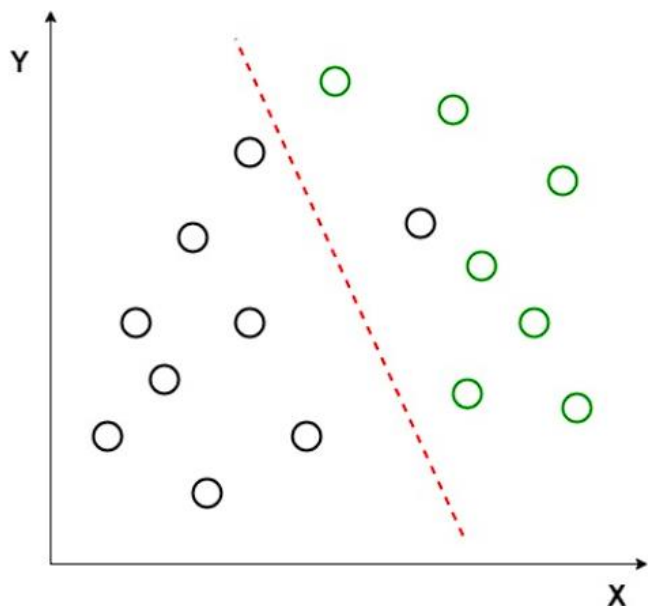
1. Maximize the distance between the means



2. Minimize the variations ("scatter") within each category

- [Scikit Learn Documentation](#)

# Linear Discriminant Analysis



*Linearly Separable Dataset*

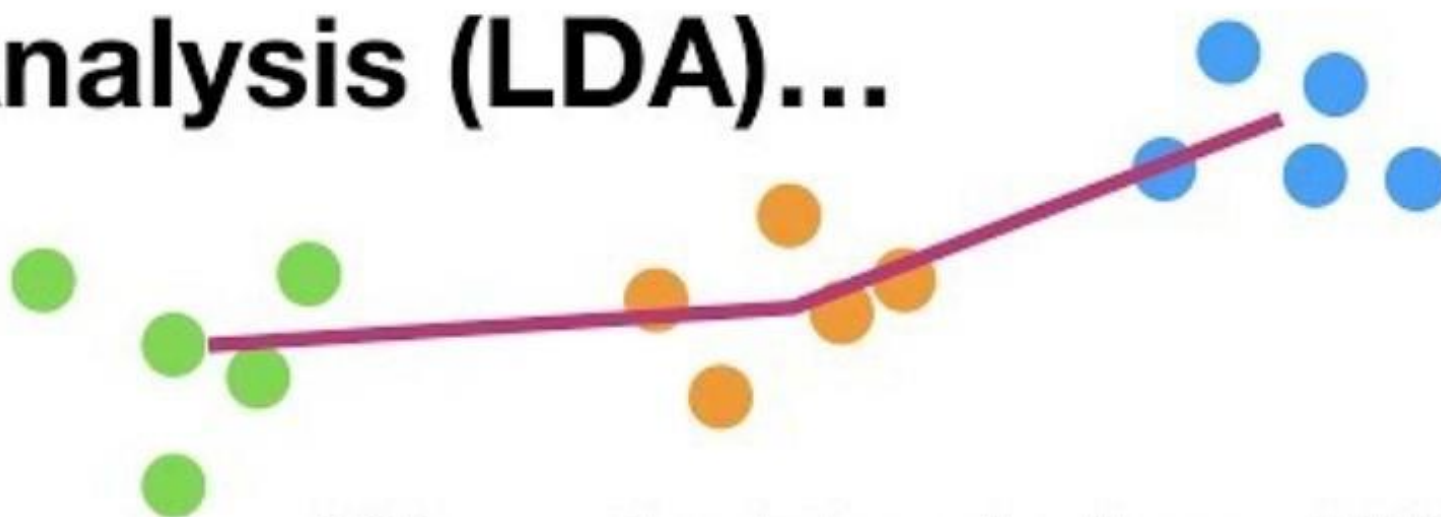
Linear Discriminant Analysis uses both axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximise the separation of the two categories and hence, reduces the 2D graph into a 1D graph.



# Linear Discriminant Analysis

---

## Linear Discriminant Analysis (LDA)...



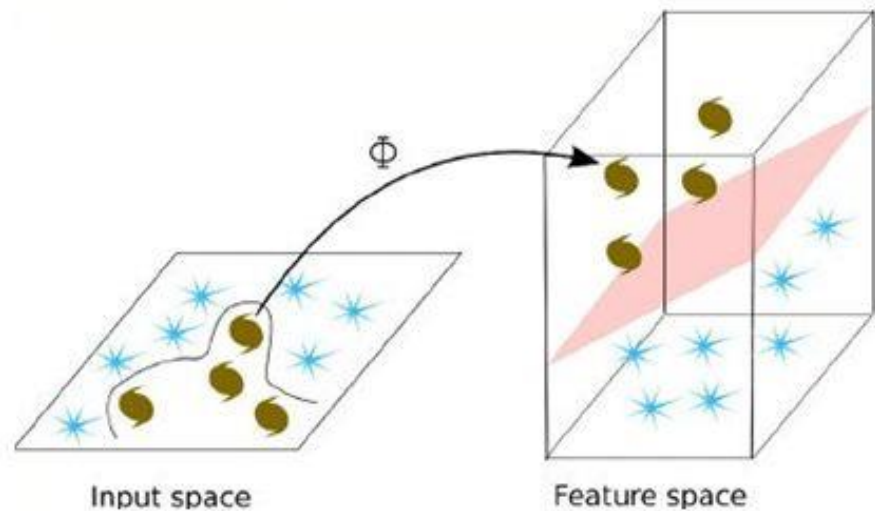
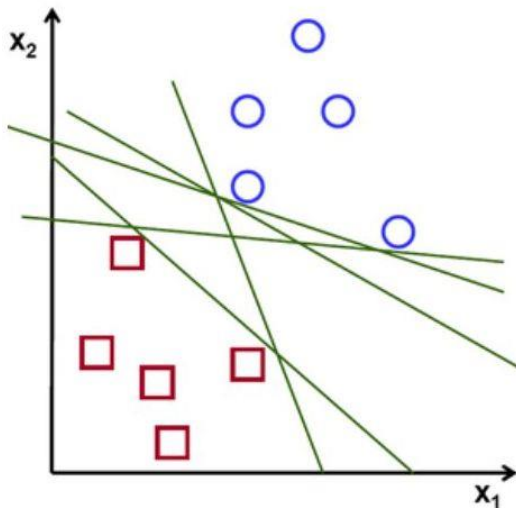
**...Clearly Explained!!!**

---

<https://youtu.be/azXCzI57Yfc?t=34>

# Linear Separators

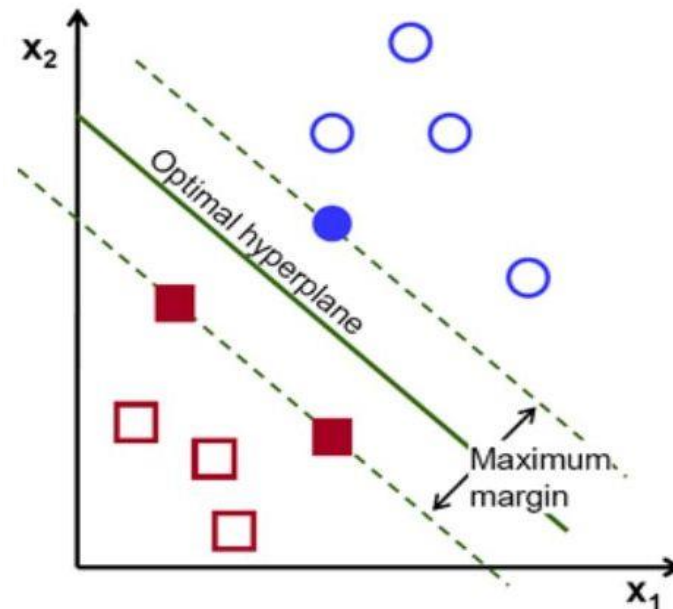
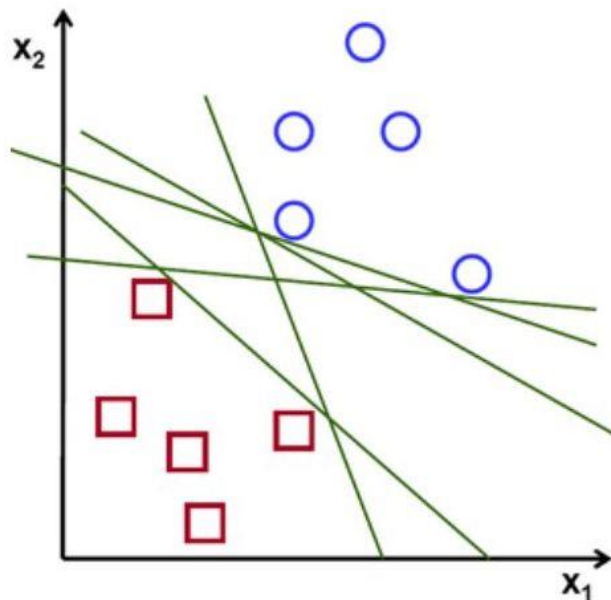
- Binary classification can be viewed as the task of separating 2 classes in feature space:



- But which of the linear separators is optimal?
- What if we cannot do it in two dimensions, but we could in 3?

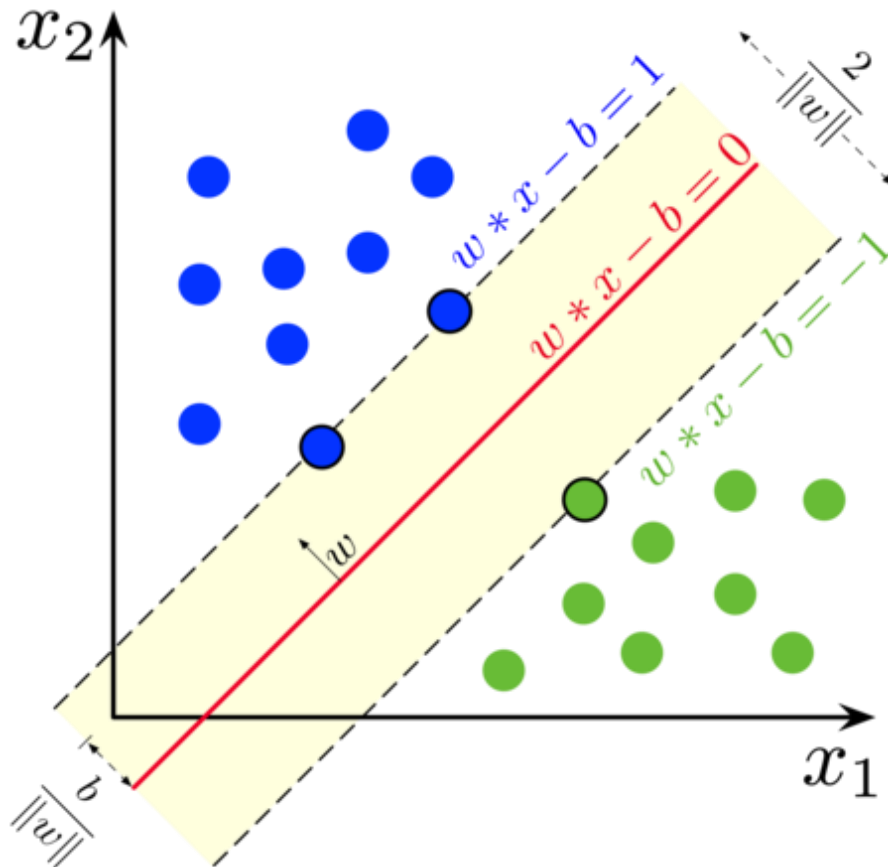
# Linear Support Vector Machine (SVM)

- Support Vector Machines can do both, based on the concept of a **hyperplane** that separates the classes



Observations on the margin are the support vectors that define the optimal hyperplane

# Linear SVM



- Maximum Margin classification is strictly based on observations at the margin between classes.
- An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those points lie nearest to it, called support vectors.

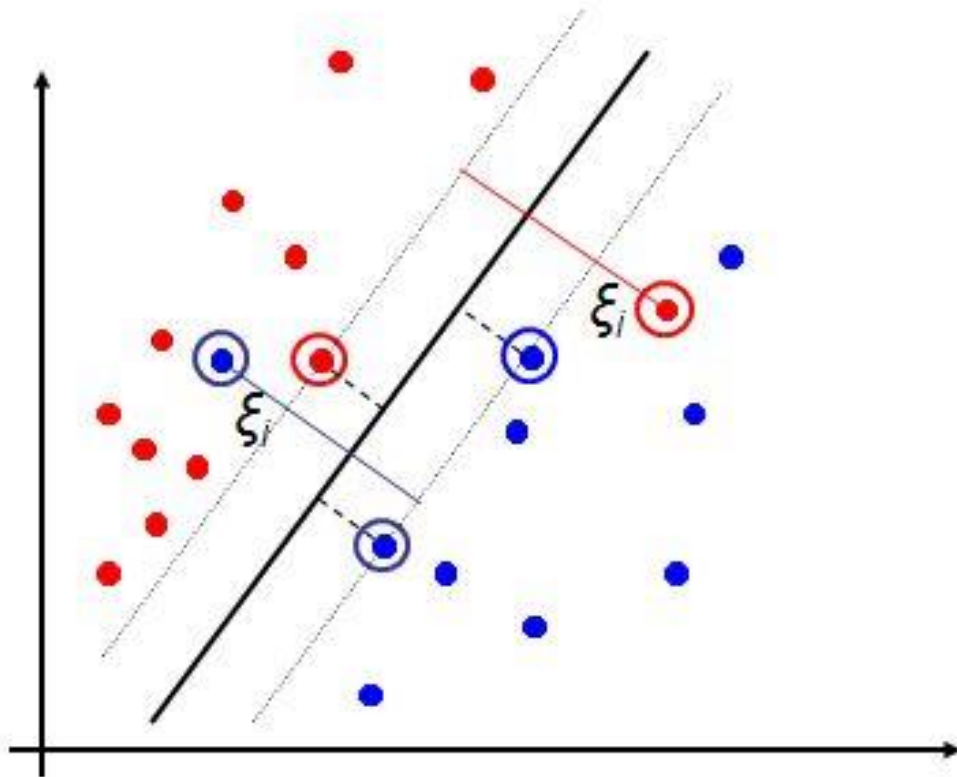
minimize  
 $\mathbf{w}, b$

$$\|\mathbf{w}\|_2^2$$

subject to

$$y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 \quad \forall i \in \{1, \dots, n\}$$

# Soft Margin Classification



- Parameter  $C$  is used for controlling the outliers: low  $C$  implies we are allowing more outliers; high  $C$  implies we are allowing fewer outliers.

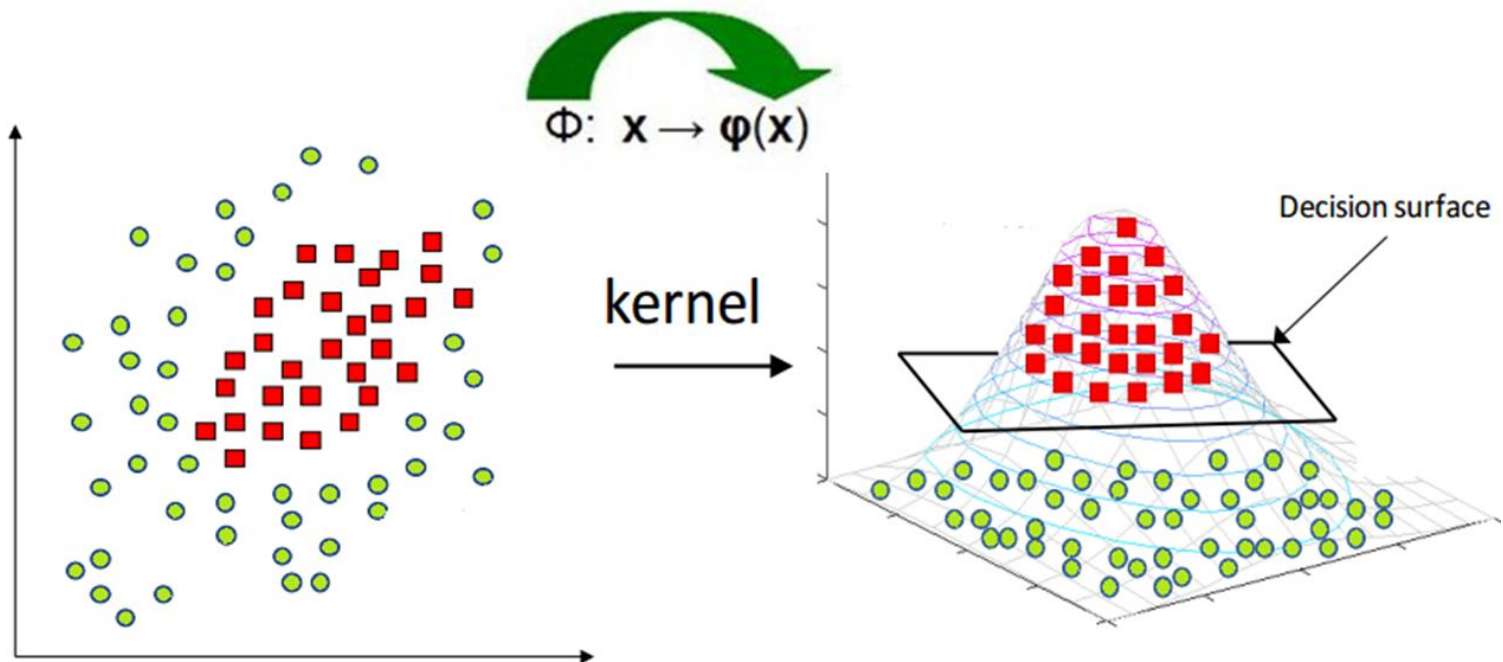
- [Scikit Learn Documentation](#)

$$\begin{aligned} &\underset{\mathbf{w}, b, \zeta}{\text{minimize}} && \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \zeta_i \end{aligned}$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0 \quad \forall i \in \{1, \dots, n\}$$

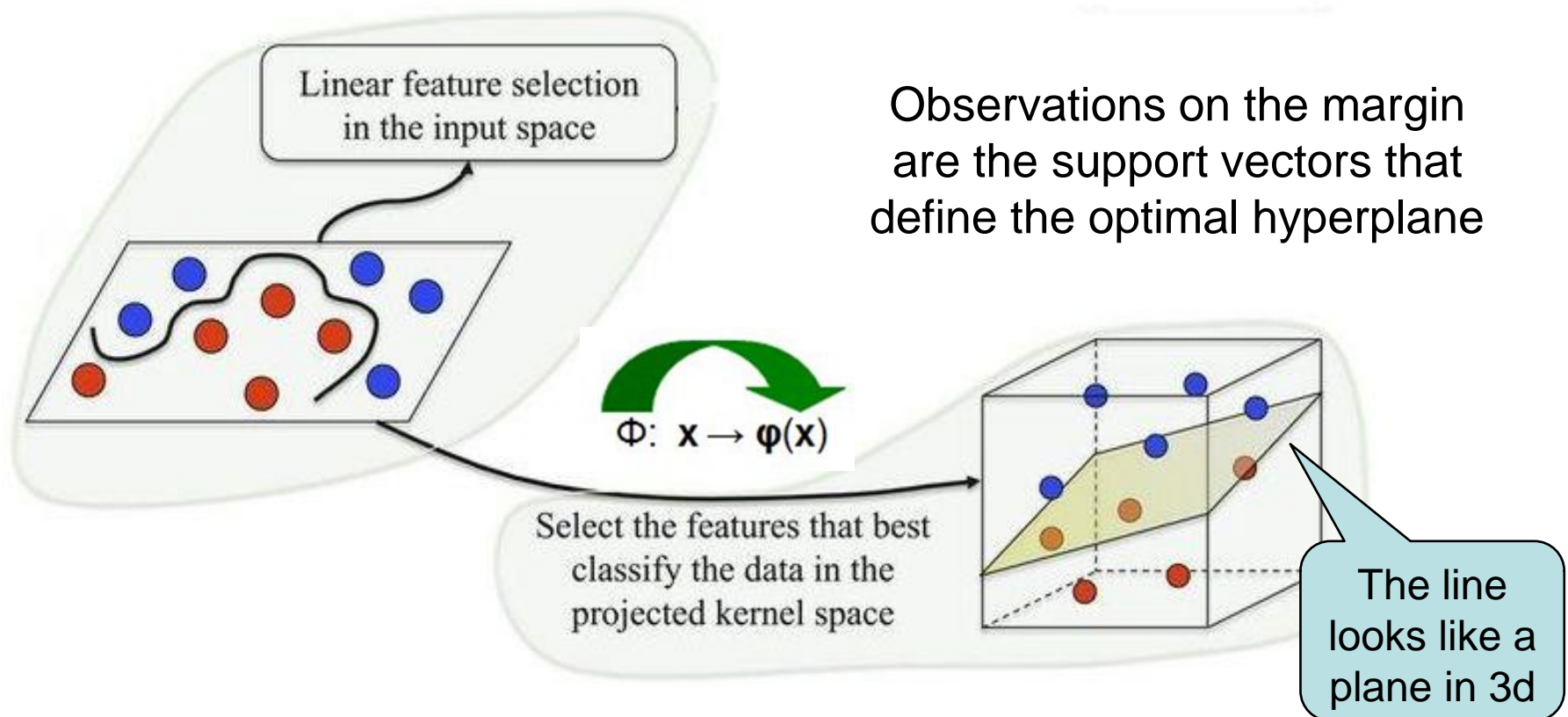
# SVM Kernel

- Kernelization maps non-linear data into another space which may either be of same dimension or higher dimension to improve the linear separability.
- [Scikit Learn Documentation](#)



# SVM Kernel

- Kernelization is helpful if the right kernel can be found.
- However, finding the suitable kernel is often a challenge.



# SVM



[https://youtu.be/\\_YPScrckx28](https://youtu.be/_YPScrckx28)

Detailed Explanation (with Math): <https://youtu.be/ny1iZ5A8ilA>

# Non-Linear Algorithms

- Nonlinear machine learning algorithms do not make strong assumptions about the form of the mapping function
- By not making assumptions, they are free to learn any functional form from the training data.
- Nonlinear methods may achieve better accuracy at the expense of requiring more data and training time.

# Naïve Bayes

- The Naïve Bayes model is comprised of two types of probabilities that can be calculated directly from the training data:
  - The probability of each class.
  - The conditional probability for each class given each x value
- Once calculated, the probability model can be used to make predictions for new data using Bayes Theorem.
- It is common to assume a Gaussian distribution (bell curve)
- Naive Bayes is called naive because it assumes that each input variable is independent.
  - This is a strong assumption and unrealistic for real data
  - Nevertheless, the technique is very effective on a large range of complex problems.
- [Scikit Learn Documentation](#)

# Naïve Bayes

- The Naïve Bayes model is comprised of two types of probabilities that can be calculated directly from the training data:
  - The probability of each class.
  - The conditional probability for each class given each x value

$$P(Y|X) = \frac{P(X \text{ and } Y)}{P(X)}$$

# Naive Bayes

## Guassain Naive Bayes....

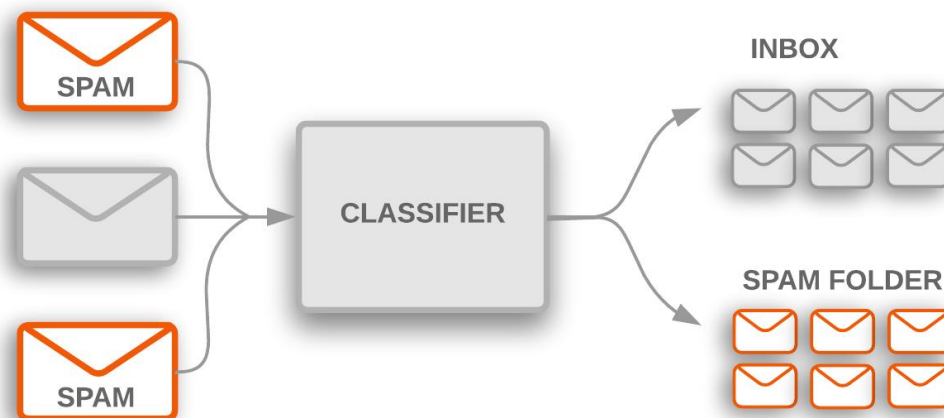


## ...Clearly Explained!!!

<https://youtu.be/H3EjCKtIVog?t=47>

# Applications of NB

- Naive Bayes is mostly applied to:
  - **real time classification** - the speed of Naive Bayes allows it to classify things in real time
  - **text classification** - Naive Bayes functions well for text classification. It is popularly used for categorizing news, sentiment analysis, and to identify spam email.

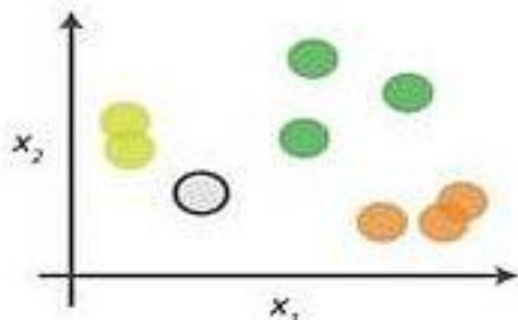


# k-Nearest Neighbors

- The KNN algorithm is simple and effective but can be very slow
- The model representation for KNN is the entire training dataset. This approach is known as a *lazy learner*
- Predictions are made for a new data point by searching through the entire training set for the K most similar instances (the neighbours) and summarizing the output variable for those K instances.
- The trick is in how to determine the similarity between the data instances. The simplest technique if your attributes are all of the same scale (all in inches for example) is to use the Euclidean distance, a number you can calculate directly based on the differences between each input variable.

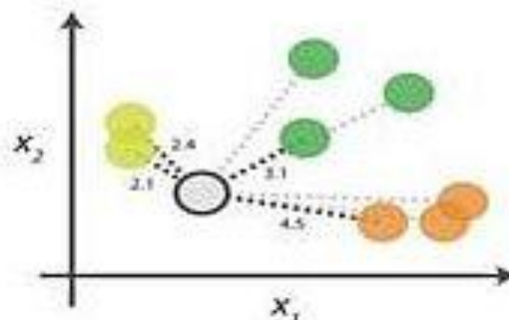
# kNN Algorithm

## 0. Look at the data











Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

## 1. Calculate distances









Start by calculating the distances between the grey point and all other points.

## 2. Find neighbours

Point Distance			
	.. 	2.1	→ 1st NN
	.. 	2.4	→ 2nd NN
	.. 	3.1	→ 3rd NN
	.. 	4.5	→ 4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

## 3. Vote on labels

Class	# of votes	
	2	→ Class  wins the vote! Point  is therefore predicted to be of class  .
	1	
	1	

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

# k-Nearest Neighbors



<https://youtu.be/0p0o5cmgLdE>

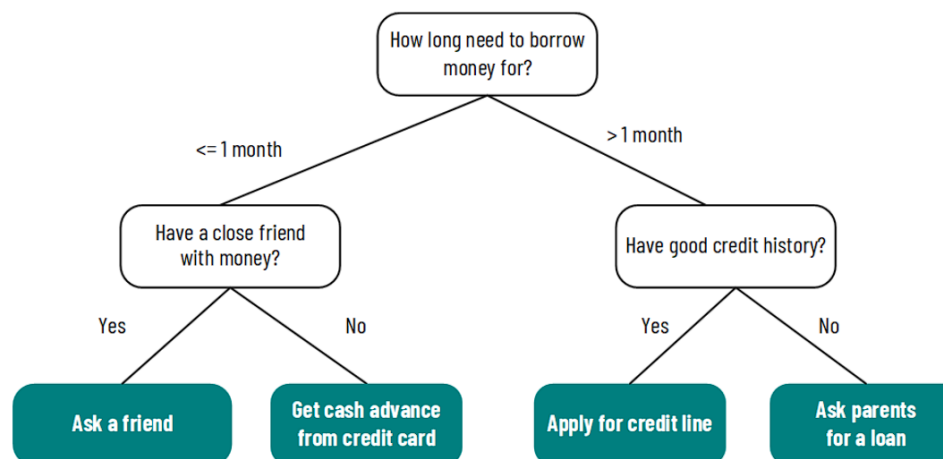
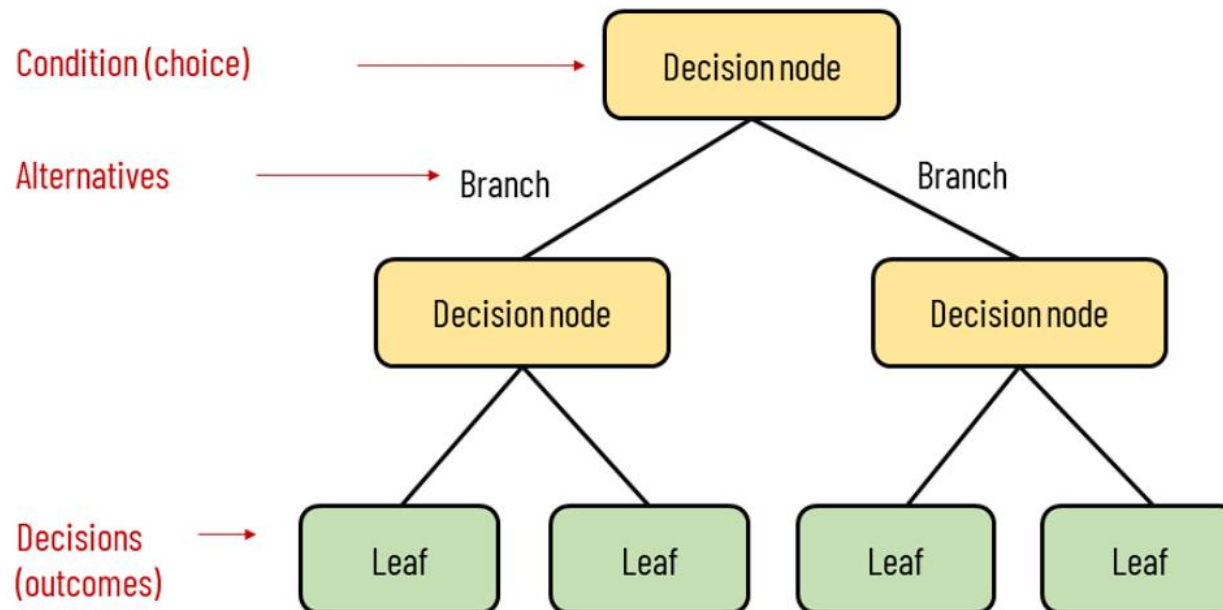
# KNN

- KNN can require a lot of memory or space to store all of the data, but only performs a calculation (learns) when a prediction is needed, just in time.
- How to choose a good value of  $k$  for a particular dataset? Try several different values and compute the error rate using **cross-validation**.
- The idea of distance or closeness can break down in very high dimensions (lots of input variables) which can negatively affect the performance of the algorithm on your problem. This is called *the curse of dimensionality*. It means you need to only use the input variables that are most relevant to predicting the output variable (**feature selection**).
- [Scikit Learn Documentation](#)

# Decision Trees

- The representation of the decision tree model is a binary tree, straight from the algorithms and data structures textbook.
- Each node represents a single input variable ( $x$ ) and a split point on that variable (assuming the variable is numeric).
- The leaf nodes of the tree contain an output variable ( $y$ ) which is used to make a prediction. Predictions are made by walking the splits of the tree until arriving at a leaf node and output the class value at that leaf node.
- [Scikit Learn Documentation](#)

# Decision Trees

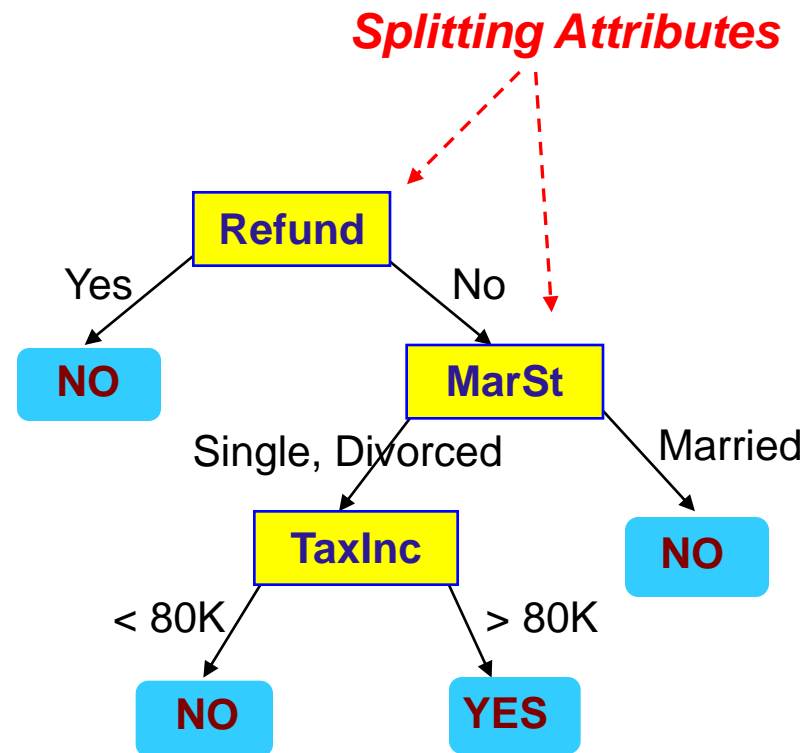


# Example 1 - Decision Tree

*categorical*  
*categorical*  
*continuous*  
*class*

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

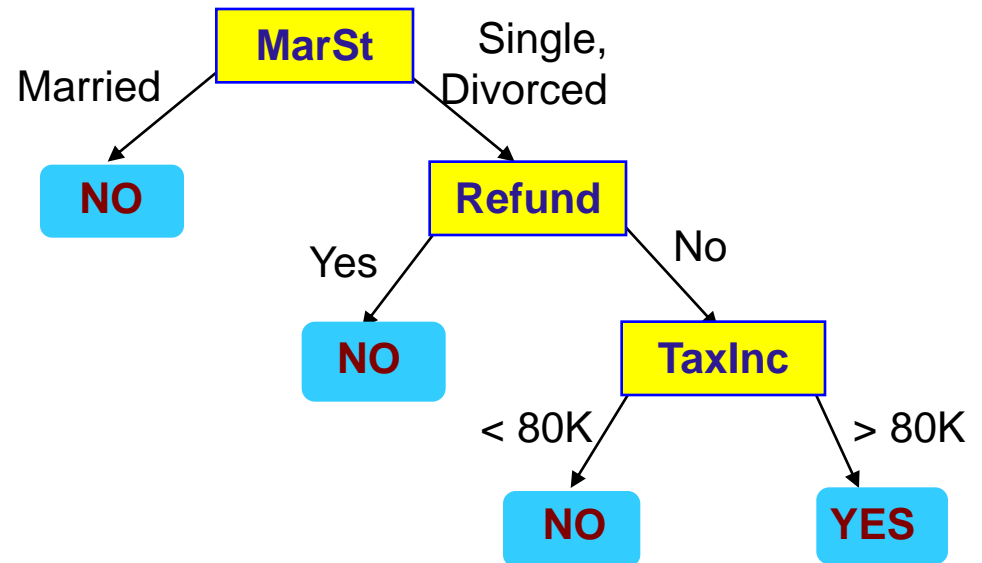


Model: Decision Tree

# Example 2 - Decision Tree

*categorical*  
*categorical*  
*continuous*  
*class*

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



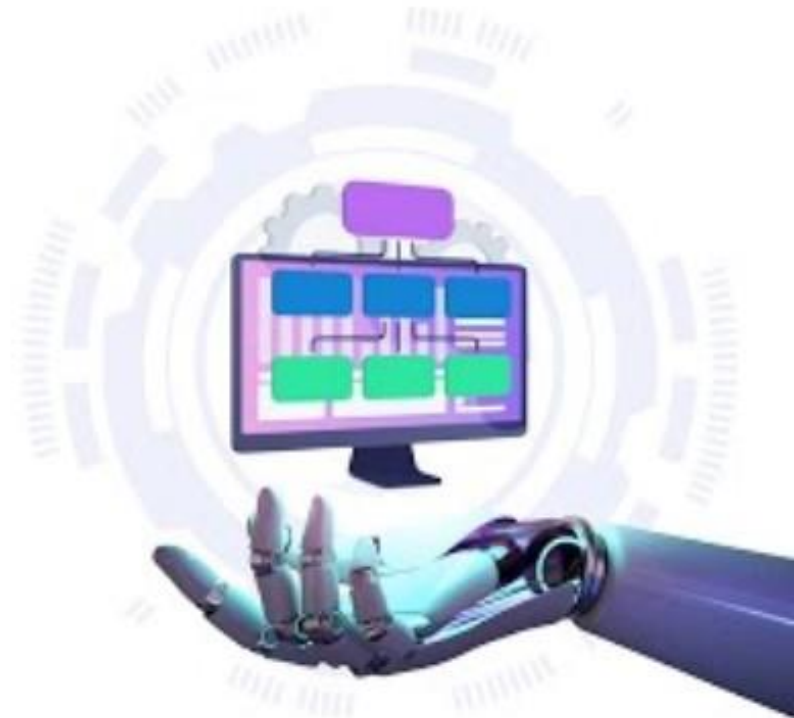
**There could be more than one tree that fits the same data!**

# Decision Tree

datasciencedojo  
— data science for everyone —

## DECISION TREES

FOR MACHINE LEARNING

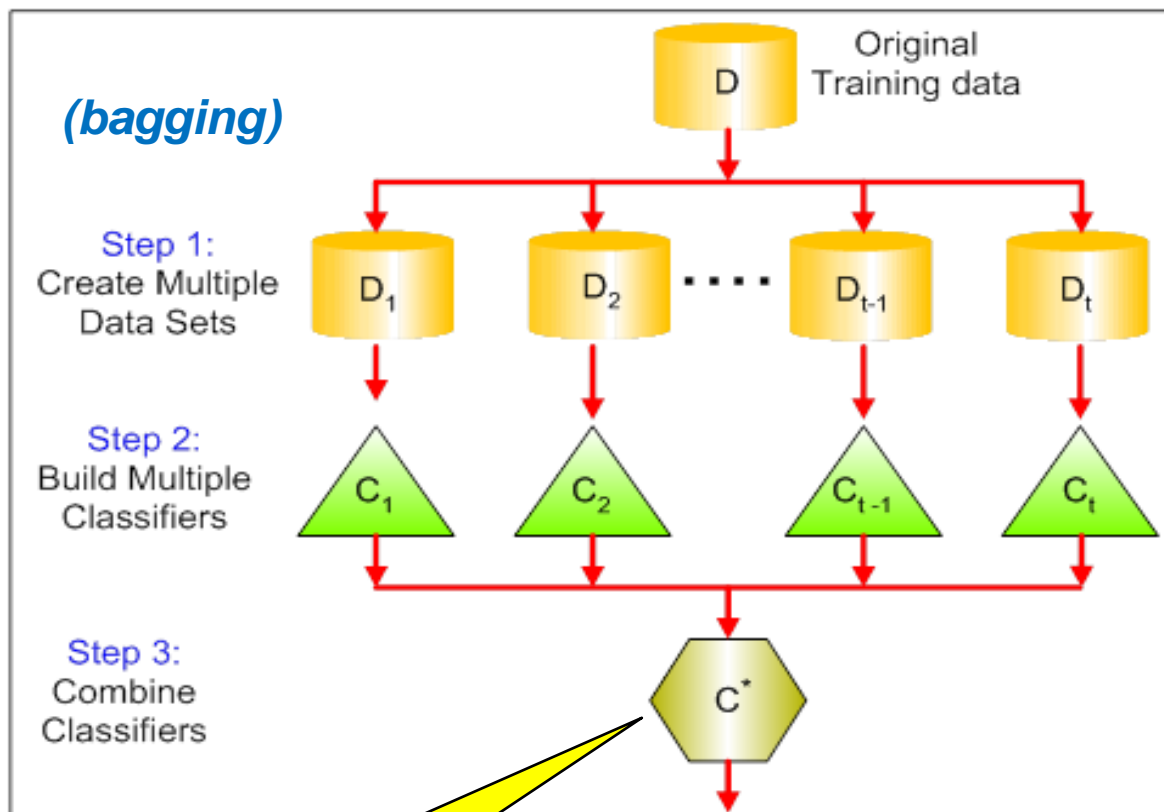


<https://youtu.be/7YTeMIq6-9s>

# Generic Tree Growing Algorithm

1. Start with entire dataset and create a root node.
2. Select the best attribute – the one that best splits data based on a chosen metric.
3. Split the dataset into subsets based on selected attribute.
4. For each subset, create a child node and assign subset to this node.
5. Repeat recursively Steps 2 – 4 until one of the stopping criteria is met.
6. Assign most common class label of the instances in the node to the leaf node.

# Ensemble Methods



**Voting  
Algorithm**

**Goal is to reduce the  
variance of predictions**

**Use different  
sub-samples of  
the data set**

**Combine results  
from multiple  
instances of a  
classifier**

# Ensemble Methods

- Use different sub-samples of the data set
- Combine results from multiple instances of a classifier
- Goal is to reduce the variance of predictions
- In **Bagging**, data subsets are drawn randomly with replacement from the training dataset at the beginning
- In **Boosting**, every new subset comprises the elements that were misclassified by previous models
- In **Bagging**, classifiers run in parallel, and results are aggregated when all of them finish
- In **Boosting**, classifiers run in sequence, and results are aggregated at the end of each run

# Random Forest

- Random Forest is a bagging method that uses a large number of deep trees, fitted on bootstrap samples
- The Random forest algorithm adds the concept of random feature subspace selection to create more robust models:
  - In addition to sampling the observations in the dataset we also sample the features, and keep only a random subset of them to generate each bootstrap sample
- This means all trees use different information to make their decisions, which reduces the correlation between their outputs
- [Scikit Learn Documentation](#)

# Random Forest

Training dataset

$X_1$	$X_2$	$X_3$	$X_4$	$Y$
a1	b1	c1	d1	1
a2	b2	c2	d2	2
a3	b3	c3	d3	1
a4	b4	c4	d4	1
a5	b5	c5	d5	2

Bootstrap

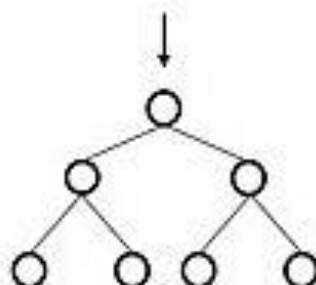
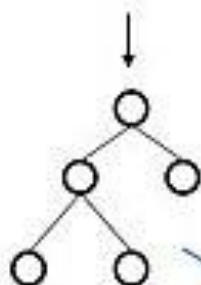
$X_1$	$X_3$	$X_4$	$Y$
a1	c1	d1	1
a2	c2	d2	2
a5	c5	d5	2

$X_2$	$X_3$	$X_4$	$Y$
b1	c1	d1	1
b3	c3	d3	1
b4	c4	d4	1

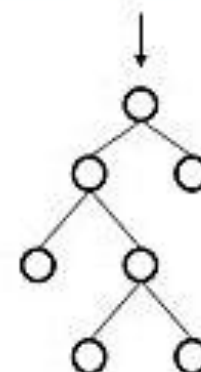
...

$X_1$	$X_2$	$Y$
a2	b2	2
a3	b3	1
a5	b5	2

Ensemble of trees



...



Aggregation

Majority decision

# Random Forest

## Random Forests Part 1...

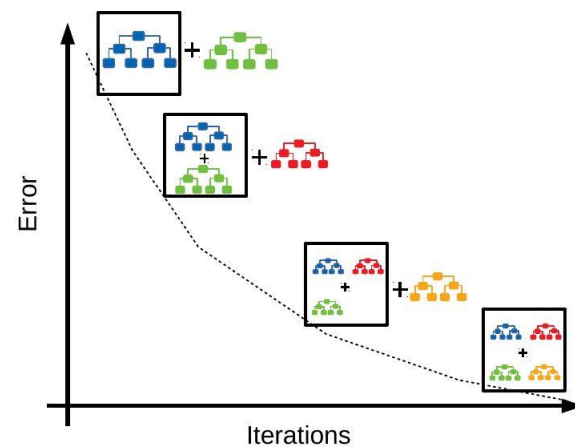
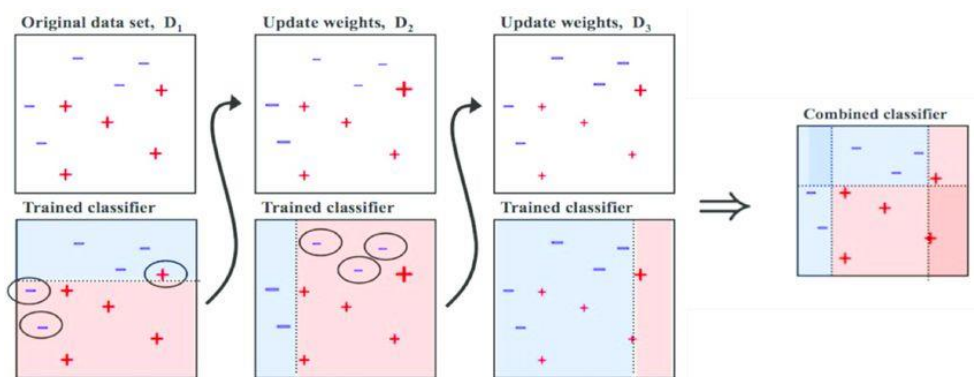


**Building, using and evaluating,  
clearly explained!!!**



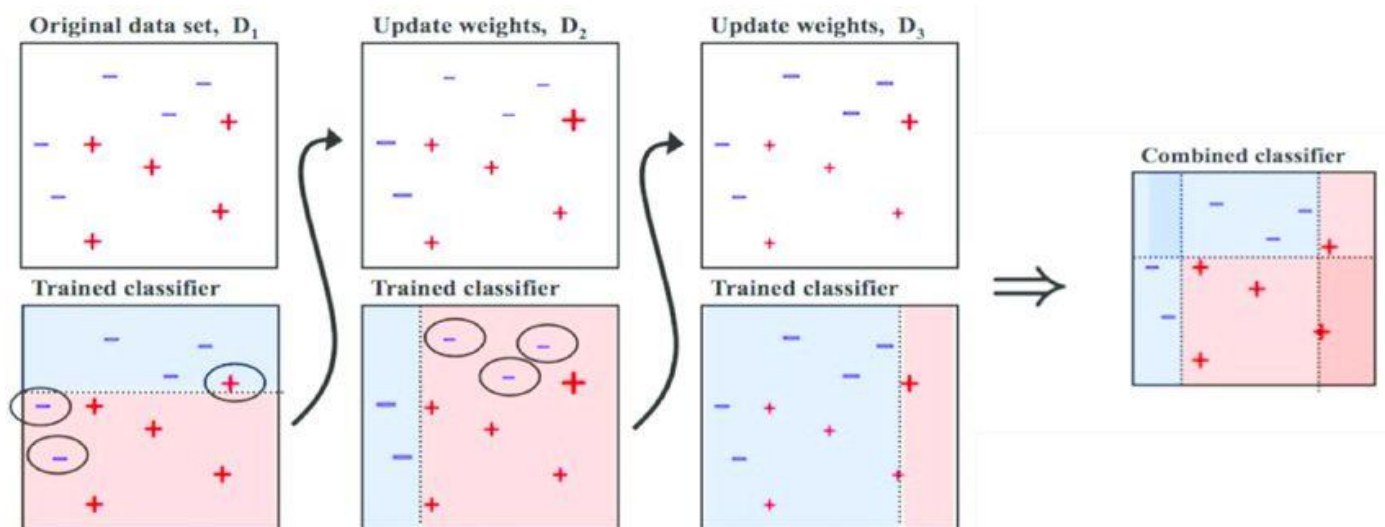
[https://youtu.be/J4Wdy0Wc\\_xQ?t=9](https://youtu.be/J4Wdy0Wc_xQ?t=9)

- **Boosting classifiers** aggregate the individual predictions of several instances of an estimator on repeatedly modified versions of the data.
- Boosting starts with one instance, and adds new instances one after another
- Each model in the sequence is fitted giving more importance to observations in the dataset that were misclassified by the previous models in the sequence
- **This is done based on the the *loss function* and *weighting***



# AdaBoost (Adaptive boosting)

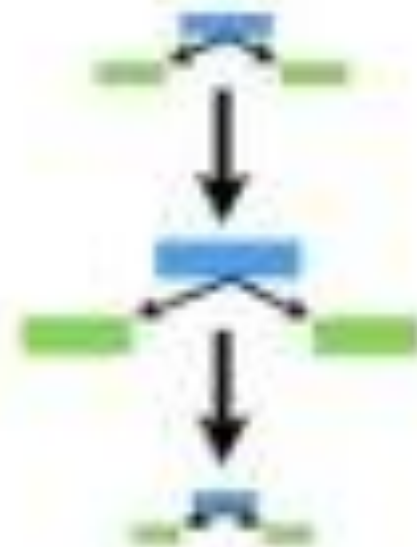
1. Train model on train set with equal weights for all samples
2. Compute error of model on train set
3. Increase weights on train cases model gets wrong
4. Train new model on re-weighted train set
5. Go back to step 2 - Repeat until tired (100+ iterations)



- [Scikit Learn Documentation](#)

# AdaBoost (Adaptive boosting)

**AdaBoost....**

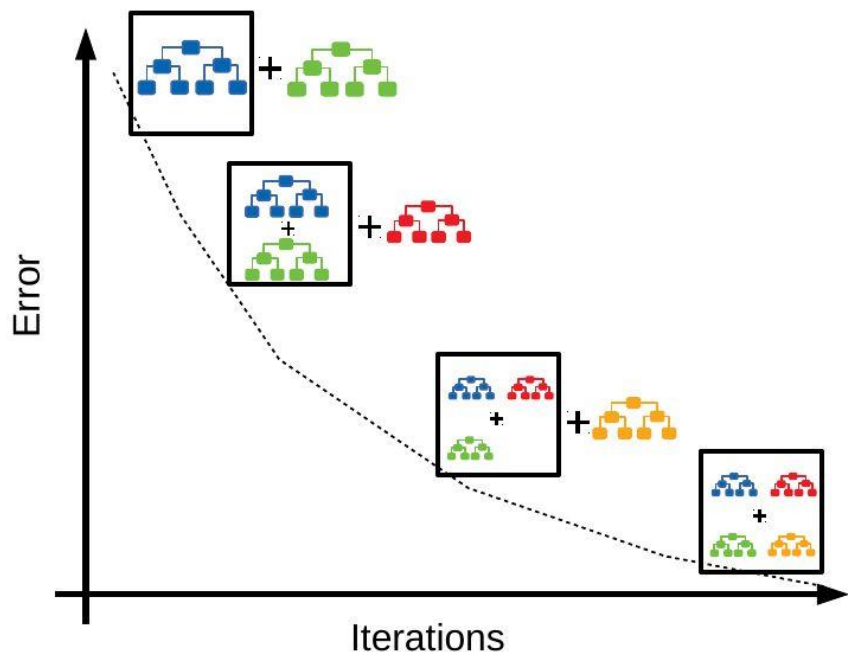


**...Clearly Explained!!!**

<https://youtu.be/LsK-xG1cLYA?t=11>

# Gradient Boosting

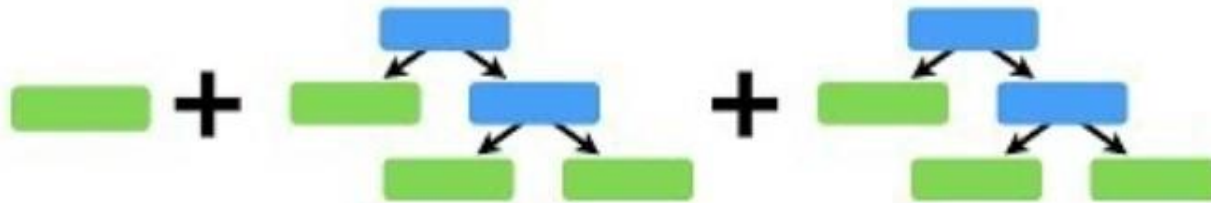
- AdaBoost was the first really successful boosting algorithm developed for binary classification
- Actually uses short decision trees
- Gradient boosting is similar to AdaBoost but uses a different process to fit the trees sequentially



1. Construct a base tree
  2. Build the next tree based on errors of the previous tree
  3. Combine the previous tree with new tree
  4. Go back to step 2 - Repeat until tired (100+ iterations)
- [Scikit Learn Documentation](#)

# Gradient Boosting

## Gradient Boost Part 1...

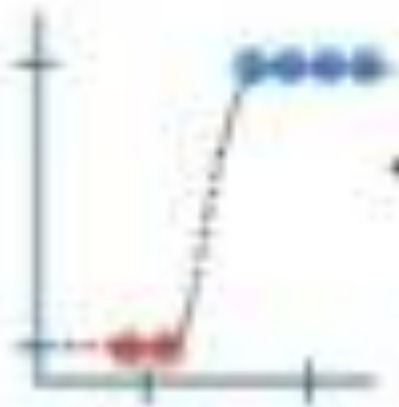


**...Regression  
Main Ideas!!!**

<https://youtu.be/3CC4N4z3GJc?t=10>

# Gradient Boosting

## Gradient Boost Part 3...

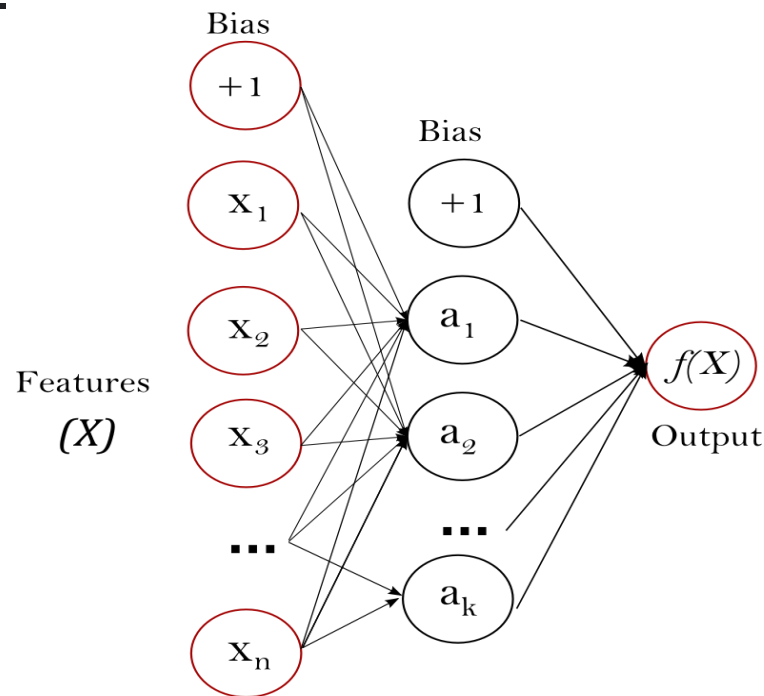


**...Classification  
Main Ideas!!!**

<https://youtu.be/jxuNLH5dXCs?t=17>

# Neural Network

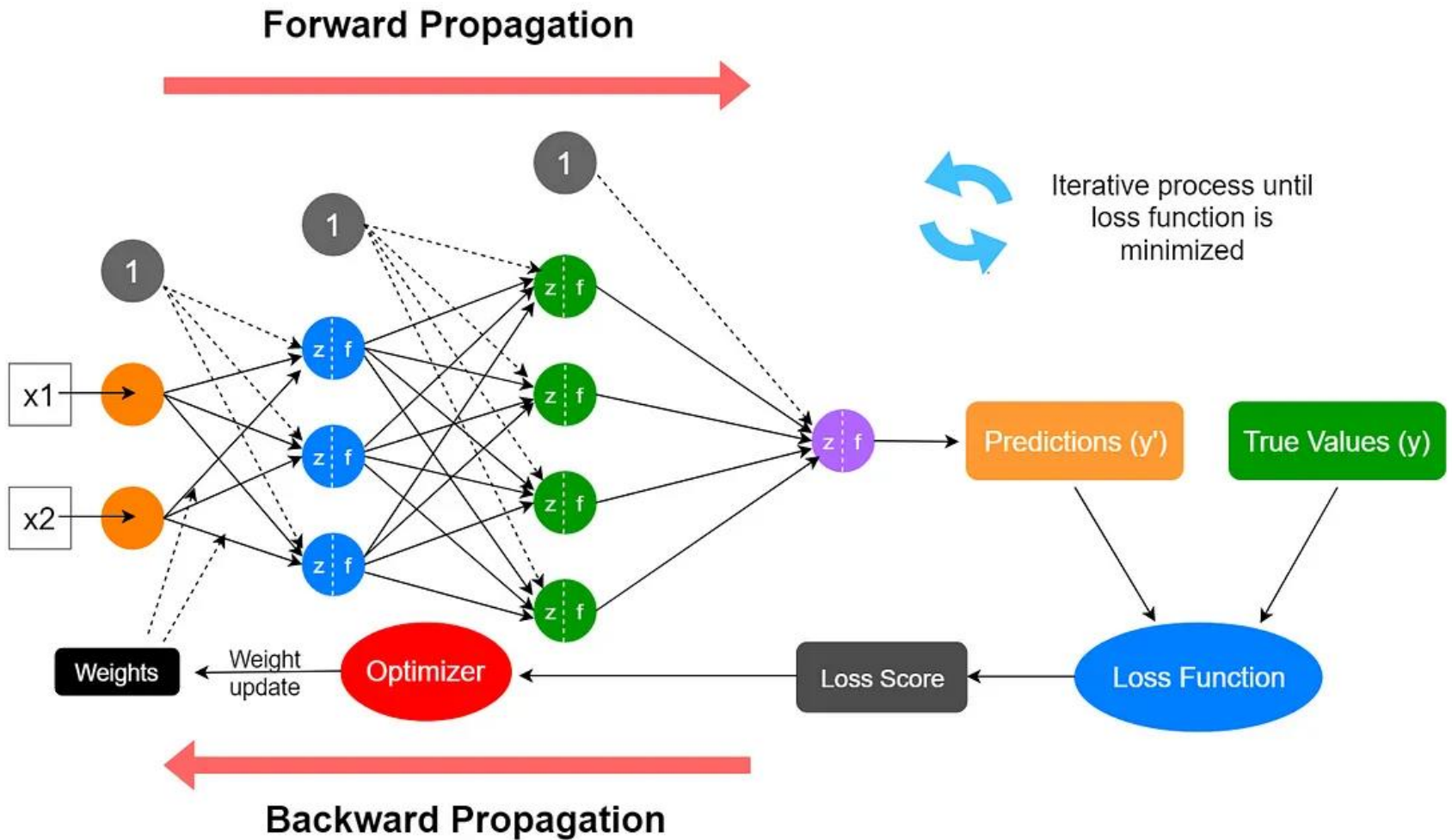
- Neural networks are complex models, which try to mimic the way the human brain develops classification rules. A neural net consists of many different layers of neurons, with each layer receiving inputs from previous layers, and passing outputs to further layers.



# Neural Network

- The learning (training) process of a neural network is an iterative process in which the calculations are carried out forward and backward through each layer in the network until the loss function is minimized.
- Fundamentally, three steps are involved:
  1. Forward propagation (Forward pass)
  2. Calculation of the loss function
  3. Backward propagation (Backward pass/Backpropagation)
- [Scikit Learn Documentation](#)

# Neural Network



# Neural Network



<https://www.youtube.com/watch?v=bfmFfD2Rlcg>

# Neural Network

## Neural Networks Clearly Explained!!!



## Look inside the black box!!!

<https://www.youtube.com/watch?v=CqOfi41LfDw&t=1s>

# Estimator Selection

- No *a-priori* best estimator; must experiment
- The estimator selection and evaluation phases provide information that can be used to refine feature selection, transformation, and scaling.
- In practice, this means starting off by generating predictions with a set of candidate algorithms, comparing their performance in terms of predictive accuracy, and making judgements which one have the most potential.

# Review Questions

1. What are the four types of supervised learning algorithms?
2. How each supervised learning algorithms are trained?

# Summary / Recap of Main Points

At the end of this topic, you should be able to:

1. Understand different types of supervised learning algorithms
2. Understand the operating details of each algorithms.