

cluster 2018

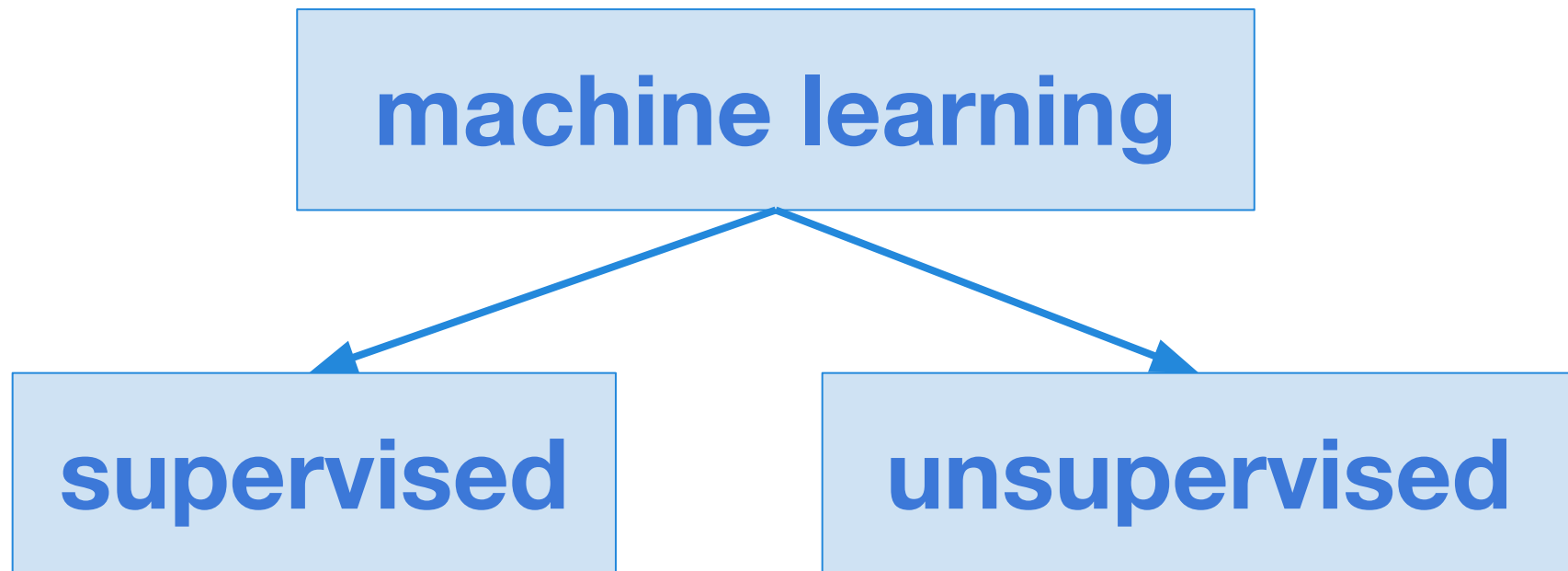
ciencia de datos en ingeniería industrial

clase_03: clasificación

agenda clase03: aprendizaje supervisado

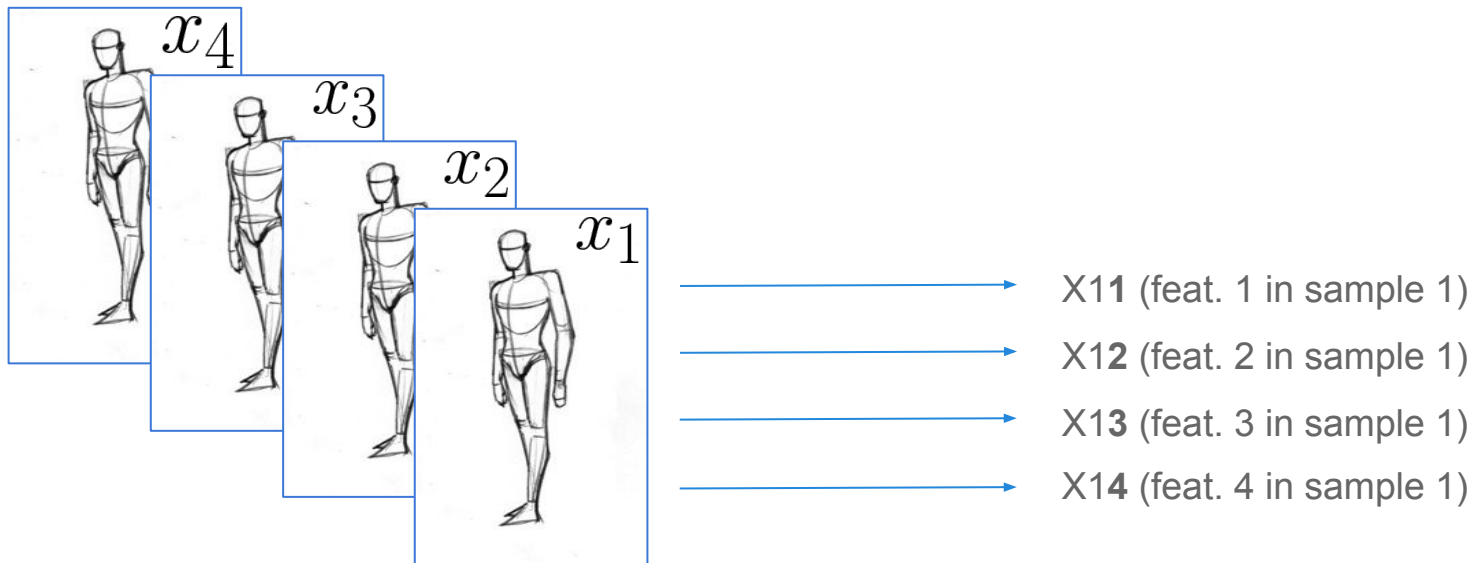
- Clasificación, binary class, multiclass
- Train, validation, test
- Cross validation
- Grid Search
- Confusion Matrix
- Performance metrics (Sens, Spec, ROC)
- Variance-bias trade off, overfitting
- Learning curve
- SVM
- KNN
- Logistic Regression

learning approaches



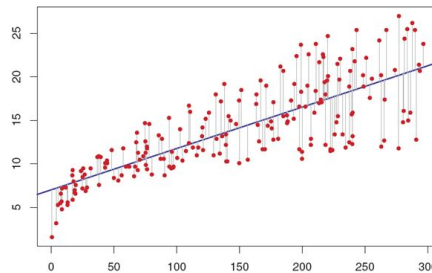
Samples and Features

$$X_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, \dots, x_{in})$$



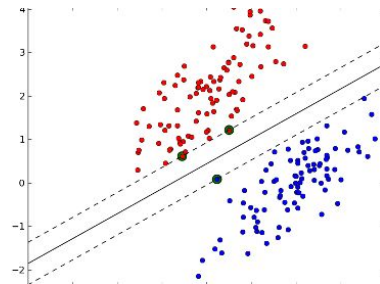
supervised learning methods

regression



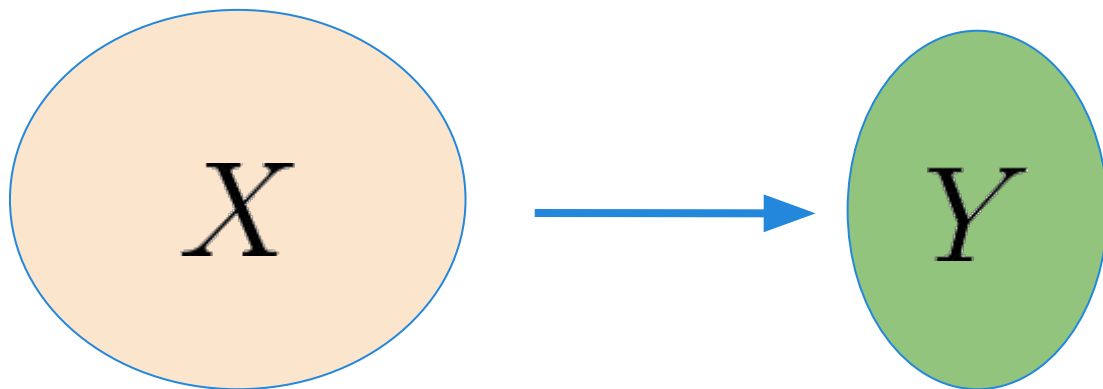
Y is continual

classification



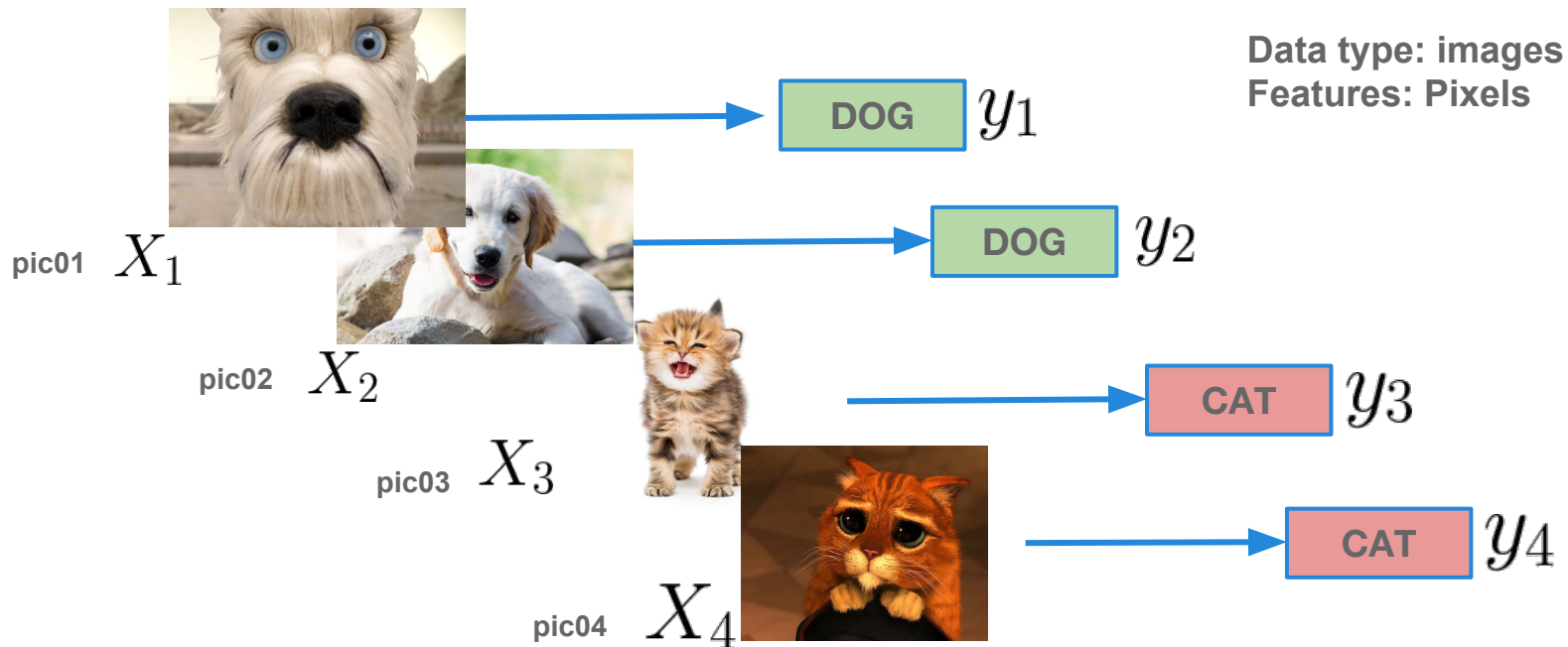
Y is categorical

supervised learning



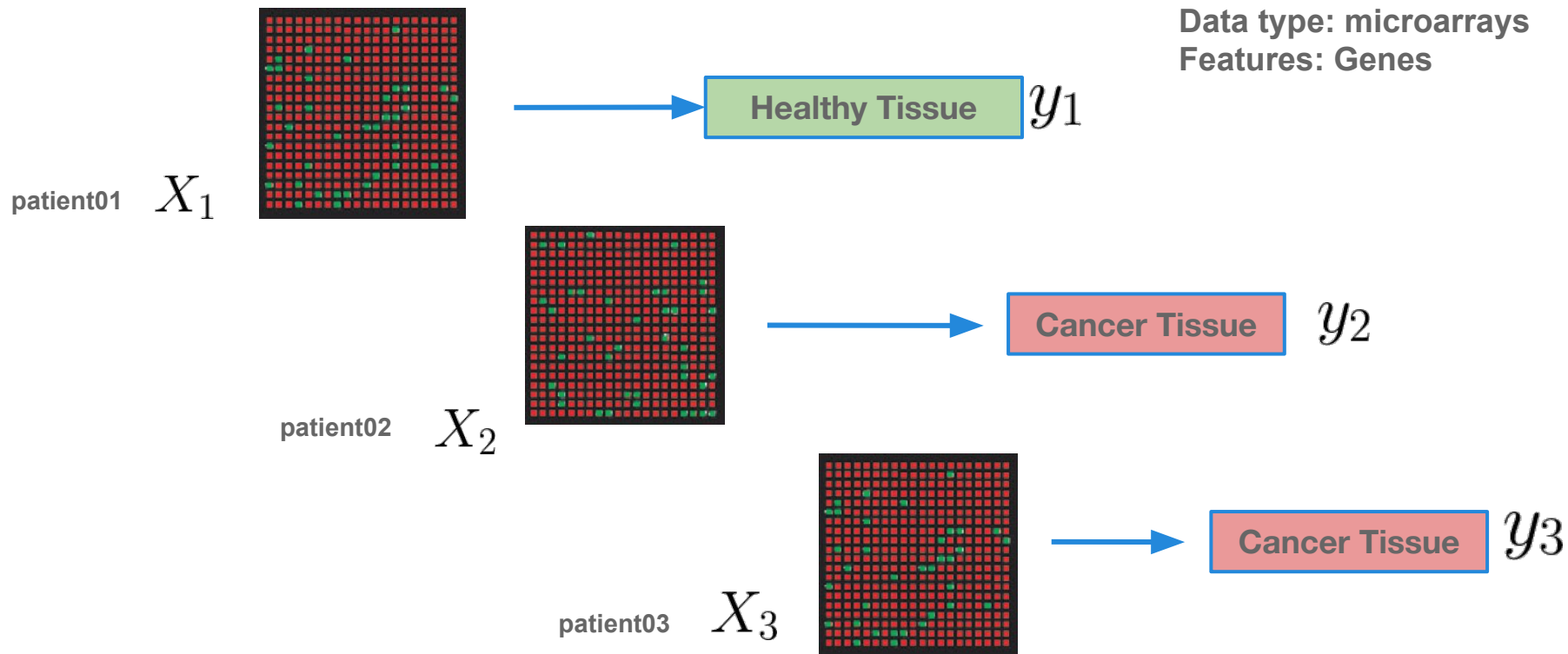
- We assume Y variable depends on X .
- What we do not know is the true function/rule $y = f(x)$.

supervised learning: classification

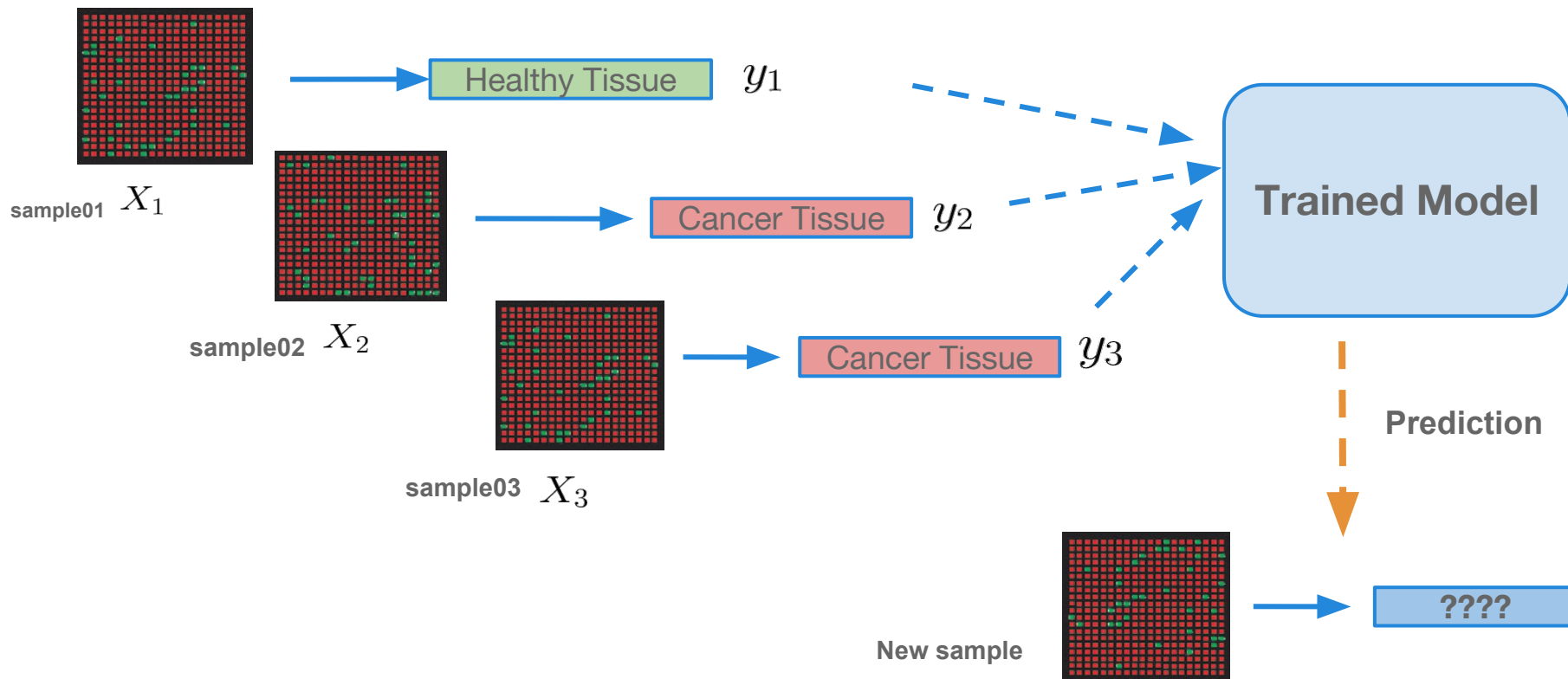


Each sample has associated a label settled by a human user.

supervised learning: classification

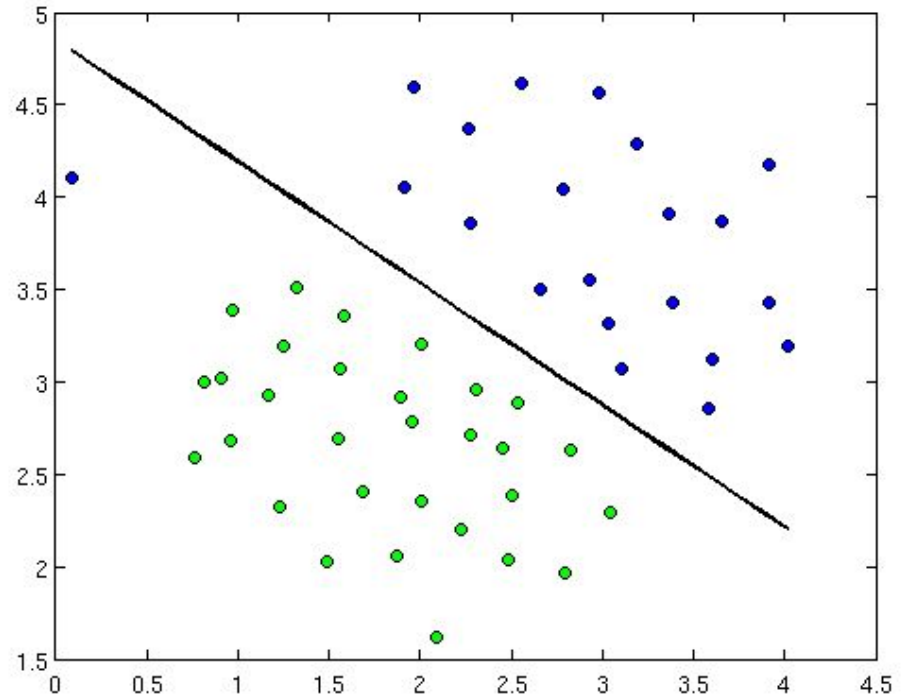


supervised learning: classification



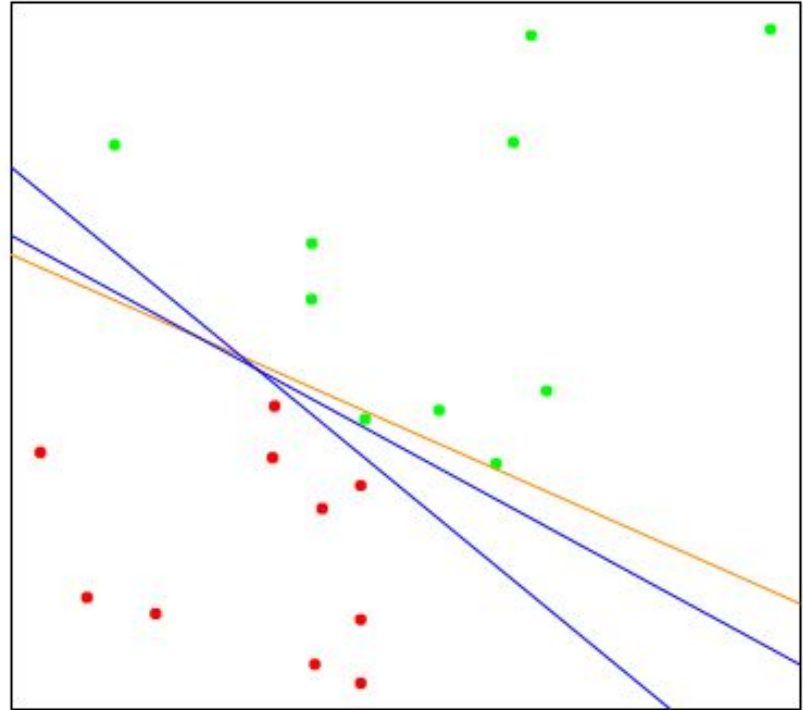
Funciones discriminantes

- Una función discriminante toma un vector input X (sample) con “ n ” features, y le asigna una de las K clases, llamada C_k .
- Cuando $C_k = 2 \rightarrow$ binaria
- Cuando $C_k > 2 =$ multiclase



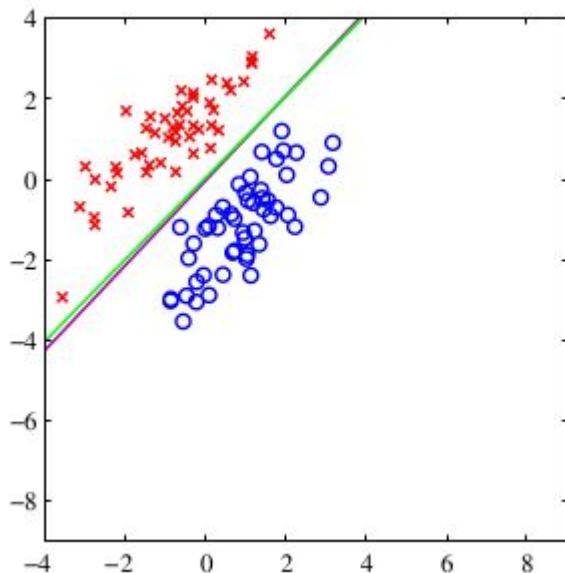
Funciones de Decisión

Para un mismo set de datos etiquetados, distintos modelos pueden generar distintas funciones de decisión (decision rules).

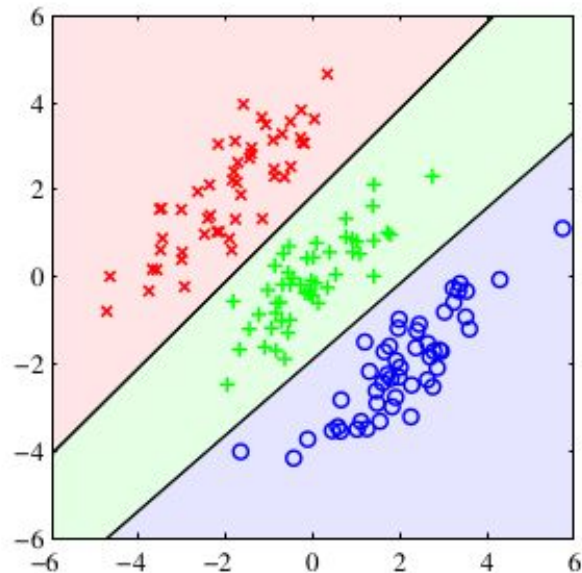


Tipos de clasificación

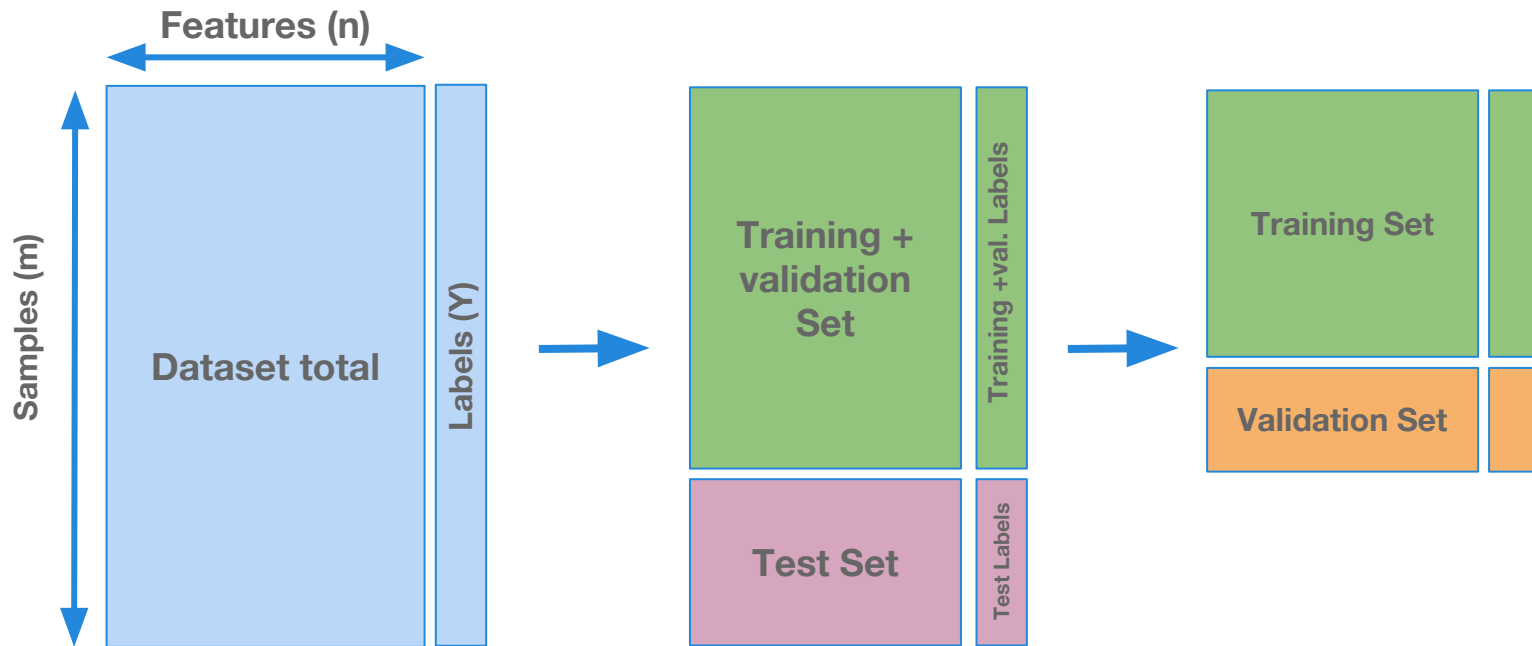
Clasificación Binaria



Clasificación Multiclase



Train, Validation, Test sets.

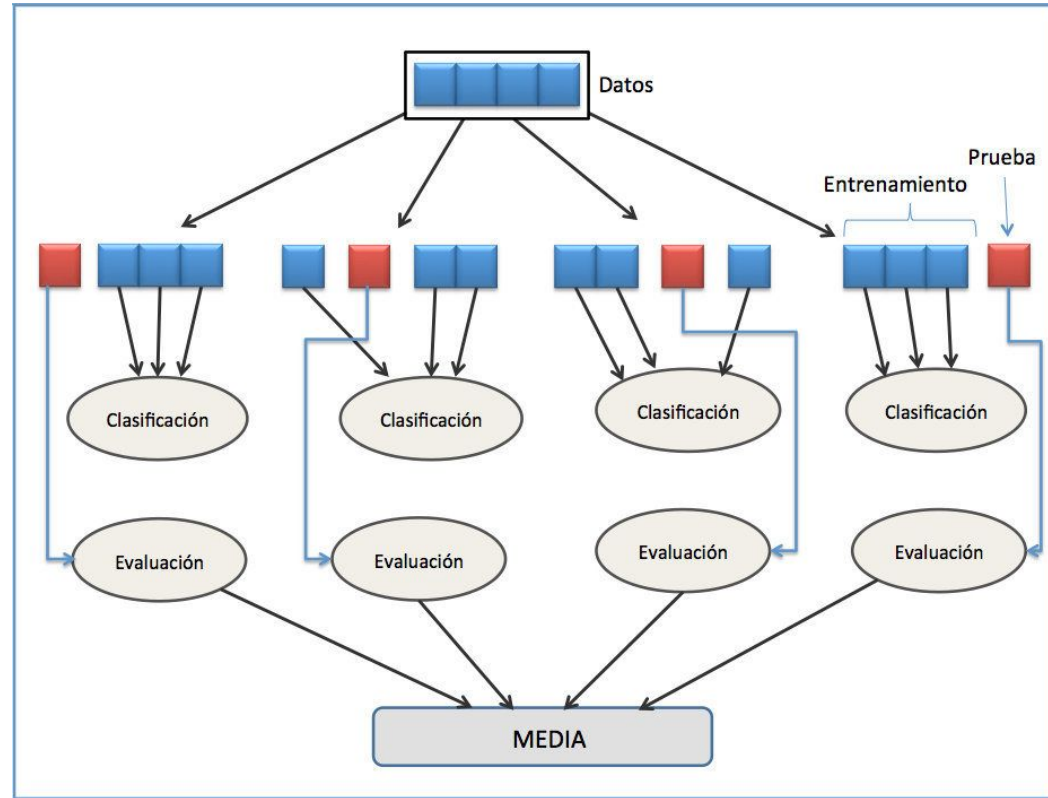


El clasificador aprenderá la regla de decisión utilizando el train set (samples + labels). Luego clasificará las muestras de test (sin mirar las labels de test) y se medirá la exactitud de clasificación en testeo.

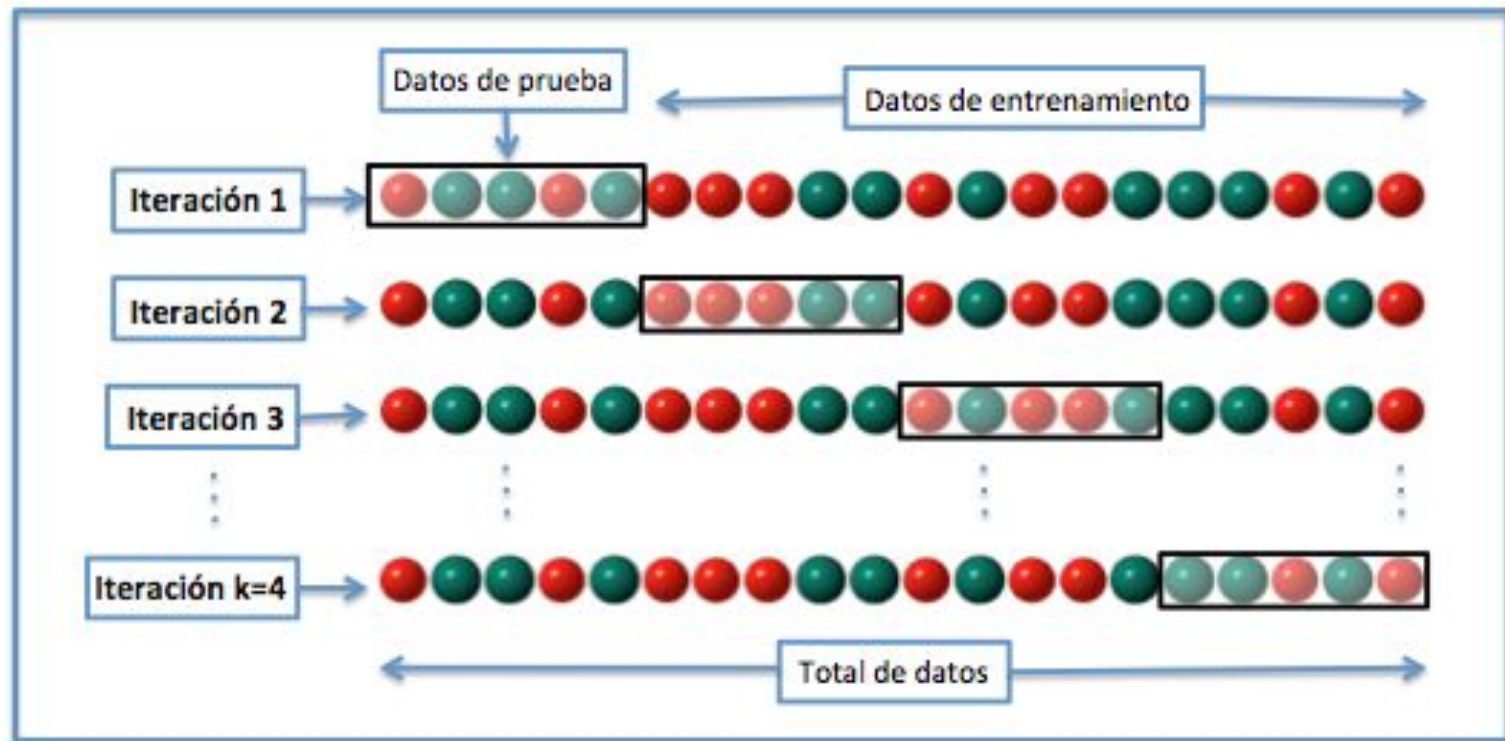
Cross - Validation en training set

Cross validation (CV) se realiza con las muestras de entrenamiento. Consiste en dividir nuestro training set en K folds (K porciones) e iterar K veces. En cada iteración, una porción se utiliza como test y el resto como train. En cada iteración se evaluará el resultado de clasificación y luego se realizará un promedio de todas las iteraciones.

El objetivo es asegurar que nuestro modelo no realiza **overfitting**.

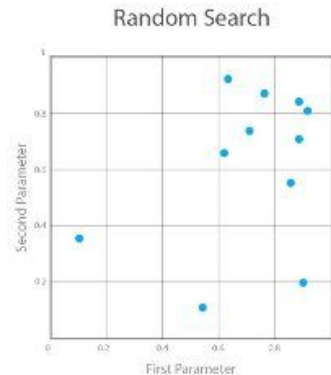
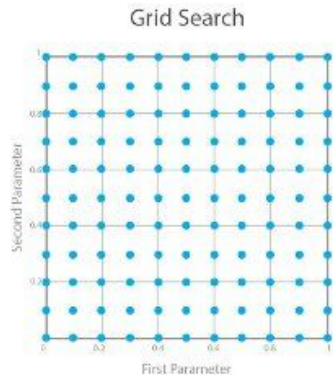
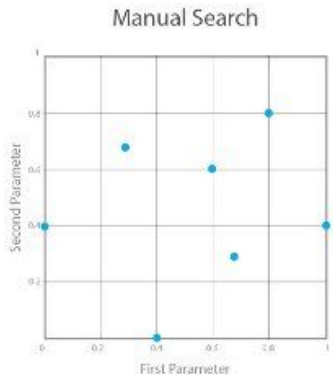


Cross Validation



Grid Search

- Los modelos de clasificación que utilizaremos consistirán de **hiperparámetros** que el usuario debe seleccionar. Estos determinarán la regla de decisión y por ende la performance del modelo.
- Para saber qué hiperparámetros seleccionar, lo que haremos es generar una lista de los mismos y probaremos todas las combinaciones posibles de ellos.
- La combinación que mayor Cross-Validation y Train Accuracy genere es la que usaremos para testear el modelo.



Pipeline: Train, Validate, Test Model

**Dividir
Train y
Test**

**Cross Validation &
Grid Search con
Train Set (utilizando
Xtrain e Ytrain)**

**Selección del
mejor modelo**

**Clasificar muestras
de Test (xtest) sin
mostrarle al modelo
las Ytest.**

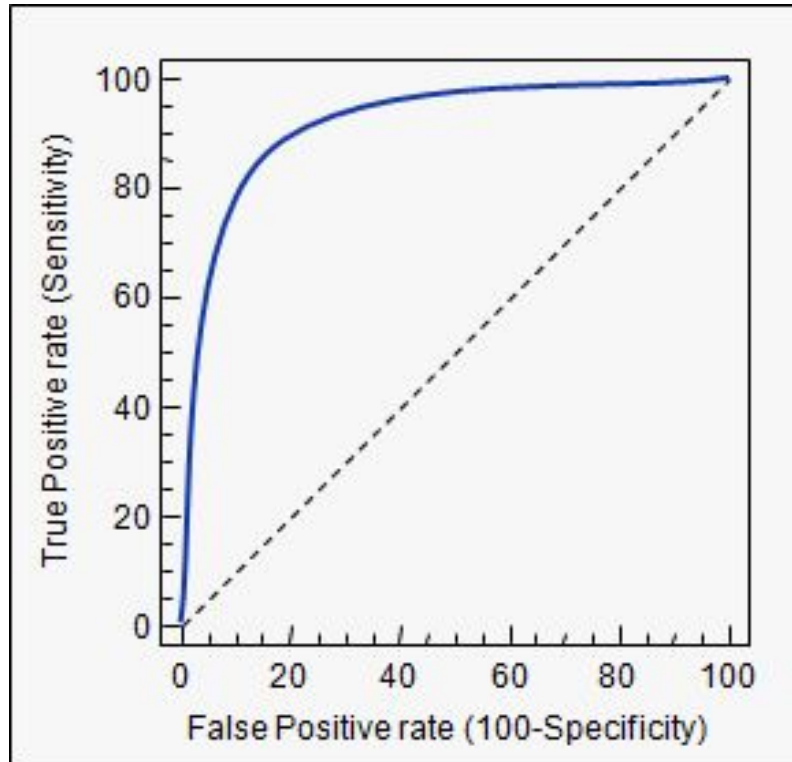
**Evaluar
resultados de
clasificación en
test (comparar
Ypred vs Ytest)**

classification results: confusion matrix

		True Label	
		Class1	Class2
Predicted Label	Class1	True Negative	False Positive
	Class2	False Negative	True Positive

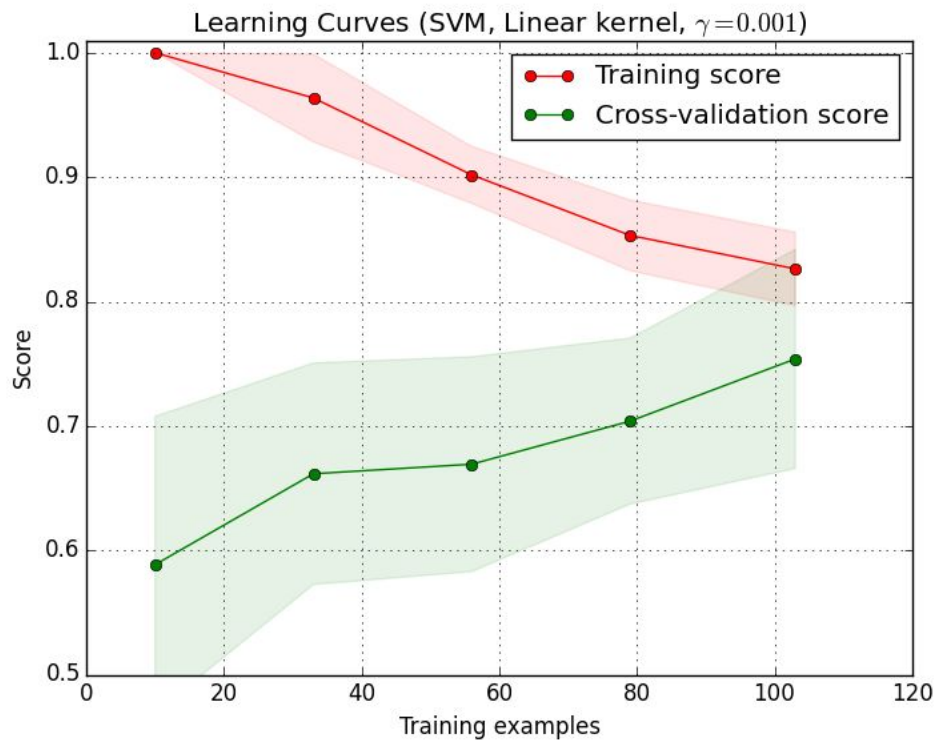
$$\text{Accuracy} = (\text{TN} + \text{TP}) / \text{Total}$$

classification results: AUC ROC

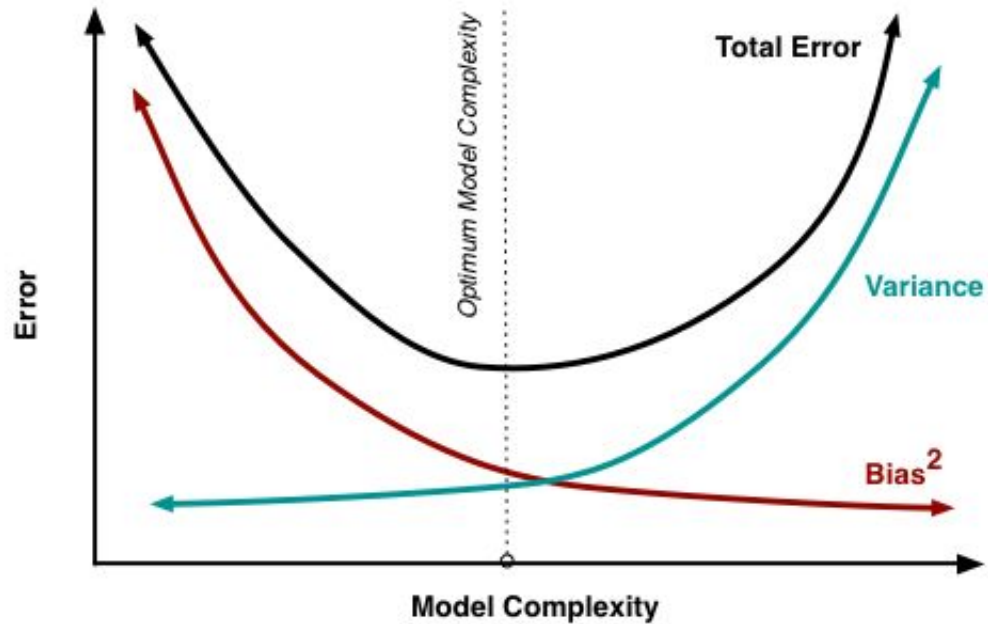


- El área bajo la curva ROC (AUC) da una idea de cuan bueno es mi clasificador independientemente del accuracy.
- Relaciona cuán bien se clasifica ambas clases. Cuanto más cerca de 1 sea el área bajo la curva, mejor mi clasificador.
- Cuanto más cerca de 0.5 sea el AUC, más similar a “arrojar una moneda” sera mi clasificador.

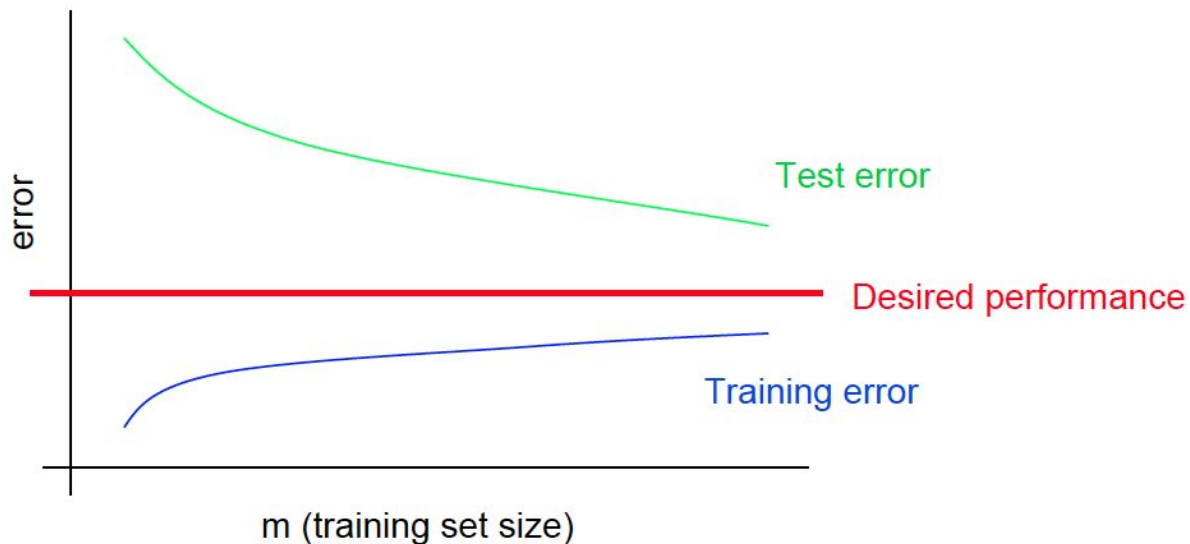
Learning Curve



Variance vs Bias



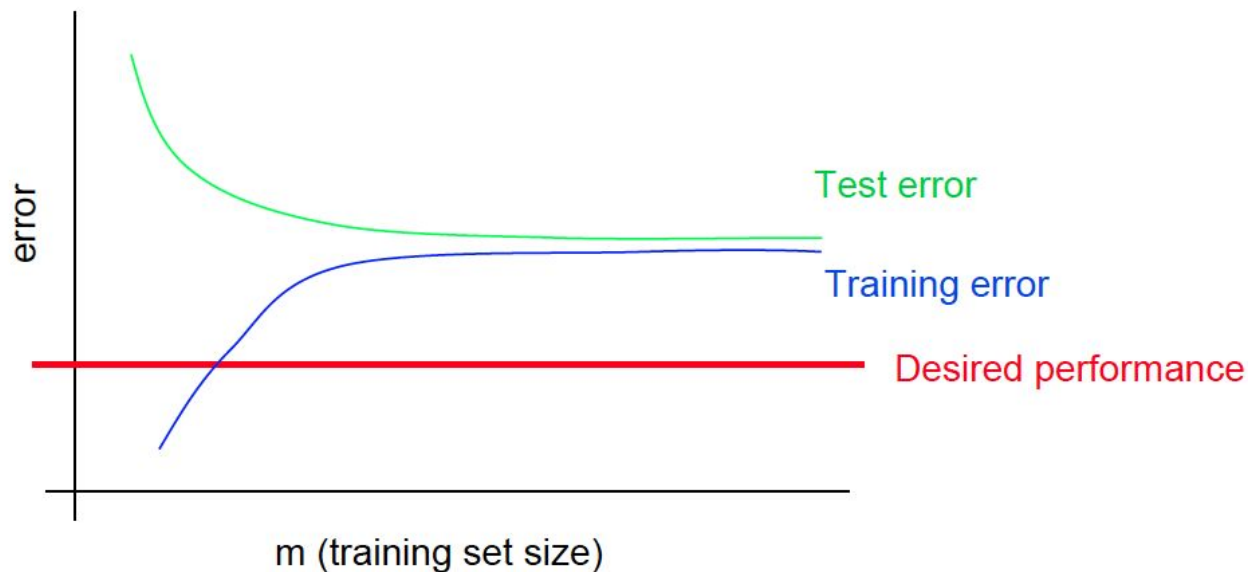
Learning curve: high variance



High variance means lower model complexity (aka, less features)

*Andrew Ng.

Learning curve: high bias



High bias means higher model complexity (aka, too many features)

*Andrew Ng.

Support Vector Machines



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google Custom Search



Previous
3. Kernel ...
Next
15. Stochas...
Up
1. Supervised...

scikit-learn v0.19.2
[Other versions](#)

Please [cite us](#) if you
use the software.

4. Support Vector Machines

4.1. Classification

1.4.1.1. Multi-class classification

1.4.1.2. Scores and probabilities

1.4.1.3. Unbalanced problems

4.2. Regression

4.3. Density estimation, novelty detection

4.4. Complexity

4.5. Tips on Practical Use

4.6. Kernel functions

1.4.6.1. Custom Kernels

1.4.6.1.1. Using Python functions as kernels

1.4.6.1.2. Using the Gram matrix

1.4.6.1.3. Parameters of the RBF Kernel

4.7. Mathematical formulation

1.4.7.1. SVC

1.4.7.2. NuSVC

1.4.7.3. SVR

4.8. Implementation details

1.4. Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for [classification](#), [regression](#) and [outliers detection](#).

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different [Kernel functions](#) can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing [Kernel functions](#) and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see [Scores and probabilities](#), below).

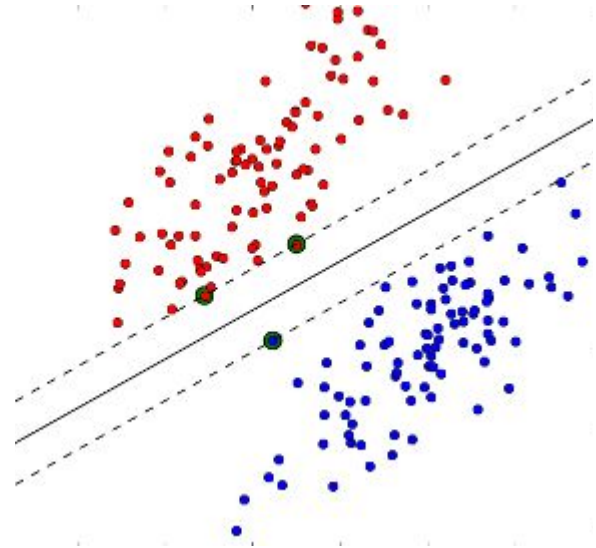
The support vector machines in scikit-learn support both dense (`numpy.ndarray` and convertible to that by `numpy.asarray`) and sparse (any `scipy.sparse`) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered `numpy.ndarray` (dense) or `scipy.sparse.csr_matrix` (sparse) with `dtype=float64`.

1.4.1. Classification

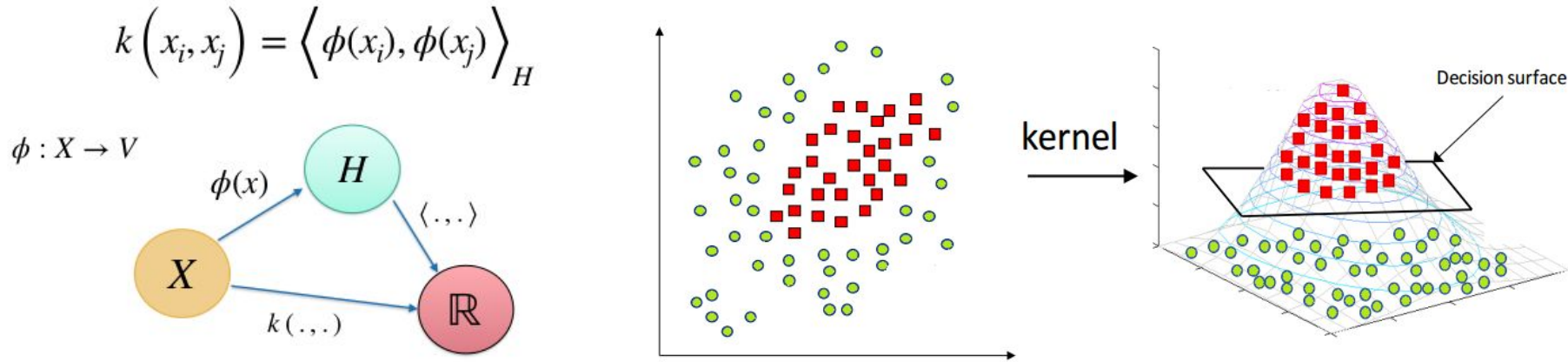
Classification Models: SVM

Support Vector Machines

- Clasificador Lineal
- Busca el hiperplano separador maximizando el margen entre clases.
- Cuando las clases no son separables linealmente se acude al “soft-margin”, penalizador que permite muestras “del otro lado”.
- El margen separador queda definido por “s” muestras. Estas muestras son llamadas support vectors.



Classification Models: SVM “Kernel Trick”



Los kernels son funciones de similitud entre muestras. Mapean nuestros datos a una dimensión desconocida donde son linealmente separables. Cuando usamos SVM, podemos aplicar un kernel para facilitar la clasificación.

SVM: Parametros

- C = “Costo” (todos los kernels)
- Γ = Kernel Gaussiano (RBF)
- Degree = Kernel Polynomial



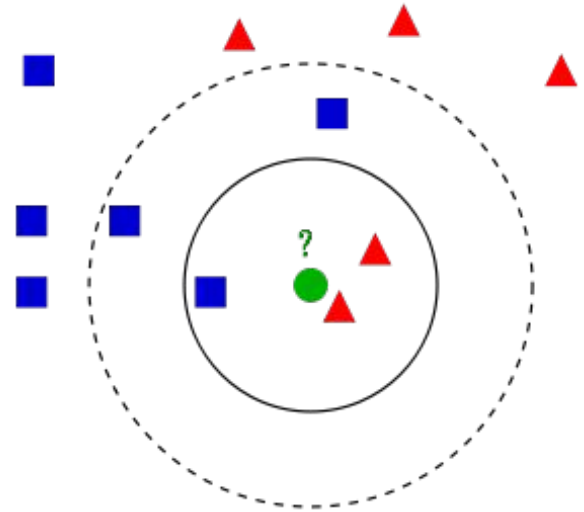
KERNELS

H
HD
HISTORY.COM

Classification Models: KNN

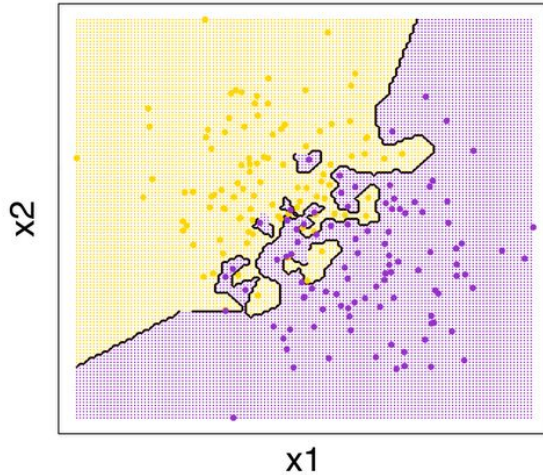
- Clasifica cada nuevo dato en el grupo que corresponda, según tenga **K** vecinos más cerca de un grupo o del otro.
- Calcula la **distancia** del elemento nuevo a cada uno de los existentes y ordena esas *distancias* para seleccionar a qué grupo al que pertenece.
- Selecciona la etiqueta (Y) que más frecuente aparece en las K clases.

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

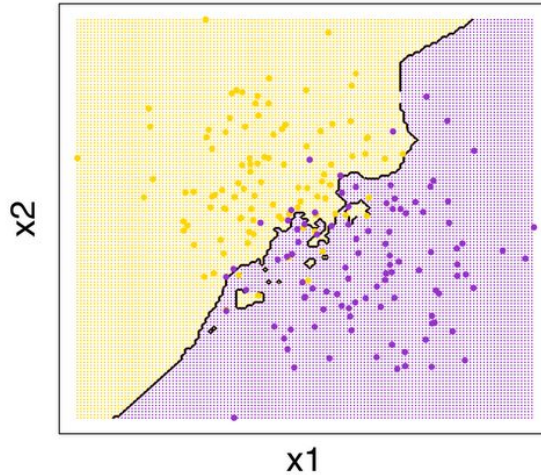


KNN: variar el parametro K

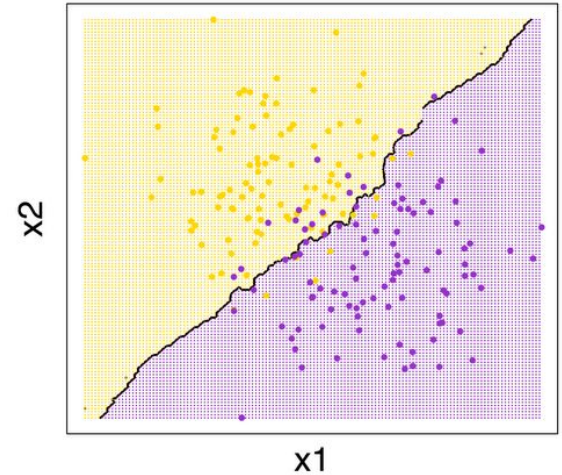
Binary kNN Classification (k=1)



Binary kNN Classification (k=5)



Binary kNN Classification (k=25)



Manos a la obra



JAKE-CLARK.TUMBLR