



# Artificial intelligence

---

*“La ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligente”*, lo cierto es que comúnmente se entiende como la combinación de algoritmos que buscan la **creación de máquinas que presenten las mismas capacidades de un ser humano.**

La inteligencia artificial (IA), es la disciplina que estudia la creación y diseño de entidades capaces de razonar por sí mismas (Cataldi & Lage, 2009)



# Artificial intelligence

## Machine Learning

## Deep learning

- Técnicas que permite a las máquinas imitar el comportamiento humano.

- Subconjunto de técnicas de IA que utilizan métodos estáticos para permitir que las máquinas mejoren la experiencia.

- Subconjunto de ML que hace factible el cálculo de la red neuronal multicapa.

# Machine Learning

---

*"El campo de estudio que brinda a las computadoras la capacidad de aprender sin ser programadas explícitamente"* Arthur Samuel.

*"Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y medida de desempeño P, si su desempeño en las tareas T, medido por P, mejora con la experiencia E"*

Tom Mitchell.

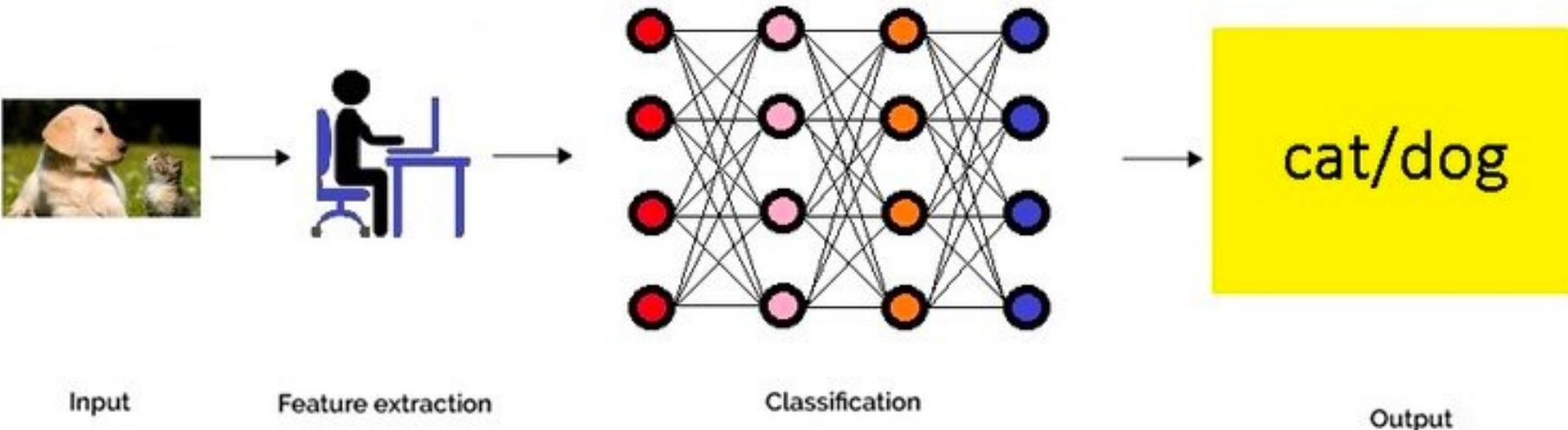
**Etiquetas** Valores que queremos predecir.

**Atributos** Las variables de entrada. Pueden ser 1 como miles.

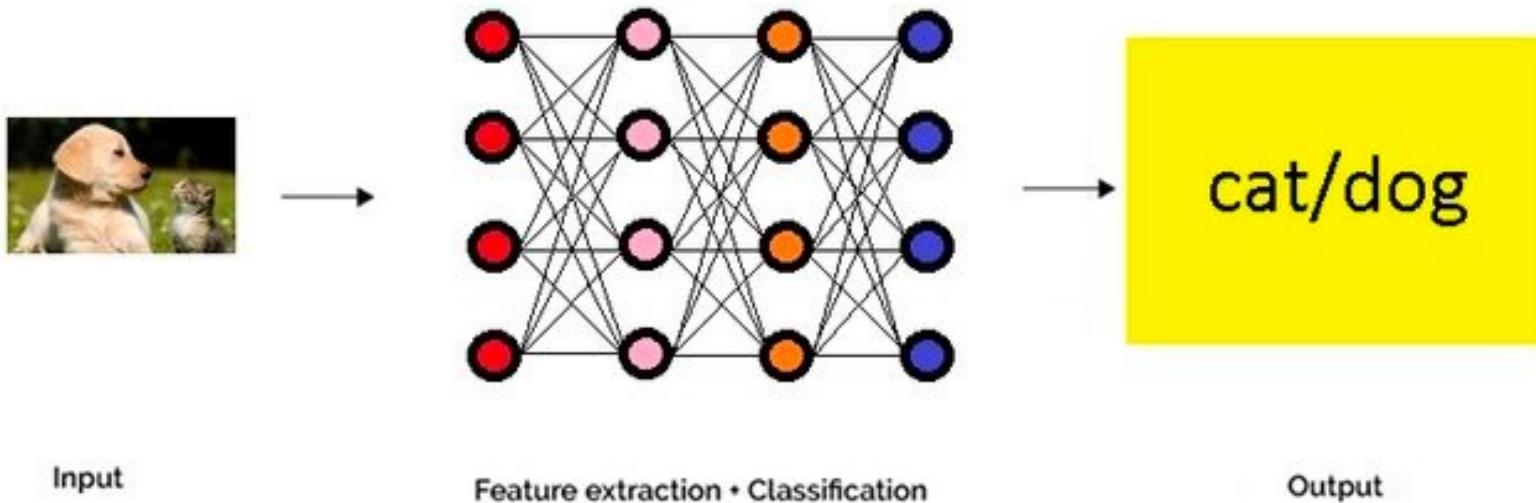
**Algoritmo** Conjunto de instrucciones ordenadas que ofrecen una solución a un problema.

**Modelo** Diseño del algoritmo de Machine Learning, se visualiza como una función que se aplica a los datos nuevos de entrada, donde la salida es el resultado final.

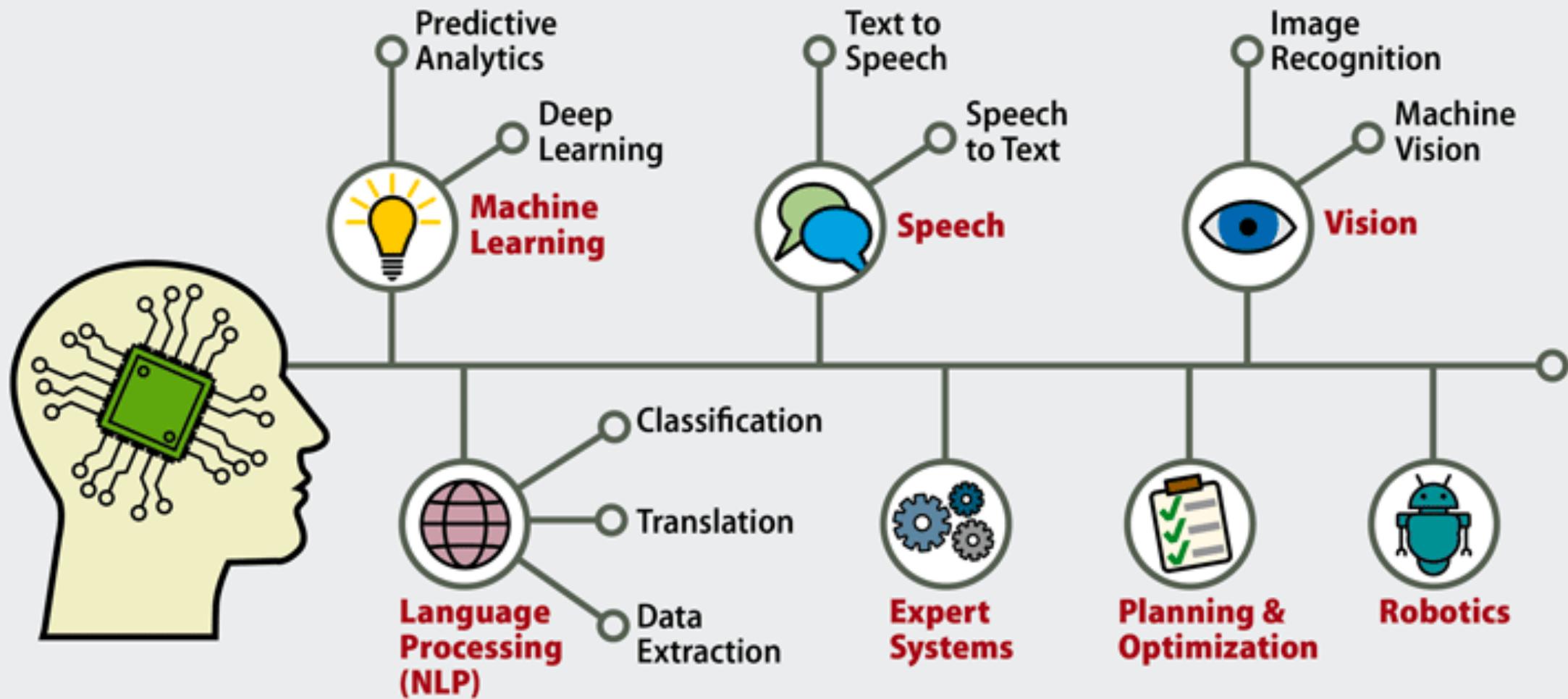
## Machine Learning



## Deep Learning

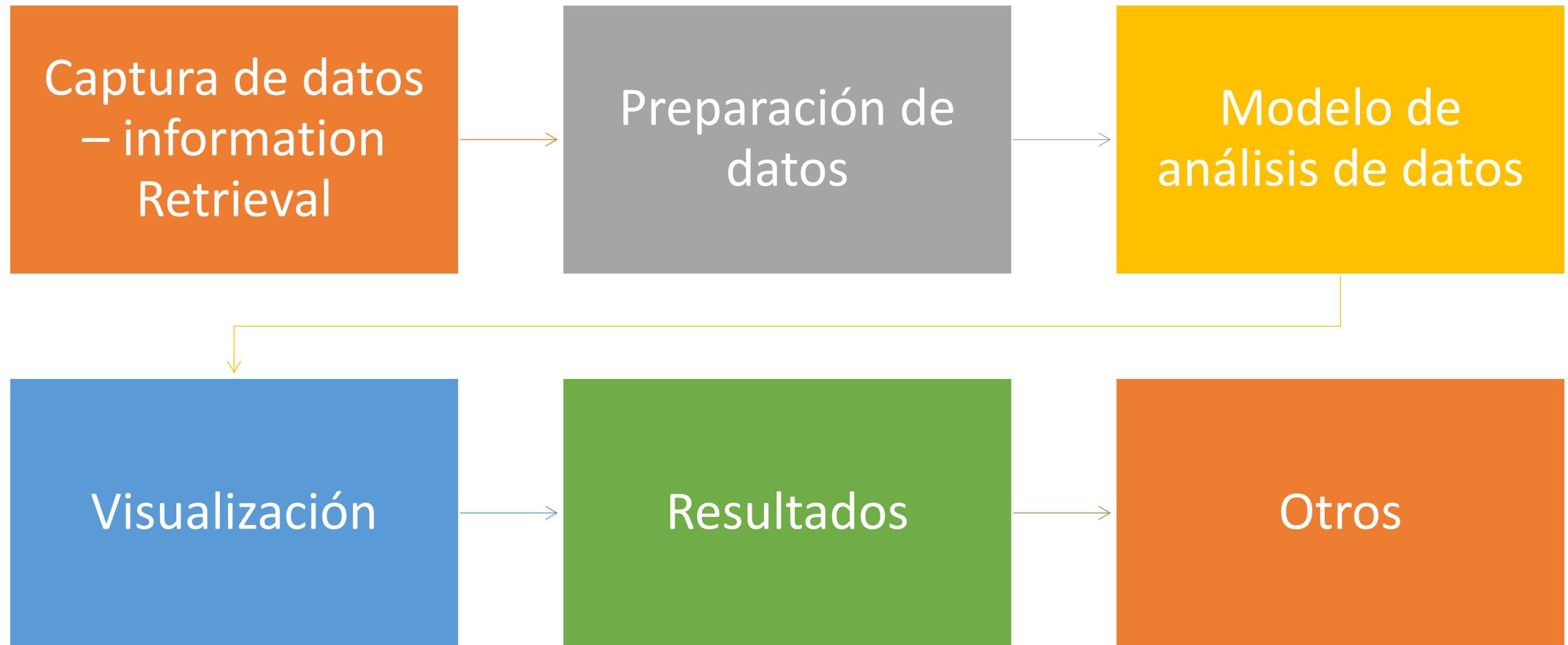


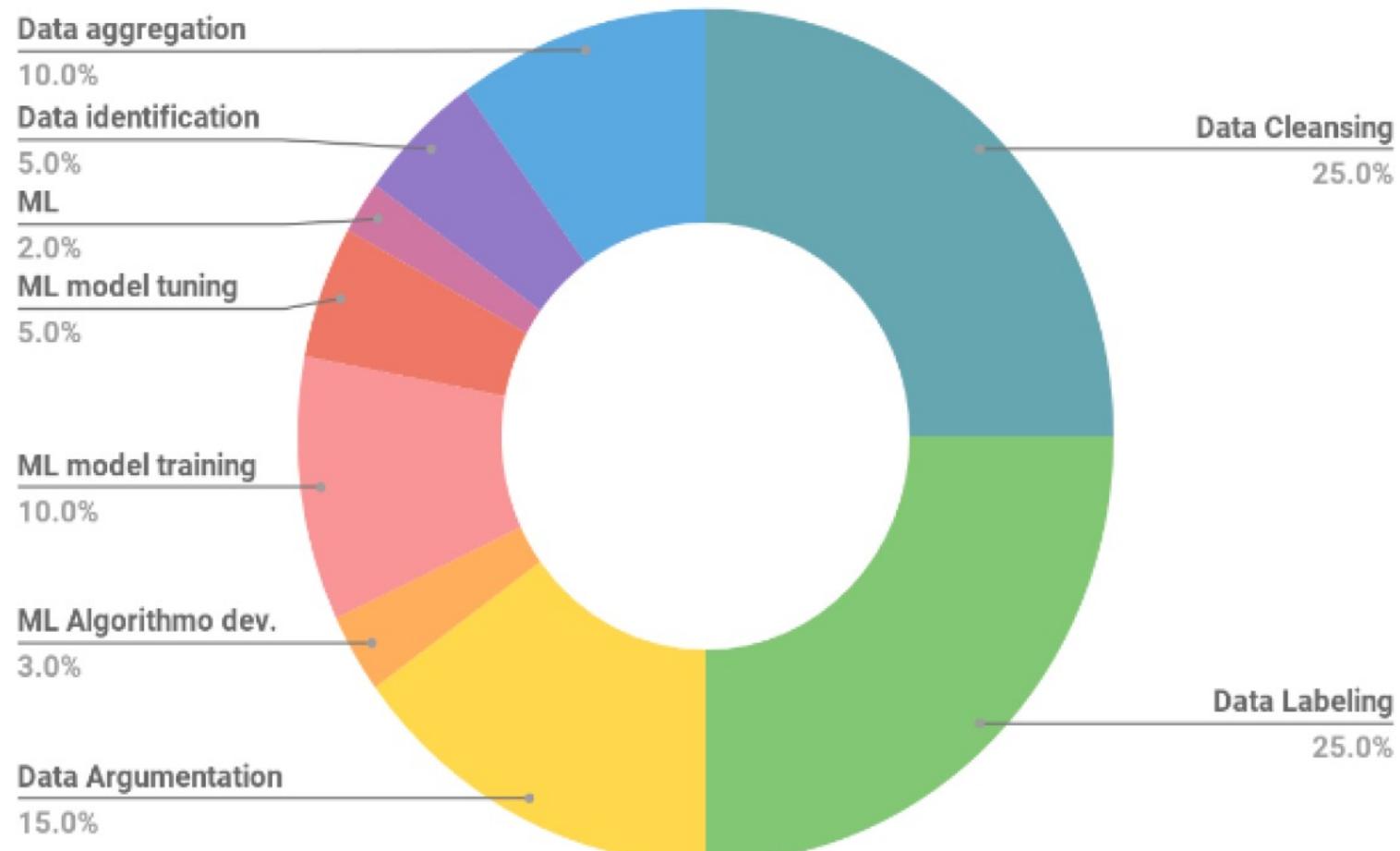
# Artificial Intelligence



# Proceso para el análisis de datos

---





# Preparación de datos

---

Dataset

Recolección de  
datos originales

Identificación de  
atributos y  
etiquetas fuentes

Estrategia de  
muestreo

Dividir datos

Transformación de datos

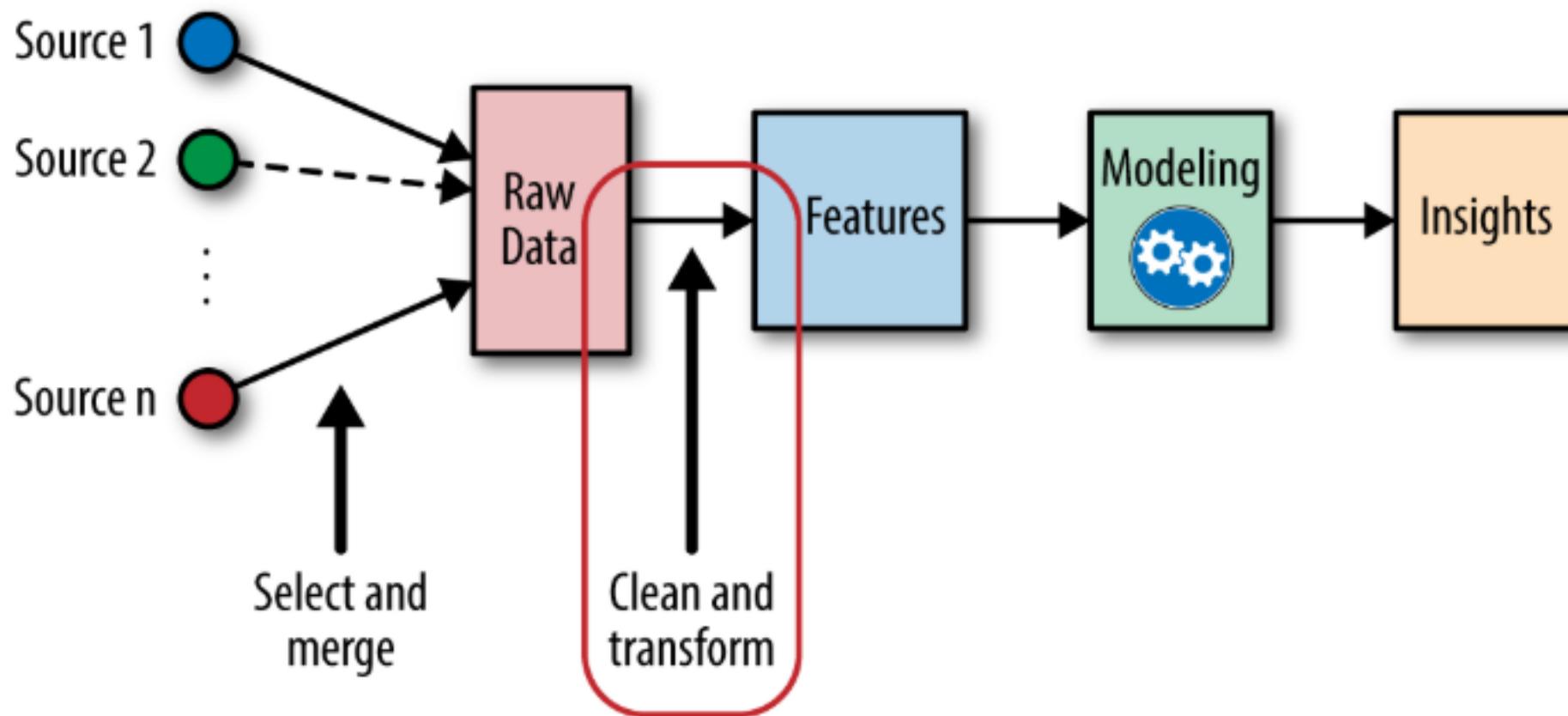
Detección y  
tratamiento de  
outlies

Escalar  
características

Datos faltantes

Feature  
Engineering

# Feature Engineering



Es el proceso de determinar qué características podrían ser útiles para entrenar un modelo y crear esas características mediante la transformación de datos.

## Transformar datos numéricos

- Normalizar
- Bucketing
- Logaritmo -  $\log(x)$
- Reciprocal -  $1 / x$
- Raíz cuadrada -  $\sqrt{x}$
- Exponencial -  $\exp(x)$
- Yeo-Johnson
- Box-Cox

	Date_Time_Combined	Status
0	2018-02-14 20:40	Delayed
1	2018-02-15 10:30	On Time
2	2018-02-14 07:40	On Time
3	2018-02-15 18:10	Delayed
4	2018-02-14 10:20	On Time

## Transformar datos categóricos

- Vocabulario.
- Hashing.
- Híbrido de Hashing y Vocabulario

	Hour_Of_Day	Status
0	20	Delayed
1	10	On Time
2	7	On Time
3	18	Delayed
4	10	On Time

# Categorical Features

Un tipo común de datos no numéricos son los datos categóricos. Por ejemplo, imagine que está explorando algunos datos sobre los precios de la vivienda y, junto con características numéricas como "precio" y "habitaciones", también tiene información sobre el "vecindario".

```
In[1]: data = [
    {'price': 850000, 'rooms': 4, 'neighborhood': 'Queen Anne'},
    {'price': 700000, 'rooms': 3, 'neighborhood': 'Fremont'},
    {'price': 650000, 'rooms': 3, 'neighborhood': 'Wallingford'},
    {'price': 600000, 'rooms': 2, 'neighborhood': 'Fremont'}
]
```

You might be tempted to encode this data with a straightforward numerical mapping:

```
In[2]: {'Queen Anne': 1, 'Fremont': 2, 'Wallingford': 3};
```

# Categorical Features

La codificación one-hot, que efectivamente crea columnas adicionales que indican la presencia o ausencia de una categoría con un valor de 1 o 0, respectivamente.

```
In[3]: from sklearn.feature_extraction import DictVectorizer  
vec = DictVectorizer(sparse=False, dtype=int)  
vec.fit_transform(data)  
  
Out[3]: array([[ 0, 1, 0, 850000, 4],  
 [ 1, 0, 0, 700000, 3],  
 [ 0, 0, 1, 650000, 3],  
 [ 1, 0, 0, 600000, 2]], dtype=int64)
```

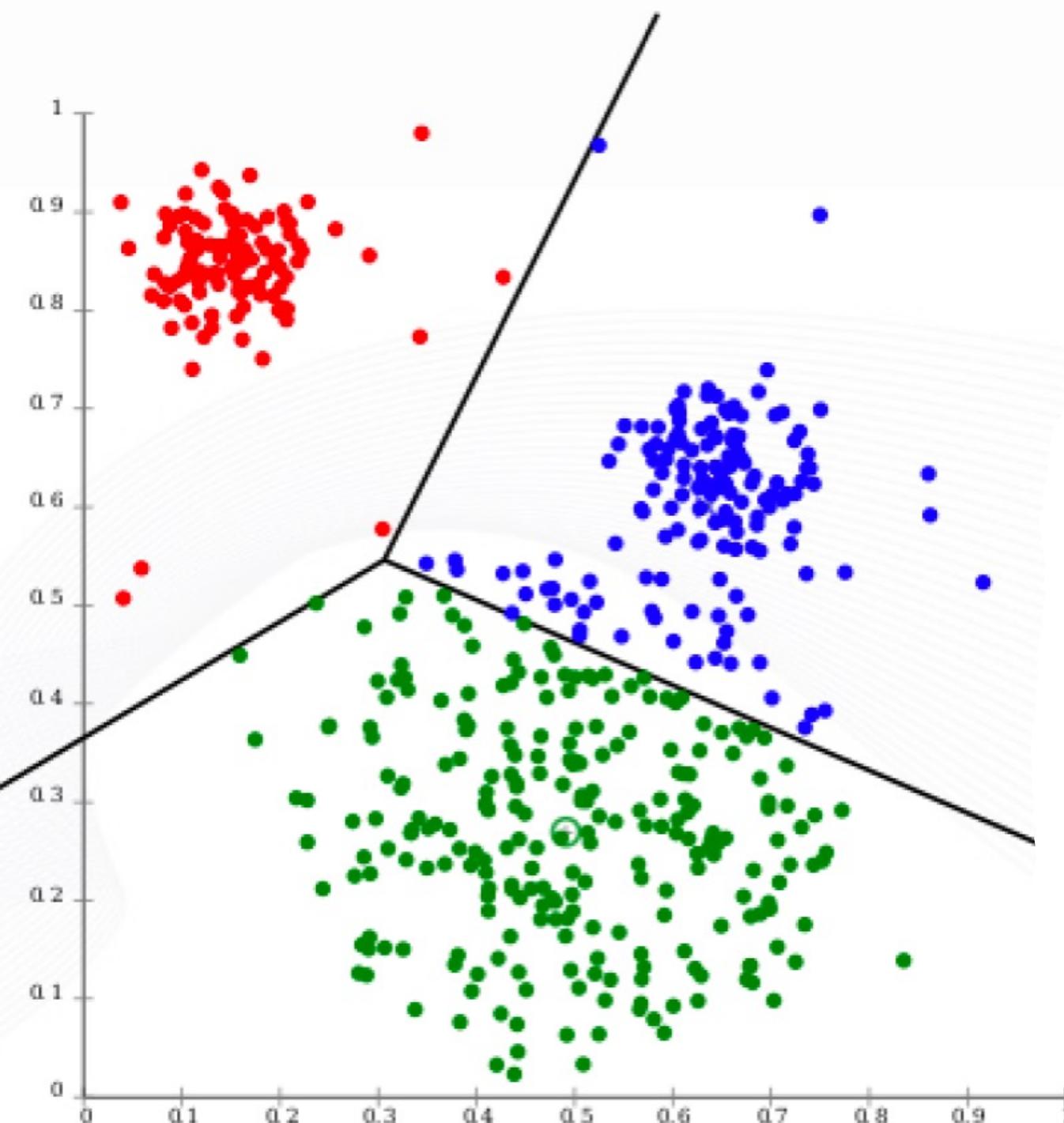
```
In[4]: vec.get_feature_names()
```

```
Out[4]: ['neighborhood=Fremont',  
 'neighborhood=Queen Anne',  
 'neighborhood=Wallingford',  
 'price',  
 'rooms']
```

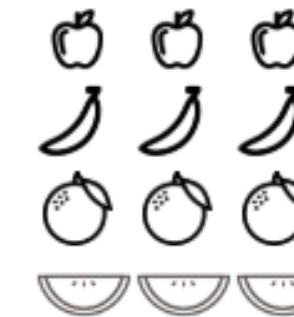
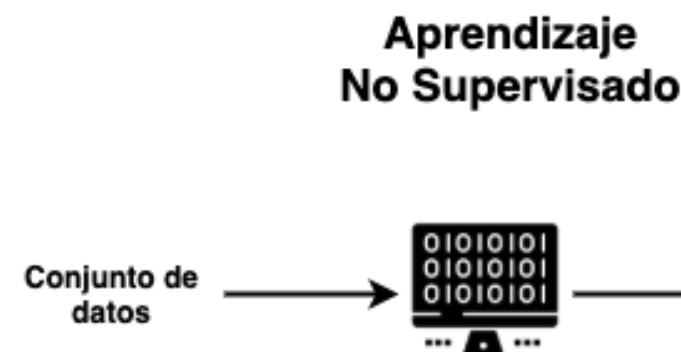
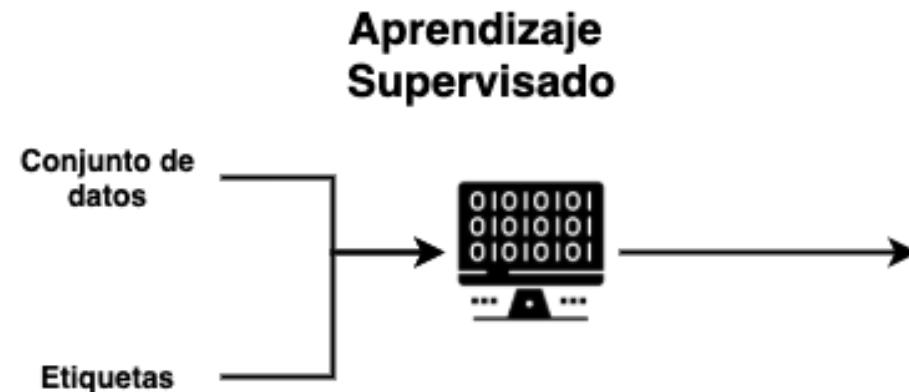
**Aprendizaje Supervisado:** Este tipo de aprendizaje se basa en lo que se conoce como información de entrenamiento. Se entrena al sistema proporcionándole cierta cantidad de datos definiéndolos al detalle con etiquetas.

**Aprendizaje NO Supervisado:** En este tipo de aprendizaje no se usan valores verdaderos o etiquetas. Estos sistemas tienen como finalidad la **comprensión** y **abstracción** de patrones de información de manera directa.

**Aprendizaje por refuerzo:** Determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de "recompensa" o premio acumulado.



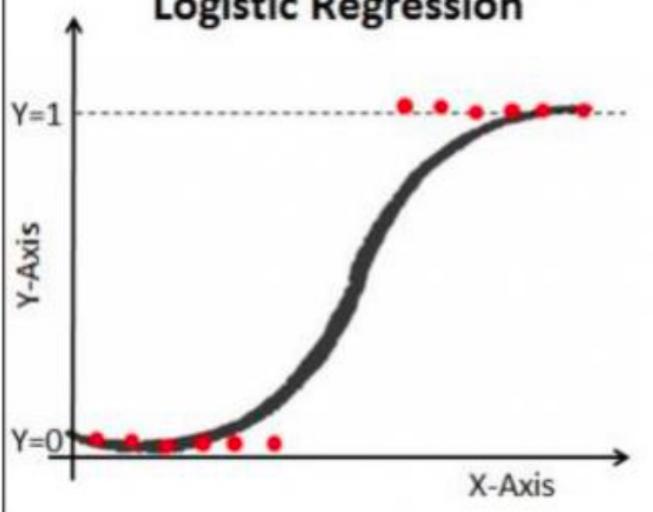
- Un modelo de clasificación predice valores discretos. Los datos se separan en un cierto número de clases.



# Algoritmos de Clasificación

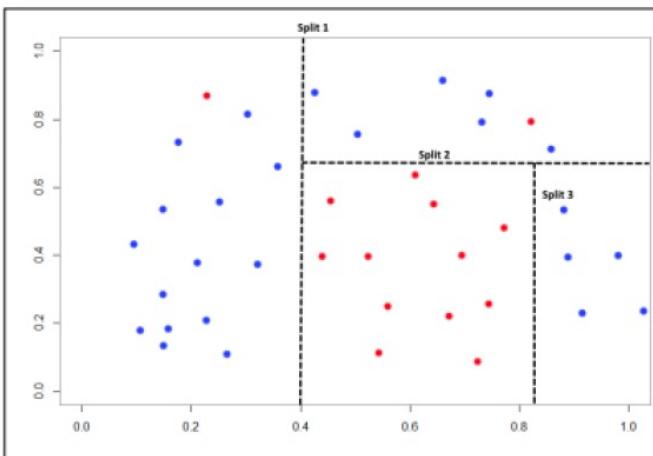
## Logistic Regression

A pesar de su nombre no es un algoritmo para problemas de regresión sino que es un método para problemas de clasificación, en los que se obtienen un valor binario entre 0 y 1.



## Decision Trees

Cada nodo interno denota una prueba en un atributo, cada rama representa el resultado de una prueba, y cada hoja nodo tiene una etiqueta. El nodo superior en un árbol es el nodo raíz (algoritmo también aplicado en regresión).

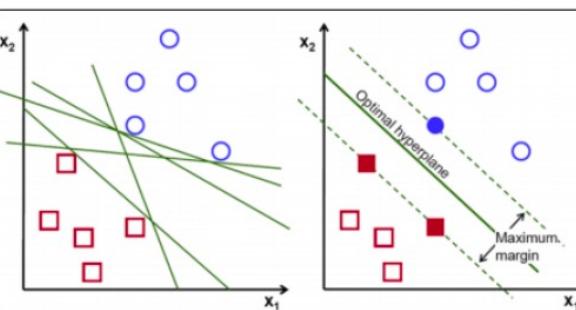


## Random Forest

Es un método que combina una cantidad grande de árboles de decisión independientes probados sobre conjuntos de datos aleatorios con igual distribución(algoritmo también aplicado en regresión).

## Support vector Machines (SVMs)

El objetivo es encontrar un hiperplano en un espacio N-dimensional que clasifica claramente los puntos de datos (algoritmo también aplicado en regresión).



# Algoritmos de Regresión

## Linear Regression

Da como resultado un valor continuo a partir de una combinación lineal de atributos de entrada.

## Regression Trees

Árboles de decisión, donde la variable de destino puede tomar valores continuos.

## Non-Linear Regression

Es una combinación no lineal de los parámetros del modelo y depende de una o más variables independientes.

## Bayesian Linear Regression

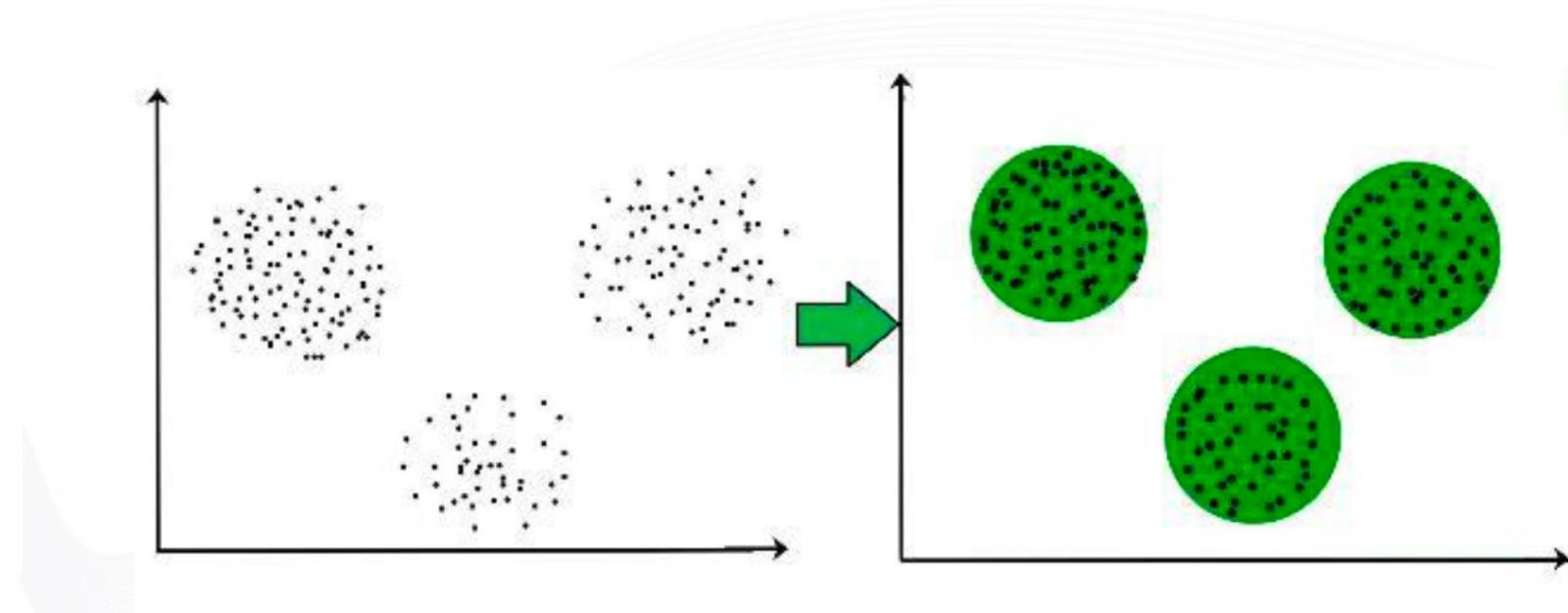
Es un enfoque de regresión lineal en la que se lleva a cabo el análisis estadístico en el contexto de la inferencia bayesiana.

## Polynomial Regression

Es un caso especial de regresión lineal donde ajustamos una ecuación polinómica en los datos con una relación curvilínea entre la variable objetivo y las variables independientes.

# Clustering

Se utilizan para agrupar datos existentes de los que desconocemos sus características en común o queremos descubrirlas. Estos métodos intentan crear puntos centrales y jerarquías para diferenciar grupos y descubrir características comunes por cercanía.



# Asociación de reglas

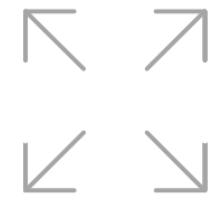
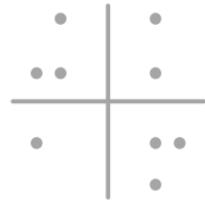
# Detección de anomalías

Se utilizan para encontrar asociaciones y relaciones entre grandes conjuntos de elementos de datos. Estas reglas muestran con qué frecuencia se produce un conjunto de elementos en una transacción.

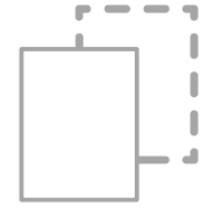
La detección de anomalías se refiere a la identificación de elementos o eventos que no se ajustan a un patrón esperado u otros elementos en un conjunto de datos que generalmente no son detectables por un humano.



**Bayesianos    Clustering**



**Arboles de decisión**



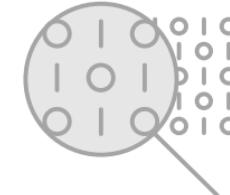
**Reducción de  
dimensionalidad**



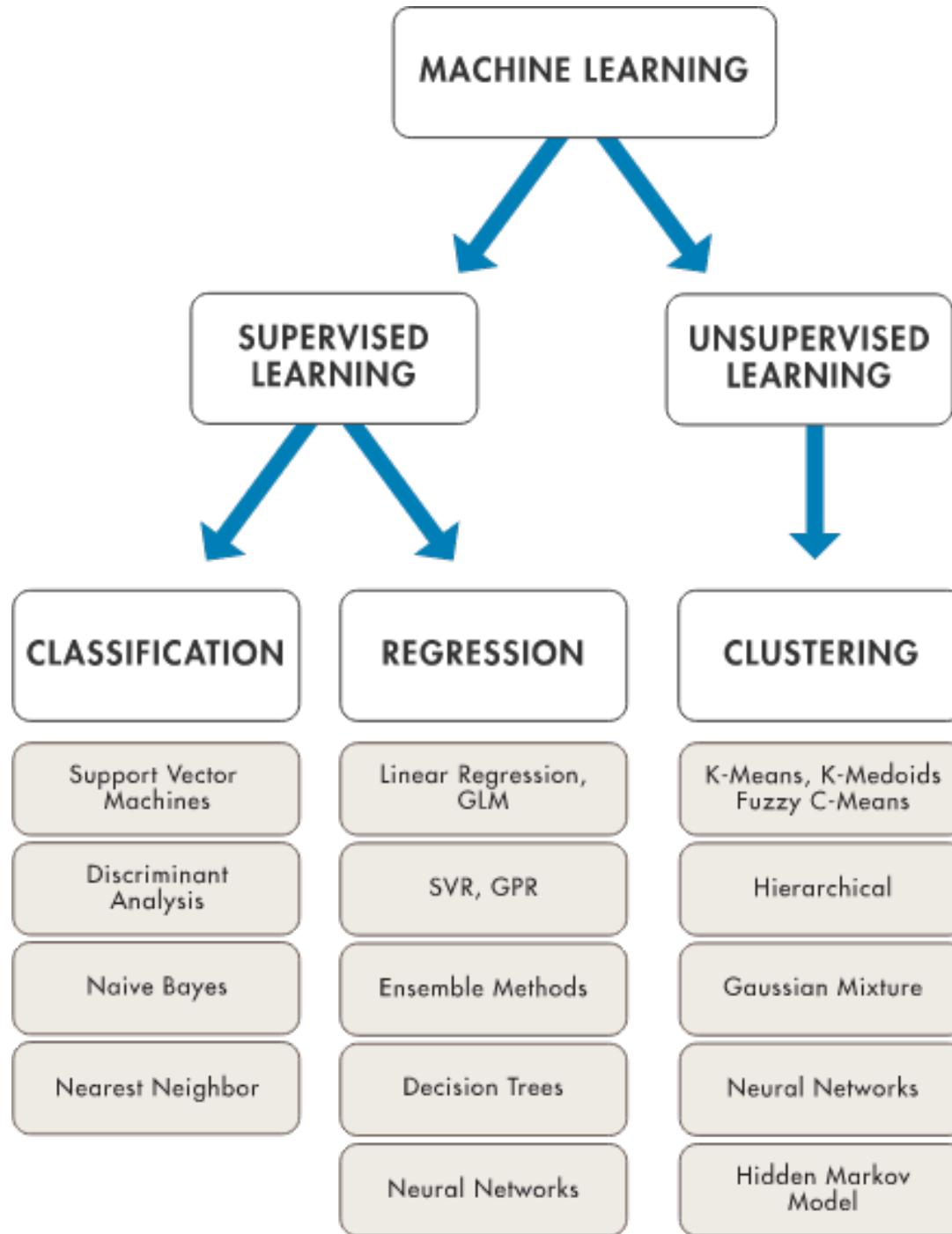
**Algoritmos basados  
en instancias**



**Regresión lineal**



**Regularización**



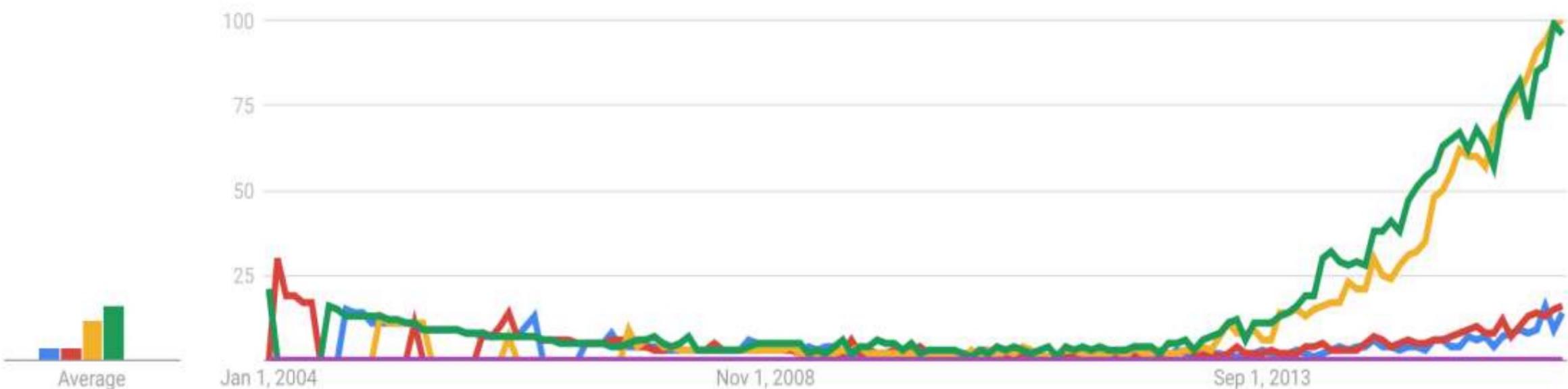
SPSS

SAS

Python

R

Scala



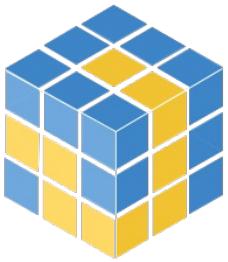
Trends in Google Searches (September 2<sup>nd</sup> 2016)



El Proyecto Jupyter es una organización sin ánimo de lucro creada para "desarrollar software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación". Creado a partir de IPython en 2014 por Fernando Pérez, el proyecto Jupyter soporta entornos de ejecución en varias docenas de lenguajes de programación. El nombre del proyecto Jupyter es una referencia a los tres lenguajes de programación principales soportados por Jupyter, que son Julia, Python y R, y también un homenaje a los cuadernos de Galileo que registran el descubrimiento de los satélites de Júpiter. El proyecto Jupyter ha desarrollado y respaldado los productos de computación interactiva Jupyter Notebook, JupyterHub y JupyterLab, la versión de próxima generación de Jupyter Notebook.

**Apoyar la ciencia de datos interactiva y la informática científica en todos los lenguajes de programación**

# Aplicación de ciencia de datos

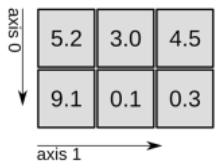


## NumPy

1D array

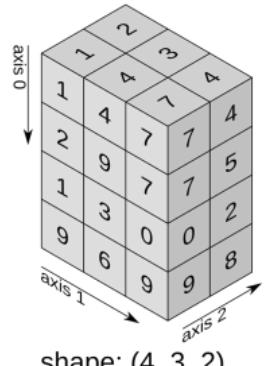


2D array



shape: (4,)

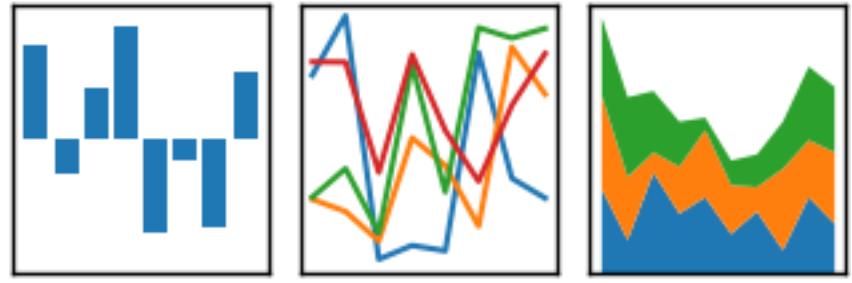
3D array



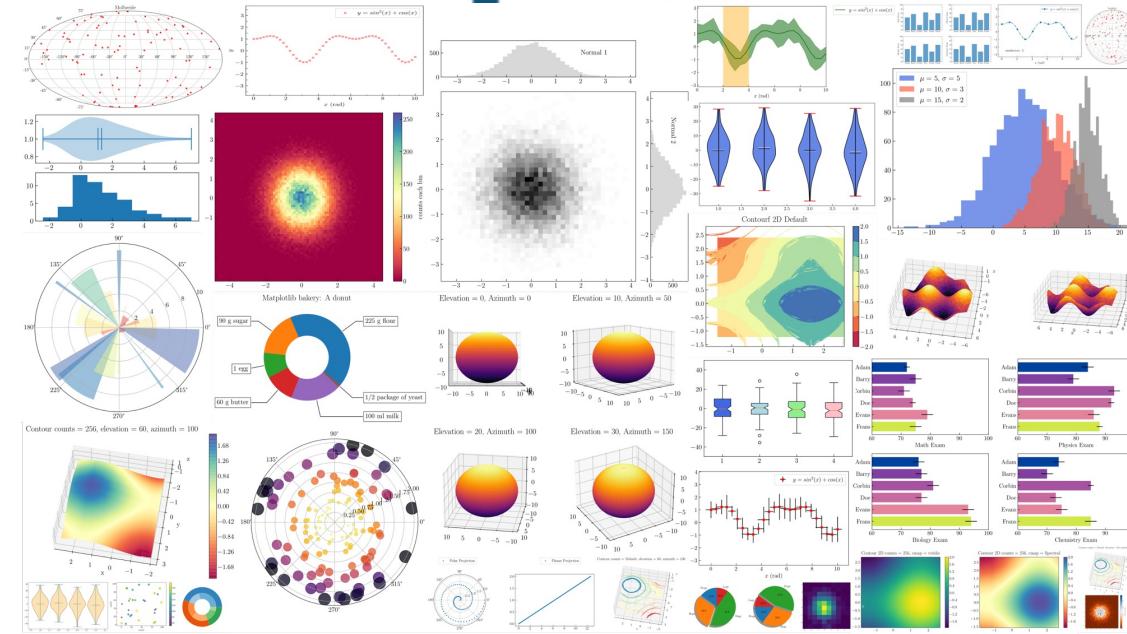
shape: (2, 3)

## pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



## matplotlib



# Evaluaciones de la clasificación

Precision	Precisión
Recall	Exhaustividad
F1-score	Valor-F
Accuracy	Exactitud
Confusion Matrix	Matriz de Confusión
True Positive	Positivos Verdaderos
True Negative	Negativos Verdaderos
False Positive	Positivos Falsos
False Negative	Negativos Falsos



0: no está interesado  
1: sí está interesado

		predicción	
		0	1
realidad	0	70	10
	1	15	5

		predicción	
		0	1
realidad	0	TN	FP
	1	FN	TP

## Precision (Precisión)

Con la métrica de precisión podemos medir la **calidad** del modelo de machine learning en tareas de clasificación. En el ejemplo, se refiere a que la precisión es la respuesta a la pregunta ¿qué porcentaje de los clientes que contactemos estarán interesados?

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

		predicción	
		0	1
realidad	0	TN	FP
	1	FN	TP

$$precision = \frac{TP}{TP + FP} = \frac{5}{5 + 10} = 0.33$$

## Recall (Exhaustividad)

La métrica de exhaustividad nos va a informar sobre la **cantidad** que el modelo de machine learning es capaz de identificar. En el ejemplo, se refiere a que la exhaustividad (recall) es la respuesta a la pregunta ¿qué porcentaje de los clientes están interesados somos capaces de identificar?

Para calcular la exhaustividad (recall) usaremos la siguiente fórmula:

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

		predicción	
		0	1
realidad	0	TN	FP
	1	FN	TP

$$\text{recall} = \frac{TP}{TP + FN} = \frac{5}{5 + 15} = 0.25$$

Es decir, el modelo sólo es capaz de identificar un **25%** de los clientes que estarían interesados en adquirir el producto. Esto significa que el modelo del ejemplo sólo es capaz de identificar 1 de cada 4 de los clientes que sí aceptarían la oferta.

## F1 (Valor-F)

El valor F1 se utiliza para combinar las medidas de precision y recall en un sólo valor. Esto es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

F1 se calcula haciendo la media armónica entre la precisión y la exhaustividad:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 2 \cdot \frac{0.33 \cdot 0.25}{0.33 + 0.25} = 0.28$$

## Accuracy (Exactitud)

La exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado. Esta es una de las métricas más usadas y favoritas ... que te recomiendo evitar! El problema con la exactitud es que nos puede llevar al engaño, es decir, puede hacer que un modelo malo (como el del ejemplo) parezca que es mucho mejor de lo que es.

El accuracy (exactitud) se calcula con la siguiente fórmula:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5 + 70}{5 + 70 + 10 + 15} = 0.75$$

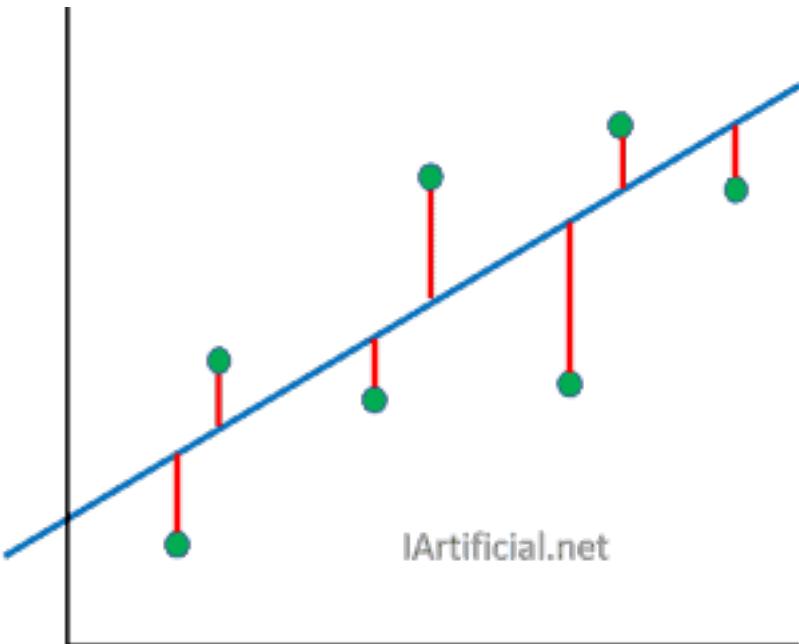
Es decir, el modelo acierta el 75% de las veces. La exactitud es una métrica muy engañosa. De hecho, si tuviésemos un modelo que siempre predijera que el cliente nunca va a estar interesado, su accuracy sería del 80%.

# Error Cuadrático Medio para Regresión

El **Error Cuadrático Medio** es el criterio de evaluación más usado para problemas de regresión. Se usa sobre todo cuando usamos aprendizaje automático supervisado. Para cada dato histórico podremos indicar el resultado correcto. Vamos a ver como se calcula.

Vamos a calcular el error cuadrático medio con un ejemplo. En la figura vemos que estamos usando una regresión **lineal (en azul)** para estimar los datos que tenemos (**los puntos verdes**). El modelo lineal tiene un error (**en rojo**) que podemos definir con la siguiente fórmula:

$$\text{error cuadrático} = (\text{real} - \text{estimado})^2$$



El valor estimado es el valor que nos da el modelo. En este caso, la línea azul.

Calculamos el error al cuadrado, en lugar del error simple, para que el error siempre sea positivo. De esta forma sabemos que el error perfecto es 0. Si no elevásemos el error al cuadrado, unas veces el error sería positivo y otras negativo. Otra posibilidad sería usar el valor absoluto, en lugar de elevarlo al cuadrado. Sin embargo, si usamos el valor absoluto, obtendremos una función no-derivable.

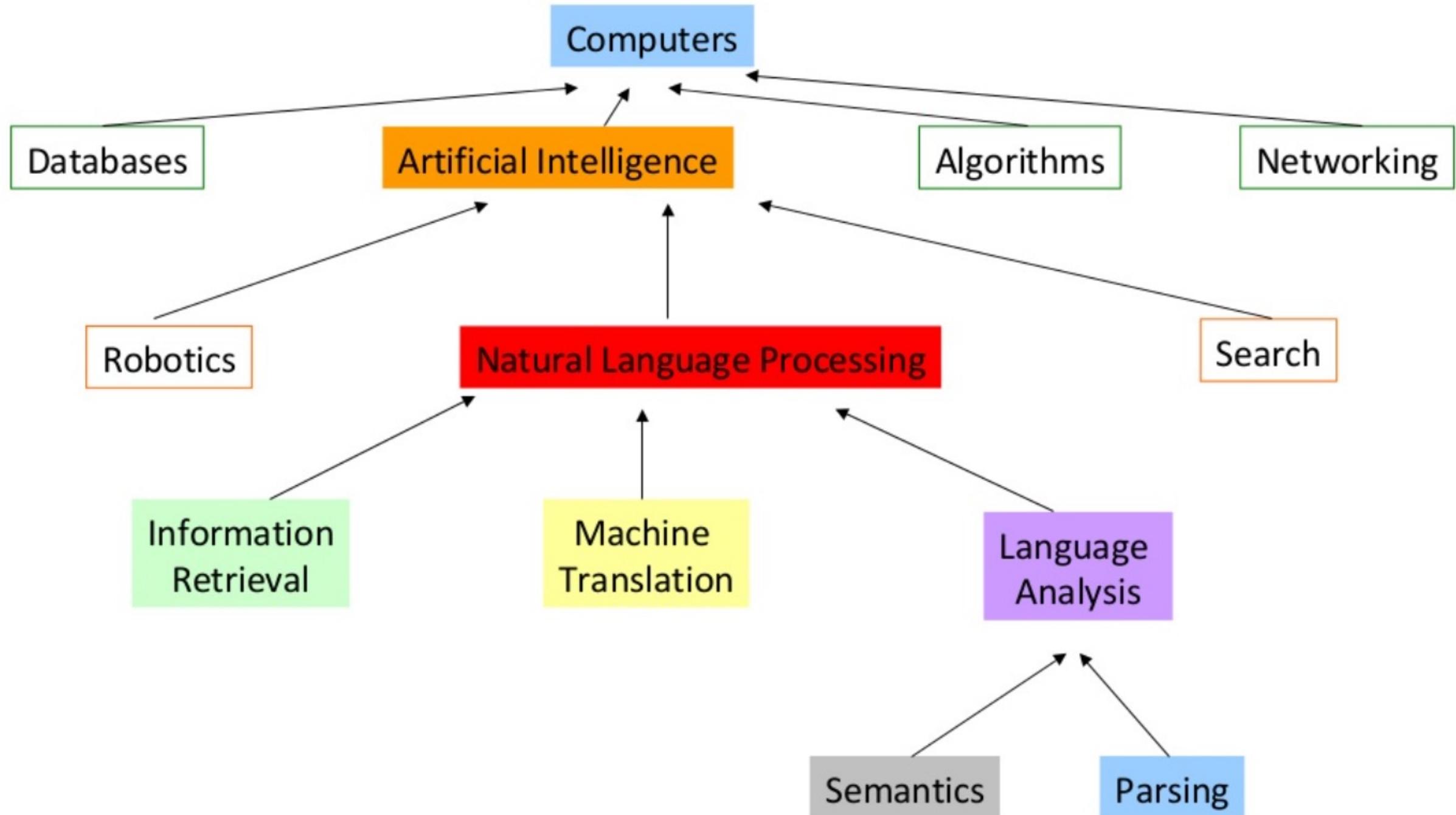
Ahora que sabemos cómo calcular el error en cada punto, podemos calcular cual es el error medio. Para ello, sumamos todos los errores y los dividimos entre el número total de puntos. Si llamamos **M** al número total de puntos nos queda la fórmula del Error Cuadrático Medio (MSE, por sus siglas en inglés, Mean Squared Error):

$$MSE = \frac{1}{M} \sum_{i=1}^M (real_i - estimado_i)^2$$

# Natural Language Processing



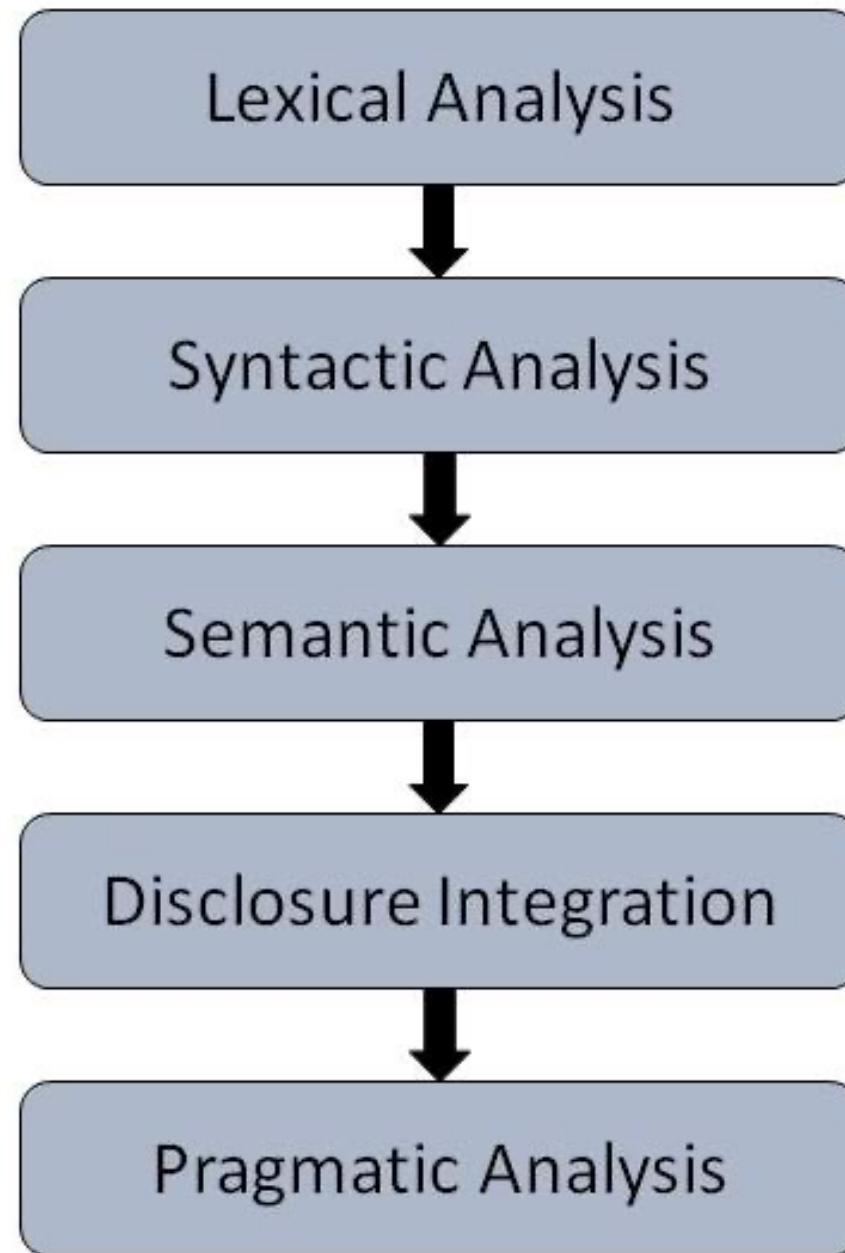
Campo de las ciencias de la computación, de la **inteligencia artificial** y **de la lingüística** que estudia las interacciones entre las computadoras y el lenguaje humano. Se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio del lenguaje natural. No trata de la comunicación por medio de lenguas naturales de una forma abstracta, sino de **diseñar mecanismos para comunicarse que sean eficaces computacionalmente** —que se puedan realizar por medio de programas que ejecuten o simulen la comunicación.



# NLP Terminology

- **Phonology**: Estudio de la organización del sonido de forma sistemática.
- **Morphology** – Estudio de la construcción de palabras a partir de unidades significativas primitivas.
- **Morpheme** – Unidad primitiva de significado en un idioma..
- **Syntax** – Organización de palabras para formar una oración. También implica determinar el papel estructural de las palabras en la oración y en las frases..
- **Semantics** – Se ocupa del significado de las palabras y de cómo combinarlas en frases y oraciones significativas..
- **Pragmatics** – Uso y la comprensión de oraciones en diferentes situaciones y cómo se ve afectada la interpretación de la oración..
- **Discourse** – Se trata de cómo la oración inmediatamente anterior puede afectar la interpretación de la siguiente.
- **World Knowledge** – incluye el conocimiento general sobre el mundo.

# Pasos del NLP



**Step 1: Lexical Analysis:** Implica identificar y analizar la estructura de las palabras. Léxico de un idioma significa la colección de palabras y frases en un idioma. El análisis léxico consiste en dividir todo el texto en párrafos, oraciones y palabras. *Ejemplo: Carried = Carry + ed*

**Step 2: Syntactic Analysis:** Implica el análisis de palabras en la oración para la gramática y la organización de las palabras de una manera que muestra la relación entre las palabras. Ejemplo: El analizador sintáctico inglés rechaza oraciones como "*The school goes to boy*" - "boy the go the to store"

**Step 3: Semantic Analysis:** Extrae el significado exacto o el significado del diccionario del texto. Se comprueba la significación del texto. Se realiza mapeando estructuras sintácticas y objetos en el dominio de tareas. El analizador semántico ignora frases como "*hot ice-cream*"

**Step 4: Discourse Integration:** El significado de cualquier oración depende del significado de la oración que la precede. Además, también aporta el significado de la oración inmediatamente posterior. “**john wanted it**” the word ‘it’ depends upon john.

**Step 5: Pragmatic Analysis:** Durante esto, lo que se dijo se reinterpreta de acuerdo con lo que realmente significaba. Implica derivar aquellos aspectos del lenguaje que requieren conocimiento del mundo real. “**DO you know what time it is?**” debe interpretarse como una solicitud.

## Tf-idf (*Term frequency – Inverse document frequency*)

**Medida numérica** que expresa cuán **relevante** es una palabra para un documento en una colección. Esta medida se utiliza a menudo como un factor de ponderación en la **recuperación de información** y la minería de texto. El valor **tf-idf** aumenta proporcionalmente al número de veces que una palabra aparece en el documento, **pero es compensada por la frecuencia de la palabra en la colección de documentos**, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras.

## ¿Qué es TF (frecuencia de términos)?

La frecuencia de términos funciona observando la frecuencia de un término en particular que le preocupa en relación con el documento. Existen múltiples medidas, o formas, de definir la frecuencia:

- Número de veces que aparece la palabra en un documento (recuento sin procesar).
- Frecuencia de términos ajustada por la longitud del documento (recuento sin procesar de ocurrencias dividido por el número de palabras en el documento).
- Frecuencia escalada logarítmicamente (por ejemplo,  $\log(1 + \text{conteo sin procesar})$ ).
- Frecuencia booleana (por ejemplo, 1 si aparece el término, o 0 si el término no aparece, en el documento).

## ¿Qué es IDF (frecuencia de documento inversa)?

La frecuencia inversa del documento analiza qué tan común (o poco común) es una palabra entre el corpus. IDF se calcula de la siguiente manera, donde **t** es el término (palabra) que buscamos para medir la frecuencia y **N** es el número de documentos (**d**) en el corpus (**D**). El denominador es simplemente el número de documentos en los que el término, **t**, aparece en.

$$idf(t, D) = \log \left( \frac{N}{\text{count}(d \in D : t \in d)} \right)$$

Image Source: <https://monkeylearn.com/blog/what-is-tf-idf/>

**Nota:** Es posible que un término no aparezca en absoluto en el corpus, lo que puede resultar en un error de división por cero. Una forma de manejar esto es tomar el conteo existente y agregar 1. Haciendo así el denominador ( $1 + \text{conteo}$ ). Un ejemplo de cómo la popular biblioteca **scikit-learn** maneja esto se puede ver a continuación.

### Scikit-Learn

- $\text{IDF}(t) = \log \frac{1+n}{1+\text{df}(t)} + 1$

### Standard notation

- $\text{IDF}(t) = \log \frac{n}{\text{df}(t)}$

La razón por la que necesitamos IDF es para ayudar a corregir palabras como "of", "as", "the", etc., ya que aparecen con frecuencia en un corpus en inglés. Por lo tanto, al tomar la frecuencia del documento inversa, podemos minimizar la ponderación de los términos frecuentes mientras hacemos que los términos poco frecuentes tengan un mayor impacto.

Finalmente, los **IDF** también se pueden extraer de un corpus de fondo, que corrige el sesgo de muestreo, o del conjunto de datos que se utiliza en el experimento en cuestión.

## TF-IDF

Para resumir, la intuición clave que motiva a **TF-IDF** es que la importancia de un término está inversamente relacionada con su frecuencia en los documentos. **TF nos brinda información sobre la frecuencia con la que aparece un término en un documento** e **IDF nos brinda información sobre la rareza relativa de un término en la colección de documentos**. Al multiplicar estos valores juntos, podemos obtener nuestro valor final de TF-IDF.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Cuanto mayor sea la puntuación de **TF-IDF**, más importante o relevante es el término; a medida que un término se vuelve menos relevante, su puntaje TF-IDF se acercará a 0.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

*A = "The car is driven on the road"; B = "The truck is driven on the highway"* Image from freeCodeCamp - How to process textual data using TF-IDF in Python (<https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/>)