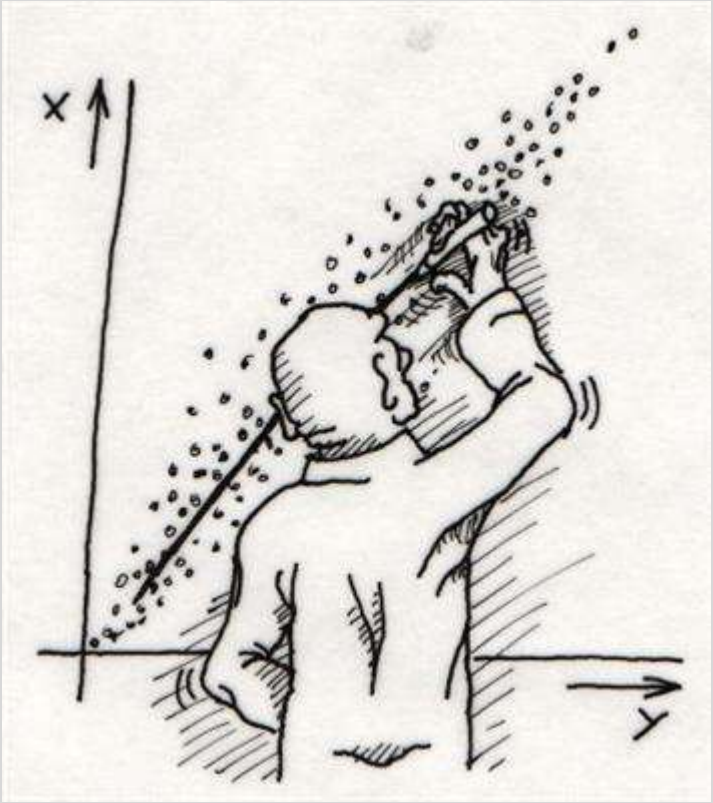


DigitalHouse > Coding School

DATA SCIENCE

MÓDULO 3

Intro a Machine Learning

OBJETIVOS DE ESTA CLASE

Este material no intenta ser una introducción exhaustiva al campo de machine learning, ni intenta ser un manual completo para el uso del paquete Scikit-Learn. Los objetivos de las próximas clases son:

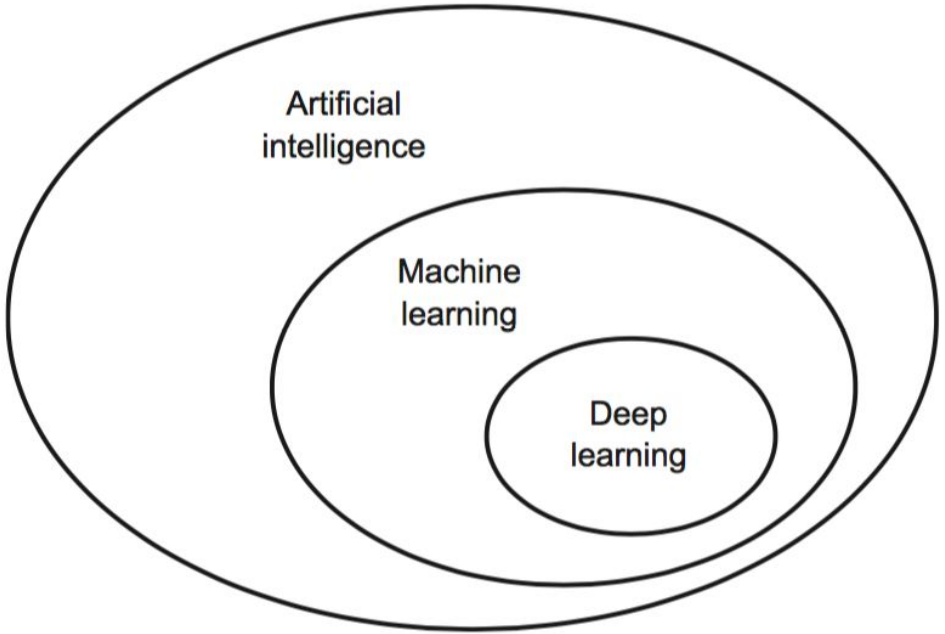
1. **Introducir el vocabulario y conceptos fundamentales de machine learning.**
2. **Introducir la API de Scikit-Learn y mostrar algunos ejemplos de su uso.**
3. **Estudiar algunos de los algoritmos más importantes, y desarrollar una intuición sobre cómo trabajan y cuándo y dónde son aplicables**

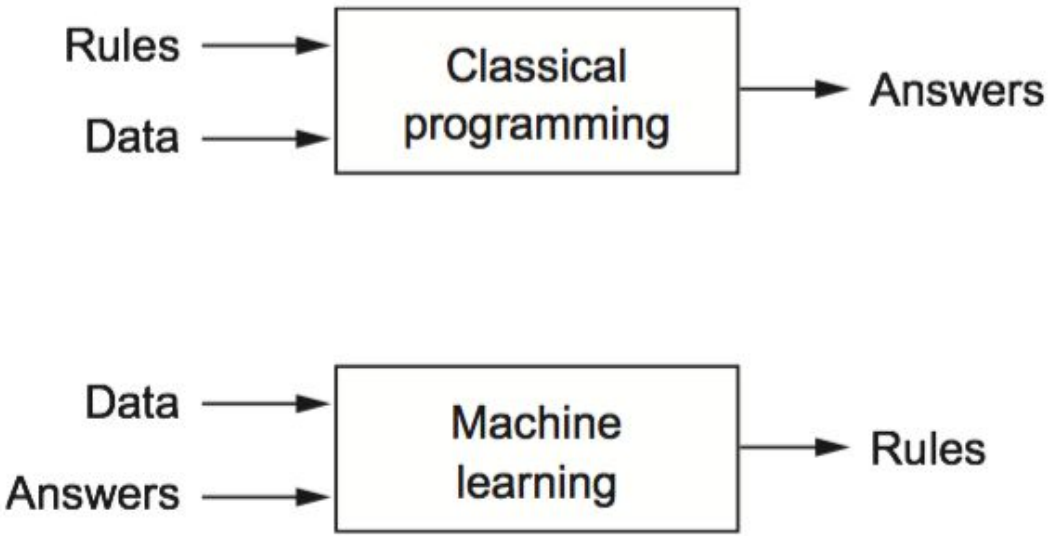
INTRODUCCIÓN

Dos definiciones de ML:

Arthur Samuel (1959): "el campo de estudio que da a las computadoras la capacidad de **aprender sin estar programado explícitamente**."

Tom Mitchell (1998): "Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna tarea T y la medida de rendimiento P, si su desempeño en la tarea T, medido por P, mejora con la experiencia E."

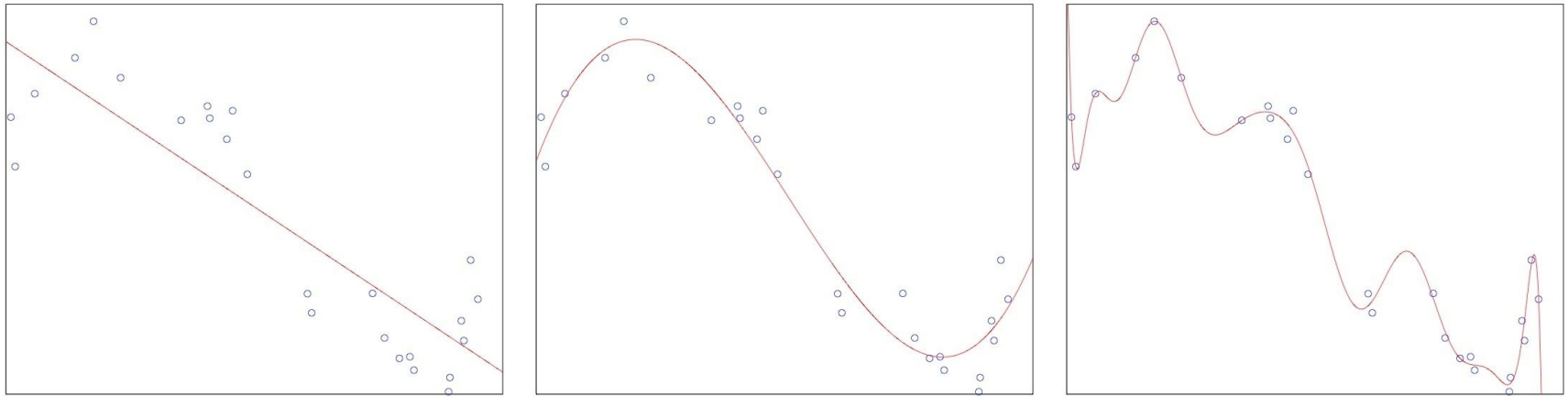




El aprendizaje entra en juego cuando le damos a estos modelos parámetros que pueden modificarse y que se adaptan a los datos observados de manera automática; de esta forma, puede considerarse que **el algoritmo aprende de los datos.**

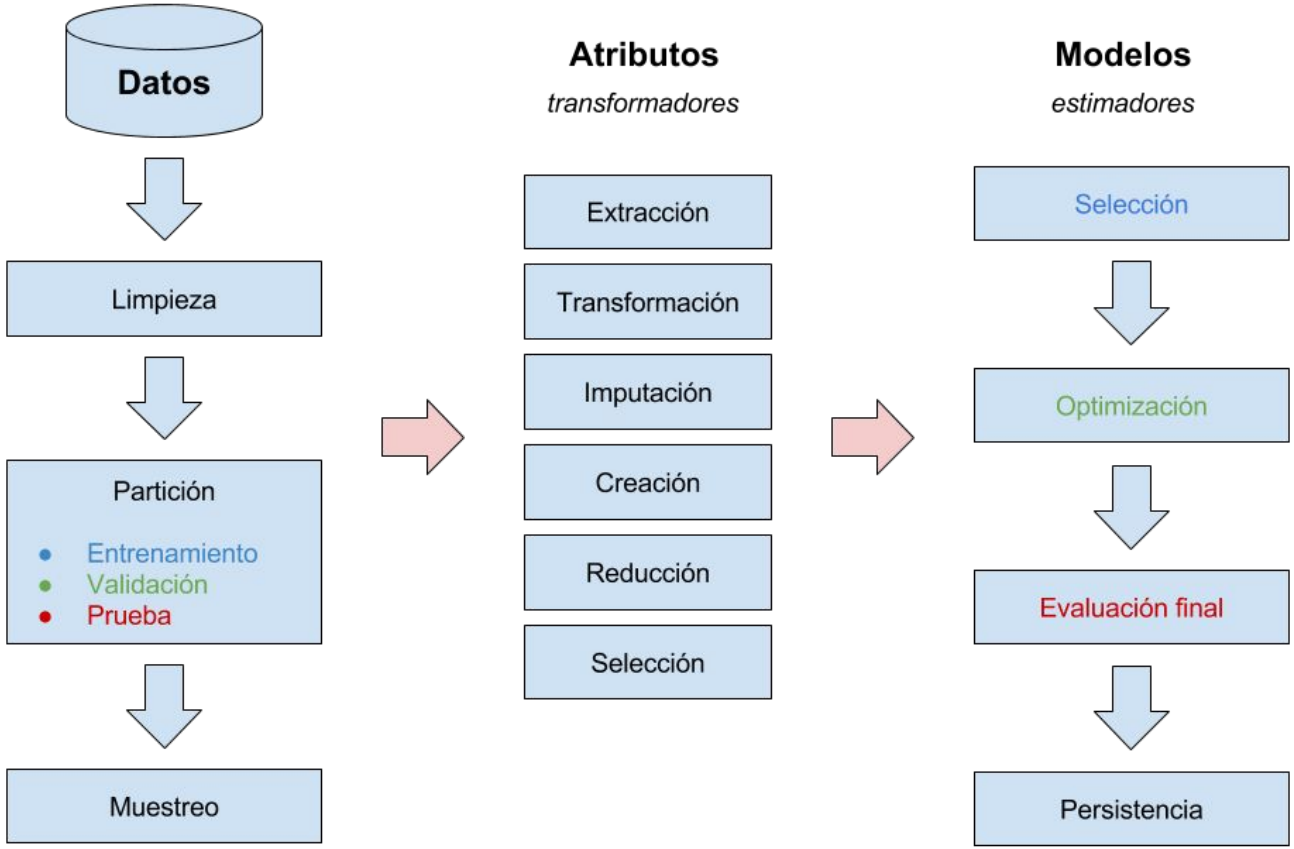
Una vez que estos modelos han sido ajustados a datos observados previamente, pueden ser usados para **predecir y entender aspectos de datos nuevos**.

Mientras estos métodos pueden ser increíblemente poderosos, para ser efectivos deben utilizarse con un **sólido conocimiento de las fortalezas y debilidades de cada método**, así como un entendimiento general de algunos conceptos, entre otros:



Metodología

Tres etapas del Flujo de trabajo típico con scikit-learn



DigitalHouse > Coding School

La mejor forma de pensar un dataset es en términos de tablas de datos.

Observaciones

Variable objetivo o target (y)

Features (X)

```
fruits.head(10)
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89

```
fruits.shape
```

```
(59, 7)
```

```
{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}
```

Este esquema de tabla deja bien en claro que la información puede pensarse como una matriz numérica bidimensional, a la cual llamaremos **Matriz de Features.**

Por convención , esta matriz de features se suele referenciar con una **variable llamada "X".**

Se asume que la matriz de features tiene forma [n_samples, n_features], y frecuentemente se almacena como un **array de Numpy** o un **DataFrame de Pandas**.

Las **muestras (filas)** siempre refieren a **objetos individuales** descritos por el dataset. Por ejemplo, una muestra podría ser una flor, una persona, un documento, una imagen, un archivo de audio, un video o cualquier cosa que pueda describirse con un conjunto de medidas cuantitativas.

Las **features (columnas)** siempre hacen referencia a distintas características que describen cada una de las muestras de manera cuantitativa. Las features pueden ser tanto valores reales como valores booleanos o enteros en algunos casos.

En los problemas de aprendizaje supervisado, además de la matriz de features X, también trabajamos generalmente con un array de **labels** o **target**, que por convención vamos a llamar usualmente **“y”.**

El array target suele ser unidimensional, con **longitud *n_samples***, y generalmente está contenido en un vector de Numpy o una Series de Pandas.

El **array target** puede tener **valores numéricos continuos o etiquetas/clases discretas.**

Un punto que vale la pena recordar: **¿cómo se diferencia el array target de las columnas de la matriz de features?**

La característica distintiva del array target es que es usualmente la cantidad que queremos *predecir a partir de los datos.*

En términos estadísticos, es la **variable dependiente.**

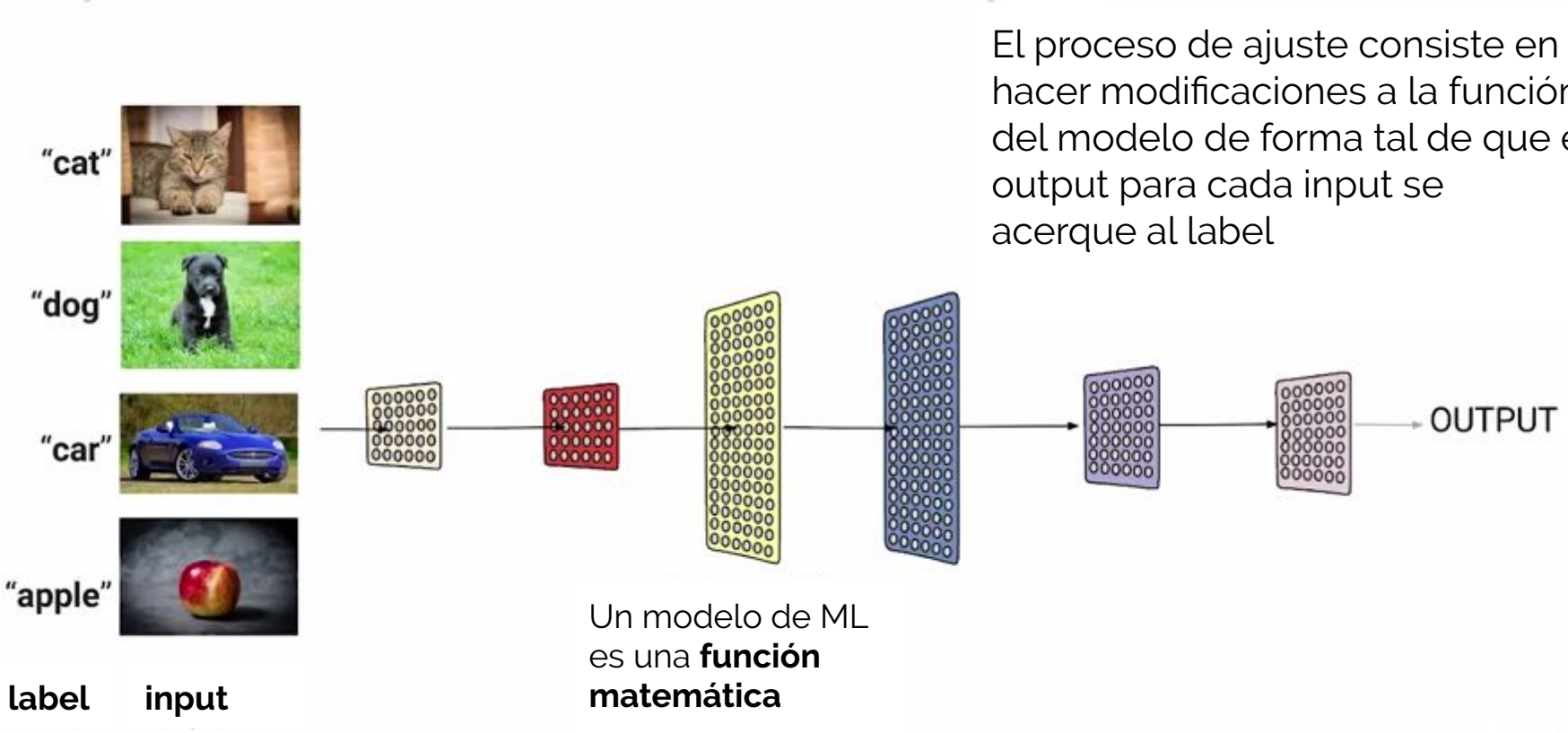
Clasificación de algoritmos en Machine Learning

	APRENDIZAJE SUPERVISADO	APRENDIZAJE NO SUPERVISADO

Continuo	• Regresión
	• Árboles de decisión
	• Random forests
		• Clustering y reducción de dimensionalidad
		  ○ SVD
		  ○ PCA
		  ○ K-means

Categórico	• Clasificación
	  ○ KNN
	  ○ Trees
	  ○ Logistic Regression
	  ○ Naive - Bayes
	  ○ SVM
		• MCA

Aprendizaje Supervisado

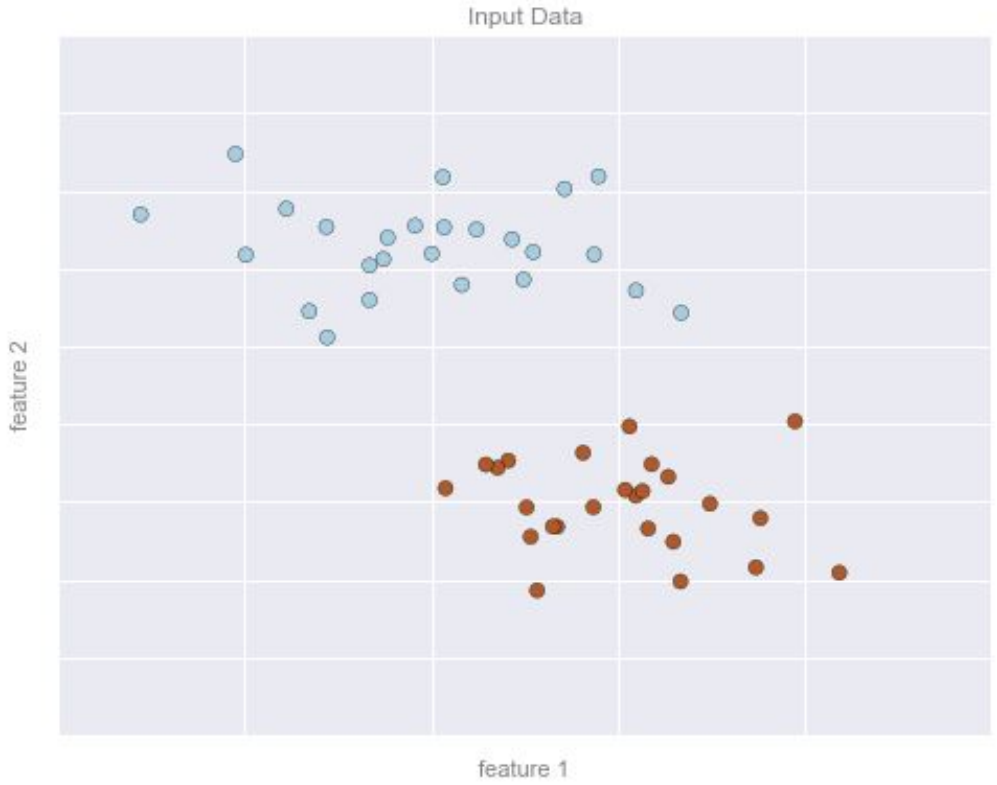


El proceso de ajuste consiste en hacer modificaciones a la función del modelo de forma tal de que el output para cada input se acerque al label

Un modelo de ML es una **función matemática**

Clasificación:

Predecir etiquetas cualitativas



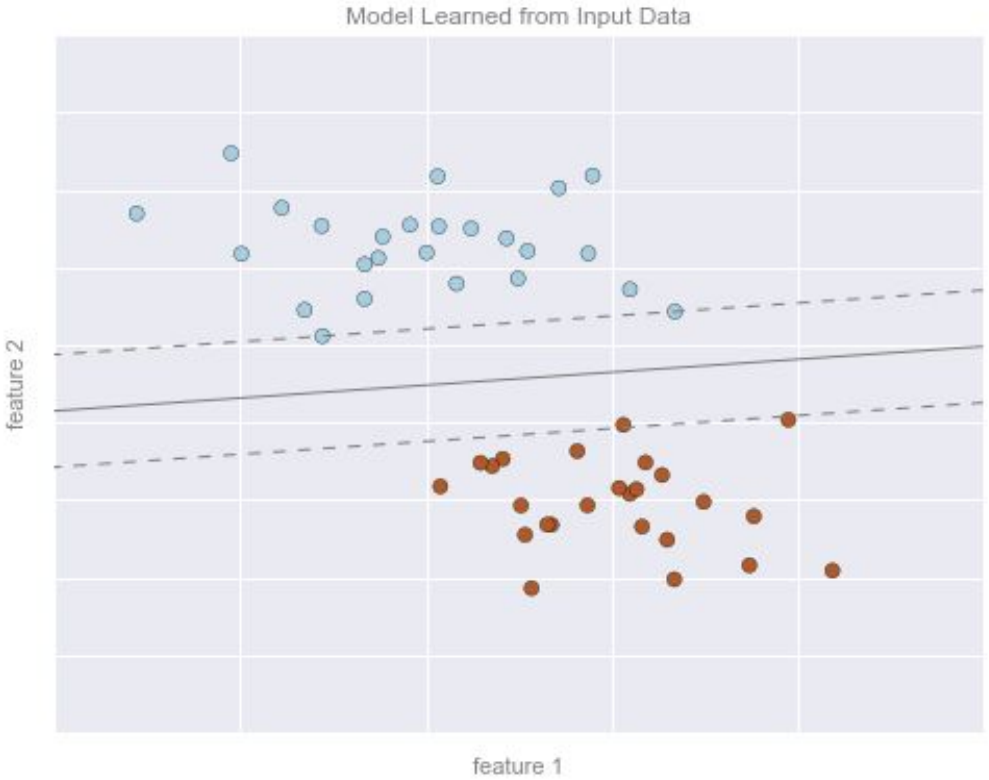
Imaginemos este dataset bi-dimensional. Tenemos dos features por cada punto, representadas por las coordenadas (x, y) en el plano.

Además, cada observación pertenece a una de dos **etiquetas de clase**, siendo la clase representada por el color del punto.

Con esta info queremos **crear un modelo** que nos permita decidir si un **nuevo punto** debería ser etiquetado como azul o rojo.

Hay varios modelos para esta tarea de clasificación, pero elegimos uno extremadamente simple

Asumimos que los dos grupos pueden separarse dibujando una **línea recta** en el plano, tal que los puntos a cada lado de la línea pertenecen al mismo grupo.

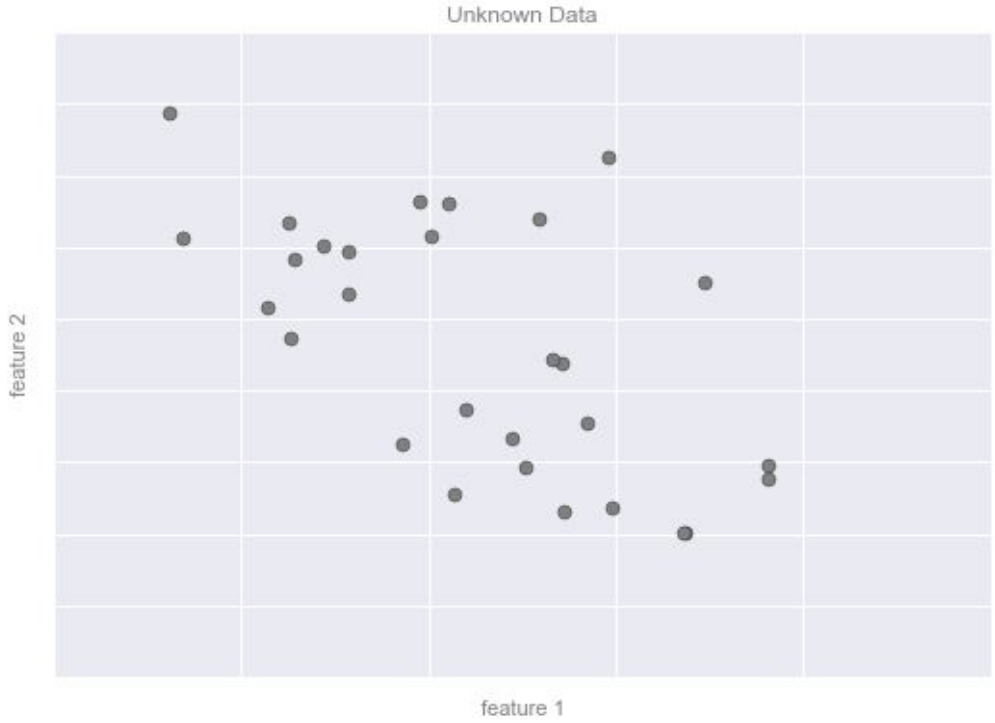


Entonces, el **modelo** es una versión cuantitativa de la afirmación "una línea separa las clases", y los **parámetros del modelo** son los números que describen la ubicación y orientación de esa línea para nuestros datos.

Los **valores óptimos** para estos **parámetros** del modelo se **aprenden de los datos**, proceso que se conoce usualmente como **entrenar el modelo.**

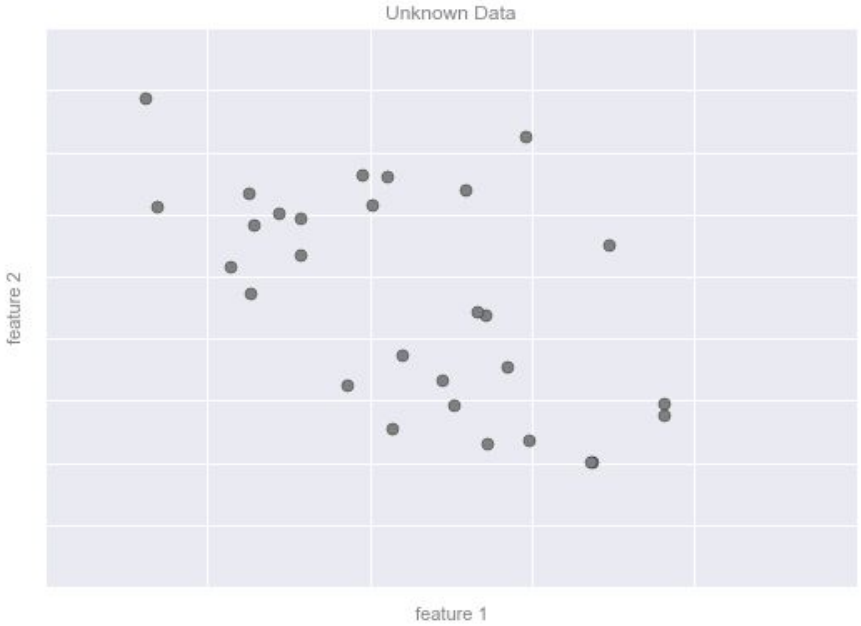
La figura muestra una representación visual de cómo luce el modelo entrenado para estos datos.

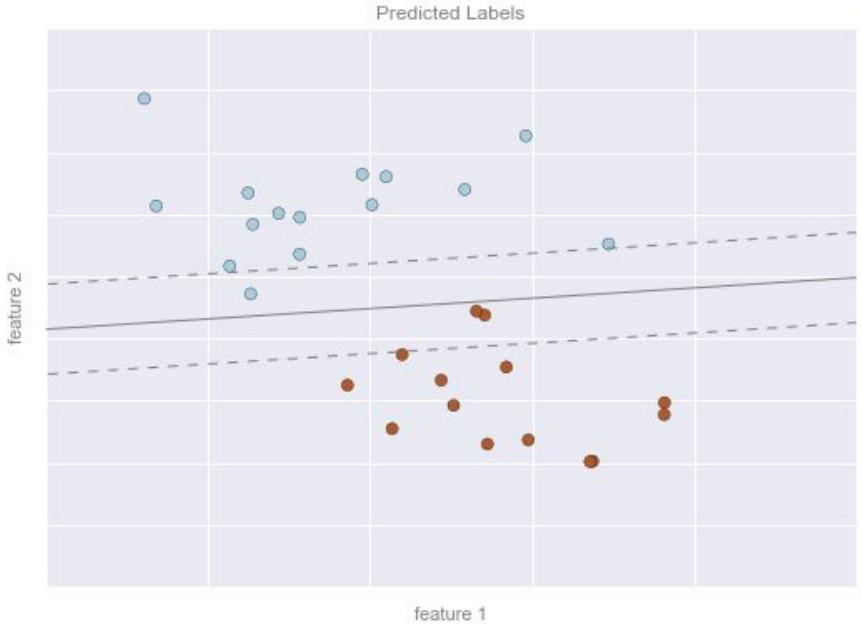
Clasificación: nuevos datos



Ahora que este modelo ha sido entrenado, **puede generalizarse a nuevos datos** sin etiquetas.

Podemos tomar un nuevo dataset, dibujar la línea de nuestro modelo y **asignar etiquetas a los nuevos puntos**, basándonos en qué lado de la línea quedaron.


Unknown Data
feature 2
feature 1


Predicted Labels
feature 2
feature 1

Por ejemplo, lo anterior es similar a la tarea de detección automática de email spam. En este caso podríamos usar la siguiente lista de features y labels:

- *feature 1*, *feature 2*, etc. → recuentos normalizados de palabras o frases importantes ("Free offer", "Congratulations", etc.)
- *label* --> "spam" o "not spam"

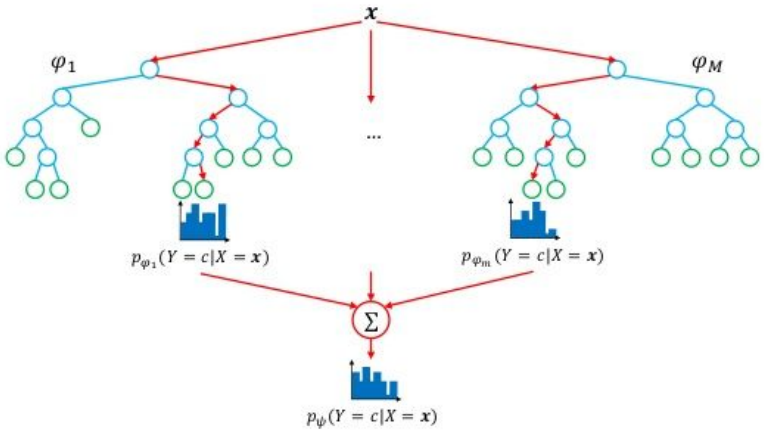
Para el **set de entrenamiento**, estas etiquetas podrían ser determinadas por una inspección individual de una muestra representativa de emails; para el resto de los emails, las labels serían determinadas usando el modelo.

Con un algoritmo de clasificación correctamente entrenado con suficientes features bien construidas (típicamente miles o millones de palabras o frases), este tipo de aproximación puede ser muy efectivo.

Veremos un ejemplo de clasificación basada en texto cuando estudiemos **Clasificación usando Naive Bayes.**

Algunos algoritmos de clasificación que discutiremos en detalle son:

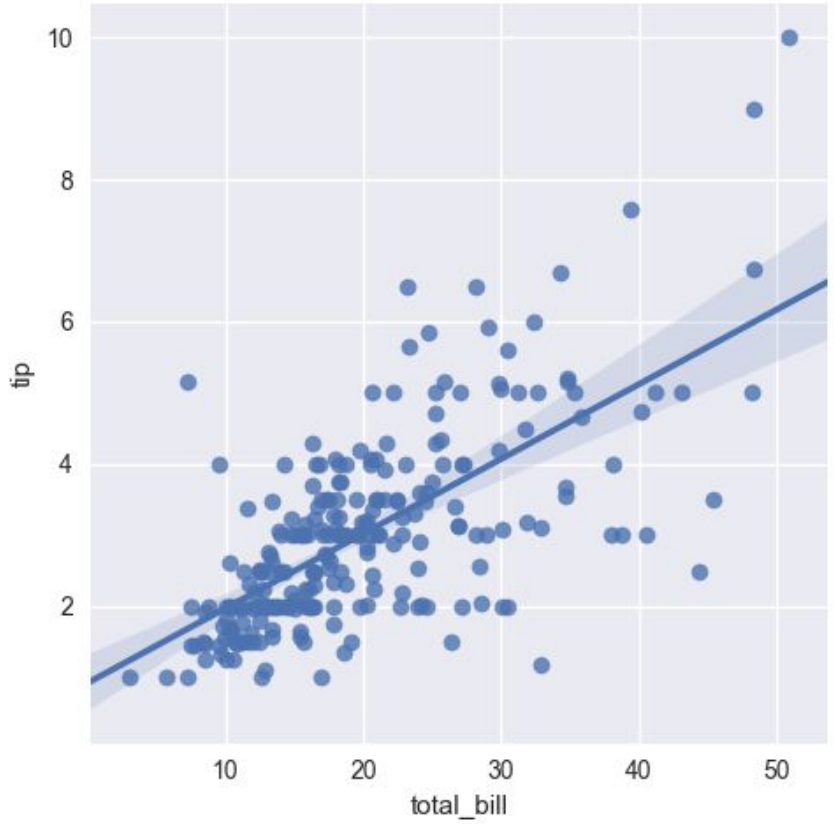
- Gaussian Naive Bayes
- Regresión Logística
- Árboles de Decisión y Random Forest



Regresión:

Predecir etiquetas cuantitativas

Un ejemplo simple con el dataset “tips”:



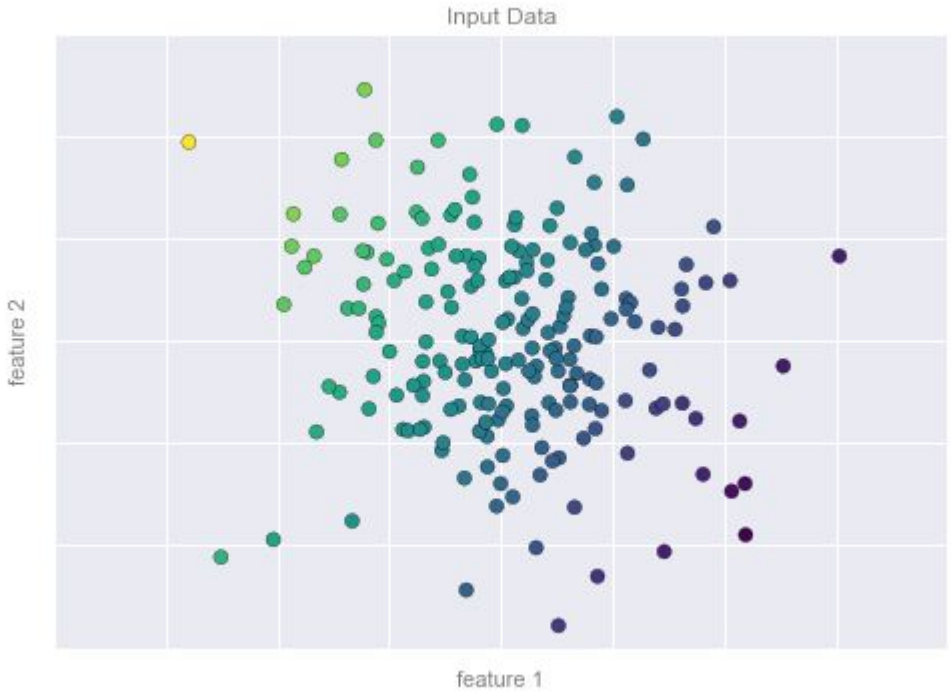
Regresión: Un ejemplo con dos features

Consideremos el dataset mostrado en la figura. Consiste en un conjunto de puntos, cada uno con una etiqueta cuantitativa.

Datos bidimensionales: tenemos dos features describiendo cada punto. El color de cada punto representa el valor contínuo de la etiqueta para esa observación.

Por ejemplo si queremos predecir performance de alumnos: las variables independientes podrían ser horas de sueño y horas de estudio.

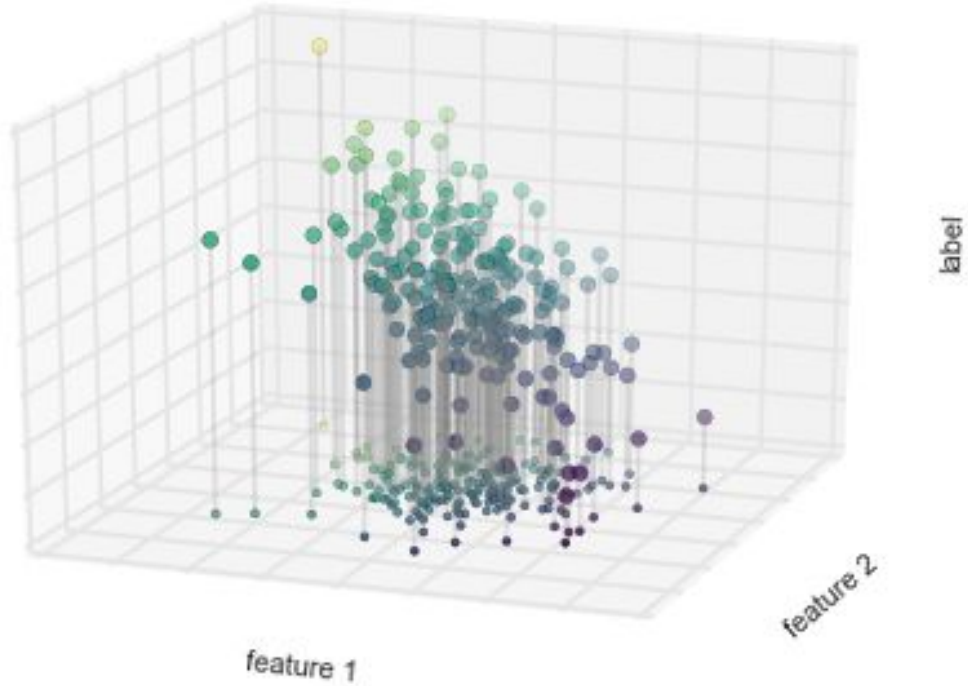
La variable dependiente podría ser los scores de un test.



Regresión: Un ejemplo con dos features

Hay varios modelos de regresión posible que podríamos usar para este tipo de datos, pero aquí usaremos una **regresión lineal para predecir el valor de las etiquetas de nuevos puntos.**

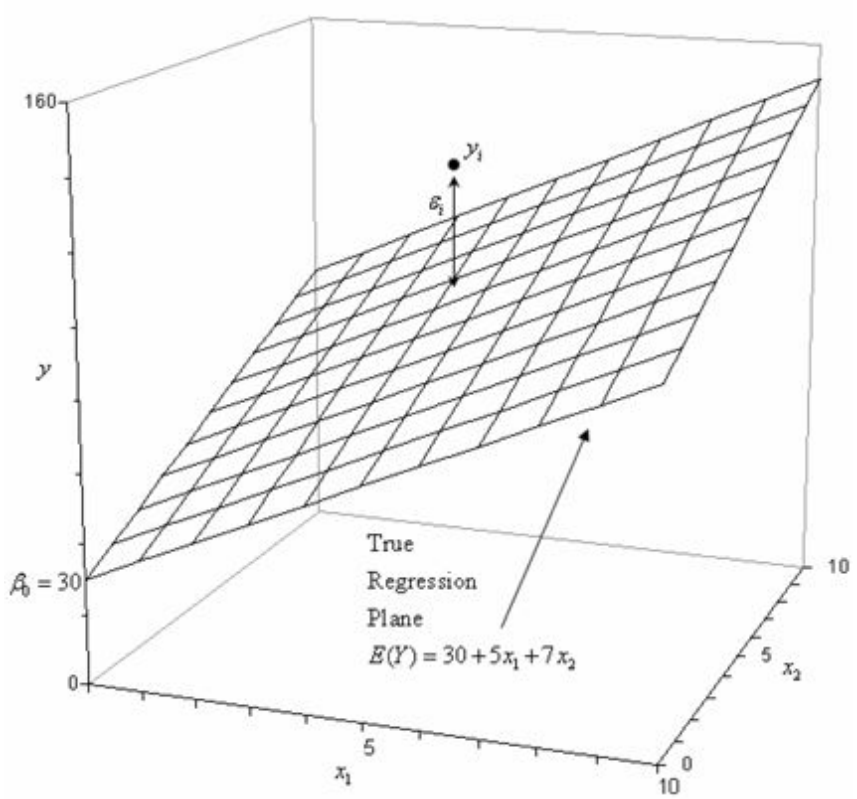
Imaginemos que **interpretamos la etiqueta como una tercera dimensión espacial**. Entonces podríamos ajustar un plano a nuestros datos.

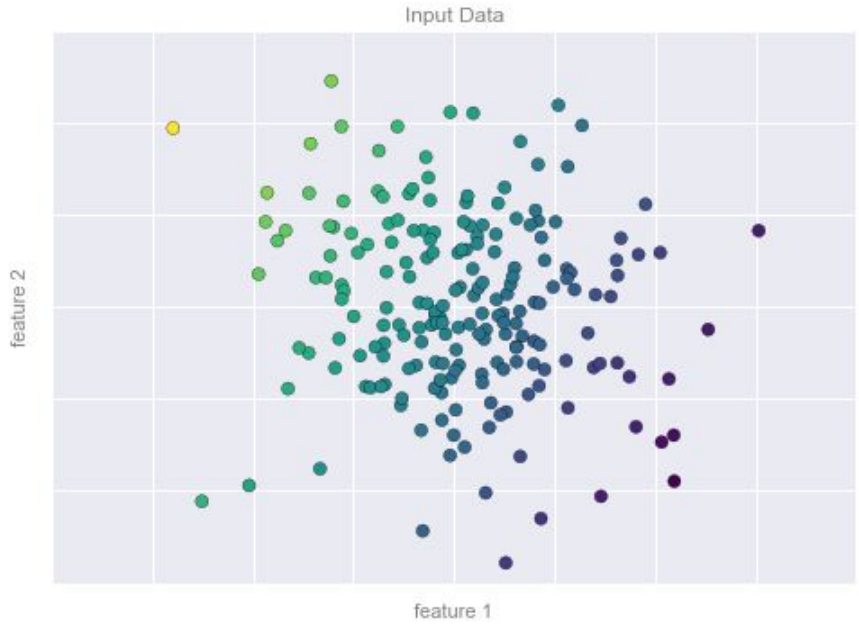


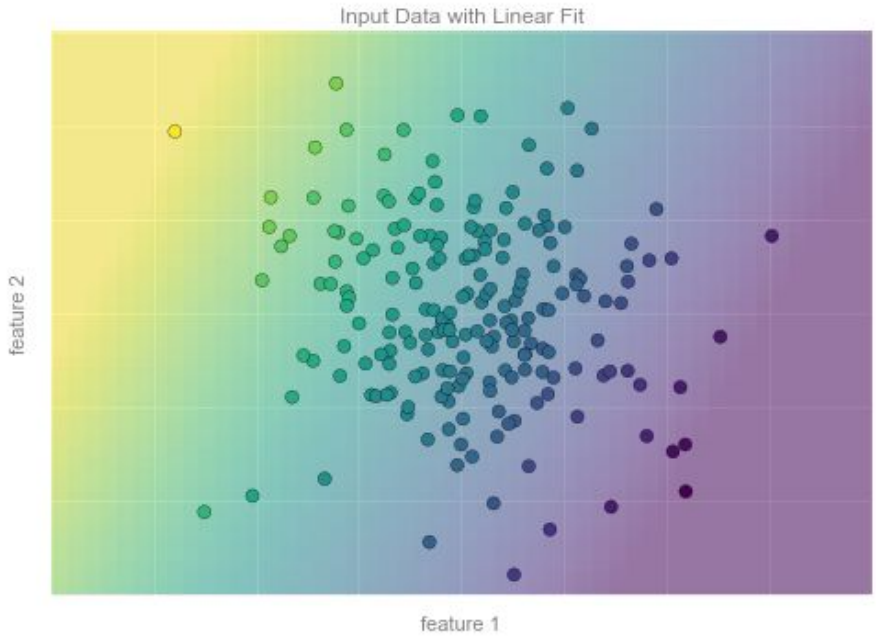
Notar que aquí el plano feature 1 - feature 2 es el mismo que la slide anterior. En este caso, sin embargo, hemos representado el valor contínuo de la etiqueta con un color y su posición en el eje de la tercer dimensión

Parece razonable que **ajustar un plano a estos datos en tres dimensiones** nos permitirá predecir una etiqueta para cada conjunto de datos nuevos.

Esta es una **generalización** de alto nivel del conocido problema de ajustar una línea a datos con una feature y una etiqueta.

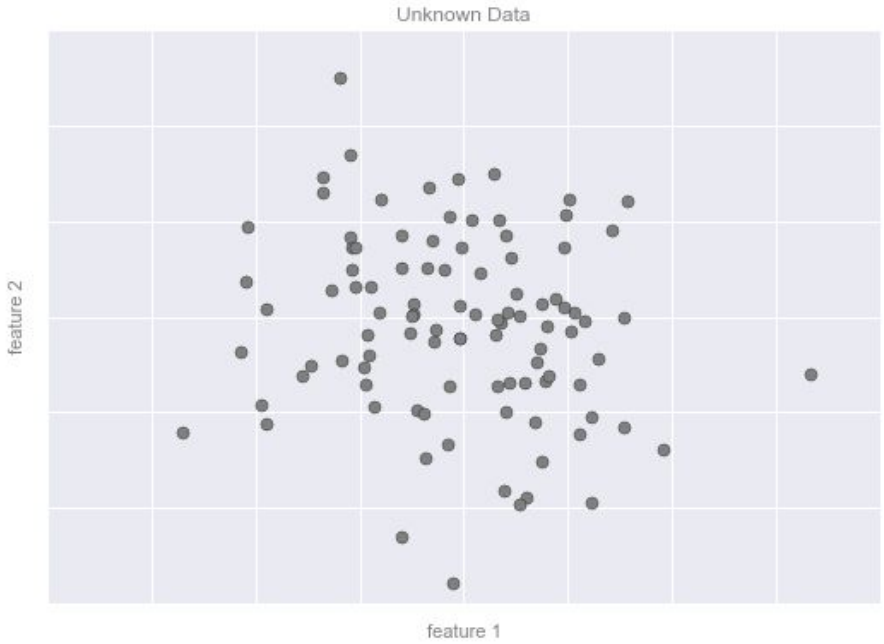


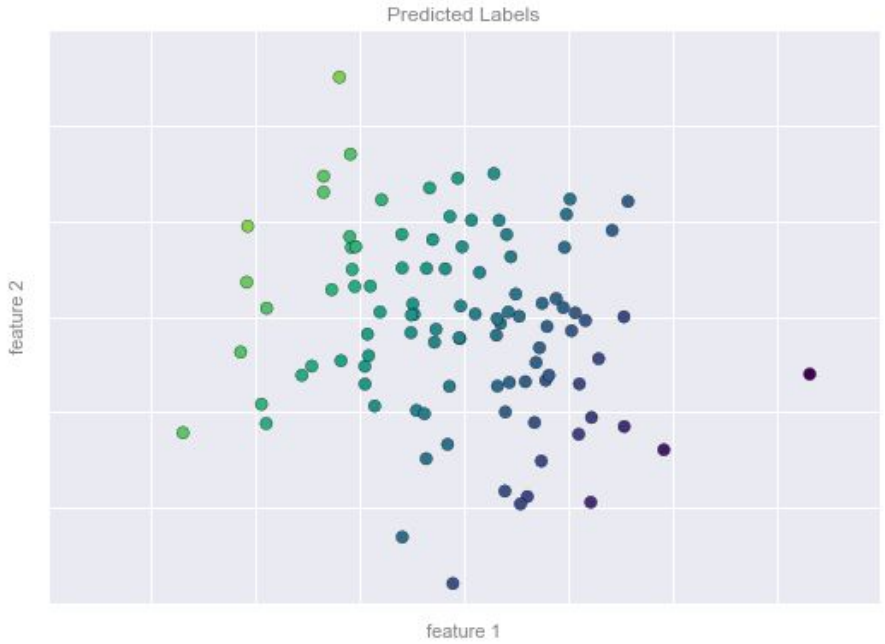




Retornando a la **proyección en dos dimensiones**: vemos los datos y vemos el ajuste lineal, codificado con un contínuo de colores

El plano de ajuste nos da lo que necesitamos para **predecir las etiquetas de los nuevos puntos**.





Como en el ejemplo de clasificación, esto puede parecer trivial con un número bajo de dimensiones. Pero el poder de estos métodos radica en que pueden ser aplicados y evaluados directamente a casos de datasets con un gran número de dimensiones.

Por ejemplo, podemos computar la **distancia a galaxias** observadas a través de un telescopio - en este caso podemos usar las siguientes features:

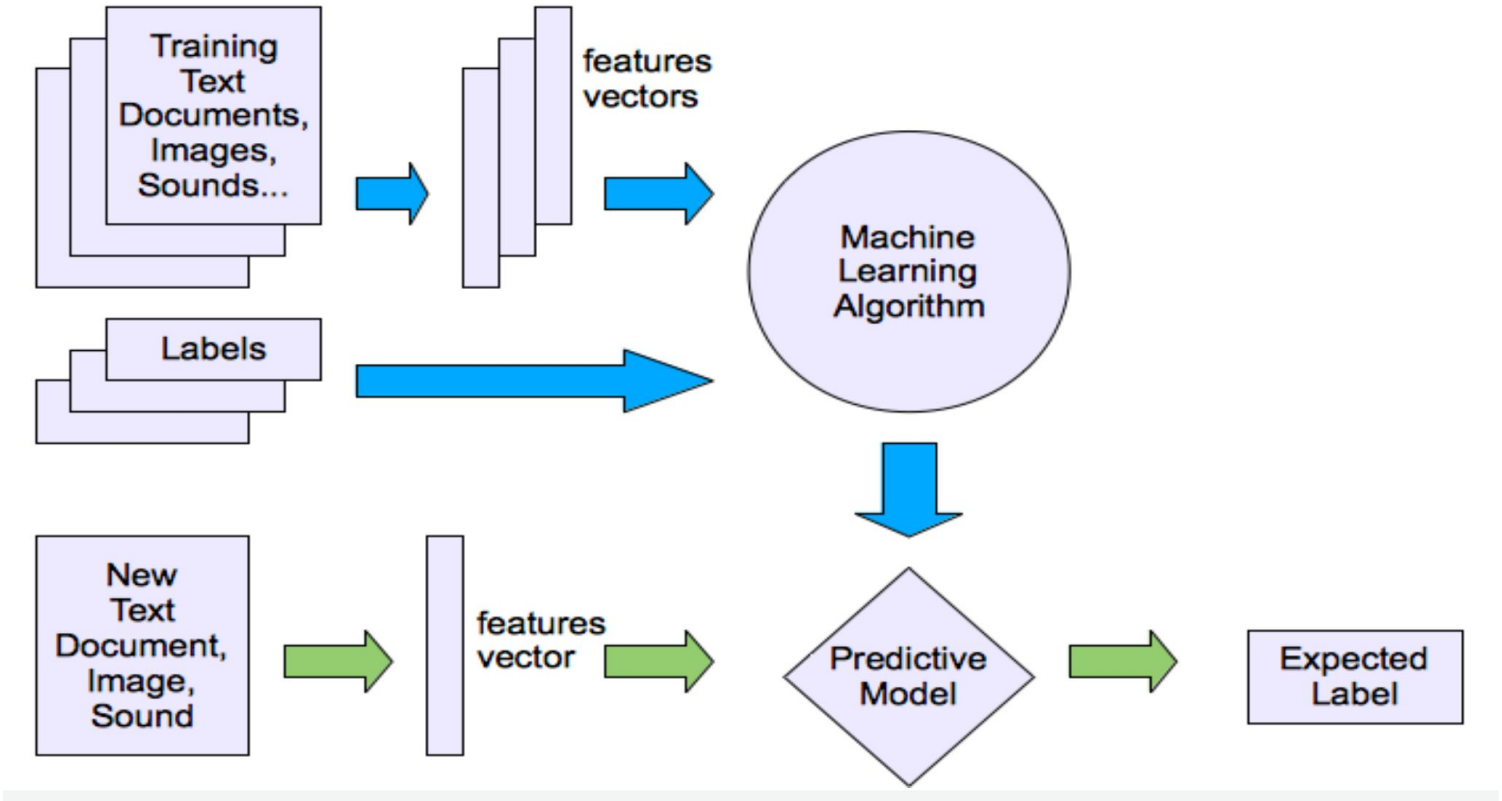
- *feature 1*, *feature 2*, etc. --> brillo de cada galaxia en una de varias longitudes de ondas o colores
- *label* <-- distancia de la galaxia

Las distancias para un pequeño número de galaxias podría determinarse a través de observaciones independientes, usualmente muy costosas.

Las distancias al resto de las galaxias podría luego ser estimadas usando un modelo de regresión apropiado, sin la necesidad de emplear los métodos de observación más costosos.

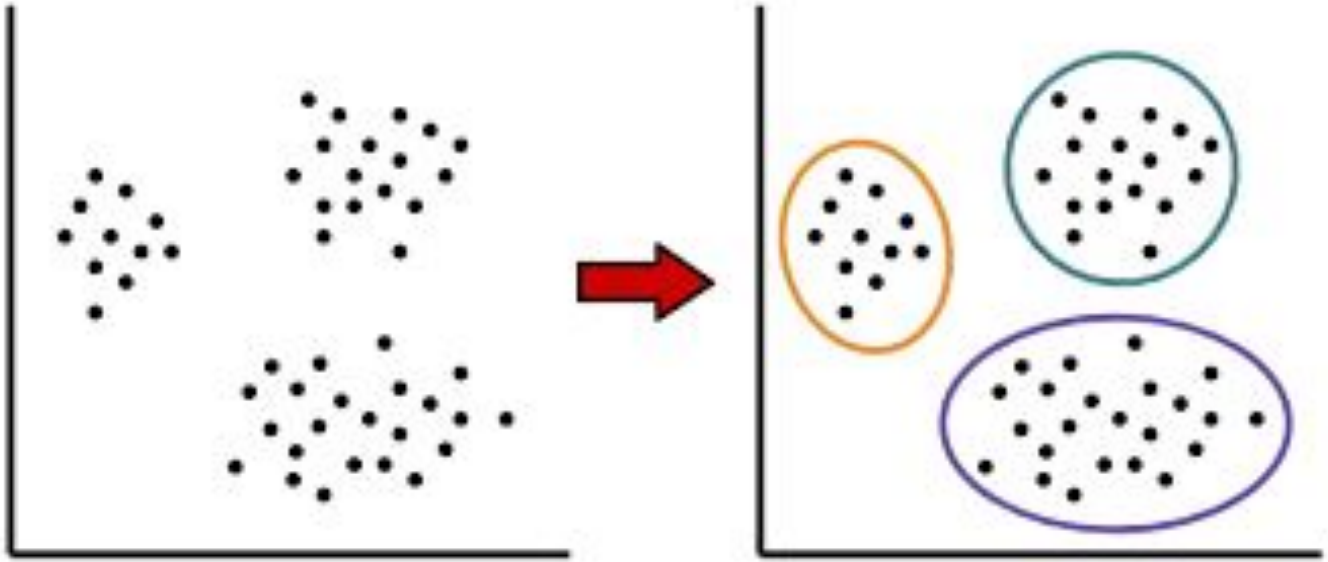
En terminología astronómica, esto es conocido como el problema "photometric redshift"

Supervised learning pipeline



Aprendizaje no supervisado

El aprendizaje no supervisado involucra modelos que describen los datos **sin referencia a ninguna etiqueta conocida**.

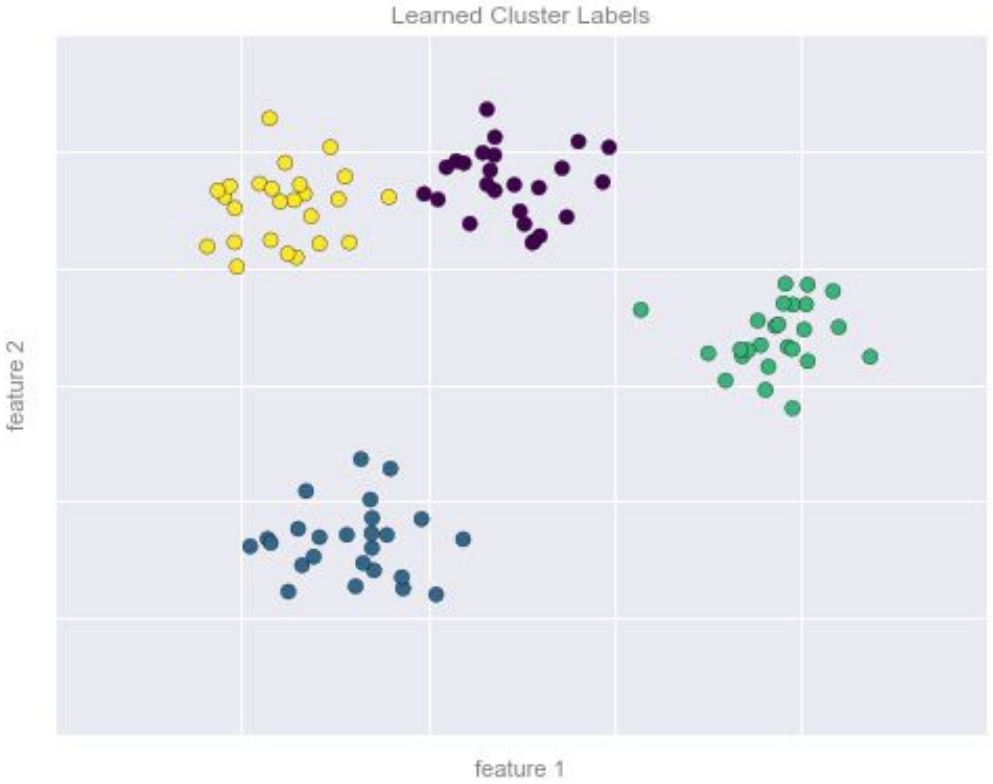


Clustering: Inferir etiquetas en datos no etiquetados

En clustering, a los datos se **los asigna automáticamente a algún número de grupos discretos.**

Por ejemplo, podríamos tener un dataset bidimensional como el que se muestra en la figura.

Visualmente cada uno de estos puntos parecería formar parte de un grupo excluyente.



Dado este input, un modelo de clustering usará la **estructura intrínseca** de los datos para determinar qué puntos están **relacionados**.

Usando el algoritmo **k-means** (muy rápido e intuitivo), encontramos los cluster mostrados en la figura

k-means ajusta un modelo que consiste en k centros de clusters; se asume que los centros óptimos son aquellos que minimizan la distancia de cada punto a su centro asignado.

Nuevamente, esto parecería ser un ejercicio trivial en dos dimensiones, pero a medida que nuestros datos se vuelven más grandes y complejos, estos algoritmos de clustering pueden ser utilizados para extraer información útil de nuestros datasets.

Vamos a discutir el algoritmo k-means en detalle más adelante. Otros algoritmos importantes que vamos a estudiar son:

- Clustering jerárquico
- DBSCAN

Reducción de dimensionalidad: Inferir estructura en los features

La reducción de dimensionalidad es otro ejemplo de algoritmo no supervisado, en el cual las etiquetas u otra **información es inferida de la estructura del propio dataset.**

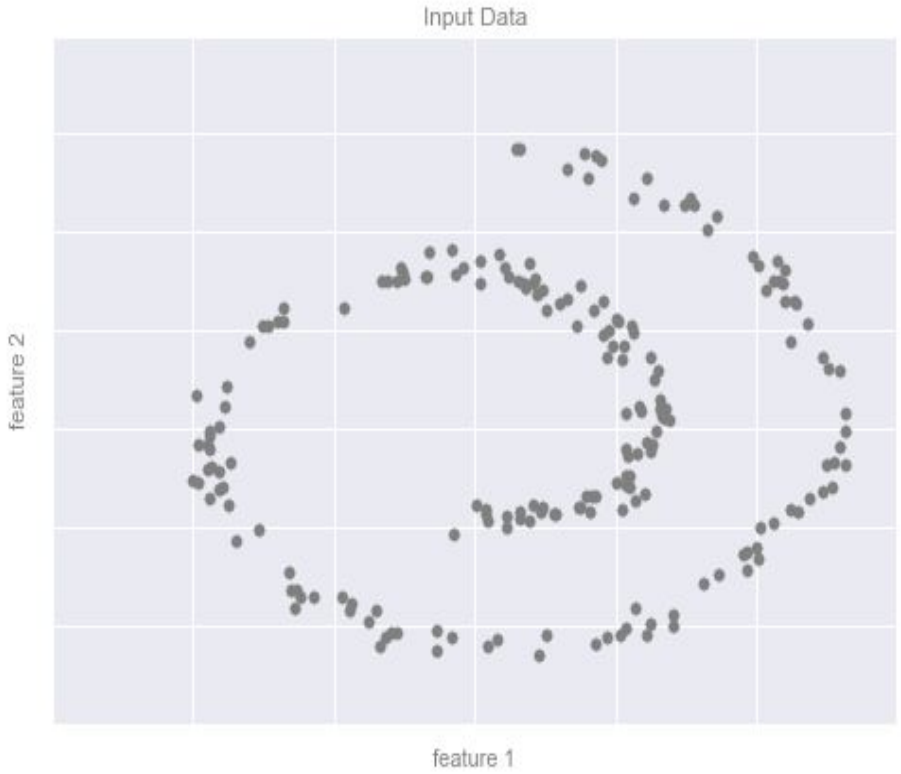
Es una técnica un poco más abstracta que los ejemplos vistos previamente pero, en términos generales, **busca obtener, de un dataset complejo, una representación de baja dimensionalidad, que de alguna manera preserve cualidades relevantes del dataset original.**

¿Qué estructura vemos en estos datos?

En cierto sentido, podríamos decir que este dataset es ***intrínsecamente*** **unidimensional.**

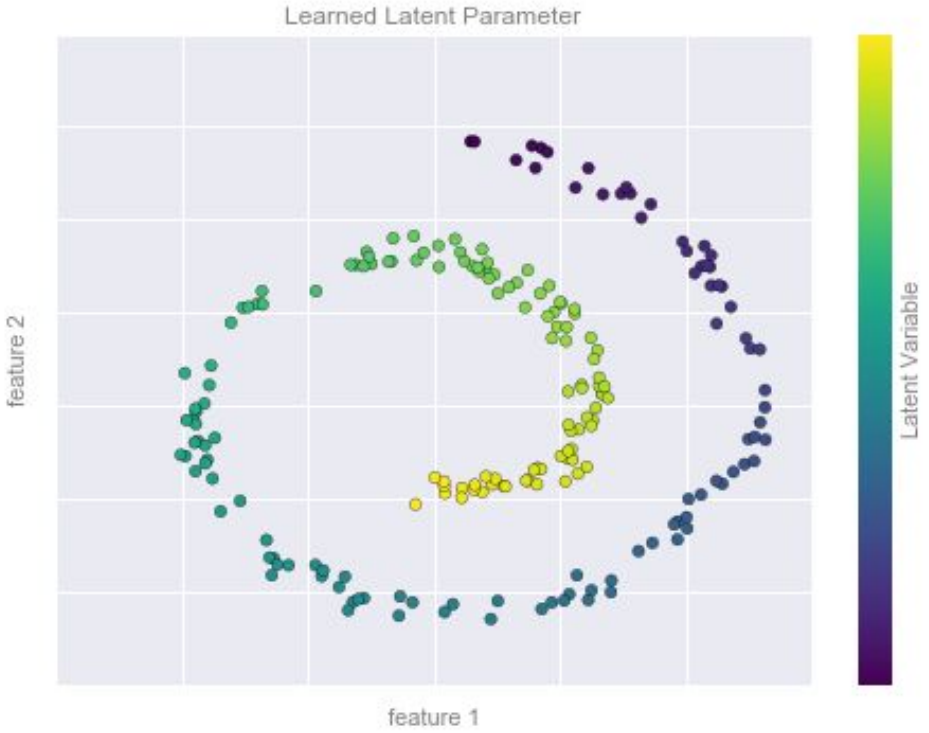
Sin embargo, esta data unidimensional está embebida en **un espacio de mayor dimensionalidad.**

Un modelo apropiado de reducción de dimensionalidad en este caso sería **sensible a esta estructura embebida no lineal**, y sería capaz de extraer una representación de baja dimensionalidad.



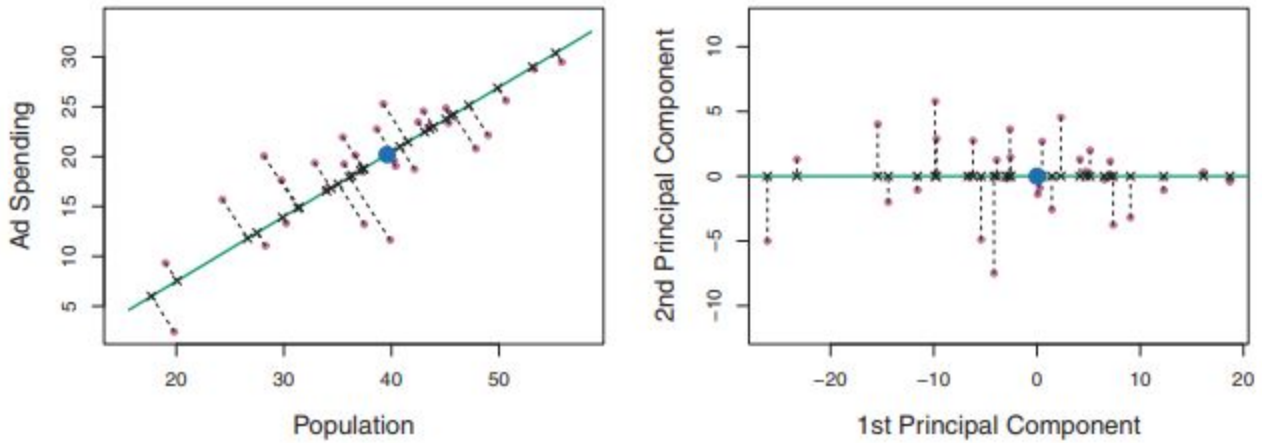
La figura muestra una visualización de los resultados del algoritmo **Isomap**, un ejemplo de un tipo de técnica conocida como **manifold learning**.

Notar que los colores (que representan la variable latente unidimensional extraída) cambian uniformemente a lo largo de la espiral, lo que indica que **el algoritmo detectó efectivamente la estructura** observada a simple vista.

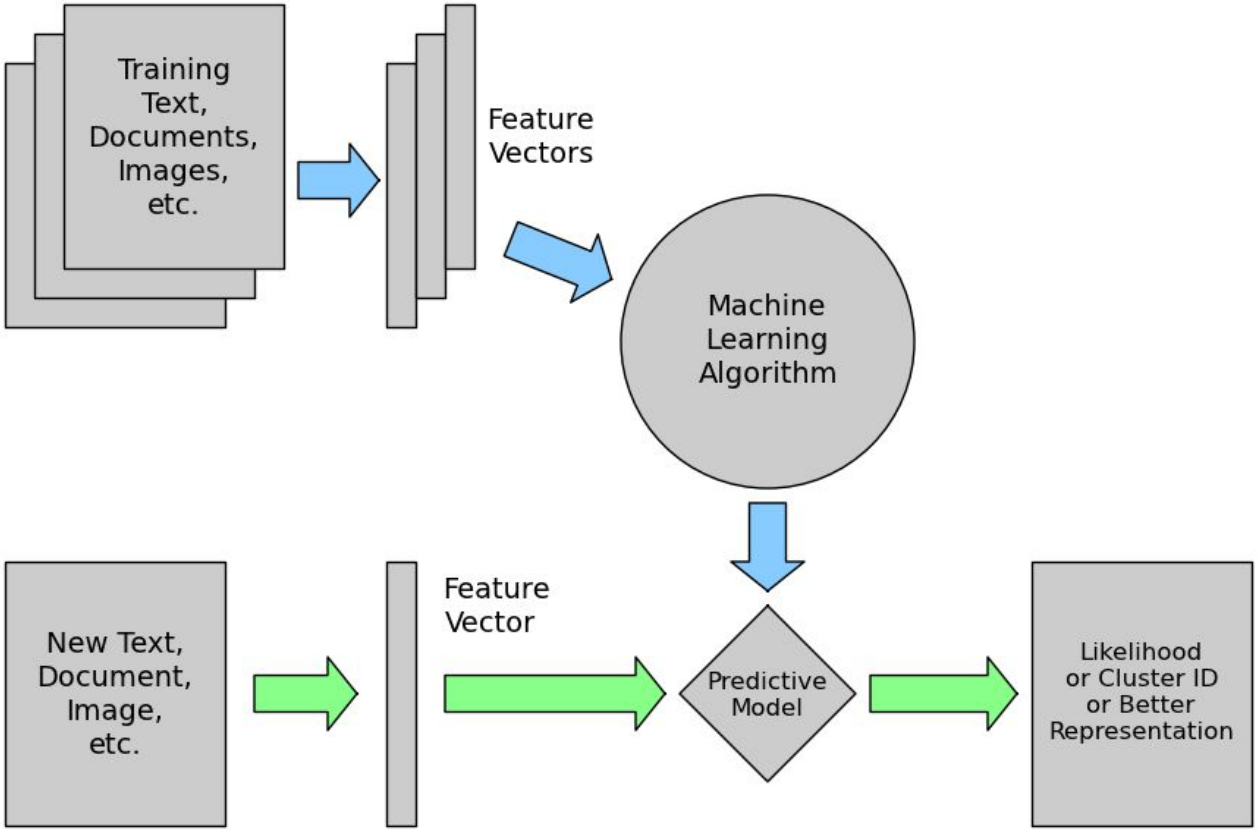


Recordemos algunos algoritmos importantes de esta categoría que estudiamos en el módulo 2::

- Principal Component Analysis (PCA)
- Manifold Learning
  - Isomap
  - T-SNE



Unsupervised learning pipeline

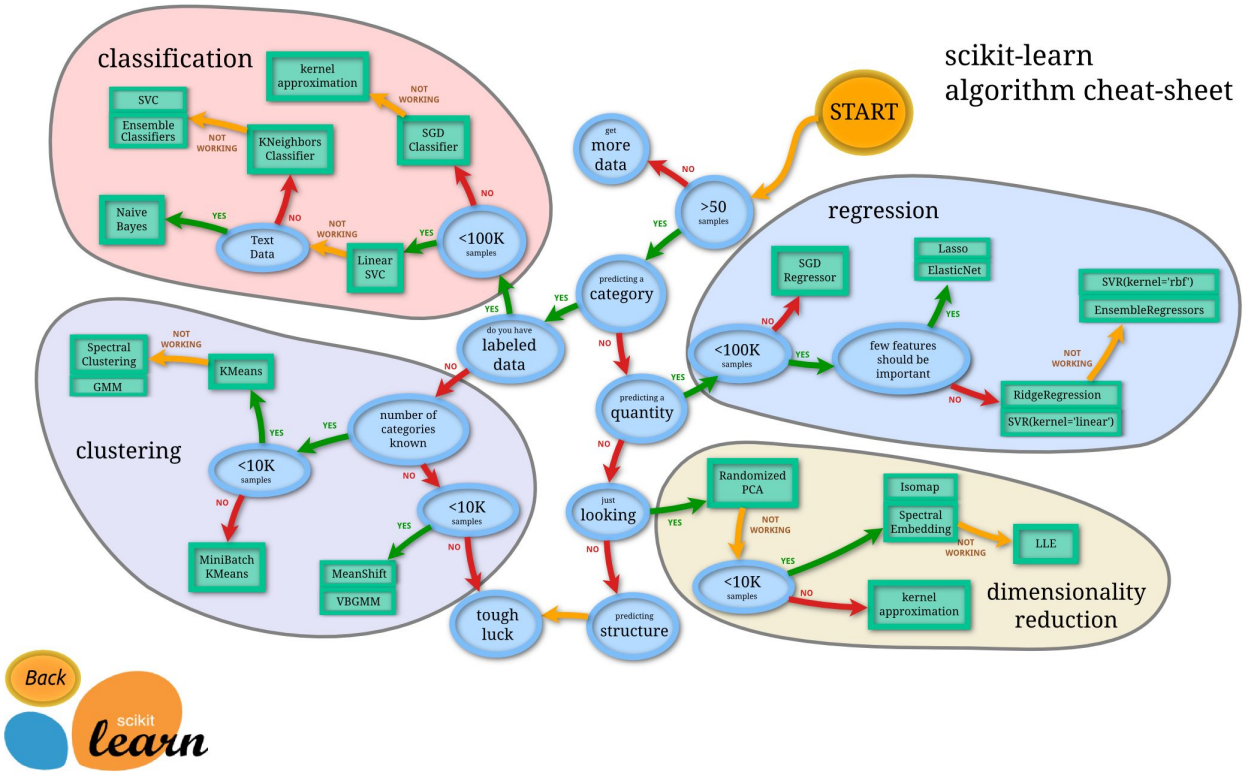


Scikit-learn

Machine Learning en Python

Hay varias librerías de Python que proveen sólidas implementaciones de muchos de algoritmos de machine learning.

Una de las más conocidas es **Scikit-Learn**, una implementación que provee versiones eficientes de un gran número de los algoritmos más usados.



Seis razones para usarlo

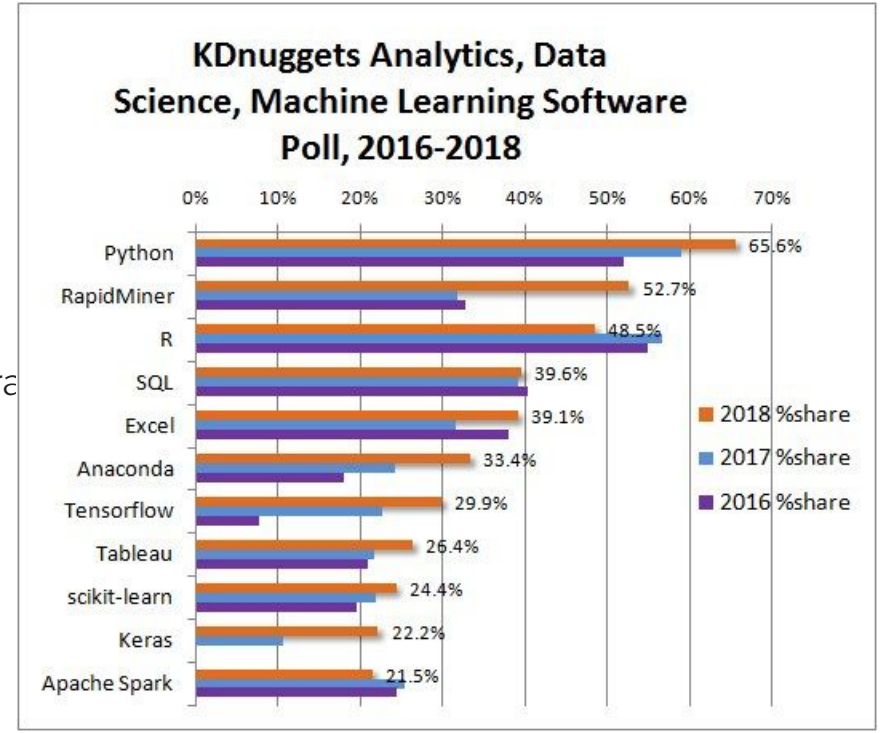
1. Compromiso con la **documentación y usabilidad**.

2. Los modelos son elegidos e **implementados por un equipo de expertos**. El grupo de colaboradores de Scikit-learn incluye expertos en machine learning y desarrollo de software.

Seis razones para usarlo

3. Scikit-learn **cubre la mayoría de las tareas de machine learning**. Incluye herramientas para muchas de las tareas estándar de machine learning (como clustering, clasificación, regresión, etc.).

4. Un conjunto impresionante de herramientas para el **procesamiento de datos con Python** han surgido en los últimos años.



Python overtakes R, becomes the leader in Data Science, Machine Learning platforms.
KDnuggets, Agosto 2017.

Seis razones para usarlo

5. **Foco**: Scikit-learn es una biblioteca de machine learning. Su objetivo es proveer a los usuarios un conjunto de algoritmos a través de una interface consistente.

6. **Escalabilidad** Scikit-learn es escalable a la mayoría de los datasets que usamos en este contexto. Muchos problemas pueden encararse usando un único servidor (con mucha memoria), reduciendo la complejidad del desarrollo de sistemas distribuidos.

Hemos visto algunos ejemplos simple de los tipos básicos de algoritmos de machine learning y qué problemas podrían resolver.

Debe quedar claro que hemos ignorado una enorme cantidad de detalles técnicos que iremos estudiando a lo largo del curso.

En resumen, hemos presentado:

- **Aprendizaje Supervisado:** Modelos que pueden predecir etiquetas basándose en datos de entrenamiento etiquetados
  - *Clasificación*: Modelos que predicen etiquetas cualitativas
  - *Regresión*: Modelos que predicen etiquetas cuantitativas

- **Aprendizaje No Supervisado**: Modelos que identifican estructura en datos sin etiquetar.
  - Clustering: Modelos que identifican grupos en las observaciones.
  - Reducción de la dimensionalidad: Modelos que detectan e identifican estructuras de menor dimensión en los features.

- Las características principales de **Scikit-learn**