Aprendizaje Automático (Parte B)
Introducción a Scikit-Learn

Marcelo Luis Errecalde[1,2]

[1]Universidad Nacional de San Luis, Argentina
[2]Universidad Nacional de la Patagonia Austral, Argentina
e-mails: merreca@unsl.edu.ar, merrecalde@gmail.com

Curso: Minería de Datos
Universidad Nacional de San Luis - Año 2018

Resumen

Los datos como tablas

- El aprendizaje automático crea modelos a partir de datos.

Los datos como tablas

- El aprendizaje automático crea modelos a partir de datos.
- Primer paso: discutir cómo representar esos datos.

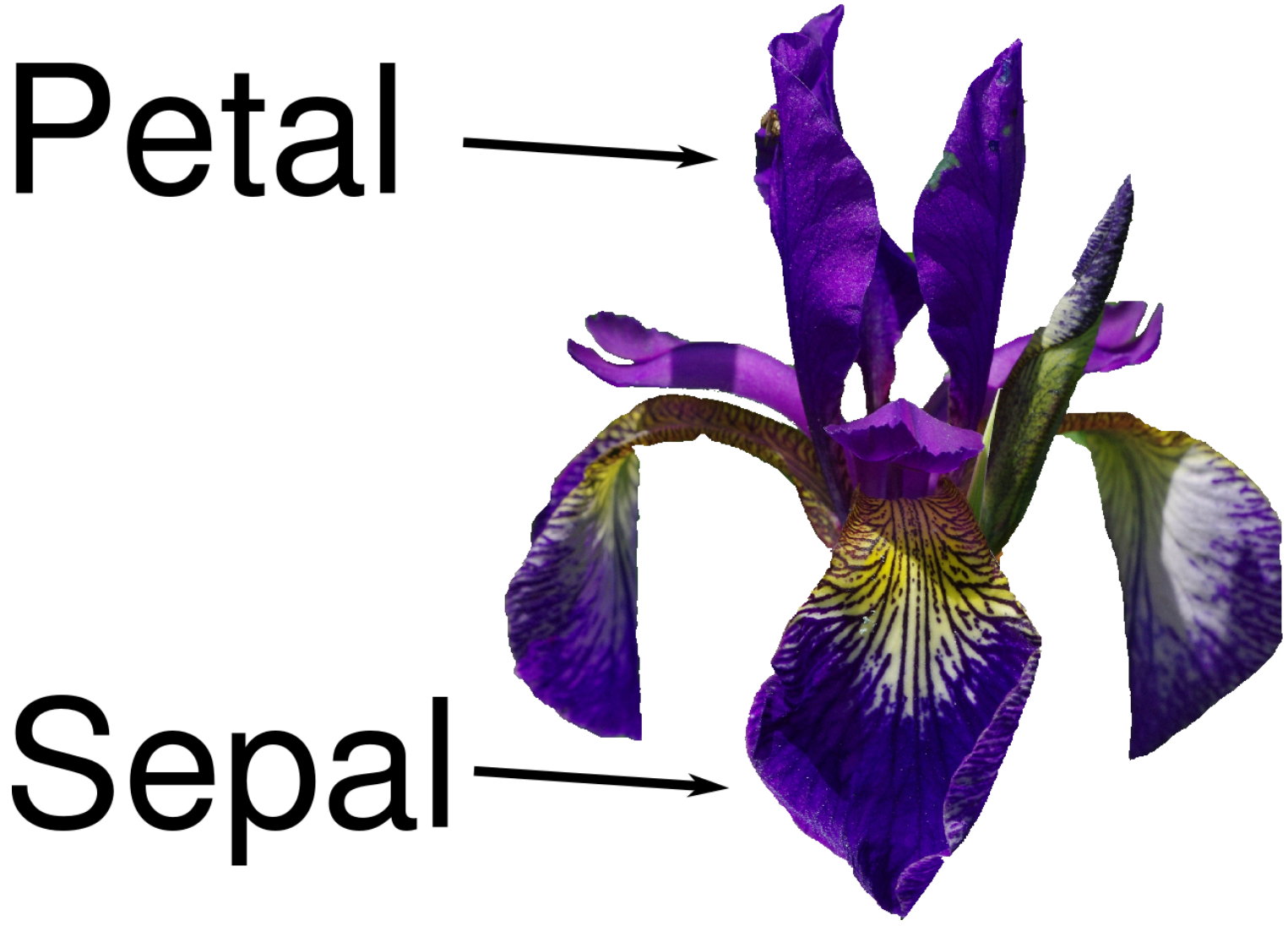
Los datos como tablas

- El aprendizaje automático crea modelos a partir de datos.
- Primer paso: discutir cómo representar esos datos.
- La mejor manera de ver a los datos en Scikit-Learn es como tablas de datos.

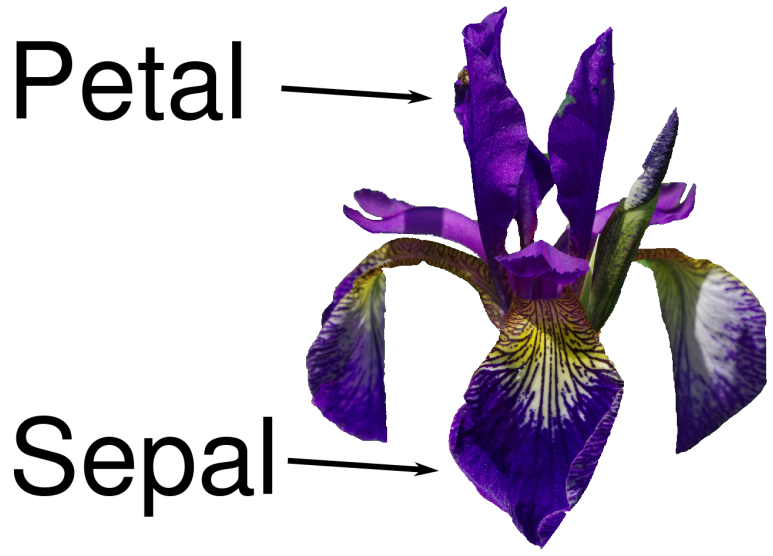
Los datos como tablas

- El aprendizaje automático crea modelos a partir de datos.
- Primer paso: discutir cómo representar esos datos.
- La mejor manera de ver a los datos en Scikit-Learn es como tablas de datos.
- Una tabla es una grilla (matriz) de 2 dimensiones donde las filas representan elementos individuales (individuos) del conjunto de datos y las columnas representan valores relacionados a las propiedades de estos elementos.

Ejemplo: el conjunto de datos `Iris`

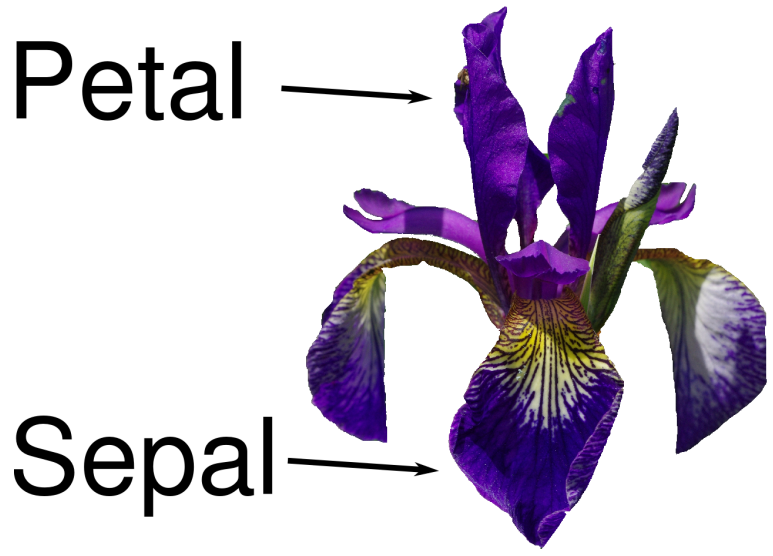


Ejemplo: el conjunto de datos `Iris`



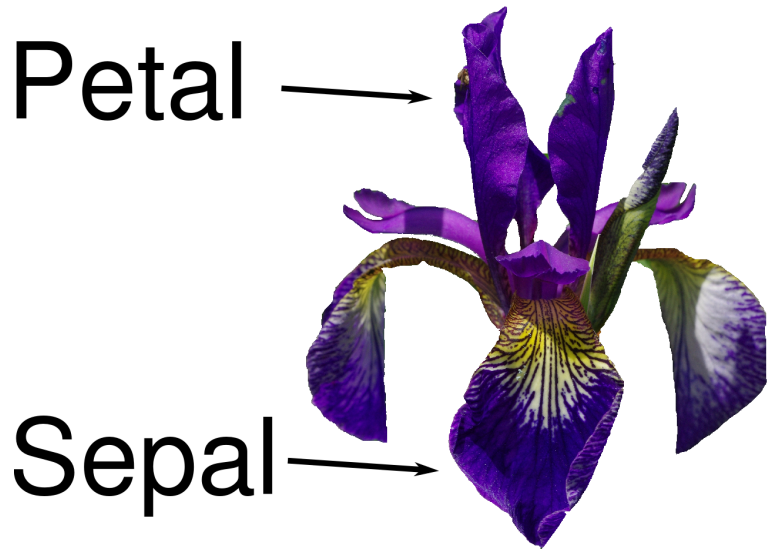
- Iris: Conjunto de datos que se remonta al trabajo del eminente estadístico Ronald Fisher a mediados de la década de 1930.

Ejemplo: el conjunto de datos `Iris`



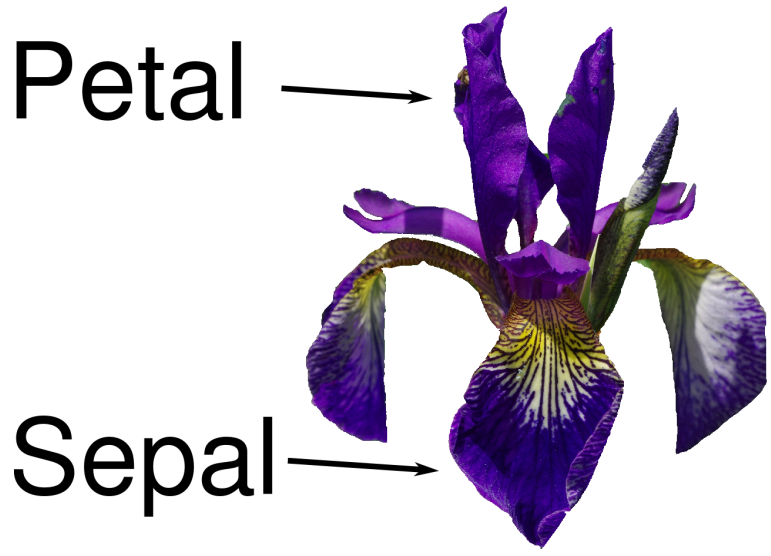
- Iris: Conjunto de datos que se remonta al trabajo del eminente estadístico Ronald Fisher a mediados de la década de 1930.
- Tal vez el conjunto de datos más famoso utilizado en minería de datos.

Ejemplo: el conjunto de datos `Iris`



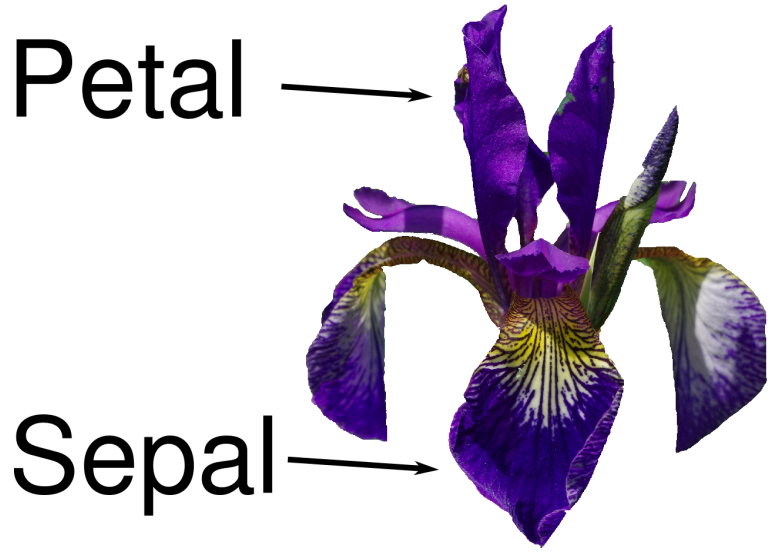
- Iris: Conjunto de datos que se remonta al trabajo del eminente estadístico Ronald Fisher a mediados de la década de 1930.
- Tal vez el conjunto de datos más famoso utilizado en minería de datos.
- Disponible en el repositorio de la Universidad de California en Irvine (repositorio UCI, Iris Plant Database).

Ejemplo: el conjunto de datos `Iris`



- Contiene información de 150 flores de lirio, con 50 ejemplos de cada una de las siguientes especies de lirio: Setosa, Versicolour y Virginica.
- Cada flor de lirio es caracterizada por 5 atributos
  - 4 numéricos con las características (features) de la flor
  - 1 nominal con la clase (especie) de la flor

Ejemplo: el conjunto de datos `Iris`



- Atributos de un lirio:
  1. `sepal_length`: longitud del sépalo (en centímetros).
  2. `sepal_width`: ancho del sépalo (en centímetros).
  3. `petal_length`: longitud del pétalo (en centímetros).
  4. `petal_width`: longitud del pétalo (en centímetros).
  5. `species`: tipo de lirio (setosa, versicolour o virginica).

Los datos como tablas

- Este conjunto de datos puede ser accedido como un `DataFrame` de Pandas usando la biblioteca Seaborn

Los datos como tablas

- Este conjunto de datos puede ser accedido como un `DataFrame` de Pandas usando la biblioteca Seaborn
- Cada fila refiere a cada flor observada (muestra)

```
In [1]: import seaborn as sns
        iris = sns.load_dataset('iris')
        iris.head()
```

```
Out[1]:   sepal_length  sepal_width  petal_length  petal_width species
        0           5.1          3.5           1.4          0.2  setosa
        1           4.9          3.0           1.4          0.2  setosa
        2           4.7          3.2           1.3          0.2  setosa
        3           4.6          3.1           1.5          0.2  setosa
        4           5.0          3.6           1.4          0.2  setosa
```

Los datos como tablas

- El número de filas del `DataFrame` (150 en este caso) es el nro. total de flores en el conjunto de datos
- El número de columnas (5 en este caso) corresponden a los 5 atributos de la colección (4 numéricos y 1 nominal)

```
In [2]: iris.shape

Out[2]: (150,5)
```

Los datos como tablas

- El número de filas del `DataFrame` (150 en este caso) es el nro. total de flores en el conjunto de datos
- El número de columnas (5 en este caso) corresponden a los 5 atributos de la colección (4 numéricos y 1 nominal)

```
In [2]: iris.shape

Out[2]: (150,5)
```

- Sin embargo, de los 5 atributos, hay uno especial (`species`) que difiere de los restantes.

Los datos como tablas

- El número de filas del `DataFrame` (150 en este caso) es el nro. total de flores en el conjunto de datos
- El número de columnas (5 en este caso) corresponden a los 5 atributos de la colección (4 numéricos y 1 nominal)

```
In [2]: iris.shape

Out[2]: (150,5)
```

- Sin embargo, de los 5 atributos, hay uno especial (`species`) que difiere de los restantes.
- Este es el atributo (la clase) que uno intenta predecir

Matriz de características (“features”) y vector objetivo (“target”)

- Scikit-Learn distingue, al menos en tareas predictivas como clasificación y regresión, esos dos tipos de atributos.

Matriz de características (“features”) y vector objetivo (“target”)

- Scikit-Learn distingue, al menos en tareas predictivas como clasificación y regresión, esos dos tipos de atributos.
- Por un lado, las características (“features”) corresponden a los atributos que describen las características o propiedades de las muestras.

Matriz de características (“features”) y vector objetivo (“target”)

- Scikit-Learn distingue, al menos en tareas predictivas como clasificación y regresión, esos dos tipos de atributos.
- Por un lado, las características (“features”) corresponden a los atributos que describen las características o propiedades de las muestras.
- Por el otro, existe un atributo objetivo (“target”) que corresponde a la clase que se busca predecir a partir de las características.

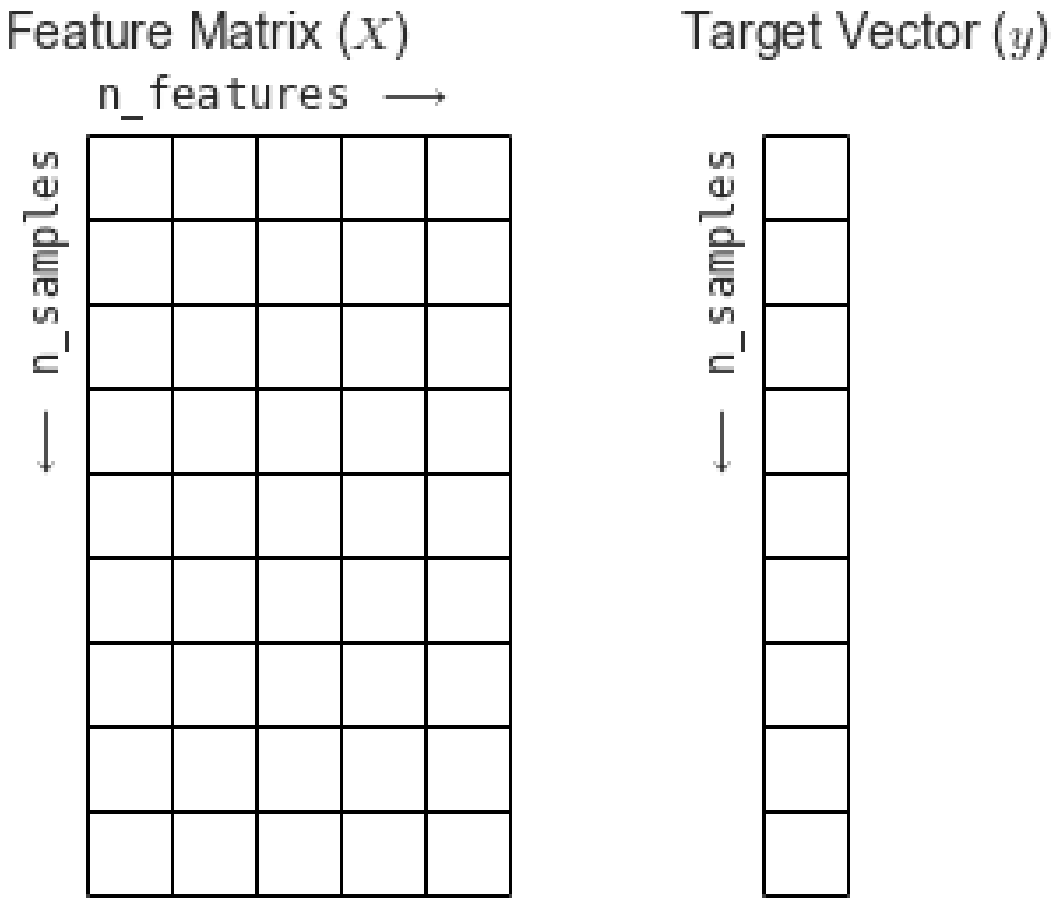
Matriz de características ("features") y vector objetivo ("target")

- Scikit-Learn distingue, al menos en tareas predictivas como clasificación y regresión, esos dos tipos de atributos.
- Por un lado, las características ("features") corresponden a los atributos que describen las características o propiedades de las muestras.
- Por el otro, existe un atributo objetivo ("target") que corresponde a la clase que se busca predecir a partir de las características.
- Así, el `DataFrame` con todos los atributos del conjunto de datos original, debe ser separado en:

Matriz de características ("features") y vector objetivo ("target")

- Scikit-Learn distingue, al menos en tareas predictivas como clasificación y regresión, esos dos tipos de atributos.
- Por un lado, las características ("features") corresponden a los atributos que describen las características o propiedades de las muestras.
- Por el otro, existe un atributo objetivo ("target") que corresponde a la clase que se busca predecir a partir de las características.
- Así, el `DataFrame` con todos los atributos del conjunto de datos original, debe ser separado en:
  1. una matriz de características de tamaño `n_samples X n_features`
  2. un vector objetivo (columna) de tamaño `n_samples`

Disposición de los datos en Scikit-Learn



Matriz de características

- La información de esta tabla puede ser pensada como un arreglo bi-dimensional o matriz cuyo “shape” es `[n_samples, n_features]`.

Matriz de características

- La información de esta tabla puede ser pensada como un arreglo bi-dimensional o matriz cuyo "shape" es `[n_samples, n_features]`.
- Por convención, suele ser almacenada en una variable de nombre `X`

Matriz de características

- La información de esta tabla puede ser pensada como un arreglo bi-dimensional o matriz cuyo "shape" es `[n_samples, n_features]`.
- Por convención, suele ser almacenada en una variable de nombre `X`
- Es a menudo contenida en alguna de las siguientes estructuras:
  1. un arreglo NumPy
  2. un `DataFrame` de Pandas
  3. matrices ralas SciPy

Matriz de características

- Las muestras (filas) de esta matriz siempre refieren a los objetos individuales descriptos por el conjunto de datos: una flor, una persona, un documento, una imágen, un archivo de sonido, un video, un objeto astronómico, etc

Matriz de características

- Las muestras (filas) de esta matriz siempre refieren a los objetos individuales descriptos por el conjunto de datos: una flor, una persona, un documento, una imágen, un archivo de sonido, un video, un objeto astronómico, etc
- Las características (columnas) siempre refieren a las distintas observaciones que describen cada muestra en una manera cuantitativa. Toman generalmente valores reales, pero pueden ser booleanos u otros valores discretos (nominales).

Matriz de características

- Las muestras (filas) de esta matriz siempre refieren a los objetos individuales descriptos por el conjunto de datos: una flor, una persona, un documento, una imágen, un archivo de sonido, un video, un objeto astronómico, etc
- Las características (columnas) siempre refieren a las distintas observaciones que describen cada muestra en una manera cuantitativa. Toman generalmente valores reales, pero pueden ser booleanos u otros valores discretos (nominales).
- En el conjunto de datos Iris, tendremos una matriz de características de "shape" `[150, 4]` correspondiente a 150 lirios descriptos por 4 característica (features) reales.

Vector objetivo

- Además de la matriz de características `X`, las tareas predictivas en Scikit-Learn requieren un vector de etiquetas u objetivo que denotaremos `y`.

Vector objetivo

- Además de la matriz de características `X`, las tareas predictivas en Scikit-Learn requieren un vector de etiquetas u objetivo que denotaremos `y`.
- El vector objetivo `y` es usualmente uni-dimensional con longitud `n_samples`.

Vector objetivo

- Además de la matriz de características `X`, las tareas predictivas en Scikit-Learn requieren un vector de etiquetas u objetivo que denotaremos `y`.
- El vector objetivo `y` es usualmente uni-dimensional con longitud `n_samples`.
- Es a menudo contenido en un arreglo NumPy o una `Series` de Pandas.

Vector objetivo

- Además de la matriz de características `X`, las tareas predictivas en Scikit-Learn requieren un vector de etiquetas u objetivo que denotaremos `y`.
- El vector objetivo `y` es usualmente uni-dimensional con longitud `n_samples`.
- Es a menudo contenido en un arreglo NumPy o una `Series` de Pandas.
- El vector objetivo contiene valores numéricos continuos en problemas de regresión y etiquetas/clases discretas en problemas de clasificación.

Vector objetivo

- Además de la matriz de características `X`, las tareas predictivas en Scikit-Learn requieren un vector de etiquetas u objetivo que denotaremos `y`.
- El vector objetivo `y` es usualmente uni-dimensional con longitud `n_samples`.
- Es a menudo contenido en un arreglo NumPy o una `Series` de Pandas.
- El vector objetivo contiene valores numéricos continuos en problemas de regresión y etiquetas/clases discretas en problemas de clasificación.
- En el conjunto de datos Iris, será un arreglo de tamaño 150, cuyos valores serán alguna de las 3 especies de lirio.

Extrayendo features y etiquetas de un `DataFrame`

- La matriz de características y el vector objetivo de etiquetas puede ser extraído de la colección Iris, con las operaciones usuales de Pandas para `DataFrame`:

Extrayendo features y etiquetas de un `DataFrame`

- La matriz de características y el vector objetivo de etiquetas puede ser extraído de la colección Iris, con las operaciones usuales de Pandas para `DataFrame`:

```
In [3]: X_iris = iris.drop('species', axis=1)
        X_iris.shape

Out[3]: (150,4)

In [4]: y_iris = iris['species']
        y_iris.shape

Out[4]: (150,)
```

Interpretando tablas como funciones

- El aprendizaje supervisado, puede ser visualizado como la tarea de aproximar una función f (desconocida):

$$f : X \rightarrow Y$$

Interpretando tablas como funciones

- El aprendizaje supervisado, puede ser visualizado como la tarea de aproximar una función f (desconocida):

$$f : X \rightarrow Y$$

- usando un conjunto de entrenamiento E, tal que cada instancia de entrenamiento $e \in E$ es un ejemplo $e = \langle x, f(x) \rangle$ del valor del rango $y = f(x)$, $y \in Y$ que f asigna a un valor del dominio x, $x \in X$.

Interpretando tablas como funciones

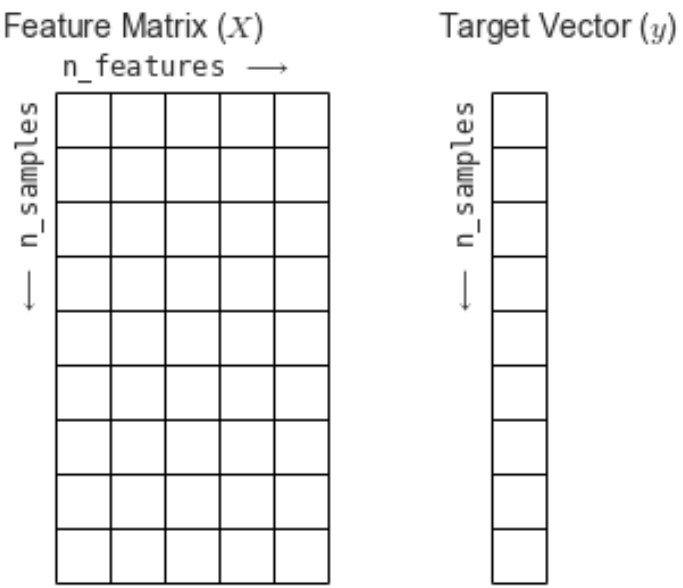
- El aprendizaje supervisado, puede ser visualizado como la tarea de aproximar una función f (desconocida):

$$f : X \rightarrow Y$$

- usando un conjunto de entrenamiento E, tal que cada instancia de entrenamiento $e \in E$ es un ejemplo $e = \langle x, f(x) \rangle$ del valor del rango $y = f(x)$, $y \in Y$ que f asigna a un valor del dominio x, $x \in X$.
- Cuando el dominio Y es numérico, el problema es de regresión, mientras que si Y es un conjunto discreto (representando etiquetas/clases), el problema es de clasificación.

Interpretando tablas como funciones (II)

- En el contexto de Scikit-learn, la tabla que contiene un conjunto de datos como Iris, representa estos ejemplos de la función f desconocida
- Cada fila de la matriz de características X representa un elemento de entrada $\vec{x}, \vec{x} \in X$ para la función f
- Cada elemento del vector objetivo y, representa el valor $y = f(\vec{x})$ que f asocia a una entrada particular $\vec{x}$



Entrenamiento de un clasificador

- Antes vimos que el aprendizaje supervisado, consiste en aproximar una función f (desconocida): $f : X \rightarrow Y$, a partir de un conjunto de entrenamiento E.

Entrenamiento de un clasificador

- Antes vimos que el aprendizaje supervisado, consiste en aproximar una función f (desconocida): $f : X \rightarrow Y$, a partir de un conjunto de entrenamiento E.
- Pero, ...

Entrenamiento de un clasificador

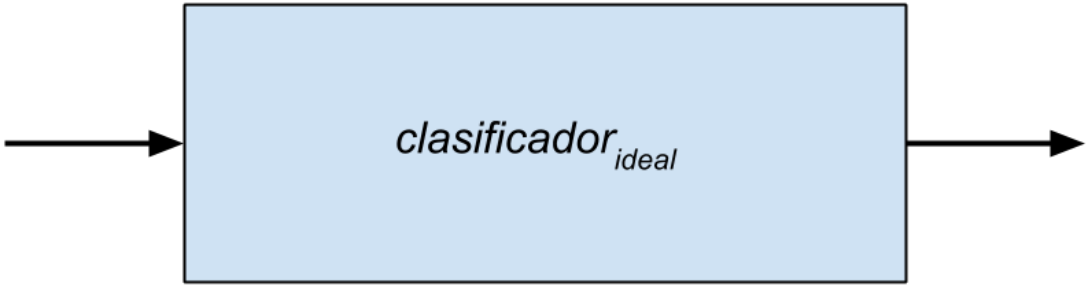
- Antes vimos que el aprendizaje supervisado, consiste en aproximar una función f (desconocida): $f : X \rightarrow Y$, a partir de un conjunto de entrenamiento E.
- Pero, ... ¿que siginifica aproximar?

Entrenamiento de un clasificador

- Antes vimos que el aprendizaje supervisado, consiste en aproximar una función f (desconocida): $f : X \rightarrow Y$, a partir de un conjunto de entrenamiento E.
- Pero, ... ¿que siginifica aproximar?
- Intuitivamente, es encontrar una hipótesis (o modelo) h, $h : X \rightarrow Y$, tal que $h(x) = f(x)$ no sólo para los elementos del conjunto de entrenamiento E, sino para todo elemento del conjunto X
- Los distintos métodos de aprendizaje, permiten obtener distintos tipos de hipótesis (modelos) para aproximar f

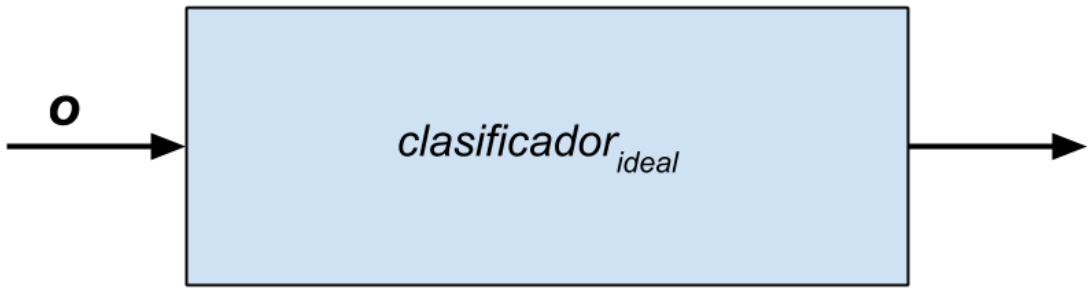
Aprendizaje automático (supervisado)

Idea intuitiva: intentar reproducir un proceso de clasificación correcto/ideal ($clasificador_{ideal}$),



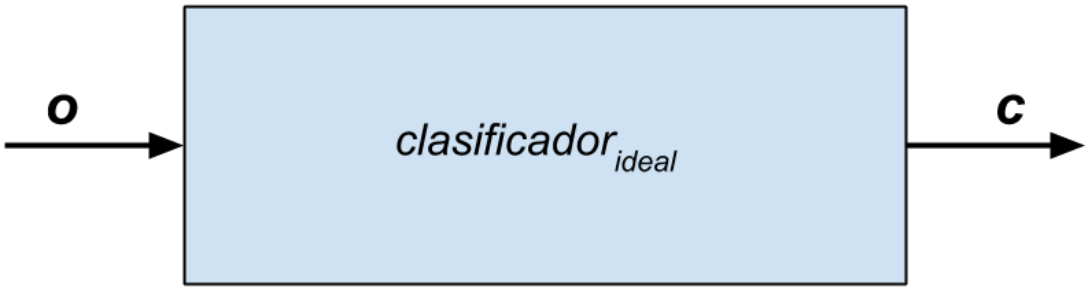
Aprendizaje automático (supervisado)

Idea intuitiva: ... el cual genera para cada entrada (objeto a clasificar) *o*



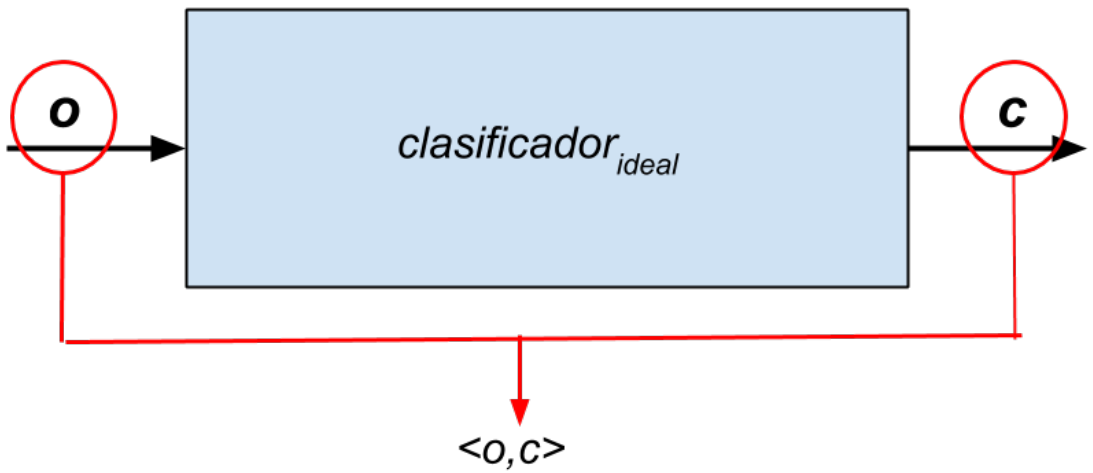
Aprendizaje automático (supervisado)

Idea intuitiva: ... el cual genera para cada entrada (objeto a clasificar) *o*, una salida *c* (la clase de *o*)



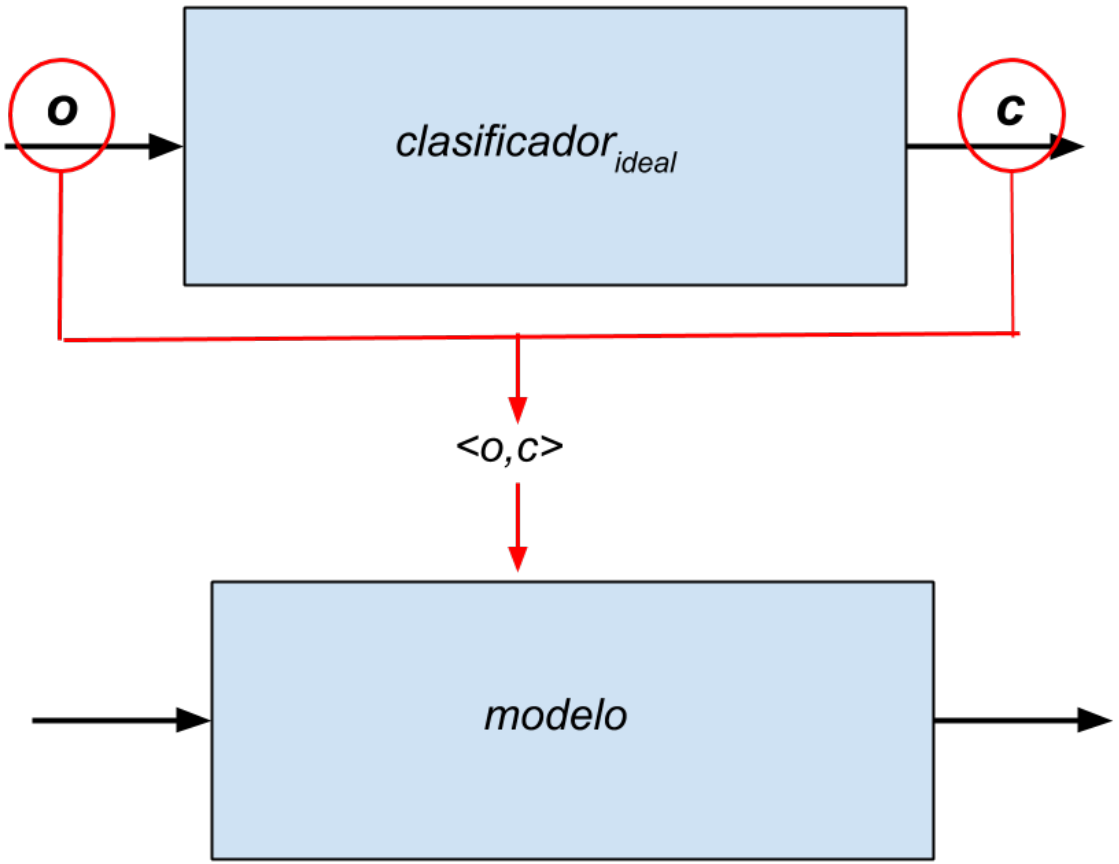
Aprendizaje automático (supervisado)

Idea intuitiva: ... usando ejemplos $\langle o, c \rangle$ del comportamiento de $clasificador_{ideal}$,



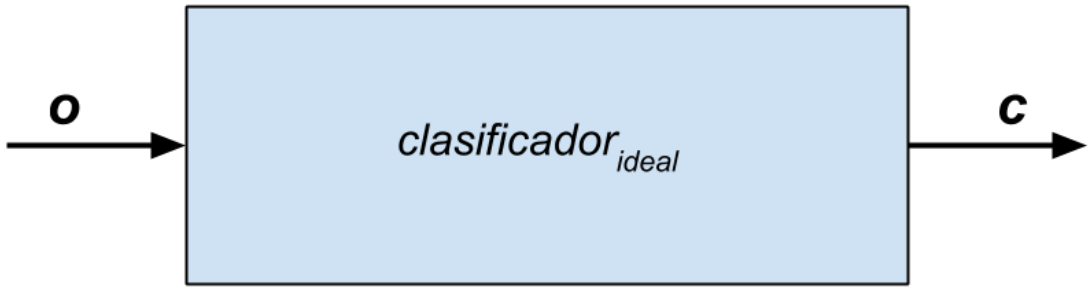
Aprendizaje automático (supervisado)

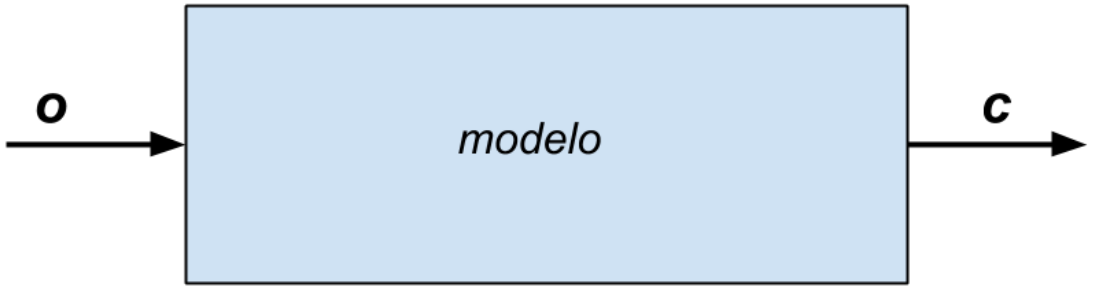
Idea intuitiva: ... usando ejemplos $\langle o, c \rangle$ del comportamiento de $clasificador_{ideal}$, para entrenar otro clasificador (*modelo*)



Aprendizaje automático (supervisado)

Idea intuitiva: ... cuyos comportamientos sean tan parecidos como sea posible.





Aprendizaje automático (supervisado)

Puntos claves:

- las salidas (clasificaciones) de $clasificador_{ideal}$ y *modelo* deberían coincidir respecto a los ejemplos de entrenamiento pero (y más importante), ...

Aprendizaje automático (supervisado)

Puntos claves:

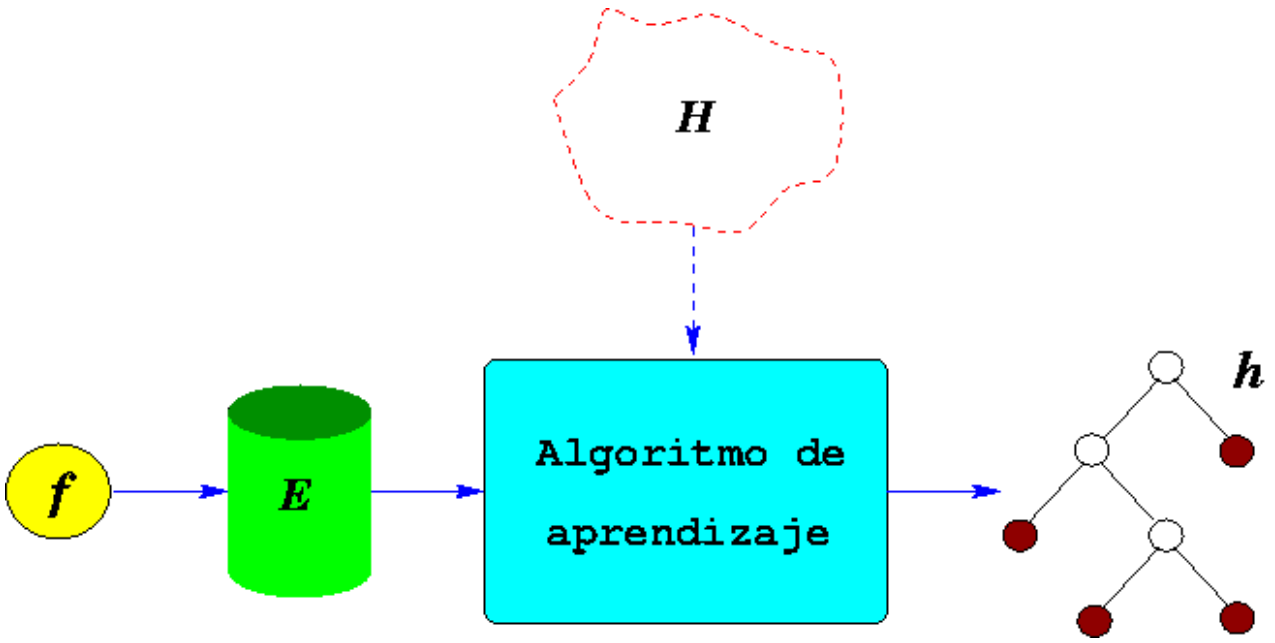
- las salidas (clasificaciones) de $clasificador_{ideal}$ y *modelo* deberían coincidir respecto a los ejemplos de entrenamiento pero (y más importante), ...
- deberían coincidir sobre casos (objetos) no presentes en el conjunto de entrenamiento (generalizar)
- Este proceso, en matemática, se conoce como aproximación de una función

Aprendizaje de un clasificador (idea intuitiva)

Idea: aproximar la función ideal de clasificación:

$$f : \mathcal{O} \mapsto \mathcal{C}$$

con un conjunto de entrenamiento E, de ejemplos $\langle o, c \rangle$, tal que $o \in \mathcal{O}$ es un objeto, y $c \in \mathcal{C}$ es la categoría que f asigna a o.



Aprendizaje de un clasificador (+ formal)

Dados

- Una función de clasificación o *clasificación objetivo* desconocida:

$$f : \mathcal{O} \rightarrow \mathcal{C}$$

Aprendizaje de un clasificador (+ formal)

Dados

- Una función de clasificación o *clasificación objetivo* desconocida:

$$f : \mathcal{O} \rightarrow \mathcal{C}$$

- Un conjunto de entrenamiento E, tal que cada ejemplo es una instancia rotulada con una de las posibles clases, $\langle o, f(o) \rangle$ donde $o \in \mathcal{O}$ y $f(o) \in \mathcal{C}$

Aprendizaje de un clasificador (+ formal)

Dados

- Una función de clasificación o *clasificación objetivo* desconocida:

$$f : \mathcal{O} \rightarrow \mathcal{C}$$

- Un conjunto de entrenamiento E, tal que cada ejemplo es una instancia rotulada con una de las posibles clases, $\langle o, f(o) \rangle$ donde $o \in \mathcal{O}$ y $f(o) \in \mathcal{C}$

Tarea: estimar f, es decir, encontrar una función:

$$h : \mathcal{O} \mapsto \mathcal{C}$$

denominada *hipótesis clasificadora o clasificador*, tal que $h(o) = f(o)$ para todo $o \in \mathcal{O}$.

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados
- Ajustar (entrenar/aprender) el modelo a los datos llamando al método `fit()` de la instancia del modelo

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados
- Ajustar (entrenar/aprender) el modelo a los datos llamando al método `fit()` de la instancia del modelo
- Aplicar el modelo a nuevos datos:

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados
- Ajustar (entrenar/aprender) el modelo a los datos llamando al método `fit()` de la instancia del modelo
- Aplicar el modelo a nuevos datos:
  1. En aprendizaje supervisado: a menudo predecimos etiquetas para datos desconocidos con el método `predict()`

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados
- Ajustar (entrenar/aprender) el modelo a los datos llamando al método `fit()` de la instancia del modelo
- Aplicar el modelo a nuevos datos:
  1. En aprendizaje supervisado: a menudo predecimos etiquetas para datos desconocidos con el método `predict()`
  2. En aprendizaje no supervisado: a menudo transformamos o deducimos propiedades de los datos con los métodos `transform()` o `predict()`

Pasos del aprendizaje y aplicación de modelos en Scikit-Learn

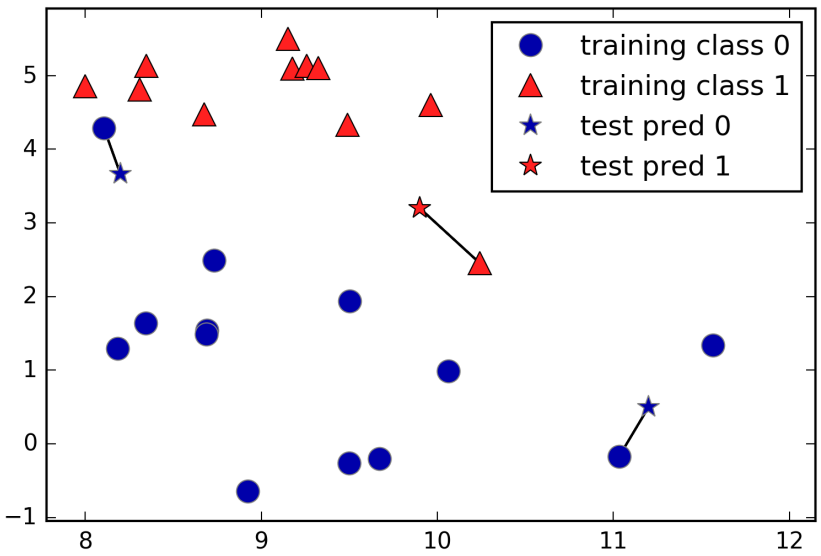
- Cargar o generar los datos a analizar
- Ubicar los datos en la matriz de features y el vector target
- Elegir una clase de modelo importando la clase de estimador apropiada de Scikit-Learn
- Elegir hiperparámetros del modelo instanciando esta clase con los valores deseados
- Ajustar (entrenar/aprender) el modelo a los datos llamando al método `fit()` de la instancia del modelo
- Aplicar el modelo a nuevos datos:
  1. En aprendizaje supervisado: a menudo predecimos etiquetas para datos desconocidos con el método `predict()`
  2. En aprendizaje no supervisado: a menudo transformamos o deducimos propiedades de los datos con los métodos `transform()` o `predict()`
- Evaluar y/o graficar los resultados

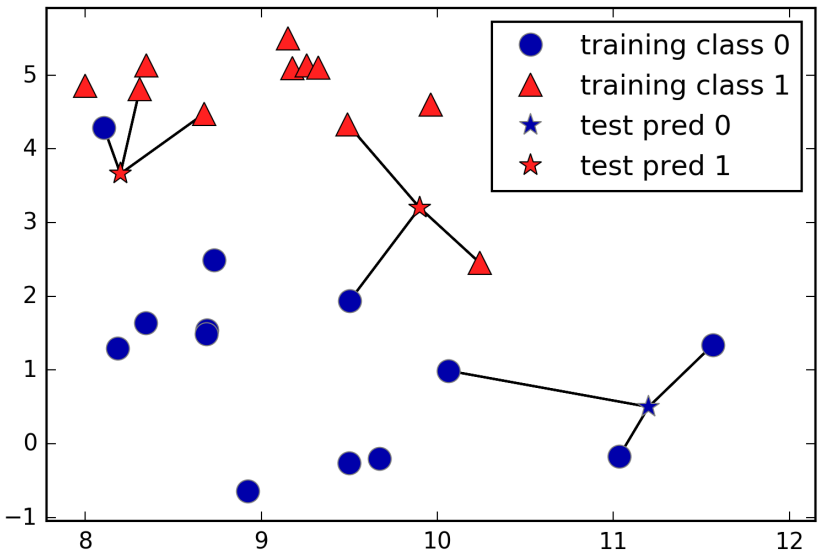
Ejemplo de aprendizaje supervisado: clasificación de Iris

- Usaremos el método más básico de aprendizaje *k*-Nearest Neighbors (*k*-NN)

Ejemplo de aprendizaje supervisado: clasificación de Iris

- Usaremos el método más básico de aprendizaje k-Nearest Neighbors (k-NN)
- Ejemplos para $k = 1$ y $k = 3$





Elección de la clase de modelo

- En Scikit-Learn, cada clase de modelo es representada por una clase de Python.

Elección de la clase de modelo

- En Scikit-Learn, cada clase de modelo es representada por una clase de Python.
- Ejemplo: para un modelo de clasificación con árboles de decisión se puede usar la clase `DecisionTreeClassifier`, para obtener un modelo de regresión lineal simple usamos la clase `LinearRegression`, etc

Elección de la clase de modelo

- En Scikit-Learn, cada clase de modelo es representada por una clase de Python.
- Ejemplo: para un modelo de clasificación con árboles de decisión se puede usar la clase `DecisionTreeClassifier`, para obtener un modelo de regresión lineal simple usamos la clase `LinearRegression`, etc
- En nuestro caso, tendremos que importar la clase `KNeighborsClassifier`

```
In [5]: from sklearn.neighbors import KNeighborsClassifier
```

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.
- Una vez que hemos decidido nuestra clase de modelo, todavía hay algunas opciones disponibles para nosotros.

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.
- Una vez que hemos decidido nuestra clase de modelo, todavía hay algunas opciones disponibles para nosotros.
- Estas elecciones, determinan completamente el modelo que se generará y se especifican mediante hiperparámetros.

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.
- Una vez que hemos decidido nuestra clase de modelo, todavía hay algunas opciones disponibles para nosotros.
- Estas elecciones, determinan completamente el modelo que se generará y se especifican mediante hiperparámetros.
- Cada clase de modelo tiene distintos hiperparámetros

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.
- Una vez que hemos decidido nuestra clase de modelo, todavía hay algunas opciones disponibles para nosotros.
- Estas elecciones, determinan completamente el modelo que se generará y se especifican mediante hiperparámetros.
- Cada clase de modelo tiene distintos hiperparámetros
- En el caso de k-NN, el más importante es el que sirve para especificar el número de vecinos a usar. Si quiciéramos clasificar con 15 vecinos, sería:

Elección de los hiperparámetros del modelo

- Un punto importante es que una clase de modelo no es lo mismo que una instancia del modelo.
- Una vez que hemos decidido nuestra clase de modelo, todavía hay algunas opciones disponibles para nosotros.
- Estas elecciones, determinan completamente el modelo que se generará y se especifican mediante hiperparámetros.
- Cada clase de modelo tiene distintos hiperparámetros
- En el caso de *k*-NN, el más importante es el que sirve para especificar el número de vecinos a usar. Si quiciéramos clasificar con 15 vecinos, sería:

```
In [6]: model = KNeighborsClassifier(n_neighbors=15)
```

Ubicar los datos en la matriz de características y el vector objetivo

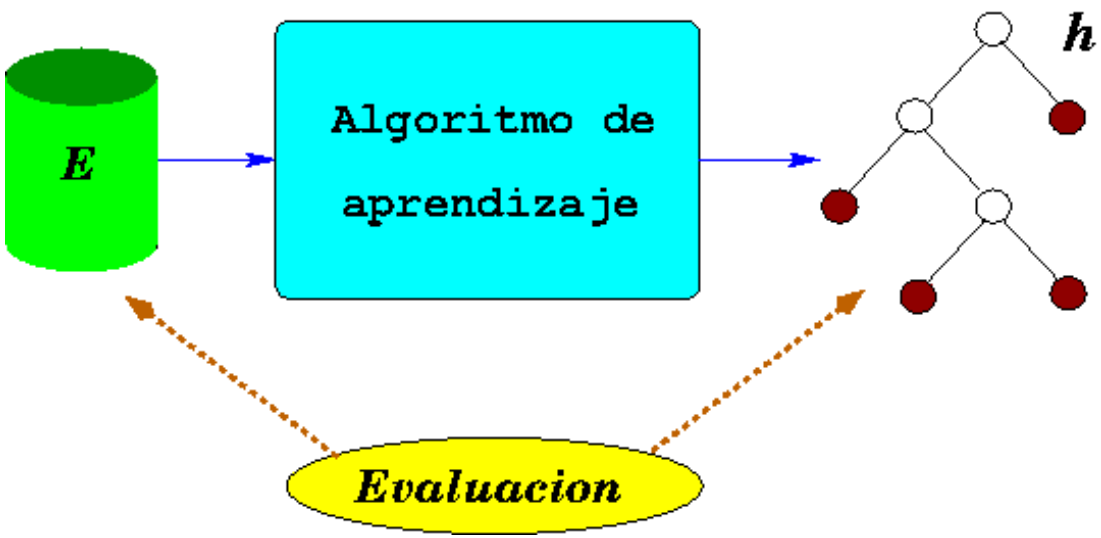
- Este paso, ya fue realizado previamente, quedando en `X_iris` la matriz de características y en `y_iris` el vector objetivo

Ubicar los datos en la matriz de características y el vector objetivo

- Este paso, ya fue realizado previamente, quedando en `X_iris` la matriz de características y en `y_iris` el vector objetivo
- En este caso, podemos entrenar el modelo con todos estos datos y evaluarlo sobre otro conjunto distinto.

Ubicar los datos en la matriz de características y el vector objetivo

- Este paso, ya fue realizado previamente, quedando en `X_iris` la matriz de características y en `y_iris` el vector objetivo
- En este caso, podemos entrenar el modelo con todos estos datos y evaluarlo sobre otro conjunto distinto.
- Sin embargo, comenzaremos analizando qué sucede cuando entrenamos y evluamos sobre el mismo conjunto de datos, como se muestra en la figura:



Ajustar el modelo a los datos de entrenamiento (aprender)

- En este paso, se ajusta el modelo a nuestros datos de entrenamiento. Es el paso en el que el modelo es efectivamente aprendido.

Ajustar el modelo a los datos de entrenamiento (aprender)

- En este paso, se ajusta el modelo a nuestros datos de entrenamiento. Es el paso en el que el modelo es efectivamente aprendido.
- Para ello, se utiliza el método `fit()` del modelo.

Ajustar el modelo a los datos de entrenamiento (aprender)

- En este paso, se ajusta el modelo a nuestros datos de entrenamiento. Es el paso en el que el modelo es efectivamente aprendido.
- Para ello, se utiliza el método `fit()` del modelo.
- En el caso de *k*-NN, el "aprendizaje" simplemente consiste en "recordar" los ejemplos de entrenamiento como están, almacenándolos de manera conveniente para su fácil acceso durante la clasificación.

Ajustar el modelo a los datos de entrenamiento (aprender)

- En este paso, se ajusta el modelo a nuestros datos de entrenamiento. Es el paso en el que el modelo es efectivamente aprendido.
- Para ello, se utiliza el método `fit()` del modelo.
- En el caso de *k*-NN, el "aprendizaje" simplemente consiste en "recordar" los ejemplos de entrenamiento como están, almacenándolos de manera conveniente para su fácil acceso durante la clasificación.

```
In [7]: model.fit(X_iris, y_iris)

Out[7]: KNeighborsClassifier(algorithm='auto', leaf_size=30,
            metric='minkowski', metric_params=None, n_jobs=1,
            n_neighbors=15, p=2, weights='uniform')
```

Ajustar el modelo a los datos de entrenamiento (aprender)

- En este paso, se ajusta el modelo a nuestros datos de entrenamiento. Es el paso en el que el modelo es efectivamente aprendido.
- Para ello, se utiliza el método `fit()` del modelo.
- En el caso de *k*-NN, el "aprendizaje" simplemente consiste en "recordar" los ejemplos de entrenamiento como están, almacenándolos de manera conveniente para su fácil acceso durante la clasificación.

```
In [7]: model.fit(X_iris, y_iris)

Out[7]: KNeighborsClassifier(algorithm='auto', leaf_size=30,
            metric='minkowski', metric_params=None, n_jobs=1,
            n_neighbors=15, p=2, weights='uniform')
```

Como se puede observar, cuando creamos el modelo en el paso anterior, algunos hiperparámetros (como la distancia entre elementos) fueron elegidos por defecto

Aplicar el modelo a los datos de evaluación

- El modelo aprendido es usualmente evaluado sobre un conjunto de datos distinto al que se usó para su entrenamiento.

Aplicar el modelo a los datos de evaluación

- El modelo aprendido es usualmente evaluado sobre un conjunto de datos distinto al que se usó para su entrenamiento.
- Sin embargo, por cuestiones de simplicidad comenzaremos analizando qué sucede cuando se usan para evaluación los mismos datos de entrenamiento.

Aplicar el modelo a los datos de evaluación

- El modelo aprendido es usualmente evaluado sobre un conjunto de datos distinto al que se usó para su entrenamiento.
- Sin embargo, por cuestiones de simplicidad comenzaremos analizando qué sucede cuando se usan para evaluación los mismos datos de entrenamiento.
- Para ello, usaremos el método `predict()`, que toma como parámetro los datos a evaluar y devuelve un arreglo con las predicciones realizadas.

Aplicar el modelo a los datos de evaluación

- El modelo aprendido es usualmente evaluado sobre un conjunto de datos distinto al que se usó para su entrenamiento.
- Sin embargo, por cuestiones de simplicidad comenzaremos analizando qué sucede cuando se usan para evaluación los mismos datos de entrenamiento.
- Para ello, usaremos el método `predict()`, que toma como parámetro los datos a evaluar y devuelve un arreglo con las predicciones realizadas.
- Este arreglo, tendrá el mismo "shape" que el vector objetivo usado en el entrenamiento.

```
In [8]: y_model = model.predict(X_iris)
        y_model.shape

Out[8]: (150,)
```

Evaluación del modelo aprendido

- Evaluar un modelo significa medir el grado de concordancia entre las predicciones realizadas (con el método `predict`) sobre un conjunto de datos y las etiquetas reales de ese mismo conjunto de datos

Evaluación del modelo aprendido

- Evaluar un modelo significa medir el grado de concordancia entre las predicciones realizadas (con el método `predict`) sobre un conjunto de datos y las etiquetas reales de ese mismo conjunto de datos
- En el contexto de nuestro ejemplo, significa medir en que medida coinciden las predicciones del clasificador *k*-NN aplicado al conjunto de datos `X_iris` (guardadas en el vector `y_model`) y las etiquetas reales de ese mismo conjunto de datos (guardadas en el vector `y_iris`)

Evaluación del modelo aprendido

- Evaluar un modelo significa medir el grado de concordancia entre las predicciones realizadas (con el método `predict`) sobre un conjunto de datos y las etiquetas reales de ese mismo conjunto de datos
- En el contexto de nuestro ejemplo, significa medir en que medida coinciden las predicciones del clasificador *k*-NN aplicado al conjunto de datos `X_iris` (guardadas en el vector `y_model`) y las etiquetas reales de ese mismo conjunto de datos (guardadas en el vector `y_iris`)
- A tal fin, la herramienta `accuracy_score` computa la fracción (proporción) de etiquetas predichas que coinciden con los verdaderos valores (o sea, los aciertos):

```
In [9]: from sklearn.metrics import accuracy_score
        accuracy_score(y_iris, y_model)

Out[9]: 0.9866666666666667
```

Entrenamiento y evaluación sobre datos separados

- En la práctica, los modelos serán aplicados sobre datos distintos a los usados durante el entrenamiento

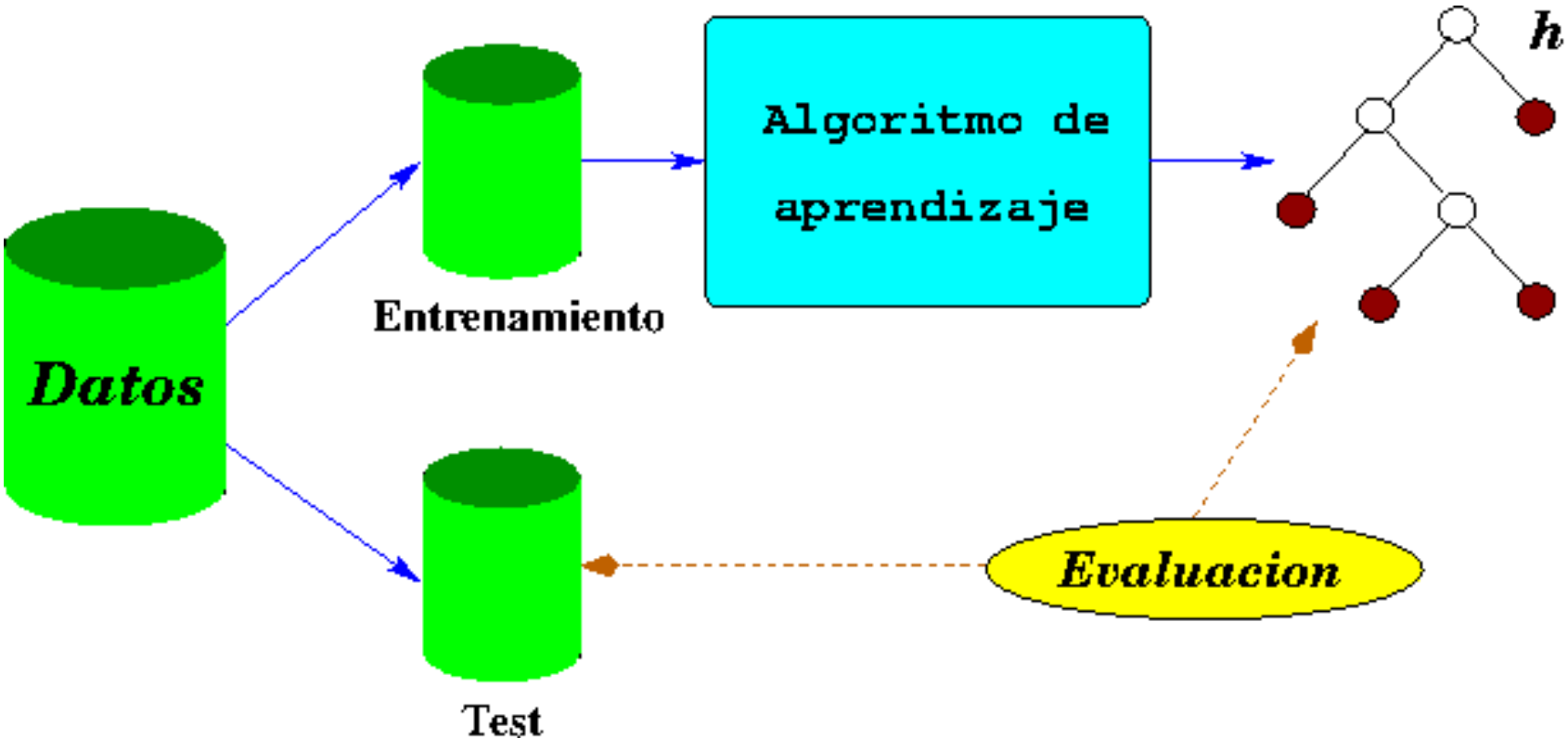
Entrenamiento y evaluación sobre datos separados

- En la práctica, los modelos serán aplicados sobre datos distintos a los usados durante el entrenamiento
- En general, la evaluación del modelo sobre el mismo conjunto de entrenamiento tiende a mostrar un desempeño optimista respecto al que observaremos en la práctica.

Entrenamiento y evaluación sobre datos separados

- En la práctica, los modelos serán aplicados sobre datos distintos a los usados durante el entrenamiento
- En general, la evaluación del modelo sobre el mismo conjunto de entrenamiento tiende a mostrar un desempeño optimista respecto al que observaremos en la práctica.
- Así, lo que en general se usa es tener un conjunto de entrenamiento separado del que se usará para evaluar, como se muestra en la próxima slide.

Entrenamiento y evaluación sobre datos separados



Entrenamiento y evaluación sobre datos separados

Esta separación de los datos en un conjunto de entrenamiento y uno de test, puede ser fácilmente realizada con la función `train_test_split`:

```
In [10]: from sklearn.model_selection import train_test_split
         Xtrain, Xtest, ytrain, ytest = train_test_split(X_iris,
                                                         y_iris,
                                                         random_state=1)
```

Entrenamiento y evaluación sobre datos separados

Esta separación de los datos en un conjunto de entrenamiento y uno de test, puede ser fácilmente realizada con la función `train_test_split`:

```
In [10]: from sklearn.model_selection import train_test_split
         Xtrain, Xtest, ytrain, ytest = train_test_split(X_iris,
                                                         y_iris,
                                                         random_state=1)
```

Entrenando luego sobre este nuevo conjunto de entrenamiento y evaluando sobre un conjunto de prueba diferente.

```
In [11]: model.fit(Xtrain, ytrain)
         y_model = model.predict(Xtest)
         accuracy_score(ytest, y_model)

Out[11]: 0.9736842105263158
```

Comentarios finales

- Todos los ejemplos de esta clase están disponibles en la notebook clase-ap-aut-ejemplos-iris.ipynb

Comentarios finales

- Todos los ejemplos de esta clase están disponibles en la notebook clase-ap-aut-ejemplos-iris.ipynb
- Algunas variantes a los conceptos vistos en esta clase están disponibles en la notebook clase-ap-aut-ejemplos-iris-v2.ipynb

Comentarios finales

- Todos los ejemplos de esta clase están disponibles en la notebook clase-ap-aut-ejemplos-iris.ipynb
- Algunas variantes a los conceptos vistos en esta clase están disponibles en la notebook clase-ap-aut-ejemplos-iris-v2.ipynb
- Como material complementario, en la notebook clase-ap-aut-material-complementario.ipynb se dan 2 ejemplos de tareas de aprendizaje automático (no supervisadas) que difieren de la tarea de clasificación abordada hasta el momento:
  1. Aprendizaje no supervisado: reducción de dimensionalidad mediante análisis de componentes principales (PCA)
  2. Aprendizaje no supervisado: clustering mediante mezcla de modelos Gaussianos(GMM)