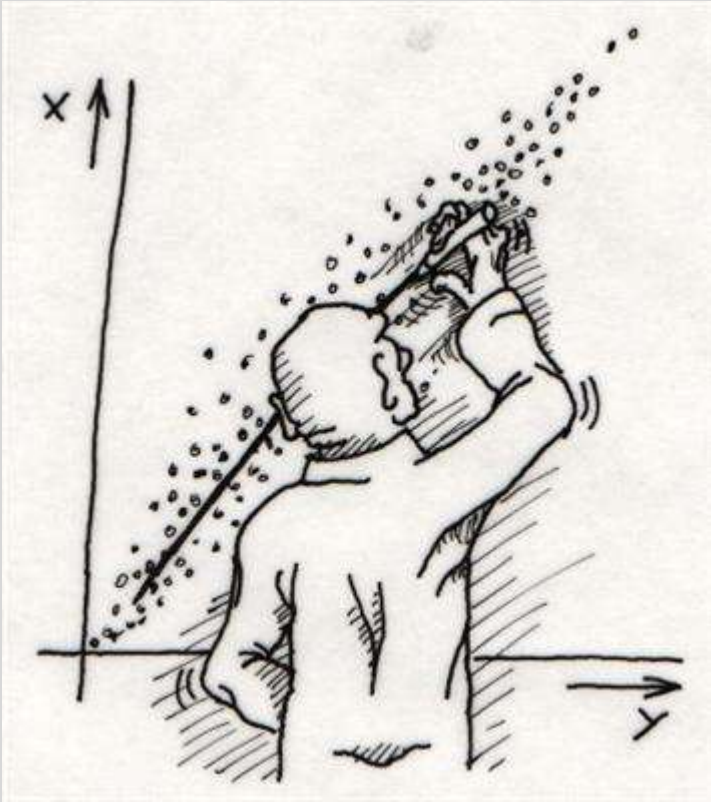


DigitalHouse > Coding School

DATA SCIENCE

MÓDULO 3

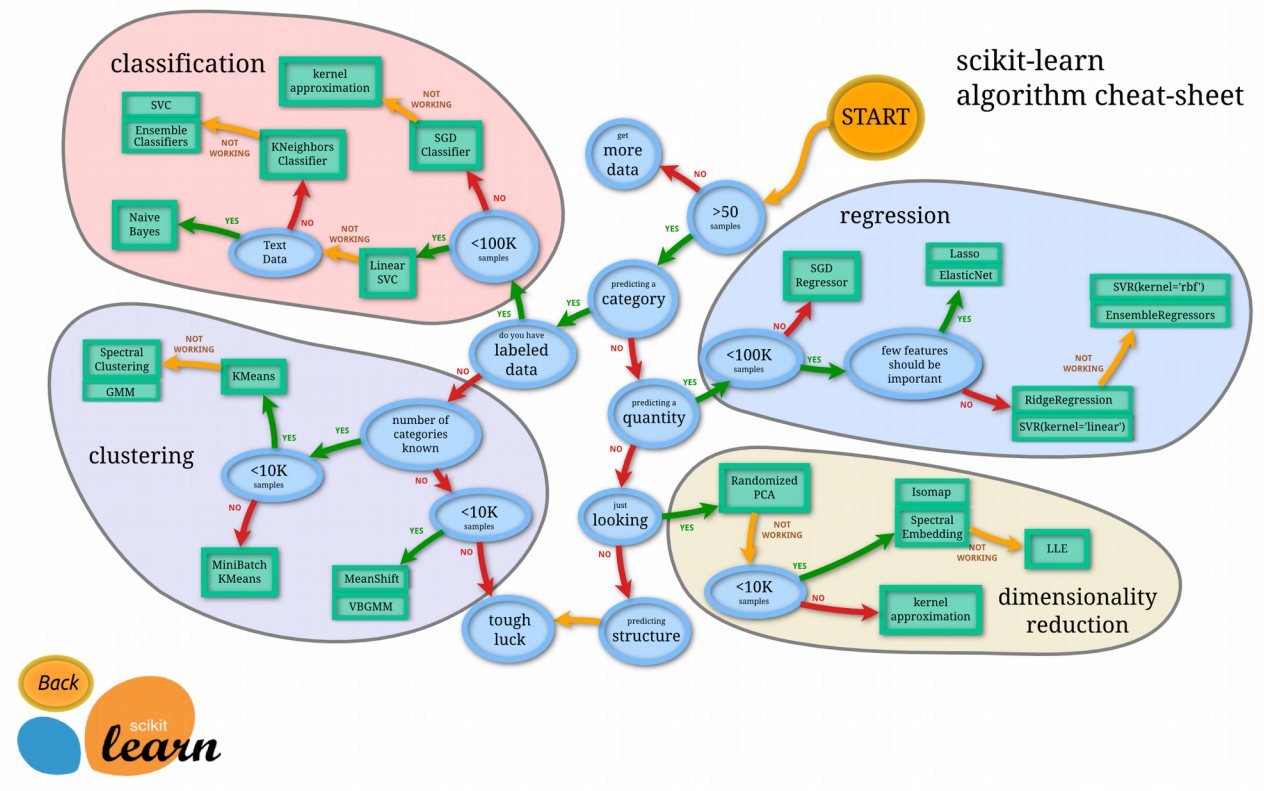
Intro a Scikit Learn

Agosto de 2017

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**.

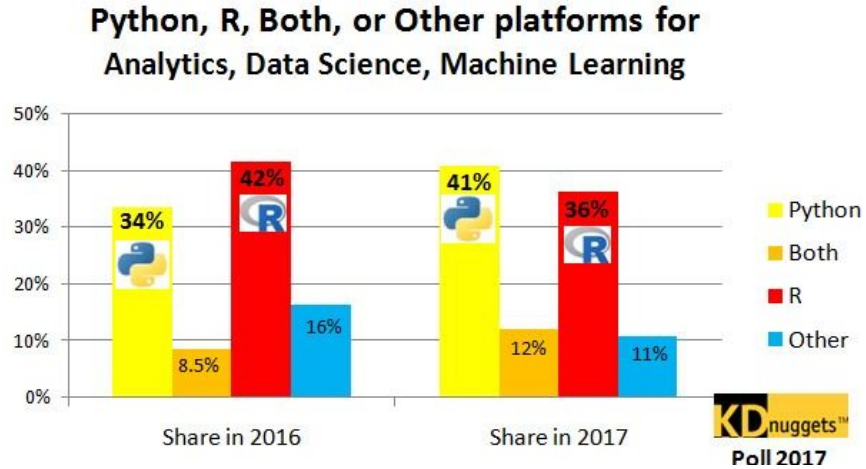
2. Los modelos son elegidos e **implementados por un equipo de expertos**. El grupo de colaboradores de Scikit-learn incluye expertos en machine learning y desarrollo de software.

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.

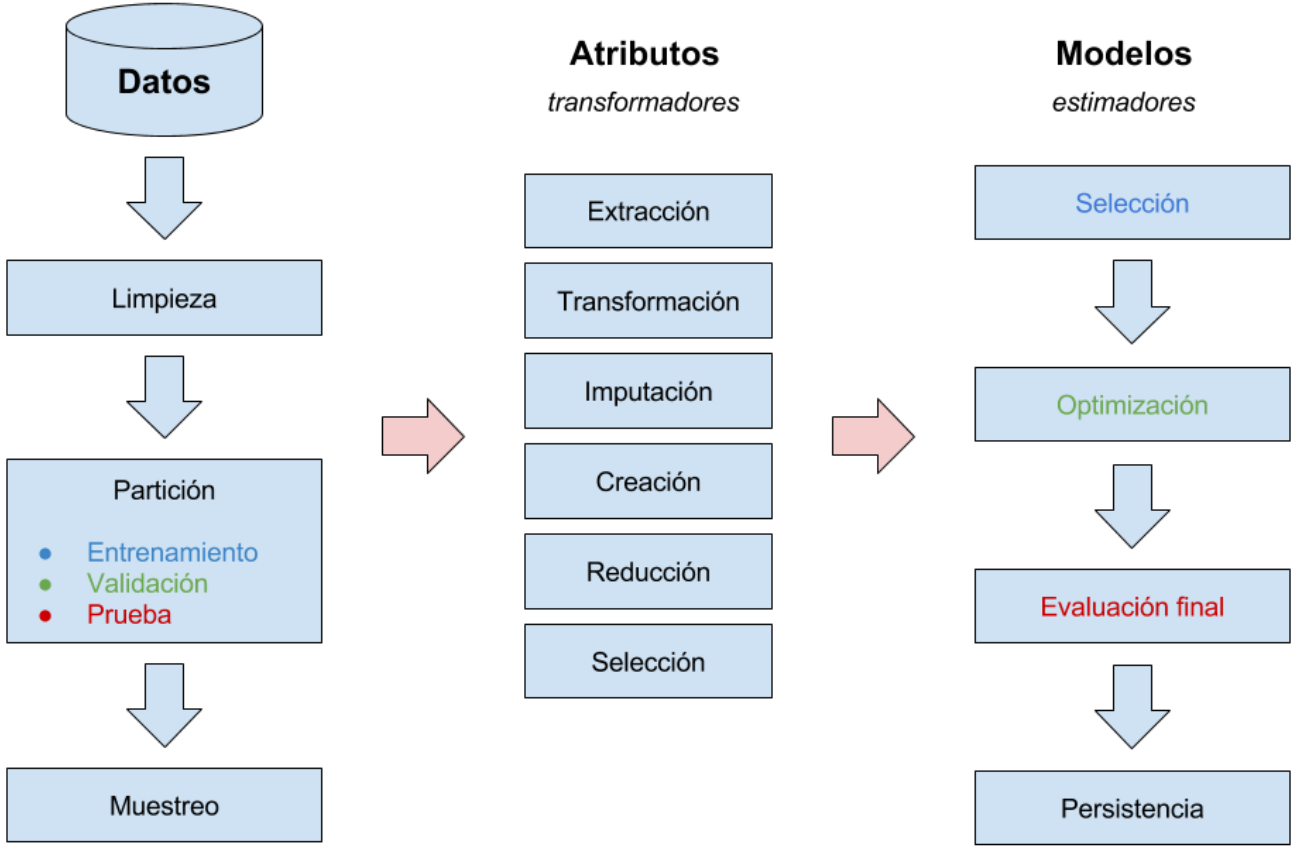
6. **Escalabilidad** Scikit-learn es escalable a la mayoría de los datasets que usamos en este contexto. Muchos problemas pueden encararse usando un único servidor (con mucha memoria), reduciendo la complejidad del desarrollo de sistemas distribuidos.

Documentación Scikit-learn: **Strategies to scale computationally**

Scikit-learn

Flujo de Trabajo:
Representación de los datos

Tres etapas del Flujo de trabajo típico con scikit-learn



La representación de los datos en Scikit-Learn

Como vimos, machine learning se trata de crear modelos a partir de los datos;

Hoy, comenzaremos discutiendo cómo los datos deben representarse para poder ser usados con estas herramientas.

La mejor forma de pensar los datos dentro de Scikit-Learn es en términos de tablas de datos.

Un dataset como una tabla

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris.head()
```

	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

Una tabla básica es una matriz bidimensional de datos, en la cual las filas representan elementos individuales del dataset, y las columnas representan cantidades relacionadas con esos elementos.

Por ejemplo, consideremos el famoso Iris dataset, analizado por Ronald Fisher en 1936.

Podemos descargar este dataset en forma de un DataFrame de Pandas, usando la librería Seaborn:

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris.head()
```

	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

Cada fila de la tabla se refiere a una única flor observada, y el número de filas es el número total de flores en el dataset.

En general, nos referiremos a las filas de la matriz como **samples o muestras**, y al número de filas lo llamaremos **n_samples**, en el código.

Cada columna de la matriz se refiere a una pieza particular de información cuantitativa que describe a cada **muestra.**

En general, nos referiremos a las columnas de la matriz como **features**, y al número de columnas como **n_features**.

Matriz de features

Este este esquema de tabla deja bien en claro que la información puede pensarse como una matriz numérica bi-dimensional, 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 [m_samples, n_features], y frecuentemente se almacena como un **array de Numpy** o un **DataFrame de Pandas**, aunque algunos modelos de Scikit-Learn también aceptan **matrices dispersas de SciPy**.

Las **muestras (filas)** siempre refieren a **objetos individuales** descriptos 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 observaciones que describen cada una de las muestras de manera cuantitativa. Las features son generalmente valores reales, pero pueden haber valores booleanos o enteros en algunos casos.

El Vector target

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 contínuos o etiquetas/clases discretas.

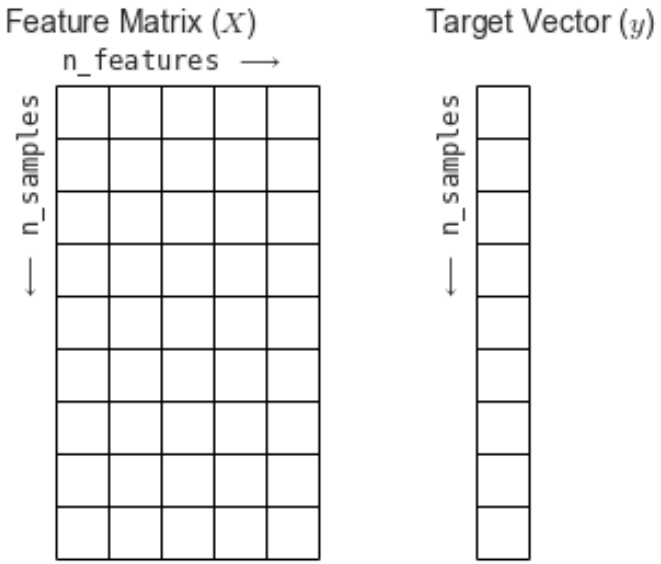
Un punto que vale la pena recordar: **cómo se diferencia el array target de las columnas de la matriz de features?**

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.

Por ejemplo, en el dataset Iris, podemos querer construir un modelo que puede predecir las especies de las flores basado en las otras medidas; en este caso, la columna de especies sería considerada como el array target.

El esquema final de los datos que usaremos con nuestros modelos



Con los datos formateados apropiadamente, podemos continuar con el estudio de la estimator API de Scikit-Learn

Preparamos la matriz de features y el array target

Con los datos formateados apropiadamente, podemos continuar con el estudio de la estimator API de Scikit-Learn

```
X_iris = iris.drop('species', axis=1)
X_iris.shape
```

```
(150, 4)
```

```
y_iris = iris['species']
y_iris.shape
```

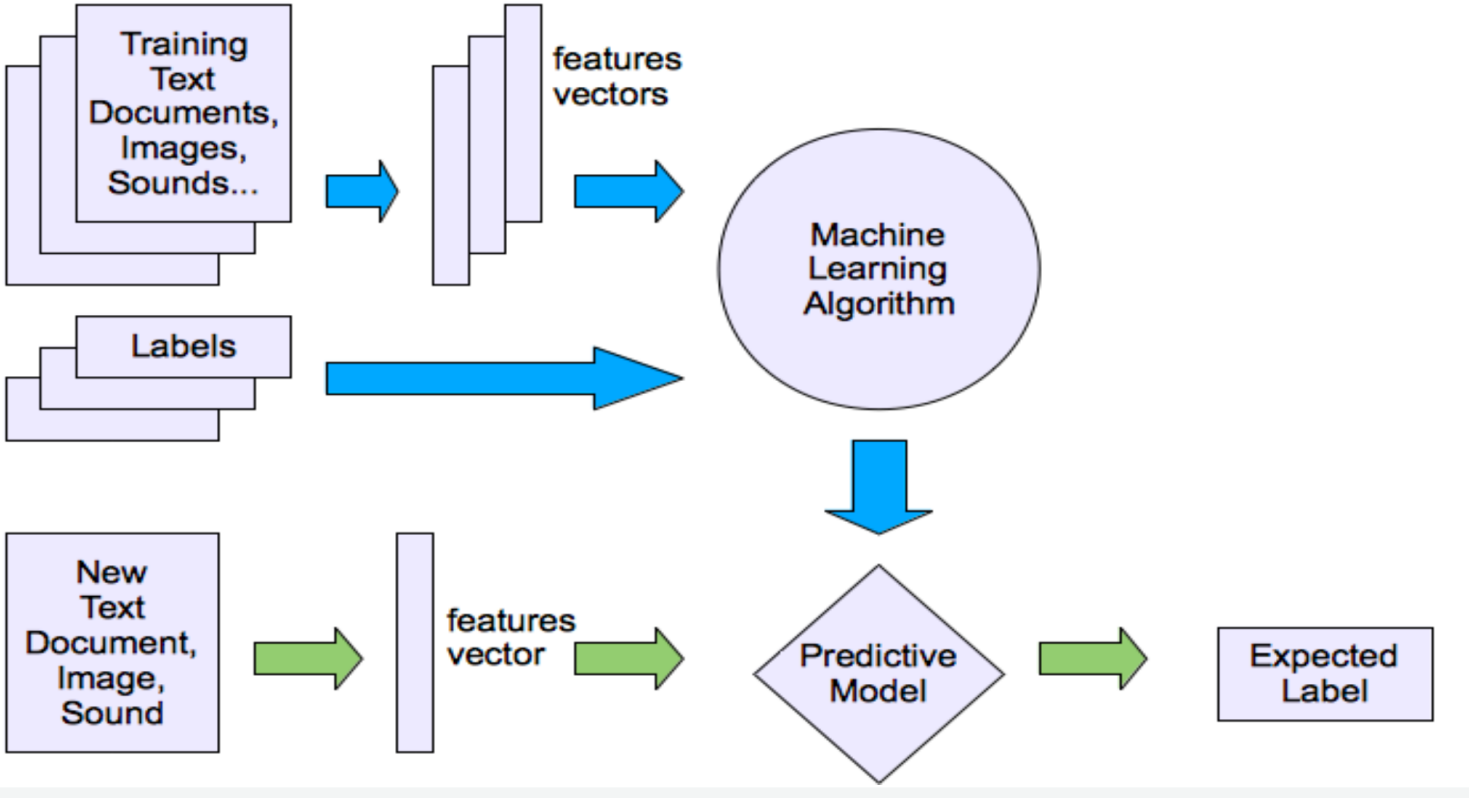
```
(150,)
```

Features y target en Iris

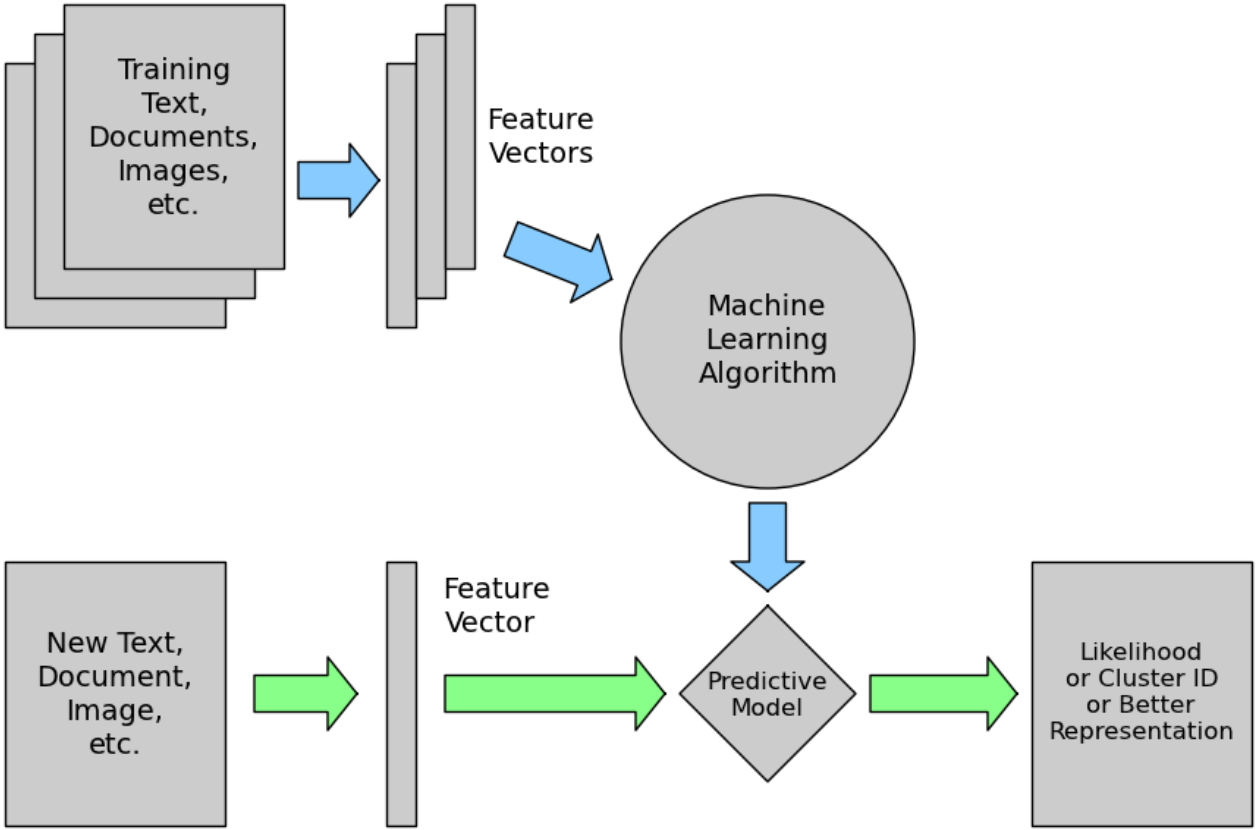
Scikit-learn

Repasemos: features & labels

Supervised learning pipeline



Unsupervised learning model



Scikit-learn

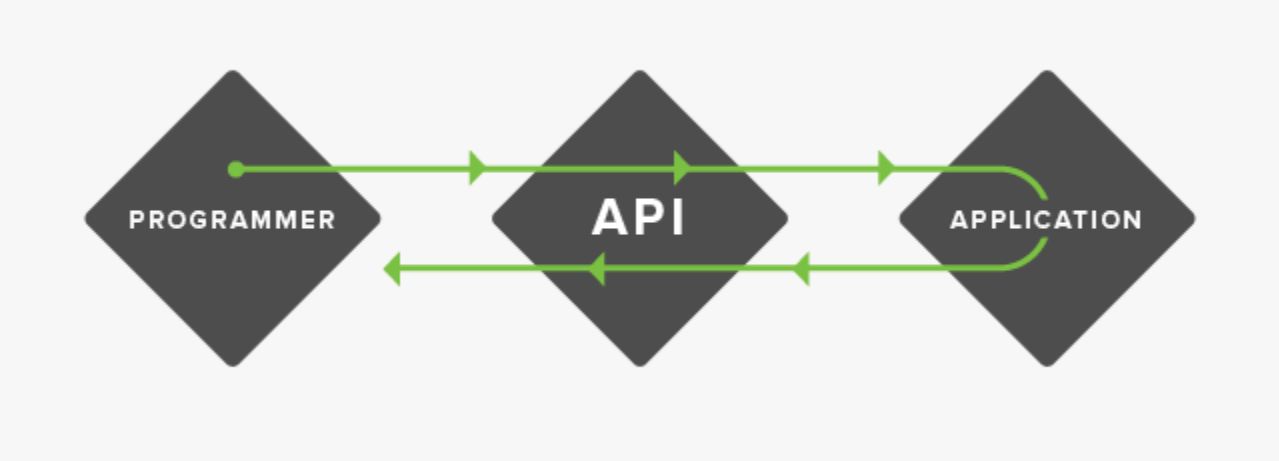
Application Programming Interface (API)

Por qué estudiamos la API de Scikit-learn?

Una vez que se entiende el uso básico de la API de Scikit-Learn para un tipo de modelo,
usar un nuevo modelo o algoritmo es relativamente fácil y directo!

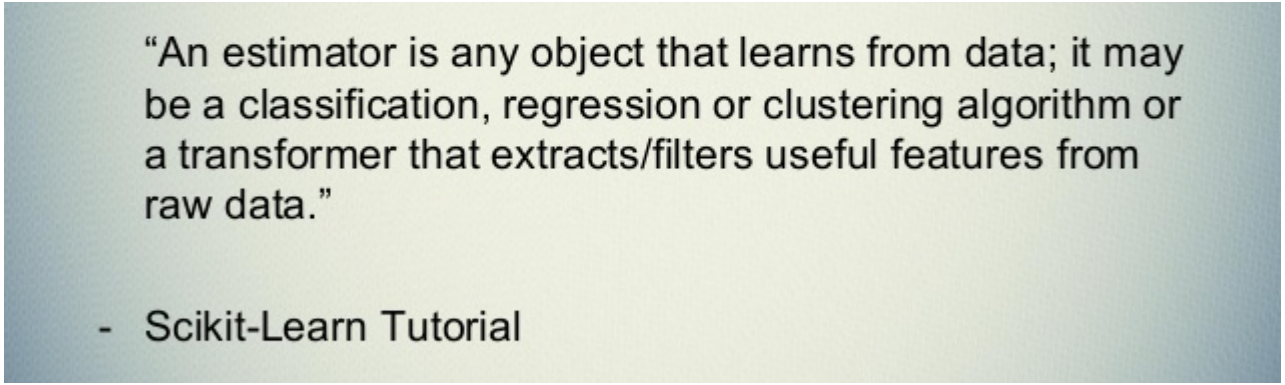
Qué es una API?

Es una **interfaz** implementada por una biblioteca o aplicación que le permite a los desarrolladores u otras aplicaciones comunicarse con ella.



La API de Scikit-Learn: “Estimator API”

Es una interfaz orientada a objetos centrada en el concepto de un estimador.
Qué es un estimador para Scikit-learn?



Comenzaremos usando la ***Estimator* API** con ejemplos sencillos, y finalmente veremos un ejemplo más interesante: **explorar un conjunto de imágenes de dígitos escritos a mano.**

La Estimator API. Notas técnicas

La API se diseñó siguiendo los principios mencionados en el paper original:

- ***Consistencia**:* Todos los objetos comparten una interfaz común formada por un conjunto limitado de métodos, documentados consistentemente.
- ***Inspección**:* Todos los parámetros especificados se exponen como atributos públicos
- ***Jerarquía de clases limitada**:* Sólo los algoritmos se representan con clases de Python; los datasets se representan con estructuras de datos estándar (arrays de Numpy, DataFrames de Pandas, matrices dispersas de SciPy) y los nombres de los parámetros usan strings estándar de Python.
- ***Composición**:* Las tareas de machine learning pueden expresarse como secuencias de algoritmos más básicos, y Scikit-Learn utiliza esta técnica siempre que sea posible.
- ***Valores Default**:* Cuando los modelos requieren parámetros definidos por el usuario, la librería define valores por defecto apropiados.

En la práctica, esto hace que Scikit-Learn sea muy fácil de usar una vez que se comprenden los principios básicos. Todos los algoritmos de machine learning en Scikit-Learn son implementados usando la *Estimator API*, la cual provee una interfaz consistente para un amplio rango de aplicaciones de machine learning.

Pasos para usar la Estimator API

La API se diseñó siguiendo los principios mencionados en el paper original:

1. Elegir una **clase de modelo** importando la clase de estimador apropiado de Scikit-Learn.
2. Seleccionar los **hiperparámetros** del modelo **instanciando la clase** con los valores deseados.
3. **Preparar los datos** en una **matriz de features** y un **array target**, como vimos previamente.
4. **Ajustar el modelo** a los datos invocando el método **fit()** de la instancia del modelo.
5. Aplicar el modelo a **nuevos datos**:
   a. Para **aprendizaje supervisado**, frecuentemente **predecimos labels** para datos nuevos usando el método **predict()**
   b. Para **aprendizaje no supervisado**, frecuentemente **transformamos o inferimos propiedades** de los datos usando los métodos **transform() o predict()**

Práctica Guiada

Avanzaremos siguiendo varios ejemplos simples de aplicación métodos de aprendizaje supervisado y no supervisado