Introducción al aprendizaje automático © EDICIONES ROBLE, S.L.

campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA
campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA
campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA

Indice

campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA

Introducción al aprendizaje automático

I. Introducción

El aprendizaje automático (o aprendizaje máquina, *machine learning*) es una de las áreas de las ciencias de la computación que mayor relevancia ha tenido en los últimos años. La explosión de los sistemas conectados, como los teléfonos móviles, tabletas y el internet de las cosas (*Internet of things*, IoT) está generando una gran cantidad de información que ha de ser procesada y analizada antes de que pierda su valor.

Técnicas de aprendizaje

Aquí es donde las técnicas de aprendizaje automático permiten convertir los datos en bruto en información con la que se pueden realizar tareas como:

- Predicciones: ¿qué clientes se cambiarán de compañía en los próximos meses?
- Identificación de patrones: ¿qué productos compran conjunta y habitualmente mis clientes?
- Identificación de grupos: ¿qué segmentos de clientes tengo?

Aplicaciones

Gracias a estas capacidades, disponer de soluciones basadas en el aprendizaje automático se ha convertido en un elemento diferenciador de gran valor para todo tipo de empresas y organizaciones. La cantidad de aplicaciones en las que pueden ser utilizadas estas técnicas son casi infinitas, entre ellas, algunas de las más populares en la actualidad son:

- Motores de búsqueda.
- Patrones de comportamiento.
- Reconocimiento de objetos en imágenes.
- Detección de fraude e intrusiones.
- Identificación de tendencias.
- Reconocimiento del habla y del lenguaje escrito, etc.

Esta primera unidad del curso comenzará con una exposición del papel del aprendizaje automático en la minería de datos y la relevancia que tiene en los procesos de descubrimiento de conocimiento en las bases de datos o KDD (*Knowlege Discovery in Databases*). También se estudiarán los diferentes tipos de análisis que se pueden realizar mediante las técnicas de minería de datos. Posteriormente, se presentará una clasificación general de los tipos de técnicas existentes dentro del aprendizaje automático, haciendo especial hincapié en la diferenciación entre el aprendizaje supervisado y no supervisado.

Finalmente, se presentarán las principales librerías y herramientas que se utilizarán en el resto del curso: scikit-learn y Theano, además de un primer modelo de ejemplo para comprobar la potencia de estas herramientas.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

Esta unidad se complementa con un notebook Python en el que se incluye todo el código utilizado y algunos ejemplos adicionales a los que se hace referencia en el texto.

II. Objetivos

Los objetivos que los alumnos alcanzarán en esta unidad son:

- Comprender el proceso KDD.
- Entender la minería de datos dentro de KDD.
- Saber lo que es un modelo en minería de datos.
- Saber diferenciar los tipos de modelos supervisados y no supervisados.
- Conocer las librerías scikit-learn y Theano.
- Comprender el algoritmo de k-vecinos y crear un clasificador.

III. El proceso de la minería de datos

La minería de datos es el conjunto de análisis utilizado para la identificación de los patrones desconocidos que se ocultan dentro de los grandes conjuntos de datos. Los análisis pueden ser realizados tanto de forma automática como semiautomática. Para esto es necesario conocer y utilizar diferentes técnicas que proceden de áreas como la gestión de bases de datos, la estadística, la inteligencia artificial y, en particular, el aprendizaje automático.

Actualmente, la minería de datos es una expresión que está de moda, por lo que en muchas ocasiones se utiliza incorrectamente para hacer referencia a todo lo que tenga que ver simplemente con el manejo de grandes volúmenes de datos, incluso en situaciones en las que no se realiza un descubrimiento de patrones en los conjuntos de datos.

La característica que define a la minería de datos es la identificación de patrones. El tipo de patrones que se pueden estudiar son múltiples: la identificación de relaciones entre conjuntos de variables (análisis de regresión), la identificación de grupos semejantes o detección de anomalías (análisis de clúster) o la identificación de hechos que suceden de forma conjunta (reglas de asociación).

Proceso KDD

La minería de datos se enmarca dentro del proceso que se conoce como KDD (*Knowlege Discovery in Databases* o Descubrimiento de Conocimiento en Bases de Datos), en este proceso, a grandes rasgos, se dan los siguientes pasos:

Selección

En el primer paso se han de seleccionar las variables que se utilizarán en el resto del proceso. Las variables se pueden dividir en variables dependientes (aquellas para las que se desea predecir o inferir su valor) y las variables independientes (aquellas que se utilizarán para realizar las predicciones). No en todos los análisis se utilizan variables dependientes, lo que se pretende es identificar nuevos patrones.

Análisis

En este paso se suelen utilizar diferentes técnicas estadísticas para la obtención de información que permitan describir los tipos de los datos que se han seleccionado. Esta información se puede utilizar para la identificación de valores atípicos (outliers), los cuales pueden distorsionar las conclusiones obtenidas, o para la existencia de valores nulos.

Procesado y transformación

En este paso se aplican transformaciones a los conjuntos de datos para que estos pueden ser utilizados en los diferentes modelos empleados en el proceso de modelado. Por ejemplo, en este paso se pueden normalizar los valores o crear nuevas variables a partir de las originales.

Minería de datos

Este es el paso en el que se construyen los modelos con los que se pretende explicar las observaciones. Por ejemplo, estos se pueden aplicar en los datos para identificar similitudes o para la predicción de cierto tipo de comportamiento.

Interpretación y evaluación

En este paso los resultados de los modelos construidos previamente han de ser interpretados y evaluados para comprobar que las predicciones realizadas son coherentes con las observaciones. En caso de que el modelo final no supere esta evaluación, el proceso puede repetirse desde el principio o a partir de cualquiera de los pasos anteriores. En caso de que se supere la evaluación, el modelo puede pasar a producción y los resultados ser utilizados en futuros análisis.

Producción

En esta fase los resultados de los modelos se pueden utilizar para solventar las necesidades de negocio. Aquí es importante evaluar continuamente los resultados para comprobar que las conclusiones obtenidas siguen siendo válidas a medida que pasa el tiempo.

El esquema de este proceso se muestra en la figura 1.1. En esta se puede ver cómo el proceso KDD comienza con la selección de los datos. Una vez seleccionados, se procede al análisis, procesado, modelado (minería de datos) y evaluación. Al terminar la evaluación de los modelos, estos pueden pasar a producción, lo que servirá para resolver los problemas del negocio y el conocimiento adquirido para realizar una nueva selección de datos con la que se inicia un nuevo proceso. KDD es un proceso cíclico en el conocimiento de los datos y los modelos mejoran a media que se itera con ellos.

Figura 1.1. Esquema del proceso KDD. *Fuente:* elaboración propia.

Algunos autores afirman que el proceso KDD incluye también otros procesos, como la limpieza e integración de los datos al comienzo del proceso y la visualización y representación final de los resultados.

Ciclo de vida

Los modelos utilizados en minería de datos tienen un ciclo de vida. El modelo inicial ha de ser creado y, si este es validado durante el proceso de evaluación, es desplegado en producción. Generalmente, el proceso de creación suele ser costoso en cuanto a recursos, siendo necesario disponer de perfiles especializados y altamente cualificados para ello. Por otro lado, la ejecución de los modelos en producción no suele ser costosa, ya que las predicciones las suele realizar de forma automática una máquina. Es corriente que el rendimiento de los modelos decaiga con el tiempo, debido a cambios de hábitos de los clientes u otros cambios que se puedan dar en el entorno. En este punto es necesario crear nuevos modelos para adaptarlos a la nueva realidad, momento en el que los perfiles altamente cualificados vuelven a ser necesarios. Esto es lo que se conoce como ciclo de vida de los modelos.

Fase de creación del modelo

En el proceso de creación de modelos es necesario importar los datos a partir de las fuentes existentes, que en la mayoría de las ocasiones son bases de datos. A partir de estas fuentes se han de extraer las posibles variables que se utilizarán en los modelos, estas variables pueden corresponder a cualquier característica objetiva que pueda ser representada mediante un valor numérico, ordinal o una categoría. En algunos tipos de modelos es necesario disponer, además, de una variable con el valor que se desea predecir posteriormente, para poder crear el modelo, a la que se denomina variable dependiente. Todo esto se ha de introducir en un algoritmo de aprendizaje automático que genera un modelo. El esquema de este proceso se muestra en la figura 1.2.

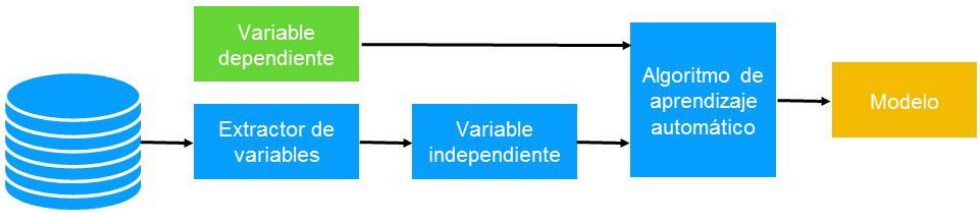

Figura 1.2. Fase de creación de modelo. *Fuente:* elaboración propia.

Fase de producción

Una vez que se ha definido el modelo en la fase de construcción, este puede pasar a producción. En esta nueva fase se han de extraer los datos necesarios para realizar las predicciones y aplicar el mismo proceso que el utilizado en la fase previa. Cambiar la forma en la que se procesan los datos, generalmente, anula los resultados que se obtienen de los modelos construidos, ya que no hay garantía de que se haya utilizado la misma información. Este es el momento en el que los datos se introducen en el modelo para obtener una predicción. El esquema de esta fase se muestra en la figura 1.3.

Figura 1.3. Fase de producción. *Fuente:* elaboración propia.

Tipos de análisis

En minería de datos, se pueden realizar diferentes tipos de análisis, los cuales se pueden clasificar en base a su complejidad. Para estos se pueden utilizar seis categorías. Ordenadas de mayor a menor complejidad, son:

Descriptivos

Descripción cualitativa de las principales características de los datos.

El análisis descriptivo es el más simple de todos, en este no se llega a crear un modelo en sí, pero los resultados obtenidos pueden ser utilizados para mejorar la compresión de los problemas planteados. Suele ser el tipo de análisis que se realiza en las primeras fases para comprender los datos de los que se dispone.

Exploratorios

Análisis de los datos en búsqueda de relaciones desconocidas.

Los análisis exploratorios son un paso más allá de los descriptivos, en estos ya se buscan relaciones entre los datos, pero estas no tienen por qué ser las respuestas directas a unos problemas de negocio. Mediante estos análisis se puede llegar a comprender que dos variables están relacionadas, sin obtener una fórmula que describa la forma exacta de esta relación.

Inferenciales

Evaluación de la validez de teorías en muestras de los datos.

En los análisis inferenciales se evalúan teorías. Gracias a ellos se pueden comprobar las relaciones que existen entre las diferentes variables de las que se dispone en los conjuntos de datos.

Predictivos

Análisis de los datos presentes para obtener predicciones para ser utilizadas en la predicción de eventos futuros.

Los análisis predictivos son aquellos en los que los modelos resultantes se pueden utilizar para realizar predicciones. Mediante estos análisis es posible predecir cómo se comportará un cliente, el mercado o un sistema ante una situación dada y poder anticipar la respuesta más adecuada en función de las necesidades.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

Causales

Estudia el efecto en una variable al cambiar otra.

Los análisis causales van más allá de las predicciones, permitiendo relacionar las causas con los efectos y el grado en el que se influyen mutuamente. Este es un nivel de conocimiento mayor, ya que no solamente se puede predecir el comportamiento de un cliente, el mercado o un sistema, sino que se puede identificar las causas que llevan al mismo y actuar sobre ellas.

Mecanicistas

Comprensión de los cambios que se producen en las variables y el efecto que genera en otras variables.

Finalmente, el tipo de análisis más complejo son los mecanicistas, en los que se puede llegar a comprender los cambios que producen unas variables en el resto. Este es el análisis menos utilizado, ya que para realizarlo es necesaria una compresión profunda de los procesos a modelar, lo que no siempre es posible.

Como se ha visto, la creación de modelos es la clave del proceso de minería de datos, para lo que se pueden utilizar las herramientas de las que provee el aprendizaje automático. Al hablar de aprendizaje automático (o aprendizaje máquina, *machine learning*) se hace referencia a la rama de la inteligencia artificial en la que se agrupan las diferentes técnicas que permiten a las máquinas aprender patrones. Este aprendizaje de patrones es lo que permite responder a las preguntas que se plantean en la minería de datos.

En la siguiente sección se realizará una descripción de los principales problemas que se pueden resolver mediante las técnicas que provee el aprendizaje automático.

IV. Tipos de aprendizaje automático

Como se ha comentado en la sección anterior, el aprendizaje automático abarca múltiples técnicas que se utilizan para la identificación de patrones en los conjuntos de datos. Para esto se usan algoritmos con los que se construyen modelos. Un modelo es una representación parcial de la realidad, generalmente una simplificación de la misma, que puede ser utilizado para la realización de predicciones. Los modelos pueden ser tanto una fórmula matemática o un conjunto de fórmulas matemáticas (por ejemplo, una ecuación lineal), conjuntos de reglas (árboles de decisión), estructuras conectadas (las redes neuronales) o sistemas de memorización.

En relación con esta simplificación de los modelos, George E. P. Box, un estadístico británico coautor del libro *Bayesian Inference in Statistical Analysis*, decía: “esencialmente todos los modelos son erróneos, pero algunos son útiles” (“*essentially, all models are wrong, but some are useful”*). En esta cita se resume el hecho de que los modelos no son la realidad, pero son útiles para tomar decisiones sobre ella.

Los algoritmos utilizados para la creación de modelos requieren disponer de un conjunto de datos para su entrenamiento.

El **entrenamiento** es el proceso mediante el cual se identifican los parámetros, las reglas o los pesos en las conexiones que permiten al modelo reproducir los resultados observados en los datos.

Modelos de aprendizaje automático

Antes de realizar el entrenamiento se ha de seleccionar el modelo más adecuado en función tanto de los datos disponibles, como del problema a resolver. En la figura 1.4. se muestran tres ejemplos de modelos.

Figura 1.4. (a)

En la figura 1.4. (a) se muestra un problema de clasificación en el que se han de separar los puntos de un color de los de otro, en esta ocasión, el mejor modelo para separar los dos grupos es una recta, mediante la cual se define una frontera.

Figura 1.4. (b)

Por otro lado, en la figura 1.4. (b) se muestra el mismo tipo de problema, pero la frontera que separa los dos conjuntos de datos no es una recta. Estos dos ejemplos son el mismo tipo de problema, pero el modelo a utilizar es distinto debido a las diferencias existentes en los datos.

Figura 1.4. (c)

Además de la necesidad de adaptar el modelo al tipo de dato, es necesario que se adecue al tipo de problema. Por ejemplo, en la figura 1.4. (c) se muestra un problema de regresión en el que el modelo que mejor se adapta a los datos es una recta.

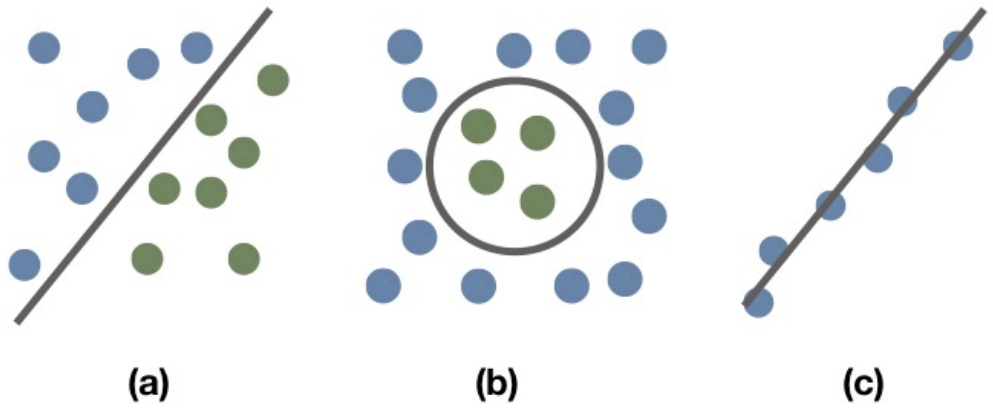

Figura 1.4. Diferentes tipos de modelos disponibles en aprendizaje automático. *Fuente:* elaboración propia.

Anotación: modelos sin proceso de entrenamiento previo

Existen modelos en los que no es necesario llevar a cabo un proceso de entrenamiento previo, como puede ser el método de los k-vecinos más próximos (k-nn, *k-nearest neighbors*). Este es un método de clasificación no paramétrico en el que se determina la clase a la que pertenece un elemento en función de las clases de los k-vecinos más cercanos o los vecinos situados en un radio. El problema se muestra de forma esquemática en la figura 1.5. En ella se plantea lo siguiente: ha de conocerse el color que le correspondería al círculo situado en el centro. Para esto se mide el color de los vecinos en un radio marcado por el círculo negro. En el ejemplo dominan los vecinos de color verde, por lo que se le asignará este color. El algoritmo k-nn es de gran utilidad para problemas de clasificación en los que las fronteras son irregulares. Este algoritmo tiene la ventaja de no requerir un proceso de entrenamiento, pero en ejecución es más lento, debido a la necesidad de consultar los valores de los vecinos.

Figura 1.5. Esquema del problema de los k-vecinos más próximos. *Fuente:* elaboración propia.

Clasificaciones de los algoritmos de aprendizaje automático

Una primera clasificación de los algoritmos de aprendizaje automático se puede llevar a cabo en base a la forma en la que se realiza el proceso de entrenamiento: con un conjunto de datos en el que la solución al problema es conocida previamente, a lo que se denomina aprendizaje supervisado, o no es conocida previamente, lo que se conoce como aprendizaje no supervisado. En resumen:

- **Aprendizaje supervisado**: los algoritmos utilizados para este tipo de aprendizaje requieren que cada una de las instancias empleadas durante el proceso de entrenamiento cuente con un atributo —al que se suele llamar variable dependiente— que el modelo debe reproducir con el resto de atributos —a los que se denominan variables independientes—. Este atributo puede ser una clase o un valor numérico. En producción, el modelo no requiere que este atributo se encuentre disponible —este será la predicción del propio modelo—, solamente necesita disponer de las variables independientes.
- **Aprendizaje no supervisado**: los algoritmos utilizados en este caso no requieren que la respuesta del problema se encuentre previamente etiquetada. A diferencia del tipo anterior, en estos modelos no se busca reproducir un resultado conocido.

campusformacion@inf.com © EDITORIAL CEP S.L. IVÁN GARCIA GARCIA

Para ejemplificar lo explicado, se puede pensar en una entidad financiera que dispone de información histórica de sus clientes. En base a la misma información se pueden crear modelos, utilizando tanto el aprendizaje supervisado como el no supervisado. Por ejemplo, se puede intentar predecir la probabilidad de que un cliente tenga impagos en créditos concedidos. Para esto se utiliza el aprendizaje supervisado, ya que suele existir un histórico de clientes que han pagado o impagado sus créditos en el pasado. El modelo se puede crear con una serie de atributos de entrada: el principal, el tipo de interés, la edad del cliente, la profesión, la antigüedad como clientes, otros productos contratados, etc. El atributo de salida será una etiqueta en la que se indica si el cliente ha impagado o no.

Por otro lado, se pueden utilizar los mismos datos de entrada en un modelo no supervisado para segmentar a los clientes en base a su relación con la entidad. En este caso los parámetros de entrada serían los mismos, pero los valores de salida no vienen determinados en los datos. De hecho, en estos casos no se suele conocer *a priori* ni el número de segmentos, ni las características de estos.

Otra posible clasificación de los algoritmos de aprendizaje automático se basa en el tipo de problema a resolver. Siguiendo esto, los problemas más habituales son:

- **Regresión**: en los problemas de regresión los algoritmos aprenden a predecir valores de una variable dependiente continua a partir de una o más variables independientes.
- **Clasificación**: en los problemas de clasificación los algoritmos aprenden a predecir las categorías de la variable dependiente a partir de una o más variables independientes.
- **Clúster**: en los problemas de clúster los algoritmos aprenden a separar los datos en base a los valores de las variables existentes en los conjuntos de datos.

V. Introducción a scikit-learn

La parte práctica de este curso se va a realizar con el lenguaje de programación Python.

Python es uno de los lenguajes más populares en la actualidad para implementación de modelos de aprendizaje automático debido a su sencillez y potencia.

Debido a que Python es un lenguaje de programación de propósito general, a pesar de su popularidad en entornos de *Data Science*, las funciones de aprendizaje automático no se incluyen en las librerías estándar. Por lo cual es necesario instalar librerías para estas tareas. En el curso, concretamente, se utilizarán las librerías scikit-learn para la mayoría de modelos supervisados y no supervisados, y Theano, para los sistemas conexionistas. Además, para la unidad de reglas de asociación, debido a que este tipo de modelos no se encuentran actualmente dentro de los que implementa scikit-learn, se utilizarán otras librerías que se importarán mediante los procedimientos estándar de Python.

En el curso se asume que el usuario se encuentra familiarizado con el lenguaje de programación en Phyton, así como que tiene conocimientos del uso de pandas y matplotlib.

Anaconda

Para realizar el curso, es recomendable instalar la distribución Anaconda. Anaconda es una distribución libre disponible tanto para Windows, macOS como Linux del lenguaje de programación Python para el procesamiento a gran escala de datos, análisis predictivo y la computación científica, que tiene como objetivo simplificar la gestión y distribución de paquetes. Para esto utiliza el sistema de gestión llamado Conda. La última versión de esta distribución se puede encontrar en la página web de Anaconda (https://www.anaconda.com).

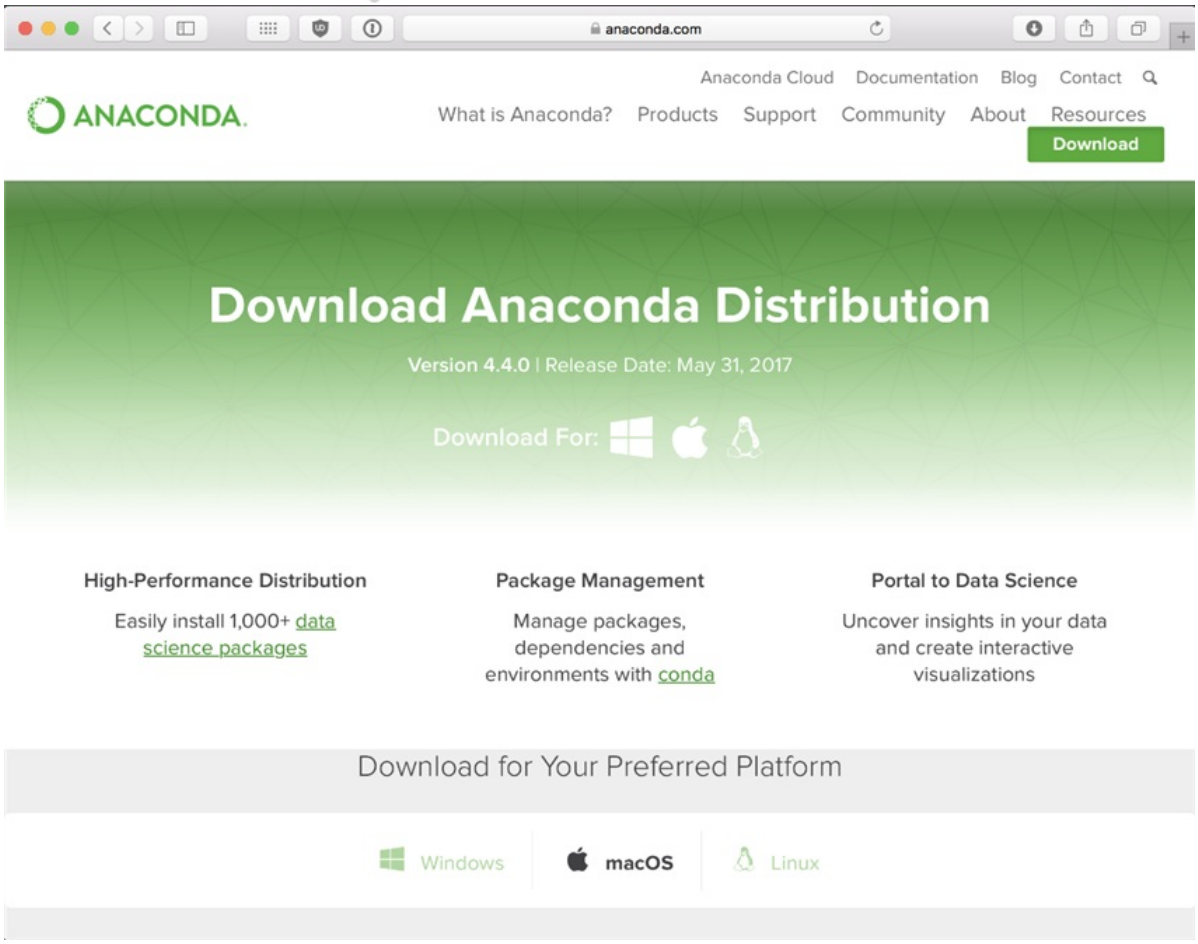

Figura 1.6. Captura de pantalla de la web en la que se puede descargar la distribución Anaconda.

La instalación de Anaconda es sencilla: para cada uno de los sistemas operativos existe un instalador que realiza todas las tareas de forma automática. En los sistemas UNIX (macOS y Linux) no es necesaria una cuenta de administrador, ya que la instalación se puede realizar en la cuenta de usuario. El instalador de Anaconda configura un entorno con Python, Jupyter Notebook, pandas, scikit-learn y otras aplicaciones y librerías, dejando el sistema listo para trabajar. Además, instala el programa Anaconda Navigator, a través del cual se pueden lanzar las aplicaciones, al mismo tiempo que instalar y actualizar paquetes. En la figura 1.7., se muestra una captura de pantalla de este programa en macOS, en otras plataformas el programa es similar.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

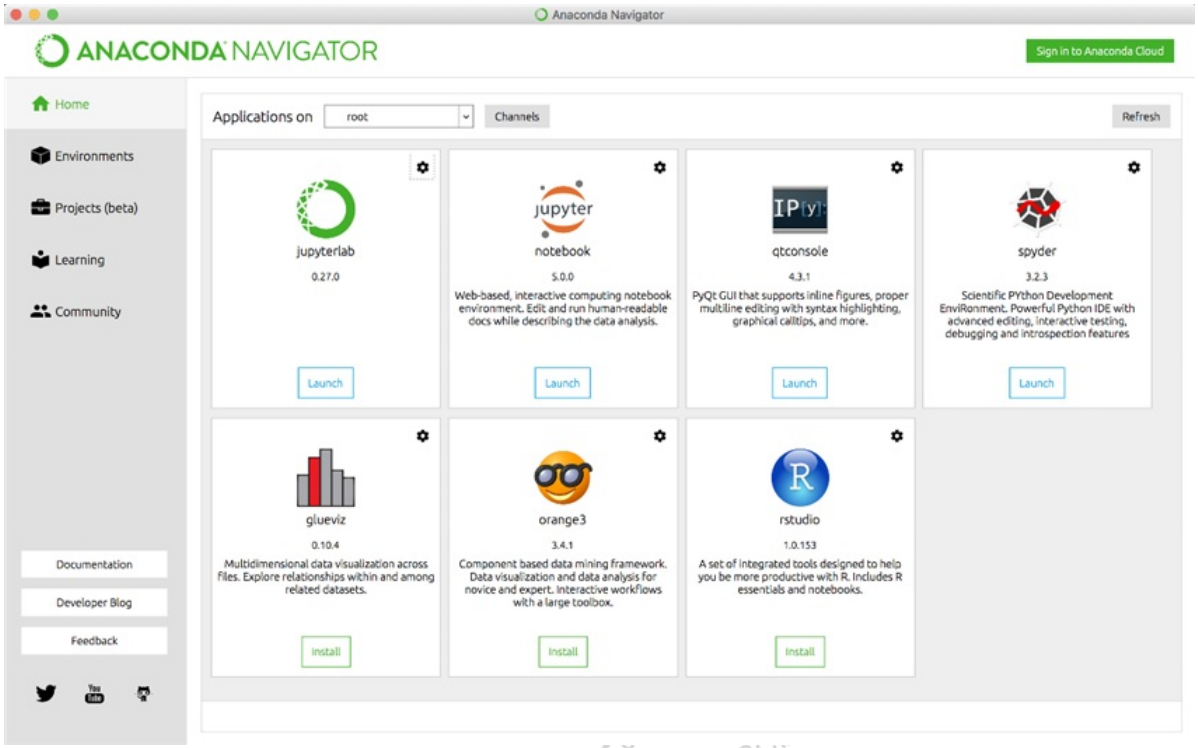

Figura 1.7. Anaconda Navigator en macOS.

campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA

Jupyter Notebook

A través del programa Anaconda Navigator se puede lanzar, en el navegador Jupyter Notebook, el editor que se utilizará para ejecutar el código Python durante el curso. Alternativamente se puede utilizar en la línea de comandos la instrucción:

```
jupyter notebook
```

Esto posibilitará lanzarlo en el navegador Jupyter Notebook. En este caso, aparecerá en el navegador una página como la que se muestra en la figura 1.8., donde se puede navegar por el sistema en busca de los archivos.

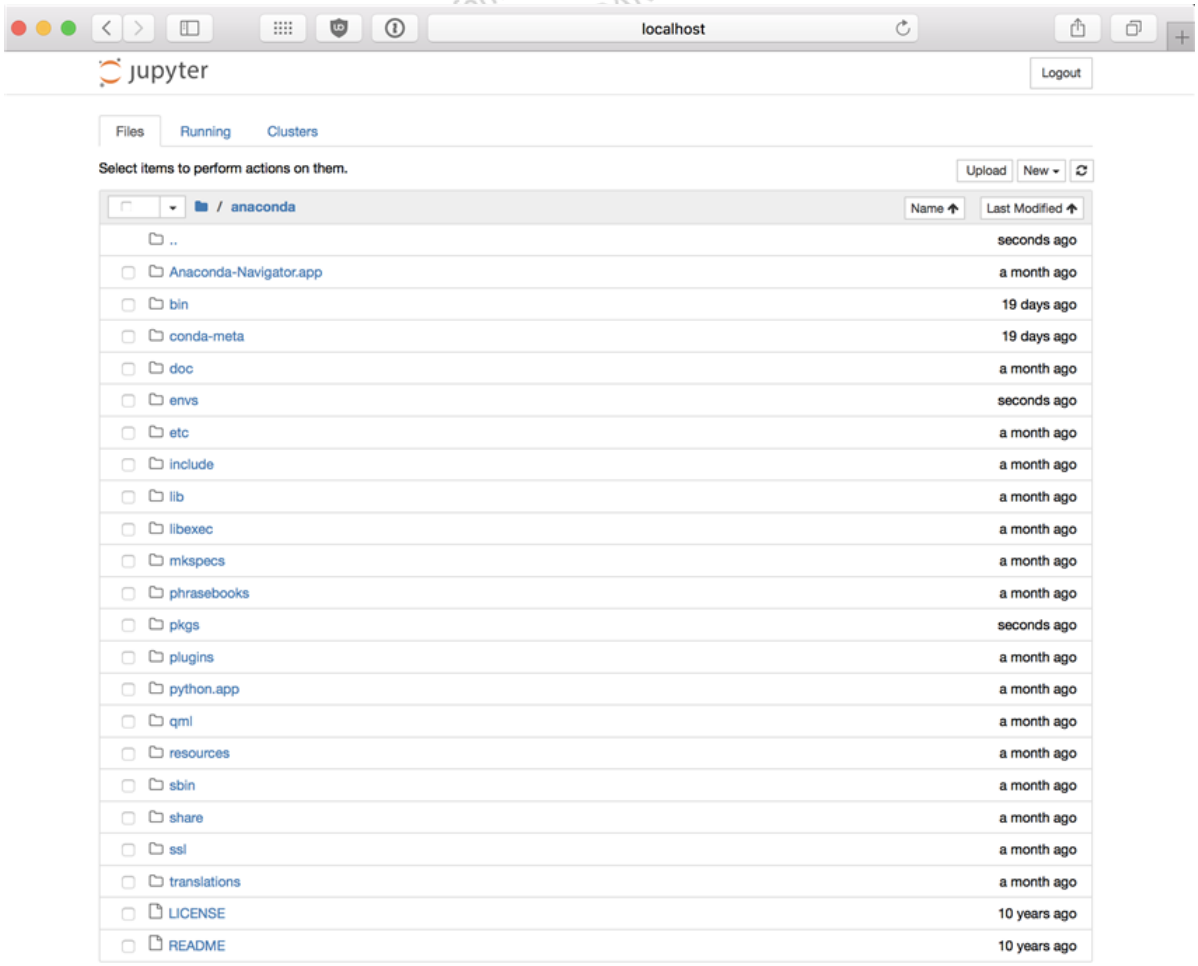

Figura 1.8. Página que aparece al lanzar Jupyter Notebook.

Scikit-learn

Scikit-learn es una librería de open source para el lenguaje de programación Python que implementa diferentes métodos de aprendizaje automático. Esta librería se incluye por defecto en la distribución Anaconda, por lo que no es necesario realizar ningún proceso de instalación adicional.

Scikit-learn cuenta con diversos algoritmos para la implementación de los modelos más utilizados en aprendizaje automático, entre los que se incluyen:

- Clasificación.
- Regresión.
- Clúster-
- Reducción de la dimensionalidad.

En la figura 1.9. se muestra un esquema reproducido de la propia web de scikit-learn, en el que se pueden ver todos los algoritmos disponibles en esta librería y un proceso para seleccionar el más adecuado, en función de los datos disponibles y del análisis que se desea realizar.

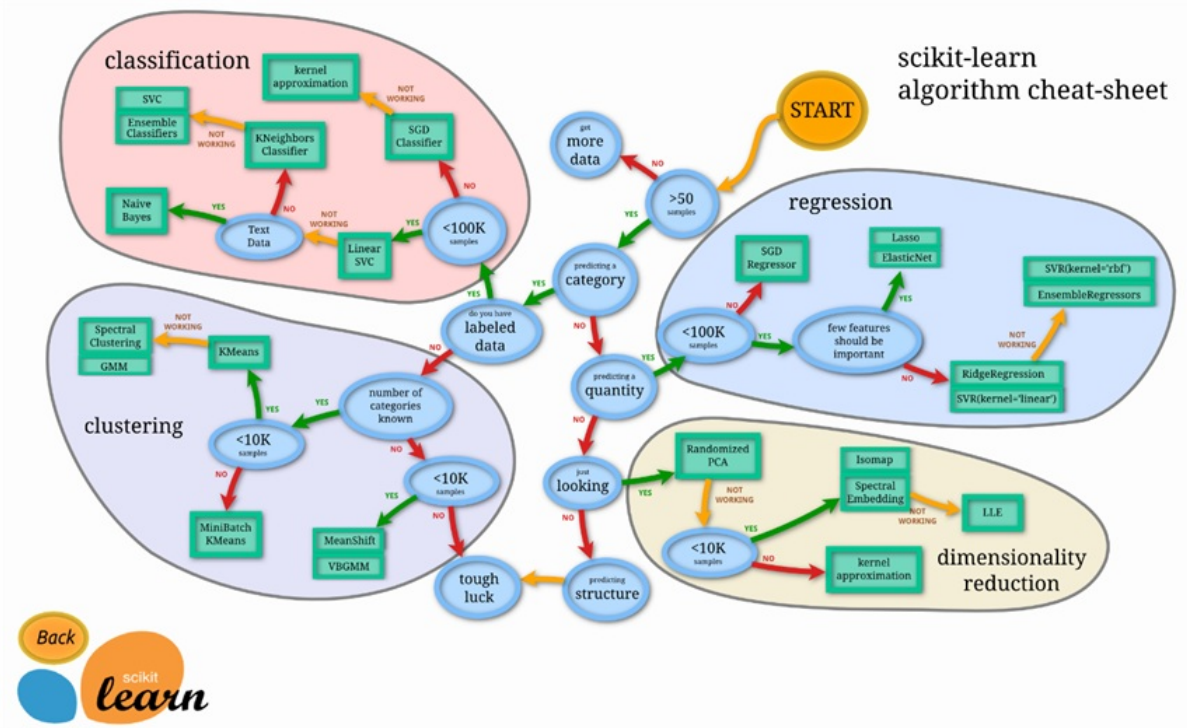

Figura 1.9. Esquema de los modelos disponibles en scikit-learn. *Fuente:* http://scikit-learn.org/stable/_static/ml_map.png]

Para seleccionar el modelo necesario, utilizando el esquema de la figura 1.9, se ha de comenzar en la casilla “start”, que se encuentra en la parte superior derecha. La primera pregunta que realiza el esquema es si se dispone de un conjunto de datos con más de 50 registros, en caso de que no sea así, recomienda recoger más datos. Este límite puede considerarse una cantidad pequeña, incluso para modelos no demasiado complejos, en la mayoría de los casos, pero debe recordarse que no es posible realizar un modelo de aprendizaje automático sin datos, tanto para los supervisados como para los no supervisados.

Una vez que se dispone de un conjunto de datos para modelar, el tipo de preguntas cambia. La primera es saber si se desea predecir una categoría. En caso afirmativo, la siguiente pregunta será si los datos se encuentran etiquetados o no. Esta pregunta se hace para dirigir el modelado a la colección de modelos de clasificación cuando se dispone de los datos etiquetados (aprendizaje supervisado) o a los de clúster, en los casos en que no (aprendizaje no supervisado). Dentro de cada grupo se realizan más preguntas para seleccionar el tipo de modelos adecuado: el tamaño de la muestra y otras informaciones sobre los datos disponibles.

En los casos en los que no se desea predecir una categoría, la pregunta será si se desea predecir una cantidad. En caso afirmativo, se procederá a construir modelos con el conjunto de herramientas de regresión. Aquí se realizan otras preguntas, como el tamaño de los datos y si es necesario seleccionar las variables para seleccionar un modelo u otro.

En el caso de que tampoco se desee predecir una cantidad, el camino que quedará es la reducción de dimensionalidad. En este caso, las opciones son pocas, inicialmente, se propone probar PCA y, en caso de que no funcione, se plantearán otras opciones diferentes en función de la cantidad de datos disponibles.

Tal como se indica en la anterior figura, scikit-learn no es la librería que debería usarse para cualquier otro tipo de análisis que se desee realizar con los datos.

Toda la información acerca de los modelos disponibles en scikit-learn se puede consultar en la página web oficial del proyecto scikit-learn (http://scikit-learn.org/stable/).

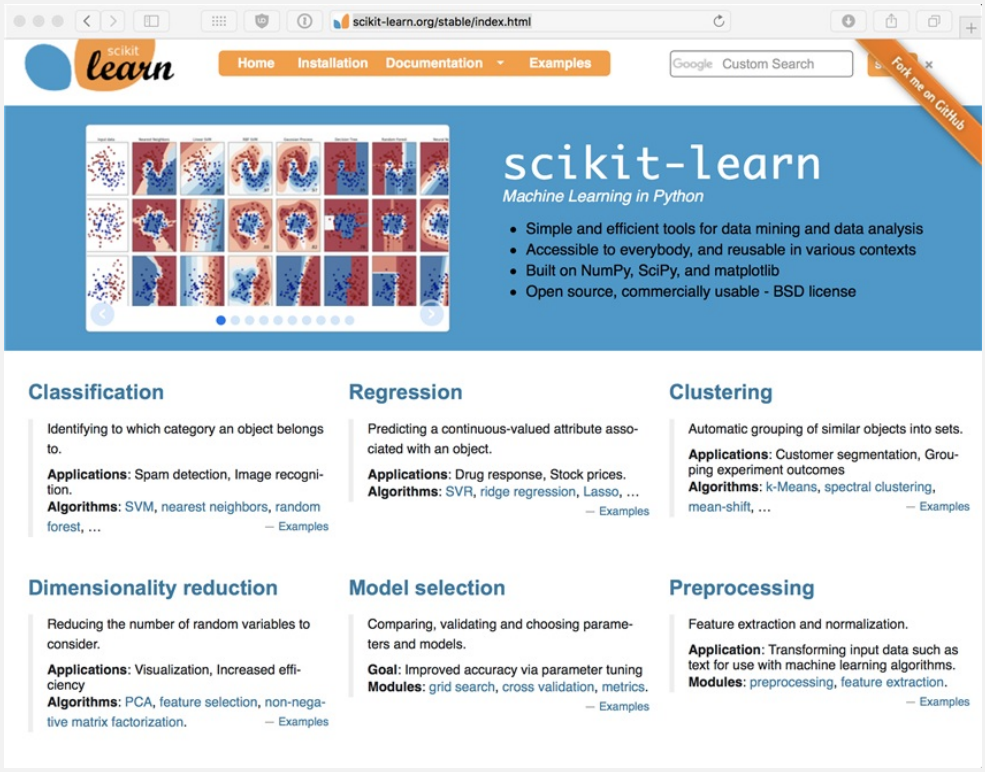

Figura 1.10. Captura de pantalla de la web oficial del proyecto scikit-learn.

La gran popularidad de scikit-learn lo convierte en un estándar de mercado, siendo, por ejemplo, la herramienta con mayor soporte en Core ML (el framework del que se provee Apple en sus sistemas para *machine learning*) para la generación de modelo de aprendizaje automático.

Theano

Theano es un paquete de software que permite la escritura de código simbólico y compilar el resultado a C para una ejecución óptima del mismo. Fue desarrollado por investigadores de aprendizaje automático en la Universidad de Montreal. Su uso no se limita a las aplicaciones de aprendizaje automático, pero fue diseñado con el aprendizaje automático en mente.

La documentación de este software se puede encontrar en la página web que alberga el proyecto Theano (http://deeplearning.net/software/theano/).

Este paquete no se encuentra disponible en la distribución estándar de Anaconda, por lo que su instalación se ha de llevar a cabo utilizando pip y conda. El procedimiento para su instalación se explicará más adelante.

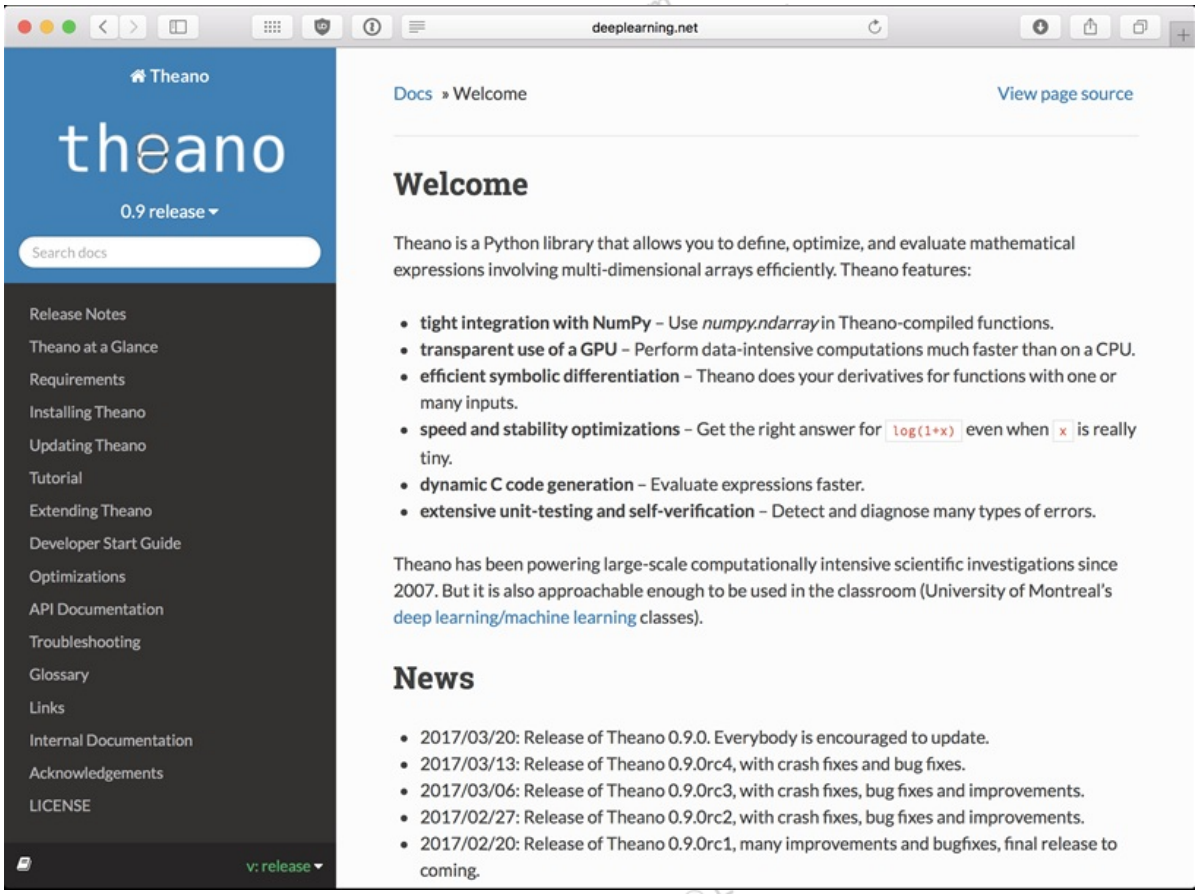

Figura 1.11. Captura de pantalla de la documentación de Theano.

Apple Developer, “Converting Trained Models to Core ML”. [En línea] URL disponible en: https://developer.apple.com/documentation/coreml/converting_trained_models_to_core_ml

VI. Uso básico de un modelo

Una de las grandes ventajas de Phyton, y en especial de scikit-learn, respecto a otras plataformas, es la coherencia para crear modelos. Prácticamente todos los algoritmos disponibles se entrenan de la misma forma, simplemente se han de cambiar algunas opciones en función del tipo de modelo, y también se ejecutan siempre igual.

Los códigos utilizados en esta sección se pueden consultar en el notebook U2_codigos.ipynb que se ha facilitado al inicio de la unidad A continuación, se muestra una captura de pantalla en la figura 1.12.

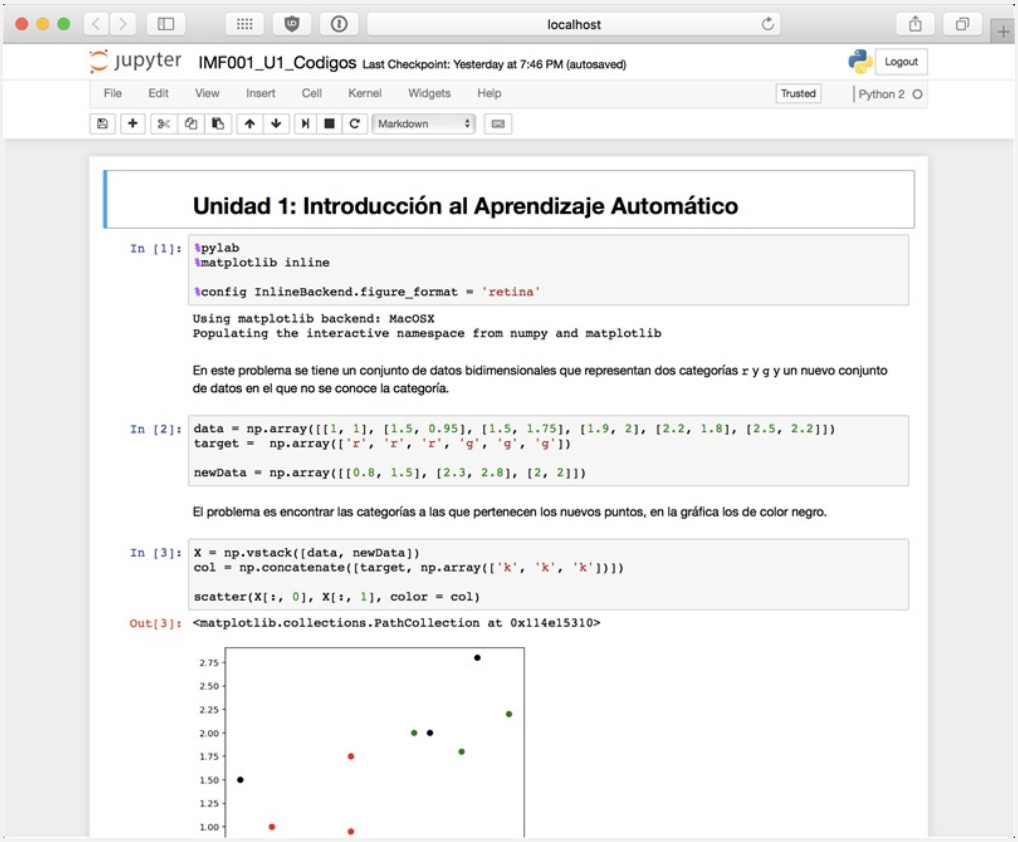

Figura 1.12. Captura de pantalla del notebook U2_ codigos.ypynb

La creación de un modelo en scikit-learn comienza siempre de la misma forma: simplemente se ha de seleccionar el modelo, importarlo y utilizar el constructor para crear un nuevo objeto del tipo seleccionado. La forma básica para esto sería: m = Model()

Model es el constructor del modelo que suministra scikit-learn y m es objeto que contiene el modelo resultante. La mayoría de modelos admiten diferentes opciones en el constructor, que indicarán la forma en la que se desea modelar los datos. Las diferentes opciones para cada uno de los modelos se verán en las secciones correspondientes.

campusformacion.imf.com - EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

Una vez creado el modelo, ha de ser entrenado, para esto se utiliza el método fit, que ha de ser llamado con un conjunto de datos y, en los casos en que el tipo de aprendizaje sea supervisado, la variable con los resultados conocidos. Esto es: m.fit(data, [target])

data son los datos de entrenamiento y target es la variable con los datos objetivos, en el caso de que el modelo sea supervisado, en los no supervisados no se utilizará. A partir de este punto el modelo m se encontrará ya entrenado y se podrán realizar predicciones utilizando el método preditc, al que se le tendrá que suministrar un conjunto de datos como el de entrenamiento. Así, para obtener una predicción, será necesario escribir: m.preditc(data)

En la predicción de los modelos supervisados, no se ha de indicar la variable dependiente, ya que esto es lo que el modelo desea predecir.

Finalmente, también es posible realizar transformación de datos utilizando las librerías de scikit-learn. Para esto se ha de crear el objeto correspondiente con sus opciones: obj = Transform()

Y, para hacer la trasformación, se llamará al método transform del objeto, de una forma similar a como se indica en la línea: datos = obj.transform(data)

A modo de ejemplo: se puede implementar un modelo de clasificación k-nn como el que se ha explicado en la figura 1.4. En scikit-learn, el constructor que permite crear este tipo de modelos se llama RadiusNeighborsClassifier y se encuentra disponible en el espacio de nombres sklearn.neighbors.

Implementar el algoritmo

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

1

Antes de implementar el algoritmo, es necesario disponer de un conjunto de dados de ejemplo, es decir, un conjunto de puntos a los que se le ha asignado una clase que se puede representar mediante un color. Por ejemplo, se puede utilizar el vector de puntos [[1, 1], [1.5, 0.95], [1.5, 1.75], [1.9, 2], [2.2, 1.8], [2.5, 2.2]], al que se asigna el color rojo a los tres primeros y verde a los tres últimos. Además, es necesario disponer de un conjunto de datos sobre el que se desea realizar una predicción, por ejemplo, se puede utilizar el vector [[0.8, 1.5], [2.3, 2.8], [2, 2]]. Estos puntos se pueden representar gráficamente en un plano, como se muestra en la figura 1.13., donde los puntos negros son los valores sin clasificar.

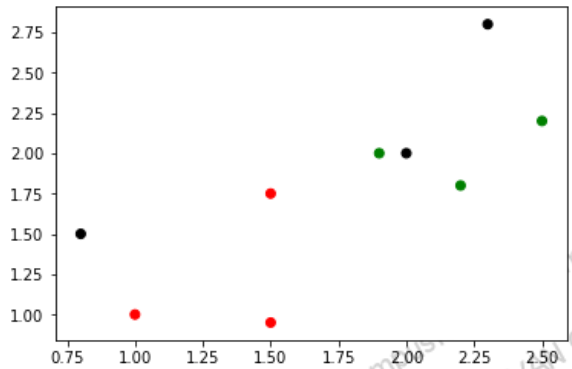

Figura 1.13. Representación de los datos utilizados para probar el algoritmo k-nn.

2

El código con el que se puede implementar el algoritmo de k-nn se puede ver en la figura 1.14. En estas líneas los datos de entrenamiento se encuentran en la variable data, los colores en target y los datos para realizar la predicción en newData.

```
from sklearn.neighbors import RadiusNeighborsClassifier

# Creamos el modelo sin entrenar:
model = RadiusNeighborsClassifier(radius = 1)

# Entrenamos el modelo:
model.fit(data, target)

# Predecimos la clase para tres puntos diferentes:
prediction = model.predict(newData)

print prediction
```

Figura 1.14. Implementación con RadiusNeighborsClassifier.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

3

La primera línea del código de la figura 1.14. realiza la importación del constructor, este paso es imprescindible para acceder al mismo. En la segunda línea se crea el objeto, indicando al constructor que el radio en el que se evaluarán los vecinos es de 1. En la tercera línea se entrena el modelo, como este es supervisado se han de indicar tanto las variables independientes como la dependiente. Para un objeto de esta familia, el entrenamiento simplemente consiste en guardar los datos, no se realiza ninguna otra operación. Finalmente, se puede realizar la predicción y sacar esta por pantalla. Se obtiene rojo para el primer punto y verde para los otros dos. Los resultados se pueden ver en la figura 1.15.

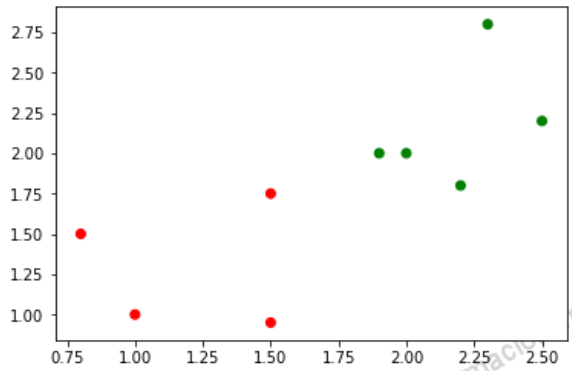

Figura 1.15. Resultados del clasificador RadiusNeighborsClassifier.

4

Al utilizar los vecinos existentes en un radio del punto en el que se realiza la predicción, puede darse la situación de no encontrar ninguno. Al intentar realizar la predicción en uno de estos puntos, el clasificador RadiusNeighborsClassifier dará un error, salvo que se indique una etiqueta específica para estos valores outlier mediante la opción outlier_label en el constructor. En el ejemplo anterior se debería escribir:

```
model = RadiusNeighborsClassifier(1, outlier_label = 'k')
```

Esto sirve para indicar que, en caso de no encontrar ningún vecino, se asocie la etiqueta ‘k’ en estos puntos.

Alternativamente, en lugar de utilizar los vecinos que se encuentran en un radio concreto del elemento que se desea clasificar, se puede usar una cantidad de vecinos independiente de los que se encuentren próximos al primero. Utilizando esta solución se puede evitar que no sea posible clasificar en una zona por falta de vecinos. Los clasificadores de este tipo se pueden implementar utilizando la clase KNeighborsClassifier, que también se encuentra disponible en el espacio de nombres sklearn.neighbors.

A continuación, un ejemplo de código en el que se vuelven a utilizar los mismos datos que en el ejemplo anterior, pero con un nuevo clasificador, se puede ver en la figura 1.16. En esta ocasión se ha fijado el número de vecinos utilizados para la clasificación a tres, es un número adecuado, ya que solamente hay seis puntos en el conjunto de datos de entrenamiento. Uno es poco y cinco comprendería a todos los vecinos.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

```
from sklearn.neighbors import KNeighborsClassifier

# Creamos el modelo sin entrenar:
model = KNeighborsClassifier(n_neighbors = 3)

# Entrenamos el modelo:
model.fit(data, target)

# Predecimos la clase para tres puntos diferentes:
prediction = model.predict(newData)

print prediction
```

Figura 1.16. Implementación con KNeighborsClassifier.

5

Al ejecutar este código se obtienen exactamente los mismos resultados que en el caso anterior, generándose la misma gráfica que se muestra en la figura 1.15. Esto se debe a que este problema es sencillo y ambos clasificadores funcionan de igual manera.

Estos modelos tan sencillos se pueden utilizar para clasificar objetos en conjuntos de datos más complejos, como puede ser el conjunto de datos Iris. Este conjunto contiene 50 muestras de tres especies de Iris (Iris setosa, Iris virginica e Iris versicolor), en las que se midieron cuatro rasgos de cada muestra: el largo y el ancho de los sépalos y pétalos. Es un conjunto de datos muy utilizado para la construcción de modelos de ejemplo, encontrándose disponible en scikit-learn.

En el ejemplo, para que sea posible representar los datos en un plano se trabajará solamente con las dos primeras columnas del conjunto de datos: el largo y el ancho de los sépalos. La importación de los datos se realiza utilizando la función load_iris, que se encuentra disponible en el conjunto de datos de ejemplo de scikit-learn (sklearn.datasets).

La carga de los datos y la construcción del modelo se pueden ver en la figura 1.17. En este código se observa que el número de vecinos se ha fijado en 5, en esta ocasión la cantidad de datos es mayor, lo que producir unos resultados más estables.

```
from sklearn.datasets import load_iris

iris = load_iris()

# Para facilitar solamente se utilizan las dos primeras celdas
X = iris.data[:, 0:2]
y = iris.target

knn = KNeighborsClassifier(5)
knn.fit(X, y)
```

Figura 1.17. Construcción de un modelo con los datos de Iris y KNeighborsClassifier.

campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA

6

Una opción para visualizar la predicción es representar en un plano los dos valores utilizados para realizar la clasificación y representar, a su vez, las predicciones en cada punto con un color diferente. Esto es lo que se efectúa mediante el listado de la figura 1.18. En este código, lo primero que se hace es definir dos mapas de colores, uno fuerte y otro un poco más suave. Posteriormente se calcula el máximo y el mínimo en los dos ejes seleccionados, a los que se le suma y resta respectivamente la unidad para dejar un margen en la gráfica. A partir de estos valores se crea una malla con una separación de 0,05. En todos los puntos de esta malla se calcula una predicción con el modelo y se representa en una gráfica.

```
from matplotlib.colors import ListedColormap

# Version clara y oscura de los coloes
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

# Creación de un conjunto de datos para
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.05),
                     np.arange(y_min, y_max, 0.05))

Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

pcolormesh(xx, yy, Z, cmap = cmap_light)
scatter(X[:, 0], X[:, 1], c=y, cmap = cmap_bold)
xlim(xx.min(), xx.max())
ylim(yy.min(), yy.max())
```

Figura 1.18. Código para la representación gráfica de los resultados.

7

Los resultados del código de la figura 1.18. se muestran en la figura 1.19. En esta se puede ver cómo el clasificado realiza su trabajo y asigna un color diferente a cada punto. En la parte inferior de la imagen, hay una zona en la que los puntos azules y verdes se superponen. En caso de ser necesaria una predicción más suave, se puede aumentar el número de vecinos utilizados para la predicción.

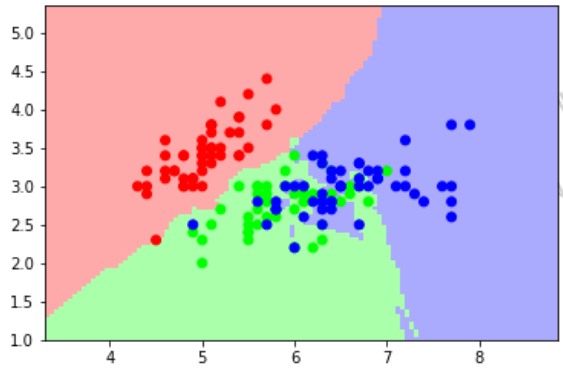

Figura 1.19. Clasificación de Iris con 5 vecinos.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA

```
8
```

Al aumentar de 5 a 15 el número de vecinos utilizados para realizar la clasificación, se puede observar que las fronteras se suavizan, como se muestra en la figura 1.20.

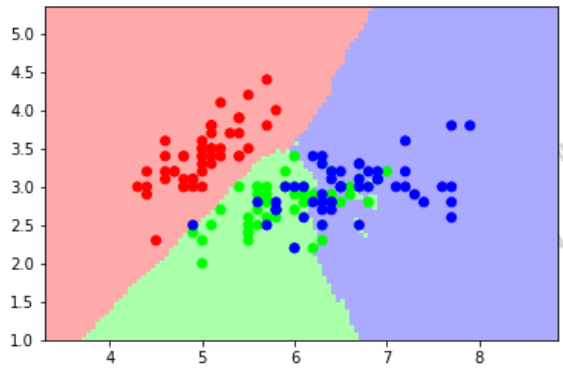

Figura 1.20. Clasificación con Iris con 15 vecinos.

VII. Resumen

Esta primera unidad ha comenzado con una introducción a la minería de datos, señalando el papel que juega dentro de los procesos KDD, mediante los cuales se puede extraer conocimiento de las bases de datos. Junto a esto se han visto los diferentes estudios que se pueden realizar en minería de datos.

La unidad ha continuado con una presentación del aprendizaje automático. Las herramientas que proporciona esta rama de la inteligencia artificial son clave dentro de la minería de datos, ya que son las que permiten identificar los patrones ocultos. Además, se ha visto una clasificación de los tipos de problemas que se pueden solucionar con el aprendizaje automático.

Posteriormente, se han presentado las librerías de Python que se van a utilizar en este curso: scikit-learn y Theano. La primera es una librería que contiene la resolución de problemas de regresión, clasificación, análisis de clúster y reducción de la dimensionalidad. La segunda permite crear con facilidad sistemas conexionistas.

La unidad ha terminado con la implementación de un modelo de ejemplo empleando scikit-learn para tomar contacto con la librería.

© EDICIONES ROBLE, S.L.

Ejercicios

Caso práctico

Utilizar la clase RadiusNeighborsClassifier para crear un clasificador que permita identificar la especie de Iris del conjunto de datos Iris en base al largo y ancho de los sépalos.

Probar el funcionamiento con los radios de 0,3, 0,5, 0,7, 0,9 y 1,1.

Solución

En este caso, se solicita volver a clasificar el largo y ancho de los sépalos, utilizando RadiusNeighborsClassifier para radios de 0,3, 0,5, 0,7, 0,9 y 1,1, en lugar de KNeighborsClassifier.

A la hora de utilizar el algoritmo indicado en el enunciado se ha de tener en cuenta que, además de las tres categorías, se puede obtener un valor indeterminado cuando no existe ningún registro en el radio indicado. En estos puntos se ha de utilizar un color diferente, para esto se puede añadir un nuevo color a cmap_light, que puede ser un gris como '#AAAAAA'.

Para la carga de los datos se pueden utilizar las mismas líneas de código empleadas anteriormente en la figura 1.17. Ahora se ha de importar RadiusNeighborsClassifier y se le ha de asignar un valor a los outliers, como tenemos 3 valores en el conjunto de datos se le puede asignar el 3, ya que 0, 1, y 2 son los utilizados en las categorías. Esto es lo se que muestra en la figura 1.21. para el primer radio.

```
from sklearn.neighbors import RadiusNeighborsClassifier

knn = RadiusNeighborsClassifier(0.3, outlier_label = 3)
knn.fit(X, y)
```

Figura 1.21. Clasificador con radio 0,3 y valor para outliers de 3.

Se ha de repetir el código de la figura 1.18, pero añadiendo un nuevo color a cmap_light, que será el utilizado para las zonas de outliers. Esto da lugar a una gráfica como la que se muestra en la figura 1.22.

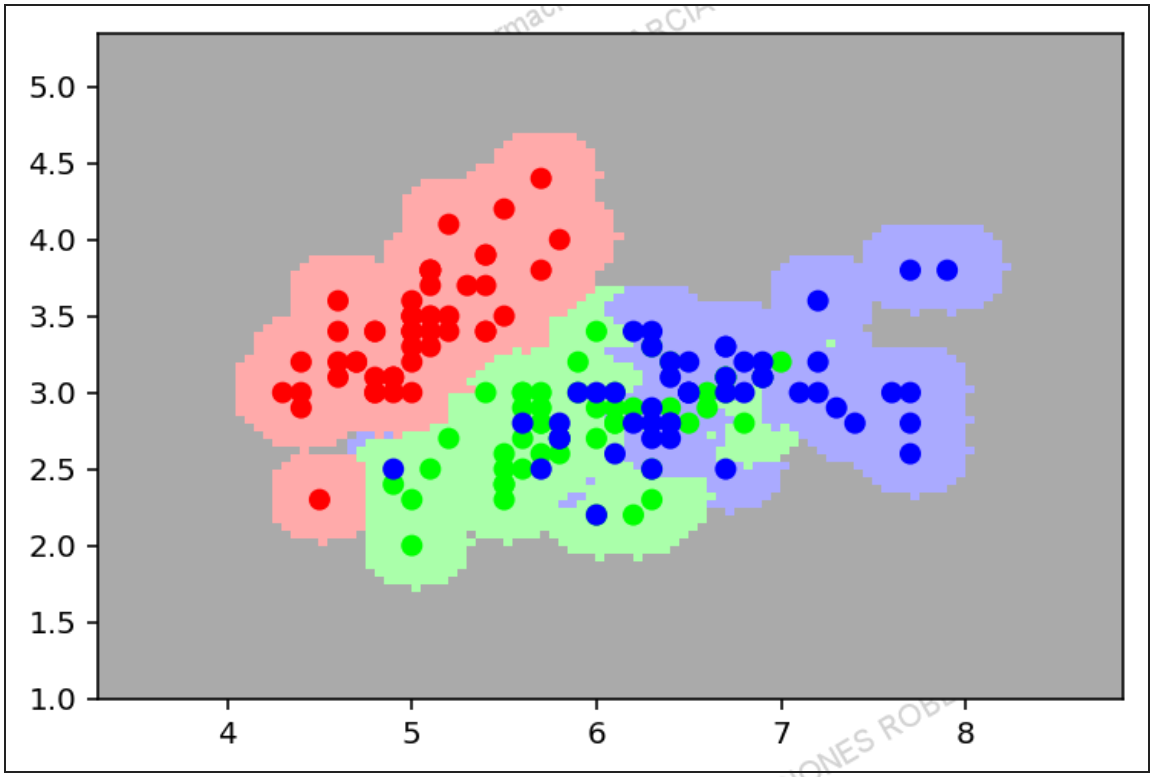

Figura 1.22. Resultado de RadiusNeighborsClassifier con radio 0,3.

En la figura 1.22. se puede observar que las fronteras son más irregulares que en el caso anterior, además de que en la mayoría del plano no hay solución. El resto de figuras se puede obtener repitiendo este proceso con diferentes valores para el radio.

El código completo del caso se puede encontrar en elnotebook U2_caso.

campusformacion.imf.com © EDICIONES ROBLE, S.L.
IVAN GARCIA GARCIA

Recursos

Enlaces de Interés

Theano documentation
http://deeplearning.net/software/theano/index.html#)

scikit-learn documentation
http://scikit-learn.org/stable/documentation.html

Bibliografía

- ***Applied Data Mining for Business and Industry.*** : Giudici P. y Figini S. Wiley; 2009, 2nd Edition
- ***Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*** : EMC Education Services. John Wiley & Sons Inc; 2015.
- ***Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking.*** : Provost F. y Fawcett T.0'Reilly; 2013.
- ***Machine Learning: Hands-On for Developers and Technical Professionals*** : Bell J. Wiley; 2014.

Glosario.

- **Aprendizaje automático**: Es el área de la inteligencia artificial que agrupa las diferentes técnicas que permiten a las computadoras aprender patrones a partir de conjuntos de datos de ejemplo.
- **Aprendizaje no supervisado**: Son los modelos de aprendizaje automático en los cuales, durante el proceso de entrenamiento, se utilizan conjuntos de datos en los que se desconoce la solución.
- **Aprendizaje supervisado**: Son los modelos de aprendizaje automático en los cuales, durante el proceso de entrenamiento, se utilizan conjuntos de datos en los que se conoce la solución.
- **Inteligencia artificial**: Es el área de las ciencias de la computación que se encarga de estudiar y proveer de inteligencia a las máquinas.
- **KDD**: Descubrimiento de Conocimiento en Bases de Datos, de las siglas en inglés Knowlege Discovery in Databases.
- **Minería de datos**: Es el campo de las ciencias de la computación que se utiliza para descubrir patrones en los conjuntos de datos.
- **Modelo**: En aprendizaje automático, un modelo es una representación parcial de la realidad, generalmente una simplificación de la misma, que se puede utilizar para la realización de predicciones.
- **Variable dependiente**: En un modelo es la variable, cuyo valor se pretende predecir.
- **Variable independiente**: En un modelo es el conjunto de variables que se utilizan para realizar las predicciones.

campusformacion.imf.com © EDICIONES ROBLE, S.L. IVAN GARCIA GARCIA