Universidad Intercontinental de la Empresa

Configuración prácticas de laboratorio

Aprendizaje Automático

Borja González Seoane

S01, 4 de septiembre de 2024

1. Preparación del entorno de trabajo

Para la realización de las prácticas de laboratorio de la asignatura se trabajará con Python y Jupyter Notebook. Por tanto, es necesario que los estudiantes tengan instalado en sus máquinas este *software*.

A continuación, se detallan algunos pasos orientativos a seguir para la instalación de Python y Jupyter en los sistemas operativos más comunes. Estas instrucciones pueden considerarse como una posible vía para la instalación, pero existen otras alternativas que también son plenamente válidas. Lo necesario es disponer de un entorno de trabajo con:

1. Python 3.11 o superior.
2. Jupyter Lab, versión rondando la 4.
3. Opcionalmente, extensión de Jupyter[1] para Visual Studio Code, para poder trabajar con *notebooks* desde este IDE.
4. Librerías de Python que se emplearán en las prácticas de laboratorio. Se irán añadiendo más a lo largo del curso:
   a) Matplotlib[2].
   b) NumPy[3].
   c) Pandas[4].
   d) Plotly[5].
   e) Scikit-Learn[6].
   f) Seaborn[7].
   g) UCI-ML-Repo[8].

[1] https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter
[2] https://matplotlib.org/
[3] https://numpy.org/
[4] https://pandas.pydata.org/
[5] https://plotly.com/python/
[6] https://scikit-learn.org/stable/
[7] https://seaborn.pydata.org/
[8] https://github.com/uci-ml-repo/ucimlrepo

Para cursar la instalación, una vez que se disponga de Python 3.11[9] o una versión superior, se puede seguir el siguiente procedimiento:

1. Opcionalmente —aunque se recomienda—, crear un entorno virtual para la asignatura. Esto es útil para evitar conflictos con otras versiones de Python o con otros paquetes que puedan estar instalados en el sistema, manteniendo las dependencias aisladas y bajo control. Además, permite compartir fácilmente el entorno de trabajo con terceros, ya que se puede exportar un manifiesto con la lista de paquetes instalados en el entorno virtual y sus versiones exactas, de forma que se pueda replicar el entorno en otra máquina con facilidad.

   Existen diferentes herramientas para la creación y gestión de entornos virtuales y dependencias en Python. En esta asignatura no se impondrá el uso de ninguna en particular, pero se recomienda que los estudiantes se familiaricen con alguna de ellas, ya que es una práctica común en el desarrollo de *software* en Python, que sin duda les será útil en el futuro.

   Como recomendación, se puede emplear Conda[10], ya que es una utilidad muy completa y versátil que permite gestionar entornos virtuales y paquetes de Python de forma sencilla. Además, Conda es muy popular en la industria, por lo que su uso puede ser beneficioso para los estudiantes. En este caso, se recomienda así mismo optar por la descarga de Miniconda, que es una versión ligera de Conda que no incluye todos los paquetes que vienen por defecto con Anaconda, y que es suficiente para la mayoría de los casos.

   Otra alternativa muy recomendable y que está ganando gran popularidad en los últimos años es Poetry[11], que es una herramienta moderna y muy completa para la gestión de dependencias, entornos virtuales, empaquetado y publicación de paquetes de Python. Poetry integra las funcionalidades para las que normalmente se empleaban varias utilidades diferentes, como `pip`, `venv`, `setuptools`, `twine`, etc., lo que simplifica y unifica el flujo de trabajo. Además, permite una definición de dependencias más clara y legible, con agrupaciones por tipo de uso, como dependencias de desarrollo, dependencias de test, dependencias de producción, etc.; inclusión de datos del proyecto, como el nombre, la versión, la descripción, el autor, la licencia, etc., y la gestión de *scripts* personalizados, como comandos de test, de empaquetado, de publicación, etc.

   No obstante, bien es cierto que la configuración inicial de Poetry puede resultar algo más compleja que la de Conda, por lo que se ponen a disposición de los estudiantes las tutorías, para resolver dudas y ayudar en la preparación del entorno de trabajo.

   Una vez instalada una de estas herramientas u otra equivalente, se puede proceder a la creación del entorno virtual. En líneas generales, el proceso de creación del entorno virtual sería el siguiente:

```
# Creación del entorno virtual con Conda
conda create --name uie-aa python=3.11
conda activate uie-aa # Activar
```

```
# Creación del entorno virtual con Poetry
poetry new uie-aa
cd uie-aa
poetry env use 3.11 # El entorno virtual se crea automáticamente
poetry shell # Activar
```

```
# Creación del entorno virtual con VEnv
python -m venv uie-aa
source uie-aa/bin/activate # Activar
```

2. Instalar Jupyter Lab en el entorno virtual, o en el sistema si no se ha creado un entorno virtual (previamente se debe activar el entorno virtual si se ha creado uno):

```
pip install jupyterlab
```

[9] https://www.python.org/downloads/

3. Se puede comprobar que Jupyter Lab se ha instalado correctamente ejecutando el siguiente comando:

```
jupyter lab
```

   El comando anterior abrirá una pestaña en el navegador web con Jupyter Lab. Si no se abre automáticamente, se puede acceder a través de la dirección que se mostrará en el *log* de la terminal, típicamente `http://localhost:8888/lab`.

4. Opcionalmente —aunque se recomienda—, instalar la extensión de Jupyter para Visual Studio Code. Para ello, se debe abrir Visual Studio Code, ir a la pestaña de extensiones y buscar `ms-toolsai.jupyter`. Una vez encontrada, se debe instalar. Esta extensión permite trabajar con *notebooks* de Jupyter directamente desde Visual Studio Code, lo cual resulta cómodo para aquellos que prefieran trabajar con este IDE, manteniendo posibles personalizaciones y configuraciones que ya tengan establecidas. Además, con esta extensión basta con abrir los *notebooks* como un archivo más, y el IDE se encargará por detrás de la gestión del *kernel* de Jupyter, sin tener que iniciar Jupyter Lab de forma explícita.

   Existen este tipo de extensiones para otros IDE populares, como JetBrains PyCharm, JetBrains DataSpell, Atom, etc.

5. Por último, se deben instalar algunas librerías de Python que se utilizarán en las primeras prácticas de laboratorio. En prácticas posteriores se irán añadiendo más librerías. A saber:

   a) `matplotlib`. Orientativamente, versión 3.7.3.

   b) `numpy`. Orientativamente, versión 1.26.4.

   c) `pandas`. Orientativamente, versión 2.2.2.

   d) `plotly`. Orientativamente, versión 5.24.0.

   e) `scikit-learn`. Orientativamente, versión 1.5.1.

   f) `seaborn`. Orientativamente, versión 0.13.2.

   g) `ucimlrepo`. Orientativamente, versión 0.0.6.

   Para instalar estas librerías, se puede emplear el siguiente comando:

```
pip install matplotlib numpy pandas plotly scikit-learn seaborn ucimlrepo
```

6. Alternativamente, se puede usar un comando *magic* de Jupyter para instalar las librerías directamente desde el *notebook*. Para ello, se debe ejecutar la siguiente celda, con las librerías que se quieran instalar, típicamente las que se vayan a emplear en ese mismo *notebook*:

```
%pip install matplotlib # Inclúyanse aquí librerías que apliquen
```

   Este comando instalará siempre las librerías en el entorno de Python que esté asociado al *kernel* de Jupyter que esté activo en ese momento.

2. Comprobación del entorno con un *notebook* de demostración

Para comprobar que el entorno de trabajo está correctamente configurado, se proporciona un *notebook* de demostración, que se puede encontrar en el Campus Virtual, `pl00_demo_notebook.ipynb`. Se debería poder visualizar y reejecutar este *notebook* tanto en el interfaz de Jupyter Lab como en Visual Studio Code. Se puede aprovechar para explorar las diferentes funcionalidades que proporcionan

cualquiera de los dos entornos: ejecutar una celda, reiniciar el *kernel*, añadir nuevas celdas, ejecutar todas las celdas, etc.

En este *notebook* también se incluye un resumen de la sintaxis básica de Markdown[12], que es el lenguaje de marcado que se emplea en los *notebooks* de Jupyter como extensión del texto plano. Con Markdown es posible dar formatos simples al texto, añadir enlaces, crear listas, etc. La sintaxsis de Markdown no está sujeta a evaluación en esta asignatura, pero se recomienda a los estudiantes que se familiaricen con ella, ya que se utiliza ampliamente en la industria para la documentación de proyectos *software*, la escritura de wikis, la redacción de posts en foros y blogs, en repositorios de GitHub o similares, etc. Además, es una posibilidad sencilla de escribir y leer, ligera y perdurable en el tiempo para gestionar notas personales o apuntes de clase.

[12] https://www.markdownguide.org/basic-syntax/