Universidad del Desarrollo
Magister en Data Science
Profesor: Víctor Navarro Aránguiz

Python para Data Science

Tarea 5

16 de julio de 2024

Fecha de entrega: 21 de Julio de 2024, 23:59:59
Objetivos:

- Familiarizarse con el manejo de versiones a través de Git.

Puntaje total: 12 puntos

Pasos previos

Git

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar y registrar cambios en su código de manera eficiente. Hoy en día es altamente utilizado en prácticamente todas las compañías.

Instalación

Existen distintas formas de instalar git:

1. Usando el instalador oficial: https://git-scm.com/download
2. A través del *anaconda prompt*: `conda install anaconda::git`
3. A través de *GitHub-Desktop*: https://desktop.github.com/download/

De todos ellos el de uso más amigable es GitHub-Desktop, pues trae consigo una interfaz gráfica. Un punto importante es que quienes ya estén acostumbrados a trabajar con VSCode ya habrán visto en algún momento el *source control*.

GitHub

GitHub es una plataforma que utiliza Git para el control de versiones y facilita la colaboración en proyectos. Deberán crearse una cuenta si es que aún no la tienen: https://github.com/signup

Python environments

Video explicativo: https://www.loom.com/share/9a0687b43ef24aab80aecb0e9dbf1e57

Un punto importante al momento de trabajar en nuevos proyectos es el uso correcto de un *environment*. Podemos entenderlo como un espacio aislado con su propia instalación de Python donde podremos instalar paquetes sin alterar otros environments. Anaconda instala por defecto el environment `base`.

Idealmente se debe usar un environment nuevo por proyecto. La razón reside en que las librerías/paquetes están en constante actualización, las que muchas veces alteran el funcionamiento de muchas componentes. Es muy común querer instalar una librería nueva que necesita por ejemplo la última versión de NumPy, y puede ser que esta última versión rompa el funcionamiento de otra librería que ya tenías instalada. Así, para evitar conflictos entre paquetes, se prefiere definir un nuevo *environment*.

Creación de un environment

Para crear un nuevo environment podemos usar el `anaconda prompt`. Una vez abierto debemos ejecutar el siguiente comando:

```
conda create --name mi_entorno python=3.12 ipykernel
```

Donde:

- `mi_entorno` será el nombre del environment
- `python=3.12` indica que queremos instalar dicha versión de Python
- `ipykernel` indica que instalaremos dicho paquete en nuestra environment, el cual nos permitirá usarlo con jupyter notebooks

Una vez instalado, podremos activarlo ejecutando:

```
conda activate mi_entorno
```

Habilitando el environment en jupyter notebook

Una vez activado aún nos falta un paso para poder utilizarlo dentro de un jupyter notebook, el cual es incluir el *kernel* (la instalación de Python) en la lista de kernels interactivos de Python. Para ello, debemos ejecutar el siguiente comando:

```
python -m ipykernel install --user --name=mi_entorno --display-name="mi_entorno (Python 3.12)"
```

Donde:

- `--name=mi_entorno` se refiere a qué environment usar
- `--display-name="mi_entorno (Python 3.12)"` indica con qué nombre queremos que aparezca en el listado

Trabajando con un repositorio git

Video explicativo: https://www.loom.com/share/8f2c93e1384c4e889e83fb40c476ab92

Crear un repositorio (3 puntos)

Utilice GitHub para crear un nuevo repositorio público, la siguiente guía podría serle de utilidad: *Quickstart for repositories*. El nombre del repositorio debe ser **tarea-05**.

Una vez creado, clone el repositorio dentro de su computadora utilizando `git`.

Creación de un README.md (3 puntos)

Inicialice la creación de un archivo `README.md` dentro del repositorio y mencione su nombre y el objetivo de la tarea 05 en formato markdown. Se espera que existan 3 commits:

1. Uno para la inicialización.
2. Uno para escribir su nombre dentro del archivo.
3. Uno para cuando se escriba el objetivo de la tarea.

Puede revisar el video explicativo o el final de la Clase05 para entender como realizar un commit[1].

[1]Notar que al ejecutar `push origin` se le pedirá hacer login en GitHub, si la terminal le está preguntando por una contraseña, deberán usar un token generado desde aquí.

Creación de un notebook (6 puntos)

A continuación, deberá crear un notebook para explorar el dataset de la flor Iris. Idealmente utilice una nueva carpeta dentro del repositorio.

Inicializando el notebook (2 puntos)

- Inicialice un notebook de exploración y agreguelo al repositorio.
- Una vez creado el notebook, utilice pandas para leer el siguiente csv: https://raw.githubusercontent.com/plotly/datasets/master/iris-data.csv
- Muestre inicialmente el tamaño (shape) del DataFrame creado, junto a las primeras 5 columnas. Realice los commits que considere pertinentes[2].

Describiendo el dataset (4 puntos)

Dentro del mismo notebook, describa (utilizando código y markdown):

- Qué columnas tiene el dataset y que representan.
- La existencia (o no) de valores nulos.
- Estadísticas de cada columna numérica (utilice `df.describe()`).
- Cuántas filas existen para cada clase de flor.

De forma similar a lo anterior, se espera que se vayan realizando los commits pertinentes para cada descripción.

Formato de Entrega

La entrega deberá ser únicamente el link a su repositorio, por ejemplo `https://github.com/mi_usuario/tarea-05`.

[2]No es necesario realizar un commit por cada tarea pequeña que se realiza