

# Tarea 1

## Git y red básica de clasificación

### Redes Neuronales Artificiales

José Ibáñez\*

\*Benemérita Universidad Autónoma de Puebla,  
Facultad de Ciencias Físico Matemáticas

12 de septiembre de 2022



#### Resumen

El presente servirá como un manual para aprender (y no olvidar) a usar Git en conjunto con GitHub, esto con el fin de implementar mi primera red neuronal artificial básica en un repositorio público (evidencia en la **Sección 3**).

## 1. Creación de repositorio en GitHub

Ya tengo una cuenta en esta plataforma; `joseiban`, fue usada en el año pasado para escribir código en Python sobre análisis de datos en colisiones de partículas.

He creado un repositorio público; `Redes-Neuronales-Artificiales-Otono-2022`, donde subiré las evidencias del progreso alcanzado en este curso, contiene un archivo `README.md` y una licencia GNU General Public License v3.0 tal y como el dr. Jorge recomienda.

---

\*`jose.ibanez@alumno.buap.mx`

## 2. Clonar el repositorio a Git

Mi ordenador tiene el sistema operativo Ubuntu 22.04.1 LTS (Jammy Jellyfish) y tiene el software de Git 2.34.1, con esto ya podemos empezar a trabajar, directamente desde la terminal de Ubuntu; creo una carpeta especial para clonar el repositorio de este curso y me dirijo hacia ella:

```
1 $ mkdir RNAotono2022 # CREAR UN DIRECTORIO LLAMADO RNAotono2022
2 $ cd RNAotono2022 # IR HACIA EL DIRECTORIO DE NOMBRE RNAotono2022
```

En la plataforma de GitHub copio el enlace del repositorio desde el botón **Code** y lo copio en la terminal con la instrucción `clone` de Git, otorgando el resultado:

```
1 $ git clone https://github.com/joseiban/Redes-Neuronales-Artificiales-
  Otono-2022.git # INSTRUCCIÓN clone PARA CLONAR EL REPOSITORIO DE GITHUB
  A GIT:
2 Clonando en 'Redes-Neuronales-Artificiales-Otono-2022'...
3 remote: Enumerating objects: 4, done.
4 remote: Counting objects: 100% (4/4), done.
5 remote: Compressing objects: 100% (4/4), done.
6 remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
7 Recibiendo objetos: 100% (4/4), 12.59 KiB | 644.00 KiB/s, listo.
```

Una vez copiado el repositorio me dirijo hacia él:

```
1 $ cd Redes-Neuronales-Artificiales-Otono-2022 # IR HACIA EL DIRECTORIO DE
  NOMBRE Redes-Neuronales-Artificiales-Otono-2022
```

Inspecciono el contenido del repositorio copiado para comprobar que es correcto, y en efecto lo es, pues contiene los archivos `README.md` y la licencia que mencioné en la **Sección 1**:

```
1 $ ls # OBSERVAR EL CONTENIDO DEL DIRECTORIO ACTUAL:
2 LICENSE README.md
```

Verifico que son los mismos archivos entrando a uno, mediante:

```
1 $ more README.md # INSTRUCCIÓN PARA MOSTRAR MÁS DEL ARCHIVO README.md:
2 # Redes-Neuronales-Artificiales-Otono-2022
3 Aquí se depositarán las evidencias del progreso alcanzado en la creación
  de redes neuronales artificiales.
```

En efecto, el contenido es el mismo. Crearé un archivo `.py` de prueba, usando el editor de textos predeterminado de mi sistema:

```
1 $ gedit prueba-1.py & # CREAR Y ABRIR PARA EDICIÓN EL ARCHIVO DE NOMBRE
  prueba-1.py
```

Al abrirse la hoja de texto escribo lo siguiente:

```
1 # Este es un primer código Python de prueba
2 import numpy as np
```

Guardo el archivo presionando CTRL+S y compruebo que esté correctamente guardado en el repositorio:

```
1 $ ls # OBSERVAR EL CONTENIDO DEL DIRECTORIO ACTUAL:
2 LICENSE  prueba-1.py  README.md
```

Ahí se encuentra. Iniciaré con los pasos para subir las actualizaciones del repositorio a GitHub:

```
1 $ git status # ESTADO ACTUAL DEL REPOSITORIO:
2 En la rama main
3 Tu rama está actualizada con 'origin/main'.
4
5 Archivos sin seguimiento:
6   (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
7   prueba-1.py
8
9 no hay nada agregado al commit pero hay archivos sin seguimiento presentes
   (usa "git add" para hacerles seguimiento)
```

Hago justamente lo que me indica la terminal; incluir el archivo de prueba creado al commit para poder subirlo a GitHub:

```
1 $ git add prueba-1.py # AÑADIR EL ARCHIVO prueba-1.py AL commit
```

Compruebo que el archivo ya se encuentre en seguimiento:

```
1 $ git status # ESTADO ACTUAL DEL REPOSITORIO:
2 En la rama main
3 Tu rama está actualizada con 'origin/main'.
4
5 Cambios a ser confirmados:
6   (usa "git restore --staged <archivo>..." para sacar del área de stage)
7   nuevos archivos: prueba-1.py
```

Una vez listo el commit guardo todo el progreso, lo que el dr. Jorge llama *tomar la foto*:

```
1 $ git commit -a # GUARDAR LOCALMENTE LOS CAMBIOS HECHOS EN EL REPOSITORIO
   CLONADO
```

En mi caso, pidió confirmación para la cuenta de GitHub, mediante el correo y la contraseña registrados en GitHub. Una vez hecho esto, se abre la interfaz del editor de textos GNU nano 6.2, aquí anoto la descripción de los cambios realizados:

```
1 He subido un primer archivo .py de prueba
2 # Por favor ingresa el mensaje del commit para tus cambios. Las
3 # líneas que comiencen con '#' serán ignoradas, y un mensaje
4 # vacío aborta el commit.
5 #
6 # En la rama main
7 # Tu rama está actualizada con 'origin/main'.
8 #
```

```

9 # Cambios a ser confirmados:
10 #     nuevos archivos: prueba-1.py
11 #

```

Guardo y cierro el archivo mediante CTRL+X y vuelvo a la terminal. Evalúo nuevamente la situación y obtengo:

```

1 $ git status # ESTADO ACTUAL DEL REPOSITORIO:
2 En la rama main
3 Tu rama está adelantada a 'origin/main' por 1 commit.
4 (usa "git push" para publicar tus commits locales)
5
6 nada para hacer commit, el árbol de trabajo está limpio

```

Esto indica que todos los cambios están guardados localmente y sólo haría falta publicar mis avances en GitHub; para esto requiero nombre de usuario y token personal de GitHub, que generé desde las Configuraciones de Desarrollador en la plataforma:

```

1 $ git push # INSTRUCCIÓN PARA SUBIR LOS CAMBIOS REALIZADOS EN GIT A LA
   PLATAFORMA GITHUB:
2 Username for 'https://github.com': joseiban # MI NOMBRE DE USUARIO
3 Password for 'https://joseiban@github.com': # MI TOKEN PERSONAL
4 Enumerando objetos: 4, listo.
5 Contando objetos: 100% (4/4), listo.
6 Compresión delta usando hasta 2 hilos
7 Comprimiendo objetos: 100% (3/3), listo.
8 Escribiendo objetos: 100% (3/3), 397 bytes | 397.00 KiB/s, listo.
9 Total 3 (delta 0), reusados 0 (delta 0), pack-reusados 0
10 To https://github.com/joseiban/Redes-Neuronales-Artificiales-Otono-2022.
   git
11 601a753..fff2ea6  main -> main

```

Con esto, el repositorio en GitHub debería estar actualizado con los cambios hechos en Git.

### 3. Implementación del código

Estando en el directorio clonado, lo primero será tener las bases de datos (mnist.pkl.gz y mnist\_loader.py); las descargo desde Microsoft Teams y las muevo manualmente hasta esta carpeta, ahora tiene lo siguiente:

```

1 $ ls # OBSERVAR EL CONTENIDO DEL DIRECTORIO ACTUAL:
2 Evidencias-de-progreso  mnist_loader.py  prueba-1.py
3 LICENSE                 mnist.pkl.gz     README.md

```

Ahora crearé un nuevo archivo .py para implementar la network:

```

1 $ gedit red_neuronal_basica_1.py & # CREAR Y ABRIR PARA EDICIÓN EL ARCHIVO
   DE NOMBRE red_neuronal_basica_1.py

```

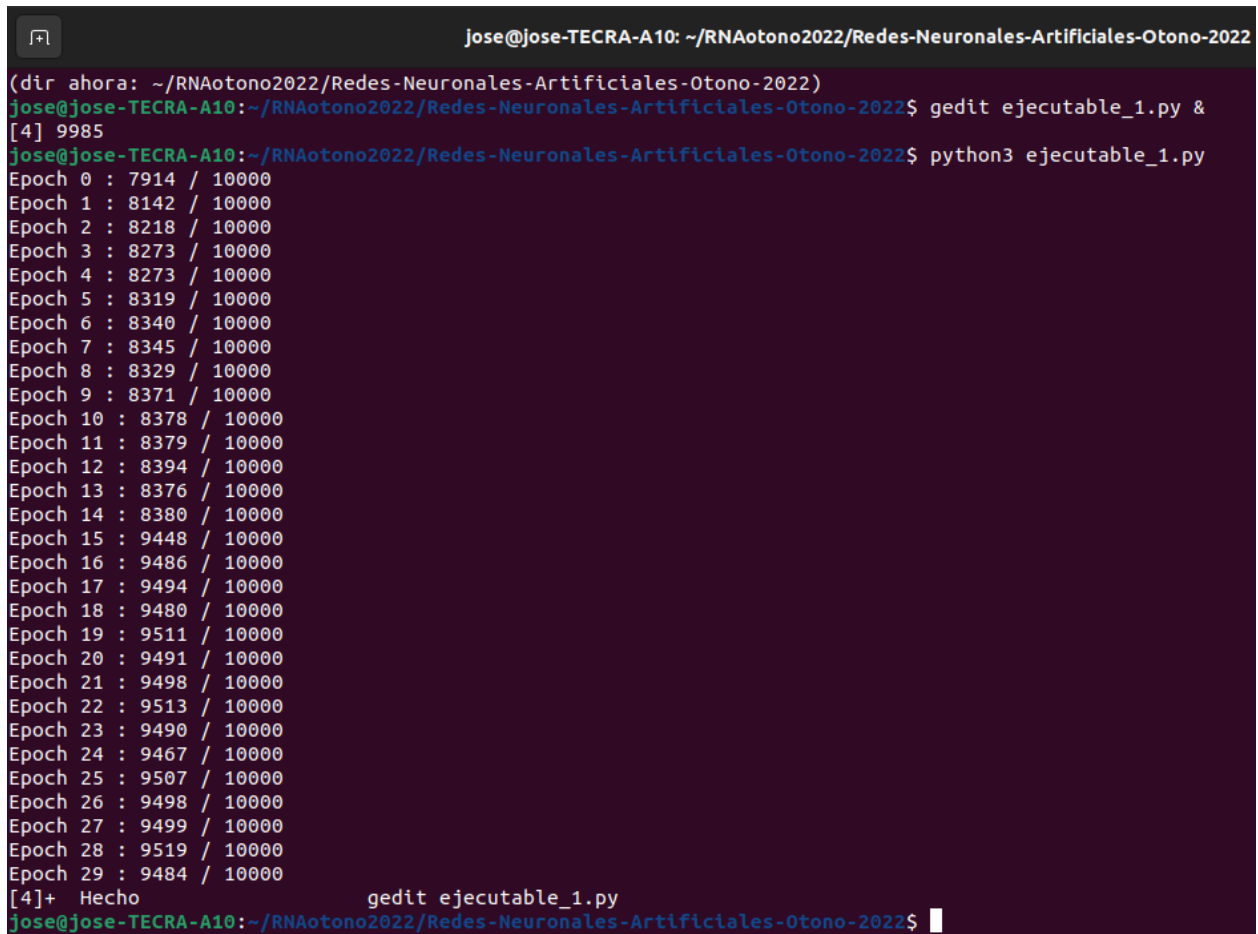
Al abrirse el editor de texto hago el código idéntico al del dr. Jorge, que ahora identifico como `red_neuronal_basica_1.py`. Creo un archivo ejecutable:

```
1 $ gedit ejecutable_1.py & # CREAR Y ABRIR PARA EDICIÓN EL ARCHIVO DE  
    NOMBRE ejecutable_1.py
```

En el que sí genero cambios; cambiar el antiguo nombre de la network por el nuevo, quedando como se ve en `ejecutable_1.py`. Ahora hago que Python 3.10.4 lo corra:

```
1 $ python3 ejecutable_1.py # EL SOFTWARE DE Python 3.10.4 CORRE EL ARCHIVO  
    ejecutable_1.py
```

y obtengo el resultado:



```
jose@jose-TECRA-A10: ~/RNAotono2022/Redes-Neuronales-Artificiales-Otono-2022  
(dir ahora: ~/RNAotono2022/Redes-Neuronales-Artificiales-Otono-2022)  
jose@jose-TECRA-A10:~/RNAotono2022/Redes-Neuronales-Artificiales-Otono-2022$ gedit ejecutable_1.py &  
[4] 9985  
jose@jose-TECRA-A10:~/RNAotono2022/Redes-Neuronales-Artificiales-Otono-2022$ python3 ejecutable_1.py  
Epoch 0 : 7914 / 10000  
Epoch 1 : 8142 / 10000  
Epoch 2 : 8218 / 10000  
Epoch 3 : 8273 / 10000  
Epoch 4 : 8273 / 10000  
Epoch 5 : 8319 / 10000  
Epoch 6 : 8340 / 10000  
Epoch 7 : 8345 / 10000  
Epoch 8 : 8329 / 10000  
Epoch 9 : 8371 / 10000  
Epoch 10 : 8378 / 10000  
Epoch 11 : 8379 / 10000  
Epoch 12 : 8394 / 10000  
Epoch 13 : 8376 / 10000  
Epoch 14 : 8380 / 10000  
Epoch 15 : 9448 / 10000  
Epoch 16 : 9486 / 10000  
Epoch 17 : 9494 / 10000  
Epoch 18 : 9480 / 10000  
Epoch 19 : 9511 / 10000  
Epoch 20 : 9491 / 10000  
Epoch 21 : 9498 / 10000  
Epoch 22 : 9513 / 10000  
Epoch 23 : 9490 / 10000  
Epoch 24 : 9467 / 10000  
Epoch 25 : 9507 / 10000  
Epoch 26 : 9498 / 10000  
Epoch 27 : 9499 / 10000  
Epoch 28 : 9519 / 10000  
Epoch 29 : 9484 / 10000  
[4]+ Hecho  
jose@jose-TECRA-A10:~/RNAotono2022/Redes-Neuronales-Artificiales-Otono-2022$ gedit ejecutable_1.py
```

Figura 3.1: Evidencia 1

Veo que esta red empezó con un porcentaje más bajo del que tuvo el dr. Jorge, ya que la de él empieza con un 91.01 % de eficacia y termina con un 95.21 %, mientras que yo empecé con un 79.14 % y terminé con un 95.19 %.

## 4. Subdirectorio de evidencias

Dentro del repositorio clonado, crearé un directorio donde guardaré mis evidencias, en mi caso yo dejaré todas mis evidencias en este PDF:

```
1 $ mkdir Evidencias-de-progreso # CREAR UN DIRECTORIO LLAMADO Evidencias-de
  -progreso
2 $ cd Evidencias-de-progreso # IR HACIA EL DIRECTORIO DE NOMBRE Evidencias-
  de-progreso
```

## 5. Subir el código a GitHub

Verifico el estado del repositorio:

```
1 $ git status # ESTADO ACTUAL DEL REPOSITORIO:
2 En la rama main
3 Tu rama está actualizada con 'origin/main'.
4
5 Archivos sin seguimiento:
6 (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
7 __pycache__/
8 ejecutable_1.py
9 mnist.pkl.gz
10 mnist_loader.py
11 red_neuronal_basica_1.py
12 red_prueba1.pkl
13
14 no hay nada agregado al commit pero hay archivos sin seguimiento presentes
   (usa "git add" para hacerles seguimiento)
```

No estoy seguro de que \_\_pycache\_\_/ sea necesaria pero la guardaré por si acaso y verifico el nuevo estado:

```
1 $ git add __pycache__/ ejecutable_1.py mnist.pkl.gz mnist_loader.py
   red_neuronal_basica_1.py red_prueba1.pkl # AÑADIR LOS ARCHIVOS
   __pycache__/ ejecutable_1.py mnist.pkl.gz mnist_loader.py
   red_neuronal_basica_1.py red_prueba1.pkl AL commit
2 $ git status # ESTADO ACTUAL DEL REPOSITORIO:
3 En la rama main
4 Tu rama está actualizada con 'origin/main'.
5
6 Cambios a ser confirmados:
7 (usa "git restore --staged <archivo>..." para sacar del área de stage)
8 nuevos archivos: __pycache__/mnist_loader.cpython-310.pyc
9 nuevos archivos: __pycache__/red_neuronal_basica_1.cpython-310.pyc
10 nuevos archivos: ejecutable_1.py
11 nuevos archivos: mnist.pkl.gz
12 nuevos archivos: mnist_loader.py
13 nuevos archivos: red_neuronal_basica_1.py
14 nuevos archivos: red_prueba1.pkl
```

Hago el commit de todo mediante `git commit -a` y se abre la interfaz de GNU nano 6.2, en donde escribo lo siguiente:

```
1 Se ha creado el primer ejemplo de una red neuronal básica
2 # Por favor ingresa el mensaje del commit para tus cambios. Las
3 # líneas que comiencen con '#' serán ignoradas, y un mensaje
4 # vacío aborta el commit.
5 #
6 # En la rama main
7 # Tu rama está actualizada con 'origin/main'.
8 #
9 # Cambios a ser confirmados:
10 #      nuevos archivos: __pycache__/mnist_loader.cpython-310.pyc
11 #      nuevos archivos: __pycache__/red_neuronal_basica_1.cpython-310.pyc
12 #      nuevos archivos: ejecutable_1.py
13 #      nuevos archivos: mnist.pkl.gz
14 #      nuevos archivos: mnist_loader.py
15 #      nuevos archivos: red_neuronal_basica_1.py
16 #      nuevos archivos: red_prueba1.pkl
17 #
```

El estado indica:

```
1 git status # ESTADO ACTUAL DEL REPOSITORIO:
2 En la rama main
3 Tu rama está adelantada a 'origin/main' por 1 commit.
4 (usa "git push" para publicar tus commits locales)
5
6 nada para hacer commit, el árbol de trabajo está limpio
```

Que ya podemos subir todos los cambios a GitHub. Cuando escribo `git push` me pide nuevamente la confirmación de mi usuario y finalmente se suben los archivos a la plataforma. No debí guardar todos los archivos creados de una sola vez, por culpa de eso todos tienen la misma descripción y, por alguna razón, la carpeta de evidencias no se sube, sospecho que esto se debe a que no contiene archivos. Esto se corregirá en la próxima versión.