Sprint

Salta, Argentina - 2022

SCI PY
LATIN AMERICA
SALTA 2022

$ whoami

- Juan Martín Loyola
  - Estudiante de doctorado de la Universidad Nacional de San Luis, Argentina.
  - Docente de la Universidad Nacional de San Luis.
  - Miembro del "Contributor Experience Team" de Scikit-learn.
  - Uno de los organizadores de las PyData Meetups San Luis.

¿Qué es un Sprint?

- Reunión de personas para contribuir a un proyecto de software.
- Público:
  - desarrolladores del proyecto;
  - personas que están comenzando a contribuir en proyectos de código libre.
- El objetivo es introducir a los participantes en los distintos pasos que deben seguir para comenzar a colaborar en un proyecto.
- Carácter colaborativo.

Pair programming

- Se alienta a los participantes a reunirse de a pares para trabajar.
- Técnica de pair programming “conductor-navegador”.
  - La conductora escribe el código;
  - mientras la navegadora observa, verifica o sugiere cambios.
  - Pueden intercambiar roles a medida que lo sientan necesario.
  - Es conveniente que la persona más experimentada sea la que tome el rol de navegadora, mientras la persona que está dando sus primeros pasos sea la que maneja el teclado y mouse (conductora). Esto garantiza que ninguno de los miembros del equipo se pierda durante el desarrollo.

Organización del Sprint

- 27/09
  - 10:00 - 10:30 horas → Presentación e introducción al Sprint.
  - 10:30 - 12:00 horas → Pre-sprint (configuración de ambiente).
- 28/09
  - 10:00 - 10:30 horas → Presentación e introducción al Sprint.
  - 10:30 - 13:00 horas → Sprint.
  - 13:00 - 14:00 horas → Pausa para comer.
  - 14:00 - 17:00 horas → Sprint.

Requisitos para aprovechar el Sprint

- Tener cuenta de GitHub.
- Conocimientos de Python.
- Tener algo de experiencia utilizando scikit-learn.
- Algo de familiaridad con Git.

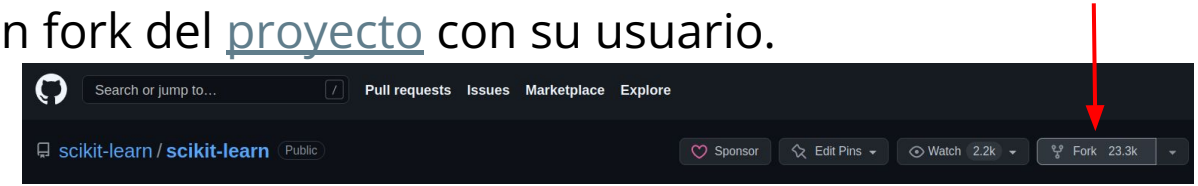
Preparación para el Sprint

- Crear cuenta de GitHub.
- Instalar Python. Para configurar el ambiente de desarrollo utilizaremos conda, por lo que recomendamos utilizar miniconda.
- Instalar Git.
- Instalar editor de texto (Visual Studio Code, Sublime Text, Atom, PyCharm, o el editor de preferencia).
- Mirar el video "Scikit-learn sprint instructions" (30 minutos). Transcripción del video en español. Notar que las instrucciones de instalación han quedado desactualizadas. Para instalar todo lo necesario para contribuir, siga esta guía en inglés.
- Mirar el video "Contributing to scikit-learn: An Example Pull Request (Reshama Shaikh)" (30 minutos). Transcripción del video en español.
- Mirar el video "Sprint Instructions for scikit-learn Vol 2" (14 minutos). Transcripción del video en español.
- Mirar el video "Mariatta Wijaya - Introduction to Sphinx Docs and reStructuredText" (15 minutos).
- **Opcional** - Mirar el video "Sphinx for Python Documentation Tutorial (Melissa Weber)".
- Leer el tutorial "Step-by-step guide to contributing on GitHub".

Clonar repositorio

Pasos para clonar el repositorio:

1. Hacer un fork del proyecto con su usuario.



2. Clonar el proyecto con `git clone git@github.com:USER/scikit-learn.git`. Reemplazar USER con su usuario de GitHub.
3. Dirigirse a la carpeta del proyecto con `cd scikit-learn`.
4. Agregar el repositorio original como repositorio remoto con `git remote add upstream https://github.com/scikit-learn/scikit-learn.git`.

Configurar ambiente de desarrollo

En la documentación de scikit-learn se encuentran los pasos para configurar el ambiente de desarrollo para distintos sistemas operativos:

- CONTRIBUTING,
- compilando el código de scikit-learn,
- paquetes necesarios para compilar la documentación.

Configurar ambiente de desarrollo (Linux + conda)

```
conda create -n sklearn-dev -c conda-forge python=3.9 numpy scipy matplotlib pytest
   cython ipykernel jupyter pytest-cov flake8 mypy

conda activate sklearn-dev

pip install black==22.3.0 pre-commit

pip install --no-build-isolation --editable .

pre-commit install



# Para poder compilar la documentación.
pip install sphinx sphinx-gallery numpydoc Pillow pandas scikit-image packaging
   seaborn sphinx-prompt sphinxext-opengraph
```

Nota sobre compilación de scikit-learn

Si se modifican archivos Cython (archivos que terminan en “.pyx” o “.pxd”), se tiene que re-compilar:

```
pip install --no-build-isolation -e .
```

Posibles Issues dónde contribuir

Los desarrolladores de scikit-learn han armado una lista de Issues para trabajar durante el Sprint.

La misma contiene Issues de distinta dificultad y en su mayoría pueden ser resueltos en el tiempo del Sprint.

Git/GitHub Workflow

Comandos normalmente utilizados para hacer un Pull Request:

```
# Partimos de nuestra rama "main"
git checkout main

# En caso de que nuestro fork no esté actualizado con lo último del repositorio hacemos:
git fetch upstream
git merge upstream main

# Creamos una rama para hacer todos los cambios
git checkout -b my_feature

# Hacemos los cambios para el Pull Request
git add modified_files

# En caso de que no quiera correr el pre-hook usar
# git commit --no-verify -m "Message"
git commit -m "Message"
```

Git/GitHub Workflow (agregar co-autores del commit)

Para indicar en GitHub que trabajamos con otras personas en un commit tenemos que colocar:

Notar que el mensaje comienza aquí

```
git commit -m "Message
>
>
Co-authored-by: name <name@example.com>
Co-authored-by: another-name <another-name@example.com>"
```

Líneas en blanco

Y termina acá

[Documentación de GitHub sobre creación de commits con varios autores](#).

Git/GitHub Workflow

```
# Controlar que no hayan cambios del upstream que tengamos que rebasear
# Puede ocurrir que mientras estamos trabajando, nuevos commits entren al
repositorio.
# En ese caso, tenemos que hacer un `rebase` para traer los nuevos cambios
# y llevar nuestros commits al final de la historia.

# Primero actualizamos los objetos y referencias del repositorio remoto
git fetch upstream

# En caso de tener miedo de que al realizar la operación, se borre nuestro
trabajo
# podemos crear una rama temporal
# git branch tmp my_feature

git rebase upstream/main  # esto puede fallar, en ese caso hacemos lo siguiente
# git rebase --abort
# git reset --hard tmp
```

Git/GitHub Workflow

```
# Finalmente, subimos los cambios a nuestro fork
git push origin my_feature

# Si todo salió bien, se puede eliminar el branch temporal
# git branch -D tmp

# Una vez que el pull request es mergeado se puede borrar el branch
git branch -d my_feature
```

Pull Request (PR) checklist

- Asignar un título útil al PR que resuma lo que hace su contribución. Algo como “Fix <ISSUE TITLE>” está bien. Pero ~~“Fix #<ISSUE NUMBER>”~~ no es claro.
- Asegurarse que el código pase los tests.
- Asegurarse que el código esté correctamente comentado y documentado.
- Controlar que la documentación generada se visualice correctamente. Para ello es necesario compilar la documentación.
- Asegurarse que todos los controles realizados por pre-commit hayan pasado.
- Ser paciente con los revisores del PR. La mayoría colabora con el proyecto en sus tiempos libres.

Comunicación con desarrolladores

En caso de tener alguna pregunta o estar trabado con algo, pueden comunicarse (en inglés) con los desarrolladores de scikit-learn durante el Sprint a través de los siguientes canales:

- Gitter
- Discord

Calendario scikit-learn

En caso de querer continuar participando de eventos organizados por el proyecto una vez terminado el Sprint, pueden seguir el calendario de scikit-learn.