CULTURA DIGITAL Y SOCIEDAD

Actividad Autónoma 4

Unidad 2: Herramientas y Metodologías en Ciencia de Datos

Tema 2: Buenas Prácticas en Programación para Ciencia de Datos

- Nombres: Yoryhi Isaac Cadena Acosta
- Fecha: 21/11/2025
- Carrera: Ciencia de Datos e Inteligencia Artificial
- Periodo académico: 2025 - 2S
- Semestre: Tercero

---

1. Introducción

En esta actividad se trabajó con un programa en Python que busca números primos desde el 1 a 100.000. Primero se analizó el código original, que era poco eficiente porque revisaba todos los posibles divisores de cada número y no aprovechaba librerías como NumPy, que permiten hacer operaciones mucho más rápidas, después se aplicaron varias técnicas de optimización para mejorar el tiempo de ejecución, junto con mediciones usando la biblioteca time y un análisis de rendimiento con cProfile.

El objetivo principal fue comparar el antes y después de optimizar el código y entender cómo cambios simples pueden mejorar mucho la velocidad de un programa.

---

2. Código original y problemas detectados

El código original verificaba si un número era primo probando todos los divisores desde 2 hasta el mismo número, lo cual genera muchísimas operaciones innecesarias, esto hace que el programa tarde más tiempo cuando trabaja con rangos grandes.

El tiempo promedio del código original fue aproximadamente: 24.60 segundos

Los principales problemas identificados fueron:

- Uso de un bucle demasiado largo para cada número.
- Falta de optimización matemática.
- Construcción de la lista de primos con bucles tradicionales en vez de list comprehensions.
- Falta de uso de librerías más eficientes como NumPy.

---

3. Código optimizado

Para mejorar el rendimiento del código se aplicaron estas técnicas:

a) Iterar solo hasta la raíz cuadrada del número

No es necesario revisar todos los divisores, solo hasta la raíz de n (√n), porque si un número tuviera un divisor mayor, el otro sería menor y ya habría sido detectado.

b) Uso de list comprehensions

Esto permite crear listas de forma más rápida y sólida que con bucles normales.

c) Uso de NumPy

NumPy permite aplicar operaciones de manera vectorizada, lo que reduce el tiempo de cálculo comparado con un bucle de Python.

Después de optimizar el programa, el tiempo de ejecución bajó aproximadamente a:

0.65 segundos

Esto representa una mejora enorme frente al código original.

---

4. Resultados

Para comparar ambas versiones se midieron los tiempos con:

- time.time() y cProfile para ver dónde se invierte el tiempo realmente

Tiempos obtenidos:

Versión	Tiempo aproximado
Original	22–25 segundos
Optimizada (NumPy)	0.4–0.6 segundos

La optimización redujo el tiempo más de un 98%.

Resultados de cProfile

cProfile mostró que:

En el código original, en la función primo() consumía casi todo el tiempo del programa.

En el código optimizado, NumPy realiza las divisiones en un solo bloque, reduciendo miles de llamadas repetidas.

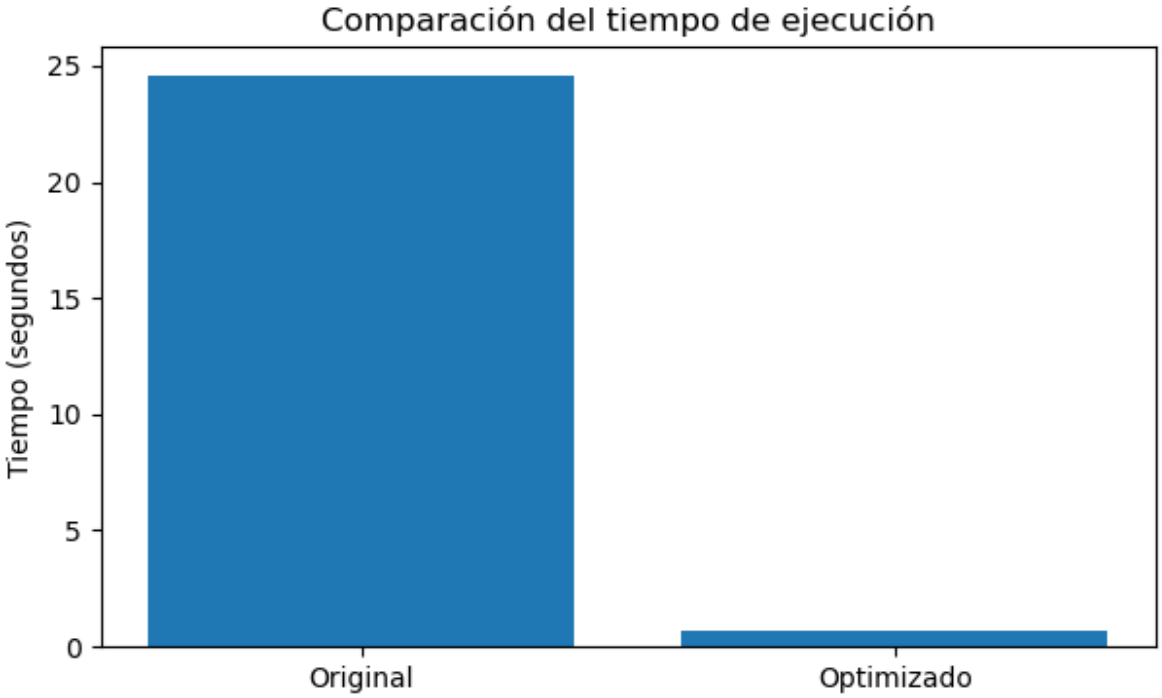
Esto confirma que el problema del código original era la cantidad de operaciones internas, y que NumPy resolvió eso con operaciones vectorizadas.

---

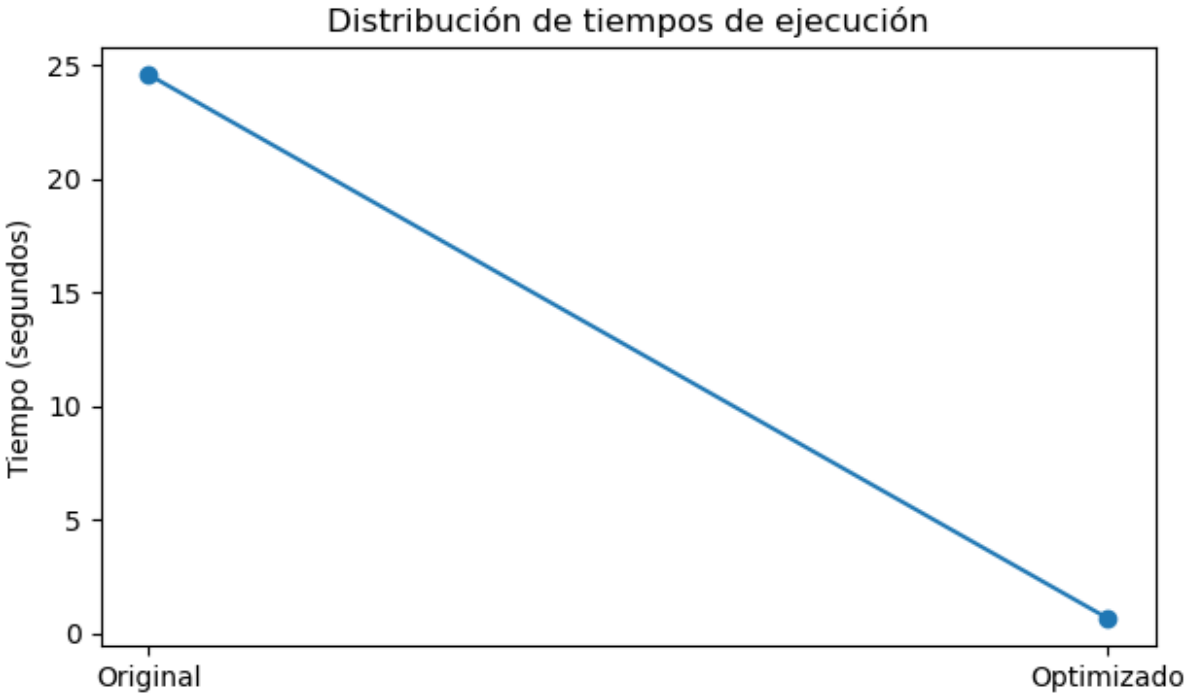
5. Gráficos generados

Se generaron dos gráficos usando Matplotlib:

- Comparación del tiempo de ejecución entre el código original y el optimizado



- Distribución de los tiempos, mostrando visualmente la diferencia entre ambas versiones.



Estos gráficos permiten apreciar claramente el impacto de las mejoras aplicadas.

---

6. Conclusiones y Recomendaciones

Conclusiones:

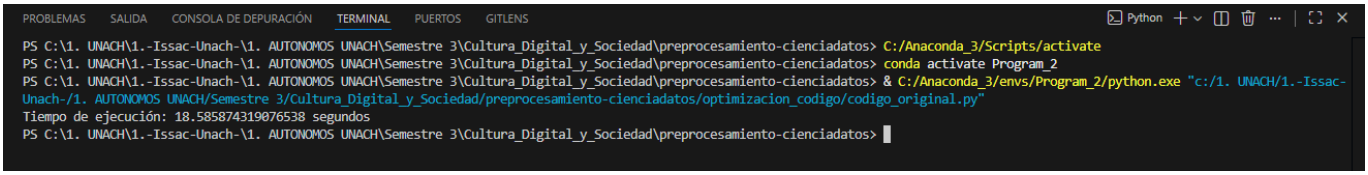
Optimizar un código puede reducir los tiempos de ejecución de manera significativa, técnicas simples como limitar el rango del bucle y usar list comprehensions generan mejoras importantes, el uso de librerías especializadas como NumPy aporta un mejor rendimiento gracias a la vectorización, el análisis con cProfile ayuda a identificar qué partes del programa consumen más tiempo y la optimización final redujo el tiempo de ejecución de 24 segundos a menos de 1 segundo, logrando una mejora notable en eficiencia.

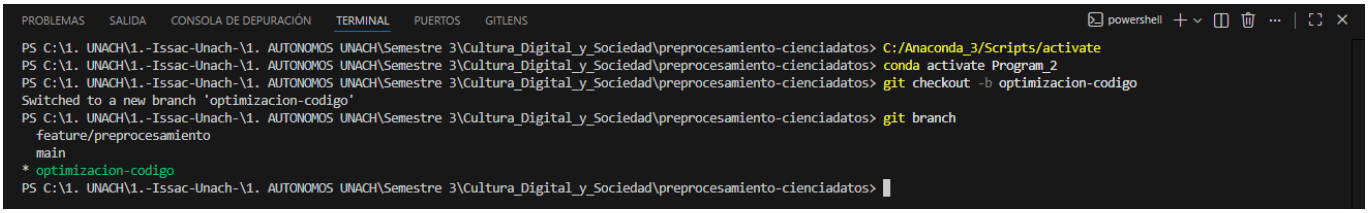
Recomendaciones:

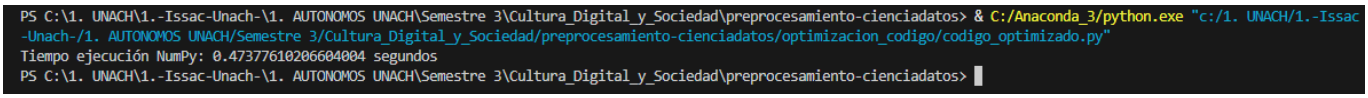
Usar funciones y herramientas que hagan el trabajo más rápido, como NumPy, cuando sea posible, revisar primero qué parte del código está tardando más antes de intentar mejorar todo, mantener el código ordenado en carpetas para que sea fácil de entender y modificar, evitar ciclos innecesarios y buscar métodos vectorizados o funciones más eficientes para reducir tiempo de ejecución y comparar siempre los tiempos antes y después de optimizar para confirmar que realmente mejoró.

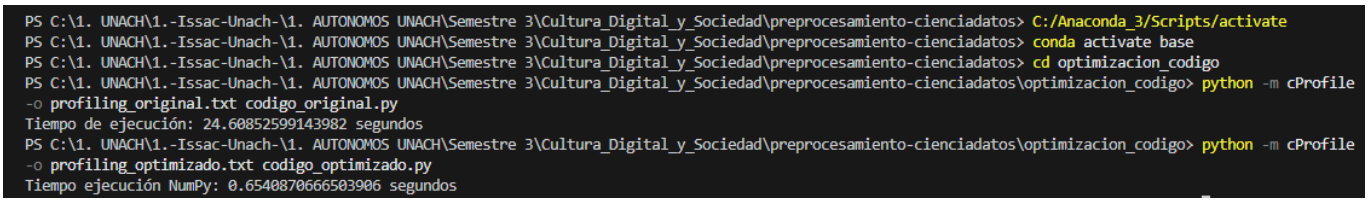
---

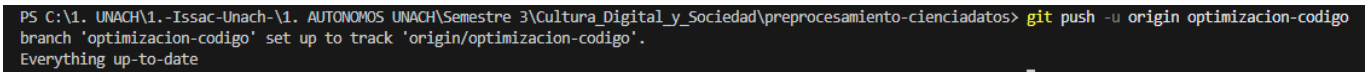
7. Anexos

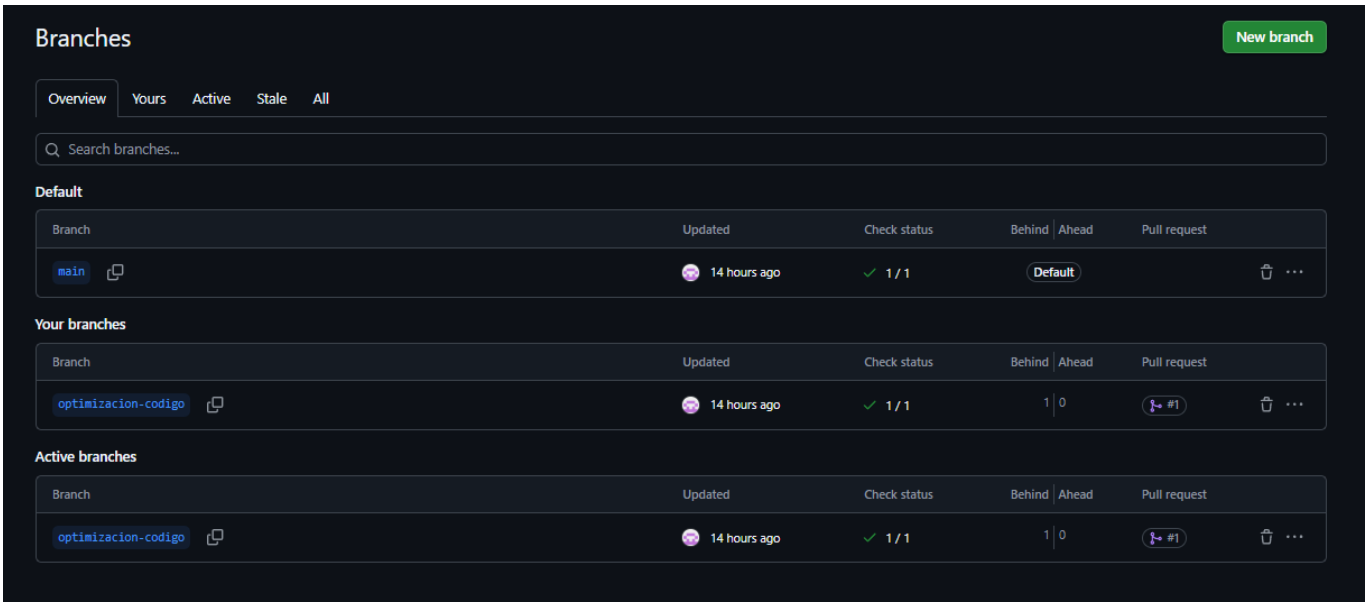


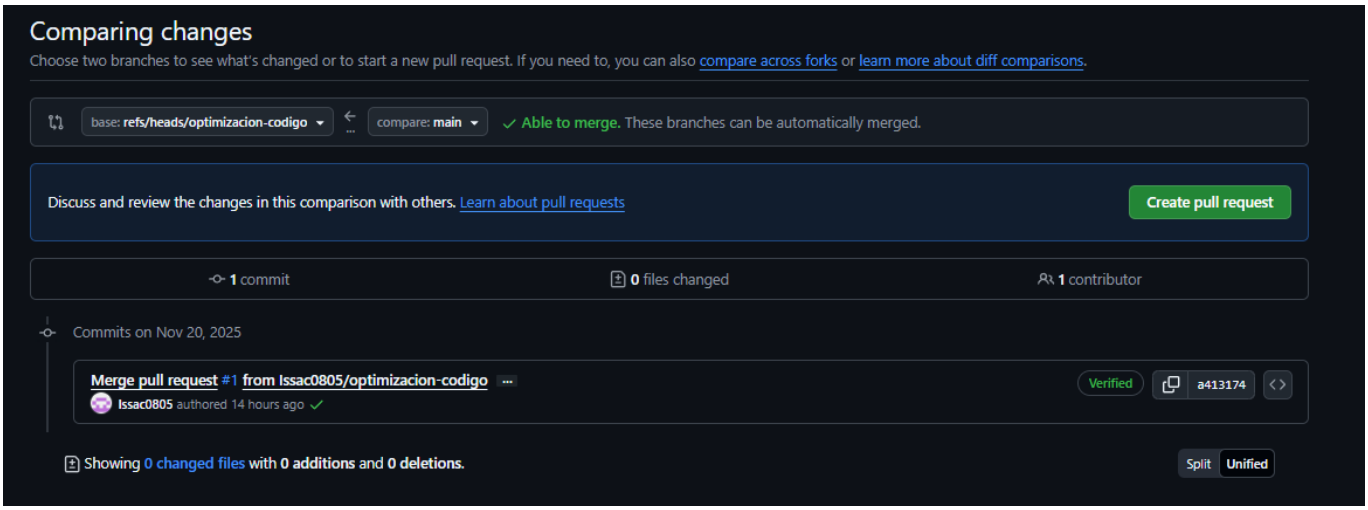


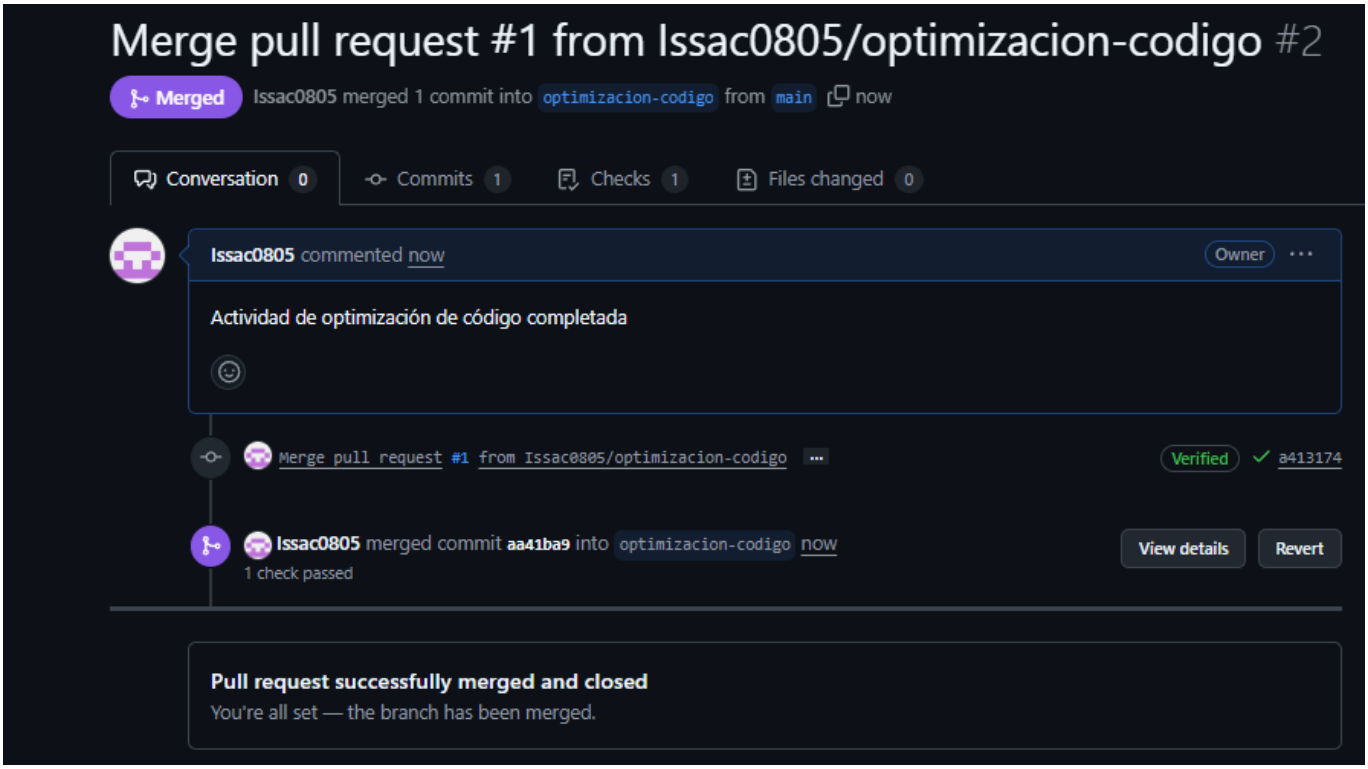


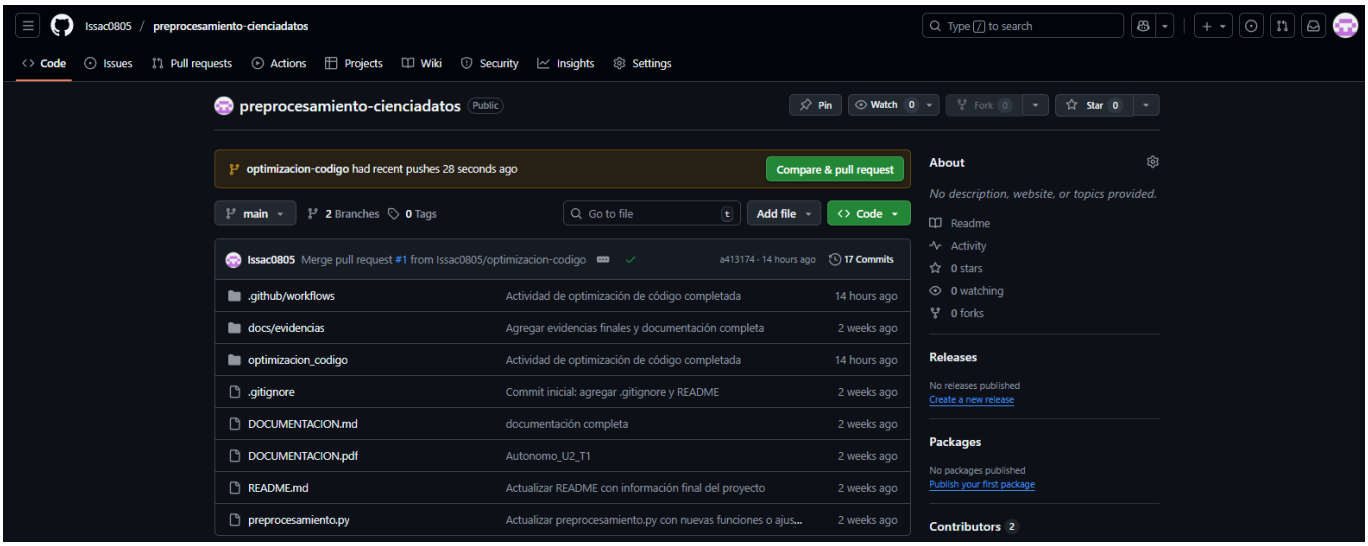





Branches
New branch
Overview
Yours
Active
Stale
All
Search branches...
Default
Branch
Updated
Check status
Behind
Ahead
Pull request
main
14 hours ago
1 / 1
Default
Your branches
optimizacion-codigo
14 hours ago
1 / 1
1
0
#1
Active branches
optimizacion-codigo
14 hours ago
1 / 1
1
0
#1


Comparing changes
Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.
base: refs/heads/optimizacion-codigo
compare: main
Able to merge. These branches can be automatically merged.
Discuss and review the changes in this comparison with others. Learn about pull requests
Create pull request
1 commit
0 files changed
1 contributor
Commits on Nov 20, 2025
Merge pull request #1 from Issac0805/optimizacion-codigo
Issac0805 authored 14 hours ago
Verified
a413174
Showing 0 changed files with 0 additions and 0 deletions.
Split
Unified


Merge pull request #1 from Issac0805/optimizacion-codigo #2
Merged
Issac0805 merged 1 commit into optimizacion-codigo from main now
Conversation 0
Commits 1
Checks 1
Files changed 0
Issac0805 commented now
Owner
Actividad de optimización de código completada
Merge pull request #1 from Issac0805/optimizacion-codigo
Verified
a413174
Issac0805 merged commit aa41ba9 into optimizacion-codigo now
1 check passed
View details
Revert
Pull request successfully merged and closed
You're all set — the branch has been merged.



8. Enlace del repositorio

https://github.com/Issac0805/preprocesamiento-cienciadatos