Classificação de Objetos Astronômicos com Algoritmo Random Forest: um comparativo entre implementações single e multi-threaded

Pedro Montuani

Introdução

- Humanidade vem produzindo um volume massivo de dados (aumentando a cada ano)
- Isso se reflete em diversas áreas humanas, incluindo na ciência

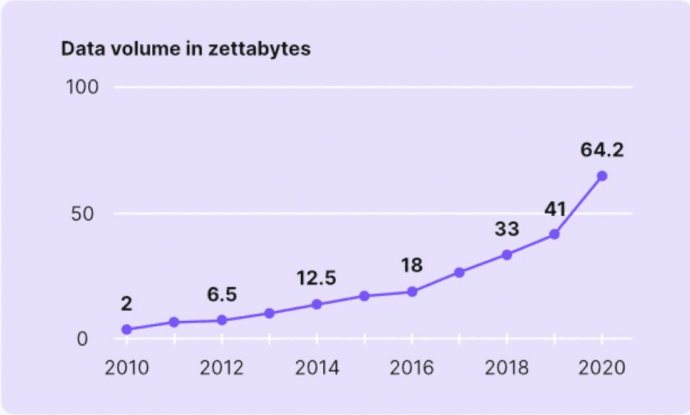


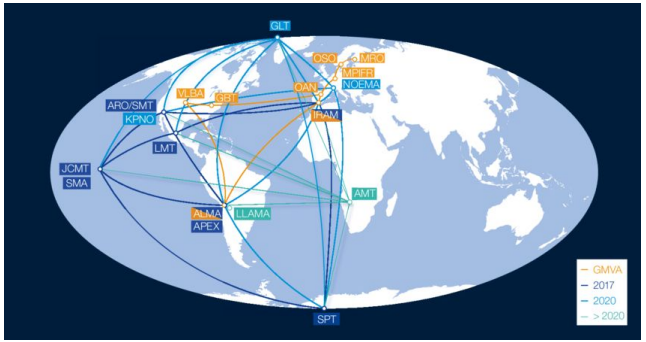
Fig. 01: volume de dados (ZB) produzidos na última década

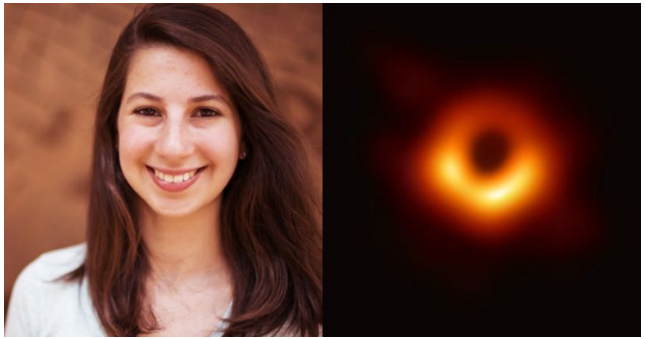
Introdução

- Astronomia: instrumentos modernos, novas técnicas e tecnologias -> grande volume de dados
- Necessidade de metodologias eficientes para lidar com essa quantidade de dados
- Machine learning, algoritmos de classificação, reconhecimento de padrões, simulações de rotas e órbitas

Introdução

- Projeto CHIRP (Continuous High-resolution Image Reconstruction using Patch priors)
- Sincronizou vários telescópios ao redor do planeta para simular uma lente gigante
- Utilizou algoritmos de classificação para diferenciar buracos negros de outros objetos
- Utilizou algoritmos de combinação reconstrução de imagem



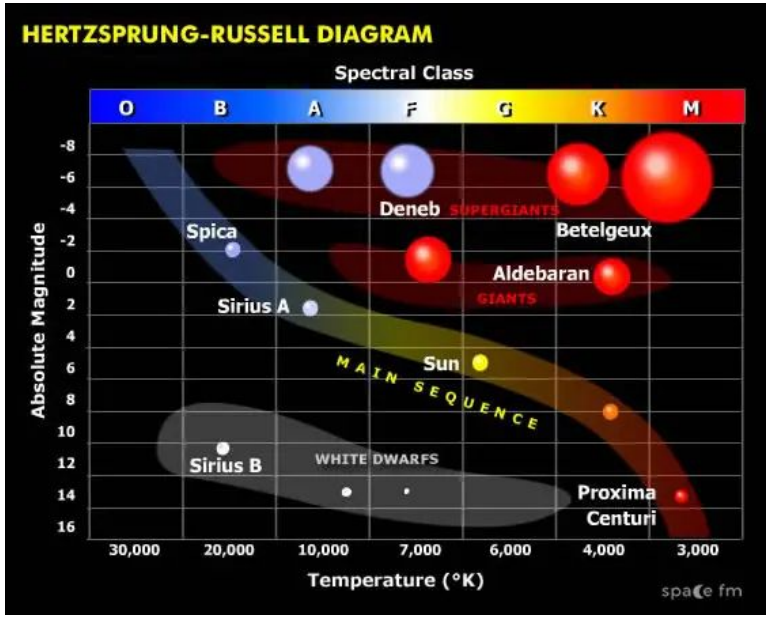


Figs. 02 e 03: Event Horizon Telescope, Katie Bouman e a primeira imagem de um buraco negro

Objetivos

Objetivos

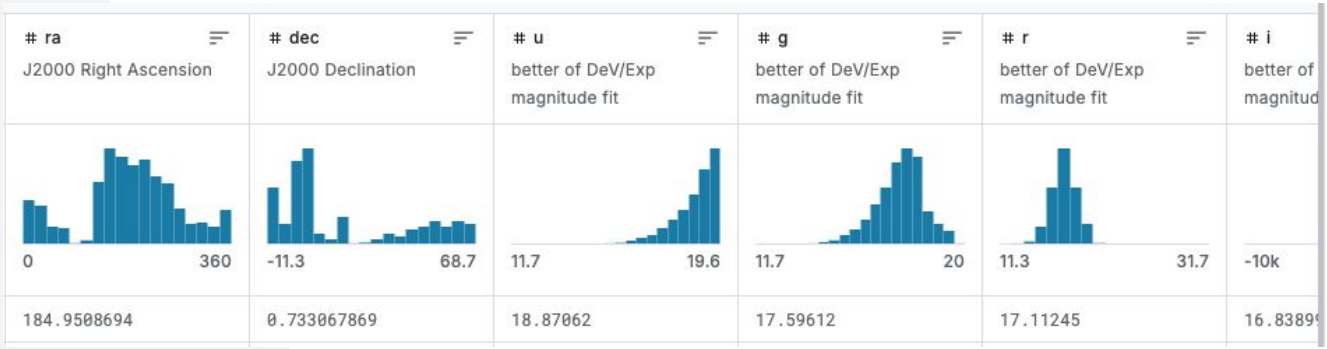
- Aplicar implementação de Random Forest em dataset de objetos astronômicos para classificação
- Implementar uma solução *multi-thread* (OMP)
- Comparar sklearn, C++ single, e C++ multi-thread



Metodologia

Metodologia

- Dataset -> Sloan Digital Sky Survey - DR18
  - CSV com 10.000 entradas de 43 *features*
  - Objetos classificados em STAR, GALAXY e QSO (*Quasi-Stellar Object* ou Quasar)



# ra J2000 Right Ascension	# dec J2000 Declination	# u better of DeV/Exp magnitude fit	# g better of DeV/Exp magnitude fit	# r better of DeV/Exp magnitude fit	# i better of magnitud
0 360	-11.3 68.7	11.7 19.6	11.7 20	11.3 31.7	-10k
184.9508694	0.733067869	18.87062	17.59612	17.11245	16.8389

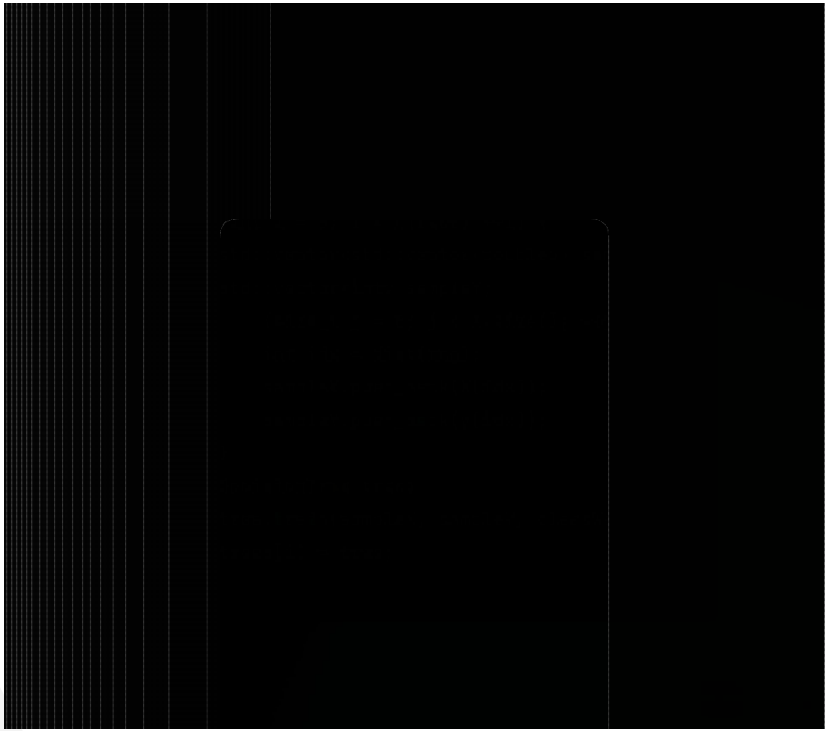
Metodologia

- Implementação
  - Python
    - Pré-processamento dos dados
    - Implementação do Sklearn como referência
    - Chamada em módulos C++
    - Avaliação

Metodologia

- Implementação
  - C++
    - Implementação de módulo para Random Forest
    - Modificação do módulo para paralelização com OMP

Metodologia

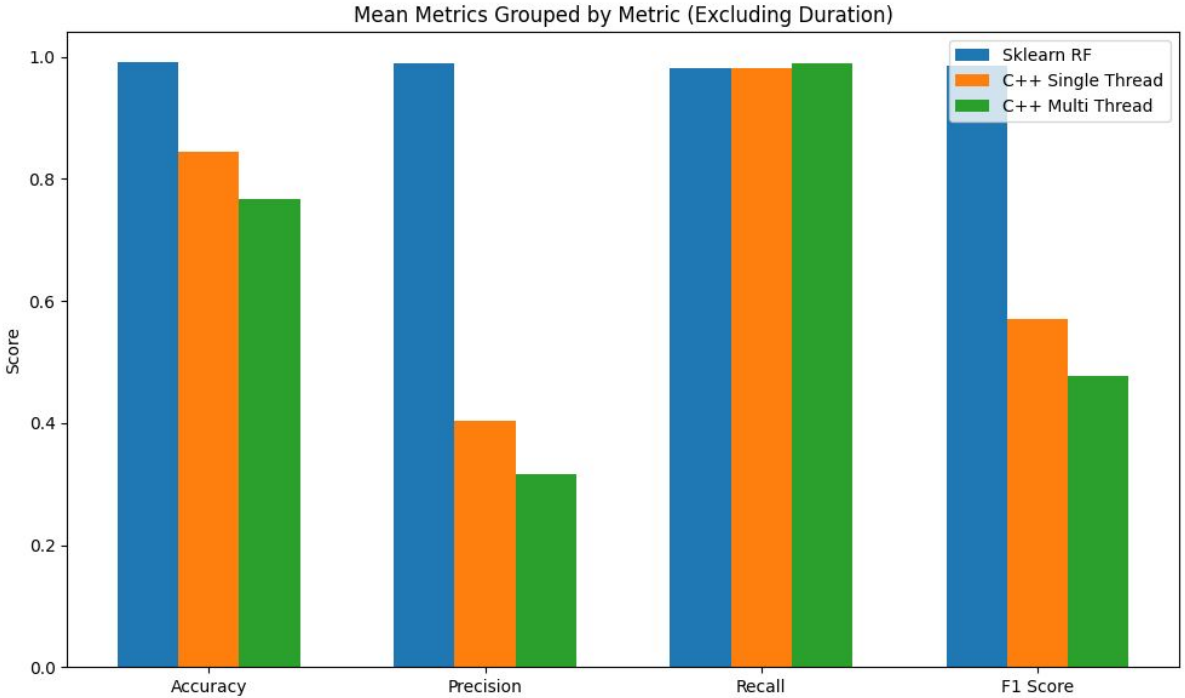


Metodologia

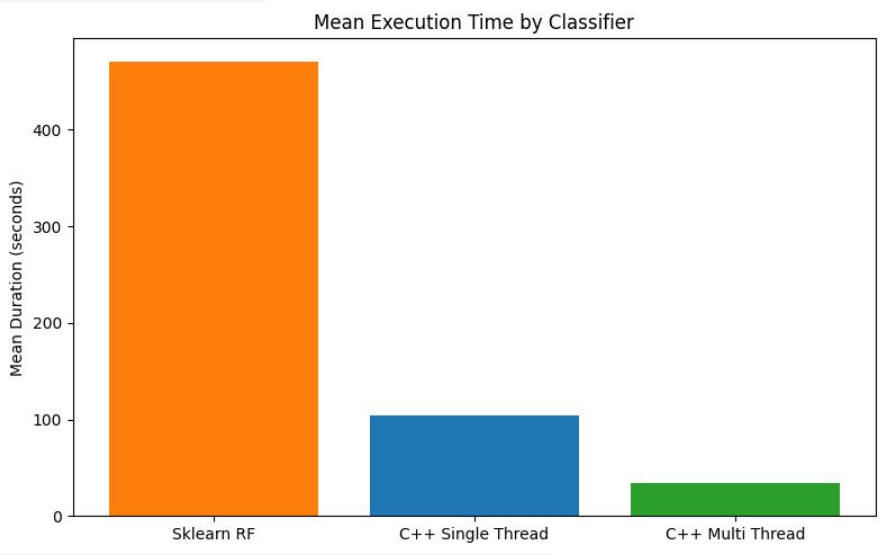
- Testes
  - Treinamento com 8000 entradas
  - Testes com 2000 entradas
  - 10x execuções -> média aritmética

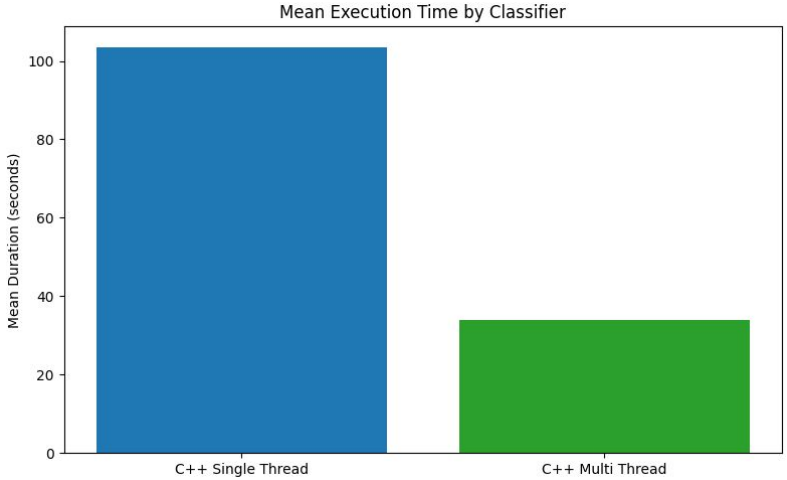
Resultados

Accuracy, precision, recall, f1-score e tempo de execução

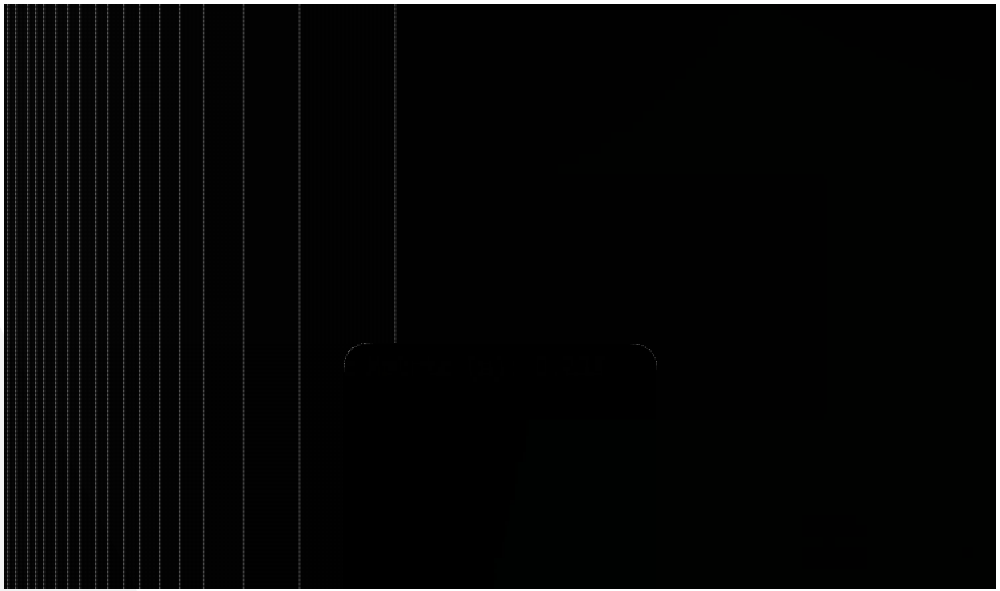


Speedup, eficiência, fração e métrica de Karp-Flatt





Speedup, eficiência, fração e métrica de Karp-Flatt



Considerações

Metodologia

- Trouxe um ganho considerável de performance no quesito tempo
- Em compensação, reduziu a acurácia e precisão
- Não necessariamente relacionado a paralelização, mas sim a implementação escolhida

Metodologia

- Baixa eficiência e valor baixo para métrica de Karp-Flatt indicam utilização ineficiente do paralelismo
  - Núcleos de processamento mal aproveitados
  - Overhead
- A paralelização é promissora, e, aplicada corretamente, pode trazer melhorias significativas aos algoritmos

Alguma dúvida?