Projeto 2 Astrometria

André Almeida Trovello

Estudo de regiões de formação estelar (Taurus) com métodos de Machine Learning

1. Query

Importando Bibliotecas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from astroquery.gaia import Gaia
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score, silhouette_samples
from astropy.table import Table
```

Criando Dataframe

```
data = pd.read_csv('galli_2019_table1.csv')
main_table = pd.DataFrame(data)
```

Tabela do Vizier

```
# Garanta que a variável 'main_table' exista.
# Se você estiver rodando o script do zero, você precisa
# carregar o arquivo CSV que você salvou do VizieR.
try:
    # Se você já tiver 'main_table' na memória (ex: num notebook), pule
    main_table
    print("Variável 'main_table' já existe na memória.")
except NameError:
    try:
        # Tente carregar do CSV
        # Substitua 'galli_2019_table1.csv' pelo nome que você usou
        main_table = Table.read('galli_2019_table1.csv')
        print("Arquivo 'galli_2019_table1.csv' carregado com sucesso.")
    except FileNotFoundError:
        print("Erro: A variável 'main_table' não foi encontrada na memória.")
        print("E o arquivo 'galli_2019_table1.csv' não \
            foi encontrado no disco.")
        print("Por favor, rode o script do VizieR \
            primeiro ou verifique o nome do arquivo CSV.")
        raise
```

```
Variável 'main_table' já existe na memória.
```

Limpar os IDs

Remove “Gaia DR2” dos nomes dos objetos mantendo apenas seus IDs

```
print("Iniciando a limpeza dos IDs...")
try:
    dr2_id_strings = main_table['GaiaDR2'].tolist()

    dr2_ids_cleaned_list = []
    ids_pulados = 0

    for s in dr2_id_strings:
        try:
            id_str = s.split(' ')[-1]
            numeric_id = np.int64(id_str)
            dr2_ids_cleaned_list.append(numeric_id)
        except (AttributeError, IndexError, ValueError):
            ids_pulados += 1
```

```
    print(f"Encontrada e limpa a lista de \
         {len(dr2_ids_cleaned_list)} IDs (DR2) válidos.")
    if ids_pulados > 0:
        print(f"({ids_pulados} linhas foram puladas por não \
            terem um ID válido)")

except (KeyError) as e:
    print(f"Erro: A coluna 'GaiaDR2' não foi encontrada na 'main_table'.")
    raise
```

```
Iniciando a limpeza dos IDs...
Encontrada e limpa a lista de          458 IDs (DR2) válidos.
(61 linhas foram puladas por não             terem um ID válido)
```

Cria a tabela para upload e a query organizada

```
upload_table = Table({'dr2_source_id_list': dr2_ids_cleaned_list})

# 1. Começa com a sua tabela (user_table)
# 2. Usa o nome correto da tabela (gaiadr3.dr2_neighbourhood)
query_dr3 = """
SELECT
    dr3.source_id, dr3.ra, dr3.dec, dr3.parallax, dr3.pmra,
    dr3.pmdec, xmatch.angular_distance,
    dr3.l, dr3.b, xmatch.magnitude_difference, dr3.ruwe
FROM
    tap_upload.my_table AS user_table
JOIN
    gaiadr3.dr2_neighbourhood AS xmatch
    ON user_table.dr2_source_id_list = xmatch.dr2_source_id
JOIN
    gaiadr3.gaia_source AS dr3
    ON xmatch.dr3_source_id = dr3.source_id
"""
```

Executa a busca

```python
print("Iniciando a busca no Gaia DR3 (com query otimizada)...")

try:
    job = Gaia.launch_job_async(
        query=query_dr3,
        upload_resource=upload_table,
        upload_table_name="my_table",
        verbose=True  # Adiciona mais informações de debug
    )

    results_dr3_members = job.get_results()

    print(f"\nBusca concluída!")
    print(f"Foram encontrados dados no DR3 para {len(results_dr3_members)} \
        das {len(dr2_ids_cleaned_list)} estrelas.")

    print("\n--- 5 primeiras linhas dos membros de Taurus (dados do DR3) ---")
    print(results_dr3_members.to_pandas().head())

    # Salve os resultados se desejar
    results_dr3_members.write('taurus_membros_dr3.csv',
        format='csv', overwrite=True)

except Exception as e:
    print(f"\nOcorreu um erro durante a busca no Gaia:")
    print(e)
```

```
Iniciando a busca no Gaia DR3 (com query otimizada)...
Launched query: '
SELECT
    dr3.source_id, dr3.ra, dr3.dec, dr3.parallax, dr3.pmra,
    dr3.pmdec, xmatch.angular_distance,
    dr3.l, dr3.b, xmatch.magnitude_difference, dr3.ruwe
FROM
    tap_upload.my_table AS user_table
JOIN
    gaiadr3.dr2_neighbourhood AS xmatch
    ON user_table.dr2_source_id_list = xmatch.dr2_source_id
JOIN
    gaiadr3.gaia_source AS dr3
    ON xmatch.dr3_source_id = dr3.source_id
```

```
'
------>https
host = gea.esac.esa.int:443
context = /tap-server/tap/async
Content-type = multipart/form-data; boundary====1762552075551===
303 303
[('Date', 'Fri, 07 Nov 2025 21:47:56 GMT'), ('Server', 'Apache/2.4.6 (SLES Expanded Support p
fips mod_jk/1.2.43'), ('Set-Cookie', 'JSESSIONID=80BA87C1ECD117FEC8645096EA715381; Path=/tap-
server; Secure; HttpOnly'), ('Location', 'https://gea.esac.esa.int/tap-
server/tap/async/17625520763930'), ('Cache-Control', 'no-cache, no-store, max-
age=0, must-revalidate'), ('Pragma', 'no-cache'), ('Expires', '0'), ('X-
XSS-Protection', '1; mode=block'), ('X-Frame-Options', 'SAMEORIGIN'), ('X-
Content-Type-Options', 'nosniff'), ('Transfer-Encoding', 'chunked'), ('Content-
Type', 'text/plain;charset=ISO-8859-1')]
job 17625520763930, at: https://gea.esac.esa.int/tap-server/tap/async/17625520763930
Retrieving async. results...
INFO: Query finished. [astroquery.utils.tap.core]

Busca concluída!
Foram encontrados dados no DR3 para 499         das 458 estrelas.

--- 5 primeiras linhas dos membros de Taurus (dados do DR3) ---
            source_id         ra        dec  parallax       pmra      pmdec  \
0  162535413750345856  60.955653  26.181023  6.758039  20.853242 -30.277735
1  162535413754166528  60.955305  26.180727  7.803457  20.133753 -27.911397
2  162541942104406784  60.958322  26.343862  6.971044  14.435075 -19.268191
3  162535345034688768  60.961924  26.181260  7.731206  19.547795 -30.009057
4   53092775104124288  61.164068  21.971754  8.147688   3.802709 -15.247629

   angular_distance           l          b  magnitude_difference       ruwe
0          5.860794  168.015225 -19.404779              0.033466  21.986296
1       1542.932129  168.015210 -19.405209              0.448453   1.159977
2          0.180421  167.895871 -19.287135             -0.025953   1.049943
3          0.095608  168.019296 -19.400658             -0.016413   2.134250
4          0.127694  171.356321 -22.234012             -0.015548   5.566028
```

2. Clustering

Carregando tabelas

```
data = pd.read_csv('taurus_membros_dr3.csv')
df = pd.DataFrame(data)
#X = df[['ra', 'dec', 'parallax', 'pmra', 'pmdec']]
#print(X['ra'])
#print(df)
# 1. Calcule as contagens de cada source_id
counts = df['source_id'].value_counts()

# 2. Use .map() para criar uma nova série do mesmo tamanho do df,
#    onde cada valor é a contagem do seu respectivo source_id.
#    Depois, compare com 2 para criar o filtro correto.
full_table = df[df['source_id'].map(counts) == 2]

full_table
```

	source_id	ra	dec	parallax	pmra	pmdec	angular_distance
34	163179006011625088	63.717943	28.099839	7.486891	8.195294	-22.940540	0.256296
35	163179006011625216	63.718513	28.099846	7.183761	9.408134	-24.033823	1809.979248
36	163179006011625088	63.717943	28.099839	7.486891	8.195294	-22.940540	1810.137573
37	163179006011625216	63.718513	28.099846	7.183761	9.408134	-24.033823	0.157825
98	152416436443721728	65.289006	27.843380	7.676996	7.989072	-26.709809	0.740511
99	152416436441091584	65.288916	27.843576	NaN	NaN	NaN	749.300537
100	152416436443721728	65.289006	27.843380	7.676996	7.989072	-26.709809	762.492981
101	152416436441091584	65.288916	27.843576	NaN	NaN	NaN	13.096863
159	152180827420224128	67.177628	27.234390	7.770386	7.754825	-25.039691	0.163206
160	152180831716415488	67.177664	27.234215	7.105773	9.080319	-26.201142	638.086182
161	152180827420224128	67.177628	27.234390	7.770386	7.754825	-25.039691	638.122437
162	152180831716415488	67.177664	27.234215	7.105773	9.080319	-26.201142	0.387675
188	146361013591547776	67.623567	24.445747	NaN	NaN	NaN	24.795189
189	146361013590374144	67.623320	24.445807	6.368969	8.908057	-24.076122	825.430115
190	146361013591547776	67.623567	24.445747	NaN	NaN	NaN	838.572693
191	146361013590374144	67.623320	24.445807	6.368969	8.908057	-24.076122	8.087403
195	145921994918655488	67.709547	23.002367	8.029621	11.949303	-14.773370	0.341585
196	145921999215653632	67.709932	23.002332	6.827561	10.848324	-16.615883	1280.887939
197	145921994918655488	67.709547	23.002367	8.029621	11.949303	-14.773370	1280.096558
198	145921999215653632	67.709932	23.002332	6.827561	10.848324	-16.615883	1.260170

	source_id	ra	dec	parallax	pmra	pmdec	angular_distance
231	3314132593934981248	68.126512	17.524747	6.752014	12.825682	-19.916297	1453.149902
232	3314132593936245248	68.126214	17.525034	6.873723	13.179530	-20.461011	0.183572
234	3314132593934981248	68.126512	17.524747	6.752014	12.825682	-19.916297	0.224493
235	3314132593936245248	68.126214	17.525034	6.873723	13.179530	-20.461011	1453.461304
312	3313386605360382720	68.982060	17.127651	6.948271	11.938078	-19.025502	1881.099121
313	3313386609655429248	68.981703	17.127255	6.674840	13.216349	-19.185871	0.187654
314	3313386605360382720	68.982060	17.127651	6.948271	11.938078	-19.025502	0.238448
315	3313386609655429248	68.981703	17.127255	6.674840	13.216349	-19.185871	1881.066895
353	148378033311467904	69.837256	25.750439	NaN	NaN	NaN	20.975664
354	148378033312276864	69.837141	25.750508	NaN	NaN	NaN	432.957581
355	148378033311467904	69.837256	25.750439	NaN	NaN	NaN	445.195953
356	148378033312276864	69.837141	25.750508	NaN	NaN	NaN	4.853233
362	148386898124771328	70.007363	25.941406	NaN	NaN	NaN	556.125671
363	148386898125118464	70.007242	25.941323	NaN	NaN	NaN	76.230293
364	148386898124771328	70.007363	25.941406	NaN	NaN	NaN	10.010338
365	148386898125118464	70.007242	25.941323	NaN	NaN	NaN	498.875671
454	155744074024631296	74.714207	28.523221	6.877693	6.051624	-29.397043	0.249245
455	155744074023050368	74.714590	28.523333	6.975570	6.545382	-31.430755	1276.910889
456	155744074024631296	74.714207	28.523221	6.877693	6.051624	-29.397043	1277.517456
457	155744074023050368	74.714590	28.523333	6.975570	6.545382	-31.430755	0.885773
479	156430822114424576	76.956081	30.401205	6.406249	2.640643	-27.397583	0.205192
480	156430817820015232	76.956544	30.401316	5.454096	4.906034	-24.577535	1491.435303
481	156430822114424576	76.956081	30.401205	6.406249	2.640643	-27.397583	1488.743896
482	156430817820015232	76.956544	30.401316	5.454096	4.906034	-24.577535	2.974982
486	3419115128091798272	77.034043	24.454042	NaN	NaN	NaN	26.817190
487	3419115132386033280	77.034161	24.454282	NaN	NaN	NaN	953.412903
488	3419115128091798272	77.034043	24.454042	NaN	NaN	NaN	933.809082
489	3419115132386033280	77.034161	24.454282	NaN	NaN	NaN	24.234728
495	3415706130944329216	78.114951	22.896915	5.714827	5.981752	-18.593437	6.100539
496	3415706130945884416	78.115074	22.897052	NaN	NaN	NaN	635.710510
497	3415706130944329216	78.114951	22.896915	5.714827	5.981752	-18.593437	643.074097
498	3415706130945884416	78.115074	22.897052	NaN	NaN	NaN	13.766609

```
df_filtered = df.sort_values(
    by=[
        'source_id',              # Agrupamento/Manutenção de Unicidade
        'angular_distance',       # 1º Critério: Mais importante (ordenação)
        'magnitude_difference'    # 2º Critério: Desempate (ordenação)
    ],
    ascending=True # Queremos o MÍNIMO (menor) de cada critério
```

```
).drop_duplicates(
    subset=['source_id'], # Coluna que define o que é 'único'
    keep='first'          # Mantém a linha que ficou no topo após a ordenação
)

print(df_filtered)
```

```
              source_id         ra        dec  parallax       pmra  \
65     45787585487165184  64.589504  16.979678  5.526606   5.315014
77     47316005432912256  64.715445  17.387881  8.223586   2.935037
96     47446336216710784  65.219760  17.778141  8.917751   3.833250
113    48192969034959232  65.497688  19.535056  6.890083  11.681591
116    48193213849576320  65.520708  19.580054  6.860571   6.973500
..                   ...        ...        ...       ...        ...
467  3419944920068400000  75.666107  24.992596  6.470762   5.246694
470  3419962546614118144  76.172500  25.165087  5.916953   2.379331
473  3419989892674105088  76.345713  25.525480  6.039720   2.918324
472  3419989896965813376  76.345092  25.525148  5.711849   2.321051
469  3420008412568724736  75.777510  25.388703  5.988247   2.991267

         pmdec  angular_distance          l          b  magnitude_difference  \
65   -9.780237          0.060572  177.715097 -23.206040              0.003076
77  -13.236989          0.136056  177.463037 -22.849257             -0.003348
96  -14.294179          0.198465  177.479749 -22.230236             -0.027531
113 -14.340750          0.201436  176.229789 -20.886831              0.041192
116 -12.281643          0.095923  176.208493 -20.841012             -0.024622
..         ...               ...         ...        ...                   ...
467 -26.609004          0.138069  177.913014 -10.135820             -0.028973
470 -16.979275          0.100264  178.047958  -9.664181             -0.036107
473 -18.287713          0.299688  177.846905  -9.324968              0.150270
472 -17.898328          0.087135  177.846841  -9.325616             -0.383138
469 -17.303800          0.226074  177.650421  -9.819310             -0.001138

         ruwe
65   0.892754
77   0.986649
96   1.436053
113  2.295600
116  1.343759
..        ...
467  1.070453
470  1.211074
```

```
473  0.872804
472  3.593245
469  1.168730

[473 rows x 11 columns]
```

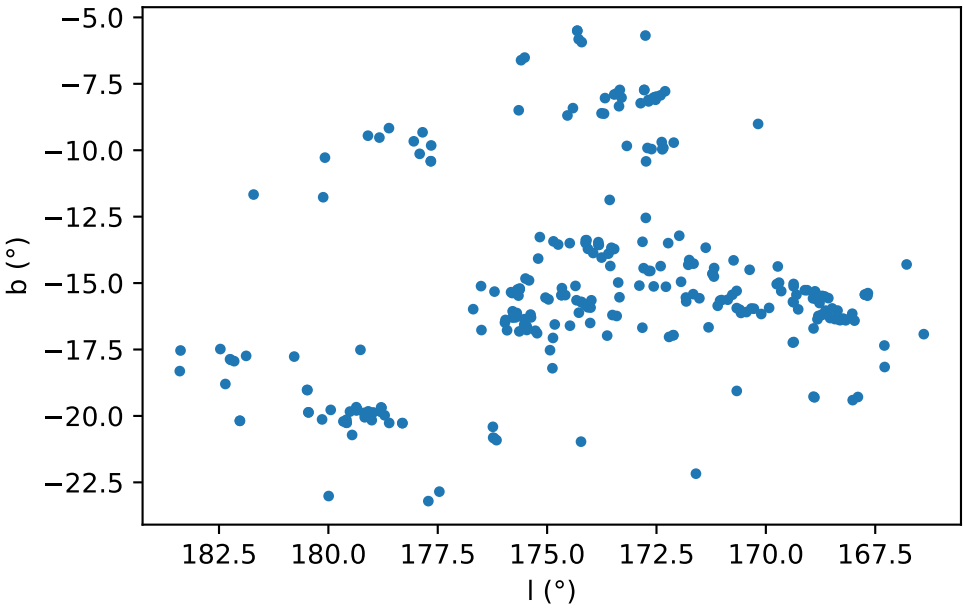
Exclui valores de RUWE <= 1.4

```
# Aplicar o filtro RUWE, como no artigo
df_cleaned = df_filtered[df_filtered['ruwe'] <= 1.4].copy()
print(f"Amostra original: {len(df_filtered)} estrelas")
print(f"Amostra limpa (RUWE <= 1.4): {len(df_cleaned)} estrelas")
```

```
Amostra original: 473 estrelas
Amostra limpa (RUWE <= 1.4): 285 estrelas
```

Plota dados

```
plt.figure()
plt.plot(df_cleaned['l'],df_cleaned['b'], '.')
plt.gca().invert_xaxis()
plt.xlabel('l (°)')
plt.ylabel('b (°)')
plt.show()
```


−5.0
−7.5
−10.0
−12.5
−15.0
−17.5
−20.0
−22.5
b (°)
182.5
180.0
177.5
175.0
172.5
170.0
167.5
l (°)