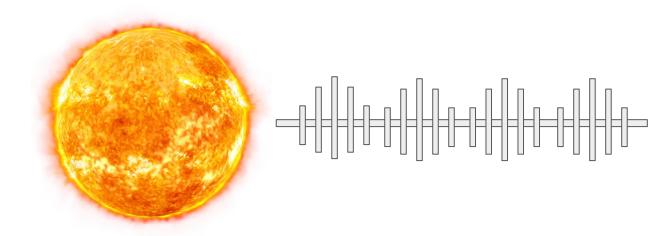


MEF/MAIE Advanced Techniques in Data Analysis

Professor José Ricardo Gonçalo

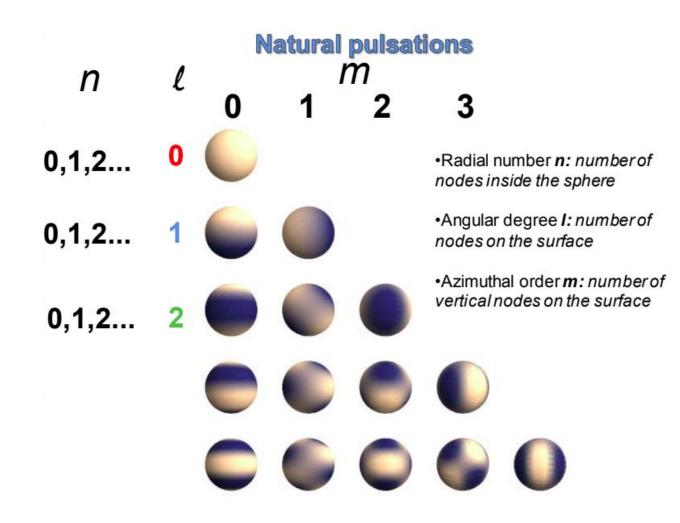
Small Separations in Red Giants Stars: quest for late-evolution signatures using Asteroseismology

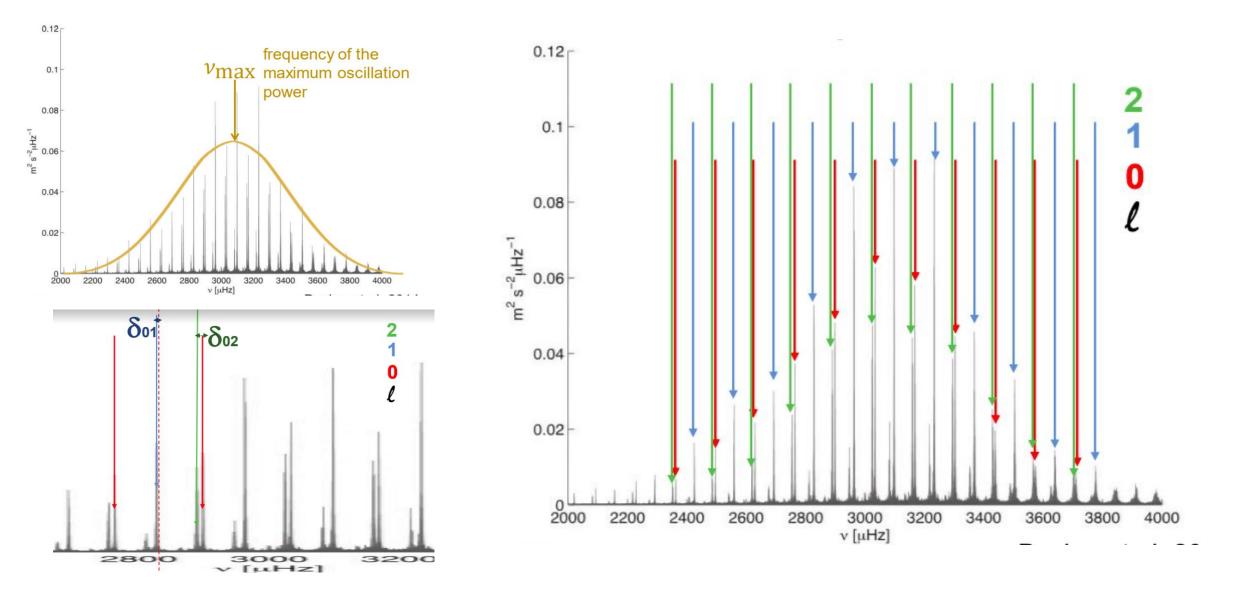
Maria Inês Ferreira, 2018296679 Simão Pedro Neto Sousa, 2017253022



Introduction

Asteroseismology

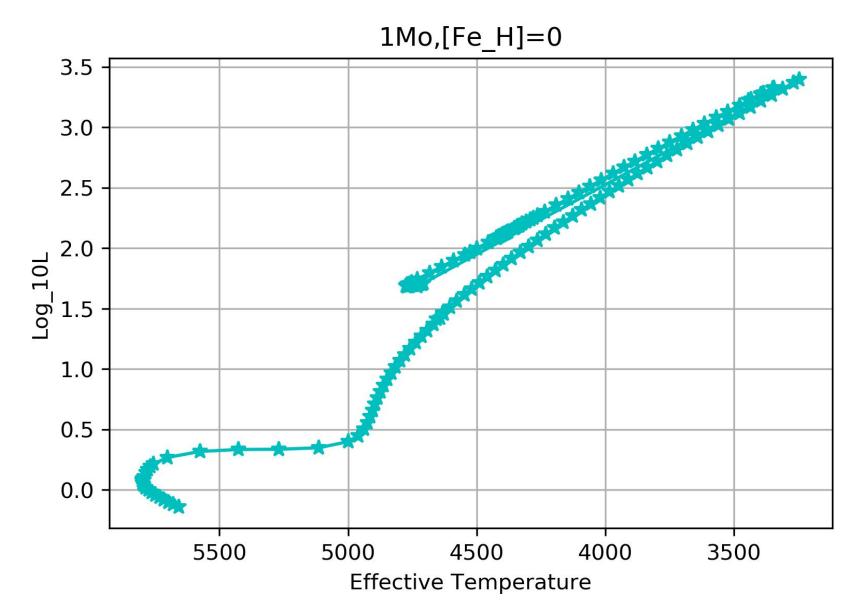




2/10

3/18

Hertzsprung-Russell Diagram



Data used

A library of frequencies, amplitudes, and lifetimes of more than 250,000 individual I=0 to 3 oscillations modes of 6,179 red giants from APOKASC sample (Pinsonneault et al. 2018) - KALLINGER SAMPLE

- *fmax*: The frequency of the maximum oscillation power in microHz;
- *dnu*, *dnu02*: The large and small frequency separation determined in the central three radial orders around fmax. All parameters are in microHz;
- *dnu_cor*: Curvature-corrected large separation in microHz;
- evo: Evolutionary stage of the star determined from the phase shift of the central radial mode (Kallinger et al. 2012) with the following code: 0 RGB star, 1 RC star, 2 secondary clump star, and 3 AGB star.



KEPLER SAMPLE:

- *TEFF*: effective temperature of the star (in Kelvin);
- *M_H:* metallicity of the star;
- *M/Msun*: stellar mass in sollar units.



Correlation Heatmap

Correlation Heatmap

ID -	- 1	-0.1	-0.098	-0.093	0.028	-0.077	0.061	-0.1	0.037	-0.093	0.042	0.07	0.046	0.11	1	-0.04	0.015	-0.075	-0.022	-0.13			
fmax -	-0.1	1	0.62	0.99	-0.12	0.97	-0.11	1	-0.15	0.99	-0.23	-0.56	-0.53	-0.42	-0.1	0.3	0.45	0.17	0.054	0.17			0.8
fmax_e -	-0.098	0.62	1	0.59	0.073	0.55	0.23	0.62	0.073	0.59	0.095	-0.35	-0.27	-0.14	-0.098	0.24	0.4	0.2	0.0085	0.29			0.8
dnu_cor -	-0.093	0.99	0.59	1	-0.12	0.99	-0.11	0.99	-0.14	1	-0.23	-0.58	-0.56	-0.41	-0.093	0.32	0.48	0.15	0.072	0.12			
dnu_cor_e -	0.028	-0.12	0.073	-0.12	1	-0.093	0.48	-0.13	0.87	-0.12	0.4	0.21	0.56	0.3	0.028	0.085	0.092	-0.052	0.068	-0.19			
dnu02 -	-0.077	0.97	0.55	0.99	-0.093	1	-0.049	0.97	-0.11	0.99	-0.19	-0.55	-0.54	-0.34	-0.077	0.35	0.51	0.11	0.1	0.028			0.4
dnu02_e -	0.061	-0.11	0.23	-0.11	0.48	-0.049	1	-0.12	0.54	-0.1	0.61	0.16	0.34	0.6	0.061	0.19	0.35	-0.016	0.054	-0.2			0.4
fm -	-0.1	1	0.62	0.99	-0.13	0.97	-0.12	1	-0.16	0.99	-0.23	-0.57	-0.54	-0.43	-0.1	0.29	0.45	0.16	0.053	0.17			
fm_e -	0.037	-0.15	0.073	-0.14	0.87	-0.11	0.54	-0.16	1	-0.14	0.47	0.17	0.45	0.36	0.037	0.081	0.14	-0.012	0.031	-0.14			
dnu -	-0.093	0.99	0.59	1	-0.12	0.99	-0.1	0.99	-0.14	1	-0.22	-0.57	-0.56	-0.41	-0.093	0.32	0.48	0.15	0.071	0.12			0.0
dnu_e -	0.042	-0.23	0.095	-0.23	0.4	-0.19	0.61	-0.23	0.47	-0.22	1	0.21	0.42	0.46	0.042	0.056	0.14	-0.0027	0.0043	-0.18			0.0
alp -	0.07	-0.56	-0.35	-0.58	0.21	-0.55	0.16	-0.57	0.17	-0.57	0.21	1	0.62	0.33	0.07	-0.2	-0.36	-0.2	0.046	-0.38			
alp_e -	0.046	-0.53	-0.27	-0.56	0.56	-0.54	0.34	-0.54	0.45	-0.56	0.42	0.62	1	0.36	0.046	-0.19	-0.35	-0.16	0.037	-0.3			
evo -	0.11	-0.42	-0.14	-0.41	0.3	-0.34	0.6	-0.43	0.36	-0.41	0.46	0.33	0.36	1	0.11	0.031	0.24	0.017	-0.066	-0.27			0.4
kepid -	1	-0.1	-0.098	-0.093	0.028	-0.077	0.061	-0.1	0.037	-0.093	0.042	0.07	0.046	0.11	1	-0.04	0.015	-0.075	-0.022	-0.13			-0.4
TEFF_ERR -	-0.04	0.3	0.24	0.32	0.085	0.35	0.19	0.29	0.081	0.32	0.056	-0.2	-0.19	0.031	-0.04	1	0.62	-0.4	0.87	-0.055			
TEFF -	0.015	0.45	0.4	0.48	0.092	0.51	0.35	0.45	0.14	0.48	0.14	-0.36	-0.35	0.24	0.015	0.62	1	-0.25	0.31	0.14			
M_H -	-0.075	0.17	0.2	0.15	-0.052	0.11	-0.016	0.16	-0.012	0.15	-0.0027	-0.2	-0.16	0.017	-0.075	-0.4	-0.25	1	-0.68	0.32			
M_H_ERR -	-0.022	0.054	0.0085	0.072	0.068	0.1	0.054	0.053	0.031	0.071	0.0043	0.046	0.037	-0.066	-0.022	0.87	0.31	-0.68	1	-0.21		-	-0.8
M/Msun -	-0.13	0.17	0.29	0.12	-0.19	0.028	-0.2	0.17	-0.14	0.12	-0.18	-0.38	-0.3	-0.27	-0.13	-0.055	0.14	0.32	-0.21	1			
	- Cl	fmax -	fmax_e -	dnu_cor -	dnu_cor_e -	dnu02 -	dnu02_e -	- m	fm_e -	- nup	dnu_e -	alp -	alp_e -	evo -	kepid -	TEFF_ERR -	TEFF -	- н ⁻ м	M_H_ERR -	- M/Msun -	6/18		

Machine Learning Code

```
1 from numpy import loadtxt
2 from keras.models import Sequential
3 from keras.layers import Dense
4 from matplotlib import pyplot
5 import pandas as pd
6 from sklearn.model_selection import train_test_split
7 import time
8 import seaborn as sns
9 import matplotlib.pyplot as plt
```

```
23 #Split in input 'X' and in output 'Y'-
                                                                                 #
24 X = stars[['fmax', 'dnu_cor', 'dnu02', 'dnu', 'TEFF', 'M_H', 'M/Msun']]
25 y = stars[['evo']]
27 #print(X)
28 #print(y)
31 #Correlation heatmap-
32 sns.heatmap(stars.corr())
33 plt.figure(figsize=(16, 6))
34 heatmap = sns.heatmap(stars.corr(), vmin=-1, vmax=1, annot=True)
35 heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
36
```

```
50 #Model-
     #define the keras model-----
52 model = Sequential()
53 model.add(Dense(10, input_dim=(7), kernel_initializer='uniform', activation='relu'))
54 model.add(Dense(6, kernel initializer='uniform', activation='relu'))
55 model.add(Dense(1, kernel initializer='uniform', activation='sigmoid'))
  model.summary()
     #compile the keras model--
60 model.compile(loss='binary crossentropy', optimizer='adam', metrics=['accuracy'])
     #fit the keras model on the dataset-------
63 history=model.fit(trainX, trainY, epochs=900, batch size=50,validation data=(valX, valY))
```

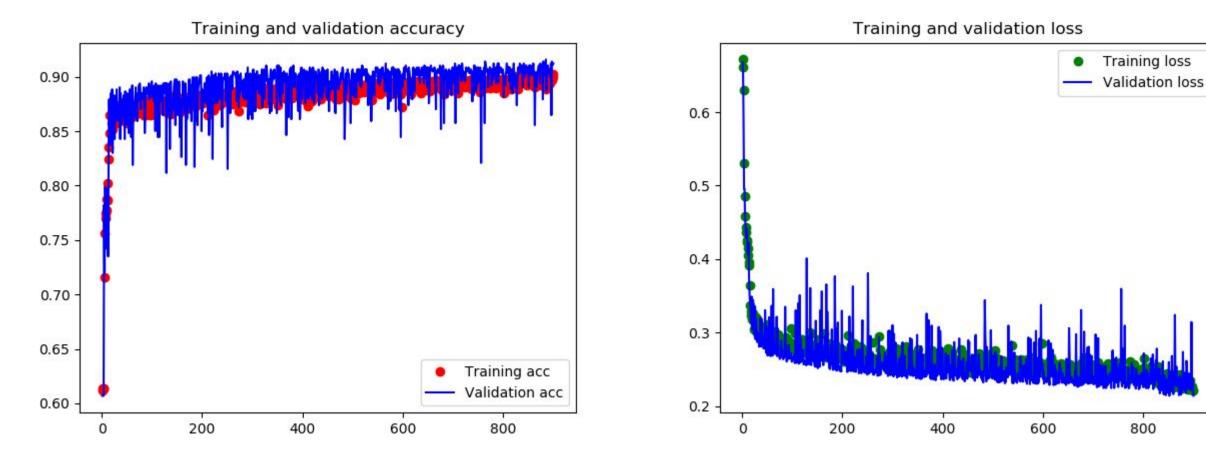
```
67 #Obtaining the best model values.....#
68 hist_stars=pd.DataFrame(history.history)
69 hist_stars['epoch'] = hist_stars.index + 1
70 cols = list(hist_stars.columns)
71 cols = [cols[-1]] + cols[:-1]
72 hist_stars= hist_stars[cols]
73 #hist_stars.to_csv('/' + 'history_stars_' + model_name + '.csv')
74 hist_stars.head()
75
76 values_of_best_model = hist_stars[hist_stars.val_loss == hist_stars.val_loss.min()]
77 print('Best model', values_of_best_model)
```

```
81 #Validation-
82 acc = history.history['accuracy']
83 val acc = history.history['val_accuracy']
84 loss = history.history['loss']
85 val loss = history.history['val loss']
86
87 epochs = range(1, len(acc) + 1)
89 plt.plot(epochs, acc, 'ro', label='Training acc')
90 plt.plot(epochs, val_acc, 'b', label='Validation acc')
91 plt.title('Training and validation accuracy')
92 plt.legend()
93 plt.show()
96 plt.plot(epochs, loss, 'go', label='Training loss')
97 plt.plot(epochs, val_loss, 'b', label='Validation loss')
98 plt.title('Training and validation loss')
99 plt.legend()
01 plt.show()
```

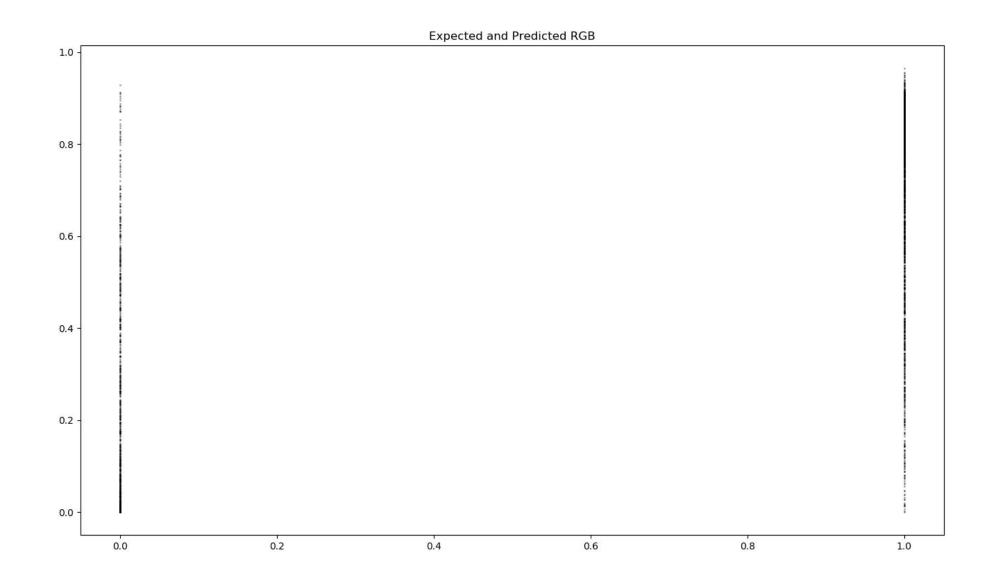
```
103
104 #Model testing------#
105 test_loss, test_acc = model.evaluate(testX, testY)
106 print()
107 print('Test Accuracy:', test_acc)
108
```

```
108 #Predictions-
109 y_pred = model.predict(X)
110 #print(y_pred)
112 plt.scatter(y,y_pred, c='k', s=1, alpha=0.3)
113 plt.title('Expected and Predicted RGB')
114 plt.show()
116
117 # get the end time-----
118 et = time.time()
119
120 # get the execution time-----
121 elapsed_time = et - st
122 print('Execution time:', elapsed_time, 'seconds')
```

Results/Conclusion



Epoch	י 900/900
88/88	3 [==============================] - 0s 792us/step - loss: 0.2219 - accuracy: 0.8999 - val_loss: 0.2139 - val_accuracy: 0.9122
Best	model epoch loss accuracy val_loss val_accuracy
851	852 0.238022 0.89719 0.213854 0.91042
18/18	3 [=========================] - 0s 794us/step - loss: 0.2108 - accuracy: 0.8996
Test	Accuracy: 0.8996350169181824
18/18	3 [=======================] - 0s 733us/step
Execu	ution time: 259.887921333313 seconds
maria	a@maria-Creator-M16-A12UC:~/Documents/TAAD/Projeto\$



18/19

The End + Questions