**Verifying RRLyare Stars from GAIA DR3 using Machine Learning**

SeungJun Ryu, Dr. Nihan Pol

Texas Tech University

**ABSTRACT**

This final project is to comprehend the knowledge of data analysis and machine learning covered in ASTR 3300, Astro-statistics. Main goal for this project is to use machine learning to verify RR Lyrae stars from variable stars data. The target dataset is GAIA Data Release 3. From the list a handful of stars will be carefully selected that we are certain of RRL and used to train the module. After such the module will analyse dataset of variable stars to sort out RRL stars. The project could also be modified to verify other binary stars. Target size is approximately 100 random selected stars. Target data size can be increased once the module is fully complete and functional.

1. **Introduction**

The Gaia DR3 catalogue is the outcome of the processing of raw data collected with the Gaia instruments during the first 34 months of the mission by the Gaia Data Processing and Analysis Consortium. The Gaia DR3 catalogue contains the same source list, celestial positions, proper motions, parallaxes, and broad band photometry in the G, GBP, and GRP passbands already present in the Early Third Data Release, Gaia EDR3. Gaia DR3 introduces an impressive wealth of new data products.

The GAIA mission is able to identify variable stars in its catalogue because it visits each target multiple times. The classifications of these variables, however, may or may not be accurate. To investigate this, a magnitude limited sample of 237 were taken that are RR Lyrae variables identified by GAIA as being RRd (double mode) type. In previous research, analysis of data from the Transiting Exoplanet Survey Satellite (TESS) was made to confirm this classification and look at their properties. Taking a step back, instead of accessing each star observation data, machine learning package from python will analyse GAIA DR3 directly. This project will work on the same sample of 237 since each star was examined and categorised into four labels of G (Good), L (requires another Look), N (No or bad data), and A (Abnormal).

In ASTR3300 course, various methods of data analysing model using machine learning packages in python was covered. This project will practice such skills with GAIA DR3, as presented.

1. **Dataset**

The data can be downloaded from VizieR. (see Reference [Webpages] – (data source) This dataset includes RR Lyrae type d stars with G < 16 mag. The dataset includes 237 stars. Columns selected were; Source, SolID, PF, P1O, EpochG, Gmagavg, BPmagavg, RPmagavg, RVavg, ptpG, ptpRV, R21G, R31G, phi21G, phi31G, FuNFreq1, FuNFreq2, Class, Pratio.

Each column of the data and their explanation is described on Table 1. Description follows the VizieR webpage. Each star in the dataset has been analysed using Tess Observation data. Each star is then labelled and separated to individual files, RRL\_G, RRL\_nG, and RRL\_B. RRL\_G are the stars that are going to be used to train as a normal RR Lyrae stars, RRL\_nG is dataset excluding RRL\_G. RRL\_B are the stars with bad dataset. File format is in .csv.

For “Good” RRd stars, observational data plot looks like [plot] in Appendix.

Table 1. Data Columns and Explanation

|  |  |
| --- | --- |
| **Column** | **Explain** |
| Source | Unique source identifier |
| SolID | Solution Identifier |
| PF | Period corresponding to the fundamental pulsation mode in the G band time series |
| P1O | Period corresponding to the first overtone pulsation mode in the G band time series |
| EpochG | Epoch of the maximum of the light curve in the G band |
| Gmagavg | Intensity-averaged magnitude in the G band |
| BPmagavg | Intensity-averaged magnitude in the BP band |
| RPmagavg | Intensity-averaged magnitude in the RP band |
| RVavg | Mean radial velocity |
| ptpG | Peak-to-peak amplitude of the G band light curve |
| ptpRV | Peak-to-peak amplitude of the radial velocity curve |
| R21G | Fourier decomposition parameter R21G: A2/A1 |
| R31G | Fourier decomposition parameter R31G: A3/A1 |
| phi21G | Fourier decomposition parameter phi21G: phi2 - 2\*phi1 |
| phi31G | Fourier decomposition parameter phi31G: phi3 - 3\*phi1 |
| FundFreq1 | First frequency of the non-linear Fourier modelling |
| FundFreq2 | Second frequency of the non-linear Fourier modelling in the G band |
| Class | [Rabcd ] Best RR Lyrae classification estimate |

1. **Method**

For the analysis methods, two models were used. Simple Logistic Regression and Random Forest Classification. Simple Logistic Regression was performed in comparison with Random Forest.

Figure 1. shows how the stars are scattered in Gmagavg vs. PF and P1O vs. PF.

A screenshot of a computer screen

Description automatically generated Figure 1. RF plot (PF vs. Gmagavg, P1O vs. PF)

A screenshot of a computer screen

Description automatically generated

Figure 2. RF plot ( Gmagavg vs. ptpG)

Figure 2 was plotted additional to the original plots for comparison with dimensionally reduced dataset.

A screenshot of a computer

Description automatically generated

Figure 3. Pie Chart Distribution

Above is the Pie chart to show how many stars are falling into which category.

A screenshot of a computer

Description automatically generated

Figure 4. Feature Importance

Feature Importance shows the focus of the Random Forest model on which Column data. Based on this plot, features were selected in Dimensionality Reduction step.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Figure 5. Train dataset modification

Figure 5 is the plot from changing the train dataset. In this procedure, there were two train datasets, Good and Bad. Plots are slightly different, and “Good” stars tend to be more scattered within the whole dataset.

A screenshot of a computer

Description automatically generated

Figure 6. Pie chart after Train Dataset Mod.

Figure 6 shows the pie chart after the modification of training dataset.

A screen shot of a computer

Description automatically generated

Figure 7. Pie Chart after Dimension Reduction

Figure 7 shows the pie chart after the Dimension Reduction. After reduction, the guess of “Good” stars slightly increased.

A screenshot of a computer screen

Description automatically generated

Figure 8. Gmagavg vs. ptpG after Dim. Reduction

Figure 8 shows the scatter plot after the Dimension Reduction. Other two plots were unable to plot since we are not using PF column.

A screenshot of a computer

Description automatically generated

Figure 9. Pie chart of Logistic Regression

Above is the pie chart of Logistic Regression. By looking at just the pie chart, it seemed fined, other than group “L” increased.

A screenshot of a computer

Description automatically generated

Figure 10. scatter plot of Logistic Regression

However, looking at Gmagavg vs. ptpG scatter plot, it is shown that Logistic Regression is not a good model. Unlike Random Forest, it seems the probability of being RRL is highly dependent on ptpG.

For each model, further verifications are required with actual TESS observation data.

1. **Results and Future Steps**

In result, Random Forest Classification performed better at learning and verifying variable stars data. As shown in Simple Logistic Regression plots, turned out as plots having possibility based on ptpG, where more expected to scattered.

With result of Random Forest Classification, setting RRL\_nG as the target data, turned out doing a decent verification. Compared to prior analysis, number wise it was good.

Future step would include verify each “Good” stars in the RRL\_nG dataset.

If those stars were originally in “L” label but classified as “G” according to the model, this can be an indicator to take a closer look or look for smaller details in the dataset. Also this can be used to determine the list of stars for additional observation.

**Reference**

[Journals / Publications]

Ryu, S., Choi, Y., Lockett, J., Patterson, A. & Carrell, K. (2023) *Double-mode RR Lyrae Variables in GAIA and TESS*.

GAIA Collaboration et al. (2023) “Gaia Data Release 3. Summary of the content and survey properties” A&A, 674, A1

[Web sources]

<https://www.cosmos.esa.int/web/gaia/data-release-3>

<https://vizier.cds.unistra.fr/viz-bin/VizieR-3?-source=I/358/vrrlyr> (data source)

[Class Material]

<https://github.com/Holang2/ASTR_3300_S2025> (ASTR3300 class material)

**Appendix**

[Plot]

A screenshot of a graph

Description automatically generated

Figure above is how a plot TESS data for each stars looks like. Example of a good RRLd star.

[Code]

Code can also be accessed in a form of Jupyter Notebook.

Code cell 1

###############################

##### ASTR 3300 Project #####

##### SeungJun Ryu (June) #####

###############################

# imports

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

import numpy as np

# Load data

rrl\_g = pd.read\_csv("./RRL\_G.csv")

rrl\_ng = pd.read\_csv("./RRL\_nG.csv")

# Label confirmed RR Lyrae

rrl\_g['label'] = 1

# negative from part of rrl\_ng

rrl\_ng\_negative = rrl\_ng.sample(n=len(rrl\_g), random\_state=42).copy()

rrl\_ng\_negative['label'] = 0

# Combine for training

training\_data = pd.concat([rrl\_g, rrl\_ng\_negative], ignore\_index=True)

# Define features

features = [

'PF', 'P1O', 'Gmagavg', 'BPmagavg', 'RPmagavg',

'ptpG', 'ptpRV', 'R21G', 'R31G', 'phi21G', 'phi31G',

'FuNFreq1', 'FuNFreq2', 'Pratio'

]

# Fill missing values

X\_train = training\_data[features].fillna(training\_data[features].mean())

y\_train = training\_data['label']

# Scale features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

# Train classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=45)

clf.fit(X\_train\_scaled, y\_train)

# Predict on entire rrl\_ng dataset (excluding those used for negative training)

rrl\_ng\_eval = rrl\_ng.drop(index=rrl\_ng\_negative.index, errors='ignore').copy()

X\_target\_scaled = scaler.transform(rrl\_ng\_eval[features].fillna(training\_data[features].mean()))

rrl\_ng\_eval['predicted\_label'] = clf.predict(X\_target\_scaled)

rrl\_ng\_eval['RR\_probability'] = clf.predict\_proba(X\_target\_scaled)[:, 1]

# Visualisation

import seaborn as sns

sns.set(style="whitegrid")

# Scatter plot of Period (PF) vs G-band magnitude, colored by probability

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_ng\_eval['PF'], rrl\_ng\_eval['Gmagavg'],

c=rrl\_ng\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("PF vs Gmagavg Colored by RR Lyrae Probability")

plt.xlabel("Fundamental Period (PF)")

plt.ylabel("Mean G-band Magnitude")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_ng\_eval['P1O'], rrl\_ng\_eval['Pratio'],

c=rrl\_ng\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("P1O vs Pratio Colored by RR Lyrae Probability")

plt.xlabel("Fundamental Period (PF)")

plt.ylabel("First Overtone (P1O)")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_ng\_eval['Gmagavg'], rrl\_ng\_eval['ptpG'],

c=rrl\_ng\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("GmagAvg vs ptpG")

plt.xlabel("ptpG")

plt.ylabel("GmagAvg")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

**code cell 2**

# import

import pandas as pd

import matplotlib.pyplot as plt

# Reload data files

rrl\_g = pd.read\_csv("./RRL\_G.csv")

rrl\_ng = pd.read\_csv("./RRL\_nG.csv")

# Rebuild training and model

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

# Label confirmed RR Lyrae

rrl\_g['label'] = 1

# Simulate a negative class from part of rrl\_ng

rrl\_ng\_negative = rrl\_ng.sample(n=len(rrl\_g), random\_state=45).copy()

rrl\_ng\_negative['label'] = 0

# Combine training data

training\_data = pd.concat([rrl\_g, rrl\_ng\_negative], ignore\_index=True)

# Define features

features = [

'PF', 'P1O', 'Gmagavg', 'BPmagavg', 'RPmagavg',

'ptpG', 'ptpRV', 'R21G', 'R31G', 'phi21G', 'phi31G',

'FuNFreq1', 'FuNFreq2', 'Pratio'

]

# Fill missing values and scale features

X\_train = training\_data[features].fillna(training\_data[features].mean())

y\_train = training\_data['label']

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

# Train classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=45)

clf.fit(X\_train\_scaled, y\_train)

# Predict on remaining rrl\_ng entries

rrl\_ng\_eval = rrl\_ng.drop(index=rrl\_ng\_negative.index, errors='ignore').copy()

X\_target\_scaled = scaler.transform(rrl\_ng\_eval[features].fillna(training\_data[features].mean()))

rrl\_ng\_eval['predicted\_label'] = clf.predict(X\_target\_scaled)

rrl\_ng\_eval['RR\_probability'] = clf.predict\_proba(X\_target\_scaled)[:, 1]

# Categorize stars based on RR Lyrae probability

def label\_star(prob):

if prob > 0.75:

return 'G' # Good candidate

elif prob < 0.5:

return 'B' # Bad candidate

else:

return 'L' # Needs a Look

rrl\_ng\_eval['RR\_label'] = rrl\_ng\_eval['RR\_probability'].apply(label\_star)

# Count labels

label\_counts = rrl\_ng\_eval['RR\_label'].value\_counts().reindex(['G', 'L', 'B'], fill\_value=0)

# Plot pie chart

plt.figure(figsize=(4,4))

plt.pie(label\_counts, labels=label\_counts.index, autopct='%1.1f%%', startangle=140,

colors=['green', 'orange', 'red'], wedgeprops={'edgecolor': 'black'})

plt.title("Distribution of RR Lyrae Candidate Labels (G, L, B)")

plt.tight\_layout()

plt.show()

label\_counts

**Code cell 3**

# Feature importance

importances = clf.feature\_importances\_

feature\_importance\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

feature\_importance\_df.sort\_values(by='Importance', ascending=False, inplace=True)

plt.figure(figsize=(8, 4))

sns.barplot(x='Importance', y='Feature', data=feature\_importance\_df, palette="viridis")

plt.title("Feature Importance from Random Forest Classifier")

plt.tight\_layout()

plt.show()

**code cell 4**

########## Labeling both good and bad ##########

# imports

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

# Load files

rrl\_g = pd.read\_csv("./RRL\_G.csv") # confirmed RR Lyrae

rrl\_b = pd.read\_csv("./RRL\_B.csv") # confirmed NOT RR Lyrae

rrl\_eval = pd.read\_csv("./RRL\_nG.csv") # unknown stars

# Label the training data

rrl\_g['label'] = 1

rrl\_b['label'] = 0

# Combine for training

training\_data = pd.concat([rrl\_g, rrl\_b], ignore\_index=True)

# Define features

features = [

'PF', 'P1O', 'Gmagavg', 'BPmagavg', 'RPmagavg',

'ptpG', 'ptpRV', 'R21G', 'R31G', 'phi21G', 'phi31G',

'FuNFreq1', 'FuNFreq2', 'Pratio'

]

# Fill missing and scale

X\_train = training\_data[features].fillna(training\_data[features].mean())

y\_train = training\_data['label']

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

# Train model

clf = RandomForestClassifier(n\_estimators=100, random\_state=45)

clf.fit(X\_train\_scaled, y\_train)

# Prepare unknown stars

X\_eval = rrl\_eval[features].fillna(training\_data[features].mean())

X\_eval\_scaled = scaler.transform(X\_eval)

# Predict

rrl\_eval['predicted\_label'] = clf.predict(X\_eval\_scaled)

rrl\_eval['RR\_probability'] = clf.predict\_proba(X\_eval\_scaled)[:, 1]

def label\_star(prob):

if prob > 0.75:

return 'G'

elif prob < 0.5:

return 'B'

else:

return 'L'

rrl\_eval['RR\_label'] = rrl\_eval['RR\_probability'].apply(label\_star)

import matplotlib.pyplot as plt

label\_counts = rrl\_eval['RR\_label'].value\_counts().reindex(['G', 'L', 'B'], fill\_value=0)

# Pie chart

plt.figure(figsize=(4, 4))

plt.pie(label\_counts, labels=label\_counts.index, autopct='%1.1f%%', startangle=140,

colors=['green', 'orange', 'red'], wedgeprops={'edgecolor': 'black'})

plt.title("Distribution of RR Lyrae Candidate Labels")

plt.tight\_layout()

plt.show()

print(label\_counts)

# more plots

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_eval['PF'], rrl\_eval['Gmagavg'],

c=rrl\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("PF vs Gmagavg Colored by RR Lyrae Probability")

plt.xlabel("Fundamental Period (PF)")

plt.ylabel("Mean G-band Magnitude")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_eval['P1O'], rrl\_eval['Pratio'],

c=rrl\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("P1O vs Pratio Colored by RR Lyrae Probability")

plt.xlabel("Fundamental Period (PF)")

plt.ylabel("First Overtone (P1O)")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_eval['Gmagavg'], rrl\_eval['ptpG'],

c=rrl\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("GmagAvg vs ptpG")

plt.xlabel("ptpG")

plt.ylabel("GmagAvg")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

**code cell 5**

########## Dimentionality Reduction ##########

# Load data

rrl\_g = pd.read\_csv("./RRL\_G.csv")

rrl\_b = pd.read\_csv("./RRL\_B.csv")

rrl\_eval = pd.read\_csv("./RRL\_nG.csv")

# Assign labels

rrl\_g['label'] = 1

rrl\_b['label'] = 0

training\_data = pd.concat([rrl\_g, rrl\_b], ignore\_index=True)

# 4 features from above (feature importance)

features = ['Gmagavg', 'ptpG', 'BPmagavg', 'RPmagavg']

# Preprocess training data

X\_train = training\_data[features].fillna(training\_data[features].mean())

y\_train = training\_data['label']

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

# Train classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=45)

clf.fit(X\_train\_scaled, y\_train)

# Predict on evaluation data

X\_eval = rrl\_eval[features].fillna(training\_data[features].mean())

X\_eval\_scaled = scaler.transform(X\_eval)

rrl\_eval['predicted\_label'] = clf.predict(X\_eval\_scaled)

rrl\_eval['RR\_probability'] = clf.predict\_proba(X\_eval\_scaled)[:, 1]

# Labeling function

def label\_star(prob):

if prob > 0.75:

return 'G'

elif prob < 0.5:

return 'B'

else:

return 'L'

rrl\_eval['RR\_label'] = rrl\_eval['RR\_probability'].apply(label\_star)

# Pie chart

label\_counts = rrl\_eval['RR\_label'].value\_counts().reindex(['G', 'L', 'B'], fill\_value=0)

plt.figure(figsize=(4,4))

plt.pie(label\_counts, labels=label\_counts.index, autopct='%1.1f%%', startangle=140,

colors=['green', 'orange', 'red'], wedgeprops={'edgecolor': 'black'})

plt.title("RR Lyrae Candidate Labels Using Top 4 Features")

plt.tight\_layout()

plt.show()

label\_counts

**code cell 6**

# plots

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_eval['Gmagavg'], rrl\_eval['ptpG'],

c=rrl\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("GmagAvg vs ptpG")

plt.xlabel("ptpG")

plt.ylabel("GmagAvg")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

**code cell 7**

##### Other Model - Logistic Regression #####

# import

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

# Load data

rrl\_g = pd.read\_csv("./RRL\_G.csv") # Confirmed RR Lyrae

rrl\_b = pd.read\_csv("./RRL\_B.csv") # Confirmed NOT RR Lyrae

rrl\_eval = pd.read\_csv("./RRL\_nG.csv") # Unknown candidates

# Add labels

rrl\_g['label'] = 1

rrl\_b['label'] = 0

training\_data = pd.concat([rrl\_g, rrl\_b], ignore\_index=True)

# Use only the top 4 features

features = ['Gmagavg', 'ptpG', 'BPmagavg', 'RPmagavg']

# Training data

X\_train = training\_data[features].fillna(training\_data[features].mean())

y\_train = training\_data['label']

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

# Train logistic regression model

clf = LogisticRegression(max\_iter=1000, random\_state=42)

clf.fit(X\_train\_scaled, y\_train)

# Evaluation data

X\_eval = rrl\_eval[features].fillna(training\_data[features].mean())

X\_eval\_scaled = scaler.transform(X\_eval)

# Predict labels and probabilities

rrl\_eval['predicted\_label'] = clf.predict(X\_eval\_scaled)

rrl\_eval['RR\_probability'] = clf.predict\_proba(X\_eval\_scaled)[:, 1]

# Labeling prob

def label\_star(prob):

if prob > 0.75:

return 'G'

elif prob < 0.5:

return 'B'

else:

return 'L'

rrl\_eval['RR\_label'] = rrl\_eval['RR\_probability'].apply(label\_star)

# Pie chart visualization

label\_counts = rrl\_eval['RR\_label'].value\_counts().reindex(['G', 'L', 'B'], fill\_value=0)

plt.figure(figsize=(4, 4))

plt.pie(label\_counts, labels=label\_counts.index, autopct='%1.1f%%', startangle=140,

colors=['limegreen', 'gold', 'crimson'], wedgeprops={'edgecolor': 'black'})

plt.title("RR Lyrae Candidate Labels (Logistic Regression)")

plt.tight\_layout()

plt.show()

print(label\_counts)

# plots

plt.figure(figsize=(8, 4))

sc = plt.scatter(rrl\_eval['Gmagavg'], rrl\_eval['ptpG'],

c=rrl\_eval['RR\_probability'], cmap='viridis', edgecolor='k')

plt.colorbar(sc, label='RR Lyrae Probability')

plt.title("GmagAvg vs ptpG")

plt.xlabel("ptpG")

plt.ylabel("GmagAvg")

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()