APOGEE's Galaxy

Section 1- Introduction

In this notebook, you're going to explore the Milky Way galaxy as probed by the most recently published APOGEE dataset (APOGEE DR17). This notebook was developed by referencing notebooks by Danny Horta, Emily Griffith, and Henry Leung, and Ted Mackereth.

You can read more about that dataset at: https://www.sdss4.org/dr17/irspec/ or in papers like https://ui.adsabs.harvard.edu/abs/2022ApJS..259...35A/abstract or https://ui.adsabs.harvard.edu/abs/2020AJ....160..120J/abstract

Step 1: Download the dataset from https://data.sdss.org/sas/dr17/apogee/spectro/aspcap/dr17/synspec_rev1/allStarLite-dr17-synspec_rev1.fits (It's about 1.7 GBs)

Question: What is the resolution of the APOGEE spectra? What wavelength range do they cover?

Answer:

Question: Describe in a few sentences what has happend to these data before they were put in this file (how were they taken, how were they processed, how were these parameters computed, were any changes or corrections applied, etc.)

Answer:

Section 2- Initial Exploration

In [ ]:
```
#pip install astropy
#pip install --upgrade numpy
```

In [ ]:
```
#Import some things
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import astropy.coordinates as coord
import astropy.units as u
from astropy.io import fits
from astropy.table import Table
from astropy.coordinates import SkyCoord
```

In [ ]:
```
#load in the data (may have to change this for wherever you downloaded your file
#in google colab you can get the file using
#!wget https://data.sdss.org/sas/dr17/apogee/spectro/aspcap/dr17/synspec_rev1/al

filename='allStarLite-dr17-synspec_rev1.fits'
tb = fits.open(filename)
header=tb[1].header
data = tb[1].data
```

Question: How many stars are in this dataset?

In [ ]:
```
#Answer:
print('Answer: There are '+str(len(data))+' targets in DR17')
```

To figure out what sort of data is included in this table, you can look at the datamodel: https://data.sdss.org/datamodel/files/APOGEE_ASPCAP/APRED_VERS/ASPCAP_VERS/allStarLite.ht or print the header

In [ ]:
```
print(header)
```

Let's figure out what type of stars these are- let's plot an HR diagram (well, really a Kiel Diagram- temperature and logg)

In [ ]:
```
plt.figure()
plt.scatter(data['TEFF'], data['LOGG'])
```

Question: There are several issues with this plot, list at least three of them

Answer:

Okay, let's make a more useful plot

In [ ]:
```
#plot stuff
plt.figure()
plt.scatter(data['TEFF'], data['LOGG'],c=data['M_H'],vmin=-1.5,vmax=0.5,s=0.1)
plt.xlabel('Teff [K]')
plt.ylabel('Logg [dex]')
plt.colorbar(label='[M/H]')
plt.xlim(7000,3000)
plt.ylim(6,-1)
```

Question: Describe the types of stars present on this plot: what evolutionary phases are they in? What mass ranges do you expect them to cover? How does the distribution in this sample compare to the distribution of stars in the field?

Answer:

Question: Plot the RAs and Decs of these stars. What parts of the sky do they come from? Describe at least two features of the selection pattern, and explain how they might have arisen. (If you get stuck, the selection functions are explained in https://ui.adsabs.harvard.edu/abs/2017AJ....154..198Z/abstract, https://ui.adsabs.harvard.edu/abs/2021AJ....162..302B/abstract , and https://ui.adsabs.harvard.edu/abs/2021AJ....162..303S/abstract)

In [ ]:
```
# code to make a plot here
```

Answer:

Question: The galaxy is often divided into an α-rich and an α-poor population. Plot the α /Metallicity versus Metallicity/H for this sample. Choose an appropriate scale and point size so that you can see the different components. Describe the populations in this data.

In [ ]:
```
#code to make a plot here (the alpha column is called ALPHA_M)
```

Answer:

Section 3- Chemical Cartography

It's all well and good to see where these stars are in our sky, but what we really want to know is where they are located in the galaxy. It turns out we have enough data to calculate that for these stars. We have RA and Dec information, proper motions in RA and DEC (Gaia), and the radial velocity information from the APOGEE spectra.

In [ ]:
```
# get rid of stars with negative distances
mask_gaia = (data['GAIAEDR3_PARALLAX']>0)

# use skycoord to input information
c = SkyCoord(ra=data['RA'][mask_gaia]*u.deg, dec=data['DEC'][mask_gaia]*u.deg,
             distance=(data['GAIAEDR3_R_MED_GEO'][mask_gaia])*u.kpc,
             pm_ra_cosdec=data['GAIAEDR3_PMRA'][mask_gaia]*u.mas/u.yr,
             pm_dec=data['GAIAEDR3_PMDEC'][mask_gaia]*u.mas/u.yr,
            radial_velocity=data['RV_CCFWHM'][mask_gaia]*u.km/u.s)

# transform to galactocentric coordinate
# the assumed values are from Schonrich et al 2010, and from Gravity collab 2022
galcen = c.transform_to(coord.Galactocentric(galcen_v_sun=[8, 254, 8] * u.km / u
```

In [ ]:
```
# Extract the galactic coordinates and motions
x = galcen.x.value
y = galcen.y.value
z = galcen.z.value

vx = galcen.v_x.value
vy = galcen.v_y.value
vz = galcen.v_z.value

#galactocentric radius (either in the disk-R or distance from the galactic cente
R = np.sqrt(x**2+y**2)
rgal = np.sqrt(x**2+y**2+z**2)
```

In [ ]:
```
#I clipped data table, including the the M/H and alpha/M vectors to the same len
data_masked=data[mask_gaia]
```

Question: Make a plot of the R (galactocentric radius) versus Z (height versus the plane of the galaxy), color coded by metallicity. What components of the galaxy are being probed? How is the metallicity correlated with position in the galaxy?

In [ ]:
```
# make a plot here of R and z.
#Don't forget if you color code by something you have to use the
#data_masked version to make sure all the arrays are the same length
```

Answer:

Question: How does the chemistry of stars vary with their position in the galaxy? Make a plot similar to Figure 4 of Hayden et al. 2015 (the last paper you read). How similar is your plot to theirs? How do they differ? Do you agree with their conclusions?

In [ ]:
```
#code to make plots.

#If you can get the 'subfigure' part working or do this in a loop that's great.
#If you can't, you can make each plot individually
#code bits that might be useful
#fig, ax = plt.subplots()
#bin1= np.where((R > 3000) & (R < 5000) & (abs(z)>2000) & (data_masked['ALPHA_M'
#ax.hist2d(data_masked['M_H'][bin1], data_masked['ALPHA_M'][bin1], bins=100, cma
# don't forget to make sure you're using the right units for R & z
```

Answer:

Section 4- Adding Ages

Question: Your colleague claims to have calculated ages for a subset of the giant stars using asteroseismology and stellar models. Write a (~paragraph) description summarizing the steps they must have taken.

Answer:

Question: List three questions you would need to ask about their analysis in order to understand the details of it better.

Answer:

In [ ]:
```
#Read in file
agefilename='APOKASC705_AGE_short.fits'
agetb = fits.open(agefilename)
ageheader=agetb[1].header[0:14]
agedata = agetb[1].data
#put the ages in an array
ages=agedata['AGE_JT']
apogeeidAge=agedata['APOGEE_ID']
#print the header
print(ageheader)
```

Question: How many stars are in this dataset?

In [ ]:
```
#Answer:
print('Answer: There are '+str(len(agedata))+' targets with ages')
```

Question: What type of stars are these?

In [ ]:
```
#code to make plot or print things
```

Answer:

We have to match these stars to the stars with APOGEE data

In [ ]:
```
#Figure out which stars are in both lists and what their array elements are
intersect, ind_a, ind_b = np.intersect1d(data_masked['APOGEE_ID'],agedata['APOGE
print(len(ind_b))
```

Question: How does the alpha element abundance of the stars with ages correlate with age?

In [ ]:
```
#code to make a plot
#remember to make sure you're plotting the same star for each x and y coordinate
#you'll probably want something like
#agedata['AGE_JT'][ind_b], data_masked['ALPHA_M'][ind_a]
```

Answer: describe the plot here

Question: How does the carbon-to-nitrogen ratio correlate with age? Do you agree with papers that have tried to use this as an age diagnostic?

In [ ]:
```
#code to make plot here.
#Note that [C/N]= [C/Fe]-[N/Fe] because the square brackets mean that things hav
```

Answer:

Section 5- Machine Learning

It's great that you have ages for 11,000 or so stars, but wouldn't it be better to have ages for a sizable fraction of the 700,000 stars in the full sample? We know from stellar evolution (and the plots that you've made) that the age of the star should be correlated with it's position on the HR diagram, temperature and surface gravity; its composition, metallicity and α abundance; and its mass, which scales with the carbon and nitrogen at the surface for giants. So we should in theory be able to come up with some scheme that can learn these relationships. In this case the scheme we're going to use is a neural network.

In [ ]:
```
#by default the machine learning stuff tries to use a GPU.
#If you don't have a GPU (or want to use a CPU instead you need something like t
#You shouldn't need this on google colab

import os
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
```

In [ ]:
```
from tensorflow import keras
```

We are going set up a data structure that contains only the parameters we want to use for training the neural network. We also want to remove anything that has a NaN in it, and normalize things for

simplicity.

In [ ]:
```
fullx = np.dstack([data_masked['TEFF'][ind_a],data_masked['LOGG'][ind_a], data_m
                   data_masked['C_FE'][ind_a], data_masked['N_FE'][ind_a]])[0]
fully = np.dstack([agedata['AGE_JT'][ind_b]])[0]

#remove non-finite entries!
mask = np.all(np.isfinite(fullx), axis=1) & np.all(np.isfinite(fully), axis=1)
fullx, fully = fullx[mask], fully[mask]

scaling_x = np.median(fullx, axis=0)
scaling_y = np.median(fully, axis=0)

fullx, fully = fullx/scaling_x, fully/scaling_y
```

For the neural network I want everyone to choose a different number of nodes in each layer, a different number of layers, and a different number of training iterations.

In [ ]:
```
#pick some numbers
neurons_per_layer=12
layers=3
iterations=10
```

If you add or subtract layers, you have to do that manually

In [ ]:
```
#start with an input layer
inputs = keras.Input(shape=(5,))
#now we add the Dense layers (indicating the previous layer in the brackets foll

#change this part if you're changing the number of layers
layer1 =keras.layers.Dense(neurons_per_layer, activation='relu')(inputs)
layer2 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer1)
layer3 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer2)
#layer4 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer3)
#layer5 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer4)
#layer6 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer5)
#layer7 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer6)
#layer8 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer7)
#layer9 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer8)
#layer10 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer9)
#layer11 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer10)
#layer12 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer11)
#layer13 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer12)
#layer14 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer13)
#layer15 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer14)
#layer16 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer15)
#layer17 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer16)
#layer18 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer17)
#layer19 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer18)
#layer20 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer19)
#layer21 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer20)
#layer22 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer21)
#layer23 = keras.layers.Dense(neurons_per_layer, activation='relu')(layer22)

#then the output layer
outputs = keras.layers.Dense(1)(layer3)
```

```
# then we put that all together in the Model object
model = keras.Model(inputs=inputs, outputs=outputs, name='test')
#and we can print a summary to check it all went to plan
model.summary()
```

In [ ]:
```
model.compile(loss=keras.losses.MeanSquaredError(), optimizer=keras.optimizers.A
```

When you train a machine learning model, you always want to leave out a testing set, a set of stars where you know what the correct answer, but the neural network hasn't seen, so that you can test its response.

If your last name is starts with a letter before earlier in the alphabet than M, leave the last 2000 stars out of the training. If your last name starts with M or later, leave the first 2000 stars out of the training.

In [ ]:
```
#last name before M
trainbin=slice(0,-2002)
testing=slice(-2001,-1)


#last name M or later
#trainbin=slice(2001,-1)
#testing=slice(0,2000)


x_train, y_train = fullx[trainbin], fully[trainbin]
x_test, y_test = fullx[testing], fully[testing]
```

Okay now train the neural network

In [ ]:
```
model.fit(x_train, y_train, epochs=iterations, validation_split=0.05, batch_size
```

Now that you have trained the neural network, use it to predict what the ages of the stars in the testing set should be.

In [ ]:
```
predictions = model.predict(x_test)
print(len(predictions))
```

How well did your neural network do? Let's plot and print things! The metric we're going to use is whether the neural network predicted age is within 30% of the correct answer or not.

In [ ]:
```
metric=0.3 #is the accuracy better than 30%?
goodfit=np.where(((1-metric) < predictions/y_test) & ((1+metric) > predictions/y
badfit=np.where(((1-metric) > predictions/y_test) | ((1+metric) < predictions/y_

print ('With ', neurons_per_layer, 'neurons per layer, ', layers, 'layers, and '
print ('using the training set', trainbin)
print (len(goodfit[0])/len(y_test)*100, 'percent of the ages are good')
print (len(badfit[0])/len(y_test)*100, 'percent of the ages are bad')
```

In [ ]:
```
#remember that we scaled our input/output
plt.scatter( (y_test*scaling_y),(predictions*scaling_y), s=1.)
plt.scatter( (y_test[goodfit]*scaling_y),(predictions[goodfit]*scaling_y), s=1.)
plt.xlim(0.,14.)
plt.plot([0.,14.], [0.,14.])
plt.plot([14,0,14], [14*(1-metric), 0, 14*(1+metric)])

plt.ylabel(r'Predicted age [Gyr]')
plt.xlabel(r'Seismic Age [Gyr]')
plt.ylim(0,14)
#ax[1].scatter(predictions[:,0]*scaling_y[0], fully[-600:,0]*scaling_y[0], s=1.)
#ax[1].plot([0.5,3.5], [0.5,3.5])
#ax[1].set_xlabel(r'predicted mass $\mathrm{[M_\odot]}$')
#ax[1].set_ylabel(r'APOKASC-2 mass $\mathrm{[M_\odot]}$')
```

Question: How satisfied are you with the performance of your neural network? How does it compare to the direct [C/N]-age relation you plotted above?

Answer:

Now that you have a sense of how well your neural network is doing, forge ahead and predict ages for the whole data sample!

In [ ]:
```
DR17x = np.dstack([data_masked['TEFF'],data_masked['LOGG'], data_masked['M_H'],
                   data_masked['C_FE'], data_masked['N_FE']])[0]
print(len(data_masked['TEFF']))

DR17x= DR17x/scaling_x
predictionsDR17 = model.predict(DR17x)
```

When making inferences from machine learning, you should always be careful about extrapolating outside the reach of your training data. Make at least one cut to the dataset to make your full sample more consistent with your training sample.

In [ ]:
```
# code to define a good sample
#I used np.where for this
#good=np.where()
```

Question: What cut did you make and why?

Answer:

In [ ]:
```
plt.scatter( R[good]/1000,z[good]/1000,c=predictionsDR17[good]*scaling_y,vmin=0,
plt.colorbar(label='Predicted Age [Gyr]')
plt.ylabel(r'Galactic Height [kpc]')
plt.xlabel(r'Galactocentric Radius [kpc]')
#plt.ylim(-5,5)
#plt.xlim(0,10)
```

Question: Describe your age map of the galaxy. How does it compare to other age maps of the galaxy? In what ways do you think it is better? In what ways is it less good?

Answer:

Section 6- Extension

When you turned in your summary of the Hayden et al. 2015 paper, you described "5. Given the results of this study, what study would you want to do next with a dataset like this?" Write a summarized version of that question in the question section below, make the appropriate plots to attempt it, and describe the answer you found below. Were you able to answer it with this improved data set? If not, why not?

Question:

In [ ]:
```
# code to manipulate data?
# code to make plots?
```

Answer:

In [ ]:

In [ ]: