intro_project

April 27, 2022

[1]:
```
# Using ML to classify LAEs in the NEP Field
import tables as tb
import numpy as np

import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

from astropy import constants as const
from astropy.table import Table, column, join
from astropy.io import fits
import astropy.units as u
from astropy.coordinates import SkyCoord
from astropy.visualization import ZScaleInterval

from regions import CircleSkyRegion, CirclePixelRegion

from hetdex_api.survey import Survey, FiberIndex
from hetdex_api.config import HDRconfig
from hetdex_api.detections import Detections
from hetdex_api.elixer_widget_cls import ElixerWidget

from hetdex_tools.get_spec import get_spectra

import pandas as pd
import seaborn as sb
```

[2]:
```
det_object = Detections('hdr2.1', loadtable = False)
```

[3]:
```
# Once it has loaded you want to filter out the data by selecting those that␣
↪are in the NEP field
# to do this I will give you the verticies of a box that will encompass all the␣
↪NEP field - Oscar

# The center of the NEP field is given by:
# NEP Central Coordinates:
# R.A. = 18hours00minutes00seconds, decl. = 66 degree 33minute 38.552 arcmin
```

```
# Then make a radius of 3.5 degrees centered above and find all the RA and DEC␣
↪coordinates
# in the DF that are within this circle

# creating the circle region in the sky (NEP field)
ra = '18h00m00s'
dec = '+66d33m38.552s'
center_sky_coords = SkyCoord(ra, dec, frame = 'icrs')

maskregion = det_object.query_by_coords(center_sky_coords, 3.5 * u.deg)
detects_in_NEP = det_object[maskregion]     # Sources within the NEP footprint
```

[8]:
```
spec_test = spec_table = detects_in_NEP.get_spectrum(detects_in_NEP.detectid[0])
```

[9]:
```
spec_test
```

[9]:
```
<Table length=1036>
 wave1d            spec1d                       spec1d_err
Angstrom 1e-17 erg / (Angstrom cm2 s) 1e-17 erg / (Angstrom cm2 s)
float32           float32                      float32
-------- ---------------------------- ----------------------------
  3470.0                    1.3739702                     3.301174
  3472.0                    1.3739702                     3.301174
  3474.0                    1.3739702                     3.301174
  3476.0                    1.3739702                     3.301174
  3478.0                    1.3739702                     3.301174
  3480.0                    1.3739702                     3.301174
  3482.0                    1.3739702                     3.301174
  3484.0                    1.3739702                     3.301174
  3486.0                    1.3739702                     3.301174
     …                           …                            …
  5522.0                    1.9421544                    1.8177477
  5524.0                    1.9345994                    1.8484716
  5526.0                    1.9345994                    1.8484716
  5528.0                    1.9345994                    1.8484716
  5530.0                    1.9345994                    1.8484716
  5532.0                    1.9345994                    1.8484716
  5534.0                    1.9345994                    1.8484716
  5536.0                    1.9345994                    1.8484716
  5538.0                    1.9345994                    1.8484716
  5540.0                    1.9345994                    1.8484716
```

[4]:
```
num_detects = np.size(detects_in_NEP.detectid)    # getting the number of␣
↪detections in NEP which is 69799

# this part is creating a 2d list which will then be inputting into a␣
↪dictionary to be converted into an astropy Table
```

```
# 2d list because I wanted a list of lists to hold all the spec1d and spec1d␣
 ↪error values
rows, cols = (num_detects, 1)
spec_ls = [[0 for i in range(cols)] for j in range(rows)]

# going from a list to an astropy array
for i in range(len(spec_ls)):
    spec_table = detects_in_NEP.get_spectrum(detects_in_NEP.detectid[i])
    spec_ls[i] = np.array(spec_table['spec1d'])    # turned into array because␣
 ↪astropy column ['spec1d'] was confusing to work with
```

[5]:
```
# same code as above except this one is for the spec1d error column
# only change is getting erorr numbers so all past comments apply to here too
# I would've put them in the same block but these blocks take awhile to finish␣
 ↪running so I didn't want to overload
specErr_ls = [[0 for i in range(cols)] for j in range(rows)]

# going from a list to an astropy array
for i in range(len(specErr_ls)):
    spec_table = detects_in_NEP.get_spectrum(detects_in_NEP.detectid[i])
    specErr_ls[i] = np.array(spec_table['spec1d_err'])
```

Want to make a table where the first row is the detectid, then I have rows for, spect1d (array), spec1d error (array), line info like EW, line flux, magnitude, RA, and DEC.

[6]:
```
# Using a dict of column data to initialize an astropy Table
dic_to_table = {'detects': detects_in_NEP.detectid,
               'wavelength': detects_in_NEP.wave,
                'ra': detects_in_NEP.ra,
                'dec': detects_in_NEP.dec,
                'spec1d': spec_ls,
               'spec1d_err': specErr_ls}

detect_info = Table(dic_to_table)
```

[7]:
```
detect_info
```

[7]:
```
<Table length=69799>
 detects   wavelength …       spec1d [1036]            spec1d_err [1036]
  int64     float32   …          float32                    float32
---------- ---------- … ------------------------- -----------------------
2100395300     4795.6 …     1.3739702 .. 1.9345994   3.301174 .. 1.8484716
2100395301     3684.0 …  0.25948396 .. 0.104944706  4.9196553 .. 1.5617634
2100395303    4407.54 … -0.18306294 .. 0.120995596    3.343359 .. 1.752679
2100395308    3779.01 …      3.5947866 .. 29.83107  5.0278735 .. 3.9272287
2100395309    3779.08 …       7.810404 .. 8.485255  4.6782527 .. 2.2743156
2100395310    3777.96 …     5.4102297 .. 16.955706    5.465975 .. 3.143807
```

2100395312	3540.37 …	7.232119 .. 36.326458	5.369271 .. 4.3121314
2100395314	3779.73 …	5.690473 .. 39.514835	4.060331 .. 5.03607
2100395323	3783.45 …	6.0763054 .. 14.341705	4.181327 .. 2.8485444
2100395329	3911.59 …	3.6374705 .. 9.342827	3.9804099 .. 2.4529374
…	… …	…	…
2102591133	3621.42 …	2.9745781 .. -0.15291205	8.845858 .. 2.1556869
2102591135	3798.57 …	6.340416 .. 43.41358	4.712274 .. 6.132177
2102591136	4496.66 …	0.41709733 .. -0.05633008	4.158766 .. 0.99008995
2102591137	3960.42 …	-0.2628514 .. 0.08616114	5.2677093 .. 1.1210852
2102591139	4664.4 …	-0.28486162 .. 0.3934254	3.0511537 .. 0.97738653
2102591140	4358.55 …	0.25720462 .. -0.06428938	4.398488 .. 1.6613194
2102591141	4198.65 …	0.8538195 .. 0.037515007	4.3777156 .. 1.1644461
2102591142	5337.0 …	-0.72364336 .. 0.034434147	3.8486855 .. 0.986437
2102591144	4173.24 …	1.5328524 .. 24.75373	4.7959304 .. 4.3725624
2102591145	3507.24 …	2.864891 .. 0.13571836	7.040993 .. 1.7914823

[ ]:
```python
# Once you have selected sources within the NEP footprint we can then go ahead and find some
# spectra from these sources - Oscar
spectra = detects_in_NEP.hdfile.root.Spectra
```

center_ksy.separations(skycoords entire) return indeces return distance return 3d

separate by dist mask

main goal of algorithm want it to distinguish lae vs o2 emitter. wouldn't impose cuts unless training.

cut = filter cut = signal to noise could do plya

might need cuts for taining for confident lya and o2

increase confidence by visually inspecting

could visually inspect to increase confidence.

plotting histograms to look for outliers

hetdex isn't perfect and it catches emission lines that aren't real. visual inspections helps

no need for dataframes if i found another way

save as csv with astropy table

csv into get_spec()

```
has nice documentation
```

get_spectrum good for ids ****USE*** returns all fiber spec with corresponding weights.

```
try to see if can get LAE samples. O2 samples. and ambigious samples. Clasify some of them. Si
```

detection object filter by fields. turn to astropy table and then filter.

Want psf weighted.

```
[ ]:
```