Assigment 3

This assigment focuses on getting comfortable with working with multidimensional data and linear regression. Key items include:

- Creating random n-dimensional data
- Creating a Model that can handle the data
- Plot a subset of the data along with the prediction
- Using a Dataset to read in and choose certain columns to produce a model
- Create several models from various combinations of columns
- Plot a few of the results

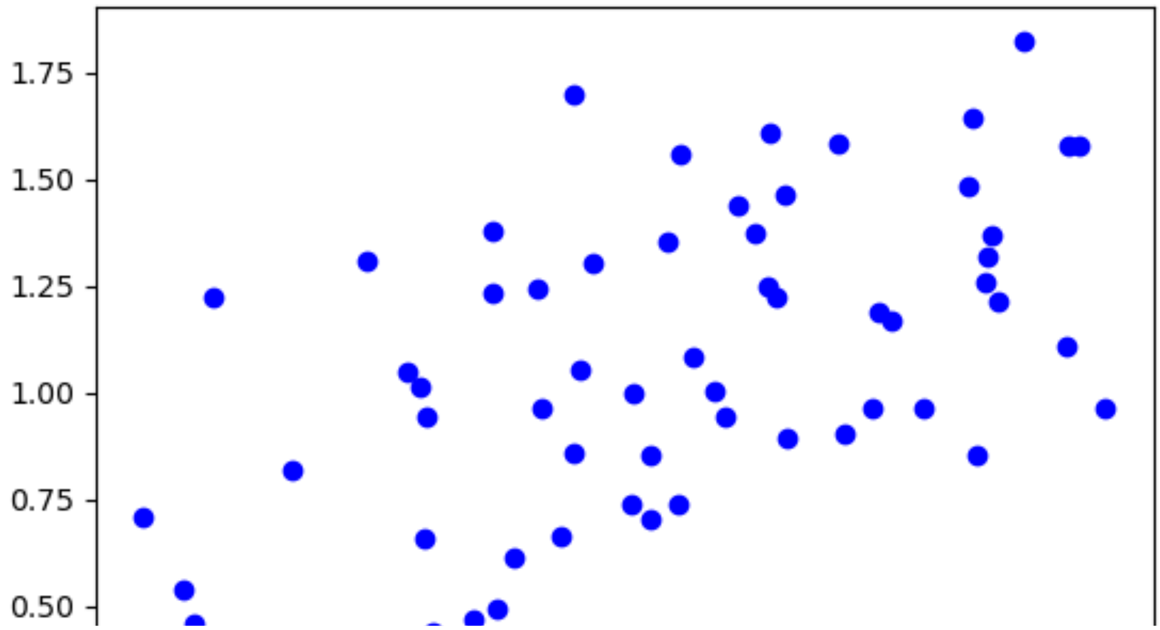
In [34]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

1. Create a 4 dimensional data set with 64 elements and show all 4 scatter 2D plots of the data x_1 vs. y, x_2 vs. y, x_3 vs. y, x_4 vs. y ¶

In [35]:
```
n = 64
x = np.linspace(0, 1, n) + np.random.rand(4, n)
x = np.vstack([x, np.ones(len(x.T))]).T
y = np.linspace(0, 1, n) + np.random.rand(n)
```

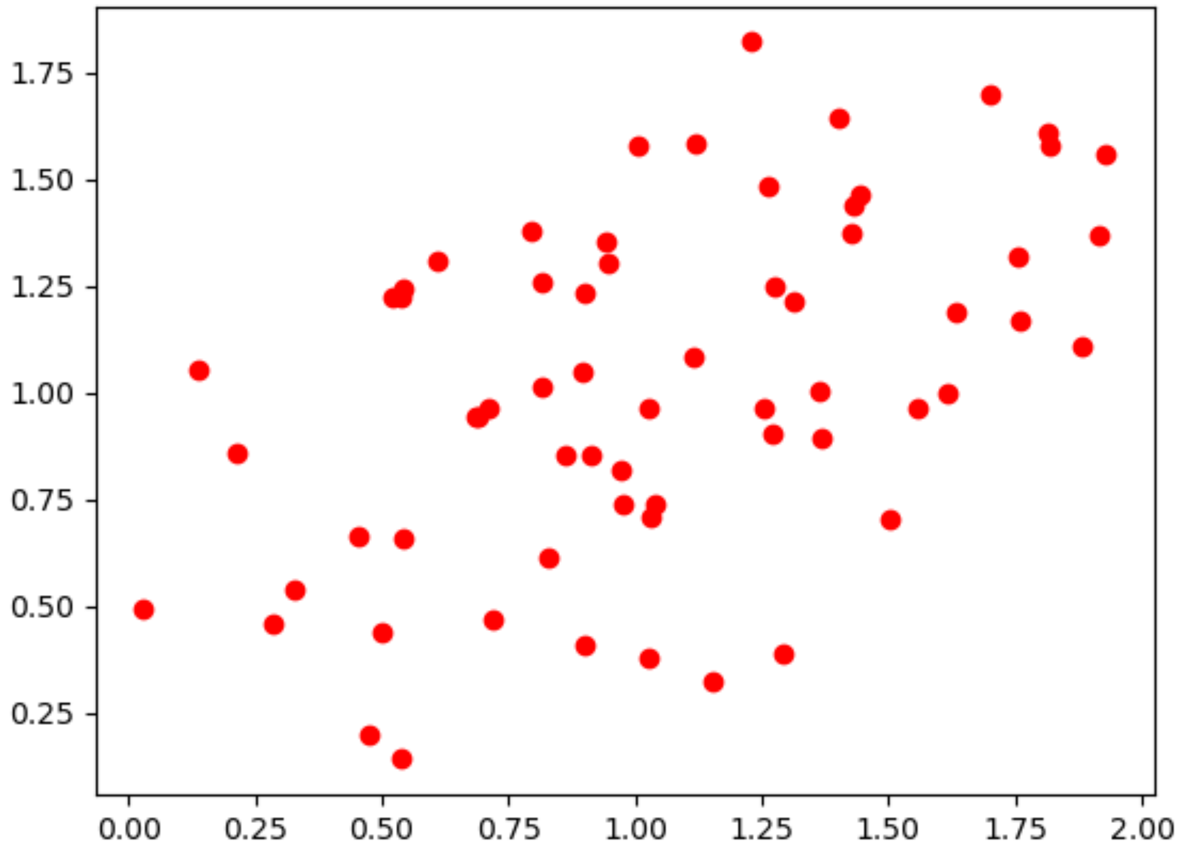
In [36]:
```
plt.scatter(x.T[0],y, c = "blue")
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x7fc8691da950>
```



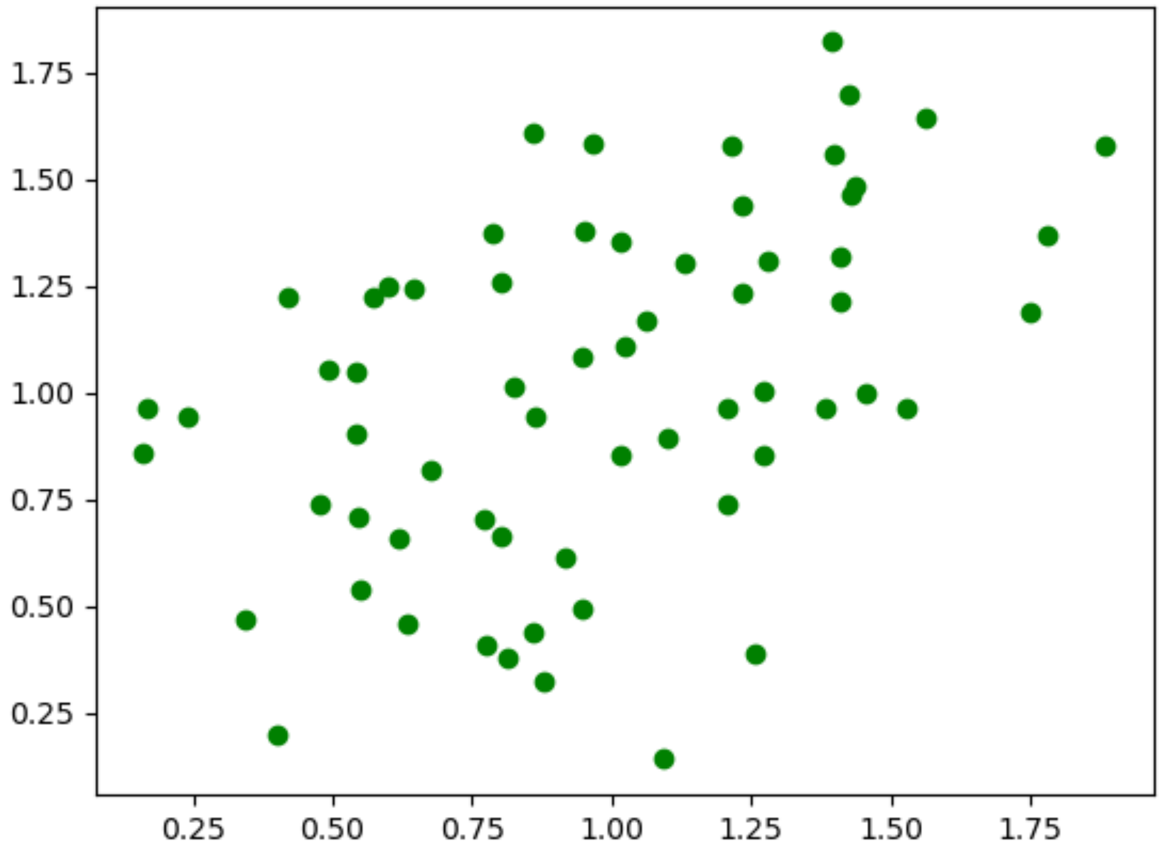
In [37]:
```
plt.scatter(x.T[1],y, c = "red")
```

Out[37]: <matplotlib.collections.PathCollection at 0x7fc86978cd00>



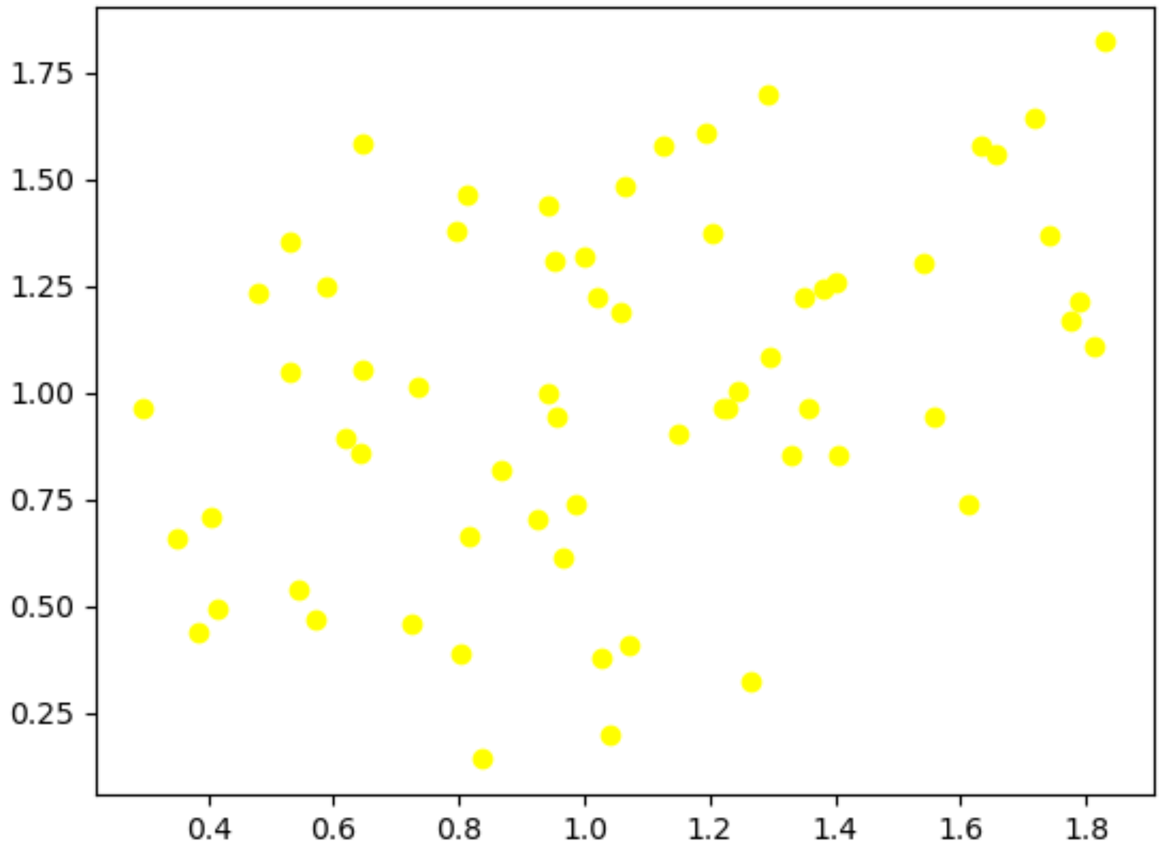
In [38]:
```
plt.scatter(x.T[2],y, c = "green")
```

Out[38]: <matplotlib.collections.PathCollection at 0x7fc869818eb0>



In [39]:
```
plt.scatter(x.T[3],y, c = "yellow")
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x7fc869c8a650>
```



2. Create a Linear Regression model (LIKE WE DID IN CLASS) to fit the data. *Use the example from Lesson 3 and DO NOT USE a library that calculates automatically*. We are expecting 5 coefficients to describe the linear model.

After creating the model (finding the coefficients), calculate a new column $y_p = \Sigma \beta_n \cdot x_n$

$\beta = (X^T X)^{-1} Y^T X$

In [40]:
```
left  = np.linalg.inv(np.dot(x.T, x)) # Just showing the by hand metho
right = np.dot(y.T, x)
np.dot(left, right)
```

Out[40]:
```
array([ 0.49650196,  0.15756539,  0.06928155, -0.01402025,  0.2379810
9])
```

In [41]:
```
beta = np.dot(left, right)
beta
```

Out[41]:
```
array([ 0.49650196,  0.15756539,  0.06928155, -0.01402025,  0.2379810
9])
```

In [42]:
```
test = np.linalg.lstsq(x,y,rcond=None)[0] # Got a warning that didn't
test
```

Out[42]:
```
array([ 0.49650196,  0.15756539,  0.06928155, -0.01402025,  0.2379810
9])
```

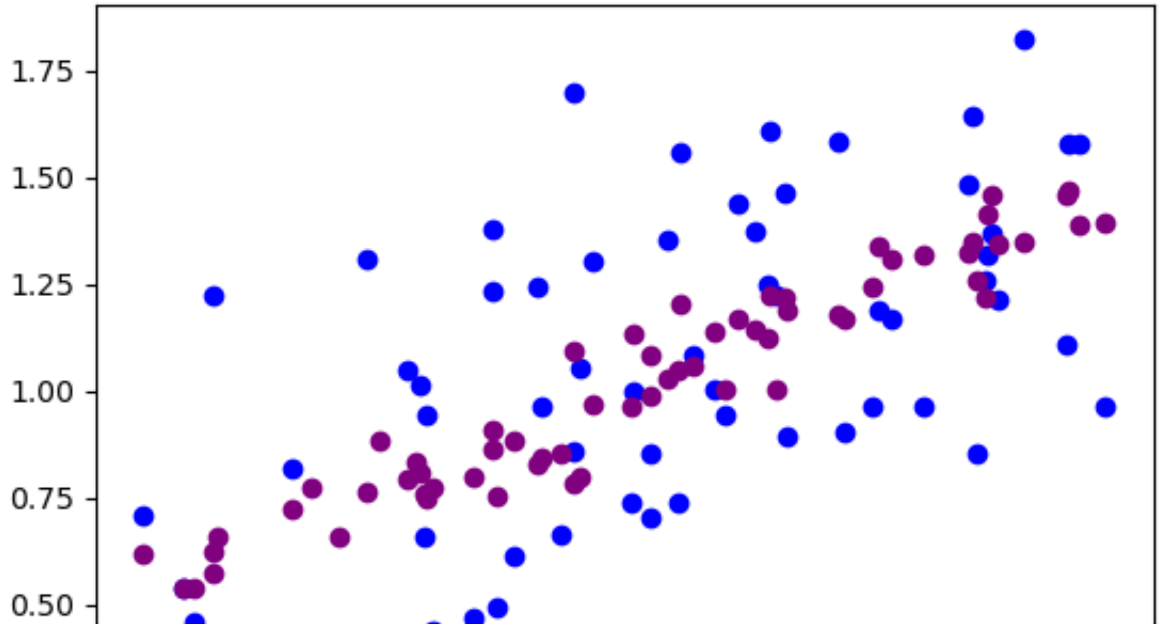
In [43]:
```
y_pred = np.dot(x, beta) # I know the question implies a single column
y_pred
```

Out[43]:
```
array([0.75210303, 0.8002714 , 0.84140595, 0.78093716, 0.79672387,
       0.65795759, 0.62496426, 0.8811053 , 0.53629223, 0.83108435,
       0.61943274, 0.54008205, 0.66027305, 0.74819913, 0.79338409,
       0.75838359, 0.77264783, 0.96211602, 0.77508068, 0.88108606,
       1.12434503, 0.80657736, 0.85111509, 0.57339169, 0.72314439,
       0.82925567, 0.90589558, 1.18766032, 1.02656337, 0.98906336,
       1.21869818, 1.00364501, 0.86525079, 1.17049904, 1.0588461 ,
       0.76255208, 1.13650141, 1.24490559, 1.14399017, 1.17835633,
       1.25947642, 1.00447843, 1.08103792, 1.04928551, 1.31823608,
       1.09388428, 1.21782605, 1.32338369, 1.13292071, 1.1658644 ,
       1.30896043, 1.34915632, 0.9668831 , 1.22130272, 1.47021375,
       1.34649749, 1.41422782, 1.39031924, 1.3373197 , 1.34192362,
       1.39123121, 1.20231874, 1.45639   , 1.45937604])
```

3. Plot the model's prediction as a different color on top of the scatter plot from Q1 in 2D for all 4 of the dimensions ($x_1 \rightarrow y_p, x_2 \rightarrow y_p, x_3 \rightarrow y_p, x_4 \rightarrow y_p$)

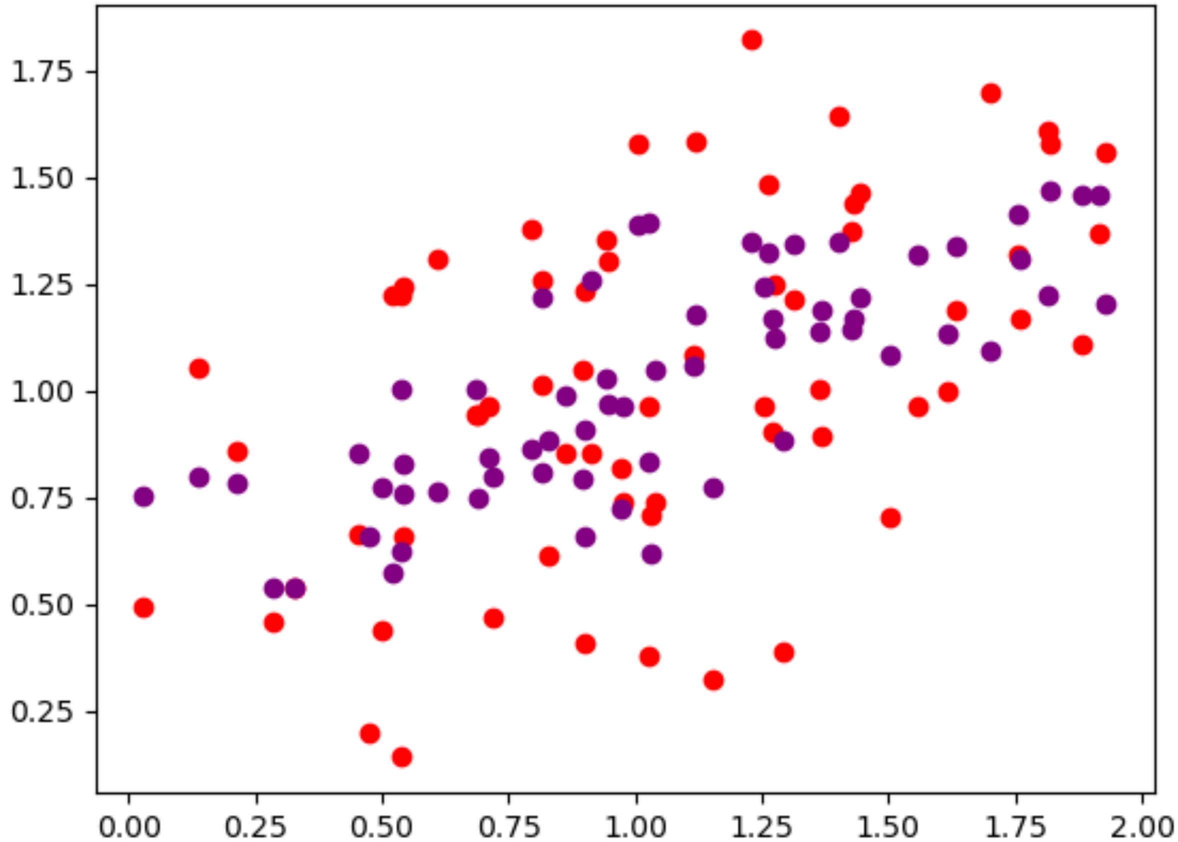
In [44]:
```
plt.scatter(x.T[0], y, c = "blue")
plt.scatter(x.T[0], y_pred, c = "purple")
```

Out[44]: <matplotlib.collections.PathCollection at 0x7fc869d15f60>



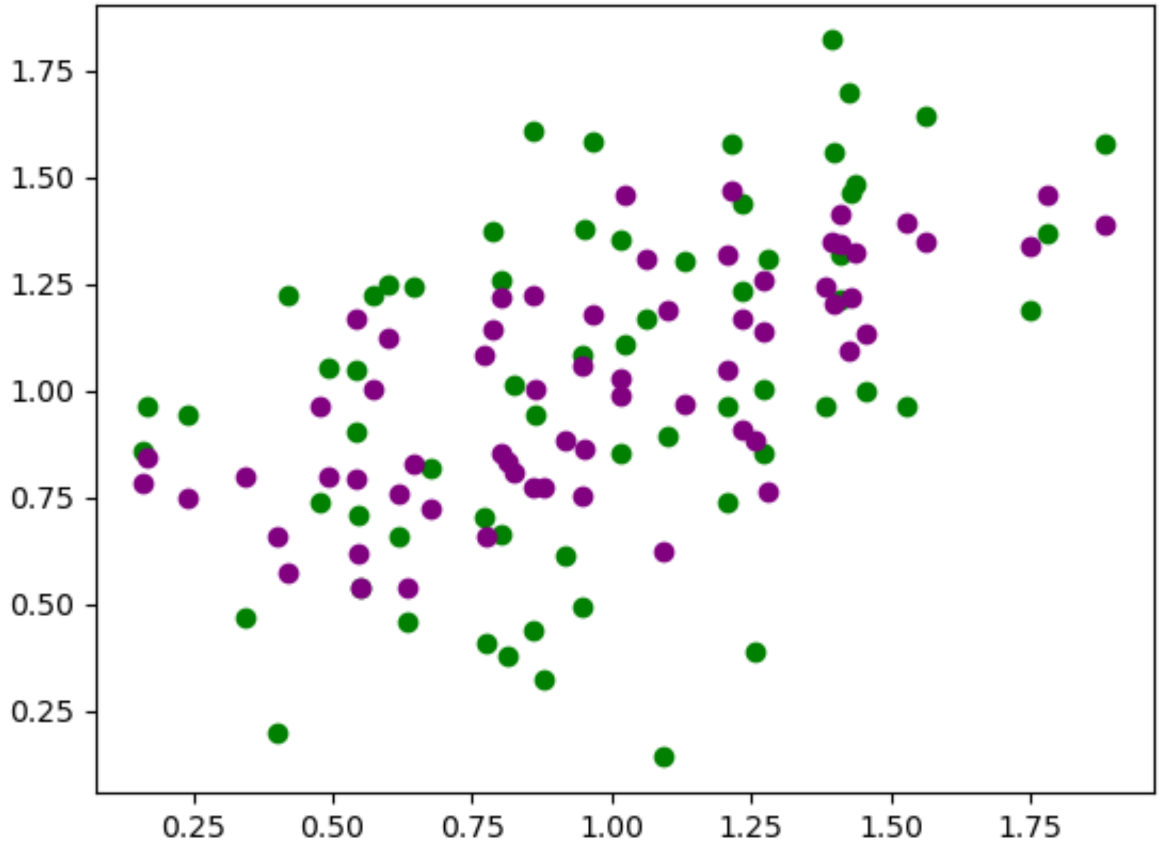
In [45]:
```
plt.scatter(x.T[1], y, c = "red")
plt.scatter(x.T[1], y_pred, c = "purple")
```

Out[45]: <matplotlib.collections.PathCollection at 0x7fc869db12d0>



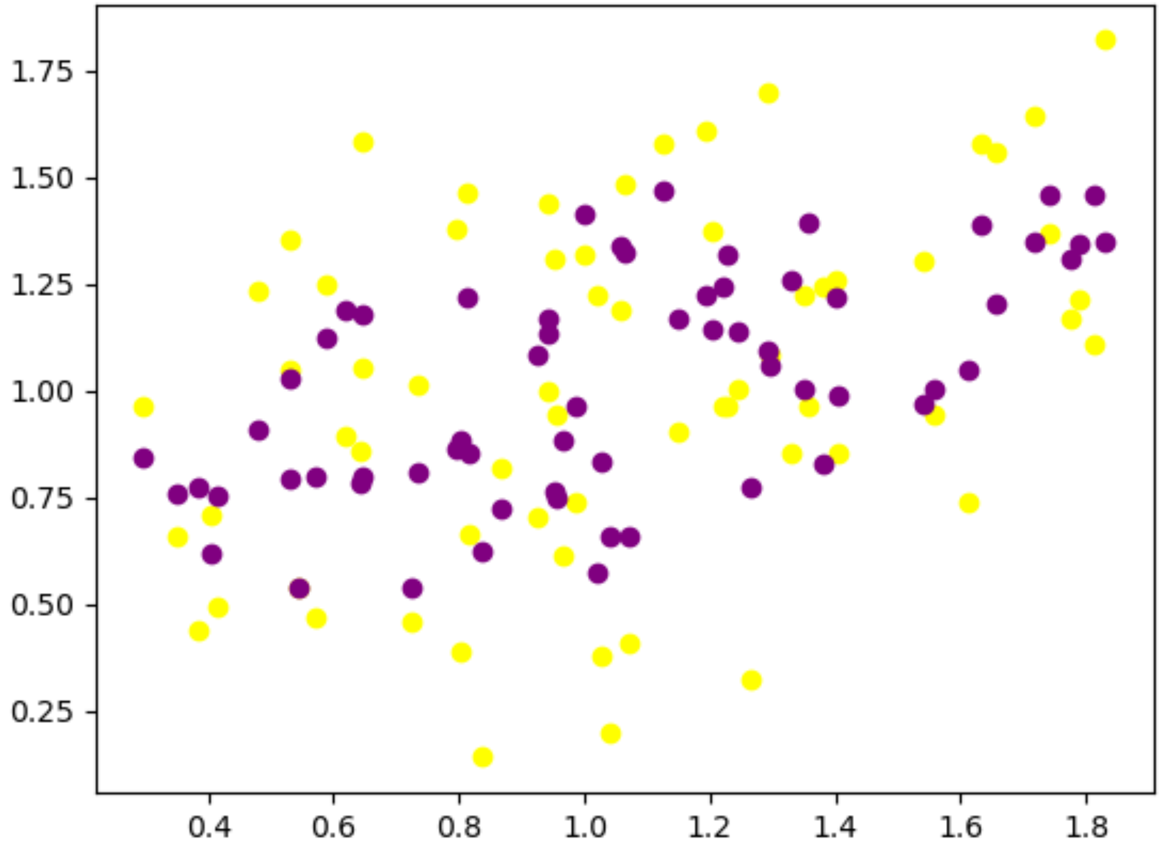
In [46]:
```
plt.scatter(x.T[2], y, c = "green")
plt.scatter(x.T[2], y_pred, c = "purple")
```

Out[46]: <matplotlib.collections.PathCollection at 0x7fc869e3c700>



In [47]:
```
plt.scatter(x.T[3], y, c = "yellow")
plt.scatter(x.T[3], y_pred, c = "purple")
```

Out[47]:
```
<matplotlib.collections.PathCollection at 0x7fc869cdf910>
```



4. Read in `mlnn/data/Credit.csv` with Pandas and build a Linear Regression model to predict Credit Rating (`Rating`). Use only the numeric columns in your model, but feel free to experiment which which columns you believe are better predicters of Credit Rating (Column `Rating`)

In [48]:
```
import pandas as pd
import numpy as np
credit = pd.read_csv('../data/Credit.csv')
credit.head()
```

Out[48]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethr
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Cauc
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	A
2	3	104.593	7075	514	4	71	11	Male	No	No	A
3	4	148.924	9504	681	3	36	11	Female	No	No	A
4	5	55.882	4897	357	2	68	16	Male	No	Yes	Cauc

Choose multiple columns as inputs beyond `Income` and `Limit` but clearly, don't use `Rating`

In [49]:
```
columns = ['Income', 'Limit', "Balance", "Education"]
X = credit[columns].values

X = np.vstack([X.T, np.ones(len(X))]).T
X
```

Out[49]:
```
array([[1.48910e+01, 3.60600e+03, 3.33000e+02, 1.10000e+01, 1.00000e+
00],
       [1.06025e+02, 6.64500e+03, 9.03000e+02, 1.50000e+01, 1.00000e+
00],
       [1.04593e+02, 7.07500e+03, 5.80000e+02, 1.10000e+01, 1.00000e+
00],
       ...,
       [5.78720e+01, 4.17100e+03, 1.38000e+02, 1.20000e+01, 1.00000e+
00],
       [3.77280e+01, 2.52500e+03, 0.00000e+00, 1.30000e+01, 1.00000e+
00],
       [1.87010e+01, 5.52400e+03, 9.66000e+02, 7.00000e+00, 1.00000e+
00]])
```

In [50]:
```
Y = credit['Rating']
Y
```

Out[50]:
```
0      283
1      483
2      514
3      681
4      357
      ...
395    307
396    296
397    321
398    192
399    415
Name: Rating, Length: 400, dtype: int64
```

In [51]:
```
Left  = np.linalg.inv(np.dot(X.T, X))
Right = np.dot(Y.T, X)
np.dot(Left, Right)
```

Out[51]:
```
array([ 1.34036346e-01,  6.26373011e-02,  1.48796968e-02, -3.43052578
e-01,
        4.91302144e+01])
```

In [52]:
```
Beta = np.dot(Left, Right)
Beta
```

Out[52]:
```
array([ 1.34036346e-01,  6.26373011e-02,  1.48796968e-02, -3.43052578
e-01,
        4.91302144e+01])
```

In [53]:
```
Y_pred = np.dot(X, Beta)
Y_pred
```

Out[53]:
```
array([278.17761811, 487.85686136, 511.16502905, 674.96680224,
       362.79163541, 577.61561585, 263.0635652 , 514.19086427,
       257.5523597 , 499.34831911, 582.14720802, 127.77545175,
       393.04857285, 501.55806736, 255.57807578, 204.8362672 ,
       283.11733697, 328.57804889, 465.82126298, 482.31153598,
       226.48079068, 462.20856298, 213.19296052, 385.48187672,
       155.47798223, 326.29527451, 287.49777934, 338.98316868,
       936.47734621, 412.30810878, 419.73534685, 218.70826299,
       561.43354841, 164.46763043, 212.76660943, 214.60414135,
       469.50962871, 472.28062691, 299.40460335, 268.06298717,
       258.15033424, 556.14400174, 354.6594563 , 455.72148363,
       463.88996014, 544.32210569, 380.60485023, 339.37361198,
       191.01119404, 350.10961244, 383.23870869, 299.33814009,
       400.78019142, 404.5003749 , 141.99123487, 162.66795394,
       353.27423029, 355.8734942 , 267.6188798 , 389.75404498,
```

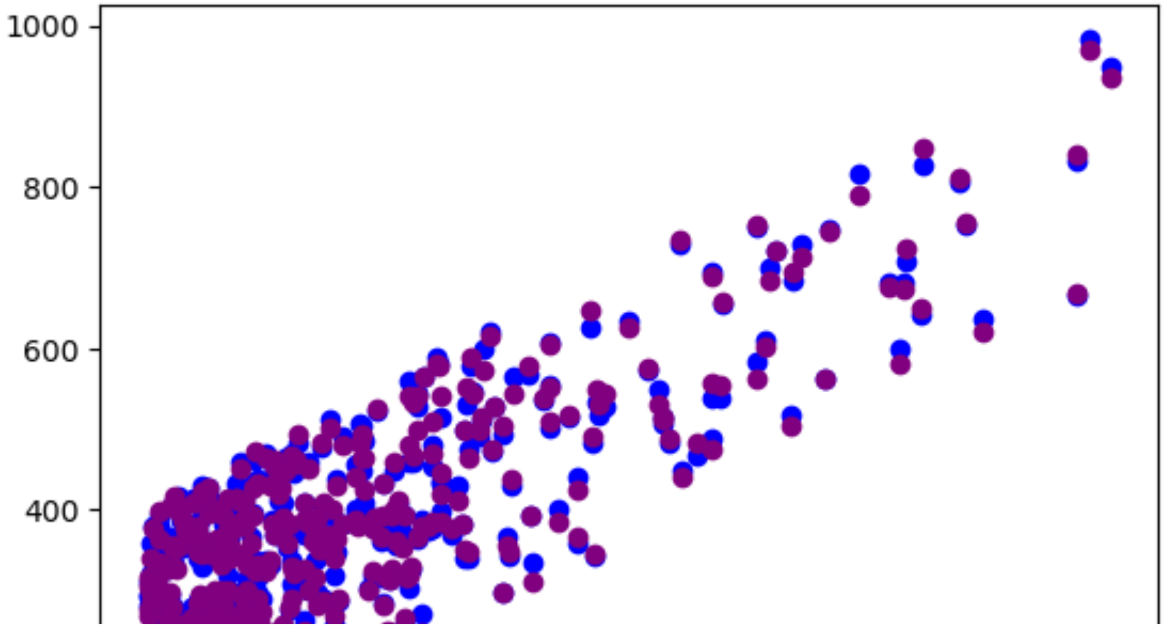
```
382.00657332, 243.63998293, 155.50827969, 234.95815063,
232.26132413, 315.46466455, 688.41314318, 377.75648656,
411.90704155, 494.21589308, 301.32955443, 532.94598227,
364.25860007, 339.45103433, 399.53444234, 247.68497993,
261.92646539, 252.76994453, 483.3380427 , 177.98644201,
268.15774437, 319.64859413, 333.68518773, 136.66357957,
233.21657558, 847.6797066 , 460.653058  , 188.83661504,
326.86559965, 542.42767802, 422.79894632, 440.55348034,
226.17870666, 400.22616552, 241.68871413,  98.74098094,
402.86458489, 261.31377518, 240.36653698, 605.74286773,
286.63901347, 204.86051343, 551.38150838, 676.76179507,
356.8775492 , 247.00448774, 129.98485464, 250.00609821,
440.07005592, 253.10939118, 253.51349154, 234.53049863,
481.78201877, 465.72514647, 258.55767412, 361.8080268 ,
180.95953616, 646.31154759, 182.28173705, 135.04441213,
135.453206  , 582.20648046, 509.16762189, 127.33867094,
207.38596852, 206.2158892 , 414.98988556, 266.6796172 ,
603.20844031, 266.99853706, 300.4554146 , 142.26517742,
402.29641254, 428.14969276, 426.54386041, 268.95196303,
312.64373119, 280.00670477, 178.06865412, 735.3720704 ,
450.19843828, 490.06896018, 531.69650083, 365.56843727,
219.79776128, 349.20194754, 378.3251802 , 140.85766724,
198.97411262, 102.99832206, 420.32812053, 360.02417014,
186.02059852, 344.52790919, 248.64156322, 133.97341342,
323.65594897, 412.17955197, 408.04442413, 239.72911815,
364.16686926, 155.32651827, 541.38453671, 191.41936582,
437.84238474, 340.79936991, 229.16374416, 194.02554869,
225.67580361, 452.06546712, 177.29343858, 322.66546954,
349.0030855 , 354.99587679, 751.90523507, 183.13477257,
209.47428612, 301.39778017, 331.27595362, 541.25753498,
275.85304862, 383.33207038, 467.81292291, 311.00930024,
812.04470501, 333.33561904, 290.75405552, 183.60286409,
550.67643594, 329.63050497, 393.93549681, 683.87234506,
299.50066985, 713.16849516, 181.19406581, 396.0153092 ,
532.01983754, 300.22907488, 172.58173541, 317.12806865,
393.17167291, 528.80483946, 138.18919956, 498.71015904,
393.80329895, 298.51788543, 201.27507093, 340.28701115,
325.32045408, 650.12212368, 250.53199607, 392.37778769,
327.03522739, 385.90720742, 389.74220703, 316.66460832,
218.8075844 , 398.13538495, 153.18926869, 386.7002111 ,
430.32580198, 625.56601141, 462.3628928 , 344.4578648 ,
563.51441606, 417.34264303, 499.93943662, 412.12426136,
349.33941419, 544.38517441, 382.09791977, 357.31282351,
356.59685323, 189.59963171, 589.82094138, 230.29390498,
370.64282286, 381.21538262, 229.21675822, 273.64338546,
260.40881912, 104.19726594, 123.22890771, 480.01784941,
158.94127683, 172.91796779, 251.04241579, 186.54112158,
102.43477993, 146.30493168, 196.55292836, 251.35889252,
615.1937656 , 384.29390656, 467.56102496, 320.31619834,
158.92241716, 202.57712404, 210.29071118, 454.90334905,
```

```
       383.96487504, 669.34009074, 306.62719893, 271.43315252,
       379.02795165, 372.32035089, 365.95093231, 427.33704669,
       132.99463924, 408.25376408, 242.05686425, 363.00903039,
       290.02493348, 360.01787962, 432.14453857, 621.03977352,
       267.91858484, 371.66800553, 503.57221031, 249.52720579,
       393.38897894, 165.51915417, 578.49017118, 463.86146125,
       177.26226301, 149.74715504, 143.61499661, 248.5385157 ,
       389.07155891, 299.43789201, 253.88871538, 282.06273327,
       377.24840732, 791.38892595, 209.02034709, 136.35833911,
       378.44625849, 327.88895672, 213.10888625, 375.65716007,
       344.98463929, 274.34730683, 368.01434963, 368.72999247,
       539.37784862, 168.12017468, 289.11661927, 297.09152751,
       349.84669719, 505.2984537 , 369.93199939, 399.73389713,
       388.73736072, 549.24959325, 657.54726719, 297.19532046,
       525.94209643, 349.98837106, 140.22704585, 209.31949662,
       119.58617635, 242.4761289 , 271.17778843, 969.00779815,
       232.9630016 , 375.98306843, 721.49111148, 482.84883091,
       281.02984861, 543.39507536, 346.27779157, 301.42656992,
       385.46073808, 261.91476144, 353.97367037, 266.40502674,
       431.41090802, 100.23240313, 391.0545958 , 723.91314929,
       293.33290828, 297.07440337, 236.12768461, 285.4772064 ,
       387.23282968, 141.96598138, 396.32845024, 755.73740918,
       117.15868802, 380.93501616, 146.46871639, 380.75145697,
       514.23719098, 350.2884849 , 293.3124006 , 841.02766785,
       444.72448358, 208.67263147, 323.74982881, 335.8097898 ,
       432.47016447, 366.07460893, 376.19633198, 446.68041169,
       694.73200492, 474.49846307, 564.33907767, 277.16446312,
       424.8628462 , 574.07629351, 448.82711926, 183.89758884,
       295.68609825, 401.35304116, 358.80973565, 416.67375745,
       517.77888791, 147.6246584 , 365.36533206, 231.98757933,
       555.65908073, 574.54768079, 413.6793791 , 258.71695667,
       163.46564836, 414.26348911, 284.49383789, 133.25704559,
       494.17159194, 510.92522464, 745.39993745, 475.12867902,
       193.74803965, 130.61230125, 424.64727681, 311.43739929,
       292.63379843, 316.08411597, 207.88663946, 409.61769849])
```

5. Plot your results using scatter plots (just like in class). Show as many of your columns vs. credit rating that you can.

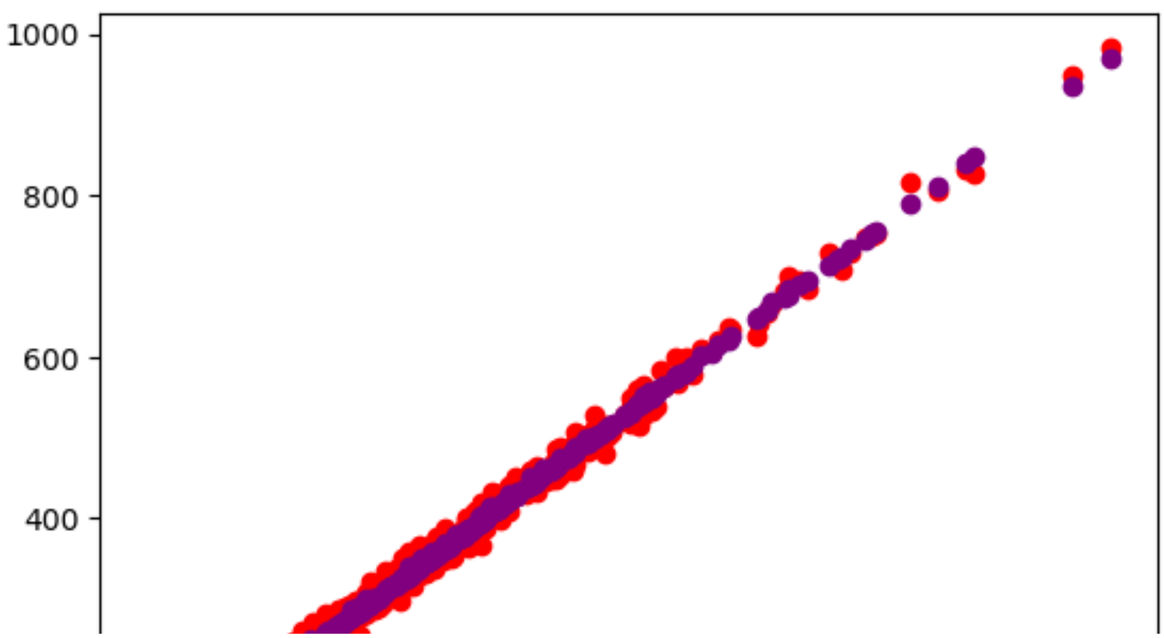
In [54]:
```
plt.scatter(X.T[0], Y, c = "blue")
plt.scatter(X.T[0], Y_pred, c = "purple")
```

Out[54]: <matplotlib.collections.PathCollection at 0x7fc86a1e6710>



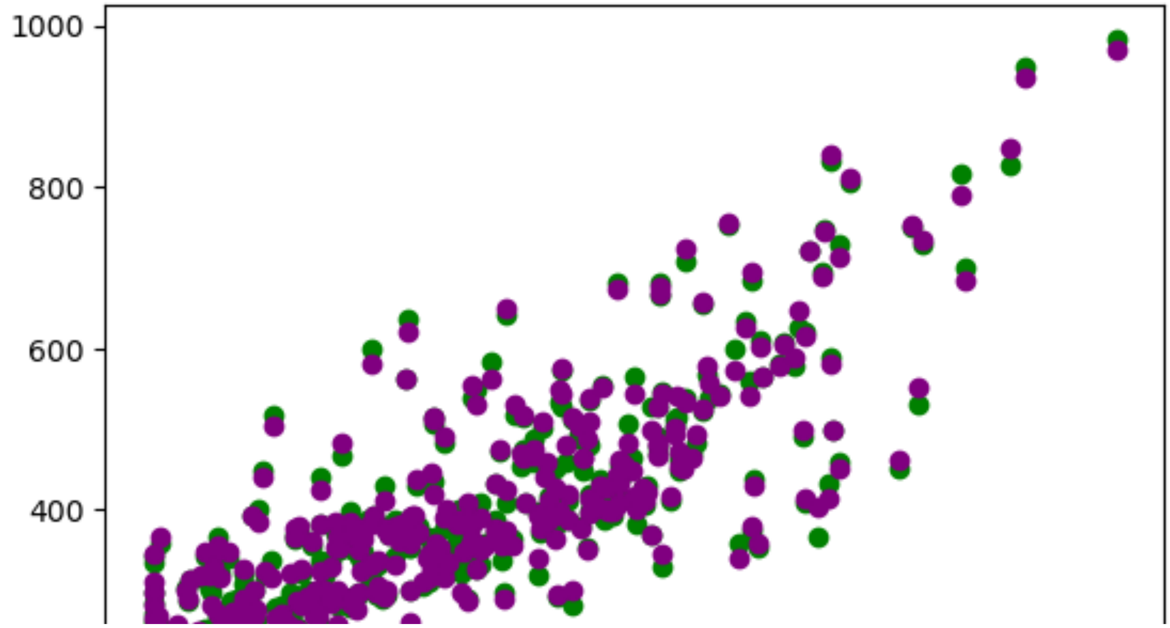
In [55]:
```
plt.scatter(X.T[1], Y, c = "red")
plt.scatter(X.T[1], Y_pred, c = "purple")
```

Out[55]: <matplotlib.collections.PathCollection at 0x7fc86a423b50>



In [56]:
```
plt.scatter(X.T[2], Y, color = "green")
plt.scatter(X.T[2], Y_pred, c = "purple")
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x7fc86a4a2e90>
```



In [57]:
```
plt.scatter(X.T[3], Y, color = "yellow")
plt.scatter(X.T[3], Y_pred, c = "purple")
```

Out[57]: <matplotlib.collections.PathCollection at 0x7fc86a8c9db0>