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

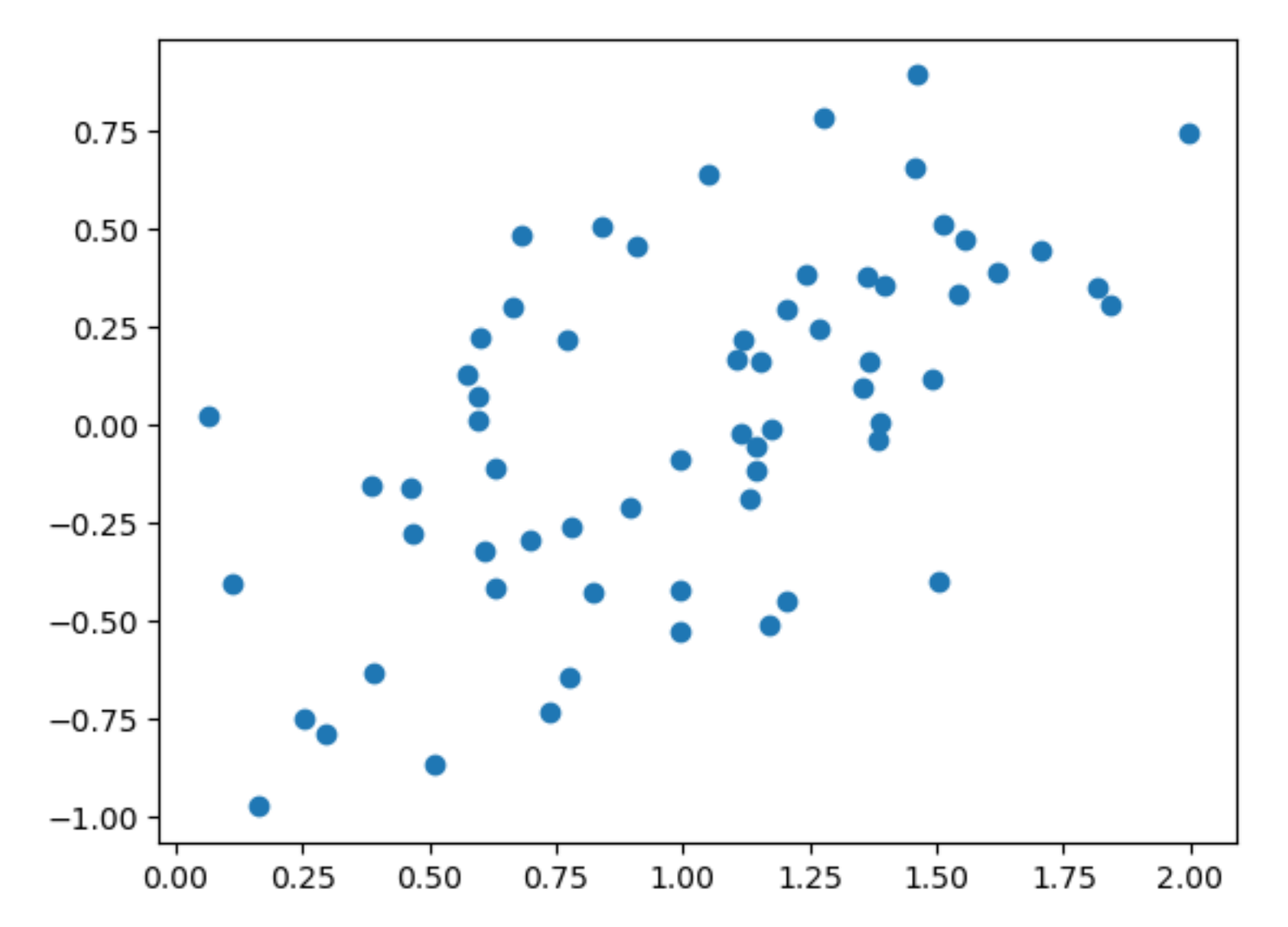
1. Create a 4 dimensional data set with 64 elements and show all 4 scatter 2D plots of the data x_1 vs. y, x_2 vs. y, x_3 vs. y, x_4 vs. y

In [13]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

In [14]:
```
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) - 1
```

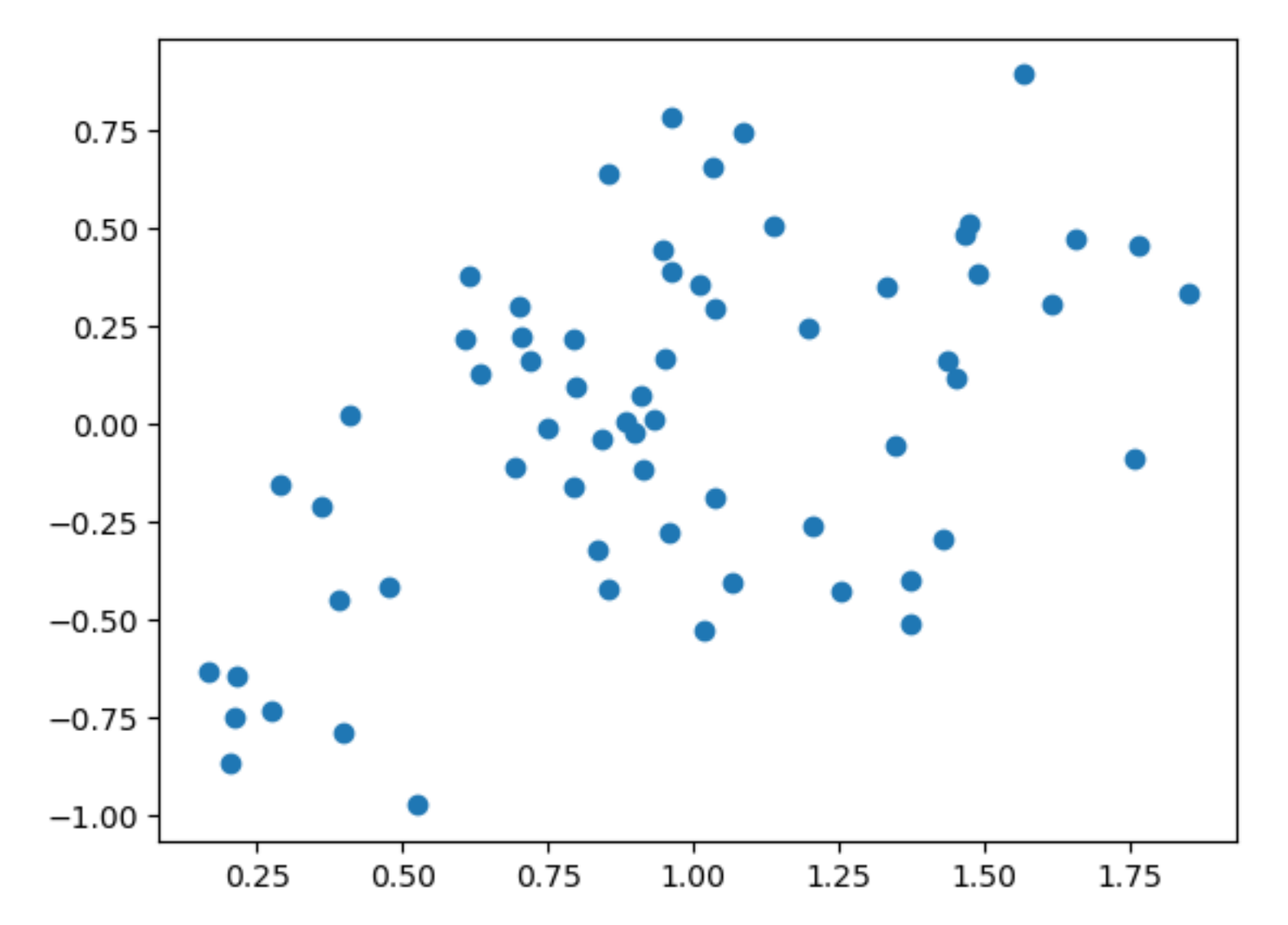
In [15]:
```
plt.scatter(x.T[0],y)
```

Out[15]: `<matplotlib.collections.PathCollection at 0x12190c340>`



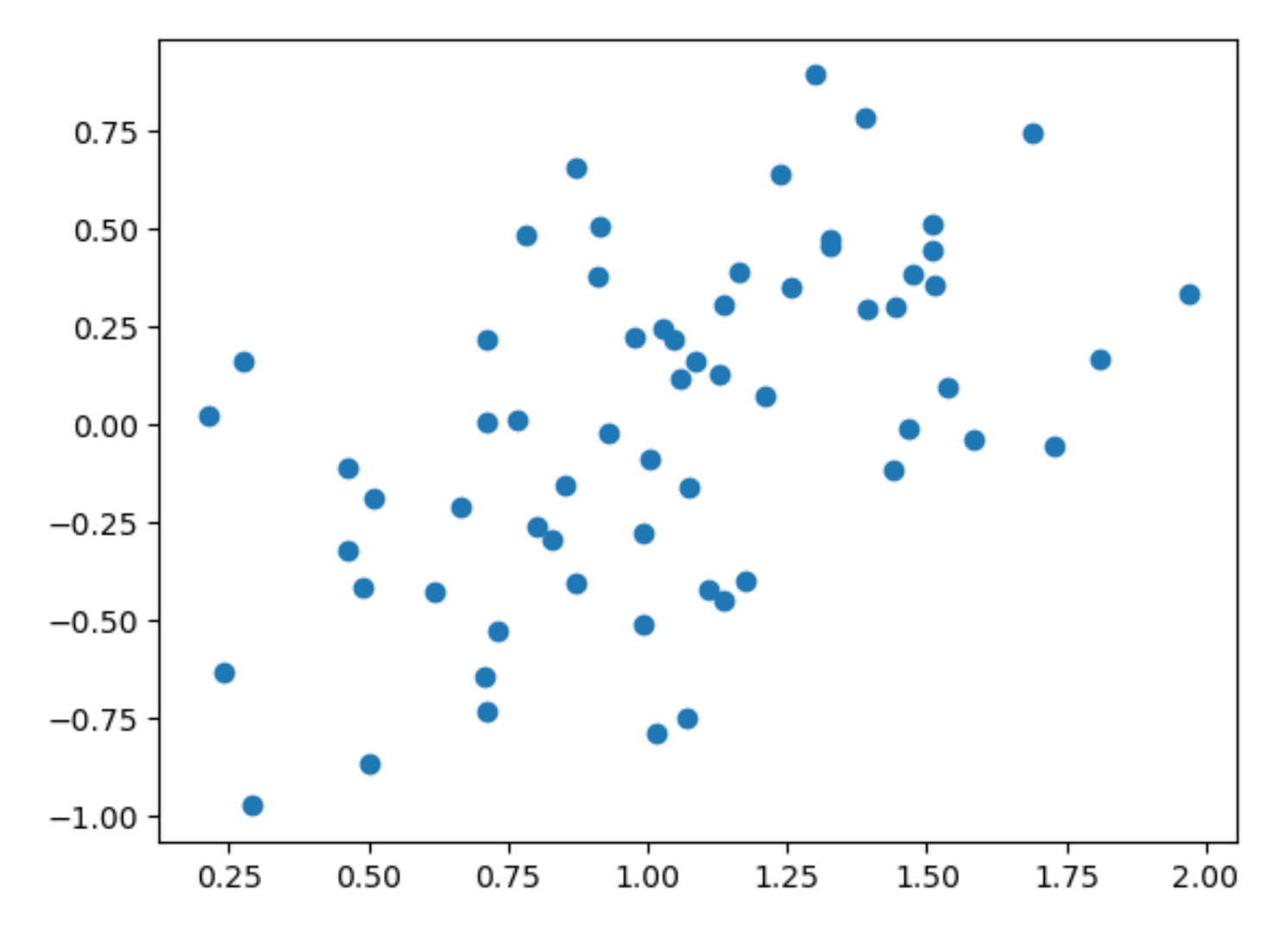
In [16]:
```
plt.scatter(x.T[1],y)
```

Out[16]: `<matplotlib.collections.PathCollection at 0x1219a0550>`



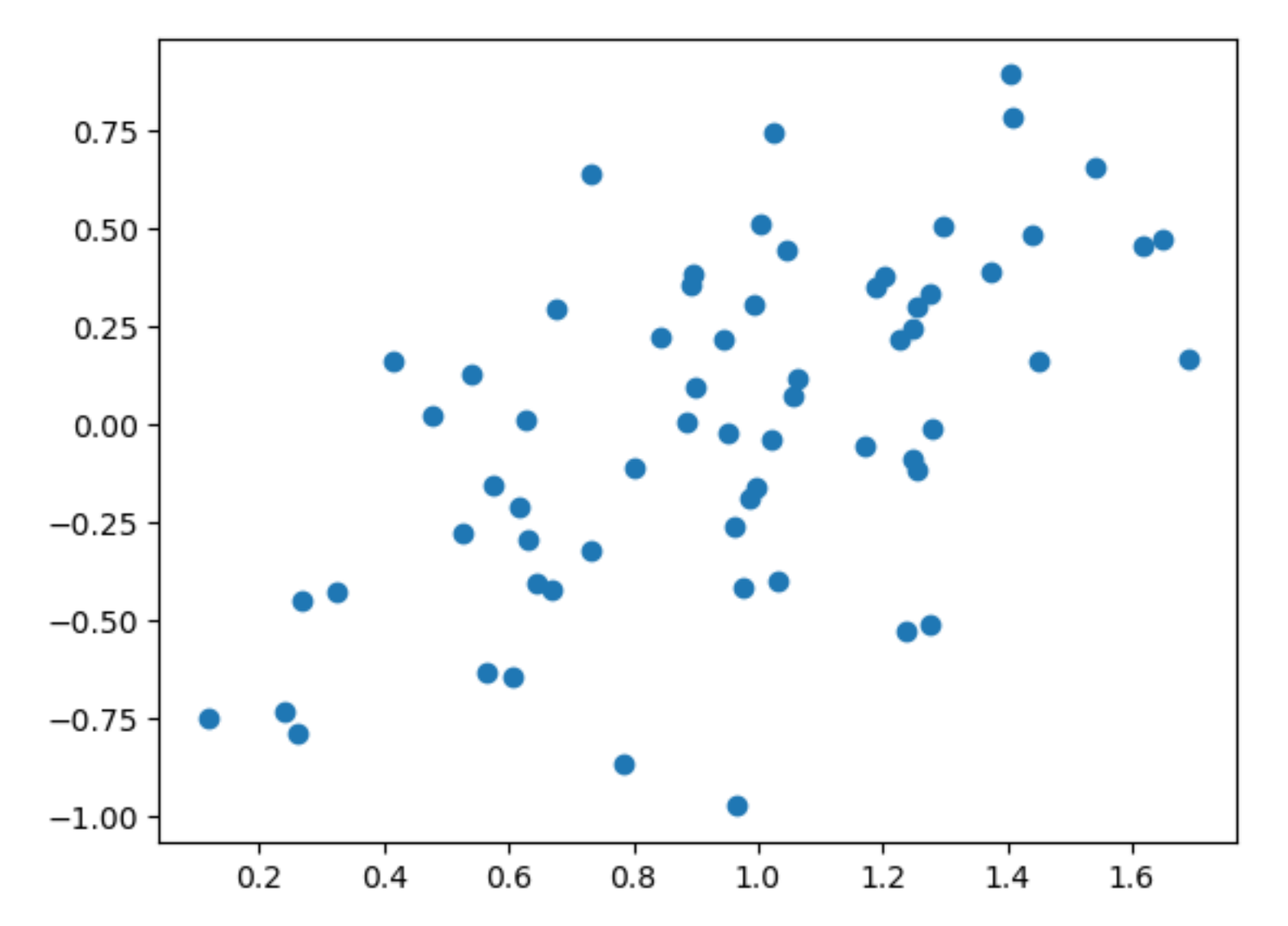
In [17]:
```
plt.scatter(x.T[2],y)
```

Out[17]: `<matplotlib.collections.PathCollection at 0x121a03550>`



In [18]:
```
plt.scatter(x.T[3],y)
```

Out[18]: `<matplotlib.collections.PathCollection at 0x121a8a980>`



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 $y_p = \Sigma\beta_n \cdot x_n$

In [24]:
```
beta_left = np.linalg.inv(np.dot(x.T, x))
beta_right = np.dot(y.T, x)
```

In [27]:
```
beta = np.dot(beta_left, beta_right)
beta
```

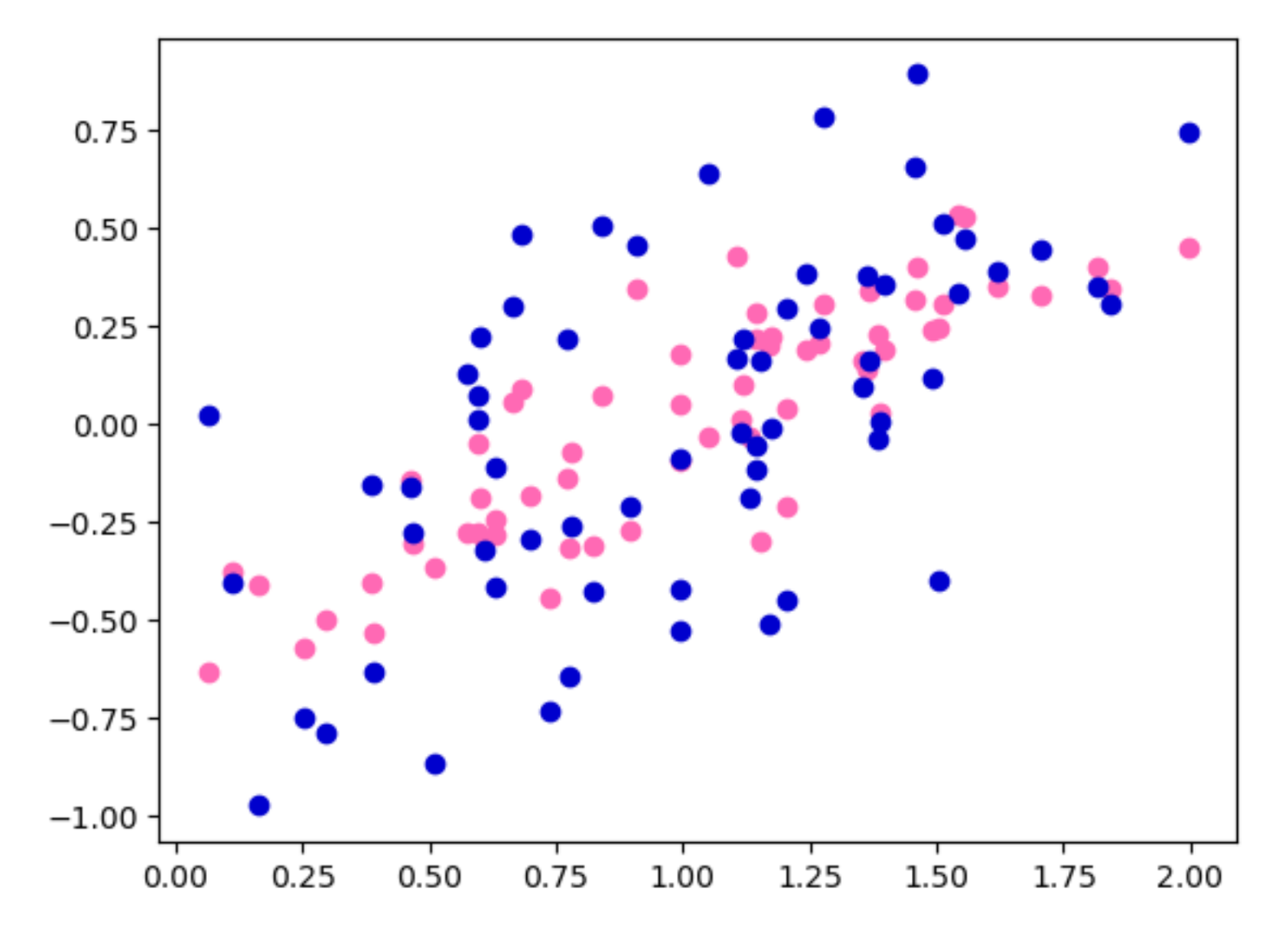
Out[27]:
```
array([ 0.28921849,  0.10199175,  0.17967162,  0.34509268, -0.89759578])
```

In [28]:
```
pred = np.dot(x,beta)
```

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 ($x_1 \rightarrow y_p, x_2 \rightarrow y_p, x_3 \rightarrow y_p, x_4 \rightarrow y_p$)

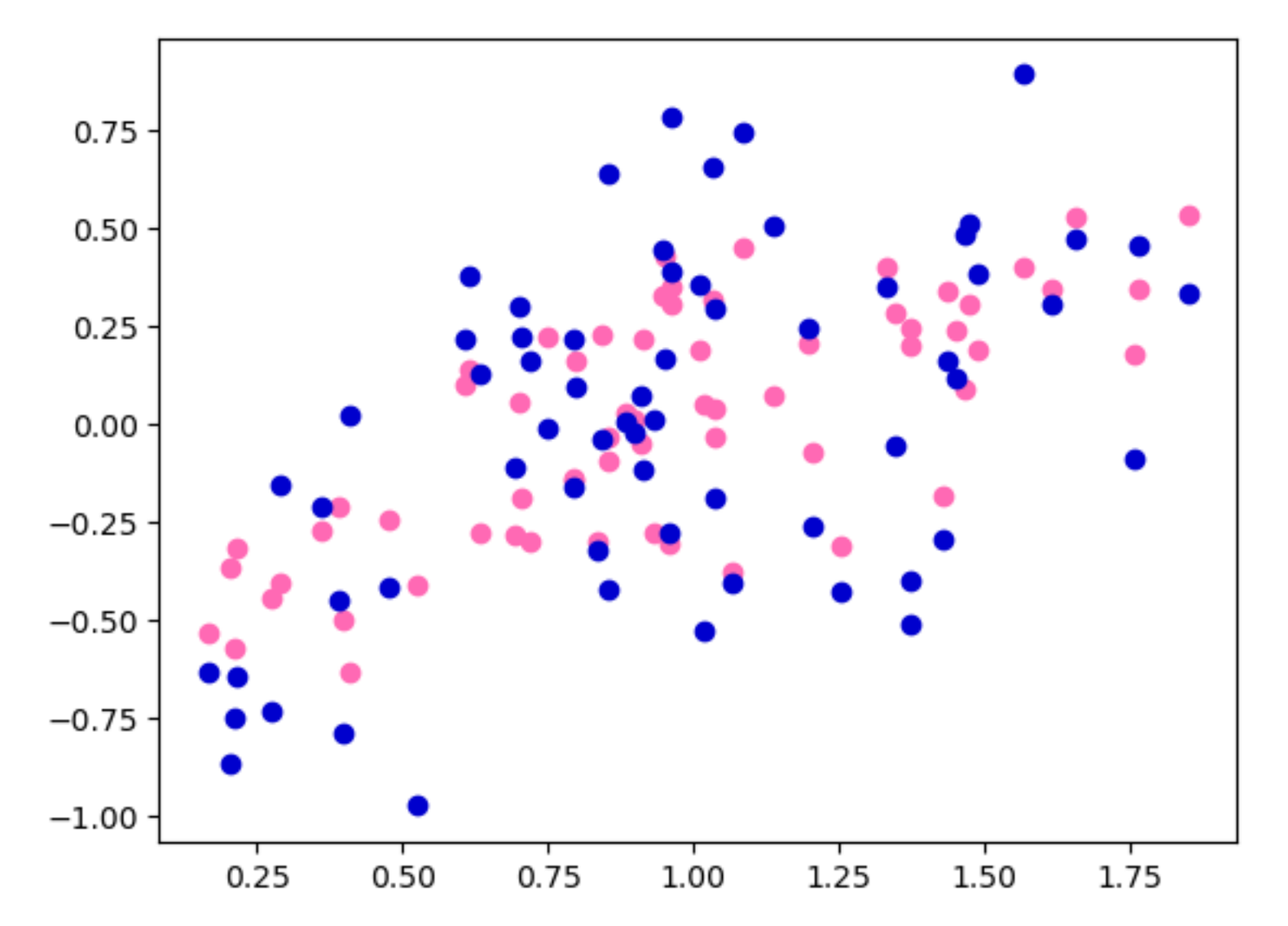
In [31]:
```
plt.scatter(x.T[0], pred, c='hotpink')
plt.scatter(x.T[0], y, c ='mediumblue')
```

Out[31]: `<matplotlib.collections.PathCollection at 0x121a34970>`



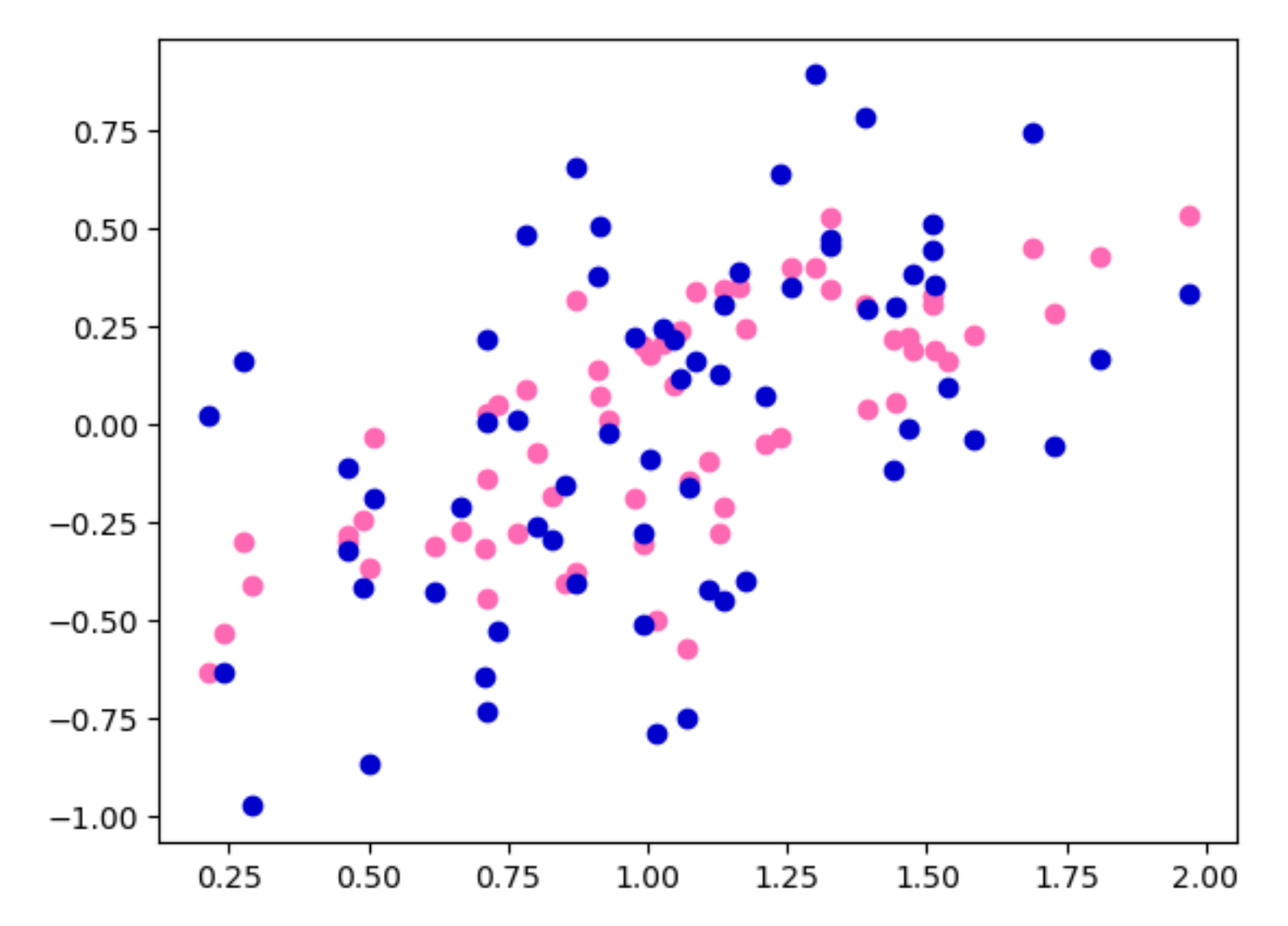
In [32]:
```
plt.scatter(x.T[1], pred, c='hotpink')
plt.scatter(x.T[1], y, c ='mediumblue')
```

Out[32]: `<matplotlib.collections.PathCollection at 0x1219c5030>`



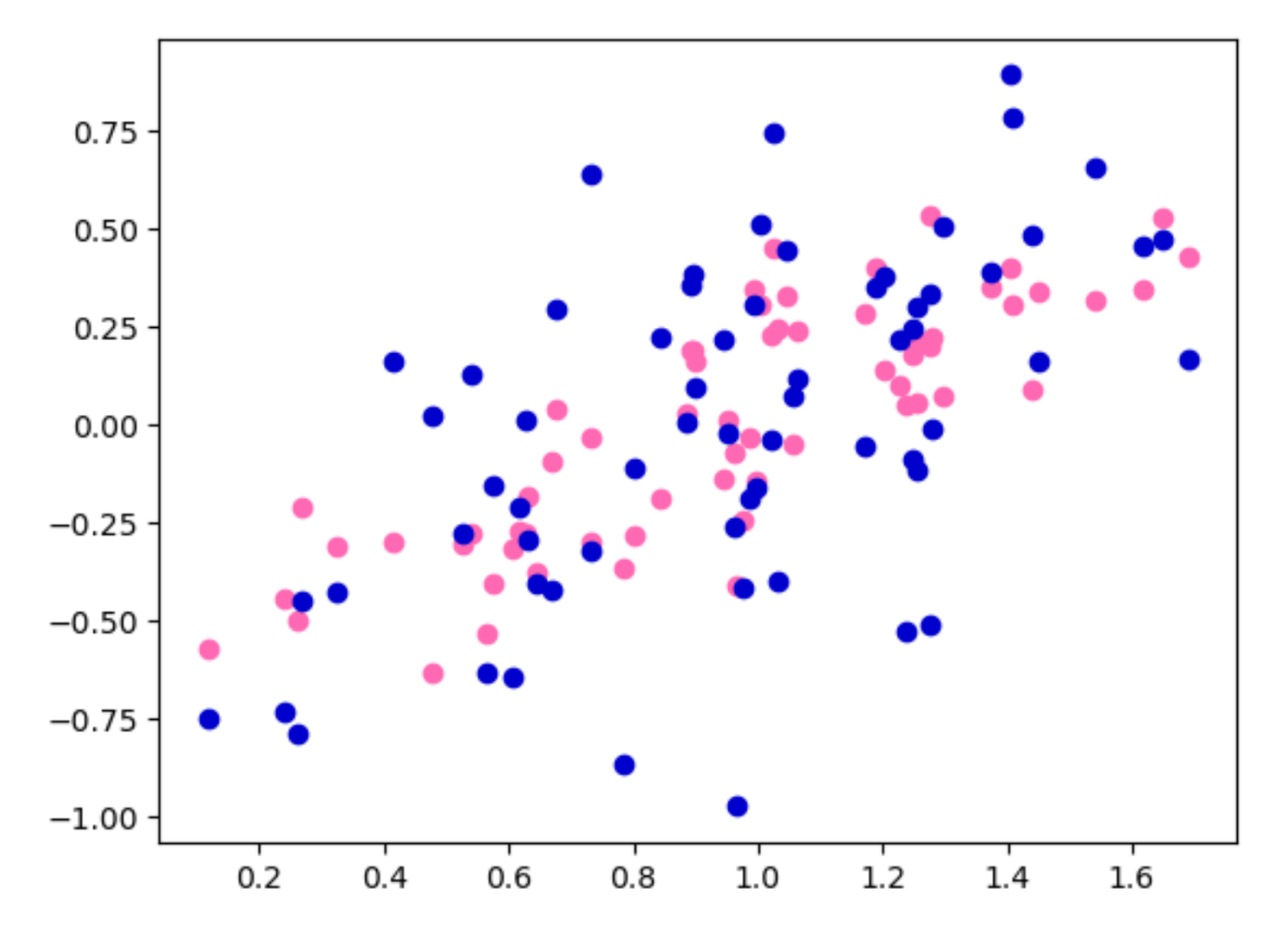
In [33]:
```
plt.scatter(x.T[2], pred, c='hotpink')
plt.scatter(x.T[2], y, c ='mediumblue')
```

Out[33]: `<matplotlib.collections.PathCollection at 0x121c5ab60>`



In [34]:
```
plt.scatter(x.T[3], pred, c='hotpink')
plt.scatter(x.T[3], y, c ='mediumblue')
```

Out[34]: `<matplotlib.collections.PathCollection at 0x12169f730>`



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 [56]:
```
import pandas as pd
import numpy as np
credit = pd.read_csv('../data/Credit.csv')
credit.head()
```

Out[56]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903
2	3	104.593	7075	514	4	71	11	Male	No	No	Asian	580
3	4	148.924	9504	681	3	36	11	Female	No	No	Asian	964
4	5	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331

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

In [68]:
```
columns = ['Income', 'Limit','Age','Education','Balance']
X = credit[columns].values

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

Out[68]:
```
array([[1.48910e+01, 3.60600e+03, 3.40000e+01, 1.10000e+01, 3.33000e+02,
        1.00000e+00],
       [1.06025e+02, 6.64500e+03, 8.20000e+01, 1.50000e+01, 9.03000e+02,
        1.00000e+00],
       [1.04593e+02, 7.07500e+03, 7.10000e+01, 1.10000e+01, 5.80000e+02,
        1.00000e+00],
       ...,
       [5.78720e+01, 4.17100e+03, 6.70000e+01, 1.20000e+01, 1.38000e+02,
        1.00000e+00],
       [3.77280e+01, 2.52500e+03, 4.40000e+01, 1.30000e+01, 0.00000e+00,
        1.00000e+00],
       [1.87010e+01, 5.52400e+03, 6.40000e+01, 7.00000e+00, 9.66000e+02,
        1.00000e+00]])
```

In [72]:
```
y = credit['Rating']
y
```

Out[72]:
```
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 [94]:
```
beta_left = np.linalg.inv(np.dot(X.T, X))
beta_right = np.dot(y.T, X)
```

In [95]:
```
beta = np.dot(beta_left, beta_right)
beta
```

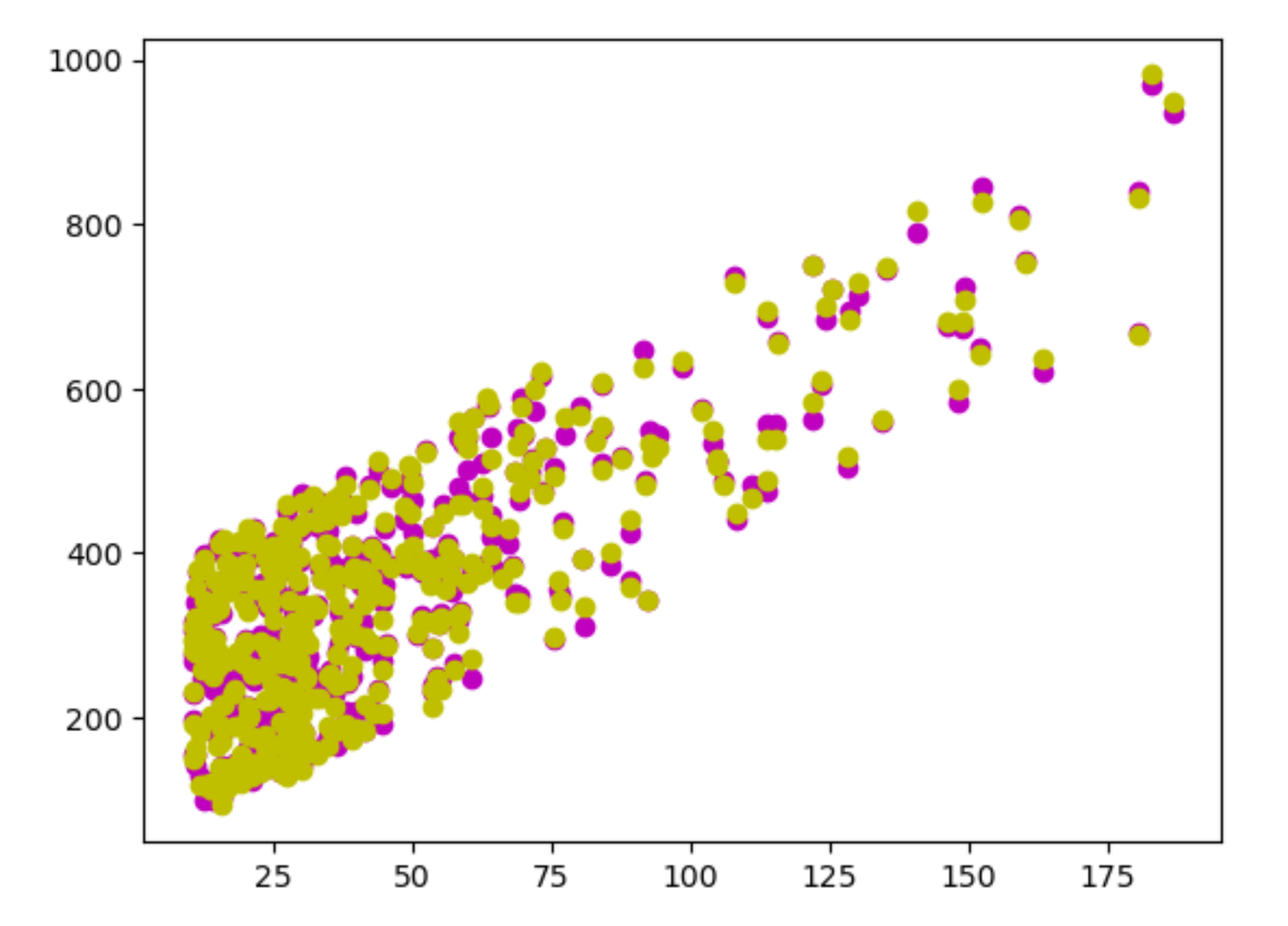
Out[95]:
```
array([ 1.32064098e-01,  6.25891484e-02,  3.25811876e-02, -3.44829447e-01,
        1.51557697e-02,  4.75140526e+01])
```

In [96]:
```
pred = np.dot(X, beta)
```

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

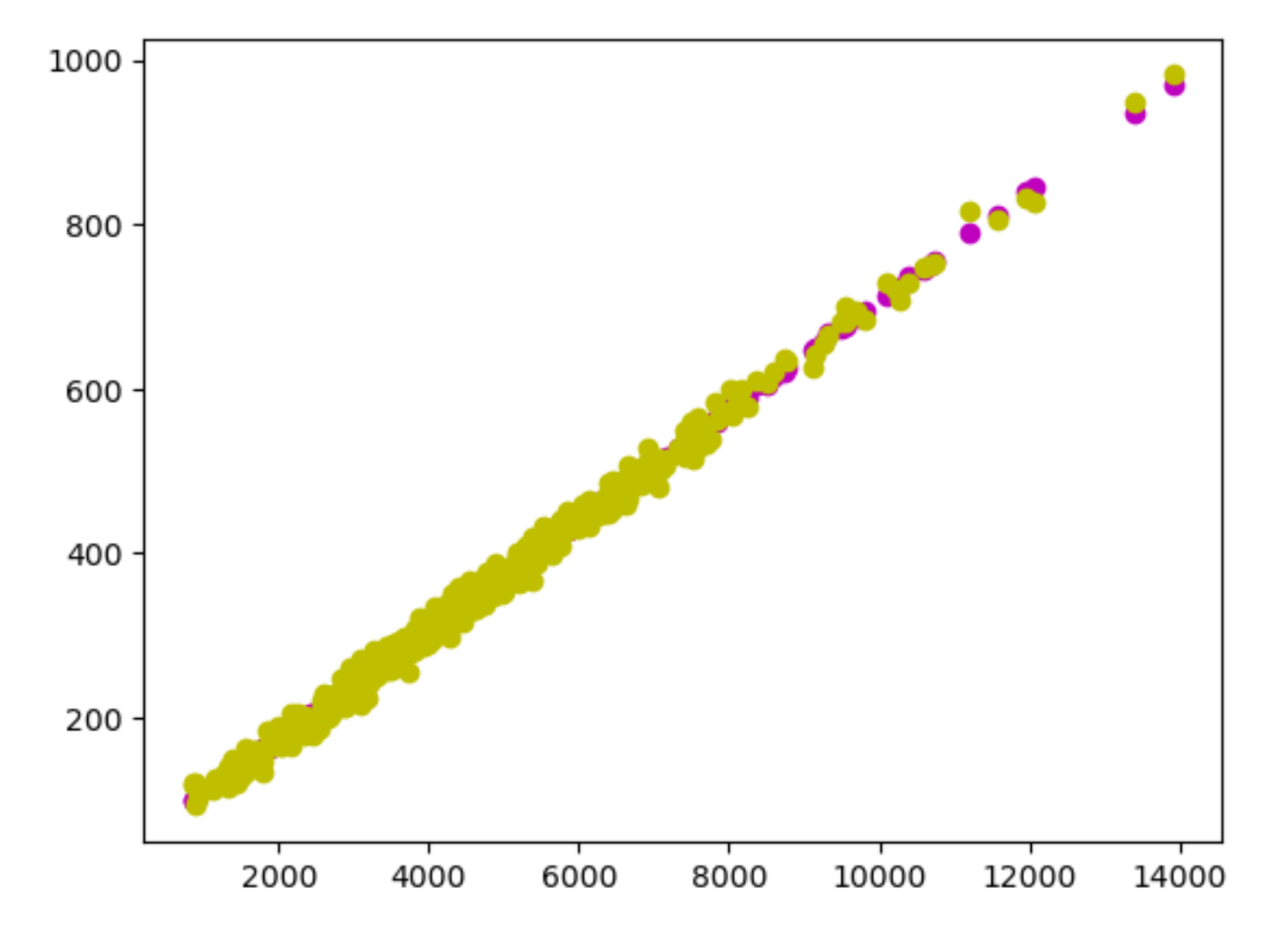
In [97]:
```
plt.scatter(X.T[0], pred, c='m')
plt.scatter(X.T[0], y, c ='y')
```

Out[97]: `<matplotlib.collections.PathCollection at 0x14bacdb70>`



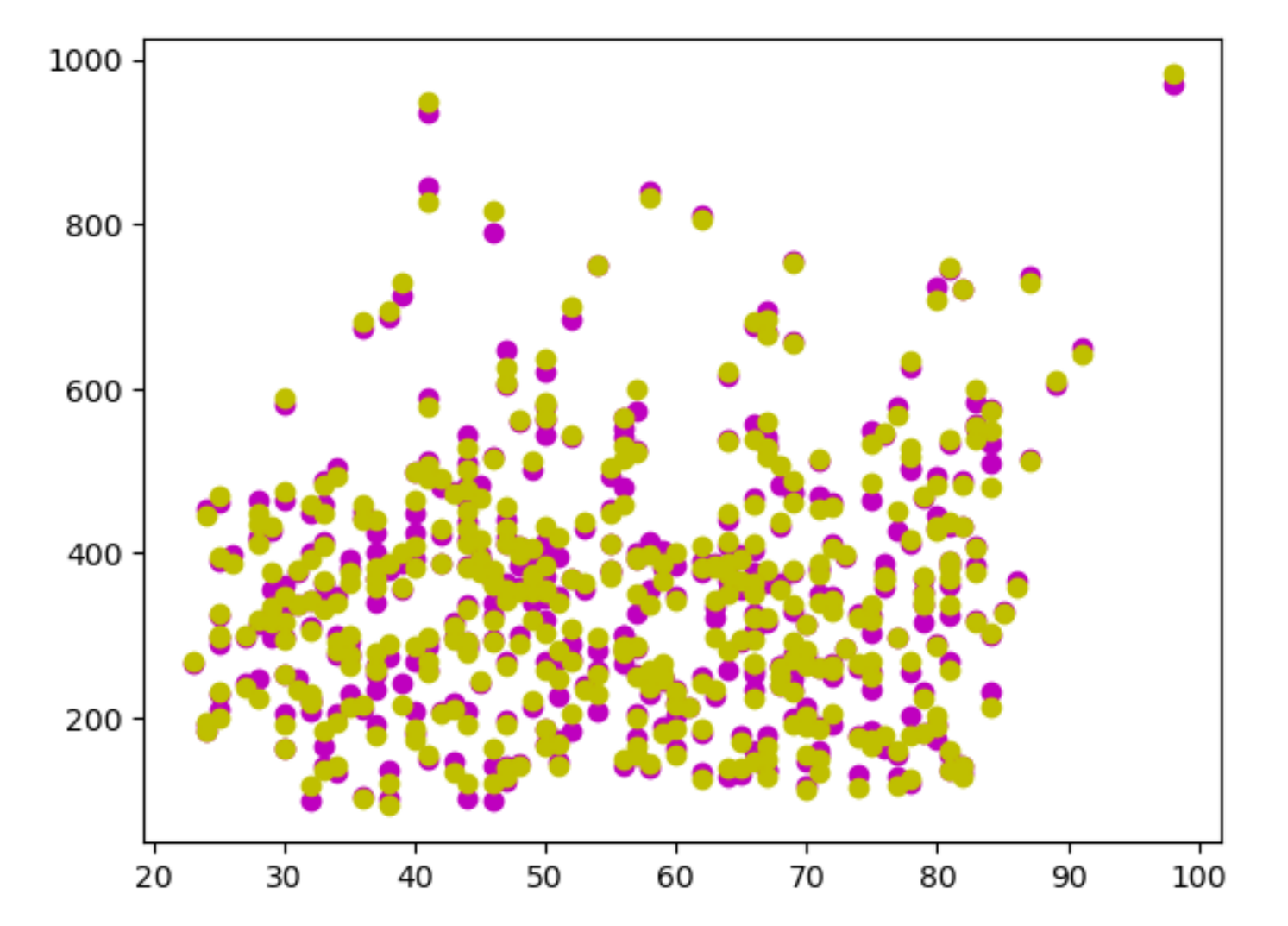
In [98]:
```
plt.scatter(X.T[1], pred, c='m')
plt.scatter(X.T[1], y, c ='y')
```

Out[98]: `<matplotlib.collections.PathCollection at 0x14b656da0>`



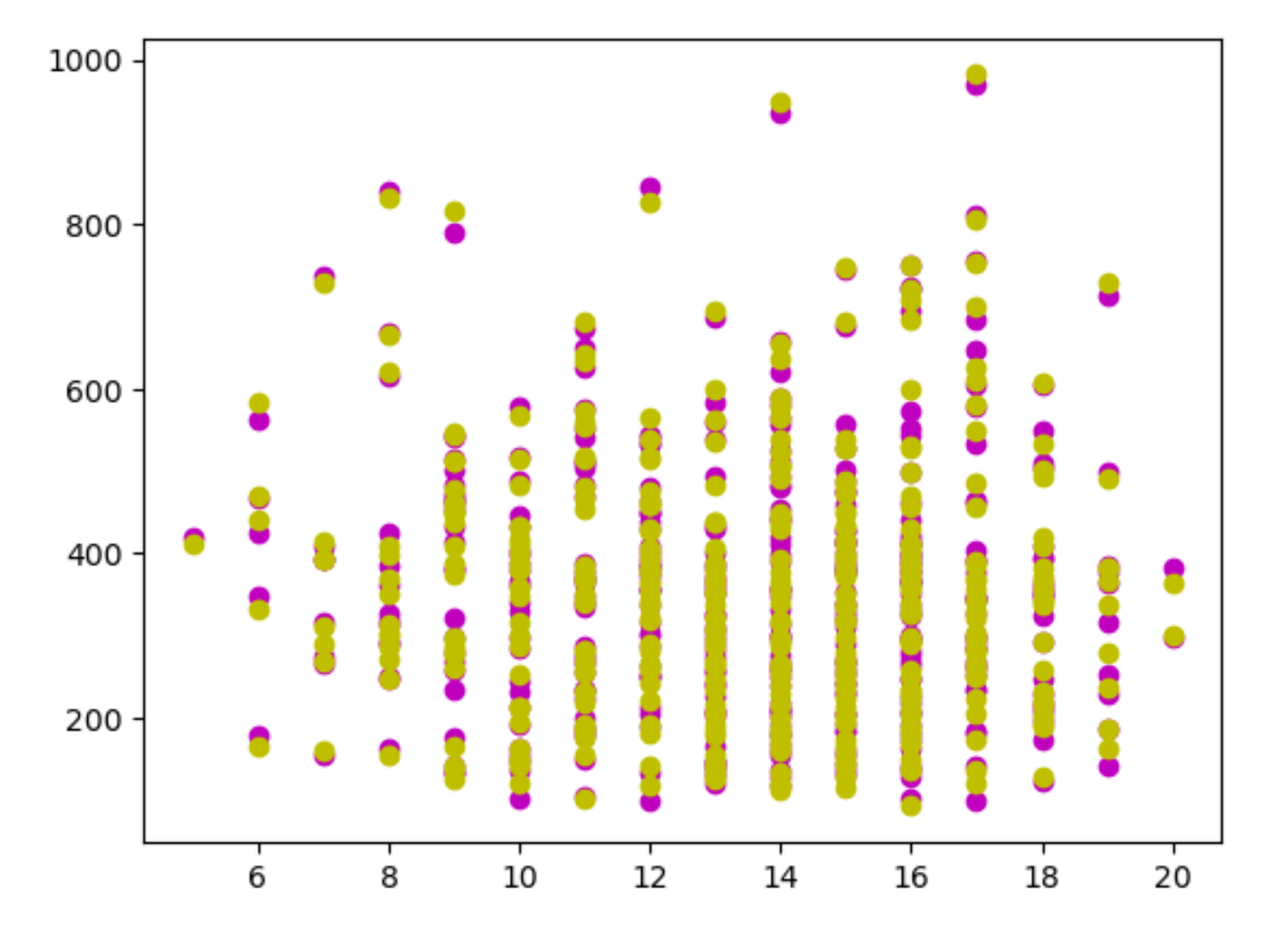
In [99]:
```
plt.scatter(X.T[2], pred, c='m')
plt.scatter(X.T[2], y, c ='y')
```

Out[99]: `<matplotlib.collections.PathCollection at 0x14bbee590>`



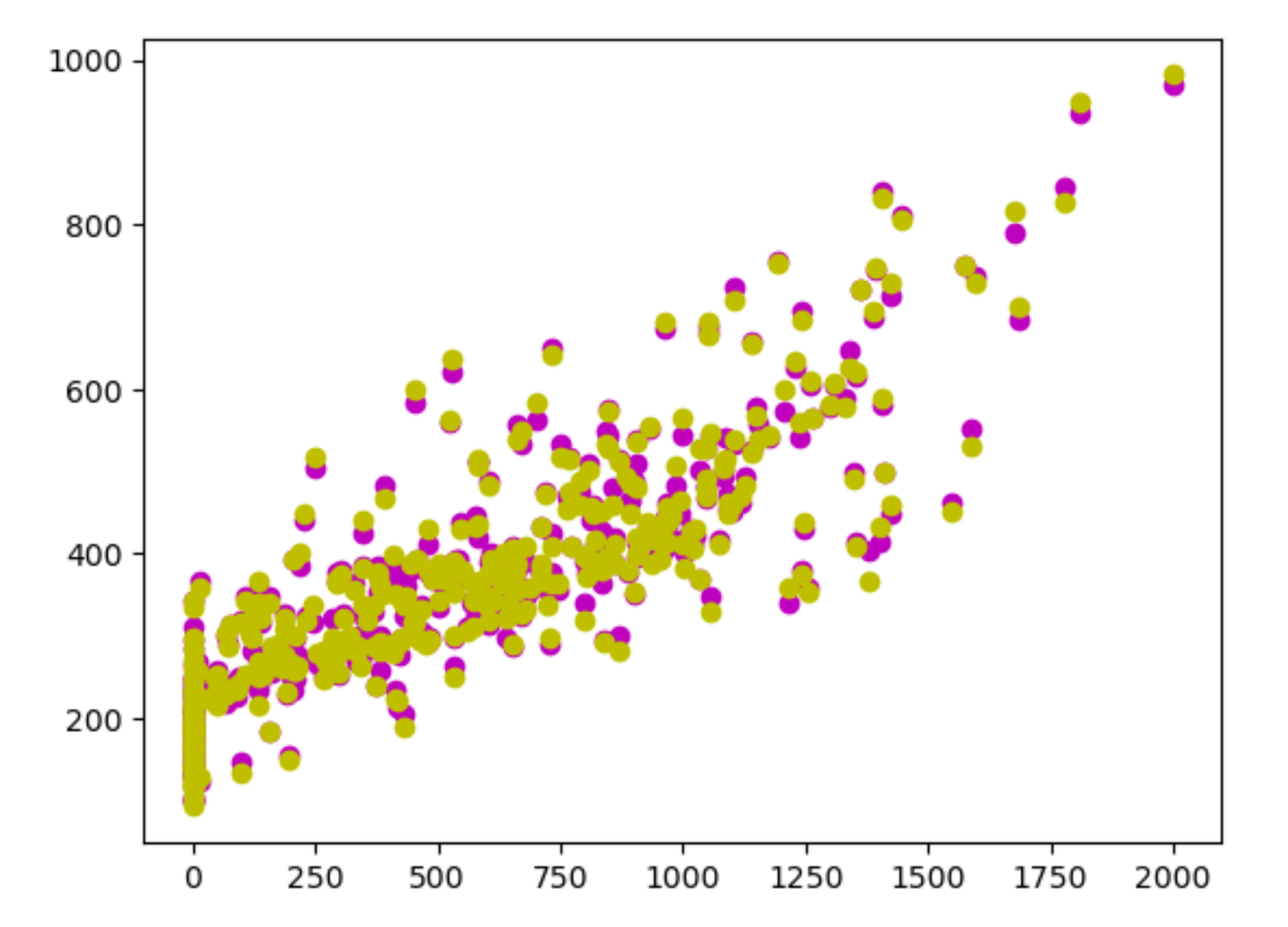
In [100…
```
plt.scatter(X.T[3], pred, c='m')
plt.scatter(X.T[3], y, c ='y')
```

Out[100]: `<matplotlib.collections.PathCollection at 0x14bb258a0>`



In [101…
```
plt.scatter(X.T[4], pred, c='m')
plt.scatter(X.T[4], y, c ='y')
```

Out[101]: `<matplotlib.collections.PathCollection at 0x14bbc4370>`



In [ ]: