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

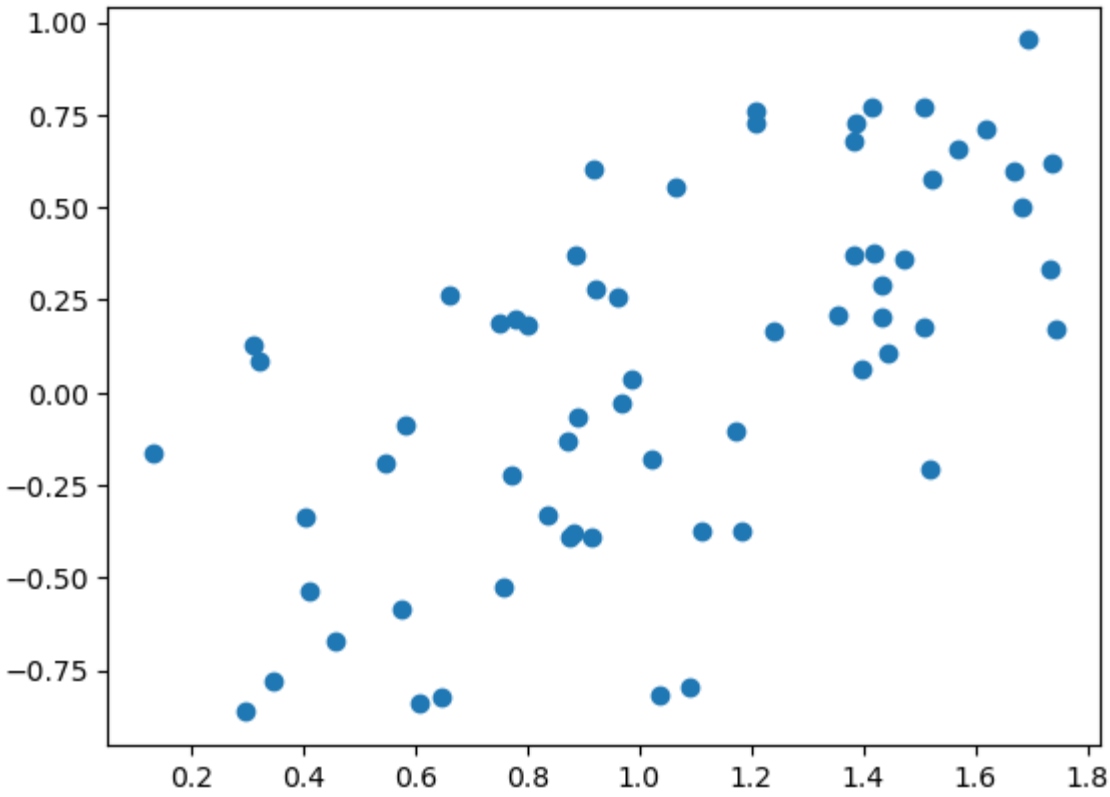
1. Create a 4 dimensional data set with 64 elements and show all 4 scatter 2D plots of the data x_1 vs. y, x_2 vs. y, x_3 vs. y, x_4 vs. y

In [2]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

In [3]:
```
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
```

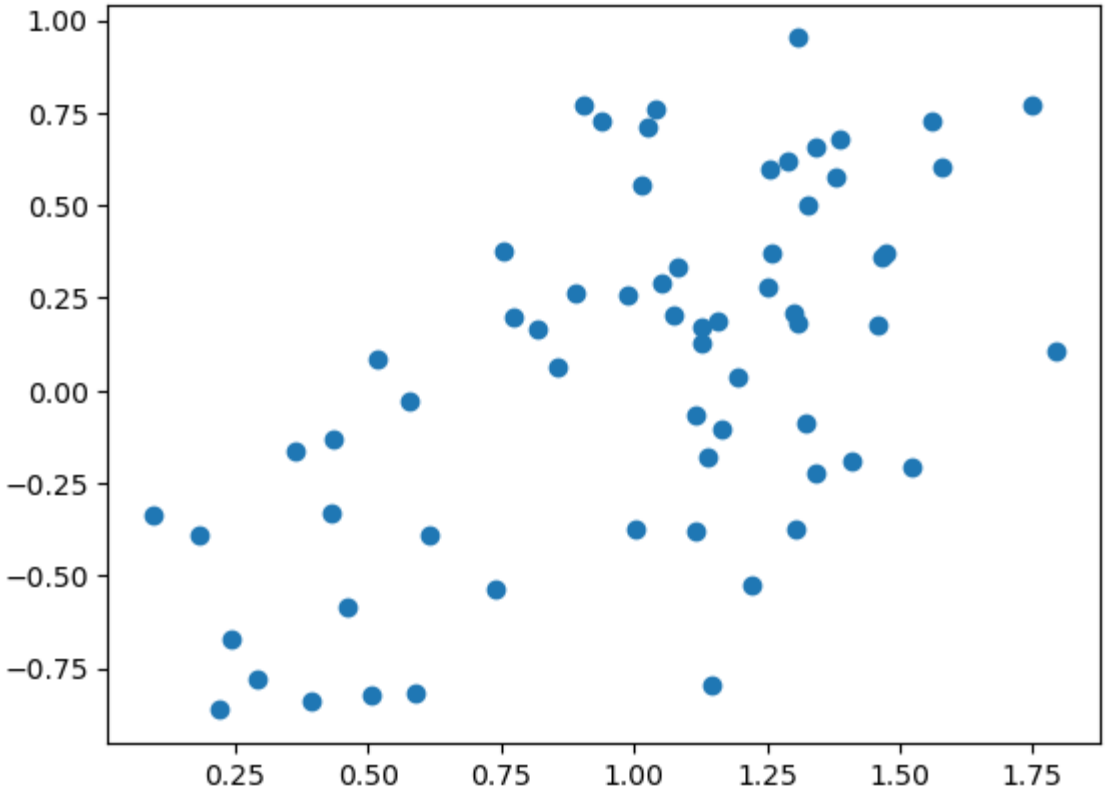
In [4]:
```
plt.scatter(x.T[0], y)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x1162fba60>
```



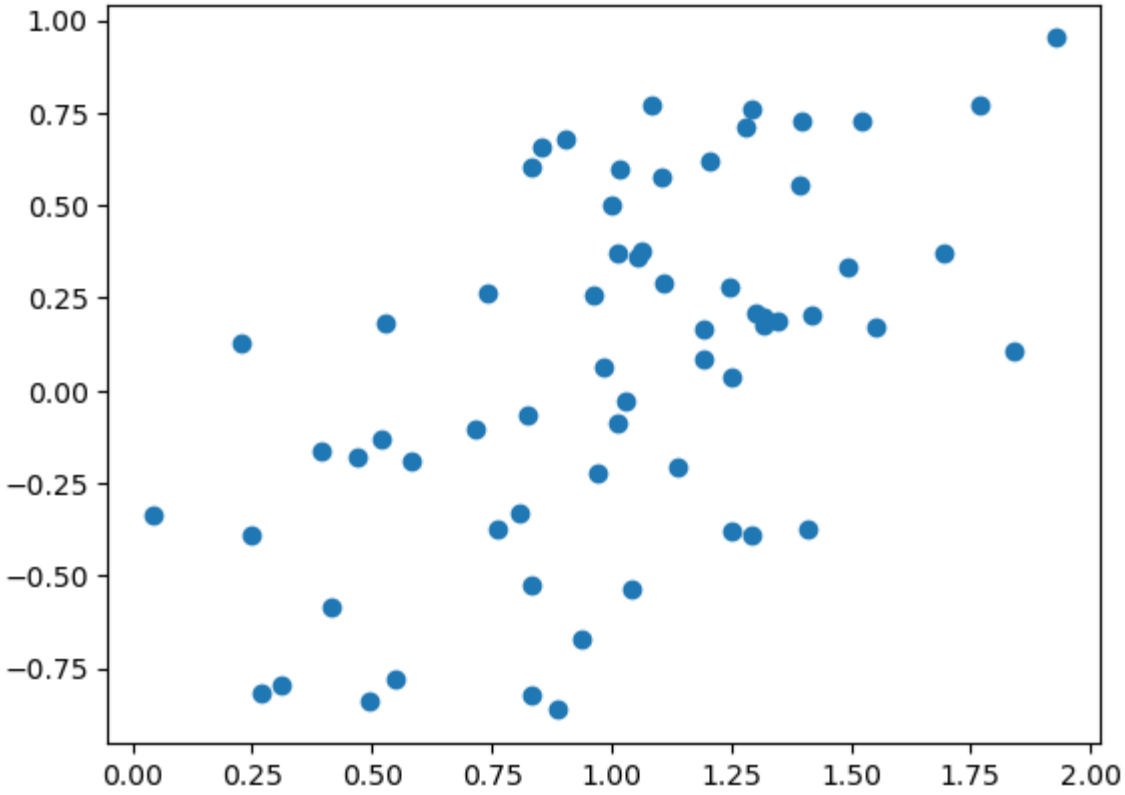
In [5]:
```
plt.scatter(x.T[1], y)
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x1165f63a0>
```



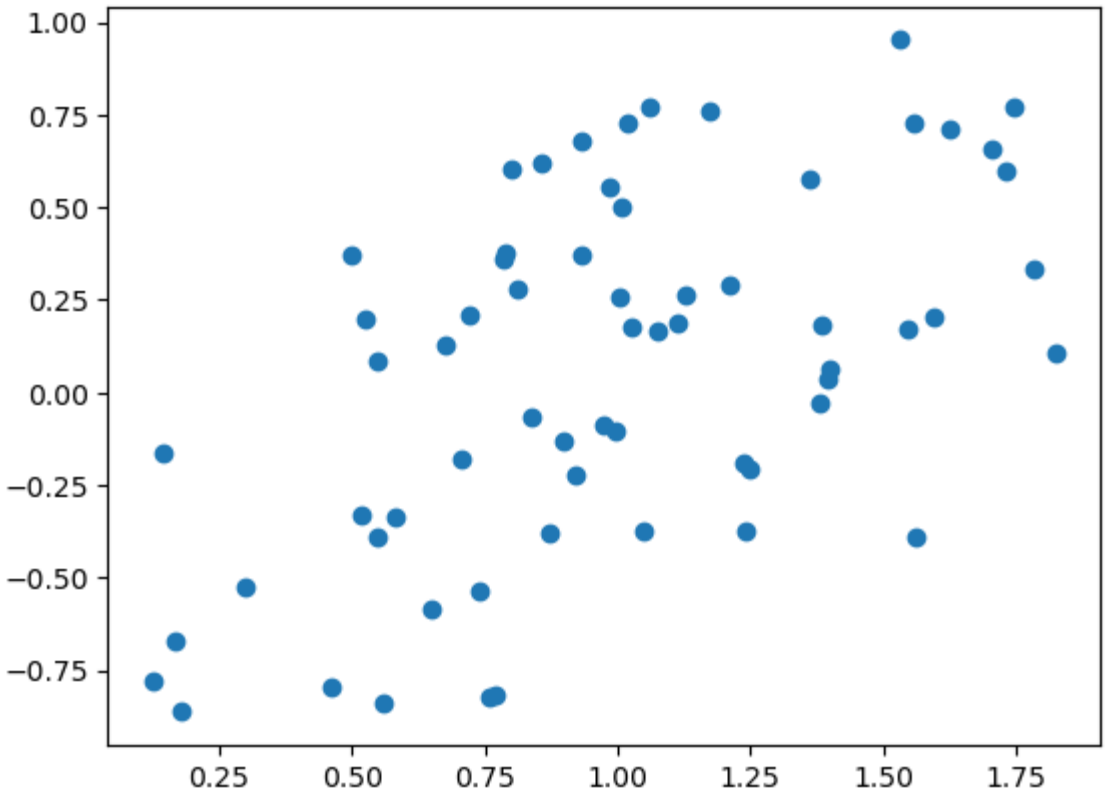
In [6]:
```
plt.scatter(x.T[2], y)
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x116743ee0>
```



In [7]:
```
plt.scatter(x.T[3], y)
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x11698af70>
```



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 [8]:
```
left = np.linalg.inv(np.dot(x.T, x))
```

In [9]:
```
right = np.dot(y.T, x)
```

In [10]:
```
bcol = np.dot(left, right)
bcol
```

Out[10]:
```
array([ 0.32354727,  0.27960904,  0.22513165,  0.1688763 , -0.9579772
6])
```

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 [11]:
```
x
```

Out[11]:
```
array([[0.40457958, 0.09485502, 0.04269266, 0.58067466, 1.        ],
       [0.64618448, 0.50712644, 0.83219516, 0.75865271, 1.        ],
       [0.60652889, 0.39426108, 0.49443371, 0.55790465, 1.        ],
       [0.1328181 , 0.36181526, 0.39575131, 0.14437078, 1.        ],
       [0.45699026, 0.24133252, 0.93663716, 0.16812544, 1.        ],
       [0.41042189, 0.74103688, 1.04005756, 0.73982277, 1.        ],
       [0.3487511 , 0.29140378, 0.54835621, 0.12480317, 1.        ],
       [0.29732332, 0.21894588, 0.88873717, 0.17903601, 1.        ],
       [0.87605723, 0.18337247, 0.24875711, 0.54767466, 1.        ],
       [1.03597276, 0.59058105, 0.27049764, 0.76985087, 1.        ],
       [1.02297126, 1.13933057, 0.4711608 , 0.70664631, 1.        ],
       [0.31204559, 1.1260271 , 0.22669622, 0.67672954, 1.        ],
       [1.08886172, 1.14798794, 0.31162929, 0.46261206, 1.        ],
       [0.83573415, 0.42876641, 0.80784833, 0.51750272, 1.        ],
       [1.17327554, 1.16501966, 0.71588828, 0.99627118, 1.        ],
       [0.7582796 , 1.22135142, 0.83433407, 0.29678714, 1.        ],
       [1.24142103, 0.81924066, 1.19367992, 1.07461399, 1.        ],
       [0.32120788, 0.51786319, 1.19088417, 0.54914335, 1.        ],
       [1.11016815, 1.00379698, 0.7623039 , 1.04759399, 1.        ],
       [0.57468168, 0.45924254, 0.41471291, 0.65103379, 1.        ],
       [0.66128029, 0.88970766, 0.74179242, 1.12822238, 1.        ],
       [0.778318  , 0.77240878, 1.31885784, 0.52310789, 1.        ],
       [0.96216433, 0.98707108, 0.96423313, 1.00533949, 1.        ],
       [0.87261473, 0.43587403, 0.51823888, 0.89879395, 1.        ],
       [0.88886241, 1.11726292, 0.82548904, 0.83970753, 1.        ],
       [0.80249786, 1.30997212, 0.5273949 , 1.38602896, 1.        ],
       [0.88535679, 1.25783133, 1.01084126, 0.49954384, 1.        ],
       [0.98507928, 1.19413471, 1.25248614, 1.3957631 , 1.        ],
       [0.97062169, 0.57765638, 1.02819094, 1.38271185, 1.        ],
       [0.92175431, 1.2502213 , 1.24684708, 0.8103658 , 1.        ],
       [1.41731468, 0.75296208, 1.0634101 , 0.78926522, 1.        ],
       [0.58141852, 1.32300417, 1.01457966, 0.97396437, 1.        ],
       [1.3963748 , 0.85803402, 0.98275023, 1.39958474, 1.        ],
       [1.18181839, 1.30470469, 1.40890794, 1.23983976, 1.        ],
       [0.5461129 , 1.40939359, 0.58042038, 1.23850853, 1.        ],
       [0.75059751, 1.15893526, 1.34824376, 1.1126536 , 1.        ],
       [1.51818779, 1.52445545, 1.13649259, 1.24881452, 1.        ],
       [0.88281415, 1.11782235, 1.248825  , 0.87121076, 1.        ],
       [0.9136138 , 0.61495854, 1.29473464, 1.56255502, 1.        ],
       [1.47372174, 1.46785088, 1.05527811, 0.78512243, 1.        ],
       [0.91999165, 1.57939341, 0.83309155, 0.80107733, 1.        ],
       [0.77333816, 1.34393942, 0.97049521, 0.92176165, 1.        ],
       [1.52086829, 1.38012188, 1.10543044, 1.36018275, 1.        ],
       [1.35554155, 1.3003723 , 1.30222263, 0.72199837, 1.        ],
       [1.38400172, 1.47555576, 1.6934468 , 0.93102148, 1.        ],
       [1.68132942, 1.32731117, 0.99939076, 1.00584814, 1.        ],
       [1.06426249, 1.0156997 , 1.39134056, 0.98441713, 1.        ],
       [1.62016715, 1.02702672, 1.28024135, 1.62758208, 1.        ],
       [1.73632762, 1.28829536, 1.20294283, 0.85510458, 1.        ],
       [1.20687697, 1.04114833, 1.29141226, 1.17461718, 1.        ],
       [1.5066274 , 1.45946162, 1.31640836, 1.02733345, 1.        ],
       [1.38444125, 1.3879552 , 0.90220708, 0.93185292, 1.        ],
       [1.57055374, 1.34382126, 0.85554797, 1.70481506, 1.        ],
       [1.74307394, 1.12601521, 1.55239153, 1.54531437, 1.        ],
```

```
       [1.50677931, 1.74991163, 1.76986247, 1.0607048 , 1.        ],
       [1.20946533, 0.93751918, 1.52345151, 1.55702588, 1.        ],
       [1.41542775, 0.90612823, 1.08166384, 1.74847667, 1.        ],
       [1.43274662, 1.05026471, 1.10926591, 1.21242336, 1.        ],
       [1.44396814, 1.79352576, 1.8410465 , 1.82504466, 1.        ],
       [1.66897577, 1.25506323, 1.01824754, 1.73087194, 1.        ],
       [1.43202681, 1.07492979, 1.41701559, 1.59554571, 1.        ],
       [1.73220578, 1.08265811, 1.49164627, 1.78592179, 1.        ],
       [1.6940332 , 1.30669089, 1.92709007, 1.53347738, 1.        ],
       [1.38793896, 1.56215008, 1.3974669 , 1.01743437, 1.        ]])
```

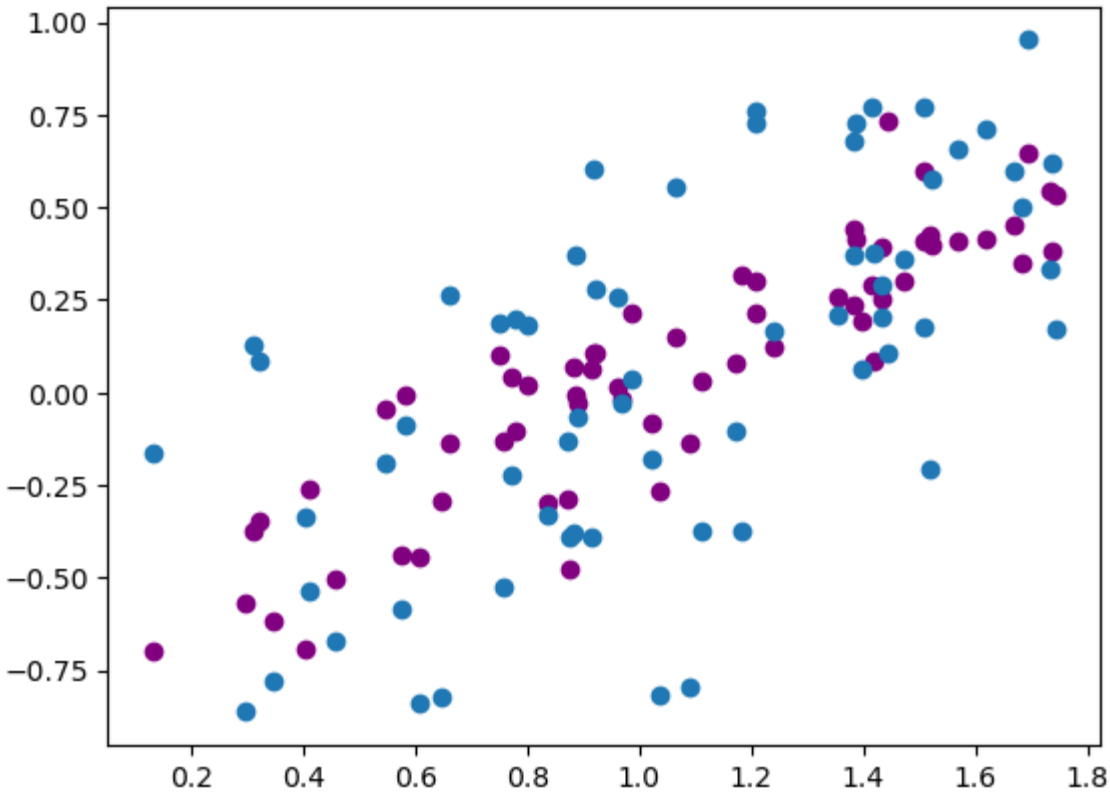
In [12]:
```
y
```

Out[12]:
```
array([-0.33915632, -0.82329793, -0.8386654 , -0.16235147, -0.6704670
2,
       -0.53844484, -0.78072967, -0.86288732, -0.39151363, -0.8194373
1,
       -0.17917742,  0.12760649, -0.79418101, -0.32919956, -0.1067549
8,
       -0.52483334,  0.16736382,  0.08364624, -0.3735886 , -0.5875215
8,
        0.26054815,  0.20055263,  0.25841948, -0.12864003, -0.0653233
9,
        0.18377867,  0.37184046,  0.0376616 , -0.0292554 ,  0.2791038
2,
        0.37815442, -0.08990823,  0.06362775, -0.37327596, -0.1905750
8,
        0.1873941 , -0.20847139, -0.38040984, -0.38812389,  0.3586191
6,
        0.60146269, -0.22064048,  0.57717606,  0.20893718,  0.3710488
8,
        0.50081927,  0.55613638,  0.71090797,  0.61998763,  0.7615520
5,
        0.17431803,  0.67628332,  0.65478984,  0.1700135 ,  0.7728810
4,
        0.72723959,  0.77012831,  0.29168548,  0.10655428,  0.5978169
1,
        0.20474252,  0.33406866,  0.95162338,  0.72697225])
```

In [13]:
```
bcolpred = np.dot(x, bcol)
```

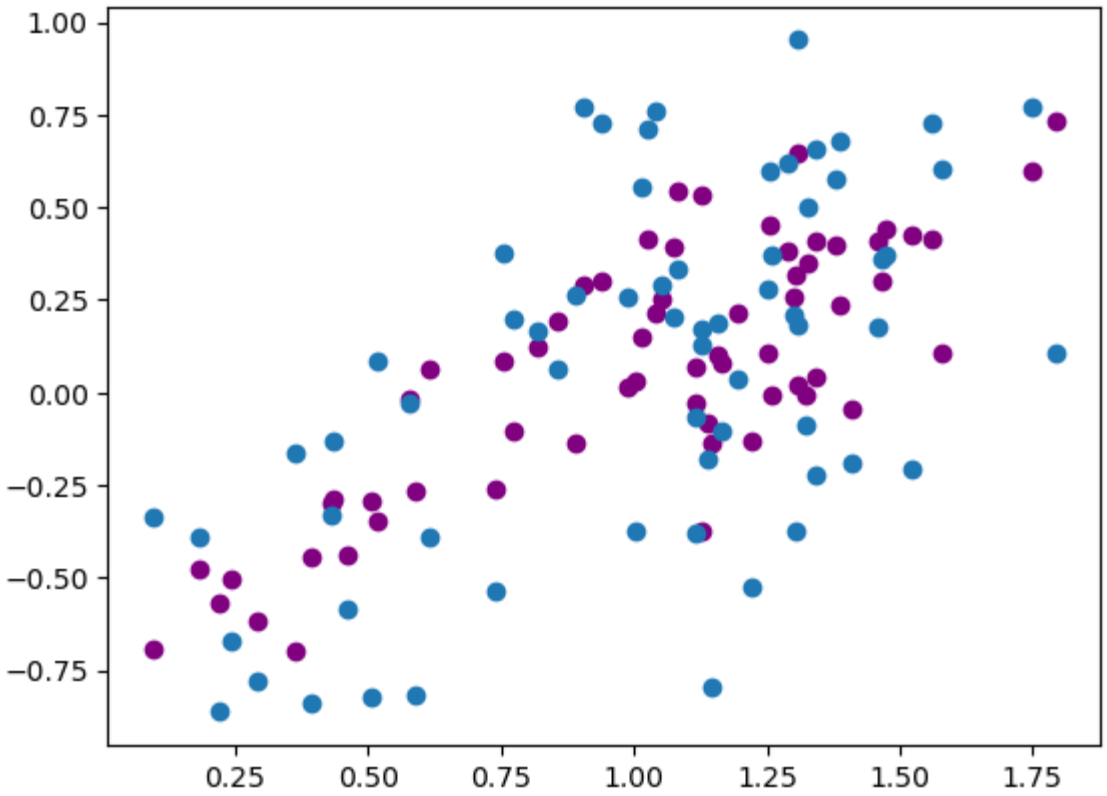
In [14]:
```
plt.scatter(x.T[0], bcolpred, c = 'purple')
plt.scatter(x.T[0], y)
```

Out[14]: <matplotlib.collections.PathCollection at 0x116e1df10>



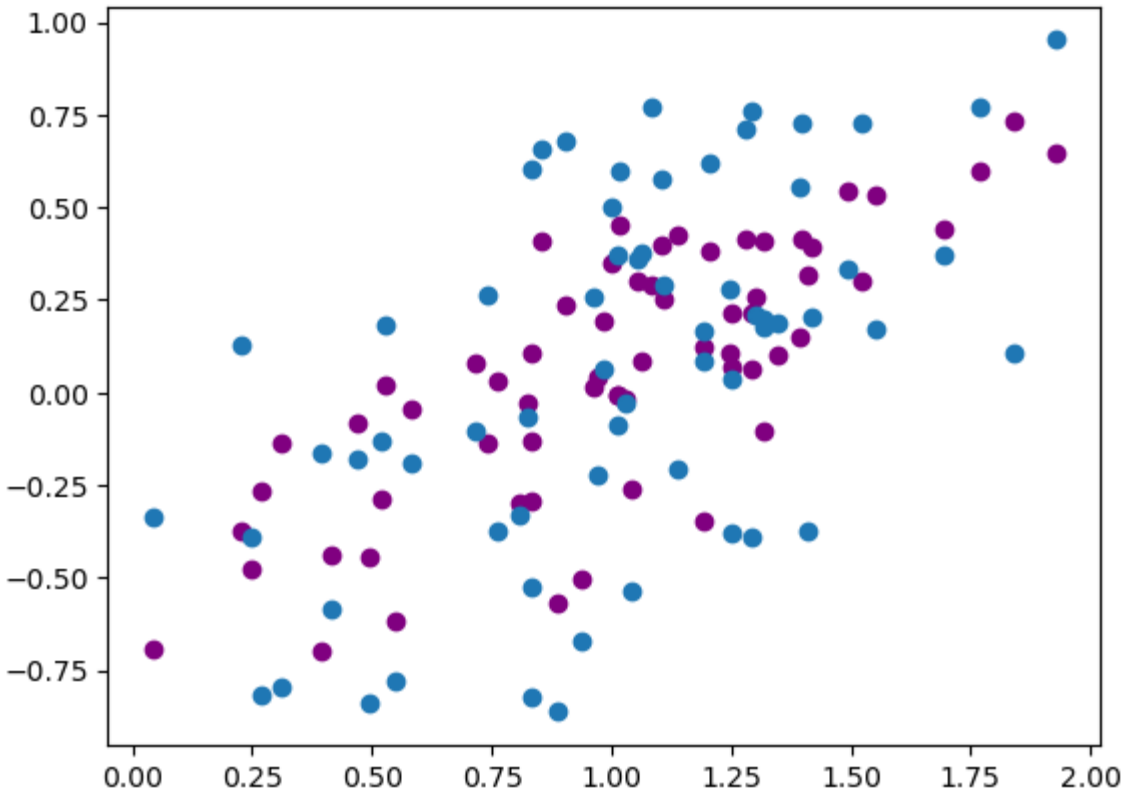
In [15]:
```
plt.scatter(x.T[1], bcolpred, c = 'purple')
plt.scatter(x.T[1], y)
```

Out[15]: <matplotlib.collections.PathCollection at 0x116f86700>



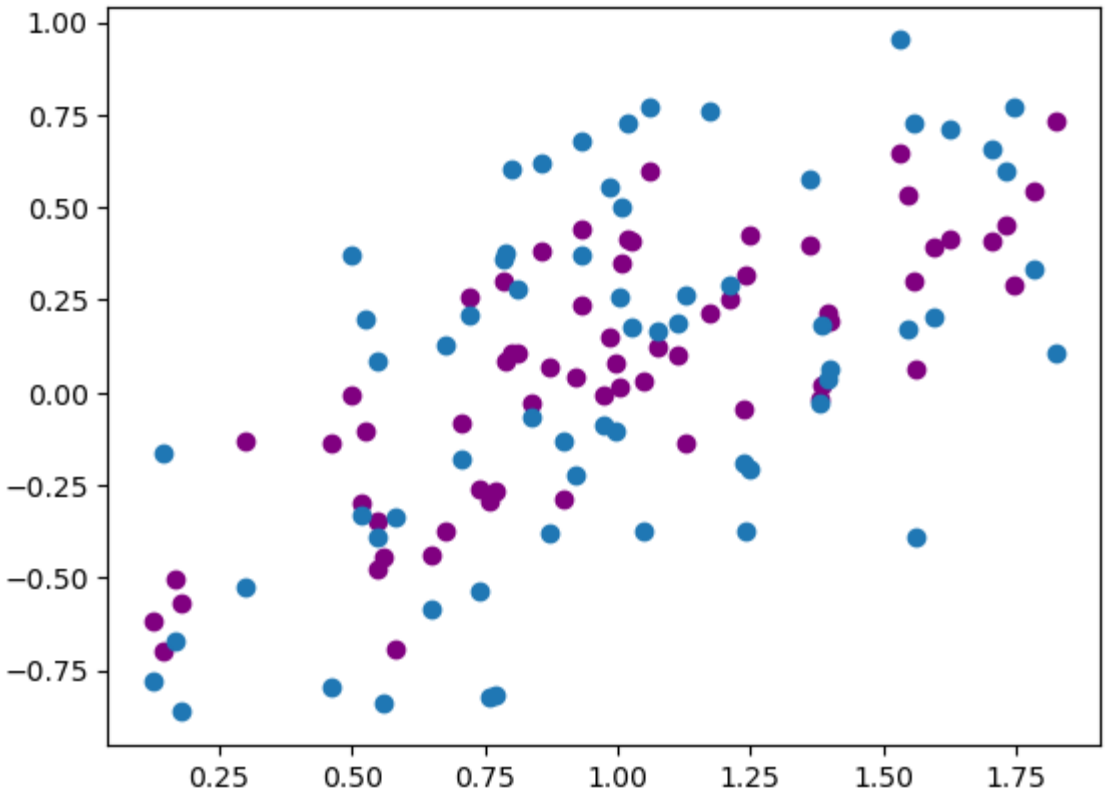
In [16]:
```
plt.scatter(x.T[2], bcolpred, c = 'purple')
plt.scatter(x.T[2], y)
```

Out[16]: <matplotlib.collections.PathCollection at 0x116feffa0>



In [17]:
```
plt.scatter(x.T[3], bcolpred, c = 'purple')
plt.scatter(x.T[3], y)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x11732b5e0>
```



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 [18]:
```
import pandas as pd
import numpy as np
credit = pd.read_csv('Credit.csv')
credit.head()
```

Out[18]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethn
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Cauca
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	Cauca

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

In [19]:
```
columns = ['Income', 'Limit', 'Cards', 'Age', 'Education']
X = credit[columns].values

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

Out[19]:
```
array([[1.48910e+01, 3.60600e+03, 2.00000e+00, 3.40000e+01, 1.10000e+
01,
        1.00000e+00],
       [1.06025e+02, 6.64500e+03, 3.00000e+00, 8.20000e+01, 1.50000e+
01,
        1.00000e+00],
       [1.04593e+02, 7.07500e+03, 4.00000e+00, 7.10000e+01, 1.10000e+
01,
        1.00000e+00],
       ...,
       [5.78720e+01, 4.17100e+03, 5.00000e+00, 6.70000e+01, 1.20000e+
01,
        1.00000e+00],
       [3.77280e+01, 2.52500e+03, 1.00000e+00, 4.40000e+01, 1.30000e+
01,
        1.00000e+00],
       [1.87010e+01, 5.52400e+03, 5.00000e+00, 6.40000e+01, 7.00000e+
00,
        1.00000e+00]])
```

In [20]:
```
y = credit['Rating']
y
```

Out[20]:
```
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 [21]:
```
left = np.linalg.inv(np.dot(X.T, X))
```

In [22]:
```
right = np.dot(y.T, X)
```

In [23]:
```
creditB = np.dot(left, right)
creditB
```

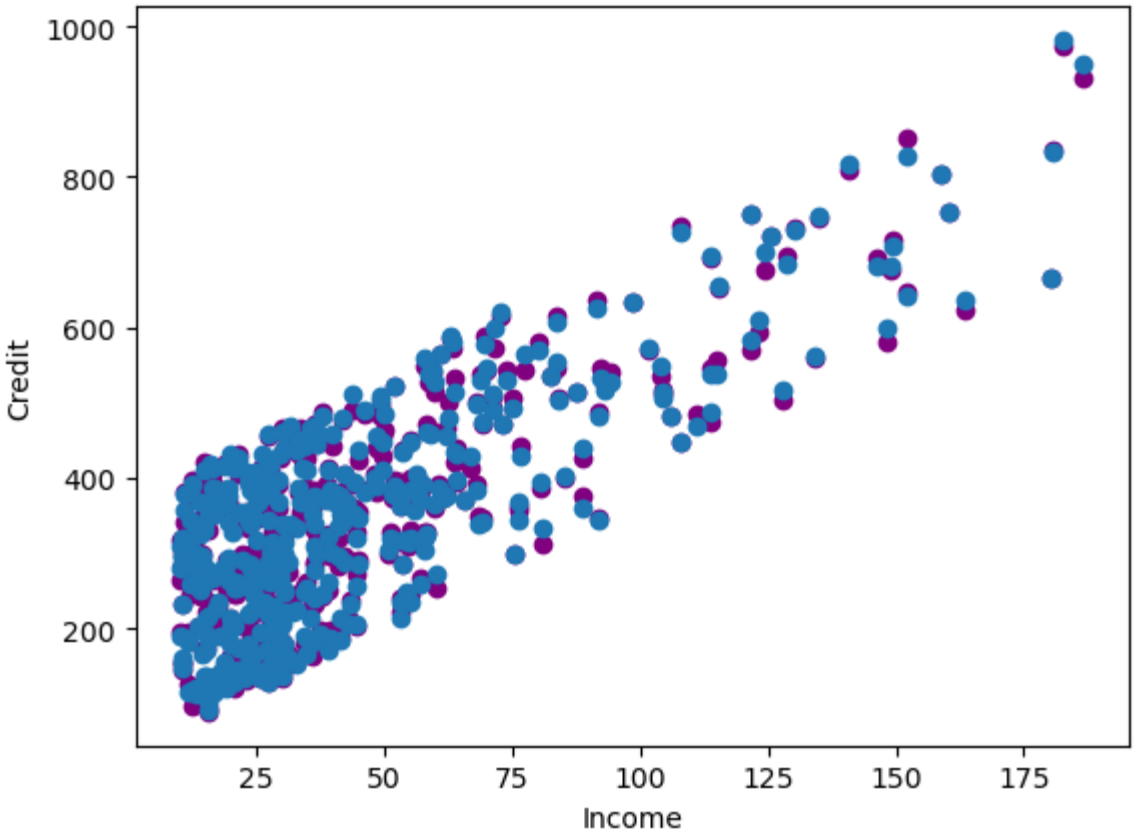
Out[23]:
```
array([ 3.35485393e-02,  6.63804553e-02,  4.85381863e+00,  7.78467717
e-04,
       -2.18197621e-01,  2.76079424e+01])
```

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

In [24]:
```
creditpred = np.dot(X, creditB)
```

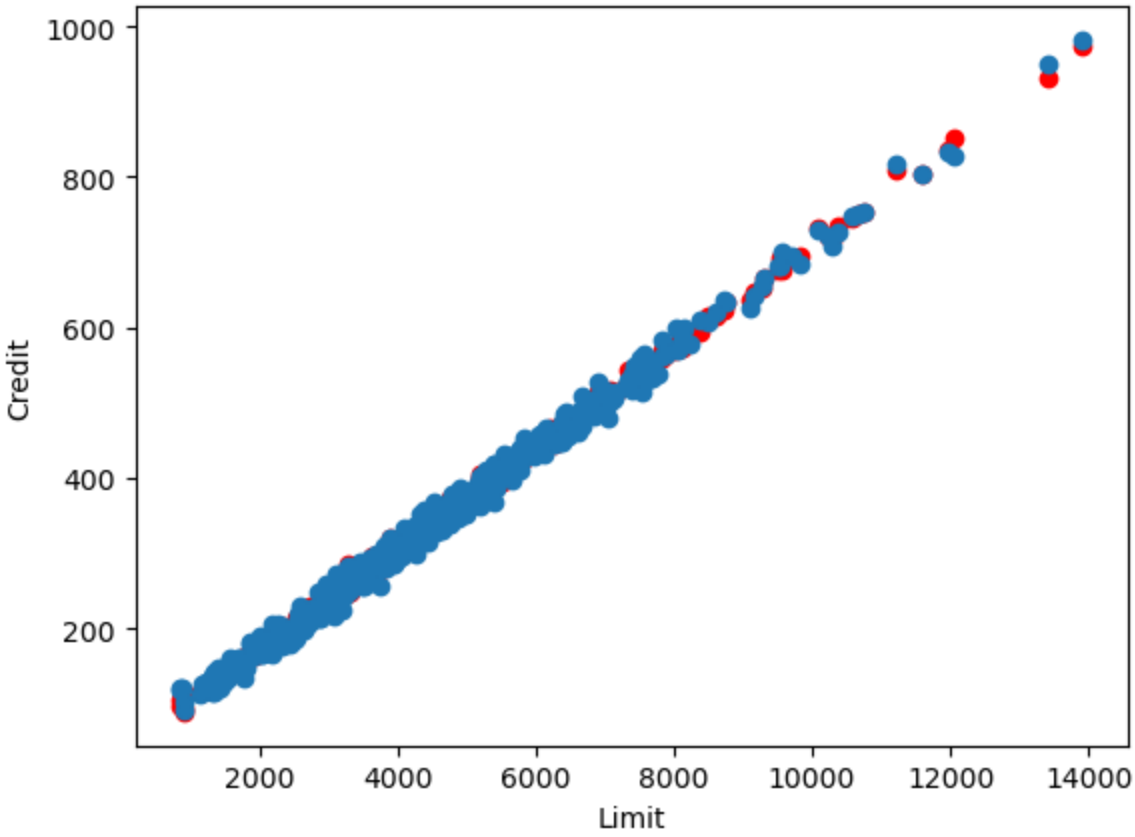
In [31]:
```
plt.scatter(X.T[0], creditpred, c = 'purple')
plt.scatter(X.T[0], y)
plt.ylabel('Credit')
plt.xlabel('Income')
```

Out[31]:
```
Text(0.5, 0, 'Income')
```



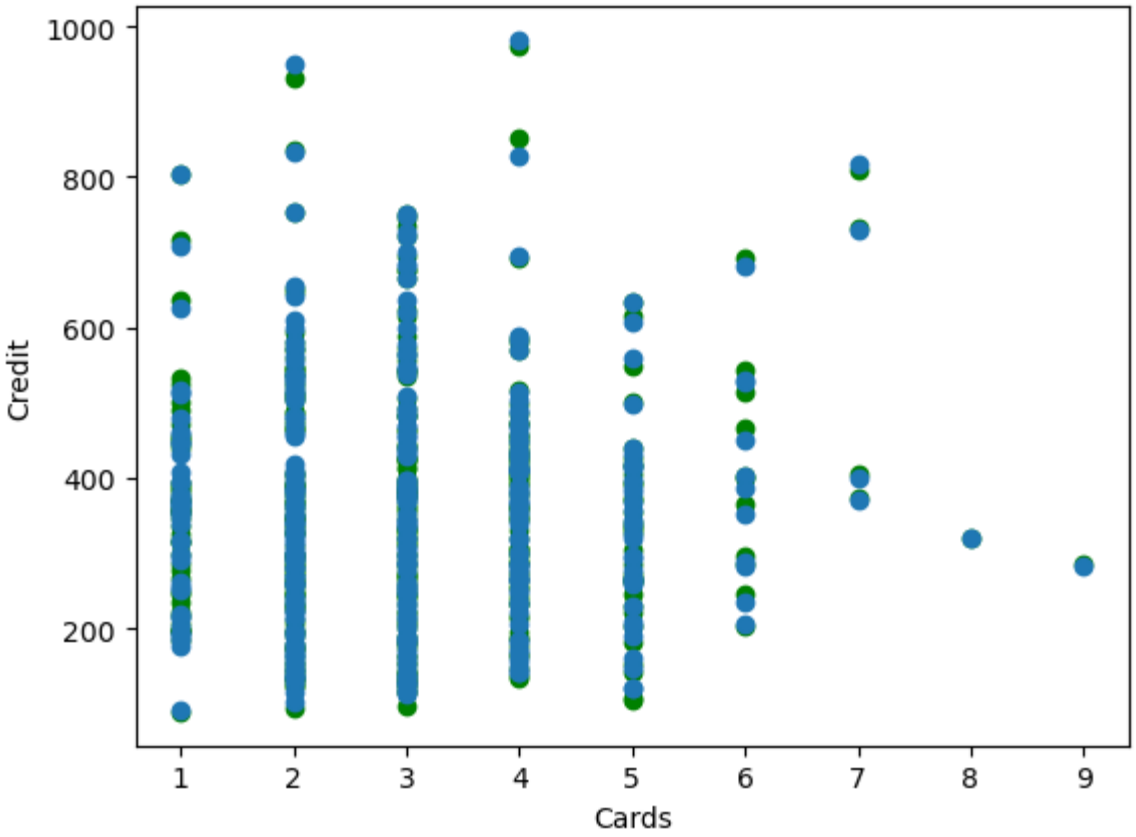
In [32]:
```
plt.scatter(X.T[1], creditpred, c = 'red')
plt.scatter(X.T[1], y)
plt.ylabel('Credit')
plt.xlabel('Limit')
```

Out[32]:
```
Text(0.5, 0, 'Limit')
```



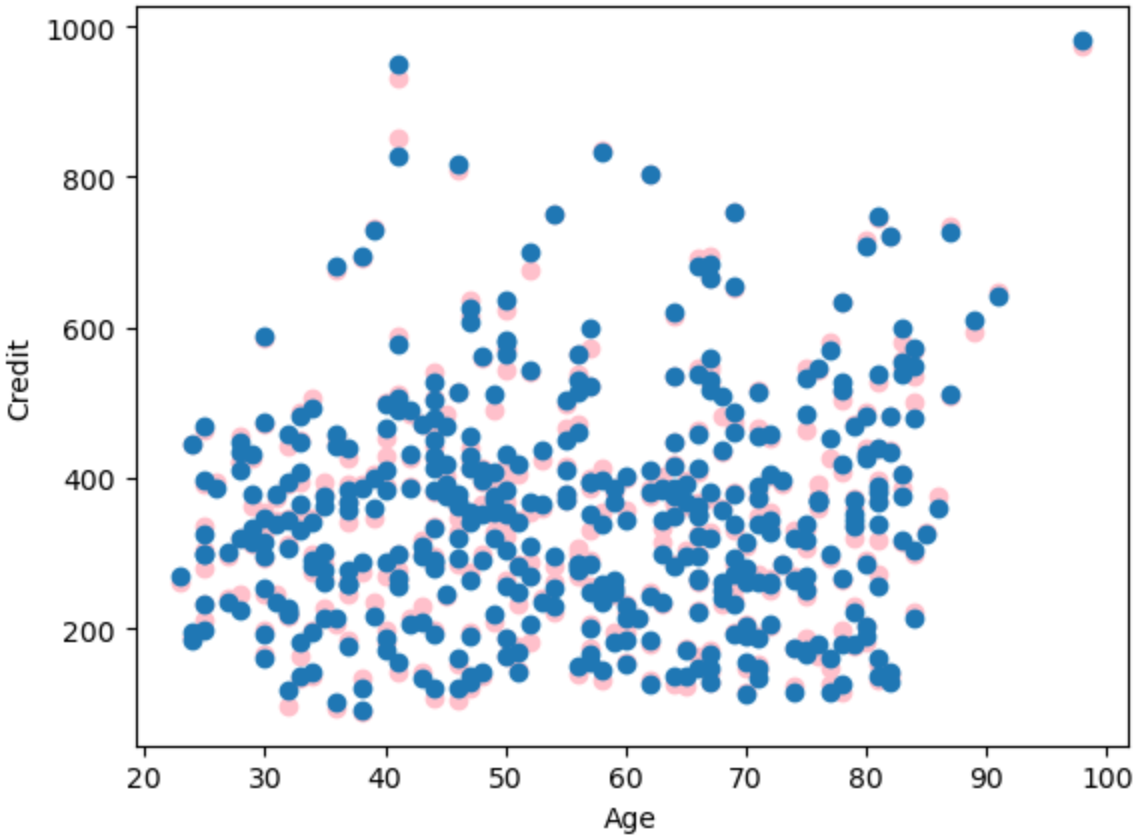
In [33]:
```
plt.scatter(X.T[2], creditpred, c = 'green')
plt.scatter(X.T[2], y)
plt.ylabel('Credit')
plt.xlabel('Cards')
```

Out[33]:
```
Text(0.5, 0, 'Cards')
```



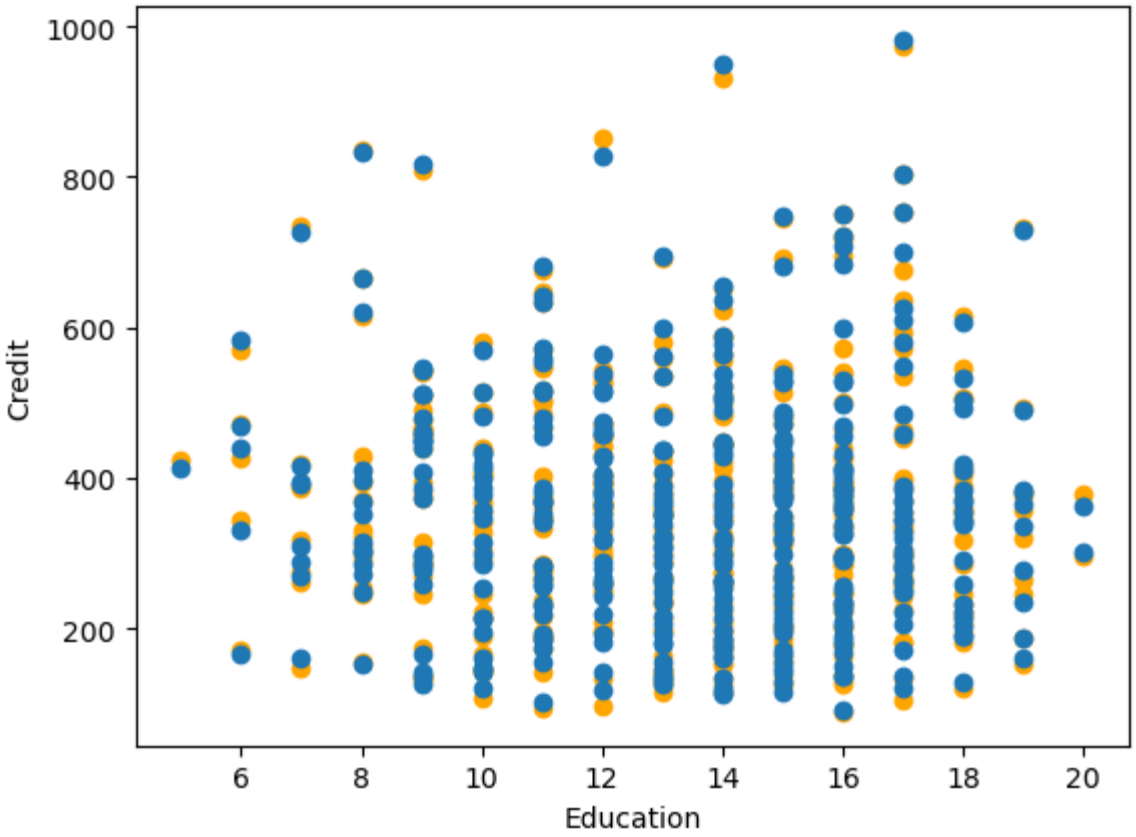
In [34]:
```
plt.scatter(X.T[3], creditpred, c = 'pink')
plt.scatter(X.T[3], y)
plt.ylabel('Credit')
plt.xlabel('Age')
```

Out[34]:
```
Text(0.5, 0, 'Age')
```



In [35]:
```
plt.scatter(X.T[4], creditpred, c = 'orange')
plt.scatter(X.T[4], y)
plt.ylabel('Credit')
plt.xlabel('Education')
```

Out[35]:
```
Text(0.5, 0, 'Education')
```



In [ ]:
```
```