Assigment 3 - Coder

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 [4]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

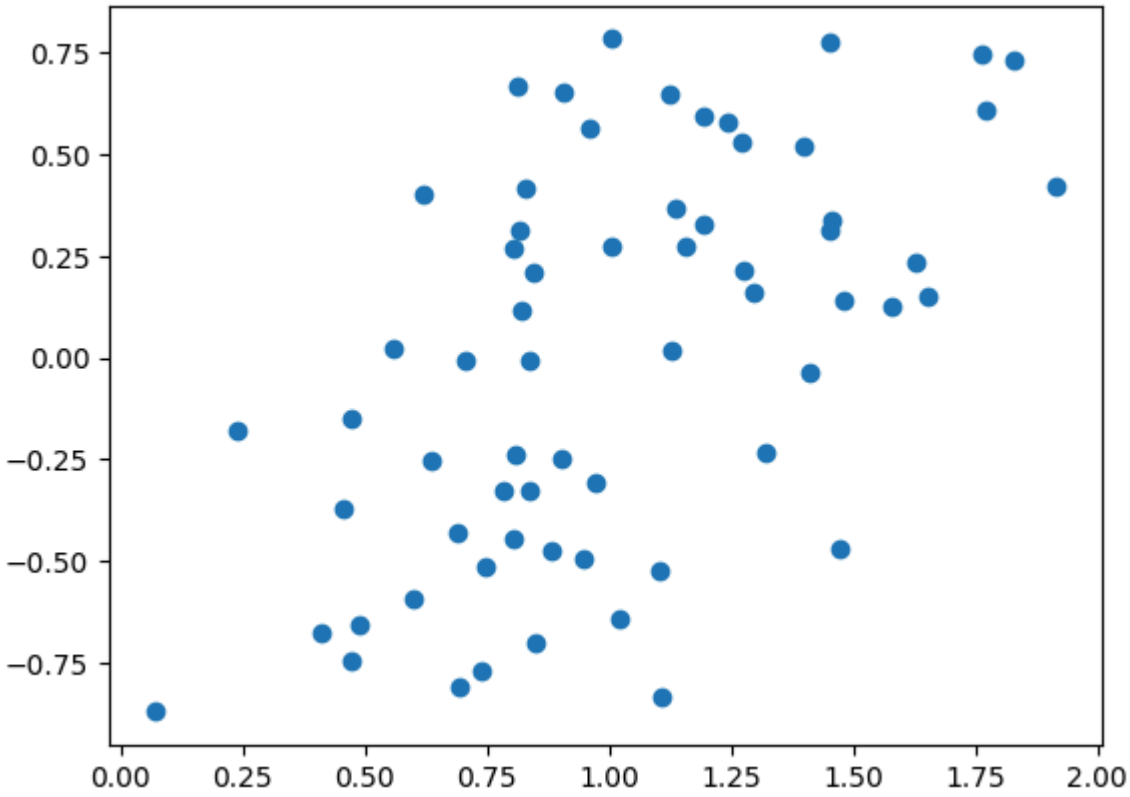
1. Create a 4 dimensional data set with 64 elements and show all 4 scatter 2D plots of the data x_1 vs. y, x_2 vs. y, x_3 vs. y, x_4 vs. y

In [5]:
```
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
```

x_1 vs. y

In [6]:
```
plt.scatter(x.T[0],y)
```

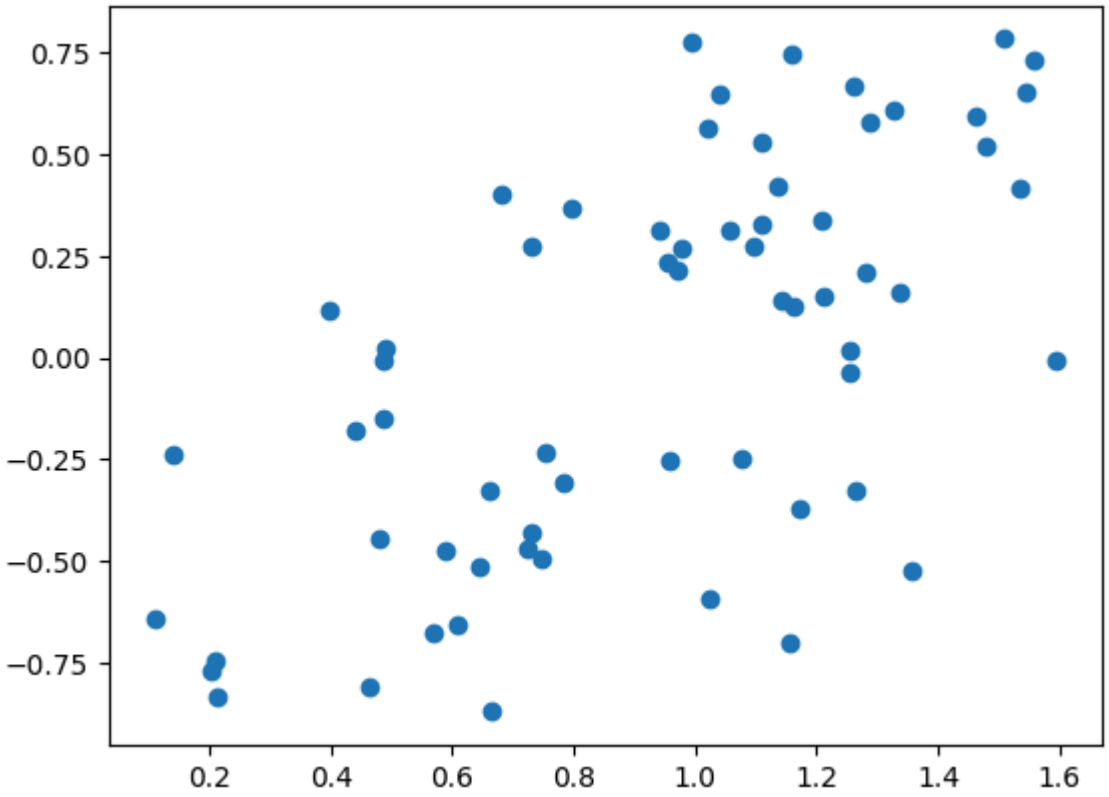
Out[6]:
```
<matplotlib.collections.PathCollection at 0x17d347f9a50>
```



x_2 vs. y

In [7]:
```
plt.scatter(x.T[1],y)
```

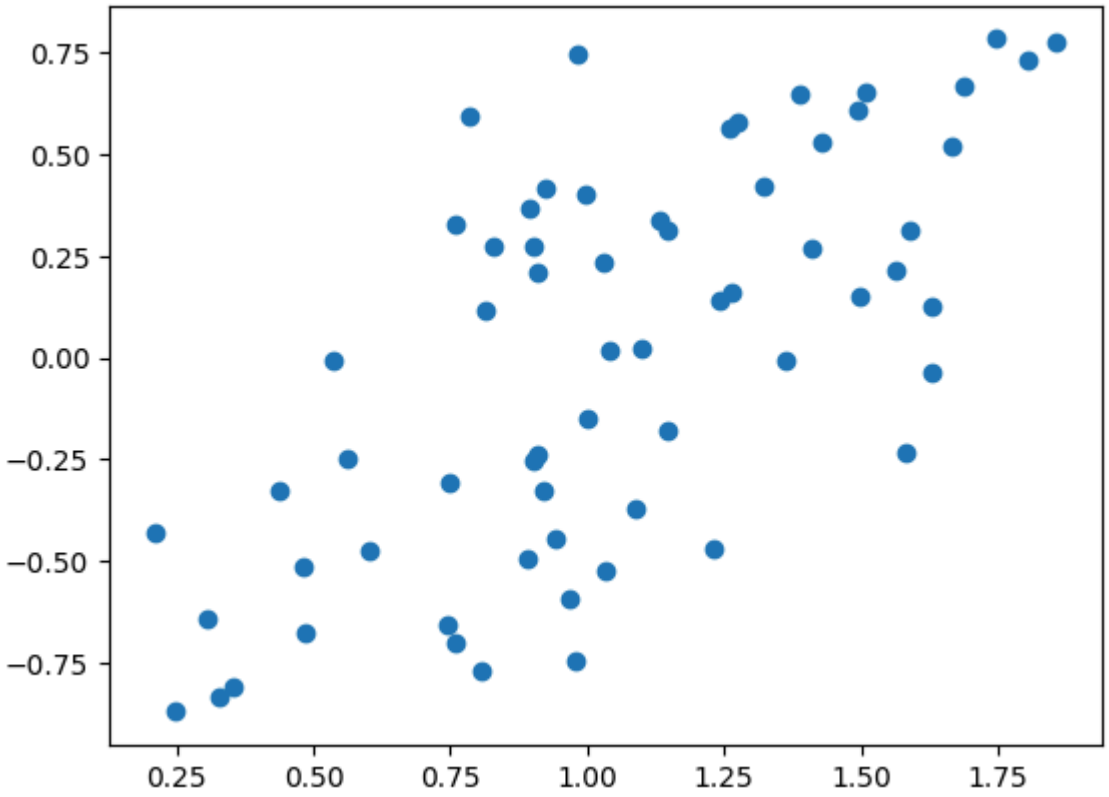
Out[7]:
```
<matplotlib.collections.PathCollection at 0x17d348d2b10>
```



x_3 vs. y

In [8]:
```
plt.scatter(x.T[2],y)
```

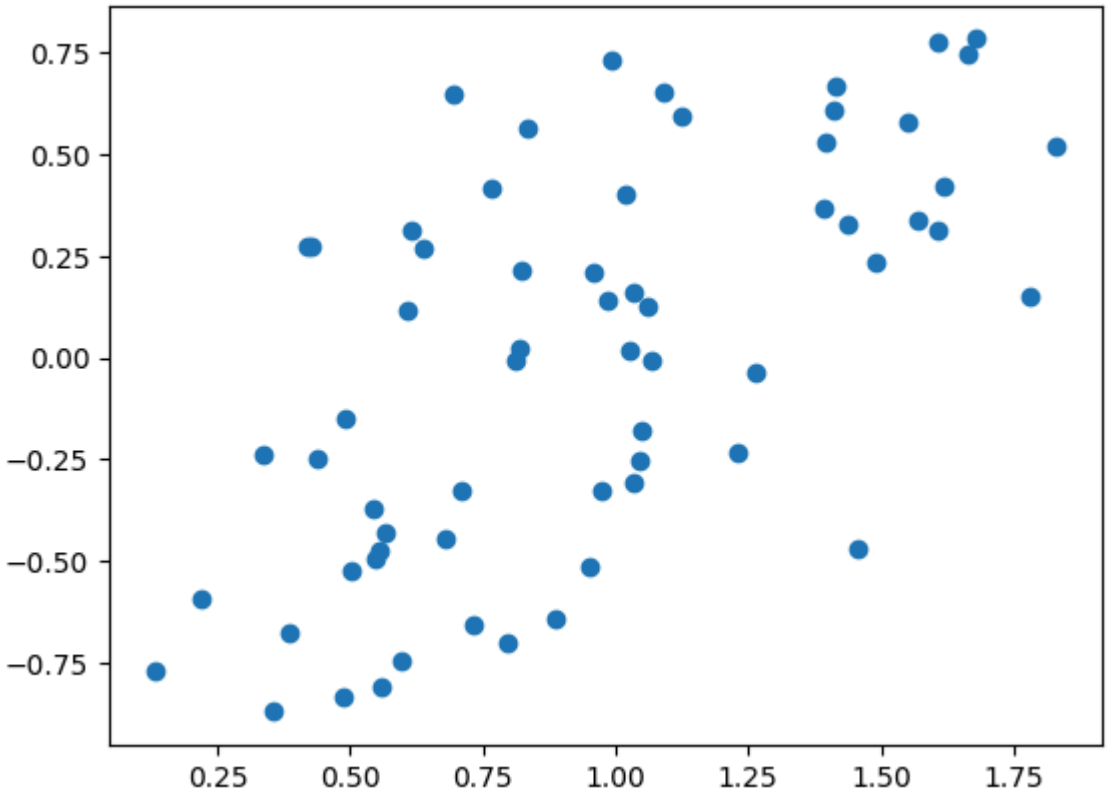
Out[8]:
```
<matplotlib.collections.PathCollection at 0x17d34954250>
```



x_4 vs. y

In [9]:
```
plt.scatter(x.T[3],y)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x17d34954550>
```



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.

In [10]:
```
left = np.linalg.inv(np.dot(x.T, x))
right = np.dot(y.T, x)
beta = np.dot(left, right)
beta
```

Out[10]:
```
array([ 0.079324  ,  0.34547193,  0.41346006,  0.29968547, -1.11888104])
```

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

In [11]:
```
pred = np.dot(x, beta)
pred
```

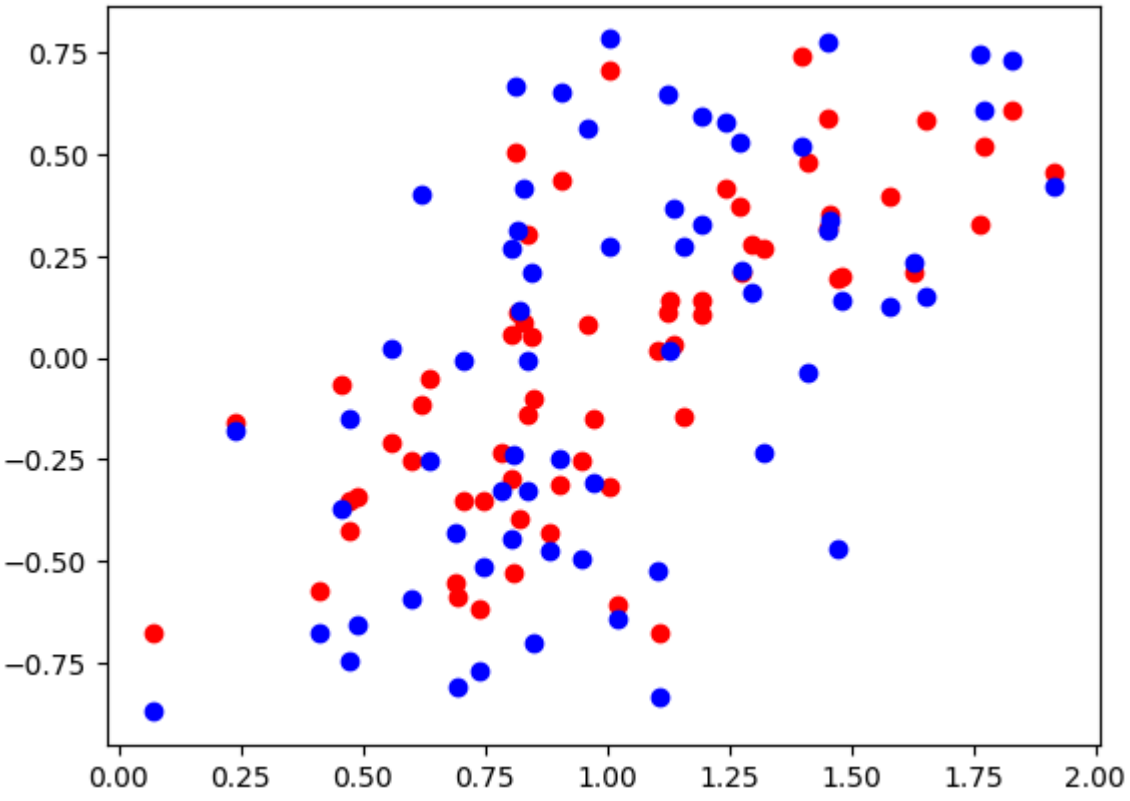
Out[11]:
```
array([-0.23511651, -0.57337132, -0.35250684, -0.67588167, -0.61807727,
       -0.34344966, -0.6079952 , -0.52956066, -0.42510123, -0.6768546 ,
       -0.58897956, -0.55622522, -0.25127487, -0.20705827, -0.15925778,
       -0.31208321, -0.10048502, -0.05111388, -0.35151656, -0.29603234,
       -0.15159598, -0.31778925, -0.42928768, -0.39849746, -0.14754238,
       -0.14258429, -0.06444804,  0.01537721, -0.25381145, -0.35251112,
        0.05746928,  0.03326615,  0.19301504, -0.11708078,  0.08838494,
        0.37370747,  0.21062488,  0.2688118 ,  0.11180347,  0.14227717,
        0.10378613,  0.05426149,  0.35318087,  0.11112027,  0.20208308,
        0.3180458 ,  0.39921541,  0.43836236,  0.50341764,  0.3049092 ,
        0.08013127,  0.47996966,  0.58298335,  0.60736512,  0.21179026,
        0.58945491,  0.73986215,  0.52108287,  0.45768518,  0.14172324,
        0.70680719,  0.32690637,  0.41673858,  0.27874718])
```

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

x_1 vs. y_p

In [12]:
```
plt.scatter(x.T[0], pred, c='red')
plt.scatter(x.T[0], y, c='b')
```

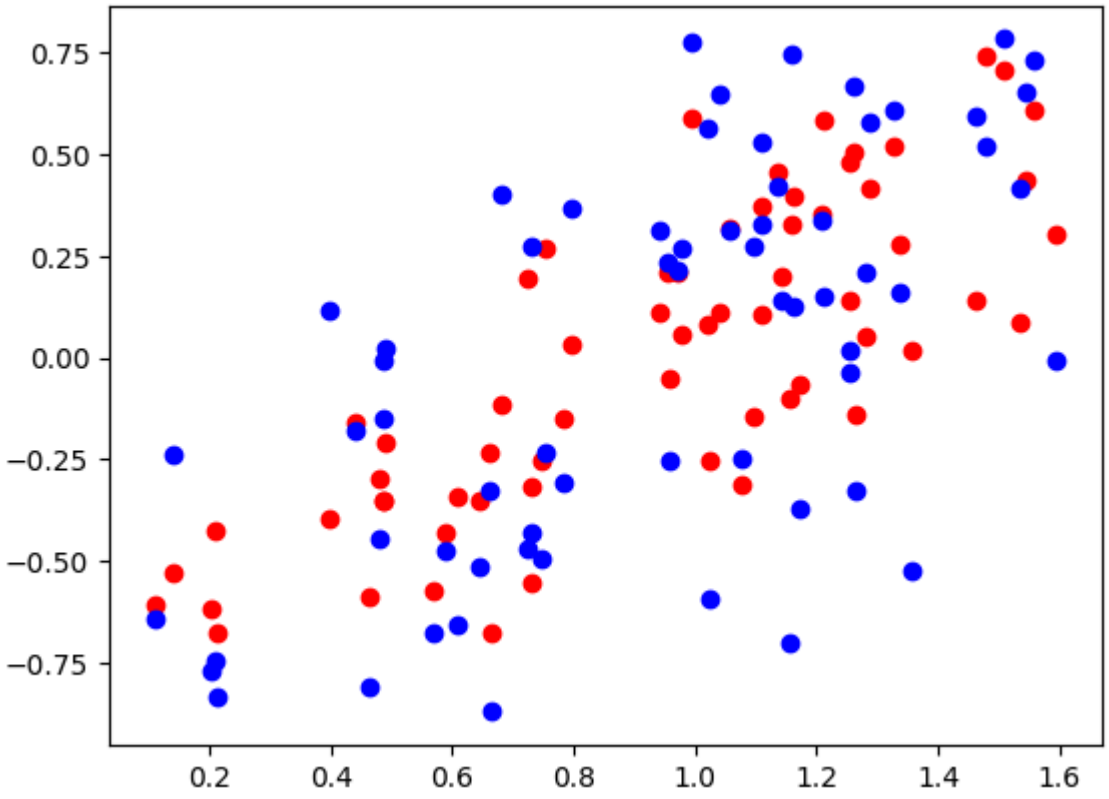
Out[12]:
```
<matplotlib.collections.PathCollection at 0x17d37a60350>
```



x_2 vs. y_p

In [13]:
```
plt.scatter(x.T[1], pred, c='red')
plt.scatter(x.T[1], y, c='b')
```

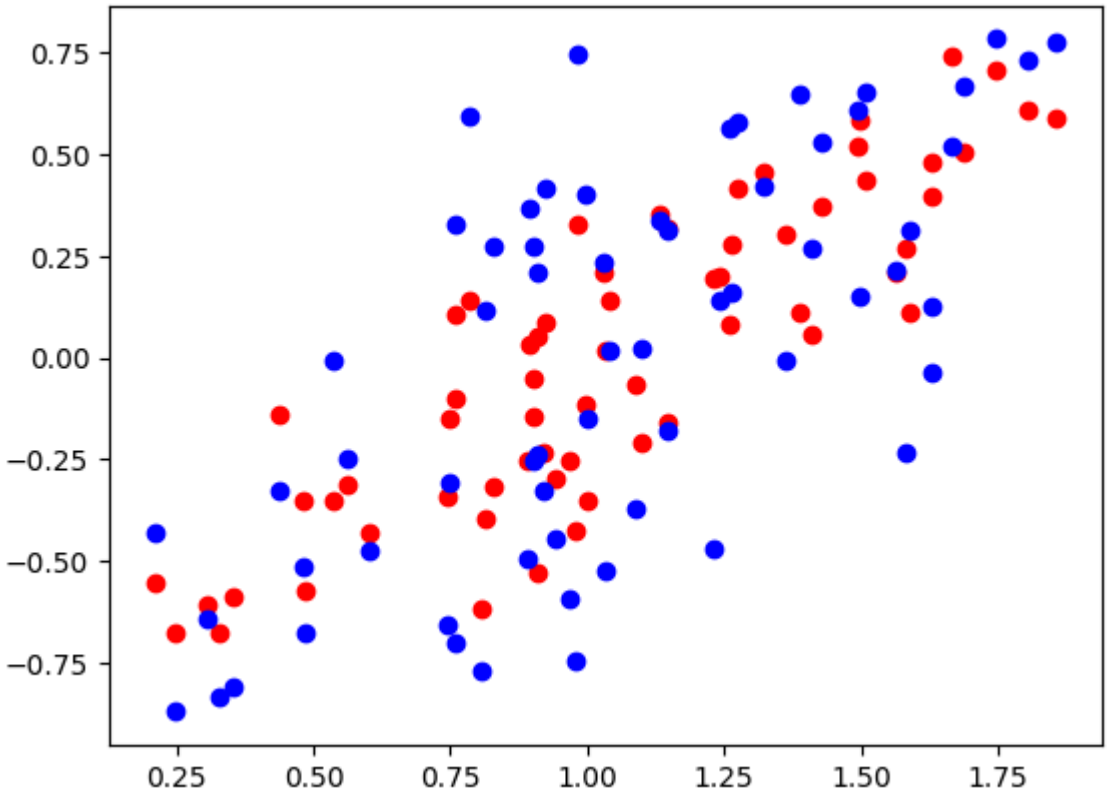
Out[13]: <matplotlib.collections.PathCollection at 0x17d37adaf50>



x_3 vs. y_p

In [14]:
```
plt.scatter(x.T[2], pred, c='red')
plt.scatter(x.T[2], y, c='b')
```

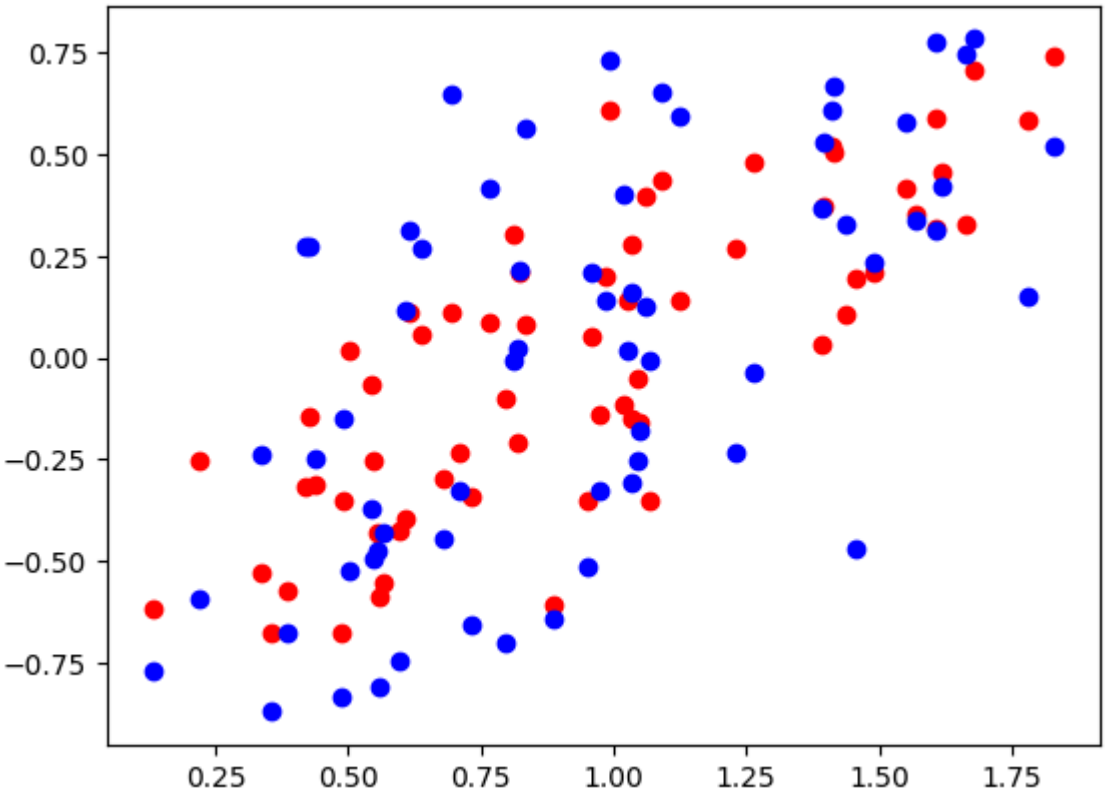
Out[14]: <matplotlib.collections.PathCollection at 0x17d37b64250>



x_4 vs. y_p

In [15]:
```
plt.scatter(x.T[3], pred, c='red')
plt.scatter(x.T[3], y, c='b')
```

Out[15]: <matplotlib.collections.PathCollection at 0x17d37bdc250>



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 [16]:
```
import pandas as pd
credit = pd.read_csv('../data/Credit.csv')
credit.head()
```

Out[16]:

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

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

In [17]:
```
columns = ['Income', 'Limit', 'Cards', 'Age', 'Education', 'Balance']
credx = credit[columns].values

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

Out[17]:
```
array([[1.48910e+01, 3.60600e+03, 2.00000e+00, ..., 1.10000e+01,
        3.33000e+02, 1.00000e+00],
       [1.06025e+02, 6.64500e+03, 3.00000e+00, ..., 1.50000e+01,
        9.03000e+02, 1.00000e+00],
       [1.04593e+02, 7.07500e+03, 4.00000e+00, ..., 1.10000e+01,
        5.80000e+02, 1.00000e+00],
       ...,
       [5.78720e+01, 4.17100e+03, 5.00000e+00, ..., 1.20000e+01,
        1.38000e+02, 1.00000e+00],
       [3.77280e+01, 2.52500e+03, 1.00000e+00, ..., 1.30000e+01,
        0.00000e+00, 1.00000e+00],
       [1.87010e+01, 5.52400e+03, 5.00000e+00, ..., 7.00000e+00,
        9.66000e+02, 1.00000e+00]])
```

In [18]:
```
credy = credit['Rating']
credy
```

Out[18]:
```
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 [19]:
```
credleft = np.linalg.inv(np.dot(credx.T, credx))
credright = np.dot(credy.T, credx)
credbeta = np.dot(credleft, credright)
credbeta
```

Out[19]:
```
array([ 9.48157743e-02,  6.42304413e-02,  4.67706085e+00,  8.06617460e-03,
       -2.30863025e-01,  8.18115721e-03,  3.10522106e+01])
```

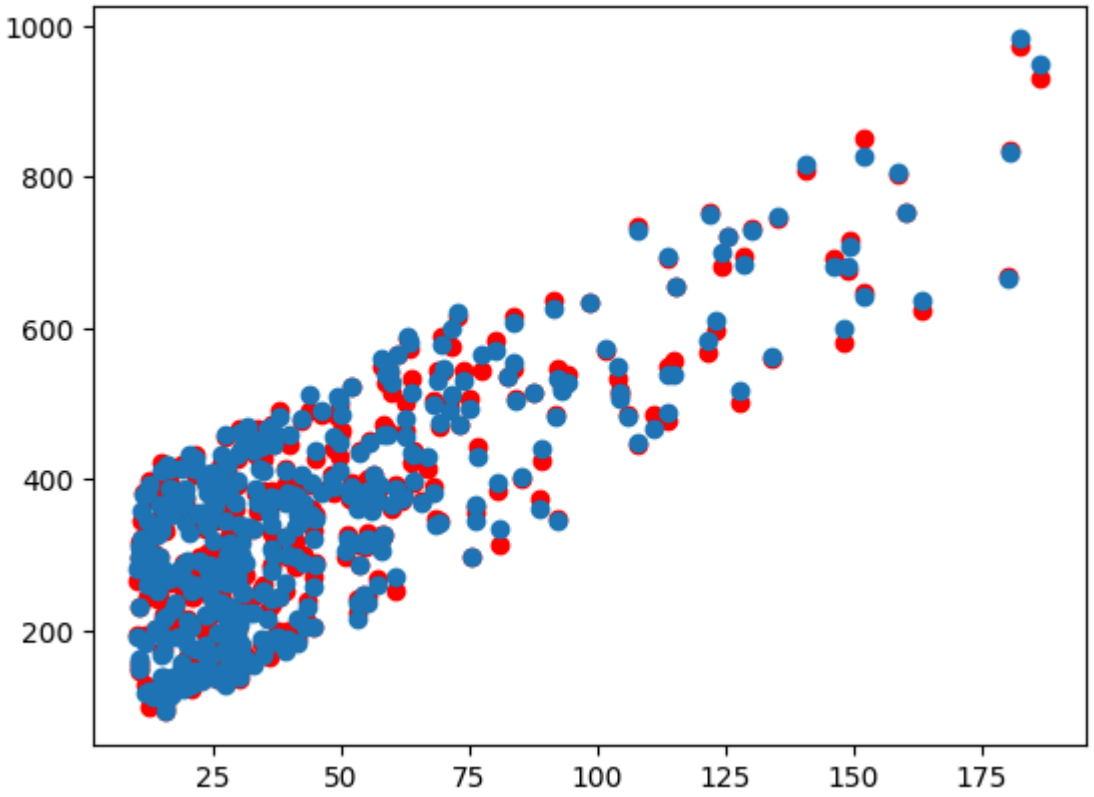
In [20]:
```
credpred = np.dot(credx, credbeta)
```

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

Income vs Credit Rating

In [27]:
```
plt.scatter(credx.T[0],credpred, c='r')
plt.scatter(credx.T[0],credy)
```

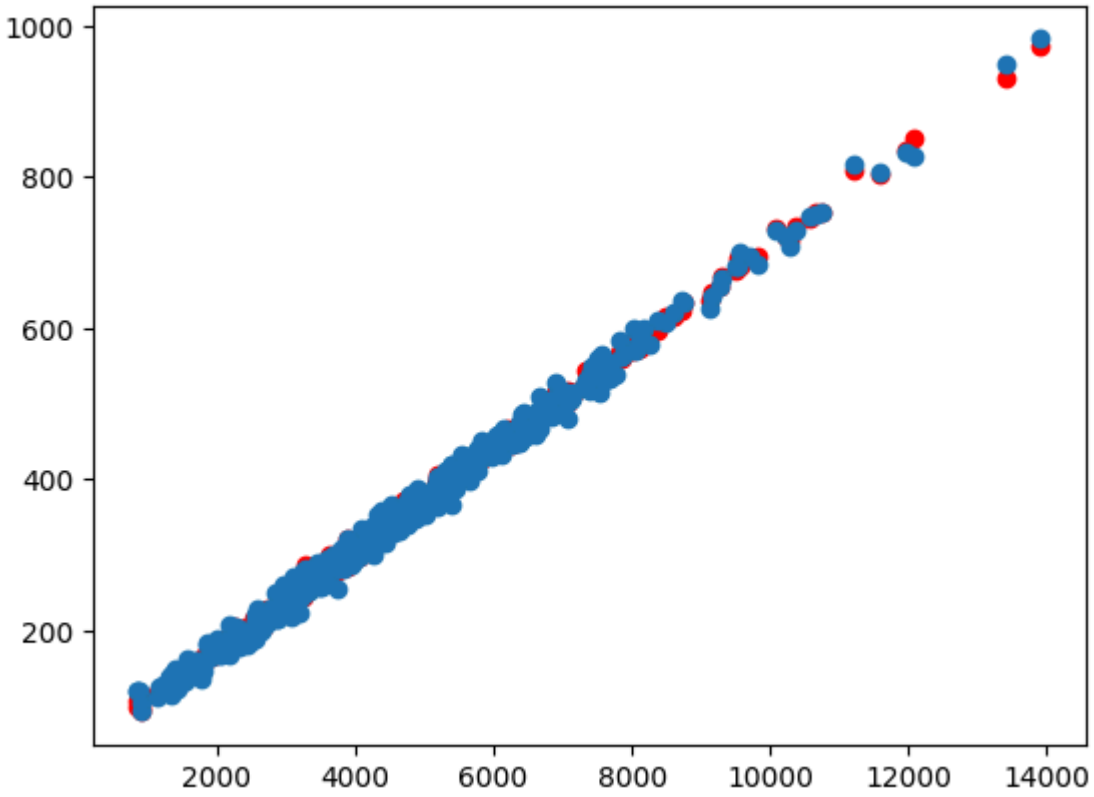
Out[27]: <matplotlib.collections.PathCollection at 0x17d349249d0>



Limit vs Credit Rating

In [28]:
```
plt.scatter(credx.T[1],credpred, c='r')
plt.scatter(credx.T[1],credy)
```

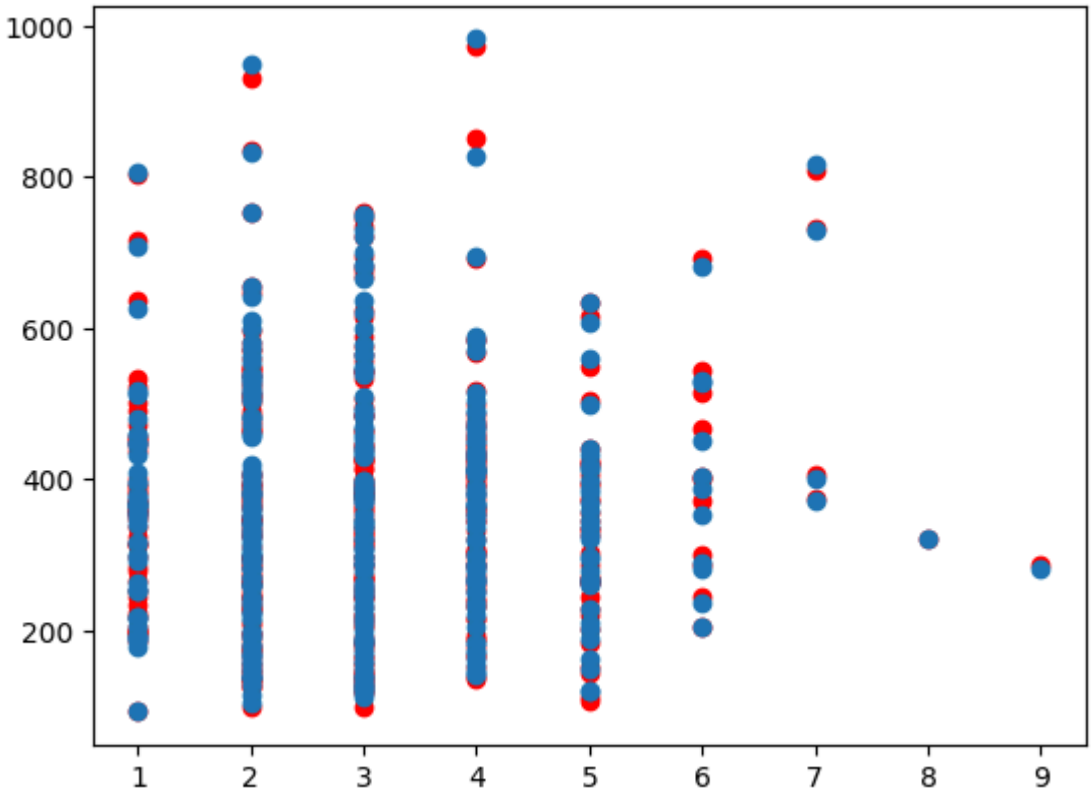
Out[28]: <matplotlib.collections.PathCollection at 0x17d399a17d0>



Cards vs Credit Rating

In [29]:
```
plt.scatter(credx.T[2],credpred, c='r')
plt.scatter(credx.T[2],credy)
```

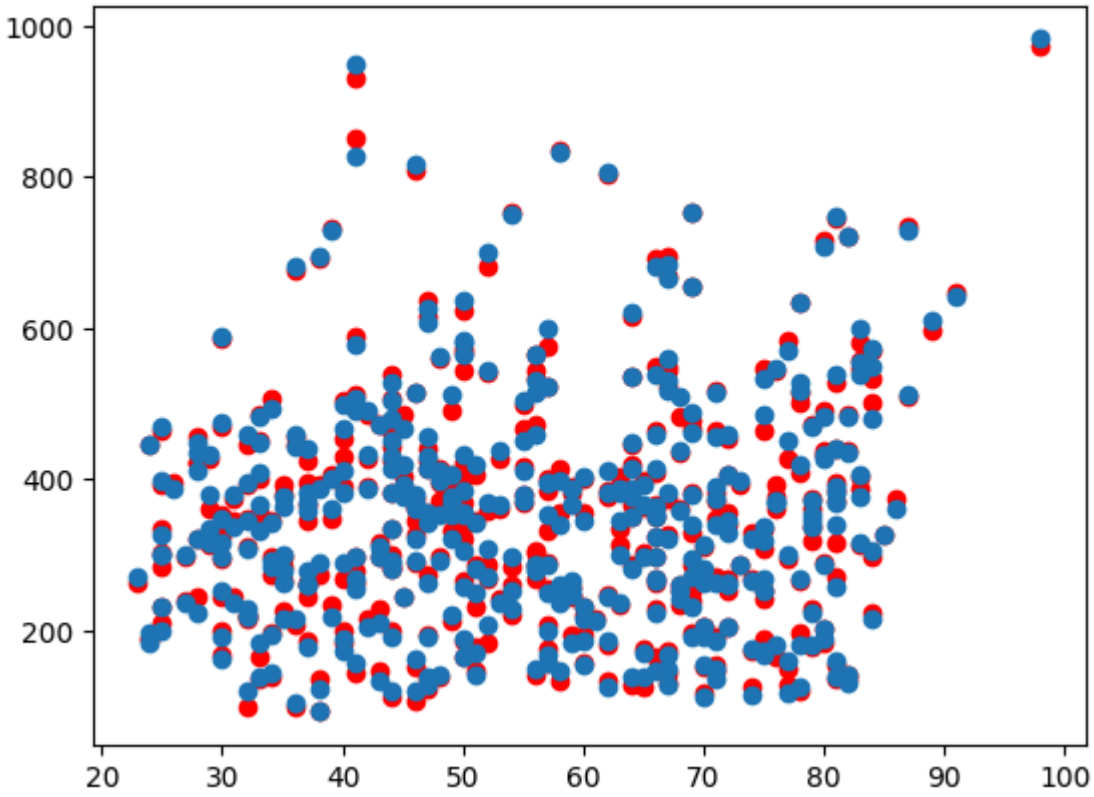
Out[29]:
```
<matplotlib.collections.PathCollection at 0x17d39d94c50>
```



Age vs Credit Rating

In [30]:
```
plt.scatter(credx.T[3],credpred, c='r')
plt.scatter(credx.T[3],credy)
```

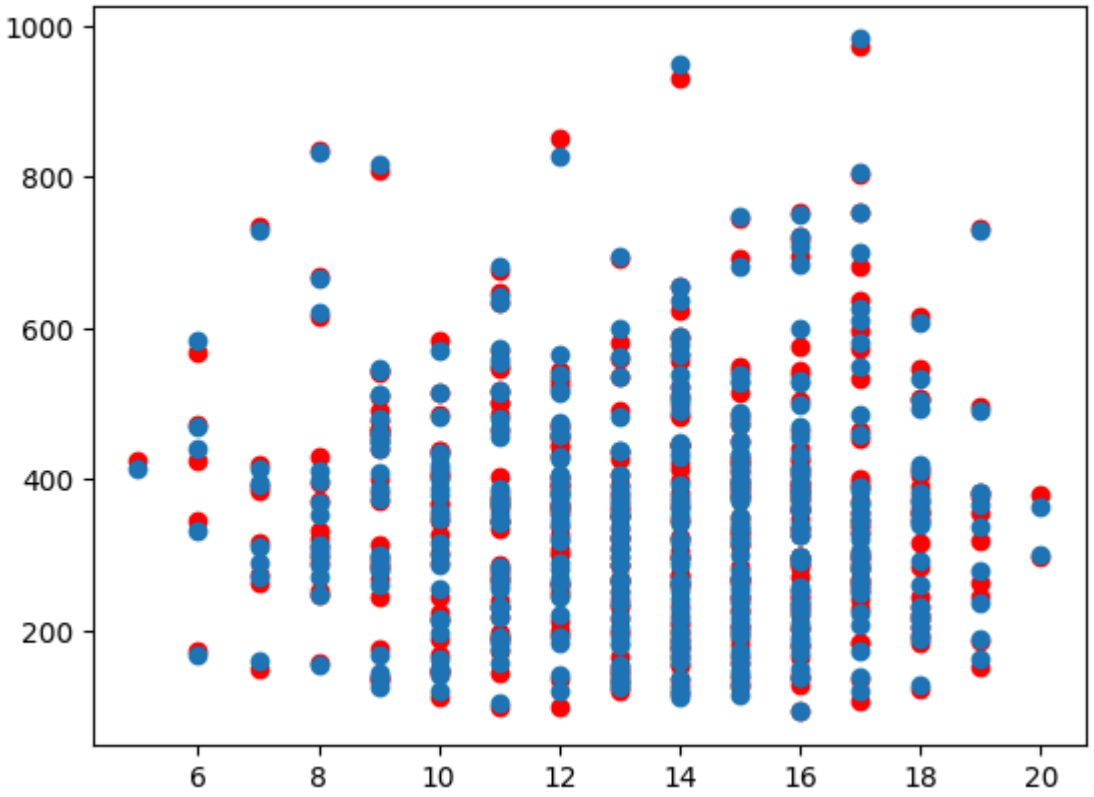
Out[30]:
```
<matplotlib.collections.PathCollection at 0x17d39dff190>
```



Education vs Credit Rating

In [31]:
```
plt.scatter(credx.T[4],credpred, c='r')
plt.scatter(credx.T[4],credy)
```

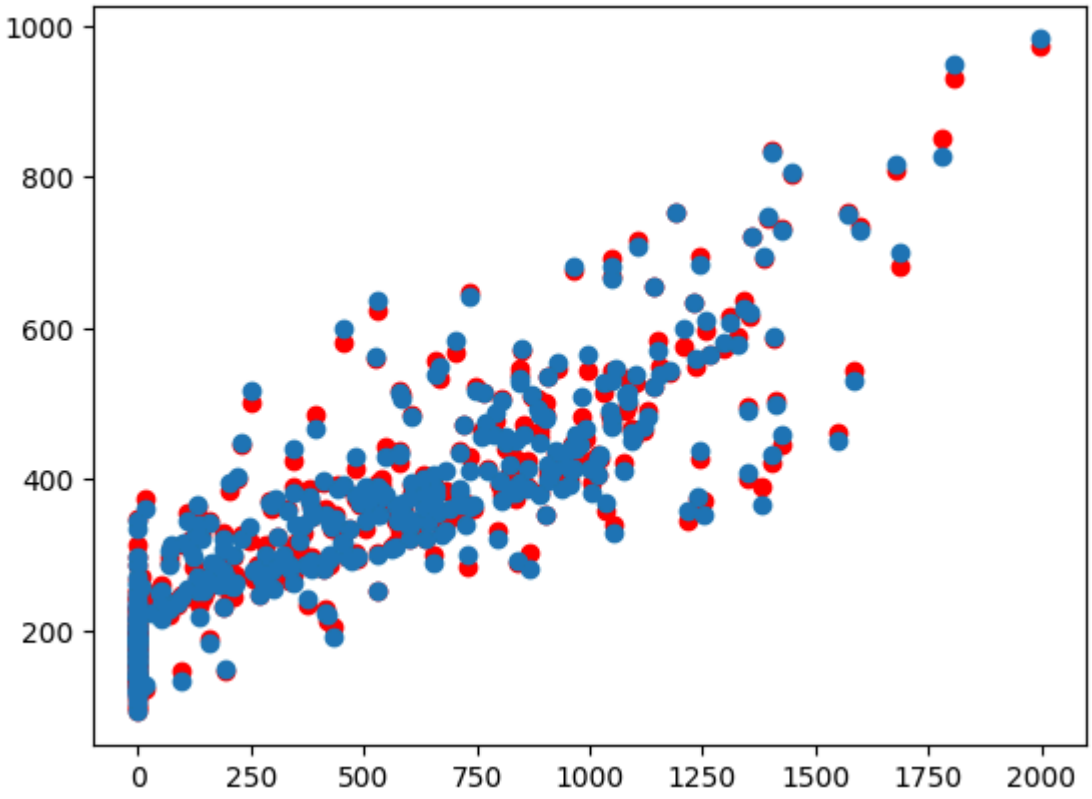
Out[31]:
```
<matplotlib.collections.PathCollection at 0x17d39dfef10>
```



Balance vs Credit Rating

In [32]:
```
plt.scatter(credx.T[5],credpred, c='r')
plt.scatter(credx.T[5],credy)
```

Out[32]:
```
<matplotlib.collections.PathCollection at 0x17d39ef3610>
```


1000
800
600
400
200
0
250
500
750
1000
1250
1500
1750
2000