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 [1]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

In [2]:
```
n = 64
x = np.linspace(0,1,n) + np.random.rand(4,n) # 4 dimensions
x = np.vstack([x, np.ones(len(x.T))]).T
y = np.linspace(0,1,n) + np.random.rand(n)
```

In [3]:
```
x
```

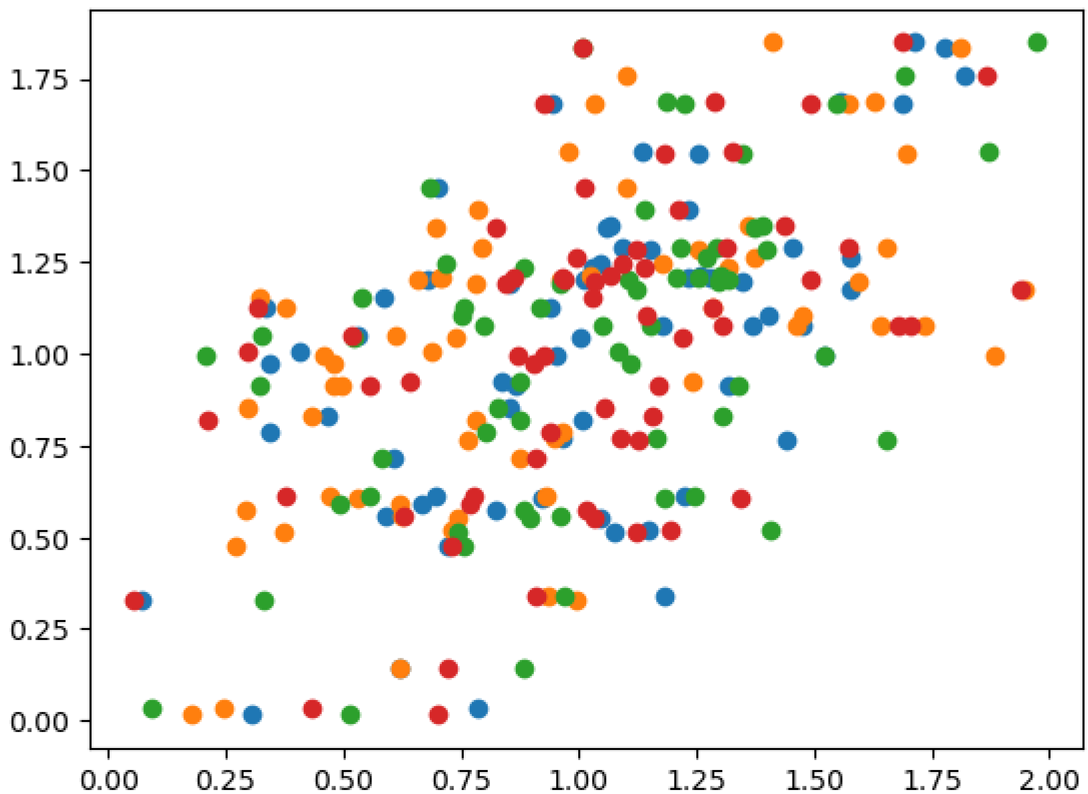
Out[3]:

```
array([[0.30472556, 0.17676081, 0.51327534, 0.70078443, 1.        ],
       [0.78497333, 0.2466505 , 0.09339956, 0.43402971, 1.        ],
       [0.62123828, 0.62023901, 0.88461168, 0.72404315, 1.        ],
       [0.07011426, 0.99257323, 0.33230041, 0.05505006, 1.        ],
       [1.00615485, 0.78289055, 0.87525988, 0.21238168, 1.        ],
       [0.52898956, 0.61175123, 0.32544255, 0.51782524, 1.        ],
       [0.82407283, 0.29057799, 0.88235672, 1.01433067, 1.        ],
       [0.69635388, 0.4691427 , 0.5550628 , 0.37811391, 1.        ],
       [0.72358618, 0.26885965, 0.75518896, 0.73232612, 1.        ],
       [0.95074424, 0.45748498, 0.20605124, 0.872455  , 1.        ],
       [1.04746923, 0.74448596, 0.89757176, 1.03221217, 1.        ],
       [0.85371322, 0.29741503, 0.82793796, 1.05474796, 1.        ],
       [0.96602411, 0.94956301, 1.16251012, 1.08657058, 1.        ],
       [0.58582745, 0.32162821, 0.53848564, 1.02855261, 1.        ],
       [0.86494443, 0.47970452, 0.32026267, 1.16941875, 1.        ],
       [0.4090757 , 0.68770116, 1.08494301, 0.29828718, 1.        ],
       [0.34428512, 0.96365881, 0.80204372, 0.93924621, 1.        ],
       [0.34210572, 0.47891735, 1.11004854, 0.90338016, 1.        ],
       [0.33626864, 0.37578943, 0.75625469, 0.31910031, 1.        ],
       [1.07540452, 0.37435878, 0.74464656, 1.12278139, 1.        ],
       [1.00452185, 0.73943971, 0.52361059, 1.21992728, 1.        ],
       [1.18174471, 0.9349873 , 0.96833899, 0.90866531, 1.        ],
       [0.67889687, 0.66007602, 1.31966692, 0.96938929, 1.        ],
       [0.66690215, 0.6195907 , 0.49218894, 0.76934628, 1.        ],
       [1.31820185, 0.49563561, 1.33862456, 0.55488114, 1.        ],
       [0.46800651, 0.43211485, 1.30329313, 1.15809837, 1.        ],
       [1.05655733, 0.69716121, 1.37509399, 0.82611885, 1.        ],
       [1.23357515, 0.78577483, 1.13790917, 1.21221318, 1.        ],
       [1.14902502, 0.72869268, 1.40537484, 1.1932247 , 1.        ],
       [0.59080188, 0.62545516, 0.96043653, 0.62876849, 1.        ],
       [1.25826415, 1.02552194, 1.30192354, 1.06675637, 1.        ],
       [1.15188575, 1.25363596, 1.39732691, 1.12105888, 1.        ],
       [0.92003617, 0.53201108, 1.18052392, 1.34448103, 1.        ],
       [0.85533463, 0.78309602, 0.96231209, 0.84567912, 1.        ],
       [0.83639152, 1.23957195, 0.87518661, 0.63920922, 1.        ],
       [0.60613996, 0.8744345 , 0.5803495 , 0.90825158, 1.        ],
       [1.22451652, 0.93052091, 1.2454674 , 0.77808094, 1.        ],
       [1.02947628, 1.31791338, 0.88334731, 1.13902649, 1.        ],
       [1.57577824, 1.37329812, 1.27227041, 0.99431429, 1.        ],
       [1.04745878, 1.17886986, 0.7191391 , 1.0940464 , 1.        ],
       [1.37081884, 1.46270043, 0.799574  , 1.30407328, 1.        ],
       [0.70122347, 1.10115792, 0.6845597 , 1.01134489, 1.        ],
       [1.40211889, 1.47378982, 0.75349005, 1.14316352, 1.        ],
       [1.23276478, 0.70721289, 1.20907838, 0.96582326, 1.        ],
       [1.2802017 , 0.70359621, 1.25325232, 0.86336933, 1.        ],
       [1.25440973, 1.69670863, 1.34895769, 1.18294865, 1.        ],
       [1.68612332, 1.57447073, 1.54808122, 0.92703235, 1.        ],
       [1.4544712 , 0.79558205, 1.21516085, 1.31181328, 1.        ],
       [1.44263919, 0.76418124, 1.65591208, 1.12487691, 1.        ],
       [0.94536131, 1.03320153, 1.22223576, 1.49159349, 1.        ],
       [1.06513225, 1.36078205, 1.39164495, 1.43769665, 1.        ],
       [1.47374689, 1.63948022, 1.05070638, 1.68087492, 1.        ],
       [1.55476479, 1.62980085, 1.18680327, 1.28938265, 1.        ],
       [1.00946926, 0.96027579, 1.1042521 , 1.49085572, 1.        ],
       [0.93931969, 1.28379161, 0.91635435, 1.2827318 , 1.        ],
       [1.34564952, 1.59450787, 1.29550622, 1.03189771, 1.        ],
       [1.77763331, 1.81188772, 1.00515535, 1.00895748, 1.        ],
       [1.13365862, 0.97613777, 1.87159927, 1.32484483, 1.        ],
       [1.52179492, 1.88567846, 1.52274598, 0.92443589, 1.        ],
       [1.81913456, 1.09972737, 1.69053419, 1.86551165, 1.        ],
```

```
       [1.57644672, 1.94738466, 1.12261768, 1.93769572, 1.        ],
       [1.17653831, 1.73548581, 1.15294747, 1.70464919, 1.        ],
       [1.71483775, 1.41096661, 1.97374068, 1.68940844, 1.        ],
       [1.09356642, 1.65228236, 1.29185169, 1.57471119, 1.        ]])
```

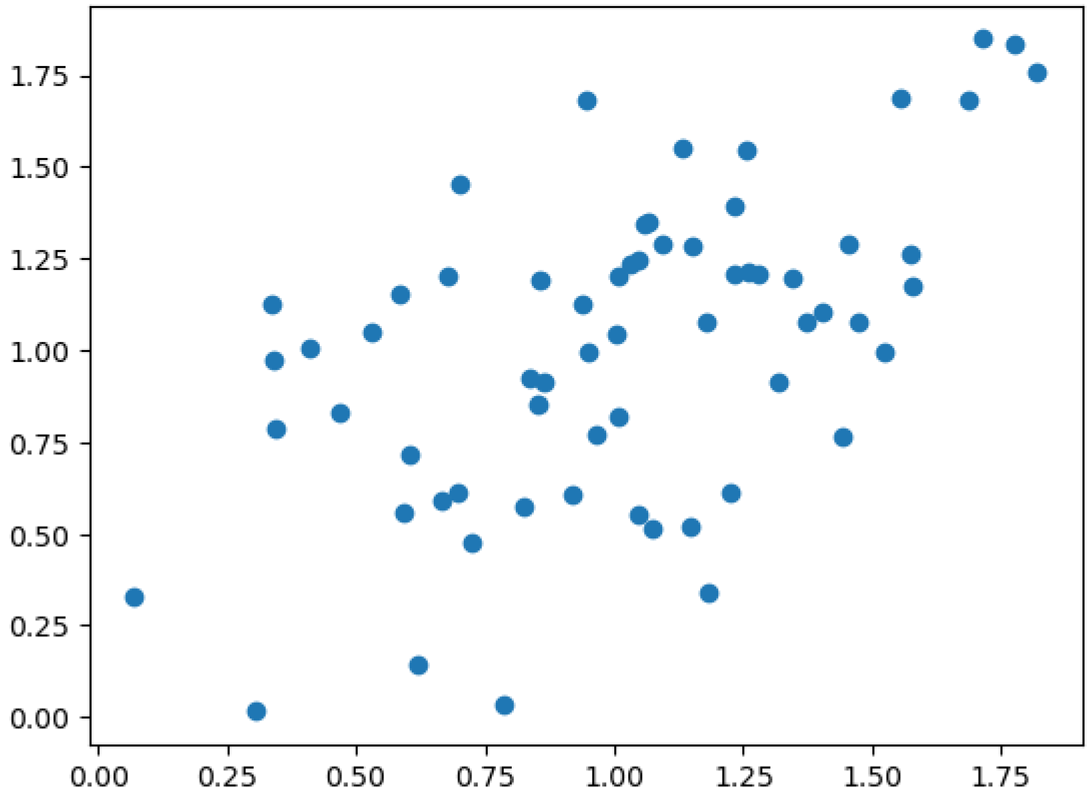
In [4]:
```
# Plotting All 4 Together
plt.scatter(x.T[0],y)
plt.scatter(x.T[1],y)
plt.scatter(x.T[2],y)
plt.scatter(x.T[3],y)
```

Out[4]: <matplotlib.collections.PathCollection at 0x2017bb284f0>



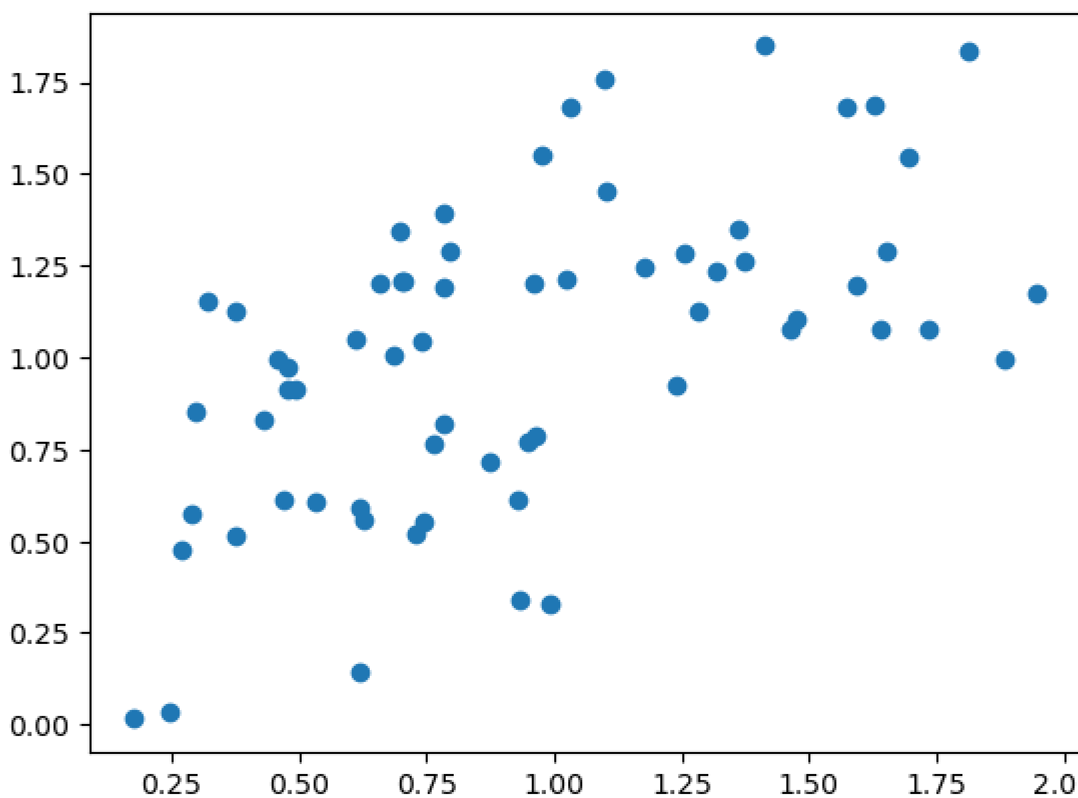
In [5]:
```
plt.scatter(x.T[0],y)
```

Out[5]: <matplotlib.collections.PathCollection at 0x2017c3e35b0>



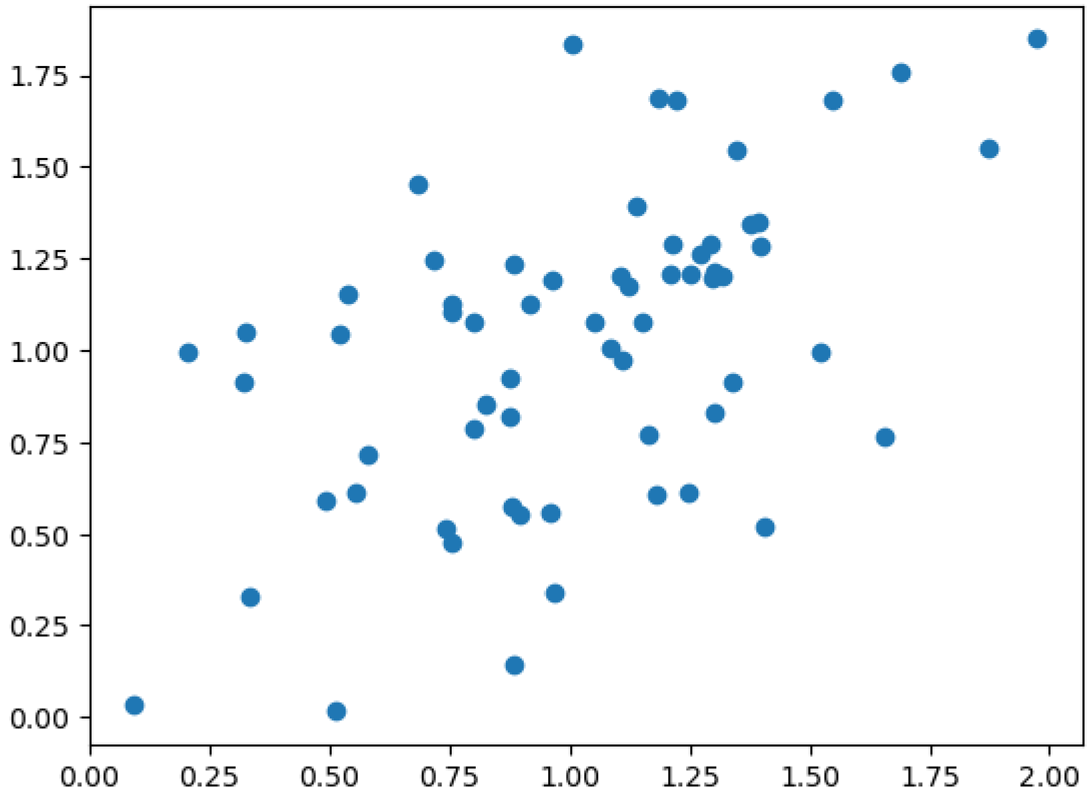
In [6]:
```
plt.scatter(x.T[1],y)
```

Out[6]: <matplotlib.collections.PathCollection at 0x2017bbae620>



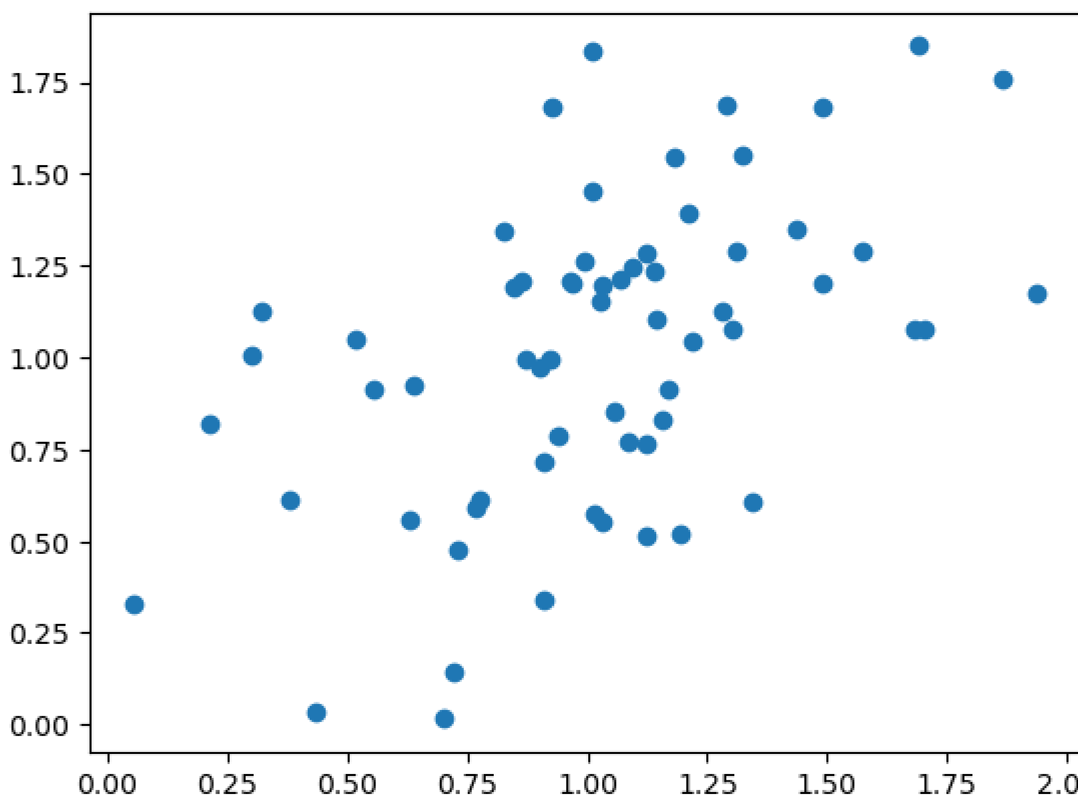
In [7]:
```
plt.scatter(x.T[2],y)
```

Out[7]: <matplotlib.collections.PathCollection at 0x2017bc360e0>



In [8]:
```
plt.scatter(x.T[3],y)
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x2017bb89ab0>
```



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 [9]:
```
# Revised Section
left = np.linalg.inv(np.dot(x.T, x))
left
right = np.dot(y.T, x)
right
np.dot(left, right)
```

Out[9]:
```
array([0.13319268, 0.28278606, 0.26444979, 0.17387034, 0.16555325])
```

In [10]:
```
# Checking Answer via Least-Squares
beta = np.linalg.lstsq(x,y)[0]
beta
```

```
C:\Users\Brett\AppData\Local\Temp\ipykernel_10428\2544466947.py:2: FutureWarning: `rc
ond` parameter will change to the default of machine precision times ``max(M, N)`` wh
ere M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to
keep using the old, explicitly pass `rcond=-1`.
  beta = np.linalg.lstsq(x,y)[0]
```

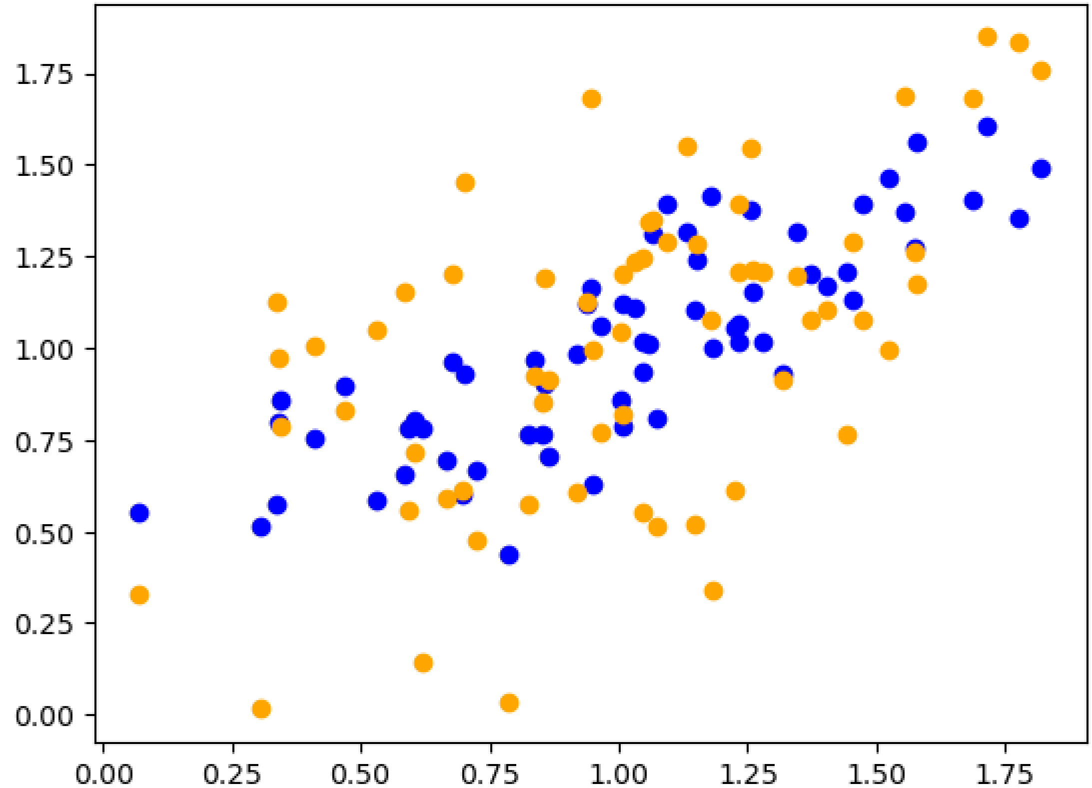
Out[10]:
```
array([0.13319268, 0.28278606, 0.26444979, 0.17387034, 0.16555325])
```

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

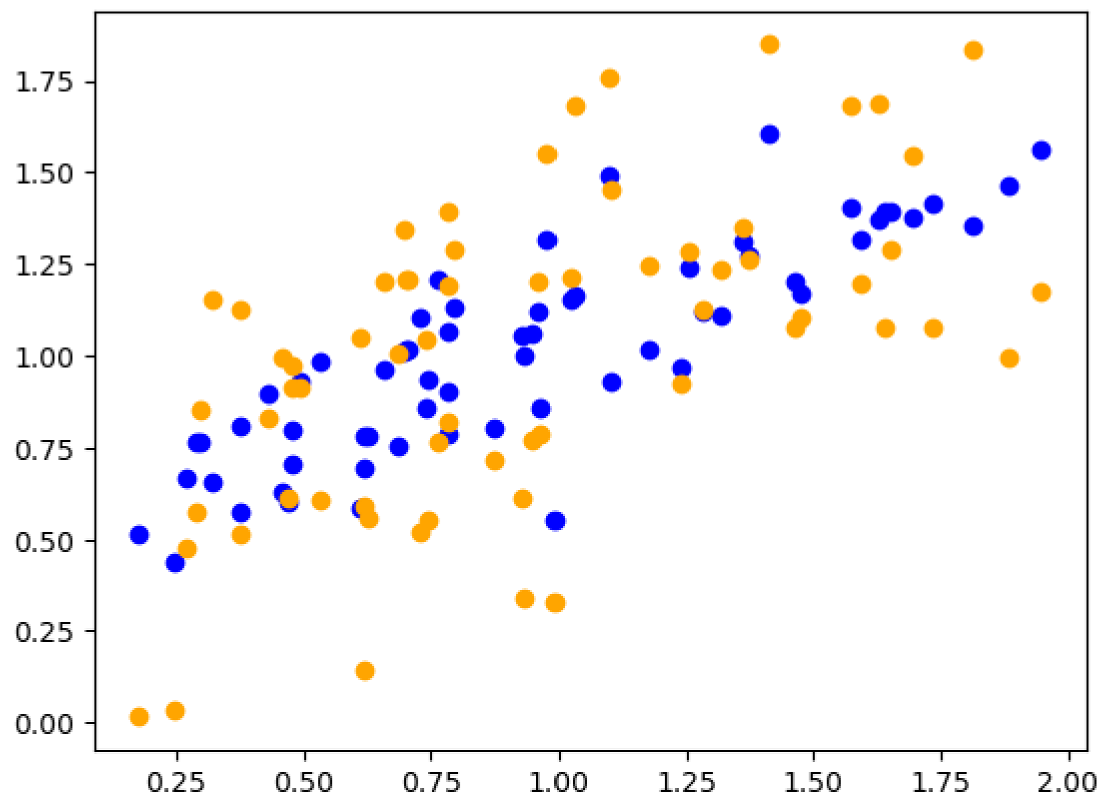
In [12]:
```
plt.scatter(x.T[0], pred, c='blue')
plt.scatter(x.T[0], y, c='orange')
```

Out[12]:
```
<matplotlib.collections.PathCollection at 0x2017bbfe410>
```



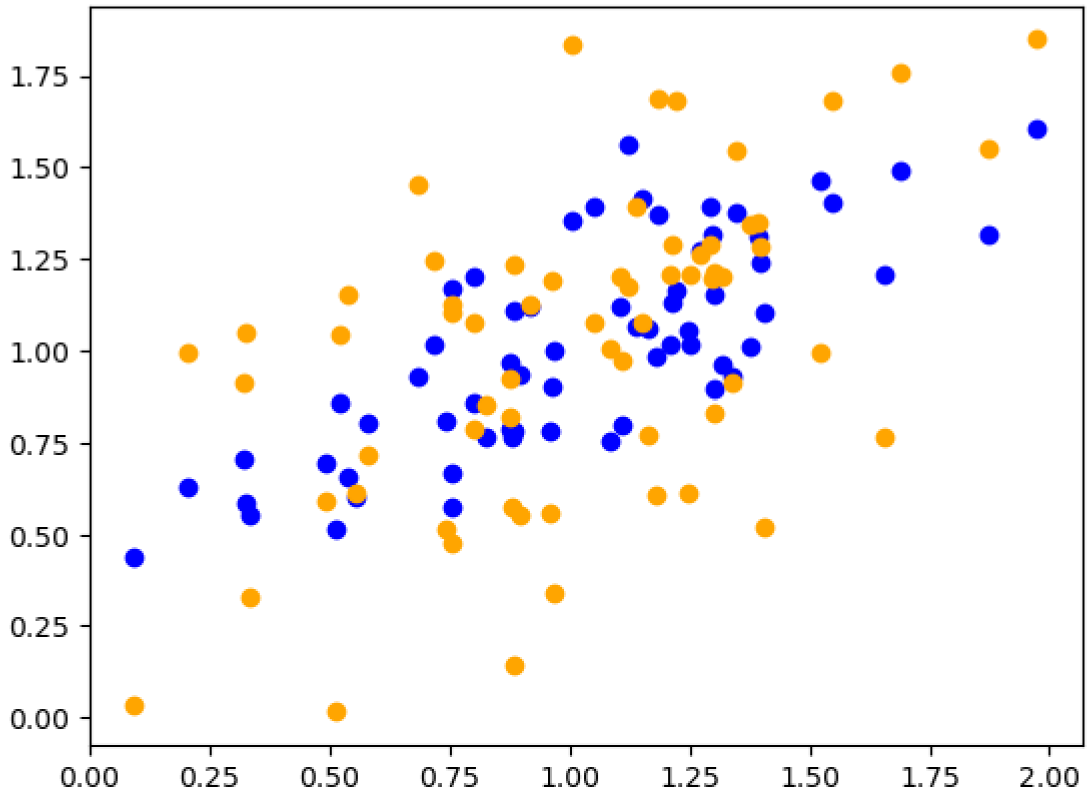
In [13]:
```
plt.scatter(x.T[1], pred, c='blue')
plt.scatter(x.T[1], y, c='orange')
```

Out[13]: <matplotlib.collections.PathCollection at 0x2017c49d750>



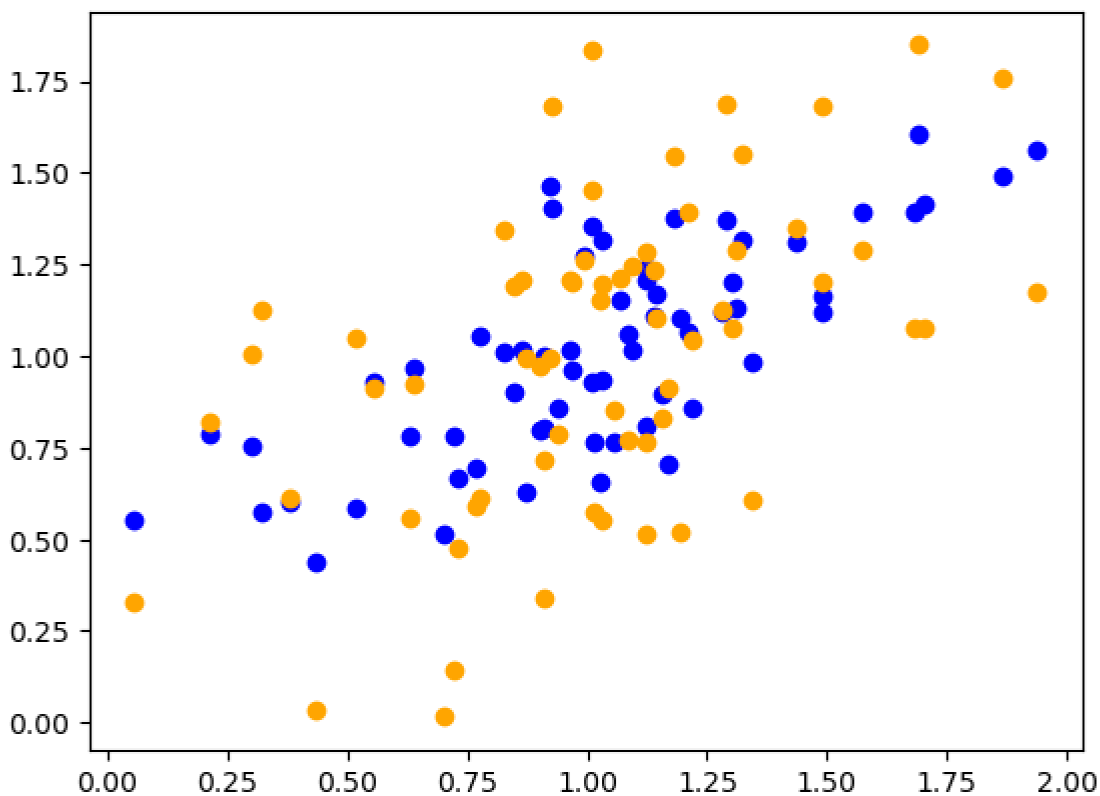
In [14]:
```
plt.scatter(x.T[2], pred, c='blue')
plt.scatter(x.T[2], y, c='orange')
```

Out[14]: <matplotlib.collections.PathCollection at 0x2017da64a30>



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

Out[15]: <matplotlib.collections.PathCollection at 0x2017daf8850>



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 [23]:
```
# Creating Model with Pandas
import pandas as pd
import numpy as np
credit = pd.read_csv('../data/Credit.csv')
credit.head()
```

Out[23]:

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

In [24]:
```
# Had to read back in code for 'X' for linear model
columns = ['Income', 'Limit','Cards','Age','Education']
X = credit[columns].values

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

In [25]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
rating_lr = LinearRegression().fit(X, y, sample_weight=None)
pct_error = (rating_lr.predict(X) - y) / y
abs(pct_error).mean()
```

Out[25]:
```
0.02822992430275092
```

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

In [26]:
```
columns = ['Income', 'Limit','Cards','Age','Education']
X = credit[columns].values

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

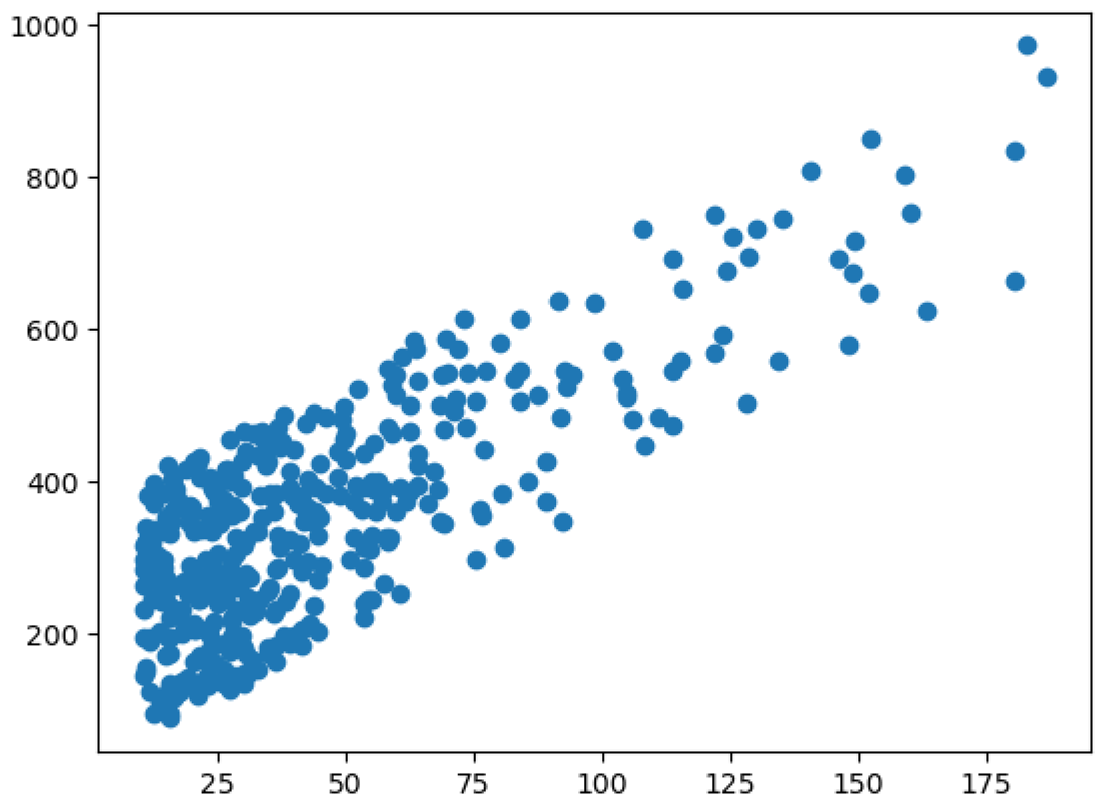
In [27]:
```
y = credit['Rating']
y
```

Out[27]:
```
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
```

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

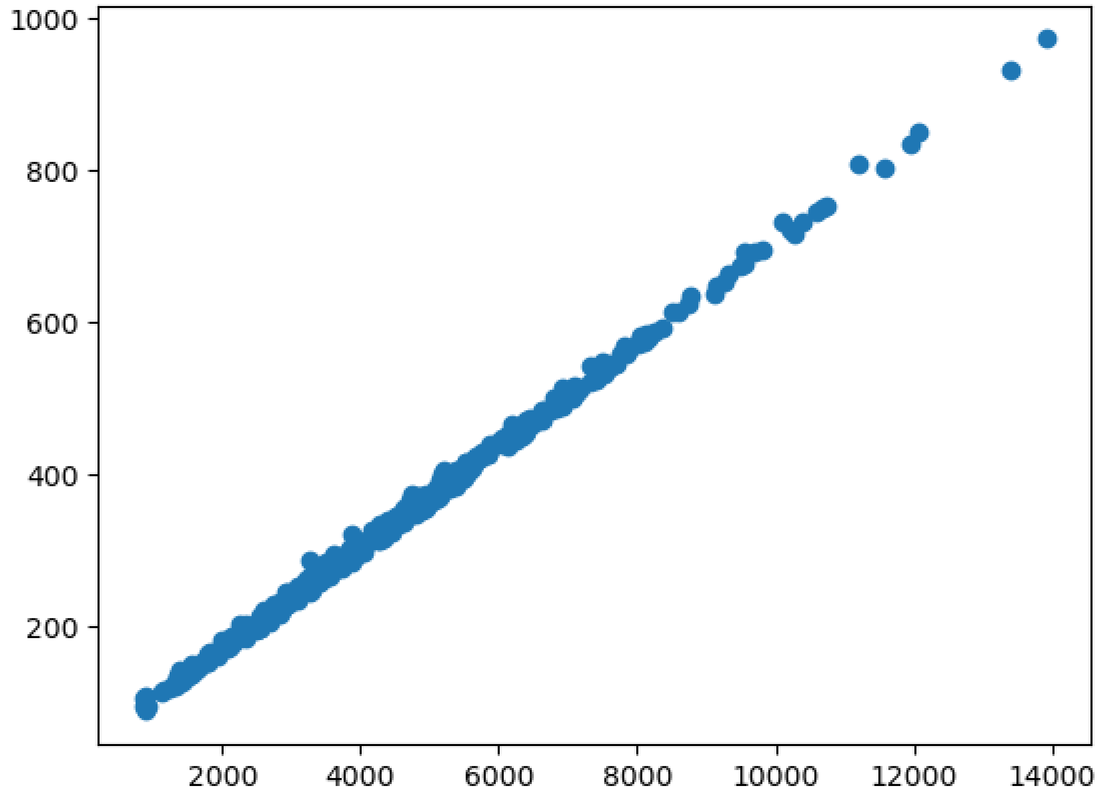
In [28]:
```
X[:, 0]
y
plt.scatter(X[:, 0],rating_lr.predict(X))
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x201028affa0>
```



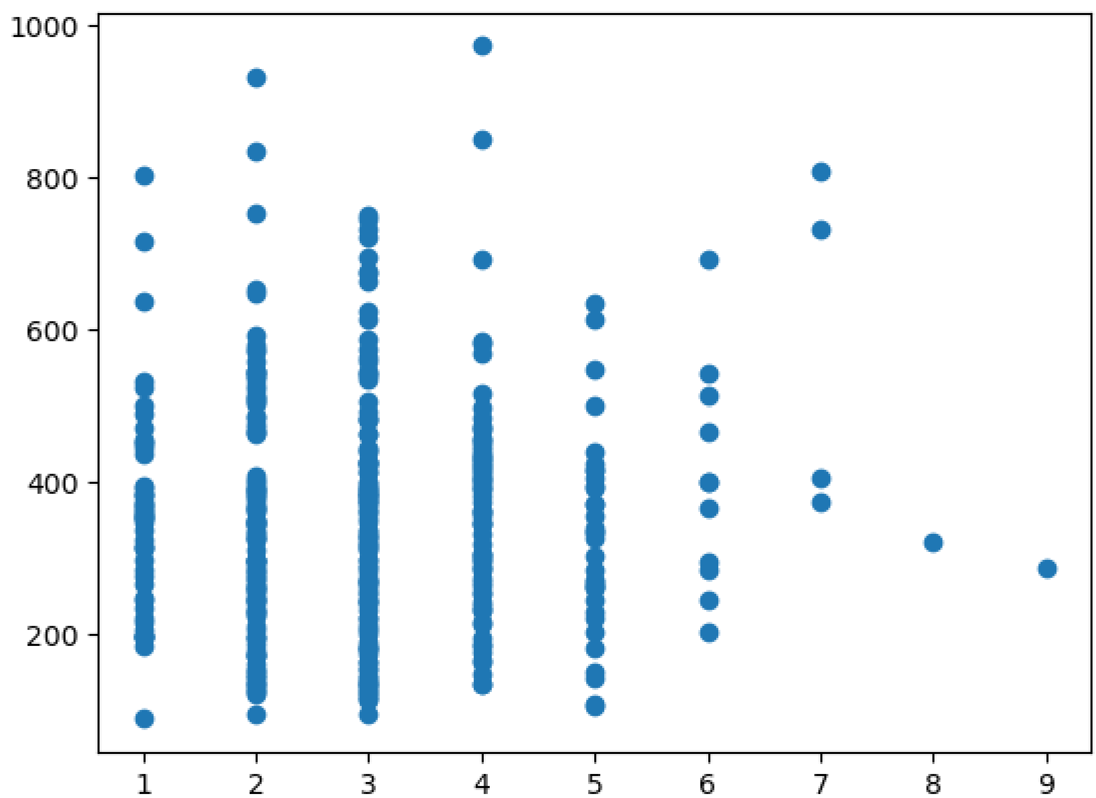
In [29]:
```
plt.scatter(X[:, 1],rating_lr.predict(X))
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x20102910b20>
```



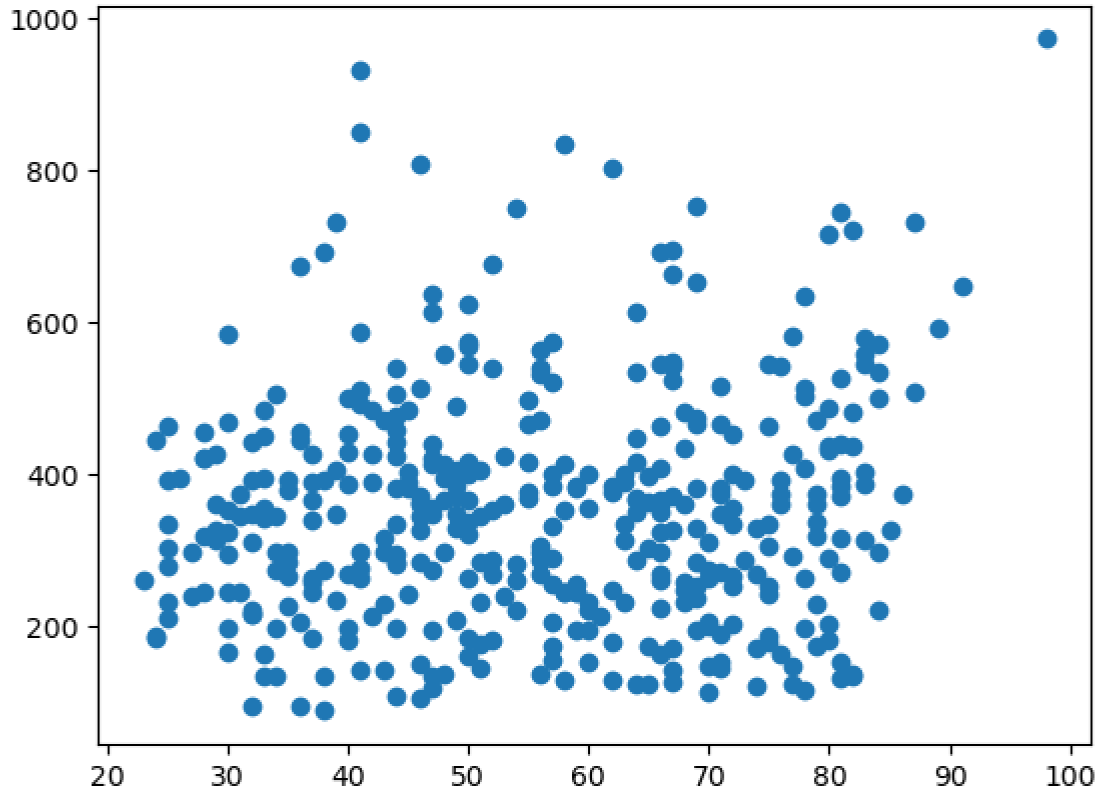
In [30]:
```
plt.scatter(X[:, 2],rating_lr.predict(X))
```

Out[30]: <matplotlib.collections.PathCollection at 0x2010296e320>



In [31]:
```
plt.scatter(X[:, 3],rating_lr.predict(X))
```

Out[31]: <matplotlib.collections.PathCollection at 0x20102b448b0>



In [32]:
```
plt.scatter(X[:, 4],rating_lr.predict(X))
```

Out[32]: <matplotlib.collections.PathCollection at 0x20102ba6a10>