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 [7]:
```
# Loading packages
import numpy as np
import matplotlib.pyplot 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

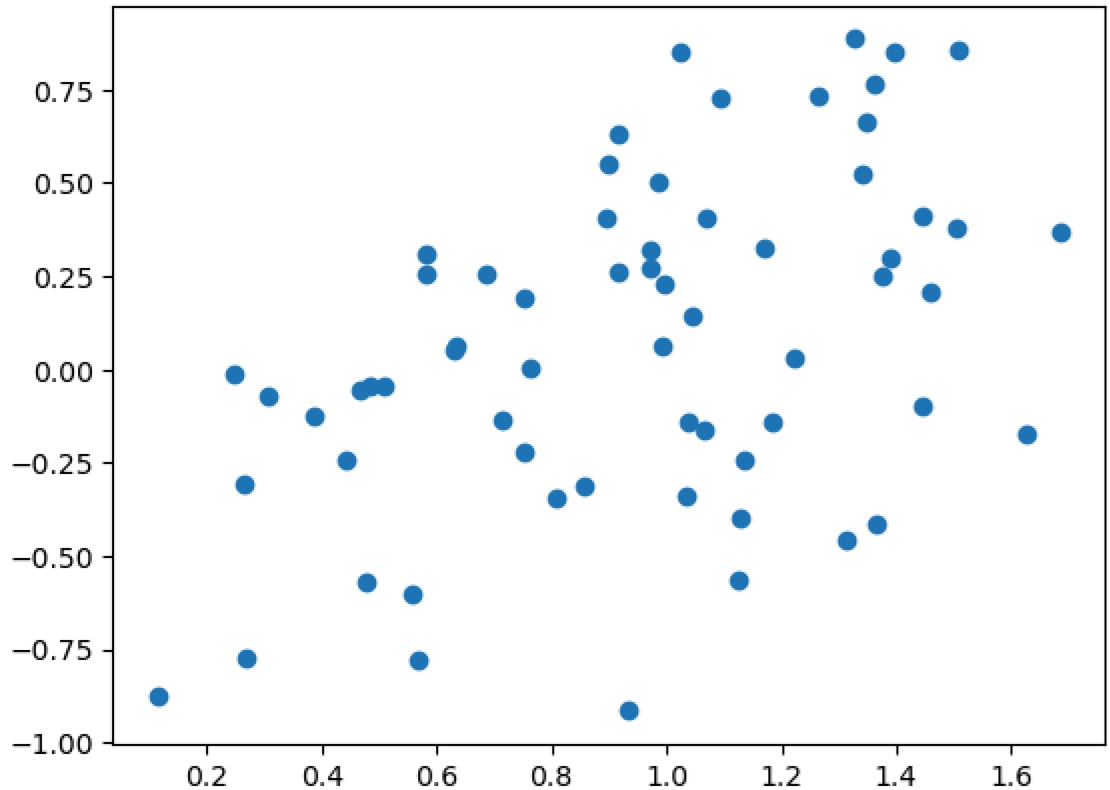
In [25]:
```
# Creating 4-dimensional dataset with 64 elements
np.random.seed(33)  # For reproducibility
n = 64

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
```

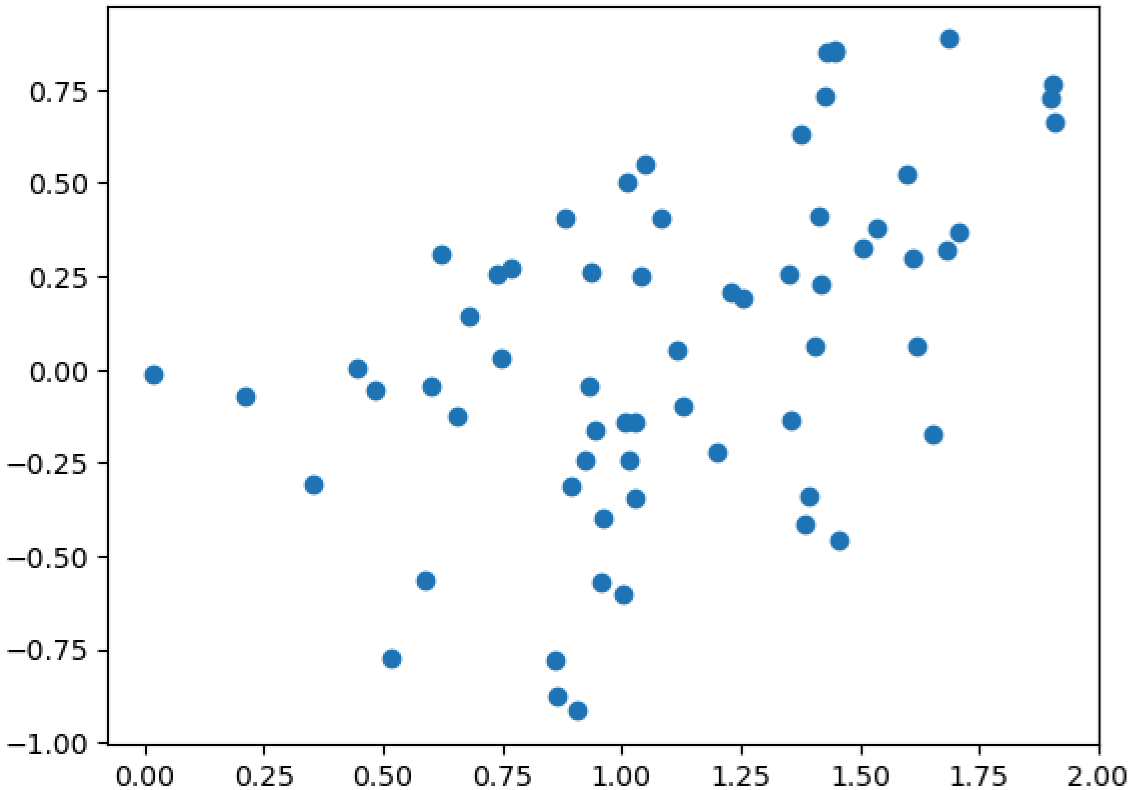
In [26]:
```
# x1 vs. y
plt.scatter(x.T[0], y)
```

Out[26]: <matplotlib.collections.PathCollection at 0x1358365fee0>



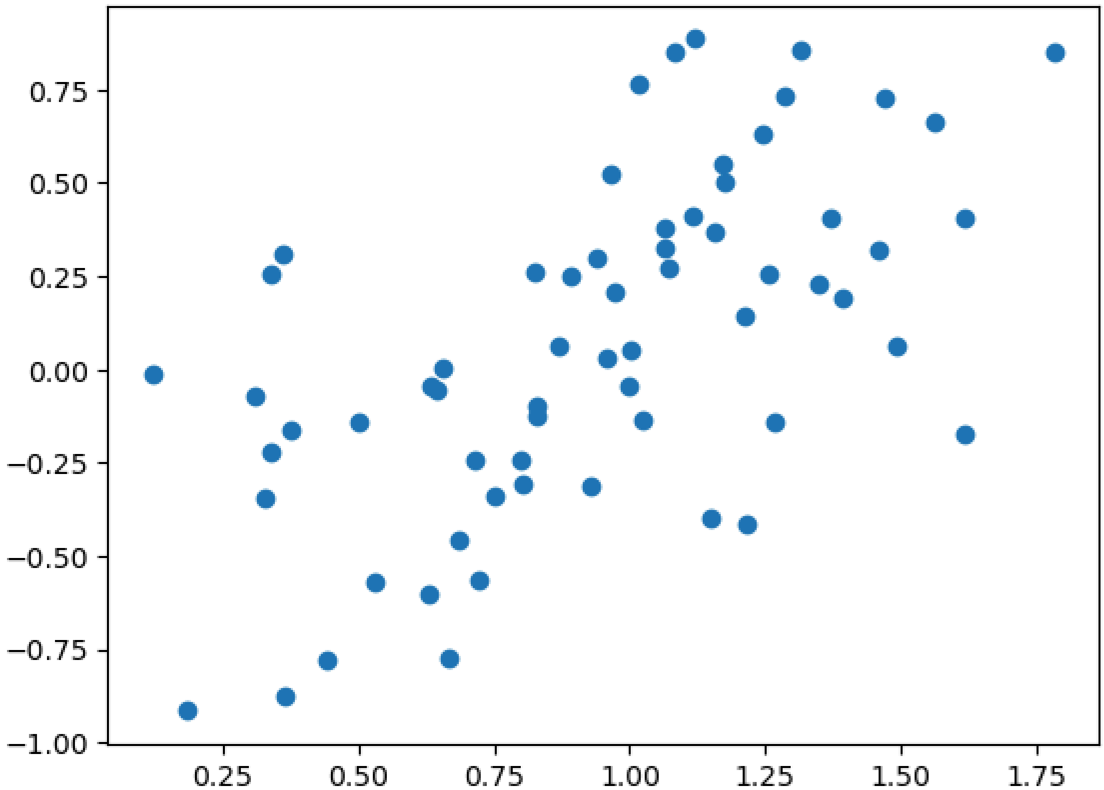
In [22]:
```
# x2 vs. y
plt.scatter(x.T[1], y)
```

Out[22]: <matplotlib.collections.PathCollection at 0x1358244b550>



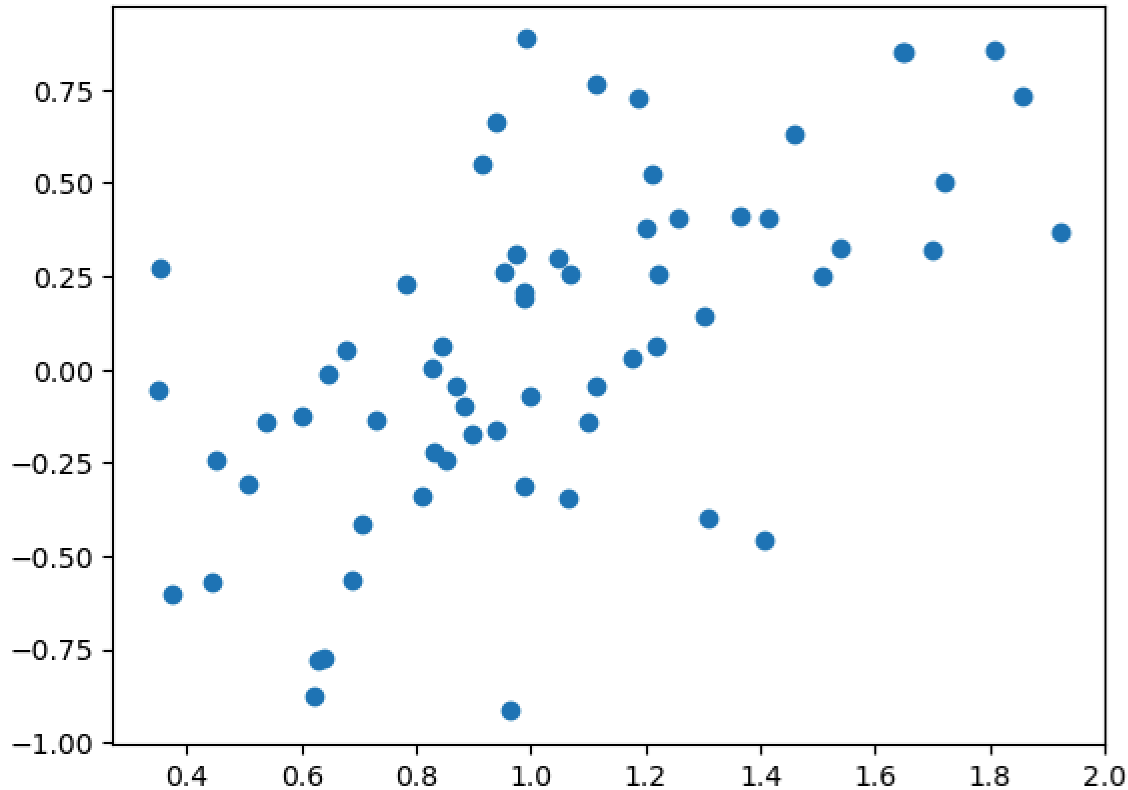
In [23]:
```
# x3 vs. y
plt.scatter(x.T[2], y)
```

Out[23]: <matplotlib.collections.PathCollection at 0x13583470640>



In [24]:
```
# x4 vs. y
plt.scatter(x.T[3], y)
```

Out[24]: <matplotlib.collections.PathCollection at 0x135834bdc70>



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 [39]:
```
# Finding coefficients
left = np.linalg.inv(np.dot(x.T,x))
right = np.dot(y.T, x)
np.dot(left,right)
```

Out[39]: array([-0.02421962, 0.09554617, 0.4453871 , 0.48721701, -0.93227561])

In [43]:
```
# Another way to find coefficients
beta = np.linalg.lstsq(x,y,rcond=-1)[0]
beta
```

Out[43]: array([-0.02421962, 0.09554617, 0.4453871 , 0.48721701, -0.93227561])

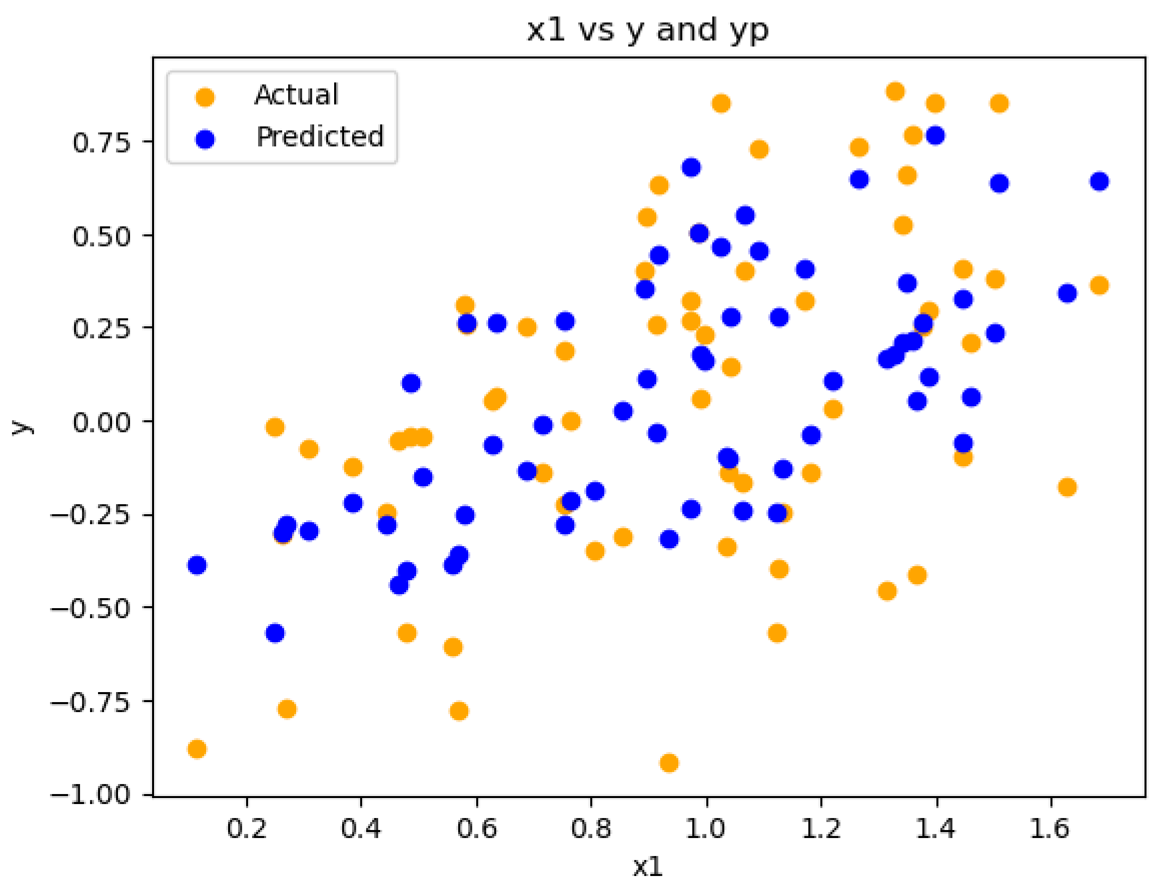
In [48]:
```
# Predicting values of y
pred = np.dot(x, beta)
pred
```

Out[48]:
```
array([-0.56745465, -0.43915582, -0.27907147, -0.29446161, -0.31541785,
       -0.30042673, -0.38672665, -0.24311076, -0.18798004, -0.06569897,
       -0.24628076, -0.35908266, -0.28044949, -0.38757113, -0.21690155,
        0.10872896, -0.12914151, -0.21194393, -0.1326253 , -0.27893707,
       -0.10167363, -0.2513246 , -0.23304064, -0.15016034,  0.26235148,
       -0.39995731,  0.05185371,  0.1010301 ,  0.2819074 , -0.0358975 ,
       -0.06040237,  0.1669894 , -0.09386495,  0.02773308,  0.35199497,
        0.2634803 ,  0.28136624,  0.26968365,  0.11383764,  0.06469109,
        0.16012043,  0.17865033,  0.26483272, -0.0076004 ,  0.40698894,
        0.32946217,  0.23776233,  0.50201277,  0.5552791 , -0.03318579,
        0.11623091,  0.68280885,  0.3430978 ,  0.20837771,  0.44305758,
        0.65116065,  0.63595341,  0.455077  ,  0.37046502,  0.46759004,
        0.64168786,  0.21247587,  0.17786747,  0.76648585])
```

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

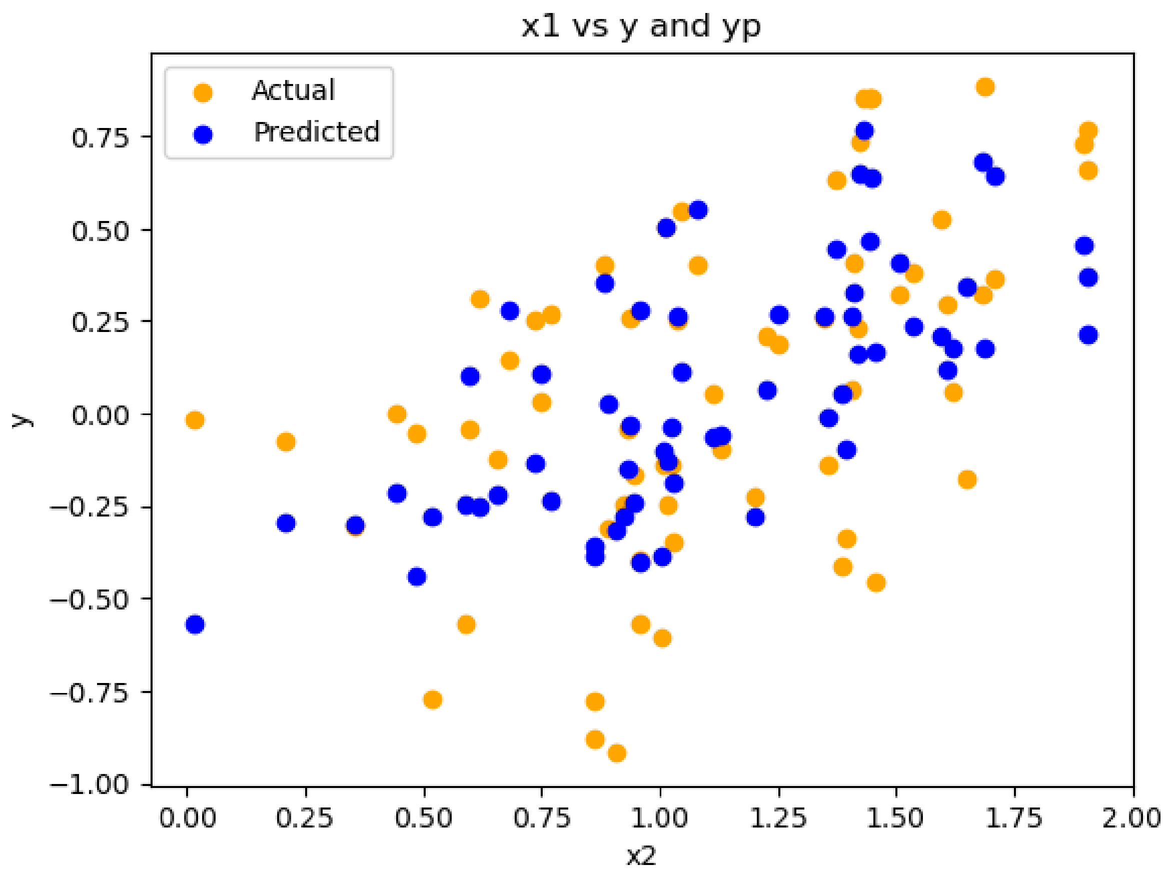
In [57]:
```
# x1 vs. y & yp
plt.scatter(x.T[0], y, c='orange', label = "Actual")
plt.scatter(x.T[0], pred, c='blue', label = "Predicted")
plt.xlabel('x1')
plt.ylabel('y')
plt.title('x1 vs y and yp')
plt.legend()
```

Out[57]: `<matplotlib.legend.Legend at 0x13584d859a0>`



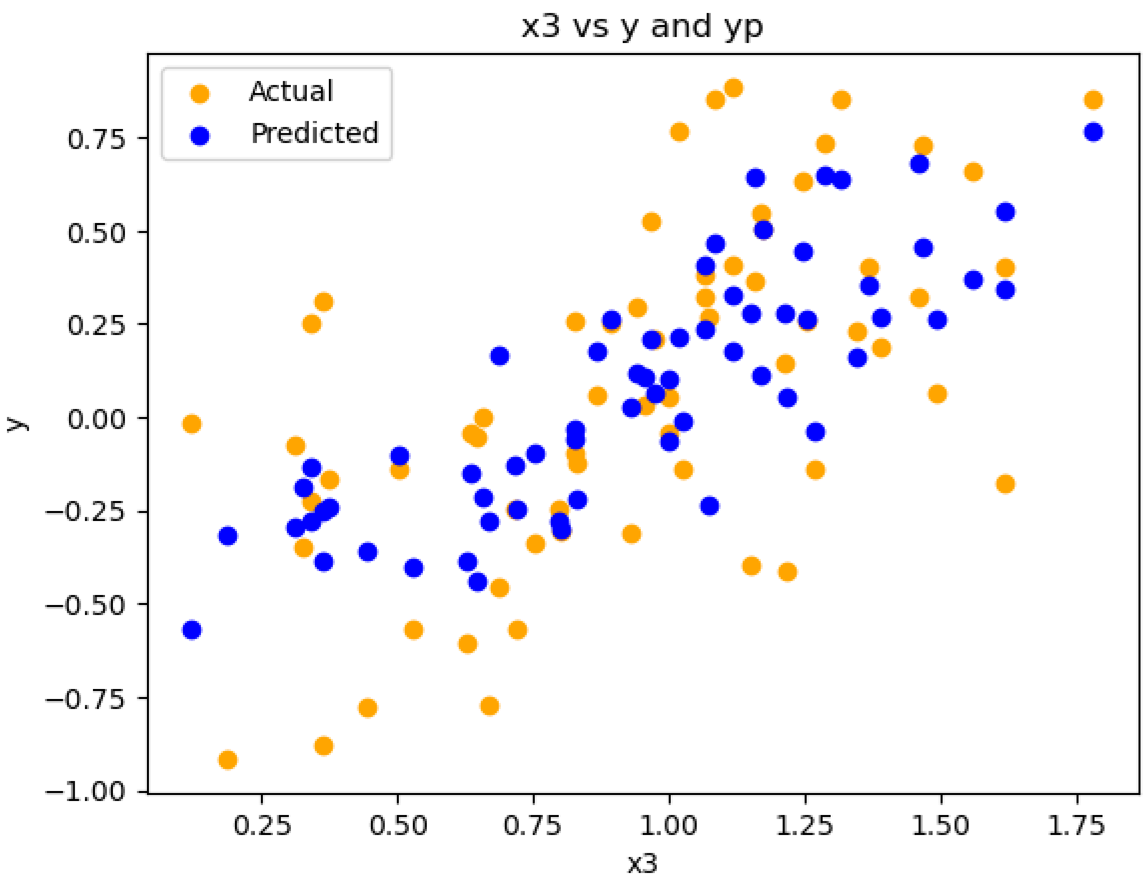
In [58]:
```
# x2 vs. y & yp
plt.scatter(x.T[1], y, c='orange', label = "Actual")
plt.scatter(x.T[1], pred, c='blue', label = "Predicted")
plt.xlabel('x2')
plt.ylabel('y')
plt.title('x1 vs y and yp')
plt.legend()
```

Out[58]: <matplotlib.legend.Legend at 0x13584d85f70>



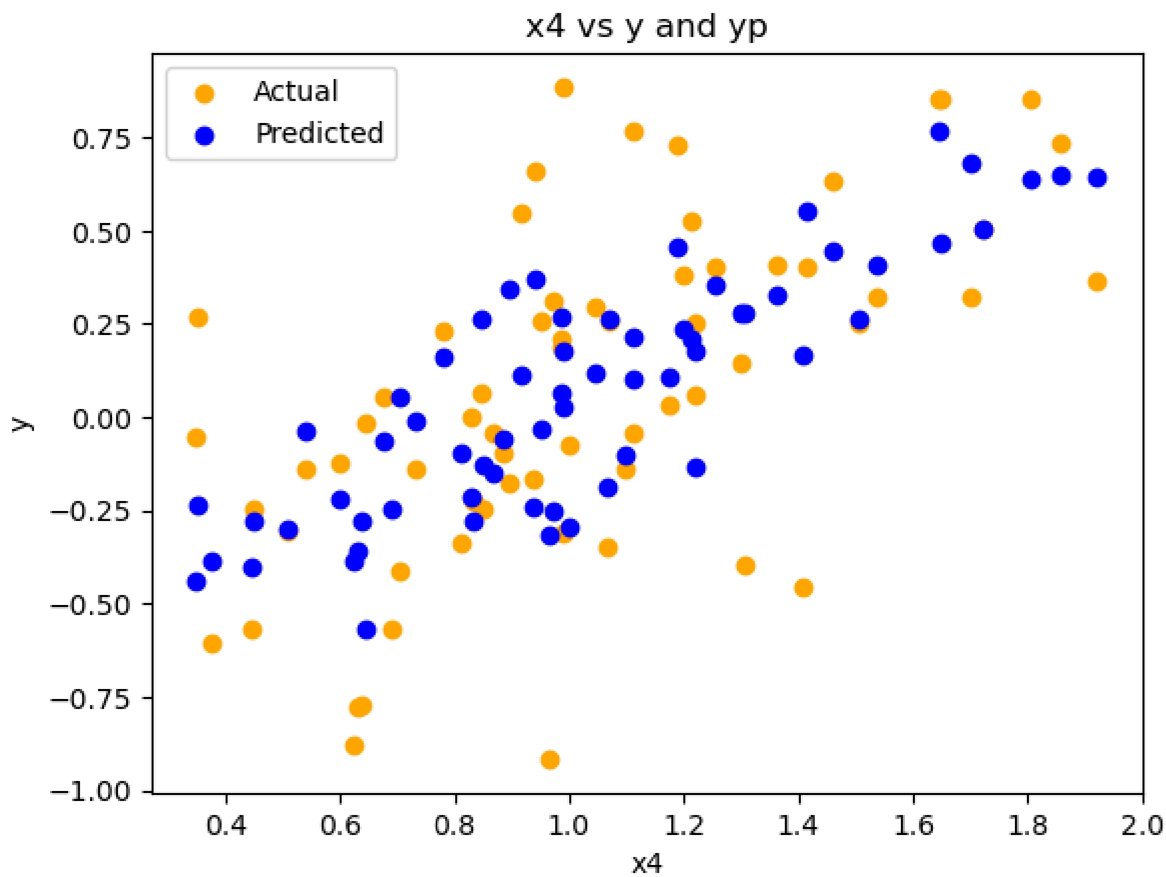
In [59]:
```
# x3 vs. y & yp
plt.scatter(x.T[2], y, c='orange', label = "Actual")
plt.scatter(x.T[2], pred, c='blue', label = "Predicted")
plt.xlabel('x3')
plt.ylabel('y')
plt.title('x3 vs y and yp')
plt.legend()
```

Out[59]: <matplotlib.legend.Legend at 0x13583d43760>



In [60]:
```
# x4 vs. y & yp
plt.scatter(x.T[3], y, c='orange', label = "Actual")
plt.scatter(x.T[3], pred, c='blue', label = "Predicted")
plt.xlabel('x4')
plt.ylabel('y')
plt.title('x4 vs y and yp')
plt.legend()
```

Out[60]: <matplotlib.legend.Legend at 0x13584e1d7f0>



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 [5]:
```
import pandas as pd
import numpy as np
credit = pd.read_csv('Credit.csv')
credit.head()
```

Out[5]:

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

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

In [61]:
```
columns = ['Income', 'Limit', 'Age', 'Education', 'Balance']
X = credit[columns].values

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

Out[61]:
```
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 [62]:
```
y = credit['Rating']
y
```

Out[62]:
```
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 [64]:
```
beta2 = np.linalg.lstsq(X,y,rcond=-1)[0]
beta2

pred2 = np.dot(X, beta2)
pred2
```

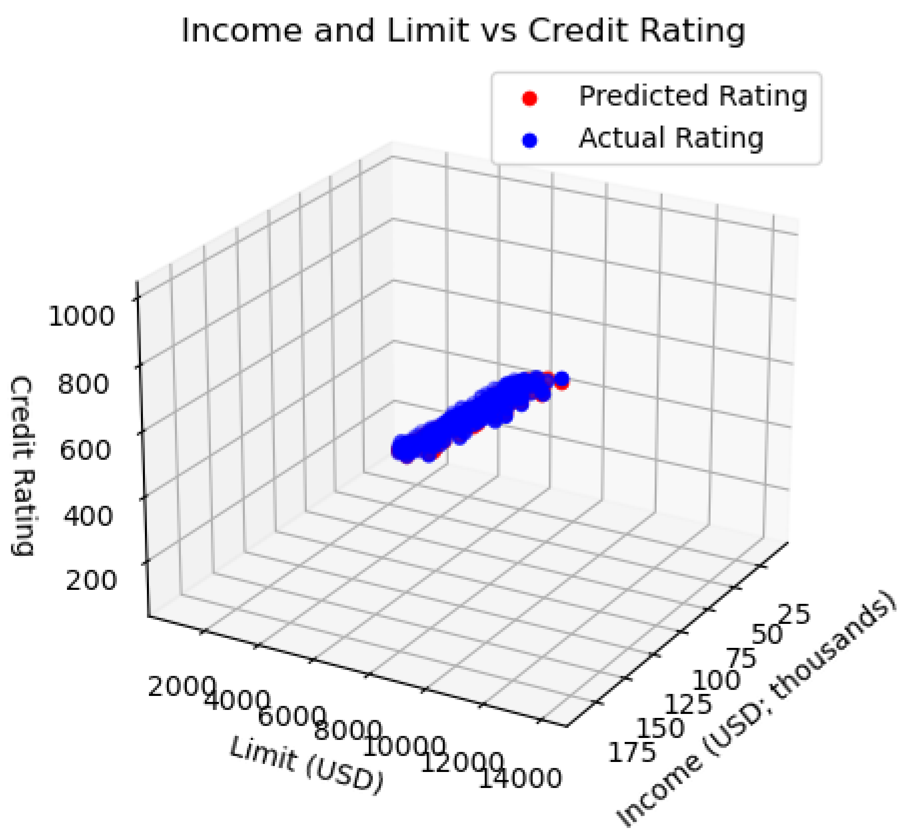
Out[64]:
```
array([277.53859614, 488.6059157 , 511.45574486, 674.01879394,
       363.10792782, 578.26257737, 262.48307715, 515.15061719,
       257.95174726, 498.93842097, 581.35674608, 128.12325543,
       392.91883044, 501.40136224, 256.22703539, 204.88937408,
       283.56484755, 329.00213929, 464.94219261, 482.00022922,
       226.74075886, 462.79450126, 213.3801447 , 385.15320316,
       155.58650495, 325.41450446, 287.21610228, 338.61807515,
       935.65754446, 412.3850226 , 419.53969793, 218.27598228,
       560.86133988, 163.66120083, 211.77294839, 214.50958334,
       469.96353754, 472.82706423, 298.52078734, 267.85466244,
       258.05335006, 556.3774445 , 354.96039997, 454.75771693,
       462.99928873, 544.07184545, 381.10670713, 339.245888  ,
       191.51797648, 350.01008639, 383.05163357, 298.35495832,
       400.00369957, 404.98564654, 141.78797574, 162.84152926,
       353.963675  , 356.00640027, 268.15713469, 390.53612265,
       381.34908562, 243.76752818, 156.24745563, 235.61985531,
       232.98756335, 314.67811364, 687.70338393, 378.33783161,
       412.70700024, 494.2401002 , 301.99473436, 533.77972591,
       364.0425098 , 338.71904935, 399.3362522 , 246.7424244 ,
       262.42530432, 251.94661612, 482.73678383, 178.33173058,
       267.65350505, 320.42659844, 333.98081203, 136.15370749,
       233.60316684, 846.98780529, 459.89593642, 188.98891   ,
       327.01411507, 542.33498822, 422.48766898, 440.27788347,
       226.00410468, 399.39315457, 241.28902011,  98.52439932,
       403.23410252, 261.90665396, 240.83028801, 605.41358598,
       286.19978719, 205.46298371, 551.4989783 , 676.81152963,
       356.91169806, 247.12707658, 130.65549632, 250.48376601,
       440.07340232, 252.56319496, 253.51160319, 234.65082179,
       481.38783648, 464.83303879, 258.13284136, 362.64528069,
       181.16366112, 645.94770484, 181.78123045, 135.92561599,
       135.87209469, 582.71196328, 510.05507231, 128.12049764,
       206.54583428, 205.50456969, 415.04518716, 267.11269007,
       604.16287031, 267.04822449, 300.65119111, 142.01733202,
       402.1973933 , 429.07674927, 427.29098108, 269.69240999,
       313.00873767, 279.32084169, 178.70197933, 736.30621579,
       449.62731646, 489.17080763, 532.40964187, 365.94123201,
       219.0514694 , 349.13625549, 378.41687254, 140.91829297,
       199.44003113, 102.72878565, 419.35784791, 360.37933212,
       185.02613484, 343.52039083, 247.74648259, 133.34490443,
       324.30968701, 412.78559848, 408.94066827, 239.56704371,
       364.68649931, 156.18062031, 541.38297742, 192.24210094,
       437.3505191 , 340.55171003, 228.45219971, 193.41997339,
       224.89272608, 451.55287289, 177.3525171 , 322.88560447,
       348.96287694, 355.19013295, 751.70006993, 183.01880299,
       208.69577211, 300.73545231, 330.44465405, 541.67073087,
       276.37978367, 384.13576057, 468.12358647, 309.95491751,
       811.9467139 , 332.52235957, 290.60571701, 184.25263208,
       551.467118  , 330.46261535, 393.70015932, 683.68046306,
       299.5313296 , 712.44035062, 181.96488984, 396.6641975 ,
       532.23234209, 301.04708094, 172.44853993, 316.83731321,
       392.75112634, 529.13540836, 138.32843012, 498.29716688,
       393.39796831, 297.79159076, 202.01291921, 339.96146777,
       324.93024898, 650.91282681, 250.73799006, 391.45612883,
       327.34463923, 385.53640511, 389.98212595, 317.37196466,
       218.96514582, 398.59142383, 153.8221702 , 387.38911569,
       430.39128113, 626.19509299, 462.2271187 , 345.03914277,
       563.09368168, 416.6059702 , 500.67222311, 411.74515837,
```

```
348.49334741, 545.01953785, 382.25767391, 356.52204572,
356.15509568, 190.14461289, 589.34828351, 229.35155438,
370.02365216, 380.56553832, 229.27844864, 272.9630585 ,
260.90195904, 103.660084  , 123.01586753, 480.00788809,
159.31827567, 173.71190357, 251.18535578, 186.3416409 ,
101.95369736, 145.97451048, 196.29864934, 251.69786566,
615.46454062, 384.25347109, 468.41793441, 320.02969053,
159.45806846, 202.74008846, 209.61762593, 454.99989848,
384.09163875, 669.3784767 , 307.05081779, 271.29238423,
379.06457329, 372.00936377, 366.69881705, 426.55981898,
133.91259966, 408.03764819, 241.54597271, 362.14988466,
289.15996409, 360.02756658, 433.07556346, 620.43124315,
267.74957918, 371.48350065, 503.95812528, 249.63876057,
392.67974767, 164.79461254, 578.31586321, 464.49793375,
177.6907973 , 149.3315725 , 143.42577437, 248.88784352,
388.74671259, 298.70753406, 254.63045418, 282.10933817,
376.40138049, 790.90173131, 208.5429343 , 136.78667698,
378.09978951, 328.38596077, 213.571755  , 376.40154523,
344.86144182, 273.77268991, 367.59818594, 369.45627514,
539.54907886, 167.96730917, 289.84192446, 296.45105549,
349.23013369, 504.51641751, 369.96567335, 399.87369339,
389.07490439, 549.72550501, 657.79474868, 296.86285292,
526.0187481 , 350.33267019, 141.15055934, 209.25535193,
120.40166493, 241.53682734, 271.05541472, 970.07592276,
233.3825037 , 376.41876063, 722.15383295, 483.27720883,
280.91994027, 542.86703704, 347.00924805, 301.26046591,
386.25240111, 262.32890461, 354.30323718, 265.36724663,
431.89468071,  99.57186288, 390.71007582, 724.39126405,
292.69308255, 297.8394378 , 235.5862777 , 285.17604799,
386.69887609, 141.26284221, 396.29077822, 755.83466136,
117.71723485, 381.40433269, 146.97580072, 381.14690706,
513.5430558 , 349.9793242 , 293.00154914, 840.74233793,
444.16069764, 208.13446261, 324.56838054, 336.67528167,
433.31514248, 365.69123596, 376.46523035, 447.39094989,
694.88669246, 474.78726172, 564.37208952, 277.64274244,
424.08393101, 574.09438129, 448.40109849, 184.51768839,
295.55101401, 400.82375895, 359.50904833, 416.42427489,
517.33751158, 147.52140629, 364.71244985, 232.84761538,
556.30404975, 575.29581016, 413.18209724, 259.0896197 ,
164.16756291, 414.43658074, 284.60414017, 133.54698313,
495.04706909, 510.42571602, 746.00492328, 474.62564406,
192.73369413, 130.99238209, 424.1473159 , 310.77405476,
293.02655498, 316.35268666, 207.48495616, 410.03810268])
```

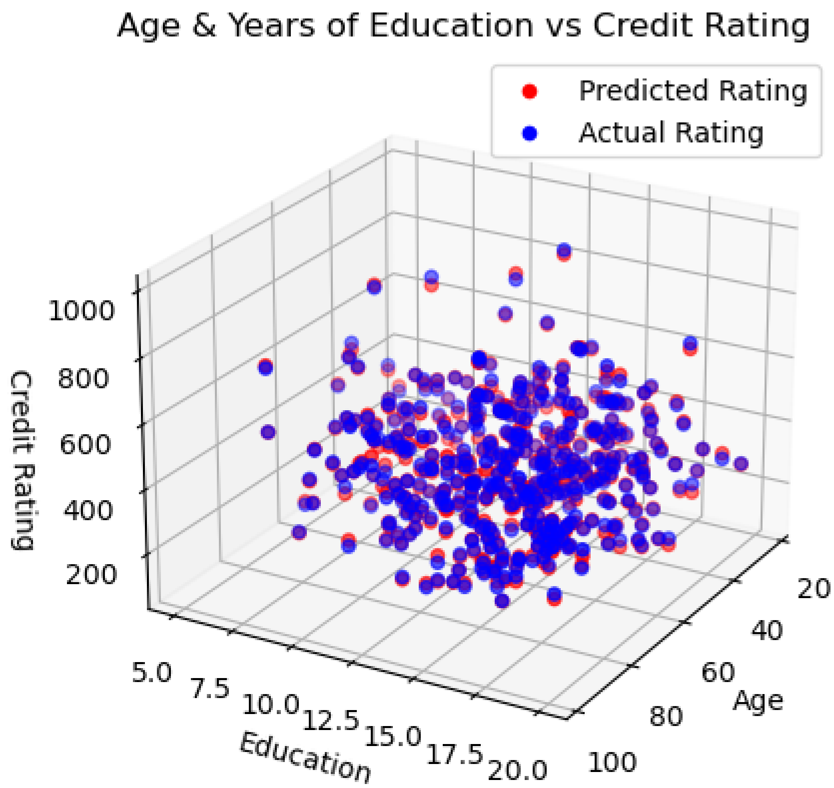
5. Plot your results using scatter plots (just like in class). Show as many of your columns vs. credit rating that you can.

In [71]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.view_init(23, 30)
ax.scatter(X.T[0], X.T[1], pred2, zdir='z', c='r', label='Predicted Rating')
ax.scatter(X.T[0], X.T[1], y, zdir='z', c='b', label='Actual Rating')
ax.set_xlabel('Income (USD; thousands)')
ax.set_ylabel('Limit (USD)')
ax.set_zlabel('Credit Rating')
plt.legend()
plt.title('Income and Limit vs Credit Rating')
plt.show()
```



In [67]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.view_init(23, 30)
ax.scatter(X.T[2], X.T[3], pred2, zdir='z', c='r', label='Predicted Rating')
ax.scatter(X.T[2], X.T[3], y, zdir='z', c='b', label='Actual Rating')
ax.set_xlabel('Age')
ax.set_ylabel('Education')
ax.set_zlabel('Credit Rating')
plt.legend()
plt.title('Age & Years of Education vs Credit Rating')
plt.show()
```



In [72]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.view_init(23, 30)
ax.scatter(X.T[3], X.T[4], pred2, zdir='z', c='r', label='Predicted Rating')
ax.scatter(X.T[3], X.T[4], y, zdir='z', c='b', label='Actual Rating')
ax.set_xlabel('Years of Education')
ax.set_ylabel('Balance (USD)')
ax.set_zlabel('Credit Rating')
plt.legend()
plt.title('Education & Balance vs Credit Rating')
plt.show()
```