Assigment 3

This assigment focuses on getting comfortable with working with multidimensional data and linear regression. Key items include:

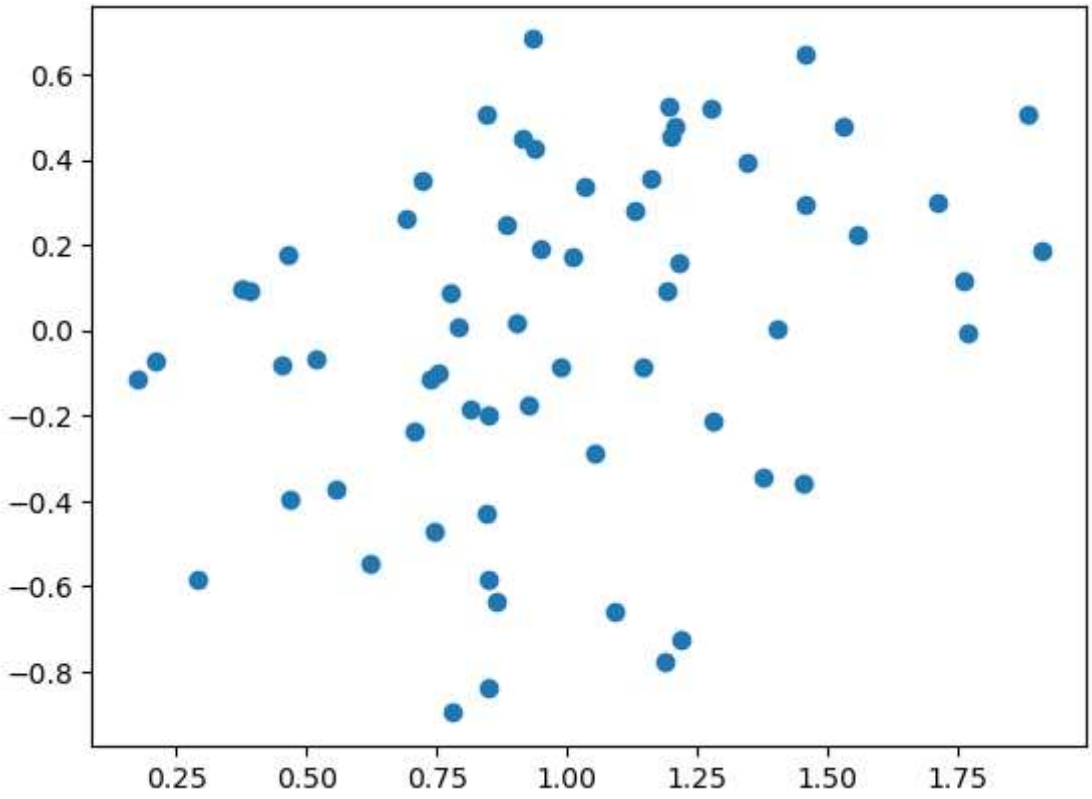
- Creating random n-dimensional data
- Creating a Model that can handle the data
- Plot a subset of the data along with the prediction
- Using a Dataset to read in and choose certain columns to produce a model
- Create several models from various combinations of columns
- Plot a few of the results

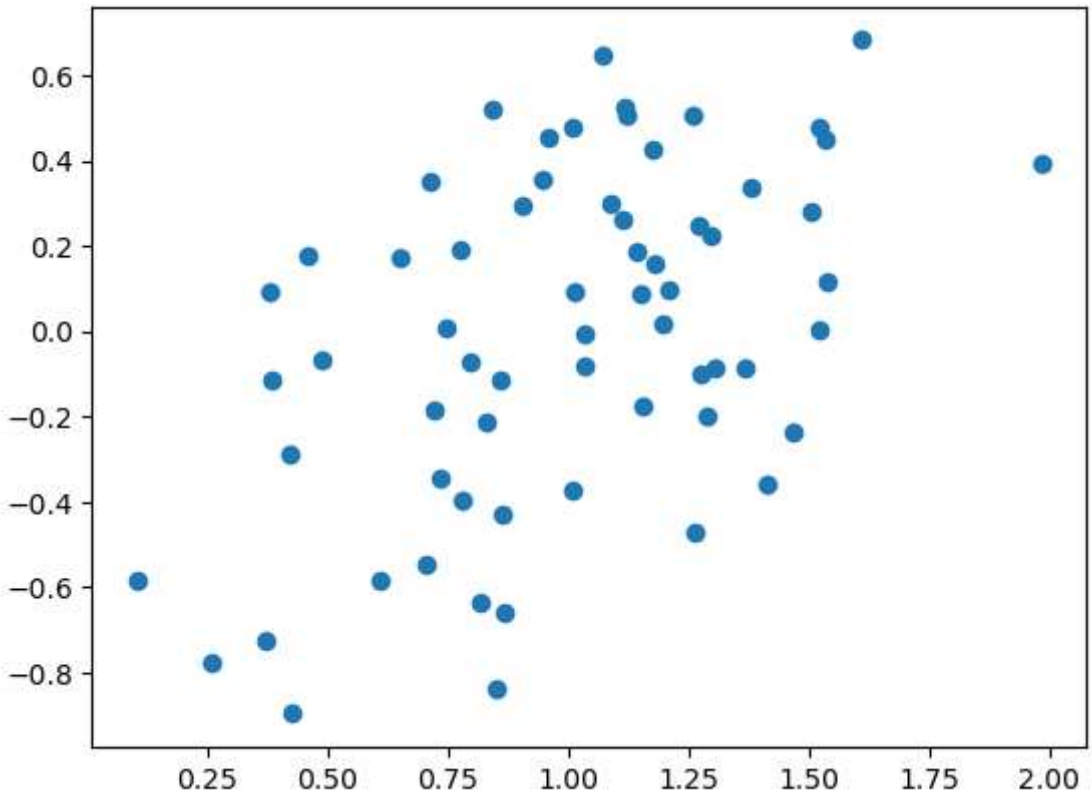
1. Create a 4 dimensional data set with 64 elements and show all 4 scatter 2D plots of the data x_1 vs. y, x_2 vs. y, x_3 vs. y, x_4 vs. y

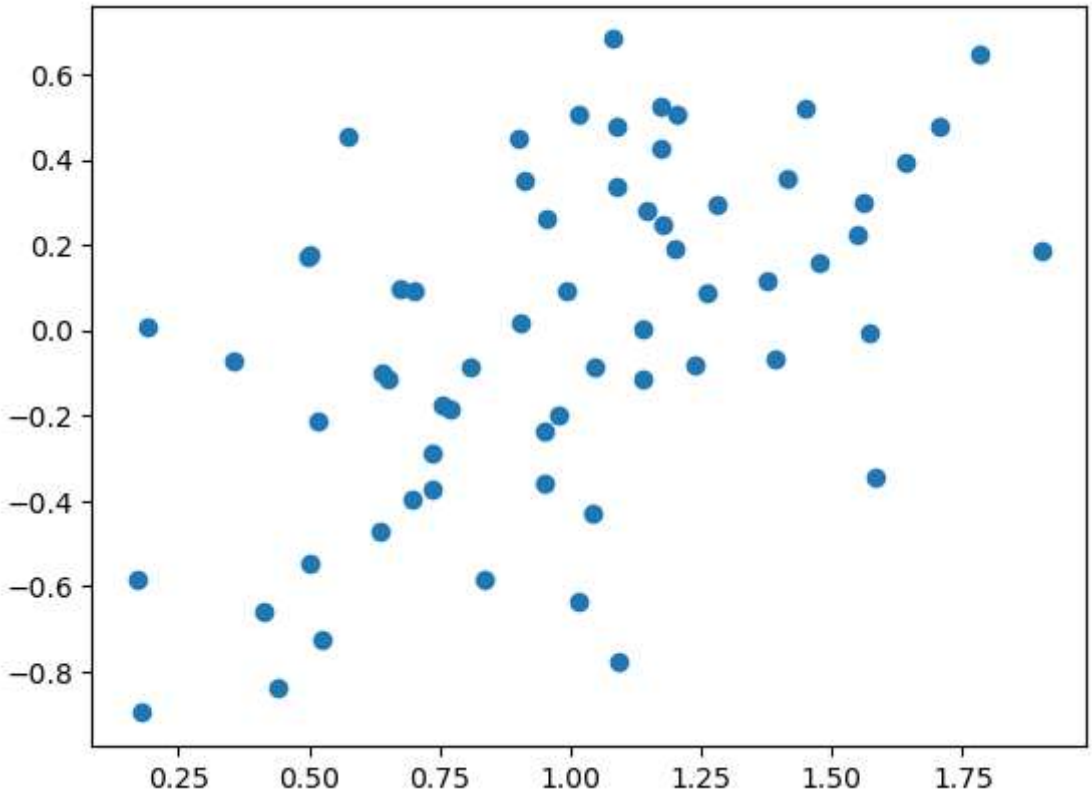
In [ ]:
```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

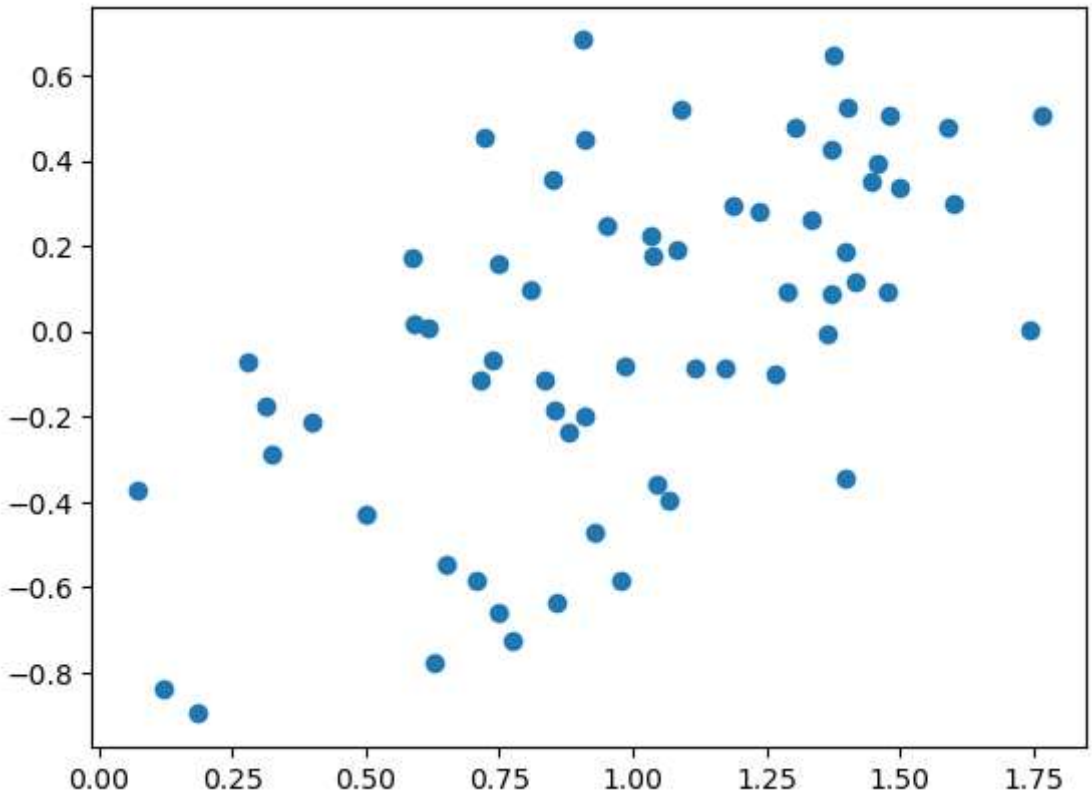
In [ ]:
```
# The data set will have 64 rows and 4 columns
n = 64
# Generate a 4 dimensional matrix with random data
x = np.linspace(0, 1, n) + np.random.rand(4, n)
# Generate a 1 dimensional vector with random data for our y value
# We are loading it with values that will force a trend
y = np.linspace(0, 1, n) + np.random.rand(n) - 1
```

In [ ]:
```
# Plot all 4 x values against the y value as separate scatter plots

plt.scatter(x[0], y)
plt.show()
plt.scatter(x[1], y)
plt.show()
plt.scatter(x[2], y)
plt.show()
plt.scatter(x[3], y)
plt.show()
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.25
0.50
0.75
1.00
1.25
1.50
1.75


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00





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 [ ]:
```
# add a column of 1s to our data set to act as a placehodler for the error value
x = np.vstack([x, np.ones(len(x.T))]).T
# This also transposes the data to make it look like a data frame
```

Making use of the following equation: $\beta = (X^T X)^{-1} Y^T X$

In [ ]:
```
# Calculate the left side first
left = np.linalg.inv(np.dot(x.T, x))
# Calculate the right side
right = np.dot(y.T, x)
# Multiply together with dot product
beta = np.dot(left, right)

# Print the results
beta
```

Out[ ]:
```
array([-0.0655046 ,  0.26822135,  0.23722764,  0.35117683, -0.79327848])
```

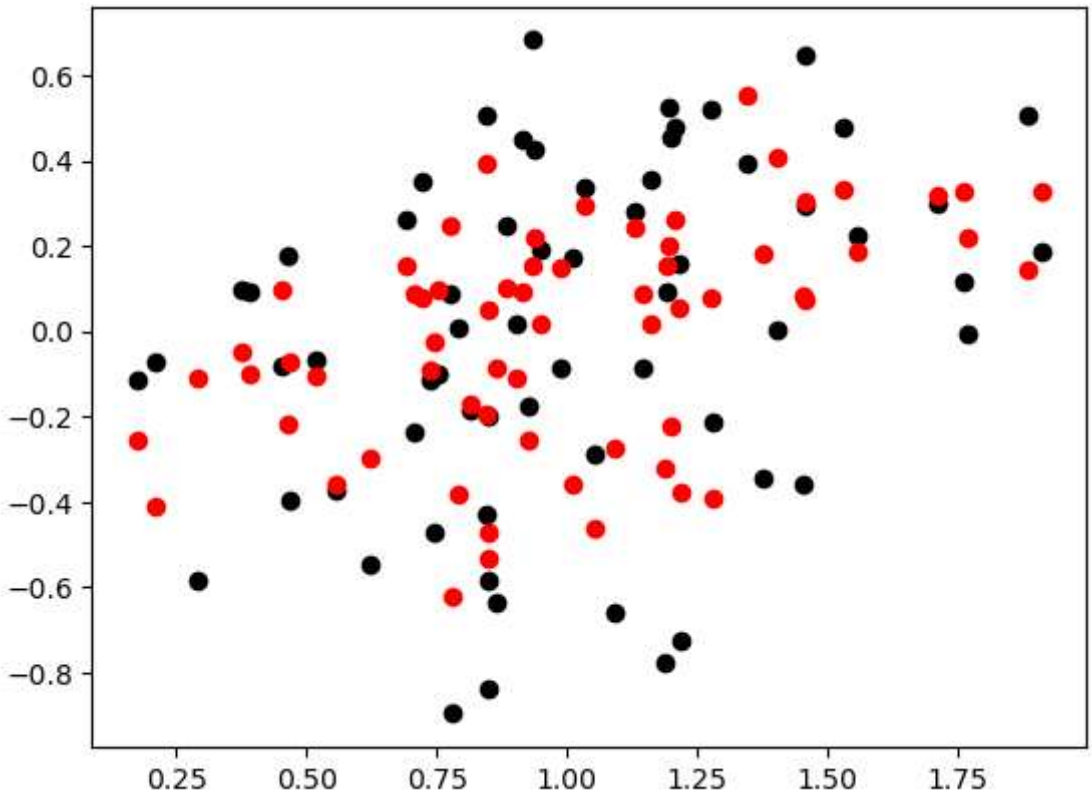
In [ ]:
```
# Predict new values using the beta values
yhat = np.dot(x, beta)
yhat
```

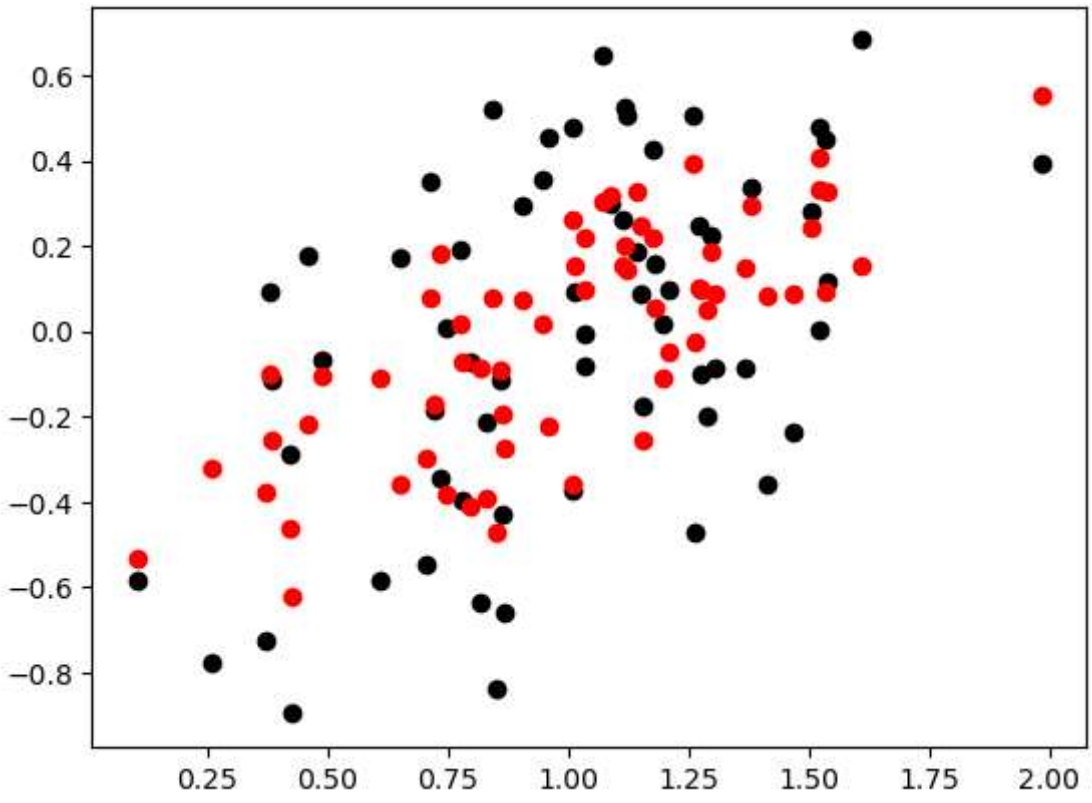
Out[ ]:
```
array([-0.47414291, -0.53182908, -0.35883527, -0.25381493, -0.19414755,
       -0.46047407, -0.62225398, -0.38266049, -0.07307276, -0.17080124,
       -0.08817313, -0.41018249, -0.25423335, -0.32147202, -0.27219967,
       -0.10790331, -0.08989423, -0.37716523,  0.09468636, -0.11035152,
        0.10286826, -0.09901914, -0.049961  , -0.39223538, -0.29686779,
       -0.0260726 ,  0.09576225,  0.08533554, -0.21788794,  0.07522543,
       -0.1072262 , -0.36133184,  0.1543365 ,  0.08195653, -0.22410835,
        0.2213973 ,  0.15502007,  0.07579787,  0.08881235,  0.01744856,
        0.18106548,  0.29391621,  0.33140248,  0.07601508,  0.04827558,
        0.24523425,  0.0565441 ,  0.1988854 ,  0.24212598,  0.394871  ,
        0.15178307,  0.01828068,  0.09180915,  0.26051724,  0.40552522,
        0.32825129,  0.1491682 ,  0.18417436,  0.22008381,  0.32925522,
        0.1450128 ,  0.3038639 ,  0.3178886 ,  0.55166974])
```

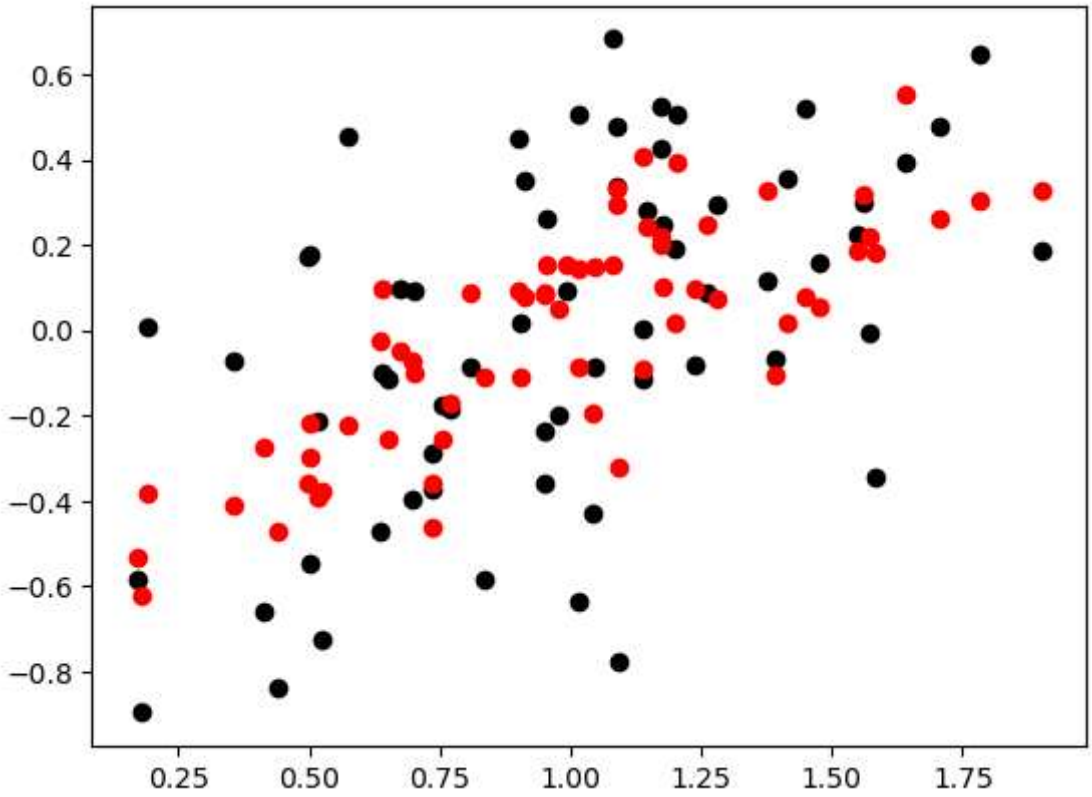
3. Plot the model's prediction as a different color on top of the scatter plot from Q1 in 2D for all 4 of the dimensions ($x_1 \rightarrow y_p, x_2 \rightarrow y_p, x_3 \rightarrow y_p, x_4 \rightarrow y_p$)

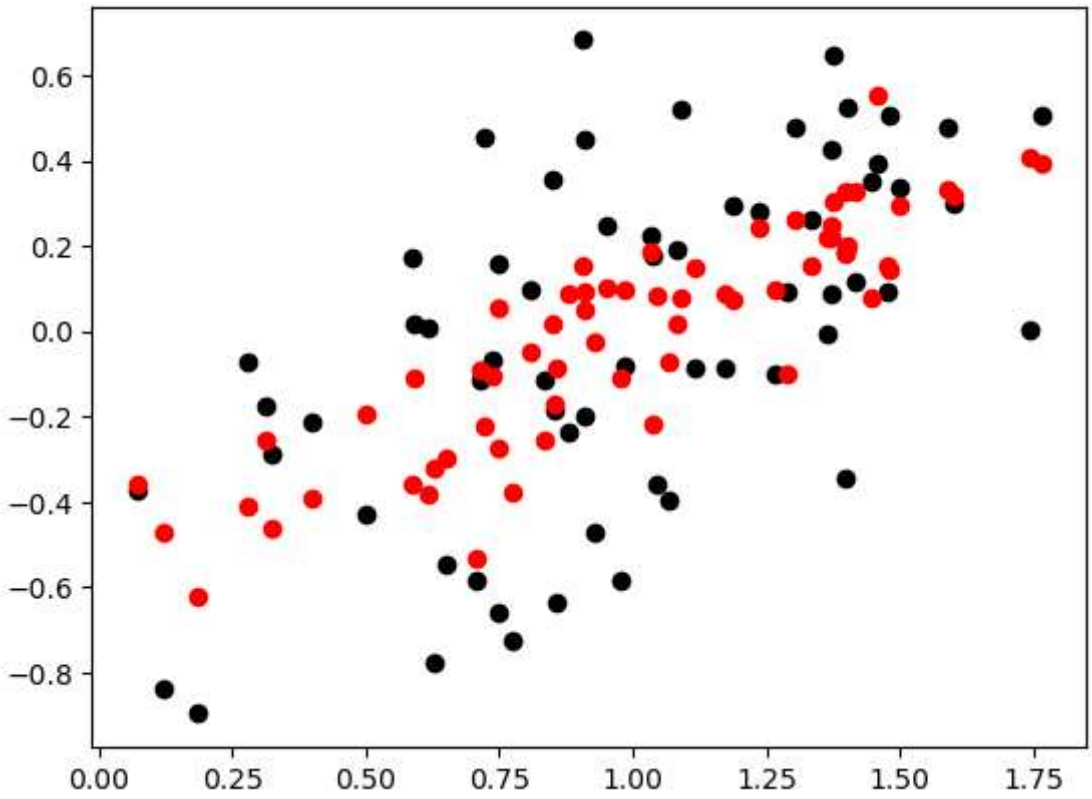
In [ ]:
```
# Plot each set of data in turn against the predicted values
plt.scatter(x.T[0], y, color='black')
plt.scatter(x.T[0], yhat, color='red')
plt.show()

plt.scatter(x.T[1], y, color='black')
plt.scatter(x.T[1], yhat, color='red')
plt.show()
```

```
plt.scatter(x.T[2], y, color='black')
plt.scatter(x.T[2], yhat, color='red')
plt.show()

plt.scatter(x.T[3], y, color='black')
plt.scatter(x.T[3], yhat, color='red')
plt.show()
```









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

Out[ ]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	E
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	

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

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

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

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

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

Out[ ]:
```
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 [ ]:
```
# add a column of 1s to our data set to act as a placehodler for the error value
#x = np.vstack([x, np.ones(len(x.T))]).T
# This also transposes the data to make it look like a data frame

# Calculate the left side first
left = np.linalg.inv(np.dot(X.T, X))
# Calculate the right side
right = np.dot(y.T, X)
# Multiply together with dot product
beta = np.dot(left, right)

# Print the results
beta
```

Out[ ]:
```
array([1.31880513e-01, 6.26253800e-02, 3.19381468e-02, 1.50244935e-02,
       4.28168820e+01])
```

In [ ]:
```
# Predict new values using the beta values
pred_score = np.dot(X, beta)
pred_score
```

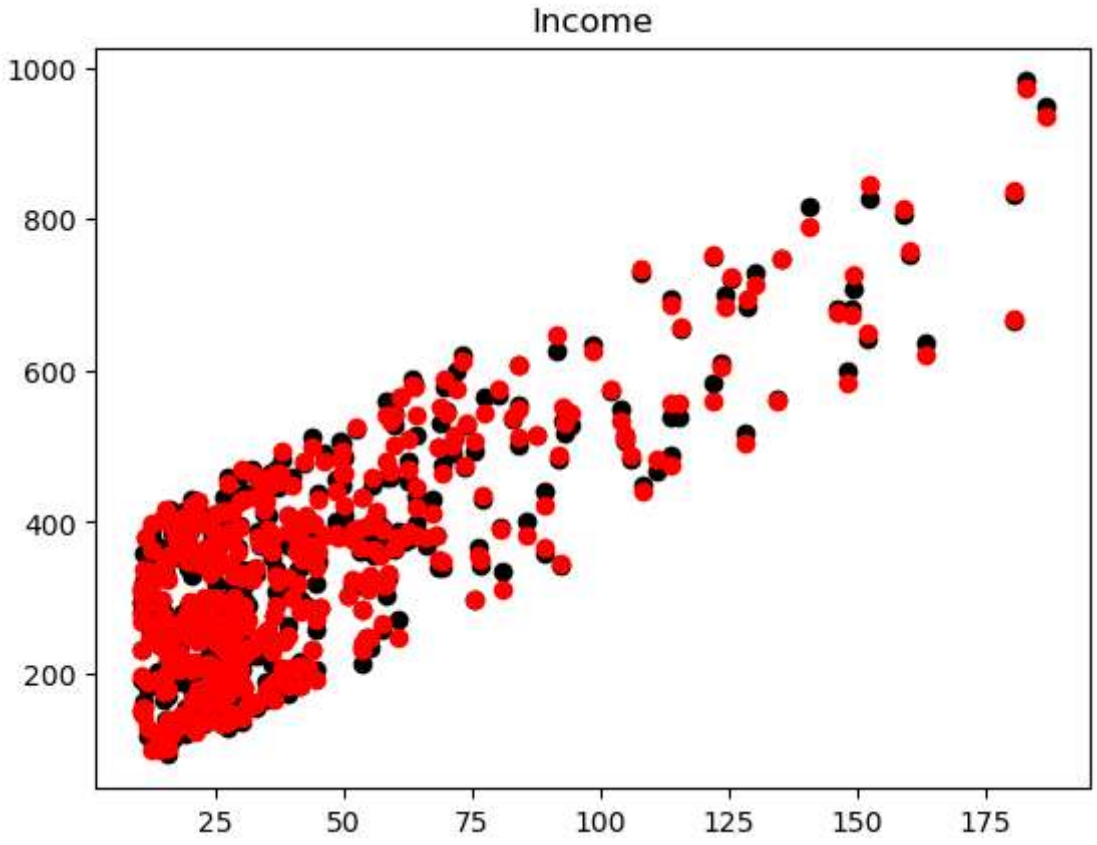
Out[ ]:
```
array([276.6968883 , 489.13120906, 510.66703854, 673.28205191,
       364.00801595, 577.08992362, 261.99231627, 513.63113601,
       257.77508011, 500.82343953, 581.56569911, 128.94693888,
       390.74954943, 499.87701353, 256.06071413, 205.41578846,
       284.80755971, 329.53665253, 463.4357167 , 480.47299757,
       227.60903702, 464.01028526, 212.18367766, 383.30573643,
       156.08680948, 326.28448495, 286.66627894, 339.50682946,
       935.97588956, 413.24801096, 416.62218029, 219.16213974,
       560.80638405, 162.45354245, 211.97948346, 213.95224597,
       469.13629151, 471.27481181, 297.32522896, 268.03261418,
       258.25662975, 556.91528222, 354.81749426, 454.96452125,
       463.88141086, 543.60962459, 381.90186944, 339.73736863,
       192.02036849, 349.80333833, 383.59065365, 300.64582903,
       399.53324085, 403.76374689, 140.21554341, 160.92307843,
       355.53420782, 356.23673249, 269.35752182, 391.76154557,
       383.62497474, 242.57455508, 153.96948716, 236.82512527,
       233.51254144, 314.86004213, 687.61304287, 379.17954874,
       413.22267792, 494.43084634, 301.47911376, 533.28101934,
       365.60916688, 339.25496047, 398.12671503, 247.64986768,
       263.2455279 , 252.49761487, 481.97347378, 178.4936783 ,
       266.13274916, 320.98013775, 333.12649048, 136.30810397,
       232.7520459 , 846.57789325, 460.46254054, 188.47187972,
       325.11611051, 540.81492369, 423.35170687, 441.17109988,
       225.85388742, 400.26742346, 241.83978848,  99.68814179,
       404.37230273, 263.12161809, 240.96846517, 607.0133764 ,
       285.36173322, 206.2663426 , 552.33682618, 677.42533195,
       356.39044185, 248.68505173, 131.12853626, 249.98661221,
       440.33164993, 254.47066379, 254.74588278, 233.81015585,
       481.58597813, 464.37237142, 258.66365222, 360.75520987,
       182.01413734, 647.21977595, 182.99377122, 136.74049204,
       136.35007763, 582.65295911, 509.22249353, 128.24635839,
       206.05352788, 206.04461653, 415.49961204, 266.2754288 ,
       605.38613252, 267.57607612, 300.07608858, 143.89718296,
       402.02470655, 428.53970451, 427.81216972, 270.57801984,
       311.15443779, 277.80431137, 177.82153166, 734.11368909,
       449.07216071, 488.04912985, 533.68235259, 364.73177997,
       219.24877835, 349.31534755, 378.93311125, 141.75238698,
       198.57650741, 101.48124875, 419.58981048, 360.22085267,
       184.15538305, 344.81493299, 248.65085192, 131.77509612,
       324.1401296 , 413.615205  , 408.43566427, 239.4174808 ,
       363.80364179, 156.31499449, 540.90629314, 193.09266701,
       436.89352863, 339.70261633, 229.00275889, 193.60553538,
       224.06523195, 451.04509104, 175.78899852, 321.36216433,
       350.57603929, 354.66776107, 752.64328123, 184.22331306,
       210.27499466, 300.93317016, 329.27155596, 540.82225949,
       277.22788603, 383.63593429, 466.60565134, 310.54488254,
       813.27240518, 332.7297916 , 290.46441272, 184.75433419,
       550.65000225, 331.36062159, 394.94609041, 684.91404781,
       299.7189172 , 714.42426048, 182.11576004, 397.52275867,
       531.78267565, 302.28372271, 173.30328761, 314.60011647,
       394.31552469, 529.68213472, 138.12534697, 499.13958832,
       391.20766835, 297.6133251 , 201.83283645, 338.68626961,
       325.4930944 , 650.15801004, 250.23808566, 391.99735948,
       327.1978347 , 387.46187445, 390.85817877, 319.28915101,
       220.52672682, 398.74788157, 152.55976172, 387.22218322,
       430.1798837 , 625.37872189, 460.62052322, 346.24463792,
       560.60222536, 417.12488758, 501.20108219, 411.9732077 ,
       347.70338943, 543.49895982, 380.71927923, 356.02636134,
       356.34059631, 189.27531563, 589.56379953, 230.92414067,
       369.20446172, 381.13365335, 231.18947771, 272.8285103 ,
```

```
261.08327444, 102.76284932, 124.53413689, 479.52935408,
159.81170998, 175.23638372, 251.3644775 , 188.23783588,
102.77926649, 145.77165774, 197.86432603, 251.18966528,
613.60528016, 383.79912229, 465.82603518, 318.18578672,
159.6023675 , 202.91940366, 209.81010666, 453.44578383,
383.62197176, 667.56321871, 306.87770363, 271.83941516,
377.87476344, 371.86744834, 367.62475767, 427.09867727,
133.34475841, 409.60965511, 241.39751433, 360.99252631,
289.65170115, 360.12982112, 431.50986195, 620.74647671,
268.28696046, 370.9959568 , 503.20056474, 247.72797789,
392.20991555, 164.61947008, 579.56005633, 465.72037101,
175.0909849 , 148.45659157, 143.92309972, 247.00608644,
389.61453139, 297.17043271, 253.76774531, 281.94674217,
376.96874894, 789.43826373, 208.3853985 , 135.54003599,
379.3688792 , 329.62724912, 215.12949535, 375.53372088,
346.08190455, 271.56431017, 367.48435669, 368.61906173,
539.43150191, 168.13081778, 287.97077654, 296.68165003,
346.65668904, 506.12830122, 370.78735645, 400.39392385,
388.20657989, 551.3378493 , 658.04589682, 297.73342038,
526.21774744, 351.24741571, 140.93376692, 210.12935793,
120.17550358, 242.09167189, 270.20501664, 971.38595702,
234.54854784, 375.90151356, 723.09014973, 483.46709638,
281.81494842, 543.8049819 , 348.1503679 , 302.12922255,
386.8023909 , 263.54561798, 354.8272446 , 263.16267429,
431.72215897,  99.02076728, 390.91656917, 725.35957422,
290.82286366, 299.03860057, 234.04638407, 286.39261355,
385.52133925, 142.45578796, 397.83964439, 757.15874094,
117.84128412, 381.25575037, 146.76744958, 379.6428178 ,
513.81011046, 351.56624998, 294.56690702, 838.98243963,
444.35602227, 208.32390289, 326.1320729 , 336.1447812 ,
432.12367448, 366.60093181, 378.3667588 , 446.22496233,
695.83284997, 475.32697158, 564.57455495, 277.82642458,
421.57897268, 574.996954  , 447.90167674, 185.01544812,
296.36465874, 401.67220787, 361.06270957, 417.28649654,
516.20195158, 146.29134308, 366.62716127, 231.63855947,
556.55436737, 574.49851088, 411.58422098, 257.5206103 ,
165.00285748, 414.95524983, 283.39981348, 131.95954075,
494.87342618, 512.04320543, 746.60369199, 475.20265032,
191.54823898, 130.78138679, 422.28618234, 310.61190371,
294.22327801, 315.8727669 , 207.32683292, 407.78348067])
```

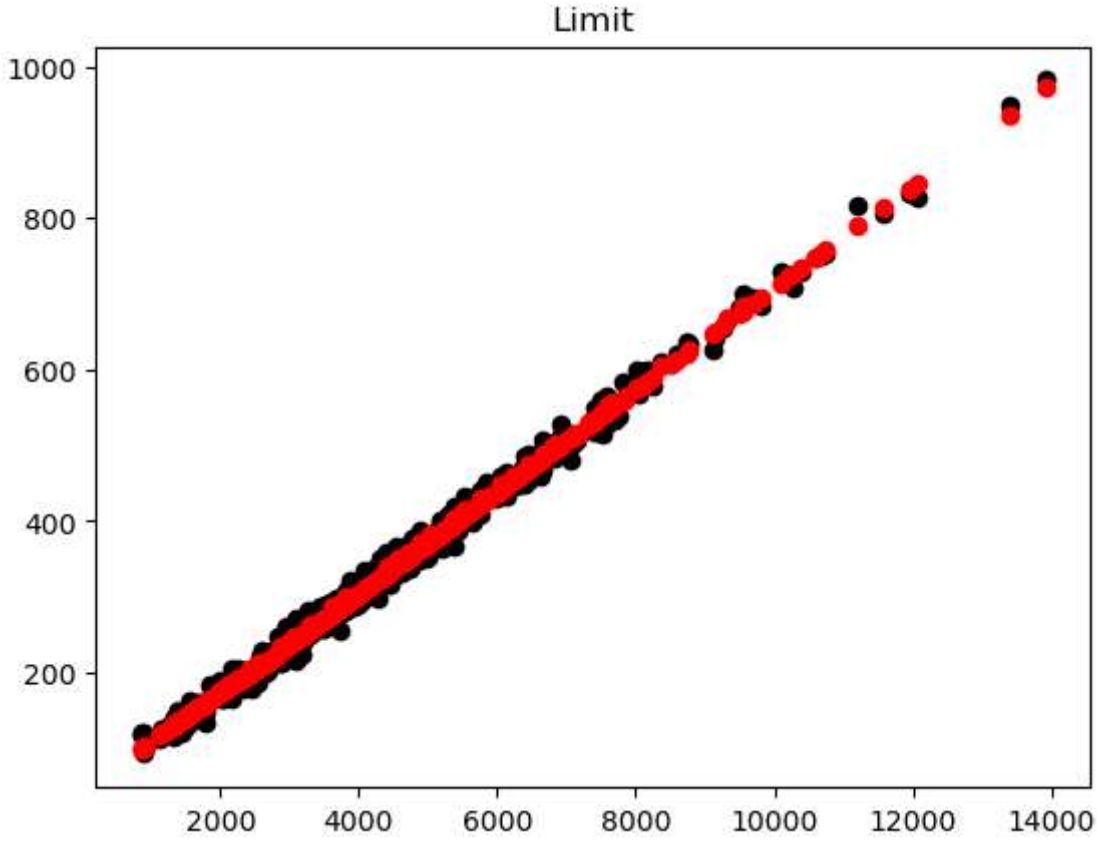
5. Plot your results using scatter plots (just like in class). Show as many of your columns vs. credit rating that you can.

In [ ]:
```python
# Plot each set of data in turn against the predicted values
for i in range(len(X.T)-1):
    print(i)
    plt.scatter(X.T[i], y, color='black')
    plt.scatter(X.T[i], pred_score, color='red')
    plt.title(columns[i])
    plt.show()
```

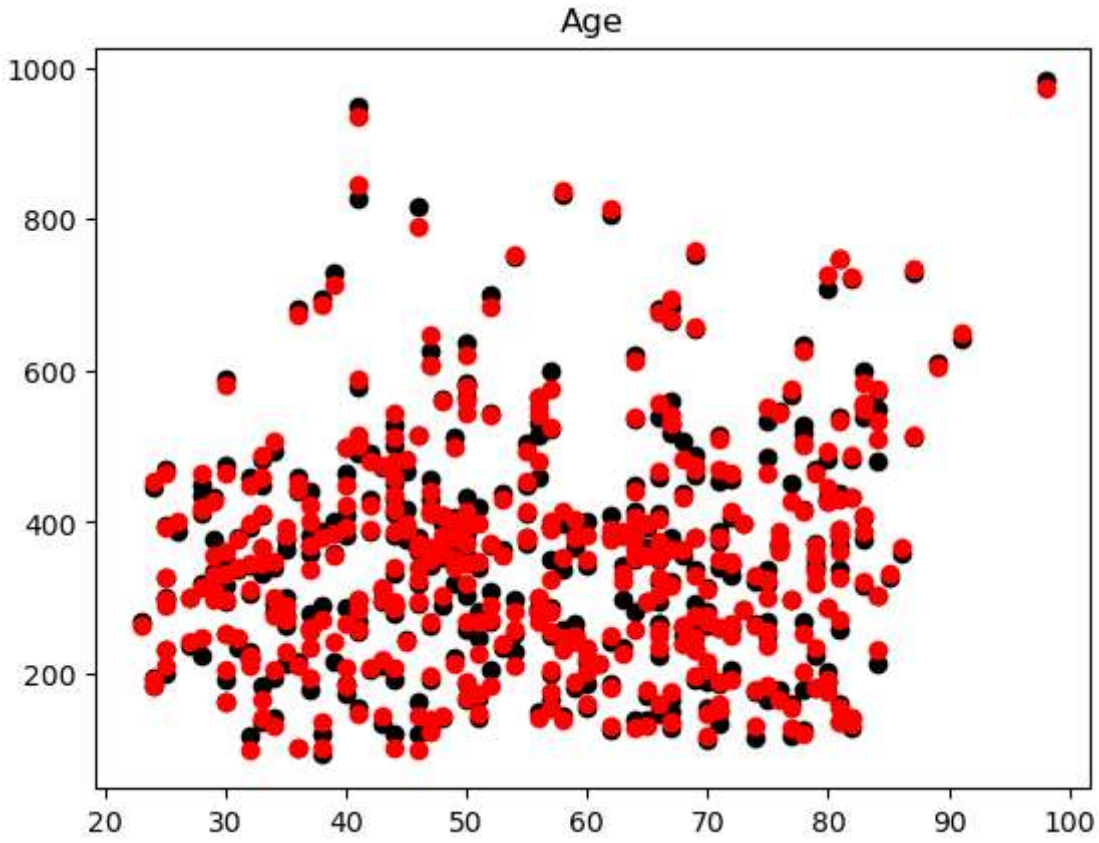
```
0
```


Income
1000
800
600
400
200
25
50
75
100
125
150
175

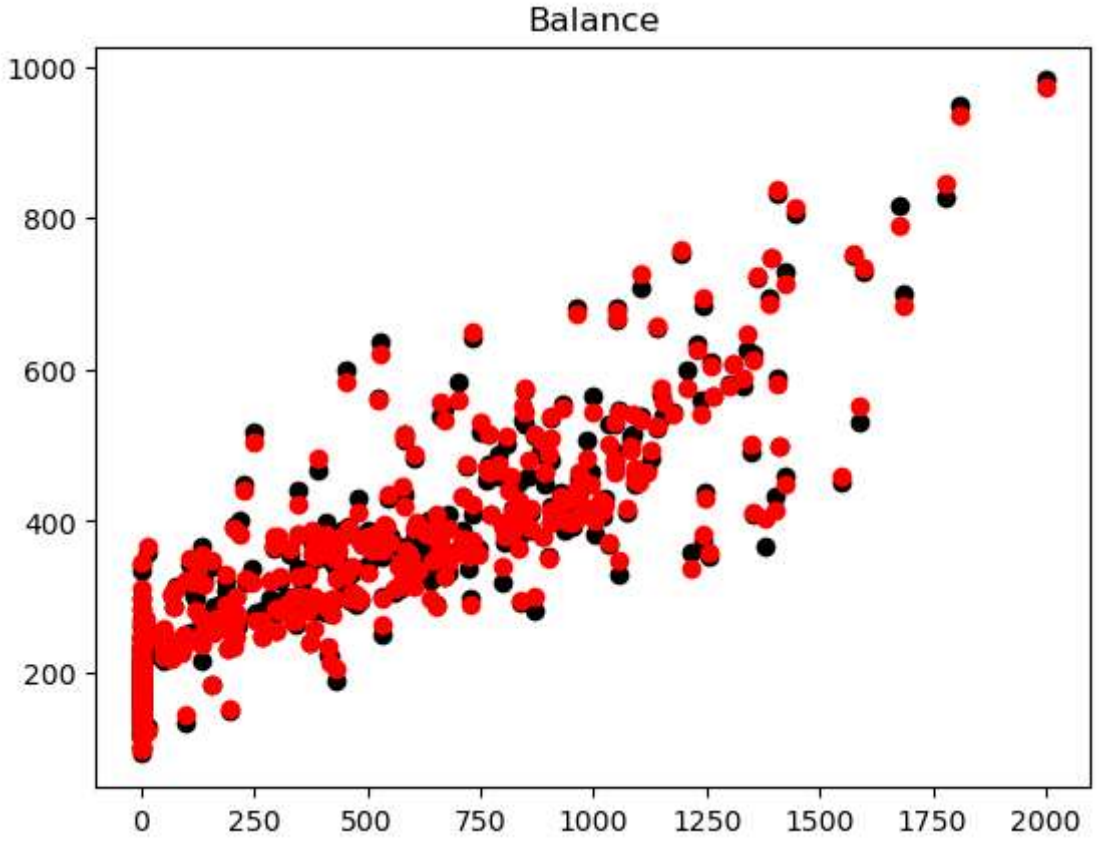
1


Limit
1000
800
600
400
200
2000
4000
6000
8000
10000
12000
14000

2


Age
1000
800
600
400
200
20
30
40
50
60
70
80
90
100

3


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