Homework 3 Pandas

September 24, 2018

1 Imports

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd
        import numpy as np
        import numpy.linalg as la
        import scipy.io as sio
        from math import floor
```

2 Load the data

```
In [2]: file = 'hwkdataNEW.mat'
        data_dict = sio.loadmat(file)
        print(data_dict.keys())
```

```
dict_keys(['__header__', '__version__', '__globals__', 'x', 'y'])
```

Extract the parts we want and examine them

```
In [3]: x = data_dict['x']
        y = data_dict['y']

In [4]: x.shape

Out[4]: (247, 14)

In [5]: y.shape

Out[5]: (247, 1)
```

We want to put this all into the pandas data format

```
In [6]: column_names = ['Age', 'Weight', 'Height', 'Adiposity Index',
                        'Neck Circumference', 'Chest Circumference',
                        'Abdomen Circumference', 'Hip Circumference',
```

```
                      'Thigh Circumference', 'Knee Circumference',
                      'Ankle Circumference', 'Extended Bicep Circumference',
                      'Forearm Circumference', 'Wrist Circumference',
                      'Body Fat Percentage']
```

```
In [7]: data = pd.DataFrame(x, columns=column_names[:-1]) # Last column contains y
```

```
In [8]: data[column_names[-1]] = y
```

```
In [9]: data.head()
```

Out[9]:

	Age	Weight	Height	Adiposity Index	Neck Circumference
0	23.0	154.25	67.75	23.7	36.2
1	22.0	173.25	72.25	23.4	38.5
2	22.0	154.00	66.25	24.7	34.0
3	26.0	184.75	72.25	24.9	37.4
4	24.0	184.25	71.25	25.6	34.4

	Chest Circumference	Abdomen Circumference	Hip Circumference
0	93.1	85.2	94.5
1	93.6	83.0	98.7
2	95.8	87.9	99.2
3	101.8	86.4	101.2
4	97.3	100.0	101.9

	Thigh Circumference	Knee Circumference	Ankle Circumference
0	59.0	37.3	21.9
1	58.7	37.3	23.4
2	59.6	38.9	24.0
3	60.1	37.3	22.8
4	63.2	42.2	24.0

	Extended Bicep Circumference	Forearm Circumference	Wrist Circumference
0	32.0	27.4	17.1
1	30.5	28.9	18.2
2	28.8	25.2	16.6
3	32.4	29.4	18.2
4	32.2	27.7	17.7

	Body Fat Percentage
0	12.6
1	6.9
2	24.6
3	10.9
4	27.8

Data is now a pandas DataFrame that contains the information we need

3 Sort into Train, Test, and Validation

Since the rows of data should be indpendent and uncorrelated, I have chosen to simply shuffle them then tag them as belonging to one of the named data sets (`train`, `test`, or `validation`).

```
In [10]: train_frac = 0.6
         test_frac = 0.2
         val_frac = 0.2

         train_len = floor(len(data) * train_frac)
         test_len = floor(len(data) * test_frac)
         val_len = floor(len(data) * val_frac)
         rows_used = train_len + test_len + val_len
         # Make sure all rows being used by adding unused
         # rows into the training set
         while rows_used < len(data):
             train_len += 1
             rows_used = train_len + test_len + val_len
         print(f'Rows in original dataset: {len(data)}')
         print(f'Rows in training dataset: {train_len}')
         print(f'Rows in testing dataset: {test_len}')
         print(f'Rows in validation dataset: {val_len}')
         print(f'Rows used: {train_len + test_len + val_len}')
```

```
Rows in original dataset: 247
Rows in training dataset: 149
Rows in testing dataset: 49
Rows in validation dataset: 49
Rows used: 247
```

```
In [11]: # Shuffle and reindex the dataset
         data = data.sample(frac=1).reset_index(drop=True)
         # Split the data into three datasets
         dataset_labels = np.zeros(len(data)).astype('str')
         dataset_labels[:train_len] = 'train'
         dataset_labels[train_len:train_len + test_len] = 'test'
         dataset_labels[-val_len:] = 'val'
         data['sample'] = dataset_labels
```

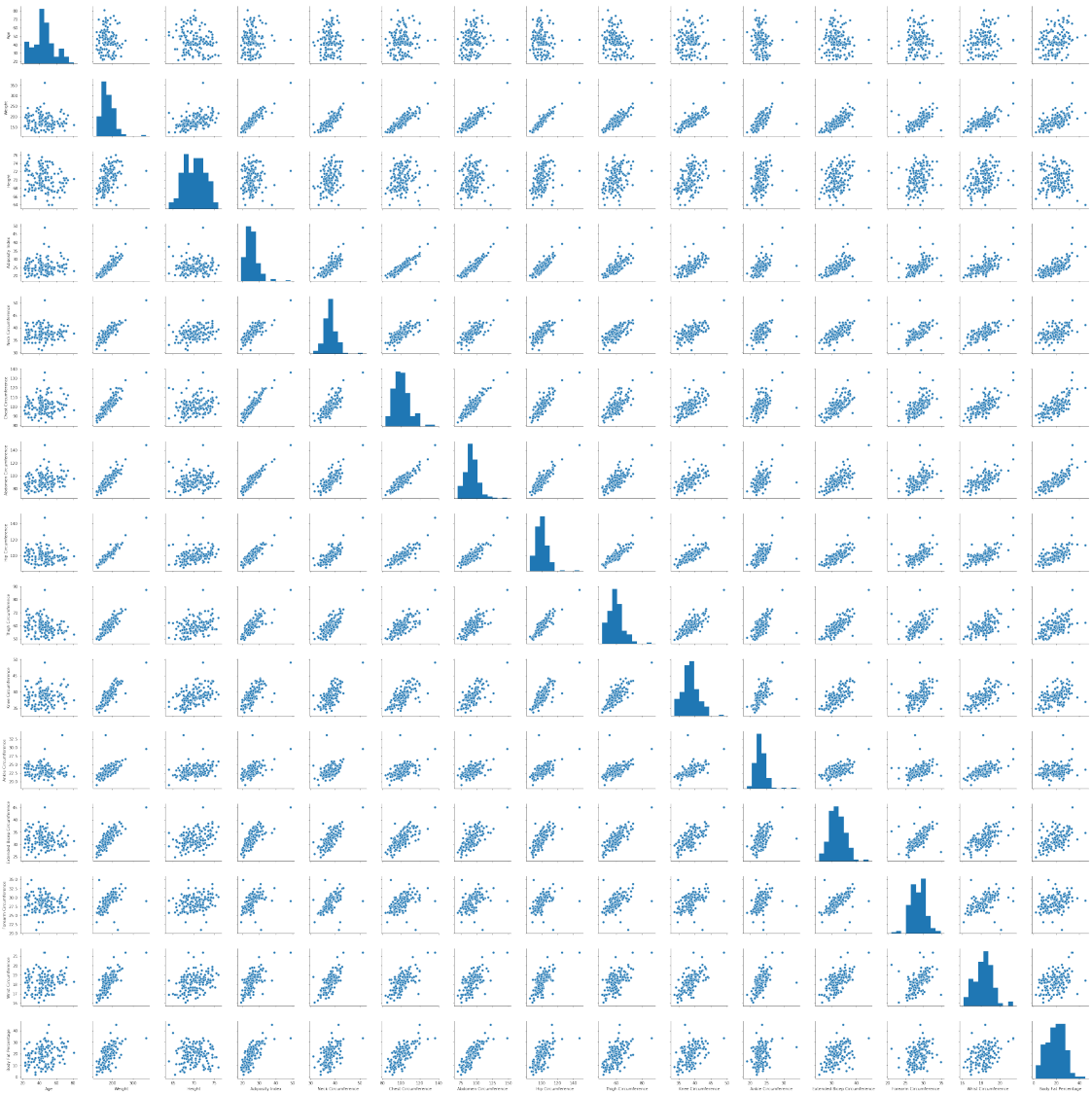
3.1 TO DO: Make sure max and min for each column is in training set

4 Regression

Explanation of regression

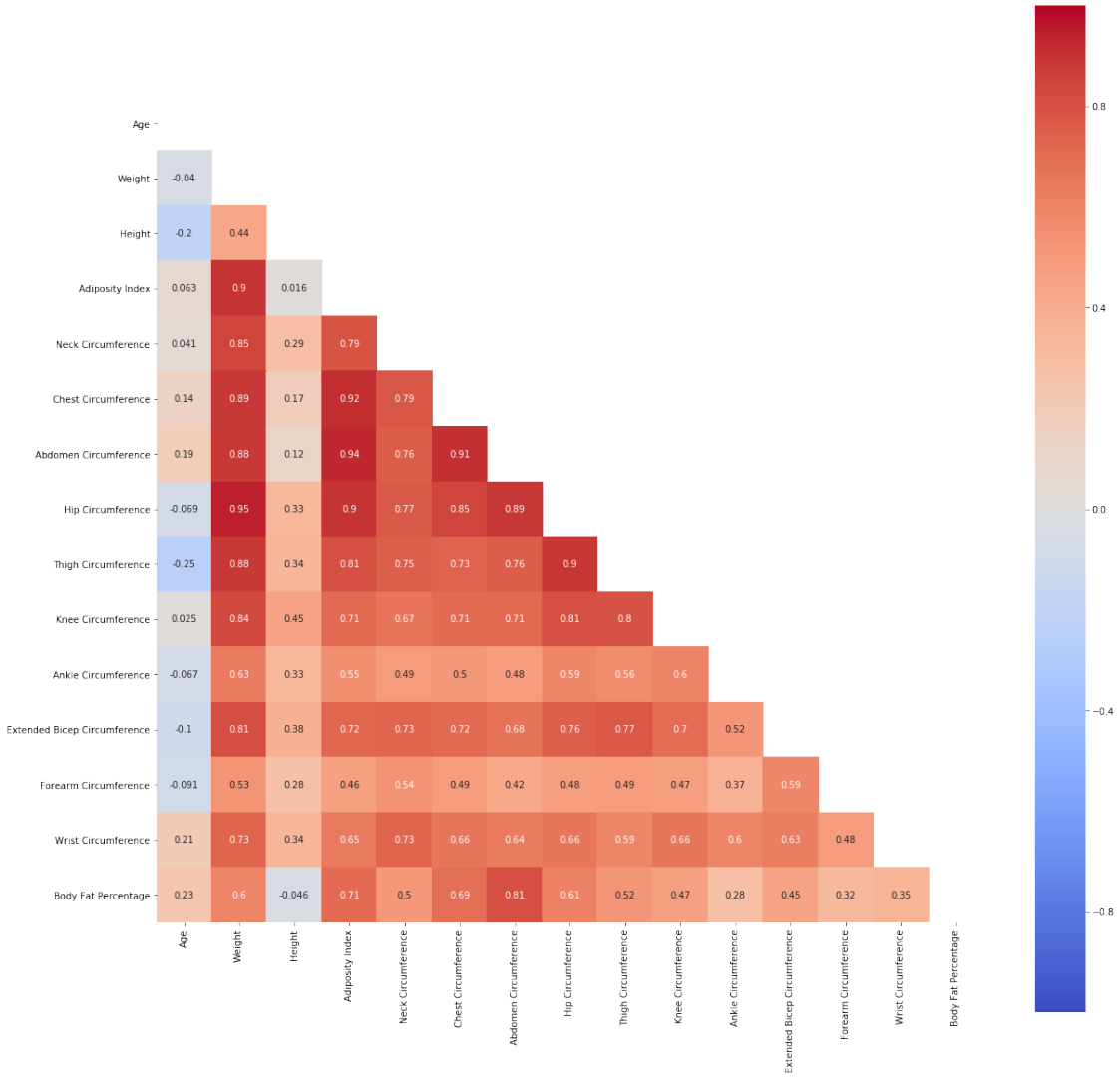
```
In [12]: train = data[data['sample']=='train']
```

```
In [13]: def regress(x, y):
             """Returns coefficient of regression"""
             # One-pad x
             o = np.ones_like(x[:,0]).reshape((-1, 1))

             x = np.hstack([x, o])

             # Perform regression
             coefficients = la.inv(x.T @ x) @ x.T @ y

             return coefficients

In [14]: def performance(x_test, y_test, coefficients):
             """Root mean squared error"""
             # One-pad x
             o = np.ones_like(x_test[:,0]).reshape((-1, 1))
             x_test = np.hstack([x_test, o])

             # Make predictions
             y_pred = x_test @ coefficients

             # Calculate RMSE
             return np.sqrt(np.sum((y_test - y_pred) ** 2) / y_test.shape[0])

In [15]: sns.pairplot(train)

Out[15]: <seaborn.axisgrid.PairGrid at 0x1a1e05cd30>
```



Wow, that's a little unweildy. Let's try just the correlation matrix

```
In [16]: plt.figure(figsize=(20,20))
         mask = np.zeros_like(train.corr(), dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True
         sns.heatmap(train.corr(), vmin=-1, vmax=1, center=0,
                     square=True, cmap='coolwarm', annot=True, mask=mask)
         plt.show()
```

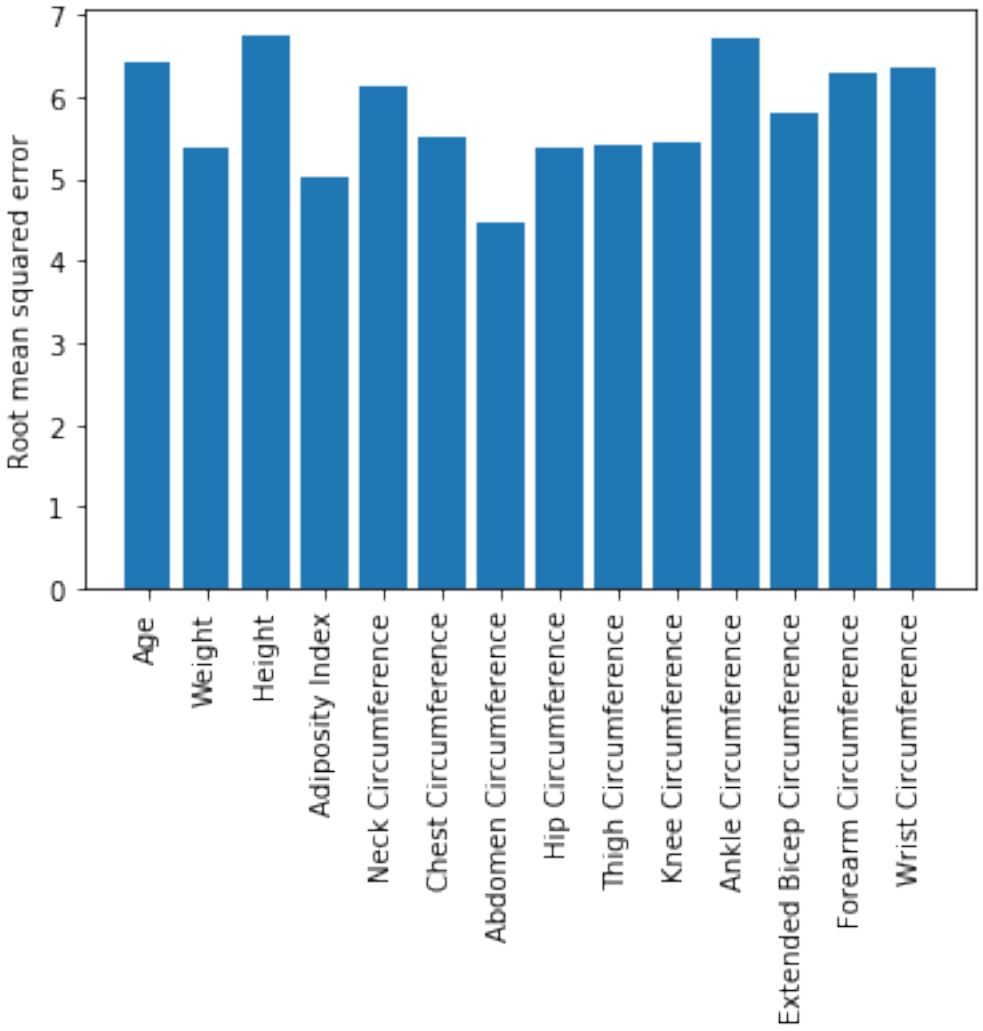


4.1 Attempt a 1-value regression for each column

```
In [17]: test = data[data['sample'] == 'test']
         test_performance = {'one_factor': {}}
         for name in column_names[:-1]:
             test_performance['one_factor'][name] = performance(
                 test[name].values.reshape(-1, 1),
                 test['Body Fat Percentage'].values,
                 regress(train[name].values.reshape(-1, 1), train['Body Fat Percentage'].values
             )
         print('Root mean squared error vs regressor')
         print('------------------------------------')
         for f, p in test_performance['one_factor'].items():
             print(f'{f}\t\t{p:0.2f}')
```

```
Root mean squared error vs regressor
------------------------------------
Age                 6.42
Weight                 5.39
Height                 6.74
Adiposity Index                5.03
Neck Circumference                6.11
Chest Circumference                5.49
Abdomen Circumference                4.47
Hip Circumference                5.36
Thigh Circumference                5.40
Knee Circumference                5.43
Ankle Circumference                6.72
Extended Bicep Circumference                5.81
Forearm Circumference                6.28
Wrist Circumference                6.35
```

```
In [18]: f = plt.figure()
         plt.bar(x=column_names[:-1], height=[v for v in test_performance['one_factor'].values
         plt.xticks(rotation=90)
         plt.ylabel('Root mean squared error')
         plt.show()
```



5 Pairwise Regression

```
In [19]: from itertools import combinations

In [20]: combs = combinations(column_names[:-1], 2)

In [21]: two_factor = {}
         for _ in combs:
             c = [i for i in _]
             p = performance(test[c].values, test['Body Fat Percentage'].values,
                             regress(train[c].values, train['Body Fat Percentage'].values))
             two_factor[tuple(c)] = p
             print(f'{c} \t\t\t\t {p:0.2f}')
```

```
['Age', 'Weight']                                  4.99
['Age', 'Height']                                  6.42
['Age', 'Adiposity Index']                                4.92
['Age', 'Neck Circumference']                                6.00
['Age', 'Chest Circumference']                                5.39
['Age', 'Abdomen Circumference']                                4.43
['Age', 'Hip Circumference']                                4.88
['Age', 'Thigh Circumference']                                4.68
['Age', 'Knee Circumference']                                4.92
['Age', 'Ankle Circumference']                                6.31
['Age', 'Extended Bicep Circumference']                                         5.45
['Age', 'Forearm Circumference']                                5.88
['Age', 'Wrist Circumference']                                6.13
['Weight', 'Height']                             5.12
['Weight', 'Adiposity Index']                                5.17
['Weight', 'Neck Circumference']                                5.38
['Weight', 'Chest Circumference']                                5.57
['Weight', 'Abdomen Circumference']                                4.48
['Weight', 'Hip Circumference']                                5.33
['Weight', 'Thigh Circumference']                                5.40
['Weight', 'Knee Circumference']                                5.45
['Weight', 'Ankle Circumference']                                5.23
['Weight', 'Extended Bicep Circumference']                                        5.46
['Weight', 'Forearm Circumference']                                5.40
['Weight', 'Wrist Circumference']                                5.17
['Height', 'Adiposity Index']                                5.11
['Height', 'Neck Circumference']                                6.36
['Height', 'Chest Circumference']                                5.56
['Height', 'Abdomen Circumference']                                4.30
['Height', 'Hip Circumference']                                5.30
['Height', 'Thigh Circumference']                                5.55
['Height', 'Knee Circumference']                                5.02
['Height', 'Ankle Circumference']                                6.84
['Height', 'Extended Bicep Circumference']                                        6.32
['Height', 'Forearm Circumference']                                6.45
['Height', 'Wrist Circumference']                                6.52
['Adiposity Index', 'Neck Circumference']                                       4.92
['Adiposity Index', 'Chest Circumference']                                       5.14
['Adiposity Index', 'Abdomen Circumference']                                         4.81
['Adiposity Index', 'Hip Circumference']                                      5.12
['Adiposity Index', 'Thigh Circumference']                                       5.15
['Adiposity Index', 'Knee Circumference']                                      5.16
['Adiposity Index', 'Ankle Circumference']                                       5.10
['Adiposity Index', 'Extended Bicep Circumference']                                                5.04
['Adiposity Index', 'Forearm Circumference']                                         5.03
['Adiposity Index', 'Wrist Circumference']                                       4.86
['Neck Circumference', 'Chest Circumference']                                          5.40
['Neck Circumference', 'Abdomen Circumference']                                            4.26
```

```
['Neck Circumference', 'Hip Circumference']                                5.37
['Neck Circumference', 'Thigh Circumference']                               5.46
['Neck Circumference', 'Knee Circumference']                               5.63
['Neck Circumference', 'Ankle Circumference']                               6.14
['Neck Circumference', 'Extended Bicep Circumference']                                5.90
['Neck Circumference', 'Forearm Circumference']                                6.14
['Neck Circumference', 'Wrist Circumference']                               6.11
['Chest Circumference', 'Abdomen Circumference']                                4.29
['Chest Circumference', 'Hip Circumference']                               5.40
['Chest Circumference', 'Thigh Circumference']                               5.46
['Chest Circumference', 'Knee Circumference']                               5.57
['Chest Circumference', 'Ankle Circumference']                               5.45
['Chest Circumference', 'Extended Bicep Circumference']                                5.54
['Chest Circumference', 'Forearm Circumference']                                5.47
['Chest Circumference', 'Wrist Circumference']                               5.31
['Abdomen Circumference', 'Hip Circumference']                               4.74
['Abdomen Circumference', 'Thigh Circumference']                                4.65
['Abdomen Circumference', 'Knee Circumference']                               4.57
['Abdomen Circumference', 'Ankle Circumference']                                4.41
['Abdomen Circumference', 'Extended Bicep Circumference']                                4.64
['Abdomen Circumference', 'Forearm Circumference']                                4.44
['Abdomen Circumference', 'Wrist Circumference']                                4.21
['Hip Circumference', 'Thigh Circumference']                               5.45
['Hip Circumference', 'Knee Circumference']                               5.44
['Hip Circumference', 'Ankle Circumference']                               5.32
['Hip Circumference', 'Extended Bicep Circumference']                                5.40
['Hip Circumference', 'Forearm Circumference']                               5.39
['Hip Circumference', 'Wrist Circumference']                               5.31
['Thigh Circumference', 'Knee Circumference']                               5.30
['Thigh Circumference', 'Ankle Circumference']                               5.39
['Thigh Circumference', 'Extended Bicep Circumference']                                5.38
['Thigh Circumference', 'Forearm Circumference']                                5.49
['Thigh Circumference', 'Wrist Circumference']                               5.46
['Knee Circumference', 'Ankle Circumference']                               5.44
['Knee Circumference', 'Extended Bicep Circumference']                                5.32
['Knee Circumference', 'Forearm Circumference']                               5.56
['Knee Circumference', 'Wrist Circumference']                               5.52
['Ankle Circumference', 'Extended Bicep Circumference']                                5.85
['Ankle Circumference', 'Forearm Circumference']                                6.46
['Ankle Circumference', 'Wrist Circumference']                               6.41
['Extended Bicep Circumference', 'Forearm Circumference']                                5.88
['Extended Bicep Circumference', 'Wrist Circumference']                                5.84
['Forearm Circumference', 'Wrist Circumference']                                6.35
```

```
In [22]: print(max(two_factor, key=two_factor.get))
         print(max([v for v in two_factor.values()]))
```

```
('Height', 'Ankle Circumference')
```

```
6.840207170399193
```

```
In [23]: print(min(two_factor, key=two_factor.get))
         print(min([v for v in two_factor.values()]))
```

```
('Abdomen Circumference', 'Wrist Circumference')
4.2060585890608495
```

6 All 14 Variables

```
In [24]: p = performance(x_test=test[column_names[:-1]].values,
                         y_test=test['Body Fat Percentage'].values,
                         coefficients=regress(
                             train[column_names[:-1]].values,
                             train['Body Fat Percentage'].values,
                         ))
         test_performance['All inputs performance'] = p
         print(f'All inputs performance\t\t{p:0.2f}')
```

```
All inputs performance                4.55
```

7 My best guess

```
In [91]: predictive_columns = ['Abdomen Circumference', 'Wrist Circumference', 'Height']
         p = performance(x_test=test[predictive_columns].values,
                         y_test=test['Body Fat Percentage'].values,
                         coefficients=regress(
                             train[predictive_columns].values,
                             train['Body Fat Percentage'].values,
                         ))
         test_performance['Predictive columns performance'] = p
         print(f'Predictive columns performance\t\t{p:0.2f}')
```

```
Predictive columns performance                4.14
```

8 Non-linear terms

In order to improve the model it's worth trying higher-order (non-linear) terms.

Add a term for the square of each input variable

```
In [26]: for c in column_names[:-1]:
             data[f'{c}^2'] = data[c] * data[c]
```

Add a term for the product of each combination of two variables

```
In [27]: combs = combinations(column_names[:-1], 2)
         for c1, c2 in combs:
             data[f'{c1} x {c2}'] = data[c1] * data[c2]

In [35]: for c in column_names[:-1]:
             data[f'{c}^-1'] = data[c] ** -1

In [36]: train = data[data['sample']=='train']
         test = data[data['sample'] == 'test']

In [37]: train.head()

Out[37]:     Age  Weight  Height  Adiposity Index  Neck Circumference  \
         0  25.0  206.50   69.75             29.8                40.9
         1  62.0  167.50   71.50             23.1                35.5
         2  52.0  206.50   74.50             26.2                40.8
         3  81.0  161.25   70.25             23.0                37.8
         4  39.0  166.75   70.75             23.5                37.0

            Chest Circumference  Abdomen Circumference  Hip Circumference  \
         0                110.9                  100.5              106.2
         1                 97.6                   91.5               98.5
         2                104.3                   99.2              104.1
         3                 96.4                   95.4               99.3
         4                 92.9                   86.1               95.6

            Thigh Circumference  Knee Circumference           ...            \
         0                 68.4                40.8           ...
         1                 56.6                38.6           ...
         2                 58.5                39.3           ...
         3                 53.5                37.5           ...
         4                 58.8                36.1           ...

            Neck Circumference^-1  Chest Circumference^-1  Abdomen Circumference^-1  \
         0               0.024450                0.009017                  0.009950
         1               0.028169                0.010246                  0.010929
         2               0.024510                0.009588                  0.010081
         3               0.026455                0.010373                  0.010482
         4               0.027027                0.010764                  0.011614

            Hip Circumference^-1  Thigh Circumference^-1 Knee Circumference^-1  \
         0              0.009416                0.014620              0.024510
         1              0.010152                0.017668              0.025907
         2              0.009606                0.017094              0.025445
         3              0.010070                0.018692              0.026667
         4              0.010460                0.017007              0.027701

            Ankle Circumference^-1  Extended Bicep Circumference^-1  \
         0                0.040650                         0.030030
```

```
1                0.044643                          0.031746
2                0.040650                          0.029499
3                0.046512                          0.031847
4                0.044643                          0.030581

   Forearm Circumference^-1  Wrist Circumference^-1
0                  0.033670                0.054348
1                  0.036630                0.053763
2                  0.032051                0.051282
3                  0.037313                0.054645
4                  0.035336                0.058480

[5 rows x 135 columns]
```

```
In [38]: train.columns
```

```
Out[38]: Index(['Age', 'Weight', 'Height', 'Adiposity Index', 'Neck Circumference',
                'Chest Circumference', 'Abdomen Circumference', 'Hip Circumference',
                'Thigh Circumference', 'Knee Circumference',
                ...
                'Neck Circumference^-1', 'Chest Circumference^-1',
                'Abdomen Circumference^-1', 'Hip Circumference^-1',
                'Thigh Circumference^-1', 'Knee Circumference^-1',
                'Ankle Circumference^-1', 'Extended Bicep Circumference^-1',
                'Forearm Circumference^-1', 'Wrist Circumference^-1'],
               dtype='object', length=135)
```

```
In [63]: train_x = train.drop(['Body Fat Percentage', 'sample'], axis=1)
         print(*train_x.columns, sep='\n')
```

```
Age
Weight
Height
Adiposity Index
Neck Circumference
Chest Circumference
Abdomen Circumference
Hip Circumference
Thigh Circumference
Knee Circumference
Ankle Circumference
Extended Bicep Circumference
Forearm Circumference
Wrist Circumference
Age^2
Weight^2
Height^2
Adiposity Index^2
Neck Circumference^2
```

```
Chest Circumference^2
Abdomen Circumference^2
Hip Circumference^2
Thigh Circumference^2
Knee Circumference^2
Ankle Circumference^2
Extended Bicep Circumference^2
Forearm Circumference^2
Wrist Circumference^2
Age x Weight
Age x Height
Age x Adiposity Index
Age x Neck Circumference
Age x Chest Circumference
Age x Abdomen Circumference
Age x Hip Circumference
Age x Thigh Circumference
Age x Knee Circumference
Age x Ankle Circumference
Age x Extended Bicep Circumference
Age x Forearm Circumference
Age x Wrist Circumference
Weight x Height
Weight x Adiposity Index
Weight x Neck Circumference
Weight x Chest Circumference
Weight x Abdomen Circumference
Weight x Hip Circumference
Weight x Thigh Circumference
Weight x Knee Circumference
Weight x Ankle Circumference
Weight x Extended Bicep Circumference
Weight x Forearm Circumference
Weight x Wrist Circumference
Height x Adiposity Index
Height x Neck Circumference
Height x Chest Circumference
Height x Abdomen Circumference
Height x Hip Circumference
Height x Thigh Circumference
Height x Knee Circumference
Height x Ankle Circumference
Height x Extended Bicep Circumference
Height x Forearm Circumference
Height x Wrist Circumference
Adiposity Index x Neck Circumference
Adiposity Index x Chest Circumference
Adiposity Index x Abdomen Circumference
```

Adiposity Index x Hip Circumference
Adiposity Index x Thigh Circumference
Adiposity Index x Knee Circumference
Adiposity Index x Ankle Circumference
Adiposity Index x Extended Bicep Circumference
Adiposity Index x Forearm Circumference
Adiposity Index x Wrist Circumference
Neck Circumference x Chest Circumference
Neck Circumference x Abdomen Circumference
Neck Circumference x Hip Circumference
Neck Circumference x Thigh Circumference
Neck Circumference x Knee Circumference
Neck Circumference x Ankle Circumference
Neck Circumference x Extended Bicep Circumference
Neck Circumference x Forearm Circumference
Neck Circumference x Wrist Circumference
Chest Circumference x Abdomen Circumference
Chest Circumference x Hip Circumference
Chest Circumference x Thigh Circumference
Chest Circumference x Knee Circumference
Chest Circumference x Ankle Circumference
Chest Circumference x Extended Bicep Circumference
Chest Circumference x Forearm Circumference
Chest Circumference x Wrist Circumference
Abdomen Circumference x Hip Circumference
Abdomen Circumference x Thigh Circumference
Abdomen Circumference x Knee Circumference
Abdomen Circumference x Ankle Circumference
Abdomen Circumference x Extended Bicep Circumference
Abdomen Circumference x Forearm Circumference
Abdomen Circumference x Wrist Circumference
Hip Circumference x Thigh Circumference
Hip Circumference x Knee Circumference
Hip Circumference x Ankle Circumference
Hip Circumference x Extended Bicep Circumference
Hip Circumference x Forearm Circumference
Hip Circumference x Wrist Circumference
Thigh Circumference x Knee Circumference
Thigh Circumference x Ankle Circumference
Thigh Circumference x Extended Bicep Circumference
Thigh Circumference x Forearm Circumference
Thigh Circumference x Wrist Circumference
Knee Circumference x Ankle Circumference
Knee Circumference x Extended Bicep Circumference
Knee Circumference x Forearm Circumference
Knee Circumference x Wrist Circumference
Ankle Circumference x Extended Bicep Circumference
Ankle Circumference x Forearm Circumference

```
Ankle Circumference x Wrist Circumference
Extended Bicep Circumference x Forearm Circumference
Extended Bicep Circumference x Wrist Circumference
Forearm Circumference x Wrist Circumference
Age^-1
Weight^-1
Height^-1
Adiposity Index^-1
Neck Circumference^-1
Chest Circumference^-1
Abdomen Circumference^-1
Hip Circumference^-1
Thigh Circumference^-1
Knee Circumference^-1
Ankle Circumference^-1
Extended Bicep Circumference^-1
Forearm Circumference^-1
Wrist Circumference^-1
```

```
In [75]: theory_1_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wrist Circumferer
         c = theory_1_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[75]: 4.1141880618732145
```

```
In [79]: theory_2_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wrist Circumferer
         c = theory_2_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[79]: 4.1141880618732145
```

```
In [80]: theory_3_columns = train_x.columns
         c = theory_3_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[80]: 29.01867845252937
```

9 Final Validation Performance Calculations

The final performance for the "best" models from the categories above will now be calculated on the validation set

```
In [82]: val = data[data['sample'] == 'val']
```

9.1 Single variable performance

```
In [83]: one_variable_columns = ['Abdomen Circumference']
         c = one_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[83]: 4.14342725159527
```

9.2 Two variable performance

```
In [84]: two_variable_columns = ['Abdomen Circumference', 'Wrist Circumference']
         c = two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[84]: 4.069138409016576
```

```
In [88]: other_two_variable_columns = ['Abdomen Circumference', 'Height']
         c = other_two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[88]: 3.6696041010002514
```

```
In [89]: another_two_variable_columns = ['Abdomen Circumference', 'Age']
         c = another_two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[89]: 3.961070321190066
```

9.3 14 variable performance

```
In [85]: fourteen_variable_columns = column_names[:-1]
         c = fourteen_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[85]: 3.751645868409825
```

Just goes to show the value of a validation set...

9.4 "Best guess" variables

```
In [90]: best_guess_columns = ['Abdomen Circumference', 'Wrist Circumference', 'Height']
         c = best_guess_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[90]: 3.8254932073596883
```

9.5 Linear in parameters

```
In [87]: linear_in_parameters_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wris
         c = linear_in_parameters_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                              regress(train[c].values, train['Body Fat Percentage'].values))

Out[87]: 3.868774955037983

In [93]: s = train['Body Fat Percentage'].std()

In [94]: s

Out[94]: 7.878909548411372

In [95]: s/2

Out[95]: 3.939454774205686

In [96]: 6.84 / s

Out[96]: 0.8681404397362517

In [97]: 3.87 / s

Out[97]: 0.4911847224823529

In [98]: from scipy.stats import norm

In [99]: norm.cdf(0.5) - norm.cdf(-0.5)

Out[99]: 0.38292492254802624

In [100]: norm.cdf(1.0) - norm.cdf(-1.0)

Out[100]: 0.6826894921370859

In [101]: norm.cdf(6.84 / s) - norm.cdf(-6.84 / s)

Out[101]: 0.6146825449271656
```