akulas_HW2

January 16, 2019

- Adam Kulas
- watiam: akulas
- student number: 20302000

References: About Feature Scaling and Normalization,Sebastian Raschka

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import preprocessing
        from IPython.display import display, HTML
```

```
In [2]: data = pd.read_csv('data/winequality-red.csv', sep=';')
        data = data.iloc[:10, :]
        display(data)
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076
5            7.4              0.66         0.00             1.8      0.075
6            7.9              0.60         0.06             1.6      0.069
7            7.3              0.65         0.00             1.2      0.065
8            7.8              0.58         0.02             2.0      0.073
9            7.5              0.50         0.36             6.1      0.071

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56
5                 13.0                  40.0   0.9978  3.51       0.56
6                 15.0                  59.0   0.9964  3.30       0.46
7                 15.0                  21.0   0.9946  3.39       0.47
8                  9.0                  18.0   0.9968  3.36       0.57
9                 17.0                 102.0   0.9978  3.35       0.80
```

```
   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
5      9.4        5
6      9.4        5
7     10.0        7
8      9.5        7
9     10.5        5
```

1 Calculate the Min Max Scaling of the Data

```
In [3]: # Calculating the min max scaling manually
        datamm = (data - data.min()) / (data.max() -data.min())
```

```
In [4]: #Calculating min max scaling using sklearn
        minmax_scale = preprocessing.MinMaxScaler().fit(data)
        data_minmax = minmax_scale.transform(data)
        data_minmax = pd.DataFrame(data_minmax)
        data_minmax.columns = data.columns
        display(data_minmax)
```

```
/home/calkulas/anaconda3/envs/ece657A/lib/python3.7/site-packages/sklearn/preprocessing/data.py
  return self.partial_fit(X, y)
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.025641          0.700000     0.000000        0.142857   0.333333
1       0.128205          1.000000     0.000000        0.285714   1.000000
2       0.128205          0.800000     0.071429        0.224490   0.818182
3       1.000000          0.000000     1.000000        0.142857   0.303030
4       0.025641          0.700000     0.000000        0.142857   0.333333
5       0.025641          0.633333     0.000000        0.122449   0.303030
6       0.153846          0.533333     0.107143        0.081633   0.121212
7       0.000000          0.616667     0.000000        0.000000   0.000000
8       0.128205          0.500000     0.035714        0.163265   0.242424
9       0.051282          0.366667     0.642857        1.000000   0.181818

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0                0.125              0.190476  0.941176  1.000000   0.294118
1                1.000              0.583333  0.647059  0.114286   0.647059
2                0.375              0.428571  0.705882  0.285714   0.558824
3                0.500              0.500000  1.000000  0.000000   0.352941
4                0.125              0.190476  0.941176  1.000000   0.294118
```

```
5                0.250              0.261905  0.941176  1.000000   0.294118
6                0.375              0.488095  0.529412  0.400000   0.000000
7                0.375              0.035714  0.000000  0.657143   0.029412
8                0.000              0.000000  0.647059  0.571429   0.323529
9                0.500              1.000000  0.941176  0.542857   1.000000

    alcohol  quality
0  0.000000      0.0
1  0.363636      0.0
2  0.363636      0.0
3  0.363636      0.5
4  0.000000      0.0
5  0.000000      0.0
6  0.000000      0.0
7  0.545455      1.0
8  0.090909      1.0
9  1.000000      0.0
```

2 Calculate the Z score of the data

```
In [5]: # calculating the z score manually
        # print(data.mean(), data.std(), data)
        data_zscore = (data - data.mean()) / data.std()
        #display(data_zscore)
```

```
In [6]: std_scale = preprocessing.StandardScaler().fit(data)
        data_zscore = pd.DataFrame(std_scale.transform(data))
        data_zscore.columns = data.columns
        display(data_zscore)
```

```
/home/calkulas/anaconda3/envs/ece657A/lib/python3.7/site-packages/sklearn/preprocessing/data.py
  return self.partial_fit(X, y)
/home/calkulas/anaconda3/envs/ece657A/lib/python3.7/site-packages/ipykernel_launcher.py:2: Data
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0      -0.498662          0.451753    -0.563489       -0.329398  -0.103362
1      -0.135999          1.630239    -0.563489        0.206831   2.170608
2      -0.135999          0.844582    -0.346763       -0.022981   1.550434
3       2.946642         -2.298048     2.470683       -0.329398  -0.206725
4      -0.498662          0.451753    -0.563489       -0.329398  -0.103362
5      -0.498662          0.189867    -0.563489       -0.406002  -0.206725
6      -0.045333         -0.202961    -0.238399       -0.559211  -0.826898
7      -0.589328          0.124396    -0.563489       -0.865627  -1.240347
8      -0.135999         -0.333904    -0.455126       -0.252794  -0.413449
9      -0.407997         -0.857676     1.387050        2.887978  -0.620174
```

```
   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0            -0.896665             -0.626576  0.731200  1.276466  -0.305196
1             2.406839              0.761143 -0.284356 -1.276466   0.957683
2             0.047193              0.214466 -0.081244 -0.782350   0.641963
3             0.519122              0.466778  0.934311 -1.605877  -0.094716
4            -0.896665             -0.626576  0.731200  1.276466  -0.305196
5            -0.424736             -0.374264  0.731200  1.276466  -0.305196
6             0.047193              0.424726 -0.690578 -0.452940  -1.357594
7             0.047193             -1.173253 -2.518578  0.288234  -1.252354
8            -1.368595             -1.299409 -0.284356  0.041176  -0.199956
9             0.519122              2.232966  0.731200 -0.041176   2.220561

    alcohol   quality
0 -0.880830 -0.620174
1  0.293610 -0.620174
2  0.293610 -0.620174
3  0.293610  0.620174
4 -0.880830 -0.620174
5 -0.880830 -0.620174
6 -0.880830 -0.620174
7  0.880830  1.860521
8 -0.587220  1.860521
9  2.348881 -0.620174
```

3 Calculate the mean substracted normalized values

```
In [7]: data_meansub = data - data.mean()
        display(data_meansub)
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0          -0.55             0.069       -0.104           -0.43     -0.001
1          -0.15             0.249       -0.104            0.27      0.021
2          -0.15             0.129       -0.064           -0.03      0.015
3           3.25            -0.351        0.456           -0.43     -0.002
4          -0.55             0.069       -0.104           -0.43     -0.001
5          -0.55             0.029       -0.104           -0.53     -0.002
6          -0.05            -0.031       -0.044           -0.73     -0.008
7          -0.65             0.019       -0.104           -1.13     -0.012
8          -0.15            -0.051       -0.084           -0.33     -0.004
9          -0.45            -0.131        0.256            3.77     -0.006

   free sulfur dioxide  total sulfur dioxide  density     pH  sulphates  \
0                 -3.8                 -14.9  0.00072  0.155     -0.029
1                 10.2                  18.1 -0.00028 -0.155      0.091
2                  0.2                   5.1 -0.00008 -0.095      0.061
```

```
3                  2.2                    11.1  0.00092 -0.195     -0.009
4                 -3.8                   -14.9  0.00072  0.155     -0.029
5                 -1.8                    -8.9  0.00072  0.155     -0.029
6                  0.2                    10.1 -0.00068 -0.055     -0.129
7                  0.2                   -27.9 -0.00248  0.035     -0.119
8                 -5.8                   -30.9 -0.00028  0.005     -0.019
9                  2.2                    53.1  0.00072 -0.005      0.211

   alcohol  quality
0     -0.3     -0.5
1      0.1     -0.5
2      0.1     -0.5
3      0.1      0.5
4     -0.3     -0.5
5     -0.3     -0.5
6     -0.3     -0.5
7      0.3      1.5
8     -0.2      1.5
9      0.8     -0.5
```

4 Calcualte distance of data

```
In [8]: from scipy.spatial import distance
```

4.1 Manhatten Distance

```
In [9]: # calculate the pairwise distance using the manhatten distance metic
        d = pd.Series(distance.pdist(data, 'cityblock'))
        dsq = pd.DataFrame(distance.squareform(d))
        np.fill_diagonal(dsq.values, np.nan)
        display(dsq)
```

```
         0        1        2        3        4        5        6        7  \
0      NaN  49.1330  25.6568  38.5512   0.0000   8.1410  30.2784  20.6742
1  49.1330      NaN  23.5562  21.4242  49.1330  41.2740  20.1894  60.7652
2  25.6568  23.5562      NaN  13.9880  25.6568  17.7978   6.6336  37.2894
3  38.5512  21.4242  13.9880      NaN  38.5512  30.6102   9.0876  48.0834
4   0.0000  49.1330  25.6568  38.5512      NaN   8.1410  30.2784  20.6742
5   8.1410  41.2740  17.7978  30.6102   8.1410      NaN  22.1374  24.5332
6  30.2784  20.1894   6.6336   9.0876  30.2784  22.1374      NaN  41.8158
7  20.6742  60.7652  37.2894  48.0834  20.6742  24.5332  41.8158      NaN
8  20.9040  68.5150  44.9992  55.8532  20.9040  28.9630  49.8344  11.0302
9  80.3650  48.5380  55.6418  52.4342  80.3650  72.4240  51.7934  91.4892

          8        9
0   20.9040  80.3650
1   68.5150  48.5380
```

```
2   44.9992   55.6418
3   55.8532   52.4342
4   20.9040   80.3650
5   28.9630   72.4240
6   49.8344   51.7934
7   11.0302   91.4892
8       NaN  100.0630
9  100.0630       NaN
```

```
In [10]: distances = pd.DataFrame([dsq.idxmin(axis=0), dsq.min(axis=0), dsq.idxmax(axis=0), dsq
         distances.columns = ['nearest_manhatten_point', 'near_manhatten_dist_val','far_manhat
         display(distances)
```

```
   nearest_manhatten_point  near_manhatten_dist_val  far_manhatten_point  \
0                      4.0                   0.0000                  9.0
1                      6.0                  20.1894                  8.0
2                      6.0                   6.6336                  9.0
3                      6.0                   9.0876                  8.0
4                      0.0                   0.0000                  9.0
5                      0.0                   8.1410                  9.0
6                      2.0                   6.6336                  9.0
7                      8.0                  11.0302                  9.0
8                      7.0                  11.0302                  9.0
9                      1.0                  48.5380                  8.0

   far_manhatten_dist_val
0                 80.3650
1                 68.5150
2                 55.6418
3                 55.8532
4                 80.3650
5                 72.4240
6                 51.7934
7                 91.4892
8                100.0630
9                100.0630
```

4.2 Cosine Distance

```
In [11]: # calculate the pairwise distance using the cosine distance metic
         d = pd.Series(distance.pdist(data, 'cosine'))
         dsq = pd.DataFrame(distance.squareform(d))
         np.fill_diagonal(dsq.values, np.nan)
         display(dsq)
```

```
          0         1         2         3         4         5         6  \
0       NaN  0.015207  0.007749  0.007776  0.000000  0.001322  0.011912
```

```
1  0.015207       NaN  0.004858  0.005985  0.015207  0.007860  0.006094
2  0.007749  0.004858       NaN  0.001097  0.007749  0.003120  0.000601
3  0.007776  0.005985  0.001097       NaN  0.007776  0.003502  0.001635
4  0.000000  0.015207  0.007749  0.007776       NaN  0.001322  0.011912
5  0.001322  0.007860  0.003120  0.003502  0.001322       NaN  0.006079
6  0.011912  0.006094  0.000601  0.001635  0.011912  0.006079       NaN
7  0.051499  0.082365  0.089001  0.086412  0.051499  0.060757  0.102130
8  0.045682  0.099274  0.088973  0.085518  0.045682  0.060301  0.101541
9  0.035050  0.019639  0.011228  0.014255  0.035050  0.025574  0.007549

          7         8         9
0  0.051499  0.045682  0.035050
1  0.082365  0.099274  0.019639
2  0.089001  0.088973  0.011228
3  0.086412  0.085518  0.014255
4  0.051499  0.045682  0.035050
5  0.060757  0.060301  0.025574
6  0.102130  0.101541  0.007549
7       NaN  0.015776  0.159302
8  0.015776       NaN  0.153601
9  0.159302  0.153601       NaN
```

```
In [12]: distances = pd.DataFrame([dsq.idxmin(axis=0), dsq.min(axis=0), dsq.idxmax(axis=0), dsq
         distances.columns = ['nearest_cosine_point', 'near_cosine_dist_val','far_cosine_point
         display(distances)
```

```
   nearest_cosine_point  near_cosine_dist_val  far_cosine_point  \
0                   4.0              0.000000               7.0
1                   2.0              0.004858               8.0
2                   6.0              0.000601               7.0
3                   2.0              0.001097               7.0
4                   0.0              0.000000               7.0
5                   0.0              0.001322               7.0
6                   2.0              0.000601               7.0
7                   8.0              0.015776               9.0
8                   7.0              0.015776               9.0
9                   6.0              0.007549               7.0

   far_cosine_dist_val
0             0.051499
1             0.099274
2             0.089001
3             0.086412
4             0.051499
5             0.060757
6             0.102130
7             0.159302
```

```
8             0.153601
9             0.159302
```

4.3 Euclidean Distance

```
In [13]: # calculate the pairwise distance using the euclidean distance metic
         d = pd.Series(distance.pdist(data, 'euclidean'))
         dsq = pd.DataFrame(distance.squareform(d))
         np.fill_diagonal(dsq.values, np.nan)
         display(dsq)
```

```
           0          1          2          3          4          5  \
0        NaN  35.860192  20.409705  26.985420   0.000000   6.325472
1  35.860192        NaN  16.404589  11.257696  35.860192  29.565511
2  20.409705  16.404589        NaN   7.296300  20.409705  14.165186
3  26.985420  11.257696   7.296300        NaN  26.985420  20.789202
4   0.000000  35.860192  20.409705  26.985420        NaN   6.325472
5   6.325472  29.565511  14.165186  20.789202   6.325472        NaN
6  25.326029  12.857342   5.071906   4.186459  25.326029  19.114166
7  13.779881  47.142170  33.084192  39.271562  13.779881  19.228955
8  16.254766  51.590491  36.554474  42.907828  16.254766  22.455276
9  68.404041  36.085200  48.199803  42.390953  68.404041  62.289232

           6          7          8          9
0  25.326029  13.779881  16.254766  68.404041
1  12.857342  47.142170  51.590491  36.085200
2   5.071906  33.084192  36.554474  48.199803
3   4.186459  39.271562  42.907828  42.390953
4  25.326029  13.779881  16.254766  68.404041
5  19.114166  19.228955  22.455276  62.289232
6        NaN  38.064344  41.487320  43.299401
7  38.064344        NaN   6.793841  81.200755
8  41.487320   6.793841        NaN  84.510798
9  43.299401  81.200755  84.510798        NaN
```

```
In [14]: distances = pd.DataFrame([dsq.idxmin(axis=0), dsq.min(axis=0), dsq.idxmax(axis=0), dsq
         distances.columns = ['nearest_euclidean_point', 'near_euclidean_dist_val','far_euclide
         display(distances)
```

```
   nearest_euclidean_point  near_euclidean_dist_val  far_euclidean_point  \
0                      4.0                 0.000000                  9.0
1                      3.0                11.257696                  8.0
2                      6.0                 5.071906                  9.0
3                      6.0                 4.186459                  8.0
4                      0.0                 0.000000                  9.0
5                      0.0                 6.325472                  9.0
6                      3.0                 4.186459                  9.0
```

```
7                       8.0             6.793841                  9.0
8                       7.0             6.793841                  9.0
9                       1.0            36.085200                  8.0

   far_euclidean_dist_val
0               68.404041
1               51.590491
2               48.199803
3               42.907828
4               68.404041
5               62.289232
6               43.299401
7               81.200755
8               84.510798
9               84.510798
```