assignment-2

February 17, 2018

1 Assignment 2

Problem 1 (15 points): This problem is an example of data preprocessing needed in a data mining process. Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

Age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
Age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

a. (3 points) Draw the box-plots for age and % fat. Explain what you can tell from this visualization of the distribution of the data.

b. (3 points) Normalize the two attributes based on z-score normalization. Include an image showing the data table with this done.

c. (3 points) Regardless of the original ranges of the variables, normalization techniques transform the data into new ranges that allow to compare and use variables on the same scales. What are the value ranges of the following normalization methods applied to this data? Explain your answer by explaining how the methods work on data in general.

d. Min-max normalization (use default target interval 0 to 1)

ii. Z-score normalization

iii. Normalization by decimal scaling.

iv. (3 points) Draw a scatter-plot based on the two variables and visually interpret the relationship between the two variables.

e. (3 points) Correlation is useful when integrating or cleaning data to see if two variables are so strongly correlated that they should be checked to see if they duplicate information. Get the full covariance and correlation matrix giving the relationships between all pairs of variables, even though there are only two. Are these two variables positively or negatively correlated?

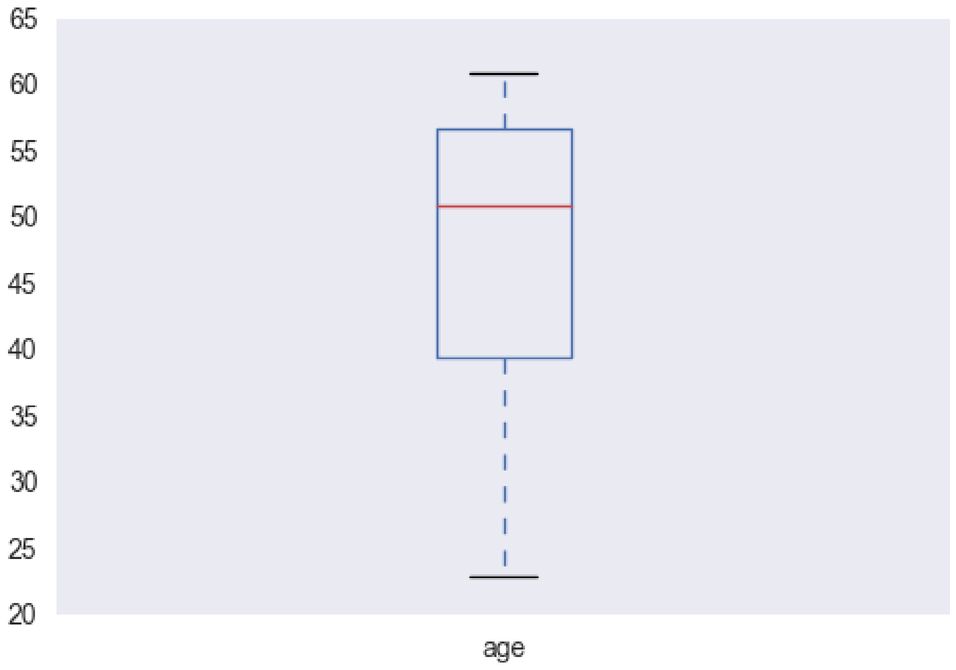
```
In [41]: #Import packages
         import pandas as pd
         import numpy as np
         import matplotlib
         from matplotlib import pyplot as plt
         from sklearn.preprocessing import MinMaxScaler
         import seaborn as sns
         %matplotlib inline
```

Problem 1-a

```
In [2]: #Create dataframe
        df = pd.DataFrame({'age': [23, 23, 27, 27, 39, 41, 47, 49, 50, 52, 54, 54,
```

```
In [14]: df.age.plot(kind = 'box', grid = False, whis = 'range')
```

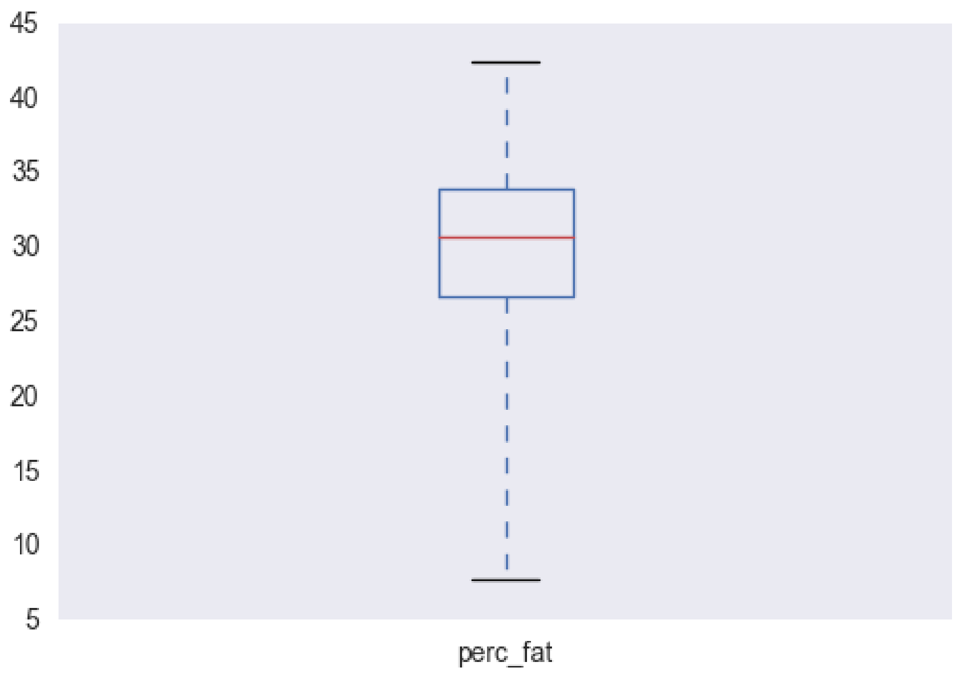
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x219e6047630>
```



The age variable has a large spread, with a min of 23 and a max of 61. Here we see that more of the age data tends to be on the older side since the median is age 51. This is skewing the data.

```
In [15]: df.perc_fat.plot(kind = 'box', grid = False, whis = 'range')
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x219e60c9c50>
```



The percent body fat variable has a slightly smaller spread, though is almost just as skewed. The min is 7.8 while the max is 42.5. Q1-Q3 have a smaller range, which would suggest that data points around the min could be considered outliers.

Problem 1-b

```
In [24]: #Compute z-score for each column
         norm = [((df['%s' %(col)] - df['%s' %(col)].mean())/df['%s' %(col)].std())

         #Create dataframe, transpose, and rename columns
         z_norm_df = pd.DataFrame.from_records(norm)

         z_norm_df = z_norm_df.transpose()
         z_norm_df.rename(columns = {0: 'z_norm', 1: 'z_perc_fat'}, inplace = True)

         #Show df
         z_norm_df
```

```
Out[24]:       z_norm  z_perc_fat
         0  -1.773592   -2.083695
         1  -1.773592   -0.246730
         2  -1.470989   -2.267391
         3  -1.470989   -1.186824
         4  -0.563178    0.282749
         5  -0.411877   -0.311564
         6   0.042028   -0.149479
```

```
7   0.193330  -0.171090
8   0.268981   0.261137
9   0.420282   0.628530
10  0.571584   1.482179
11  0.571584   0.001801
12  0.722886   0.498862
13  0.798537   0.153080
14  0.874187   0.574502
15  0.874187   0.444834
16  1.025489   1.341705
17  1.101140   0.747393
```

Problem 1-c

```
In [27]: #Min-max
         min_max_df = (df - df.min()) / (df.max() - df.min())
         min_max_df
```

```
Out[27]:          age  perc_fat
         0   0.000000  0.048991
         1   0.000000  0.538905
         2   0.105263  0.000000
         3   0.105263  0.288184
         4   0.421053  0.680115
         5   0.473684  0.521614
         6   0.631579  0.564841
         7   0.684211  0.559078
         8   0.710526  0.674352
         9   0.763158  0.772334
         10  0.815789  1.000000
         11  0.815789  0.605187
         12  0.868421  0.737752
         13  0.894737  0.645533
         14  0.921053  0.757925
         15  0.921053  0.723343
         16  0.973684  0.962536
         17  1.000000  0.804035
```

```
In [33]: #Decimal normilization (abs(max())) = 61
         dec_norm_df = df / 100
         dec_norm_df
```

```
Out[33]:      age  perc_fat
         0   0.23     0.095
         1   0.23     0.265
         2   0.27     0.078
         3   0.27     0.178
         4   0.39     0.314
```

```
5   0.41     0.259
6   0.47     0.274
7   0.49     0.272
8   0.50     0.312
9   0.52     0.346
10  0.54     0.425
11  0.54     0.288
12  0.56     0.334
13  0.57     0.302
14  0.58     0.341
15  0.58     0.329
16  0.60     0.412
17  0.61     0.357
```

i. Min-max normalization (use default target interval 0 to 1) This normilization technique transforms the raw data linearly and preserves the relationship of the raw data. The range for both age and perc_fat is (0, 1)

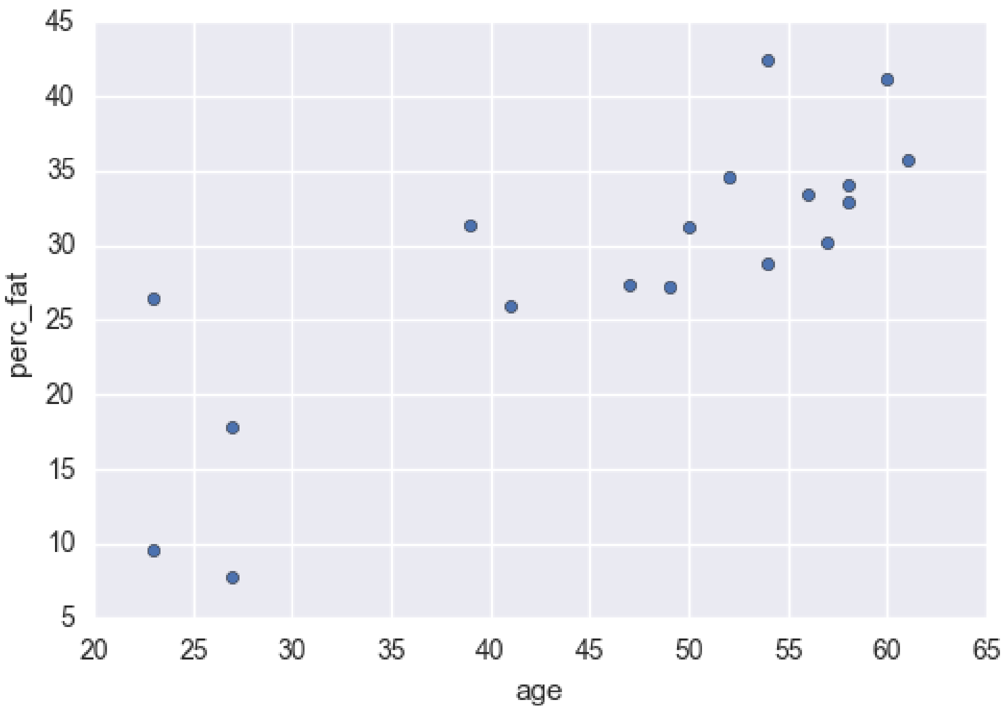
ii. Z-score normalization (computed in 1-b) This normilization technique displays the raw data in terms of how many SDs it is above or below the mean. Here, for the z_age variable the range is (-1.77, 1.10) and for z_perc_fat we see a range of (-2.26, 1.48)

iii. Normalization by decimal scaling This normilization technique transorms the data by moving the decimal point. The range for age with this normilization is (0.23, 0.61) and perc_fat is (0.078, 0.425)

Problem 1-d From the scatter plot we can see a moderate relatioship betwen age and percent body fat. Since both of the varibales had more datapoints on the high end, the relationship is more clearly seen the greater the values are. There are one or two questionable data points that would need further exploration.

```
In [34]: df.plot.scatter('age', 'perc_fat')

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x219e64c8e80>
```



Problem 1-e From the correlation matrix and the covariance matrix below, we can see that age and percent body fat are positivley correlated since r-squared is positive and the covariance shows a positive increase.

```
In [39]: #Correlation
         df.corr()
```

```
correlation-matrix                age  perc_fat
age        1.000000  0.817619
perc_fat   0.817619  1.000000
```

```
In [36]: #Covariance
         df.cov()
```

```
Out[36]:                  age    perc_fat
         age       174.732026  100.019608
         perc_fat  100.019608   85.643824
```

Problem 2 (10 points): This problem is an example of data preprocessing needed in a data mining process. Suppose a group of 12 sales price records has been sorted as follows:

5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215

Partition them into bins by each of the following methods. Show which values are in which bins. Then smooth the data using the bins, and show the new set of smoothed values. Explain how each type of smoothing affect the data and the ways they are different.

a. (5 points) equal-depth partitioning with 3 values per bin

b. (5 points) equal-width partitioning with 3 bins

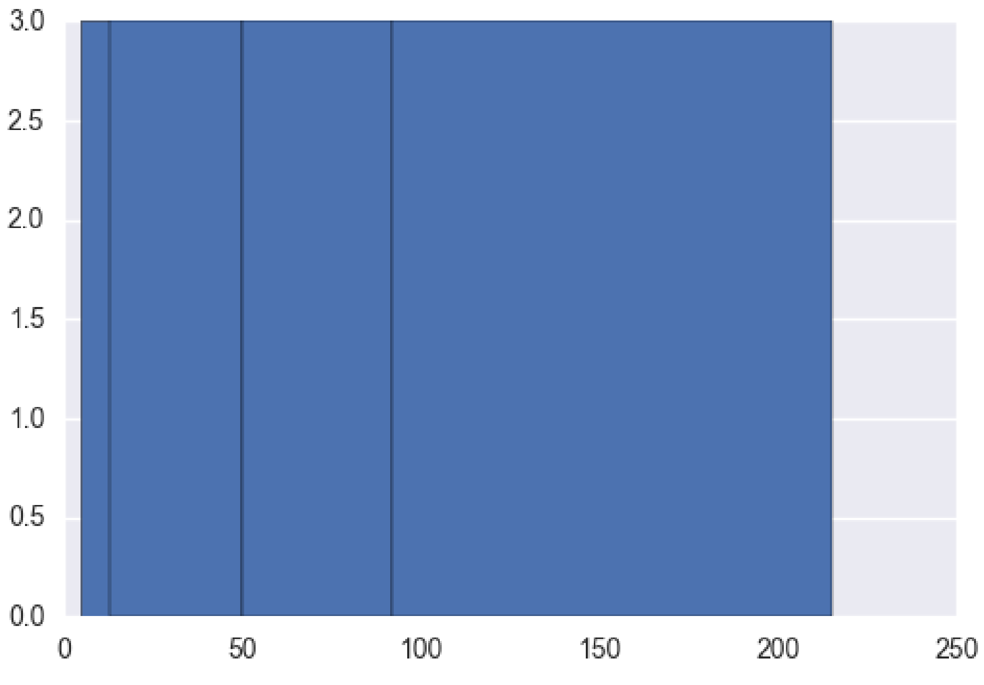
The biggest difference between these two methods is how they assign values to bins. Equal-depth is simply partitioning the data into equal chunks with each chunck containing the same amount of data points, independent of any range. Equal-width is partitioning the data, but each point falls into a defined range of values (based on (max-min)/n, and has no set number of data points that can be assigned to each bin.

Problem 2-a Equal-depth
Bin 1: 5, 10, 11
Bin 2: 13, 15, 35
Bin 3: 50, 55, 72
Bin 4: 92, 204, 215

Problem 2-b Equal-width
max = 215; min = 5; n = 3
bin width = 70
Bin 1 (5-75): 5, 10, 11, 12, 15, 35, 50, 72
Bin 2 (75-145): 92
Bin 3 (145-215): 204, 215

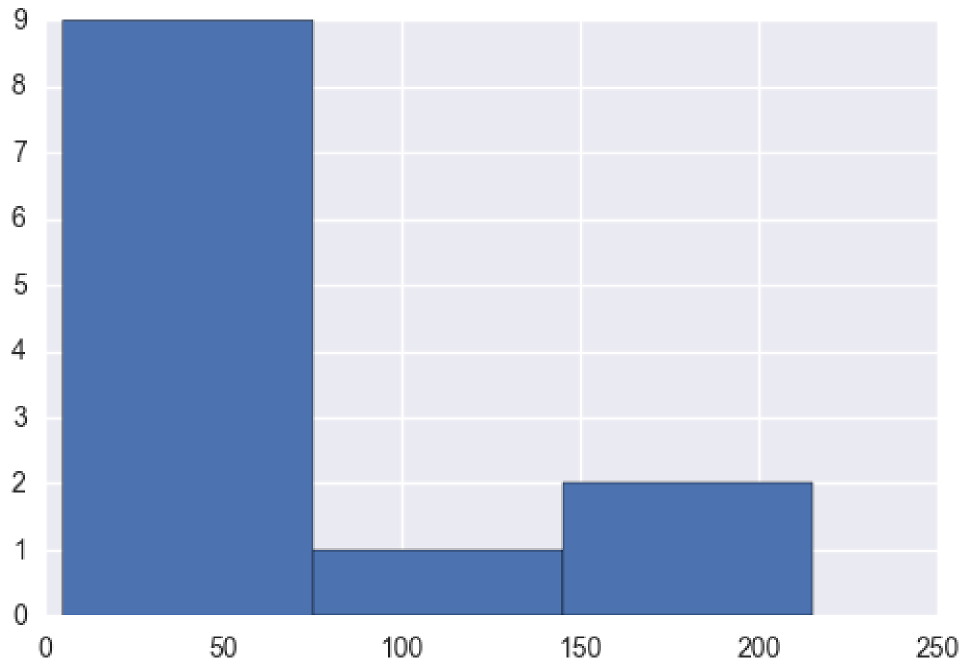
```
In [55]: #Equal-depth smoothing
         plt.hist([5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215], bins = [5, 13,
```

```
Out[55]: (array([ 3.,  3.,  3.,  3.]),
          array([  5,  13,  50,  92, 215]),
          <a list of 4 Patch objects>)
```



```
In [61]: #Equal-width smoothing
         plt.hist([5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215], bins = [5, 75,

Out[61]: (array([ 9.,  1.,  2.]),
          array([  5,  75, 145, 215]),
          <a list of 3 Patch objects>)
```



Problem 3 (10 points): Answer the following questions about the data cleaning and integration process:

a. (4 points) In real-world data, there are often rows that have missing values for some variables. Describe two methods for dealing with this problem.

b. (2 points) If we have class labels for our data, how can we use them to help get better estimates when filling in missing values?

c. (4 points) Describe two issues that may come up during data integration.

Probelm 3-a Method 1: Remove any rows from the dataset that contain missing data in one or more column(s). This method works better with larger datasets.

Method 2: Fill in missing values with the mean. Depending on the data, it might be appropriate to apply the mean in chunks if the feature experienced changes over time that inherently moved the overall mean.

Problem 3-b A classification algorithm could be used to "predict" the values for missing data based on the known data from the classes of the same type.

Problem 3-c Issue 1: Simply put, bad data. It could be bad for a number of reasons such as human dependency (free form text, manual input, etc) or the data collection process contains errors or is unreliable.

Issue 2: Redundant data can be an issue. Redundancy can resuly from having duplicate columns under different column names, or a columns that are derived from one or more columns in the same data set.

Bonus Problem (5 points): We discussed how a clustering of data can be used to smooth data, so let's consider if it could be used for repairing missing data. We discussed how class labels can be used to improve the process of filling in missing values (and you wrote about it in 3b), and we discussed how a clustering result can be used similarly to class labels. Can we cluster data and use the clustering to fill in missing values? If so, how? If not, what problem would we encounter?

Yes, we can use clustering to help fill in missing values. Assuming data could be clustered according to one or more elements, we could analyze the silarities among their elements and fill in the missing data according to something like the mean or the median of the known data in the cluster.