CSE572-Lab2-key

September 12, 2022

1 CSE 572: Lab 2

This lab contains a Data Exploration module and a practice exercise in which you will use some of the operations from the Data Exploration module.

To execute and make changes to this notebook, click File > Save a copy to save your own version in your Google Drive or Github. Read the step-by-step instructions below carefully. To execute the code, click on each cell below and press the SHIFT-ENTER keys simultaneously or by clicking the Play button.

When you finish executing all code/exercises, save your notebook then download a copy (.ipynb file). Submit 1) a link to your Colab notebook and 2) the .ipynb file on Canvas.

2 Data Exploration

The following tutorial contains examples of Python code for data exploration. You should refer to the "Data Exploration" chapter of the "Introduction to Data Mining" book (available at https://www-users.cs.umn.edu/~kumar001/dmbook/index.php) to understand some of the concepts introduced in this tutorial notebook.

Data exploration refers to the preliminary investigation of data in order to better understand its specific characteristics. There are two key motivations for data exploration: 1. To help users select the appropriate preprocessing and data analysis techniques to be used. 2. To make use of humans' abilities to recognize patterns in the data.

2.1 1. Summary Statistics

Summary statistics are quantities, such as the mean and standard deviation, that capture various characteristics of a potentially large set of values with a single number or a small set of numbers. In this tutorial, we will use the Iris sample data, which contains information on 150 Iris flowers, 50 each from one of three Iris species: Setosa, Versicolour, and Virginica. Each flower is characterized by five attributes:

- sepal length in centimeters
- sepal width in centimeters
- petal length in centimeters
- petal width in centimeters
- class (Setosa, Versicolour, Virginica)

In this tutorial, you will learn how to:

- Load a CSV data file into a Pandas DataFrame object.
- Compute various summary statistics from the DataFrame.

1. First, you need to download the Iris dataset from the UCI machine learning repository.

Code: The following code uses Pandas to read the CSV file and store them in a DataFrame object named data. Next, it will display the first five rows of the data frame.

[1]:
```
import pandas as pd

data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪iris/iris.data',header=None)
data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width',␣
 ↪'class']

data.head()
```

[1]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

How big is this dataset? Print the number of samples/records and attributes.

[2]:
```
print('There are {} samples and {} attributes in this dataset.'.format(data.
 ↪shape[0], data.shape[1]))
```

```
There are 150 samples and 5 attributes in this dataset.
```

2. For each quantitative attribute, calculate its average, standard deviation, minimum, and maximum values.

Code:

[3]:
```
from pandas.api.types import is_numeric_dtype

for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Standard deviation = %.2f' % data[col].std())
        print('\t Minimum = %.2f' % data[col].min())
        print('\t Maximum = %.2f' % data[col].max())
```

```
sepal length:
         Mean = 5.84
```

```
        Standard deviation = 0.83
        Minimum = 4.30
        Maximum = 7.90
sepal width:
        Mean = 3.05
        Standard deviation = 0.43
        Minimum = 2.00
        Maximum = 4.40
petal length:
        Mean = 3.76
        Standard deviation = 1.76
        Minimum = 1.00
        Maximum = 6.90
petal width:
        Mean = 1.20
        Standard deviation = 0.76
        Minimum = 0.10
        Maximum = 2.50
```

3. For the qualitative attribute (class), count the frequency for each of its distinct values.

Code:

[4]:
```
data['class'].value_counts()
```

[4]:
```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: class, dtype: int64
```

4. It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

Code:

[5]:
```
data.describe(include='all')
```

[5]:

	sepal length	sepal width	petal length	petal width	class
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN

```
max          7.900000      4.400000      6.900000      2.500000           NaN
```

Note that count refers to the number of non-missing values for each attribute.

5. For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

Code:

[6]:
```
print('Covariance:')
data.cov()
```

```
Covariance:
```

[6]:

	sepal length	sepal width	petal length	petal width
sepal length	0.685694	-0.039268	1.273682	0.516904
sepal width	-0.039268	0.188004	-0.321713	-0.117981
petal length	1.273682	-0.321713	3.113179	1.296387
petal width	0.516904	-0.117981	1.296387	0.582414

[7]:
```
print('Correlation:')
data.corr()
```

```
Correlation:
```

[7]:

	sepal length	sepal width	petal length	petal width
sepal length	1.000000	-0.109369	0.871754	0.817954
sepal width	-0.109369	1.000000	-0.420516	-0.356544
petal length	0.871754	-0.420516	1.000000	0.962757
petal width	0.817954	-0.356544	0.962757	1.000000

Question 1: Which two features have the strongest correlation? (ignore the diagonals, which show each feature's correlation with itself)

Answer:

The strongest correlation is between petal width and petal length (0.96 correlation coefficient)

2.2 2. Data Visualization

Data visualization is the display of information in a graphic or tabular format. Successful visualization requires that the data (information) be converted into a visual format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported.

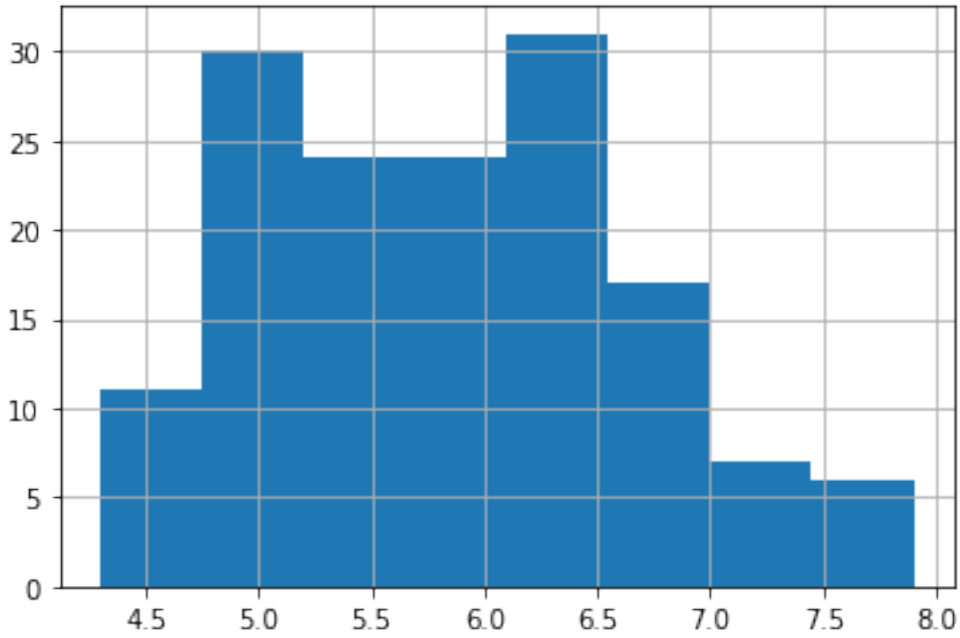
In this tutorial, you will learn how to display the Iris data created in Section 3.1.

1. First, we will display the histogram for the sepal length attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

Code:

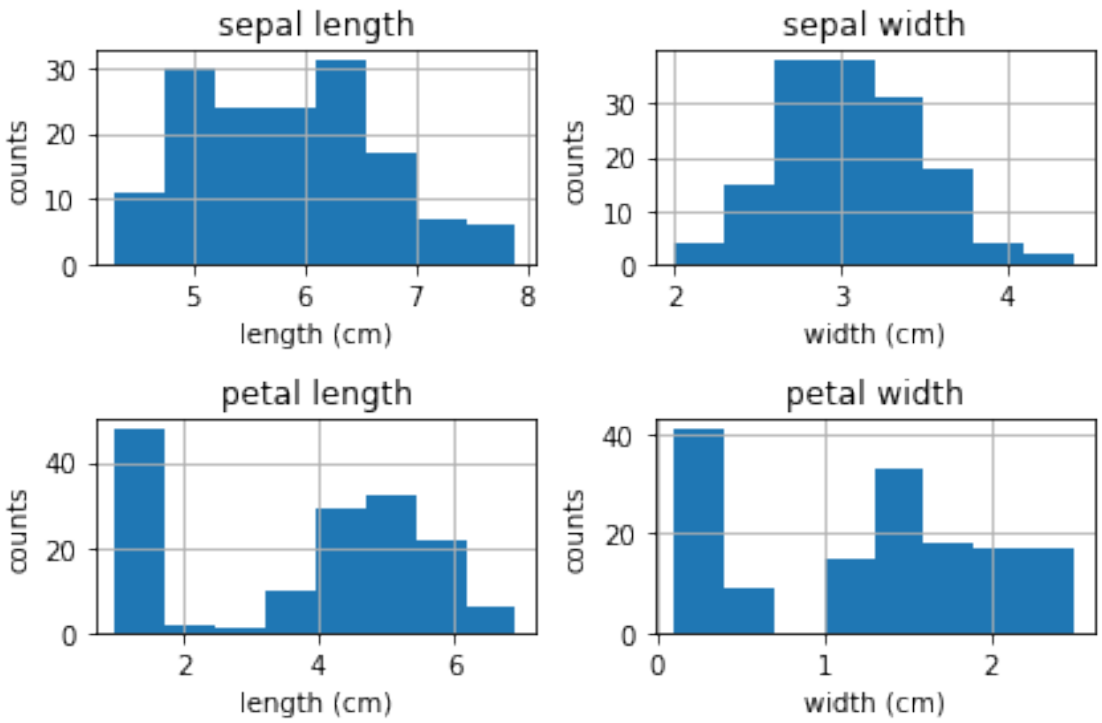
[8]:
```
%matplotlib inline

data['sepal length'].hist(bins=8)
```

[8]: <AxesSubplot:>



Plot a histogram of each of the features in a single plot.

[9]:
```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(nrows=2, ncols=2)

for i, col in enumerate(data.columns[:4]):
    data[col].hist(bins=8, ax=axes.flat[i])
    axes.flat[i].set_title(col)
    if 'length' in col:
        axes.flat[i].set_xlabel('length (cm)')
    elif 'width' in col:
        axes.flat[i].set_xlabel('width (cm)')
    axes.flat[i].set_ylabel('counts')

fig.tight_layout()
```



Question 2: What do these histograms tells us about the distribution of the values of each feature in our dataset? Are they uniformly distributed, Gaussian-distributed, or other? Do they appear to have one mode, or multiple? Describe the distribution of each of the features.

Answer:

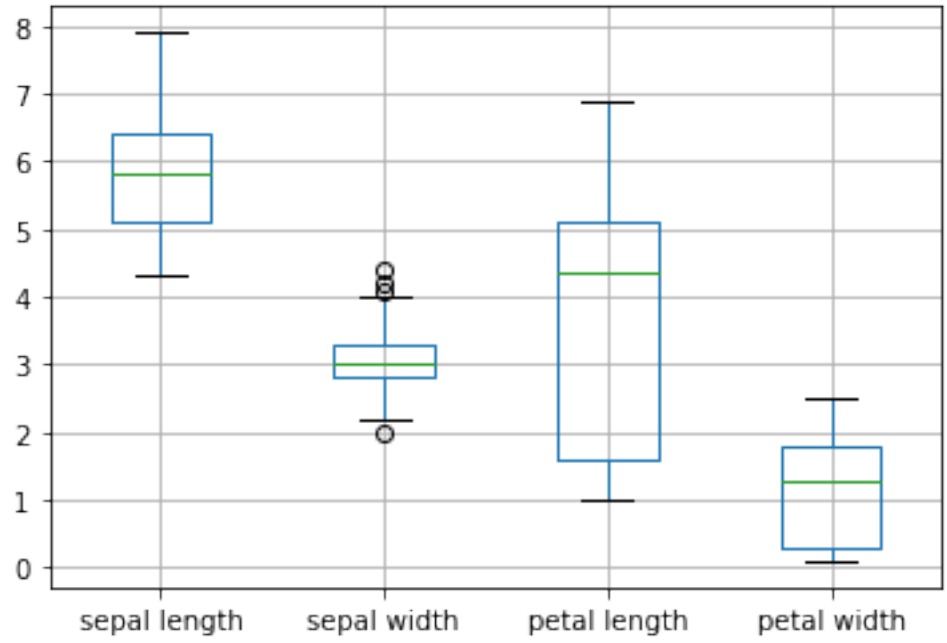
The distributions are mostly Gaussian distributed, however we see two distinct modes in petal length and petal width, and two modes that are nearby but possibly distinct in sepal length.

2. A boxplot can also be used to show the distribution of values for each attribute.

Code:

[10]:
```
data.boxplot()
```

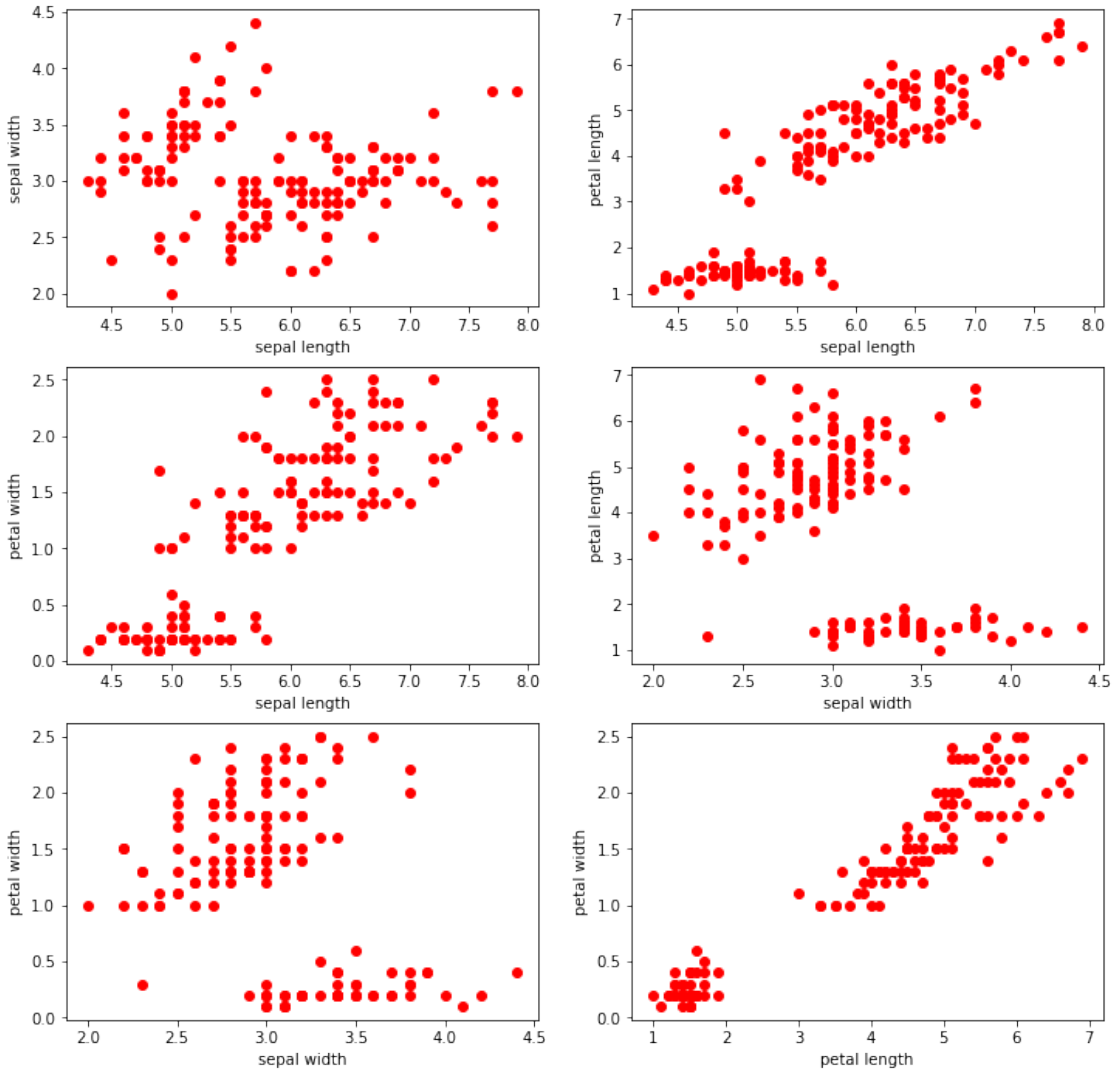
[10]:
```
<AxesSubplot:>
```



3. For each pair of attributes, we can use a scatter plot to visualize their joint distribution.

Code:

[11]:
```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(data[data.columns[i]], data[data.columns[j]],␣
↪color='red')
        axes[ax1][ax2].set_xlabel(data.columns[i])
        axes[ax1][ax2].set_ylabel(data.columns[j])
        index = index + 1
```

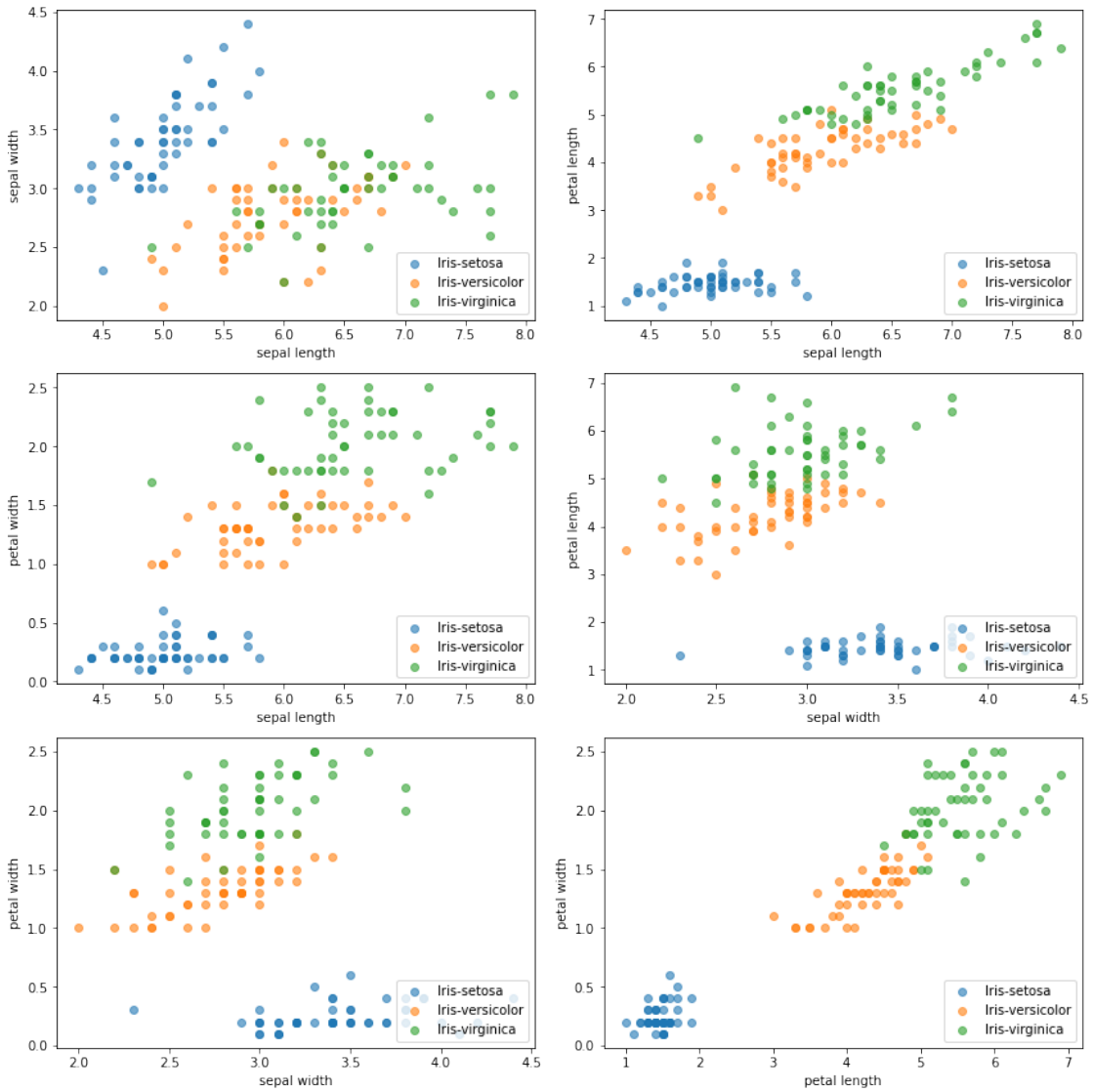


4. If we color the scatter plot points by the class value for each sample, we can see if the classes are clustered in any of the attribute pairs.

Code:

[12]:
```
fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        for iris in data['class'].unique():
            axes[ax1][ax2].scatter(data[data['class'] == iris][data.columns[i]],
                                   data[data['class'] == iris][data.columns[j]],
                                   label=iris,
```

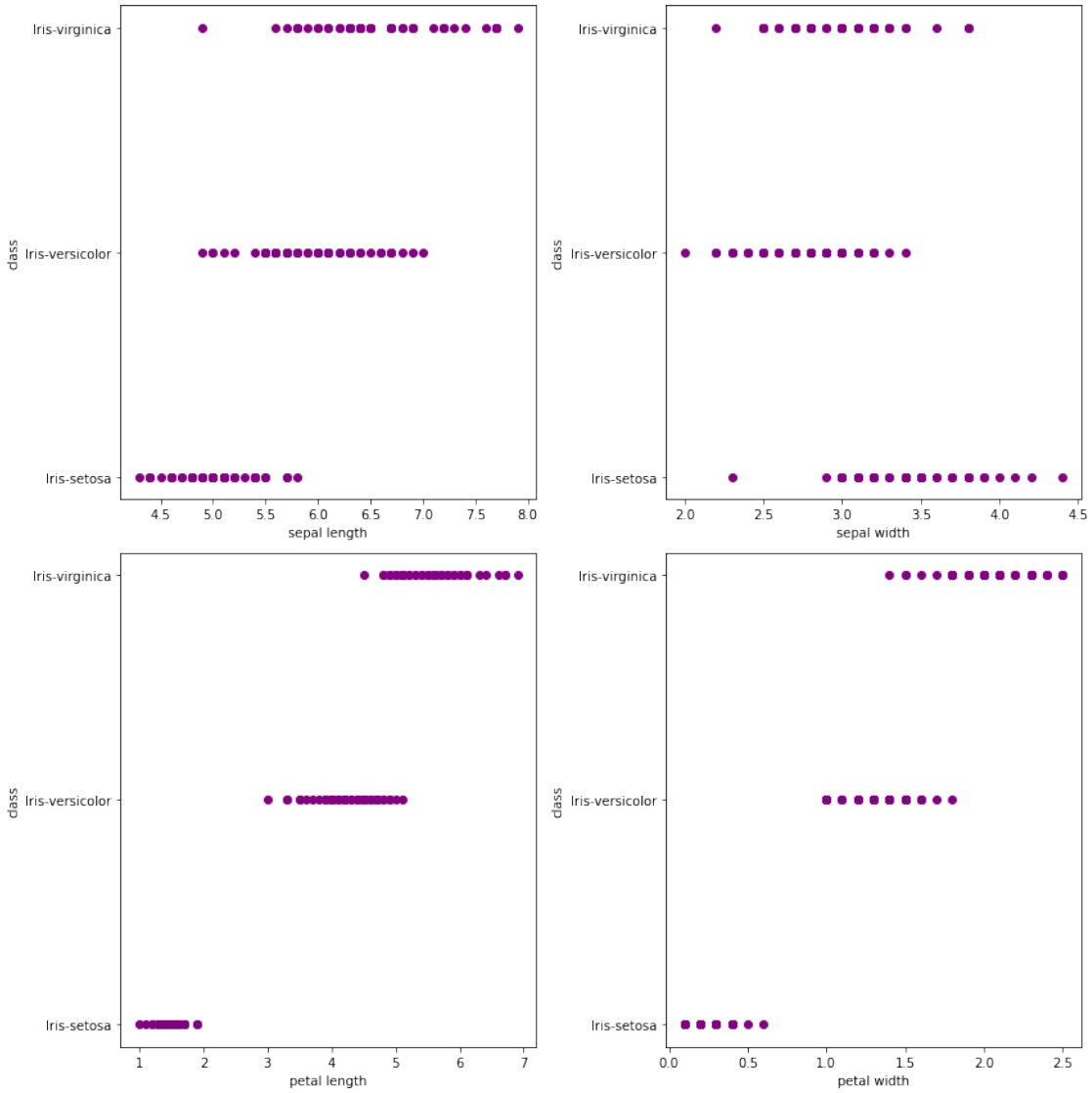
```
                                    alpha=0.6)
        axes[ax1][ax2].set_xlabel(data.columns[i])
        axes[ax1][ax2].set_ylabel(data.columns[j])
        axes[ax1][ax2].legend(loc='lower right')
        index = index + 1

fig.tight_layout()
```



5. To see which features are the best and weakest predictors of the iris class, we can visualize the feature value and class value as a scatter plot.

Code:

[13]:
```
fig, axes = plt.subplots(2, 2, figsize=(12,12))
index = 0
for col_name in data.columns[:-1]:
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(data[col_name], data['class'], color='purple')
        axes[ax1][ax2].set_xlabel(col_name)
        axes[ax1][ax2].set_ylabel('class')
        index = index + 1
fig.tight_layout()
```



Question 3: Study these plots and review your responses to Questions 1 and 2. Which features are the best predictors of the iris class? The weakest predictors? Which

combination of two features separates the classes best?

Answer:

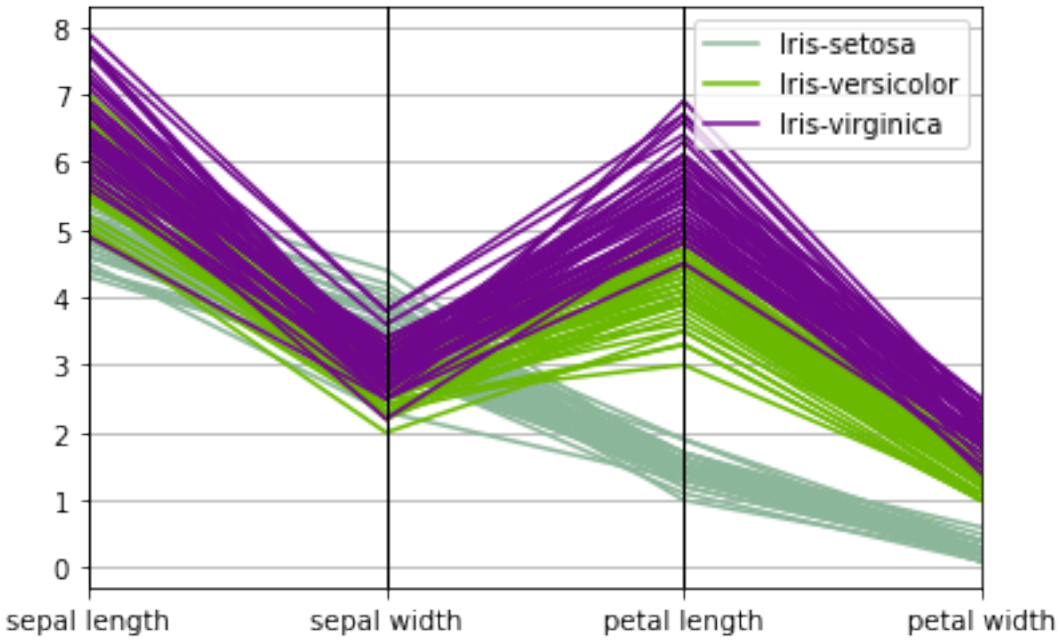
Petal length or petal width are the best predictors of the iris class. Combined, these two features show the strongest clustering and best separation of clusters of the iris classes, as illustrated in the joint distribution scatter plots colored by class value.

6. Parallel coordinates can be used to display all the data points simultaneously. Parallel coordinates have one coordinate axis for each attribute, but the different axes are parallel to one other instead of perpendicular, as is traditional. Furthermore, an object is represented as a line instead of as a point. In the example below, the distribution of values for each class can be identified in a separate color.

Code:

[14]:
```
from pandas.plotting import parallel_coordinates
%matplotlib inline

parallel_coordinates(data, 'class')
```

[14]: <AxesSubplot:>



Question 4: Which of the classes are more similar to each other in terms of their attribute distributions (i.e., their lines overlap most)? If you had to come up with a set of rules for distinguishing between iris-setosa, iris-versicolor, and iris-virginica based on their attribute values, what are some rules you might propose?

Answer:

The iris-virginica and iris-versicolor lines have the most overlap and thus the classes are most similar to each other. Some rules for separating them could include: - setosa if petal length < 2.5, virginica if petal length > 4.8, and versicolor otherwise - setosa if petal width < 1, virginica if petal width > 1.4, and versicolor otherwise

Grading note: count as correct if the rule makes sense, does not have to be exactly the rules I suggested above.

2.3 4. Data Quality Issues

Poor data quality can have an adverse effect on data mining. Among the common data quality issues include noise, outliers, missing values, and duplicate data. This section presents examples of Python code to alleviate some of these data quality problems. We begin with an example dataset from the UCI machine learning repository containing information about breast cancer patients. We will first download the dataset using Pandas read_csv() function and display its first 5 data points.

Code:

Question 5: View the documentation for the Wisconsin Breast Cancer dataset here. Who collected this dataset? When did they collect it?

Answer:

The dataset was collected between 1989 and 1991. The database was donated in 1992.

[15]:
```
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
data.columns = ['Sample code', 'Clump Thickness', 'Uniformity of Cell Size',␣
 ↪'Uniformity of Cell Shape',
                'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare␣
 ↪Nuclei', 'Bland Chromatin',
                'Normal Nucleoli', 'Mitoses','Class']

data = data.drop(['Sample code'],axis=1)
print('Number of instances = %d' % (data.shape[0]))
print('Number of attributes = %d' % (data.shape[1]))
data.head()
```

```
Number of instances = 699
Number of attributes = 10
```

[15]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
0	5	1	1	
1	5	4	4	
2	3	1	1	
3	6	8	8	
4	4	1	1	

	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	\
0	1	2	1	

```
1                    5                            7           10
2                    1                            2            2
3                    1                            3            4
4                    3                            2            1
```

```
   Bland Chromatin  Normal Nucleoli  Mitoses  Class
0                3                1        1      2
1                3                2        1      2
2                3                1        1      2
3                3                7        1      2
4                3                1        1      2
```

Print the value counts for the Class attribute.

[16]:
```
data['Class'].value_counts()
```

[16]:
```
2    458
4    241
Name: Class, dtype: int64
```

Question 6: What does a Class value of 2 indicate? What does 4 indicate?

Answer:

2 = benign, 4 = malignant

2.3.1 4.1.1 Missing Values

It is not unusual for an object to be missing one or more attribute values. In some cases, the information was not collected; while in other cases, some attributes are inapplicable to the data instances. This section presents examples on the different approaches for handling missing values.

According to the description of the data (https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(origi the missing values are encoded as '?' in the original data. Our first task is to convert the missing values to NaNs. We can then count the number of missing values in each column of the data.

Code:

[17]:
```
import numpy as np

data = data.replace('?',np.NaN)

print('Number of instances = %d' % (data.shape[0]))
print('Number of attributes = %d' % (data.shape[1]))

print('Number of missing values:')
for col in data.columns:
    print('\t%s: %d' % (col,data[col].isna().sum()))
```

```
Number of instances = 699
Number of attributes = 10
```

```
Number of missing values:
        Clump Thickness: 0
        Uniformity of Cell Size: 0
        Uniformity of Cell Shape: 0
        Marginal Adhesion: 0
        Single Epithelial Cell Size: 0
        Bare Nuclei: 16
        Bland Chromatin: 0
        Normal Nucleoli: 0
        Mitoses: 0
        Class: 0
```

Observe that only the 'Bare Nuclei' column contains missing values. In the following example, the missing values in the 'Bare Nuclei' column are replaced by the median value of that column. The values before and after replacement are shown for a subset of the data points.

Code:

[18]:
```
data2 = data['Bare Nuclei']

print('Before replacing missing values:')
print(data2[20:25])
data2 = data2.fillna(data2.median())

print('\nAfter replacing missing values:')
print(data2[20:25])
```

```
Before replacing missing values:
20     10
21      7
22      1
23    NaN
24      1
Name: Bare Nuclei, dtype: object

After replacing missing values:
20    10
21     7
22     1
23     1
24     1
Name: Bare Nuclei, dtype: object
```

Instead of replacing the missing values, another common approach is to discard the data points that contain missing values. This can be easily accomplished by applying the dropna() function to the data frame.

Code:

[19]:
```
print('Number of rows in original data = %d' % (data.shape[0]))

data2 = data.dropna()
print('Number of rows after discarding missing values = %d' % (data2.shape[0]))
```

```
Number of rows in original data = 699
Number of rows after discarding missing values = 683
```

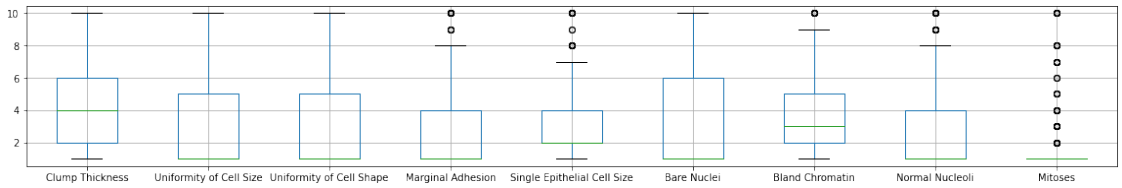
2.3.2 4.1.2 Outliers

Outliers are data instances with characteristics that are considerably different from the rest of the dataset. In the example code below, we will draw a boxplot to identify the columns in the table that contain outliers. Note that the values in all columns (except for 'Bare Nuclei') are originally stored as 'int64' whereas the values in the 'Bare Nuclei' column are stored as string objects (since the column initially contains strings such as '?' for representing missing values). Thus, we must convert the column into numeric values first before creating the boxplot. Otherwise, the column will not be displayed when drawing the boxplot.

Code:

[20]:
```
%matplotlib inline

data2 = data.drop(['Class'],axis=1)
data2['Bare Nuclei'] = pd.to_numeric(data2['Bare Nuclei'])
data2.boxplot(figsize=(20,3))
```

[20]: <AxesSubplot:>



The boxplots suggest that only 5 of the columns (Marginal Adhesion, Single Epithetial Cell Size, Bland Cromatin, Normal Nucleoli, and Mitoses) contain abnormally high values. To discard the outliers, we can compute the Z-score for each attribute and remove those instances containing attributes with abnormally high or low Z-score (e.g., if Z > 3 or Z <= -3).

Code:

The following code shows the results of standardizing the columns of the data. To standardize variables, you calculate the mean and standard deviation for a variable. Then, for each observed value of the variable, you subtract the mean and divide by the standard deviation. This results in standard scores that represent the number of standard deviations above or below the mean that a specific observation falls. For instance, a standardized value of 2 indicates that the observation falls 2 standard deviations above the mean.

Note that missing values (NaN) are not affected by the standardization process.

[21]:
```
Z = (data2-data2.mean())/data2.std()
Z[20:25]
```

[21]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
20	0.917080	-0.044070	-0.406284	
21	1.982519	0.611354	0.603167	
22	-0.503505	-0.699494	-0.742767	
23	1.272227	0.283642	0.603167	
24	-1.213798	-0.699494	-0.742767	

	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	\
20	2.519152	0.805662	1.771569	
21	0.067638	1.257272	0.948266	
22	-0.632794	-0.549168	-0.698341	
23	-0.632794	-0.549168	NaN	
24	-0.632794	-0.549168	-0.698341	

	Bland Chromatin	Normal Nucleoli	Mitoses
20	0.640688	0.371049	1.405526
21	1.460910	2.335921	-0.343666
22	-0.589645	-0.611387	-0.343666
23	1.460910	0.043570	-0.343666
24	-0.179534	-0.611387	-0.343666

Code:

The following code shows the results of discarding columns with Z > 3 or Z <= -3.

[22]:
```
print('Number of rows before discarding outliers = %d' % (Z.shape[0]))

Z2 = Z.loc[((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9),:]
print('Number of rows after discarding outliers = %d' % (Z2.shape[0]))
```

```
Number of rows before discarding outliers = 699
Number of rows after discarding outliers = 632
```

2.3.3 4.1.3 Duplicate Data

Some datasets, especially those obtained by merging multiple data sources, may contain duplicates or near duplicate instances. The term deduplication is often used to refer to the process of dealing with duplicate data issues.

Code:

In the following example, we first check for duplicate instances in the breast cancer dataset.

[23]:
```
dups = data.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
data[dups]
```

```
Number of duplicate rows = 236
```

[23]:

```
     Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
28                 2                        1                         1
35                 2                        1                         1
48                 4                        1                         1
64                 1                        1                         1
66                 4                        1                         1
..               ...                      ...                       ...
686                1                        1                         1
688                4                        1                         1
690                1                        1                         1
692                3                        1                         1
695                2                        1                         1

     Marginal Adhesion  Single Epithelial Cell Size Bare Nuclei  \
28                   1                            2           1
35                   1                            2           1
48                   3                            2           1
64                   1                            2           1
66                   1                            2           1
..                 ...                          ...         ...
686                  1                            2           1
688                  1                            2           1
690                  3                            2           1
692                  1                            2           1
695                  1                            2           1

     Bland Chromatin  Normal Nucleoli  Mitoses  Class
28                 2                1        1      2
35                 2                1        1      2
48                 3                1        1      2
64                 2                1        1      2
66                 3                1        1      2
..               ...              ...      ...    ...
686                1                1        1      2
688                1                1        1      2
690                1                1        1      2
692                1                1        1      2
695                1                1        1      2

[236 rows x 10 columns]
```

The duplicated() function will return a Boolean array that indicates whether each row is a duplicate of a previous row in the table:

[24]:
```
dups
```

[24]:
```
0      False
1      False
2      False
3      False
4      False
       …
694    False
695     True
696    False
697    False
698    False
Length: 699, dtype: bool
```

Although such duplicate rows may correspond to samples for different individuals, in this hypothetical example, we assume that the duplicates are samples taken from the same individual and illustrate below how to remove the duplicated rows.

[25]:
```
print('Number of rows before discarding duplicates = %d' % (data.shape[0]))
data2 = data.drop_duplicates()
print('Number of rows after discarding duplicates = %d' % (data2.shape[0]))
```

```
Number of rows before discarding duplicates = 699
Number of rows after discarding duplicates = 463
```

Now, combine all of these pre-processing steps to clean the dataset by 1) dropping the rows that have NaN values, 2) removing outliers with Z score Z > 3 or Z <= -3, and 3) dropping the duplicate rows.

[26]:
```
# rows remaining after dropping nans
data_clean = data.dropna()
data_clean.shape
```

[26]:
```
(683, 10)
```

[27]:
```
# rows remaining after dropping duplicates
data_clean = data_clean.drop_duplicates()
data_clean.shape
```

[27]:
```
(449, 10)
```

[28]:
```
# rows remaining after dropping outliers
data_clean = data_clean.drop(['Class'],axis=1)
data_clean['Bare Nuclei'] = pd.to_numeric(data_clean['Bare Nuclei'])

Z = (data_clean-data_clean.mean())/data_clean.std()
data_clean = Z.loc[((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9),:]
```

```
data_clean.shape
```

[28]: (435, 9)

Question 7: How many rows/samples are left after applying all three steps?

Answer:

435

Question 7 clarification:

There has been confusion about the order of operations to filter the dataset. In our dataset, there are N rows that have missing values, M rows that have outliers, and K rows that are duplicates. If you apply these filtering operations to the original dataset, the order does not matter. You do not need to do the operations in the same order they are listed. The numbers are meant to show that you must do all 3 steps, not prescribe the order (apologies for the confusion!).

That said, if you transform the dataset (e.g., standardized Z scores) then you have changed the values of the original dataset and thus you will not get the same results if you try to remove duplicates from the standardized (Z) dataframe. This does not mean you have fewer duplicates in the dataset—it just means they are now hidden. This is illustrated in the below code examples.

[29]:
```
# Load the original dataset
import pandas as pd
import numpy as np

data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None)
data.columns = ['Sample code', 'Clump Thickness', 'Uniformity of Cell Size',␣
 ↪'Uniformity of Cell Shape',
                'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare␣
 ↪Nuclei', 'Bland Chromatin',
                'Normal Nucleoli', 'Mitoses','Class']

data = data.drop(['Sample code'],axis=1)

data = data.drop(['Class'],axis=1)
data = data.replace('?',np.NaN)
data['Bare Nuclei'] = pd.to_numeric(data['Bare Nuclei'])

data
```

[29]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
0	5	1	1	
1	5	4	4	
2	3	1	1	
3	6	8	8	
4	4	1	1	
..	…	…	…	

```
694                3                            1                            1
695                2                            1                            1
696                5                           10                           10
697                4                            8                            6
698                4                            8                            8

     Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  \
0                    1                            2          1.0
1                    5                            7         10.0
2                    1                            2          2.0
3                    1                            3          4.0
4                    3                            2          1.0
..                 …                            …          …
694                  1                            3          2.0
695                  1                            2          1.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               …                …      …
694                1                1        1
695                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[699 rows x 9 columns]
```

If you apply these filtering operations to the original dataset and identify which indices to drop based on the indices that meet at least 1 of the 3 criteria for dropping then the order does not matter. See an example of this below.

[30]:
```python
def inds_nans(df):
    inds = df.isna().any(axis=1)
    print('Found {} rows that had NaN values.'.format(inds.sum()))
    return inds

def inds_dups(df):
    inds = df.duplicated()
    print('Found {} rows that were duplicates.'.format(inds.sum()))
    return inds
```

```
def inds_outliers(df):
    # In this example, we defined outliers as values that are +/- 3 standard␣
 ↪deviations
    # from the mean value. To identify such values, we need to compute the Z␣
 ↪score for
    # every value by subtracting the feature-wise mean and dividing by the␣
 ↪feature-wise
    # standard deviation (also known as standardizing the data).
    Z = (df-df.mean())/df.std()
    # The below code will give a value of True or False for each row. The row␣
 ↪will be
    # True if all of the feature values for that row were within 3 standard␣
 ↪deviations of
    # the mean. The row will be False if at leaset one of the feature values␣
 ↪for that row
    # was NOT within 3 standard deviations of the mean.
    inlier_inds = ((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9)
    # The outliers are the inverse boolean values of the above
    outlier_inds = ~inlier_inds
    print('Found {} rows that were outliers.'.format(outlier_inds.sum()))
    return outlier_inds
```

[31]:
```
data_clean = data.loc[~((inds_nans(data) | inds_dups(data)) |␣
 ↪inds_outliers(data)),:]

data_clean
```

```
Found 16 rows that had NaN values.
Found 236 rows that were duplicates.
Found 67 rows that were outliers.
```

[31]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
0	5	1	1	
1	5	4	4	
2	3	1	1	
3	6	8	8	
4	4	1	1	
..	…	…	…	
693	3	1	1	
694	3	1	1	
696	5	10	10	
697	4	8	6	
698	4	8	8	

	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	\
0	1	2	1.0	

```
1                    5                                7         10.0
2                    1                                2          2.0
3                    1                                3          4.0
4                    3                                2          1.0
..                 …                              …          …
693                  1                                2          1.0
694                  1                                3          2.0
696                  3                                7          3.0
697                  4                                3          4.0
698                  5                                4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               …                …      …
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[399 rows x 9 columns]
```

However, if you chain the operations, the order does matter. See below.

[32]:
```
def drop_nans(df):
    newdf = df.dropna()
    print('Dropped {} rows that had NaN values.'.format((df.shape[0]-newdf.
 ↪shape[0])))
    return newdf

def drop_dups(df):
    newdf = df.drop_duplicates()
    print('Dropped {} rows that were duplicates.'.format((df.shape[0]-newdf.
 ↪shape[0])))
    return newdf

def drop_outliers(df):
    # In this example, we defined outliers as values that are +/- 3 standard␣
 ↪deviations
    # from the mean value. To identify such values, we need to compute the Z␣
 ↪score for
    # every value by subtracting the feature-wise mean and dividing by the␣
 ↪feature-wise
```

```
    # standard deviation (also known as standardizing the data).
    Z = (df-df.mean())/df.std()
    # The below code will give a value of True or False for each row. The row␣
↪will be
    # True if all of the feature values for that row were within 3 standard␣
↪deviations of
    # the mean. The row will be False if at leaset one of the feature values␣
↪for that row
    # was NOT within 3 standard deviations of the mean.
    inlier_inds = ((Z > -3).sum(axis=1)==9) & ((Z <= 3).sum(axis=1)==9)
    # The below code will only retain the rows for which the above operation␣
↪returned True
    # Thus only the inliers are stored in newdf
    newdf = df.loc[inlier_inds,:]
    print('Dropped {} rows that were outliers.'.format((df.shape[0]-newdf.
↪shape[0])))
    return newdf
```

[33]:
```
# Drop nans, duplicates, then outliers
data_clean1 = drop_nans(data)
data_clean1 = drop_dups(data_clean1)
data_clean1 = drop_outliers(data_clean1)

data_clean1
```

```
Dropped 16 rows that had NaN values.
Dropped 234 rows that were duplicates.
Dropped 14 rows that were outliers.
```

[33]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
0	5	1	1	
1	5	4	4	
2	3	1	1	
3	6	8	8	
4	4	1	1	
..	…	…	…	
693	3	1	1	
694	3	1	1	
696	5	10	10	
697	4	8	6	
698	4	8	8	

	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	\
0	1	2	1.0	
1	5	7	10.0	
2	1	2	2.0	
3	1	3	4.0	

```
4                    3                            2          1.0
..                 …                          …          …
693                  1                            2          1.0
694                  1                            3          2.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               …                …      …
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[435 rows x 9 columns]
```

[34]:
```
# Drop nans, outliers, then duplicates
data_clean2 = drop_nans(data)
data_clean2 = drop_outliers(data_clean2)
data_clean2 = drop_dups(data_clean2)

data_clean2
```

```
Dropped 16 rows that had NaN values.
Dropped 51 rows that were outliers.
Dropped 233 rows that were duplicates.
```

[34]:
```
     Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
0                  5                        1                         1
1                  5                        4                         4
2                  3                        1                         1
3                  6                        8                         8
4                  4                        1                         1
..               …                      …                       …
693                3                        1                         1
694                3                        1                         1
696                5                       10                        10
697                4                        8                         6
698                4                        8                         8
```

```
     Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  \
0                    1                            2          1.0
1                    5                            7         10.0
2                    1                            2          2.0
3                    1                            3          4.0
4                    3                            2          1.0
..                 ...                          ...          ...
693                  1                            2          1.0
694                  1                            3          2.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               ...              ...      ...
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[399 rows x 9 columns]
```

[35]:
```
# Drop outliers, nans, then duplicates
data_clean3 = drop_outliers(data)
data_clean3 = drop_nans(data_clean3)
data_clean3 = drop_dups(data_clean3)

data_clean3
```

```
Dropped 67 rows that were outliers.
Dropped 0 rows that had NaN values.
Dropped 233 rows that were duplicates.
```

[35]:
```
     Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
0                  5                        1                         1
1                  5                        4                         4
2                  3                        1                         1
3                  6                        8                         8
4                  4                        1                         1
..               ...                      ...                       ...
693                3                        1                         1
```

```
694                3                           1                           1
696                5                          10                          10
697                4                           8                           6
698                4                           8                           8

     Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  \
0                    1                            2          1.0
1                    5                            7         10.0
2                    1                            2          2.0
3                    1                            3          4.0
4                    3                            2          1.0
..                 …                            …          …
693                  1                            2          1.0
694                  1                            3          2.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               …                …      …
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[399 rows x 9 columns]
```

[36]:
```
# Drop outliers, duplicates, then nans
data_clean4 = drop_outliers(data)
data_clean4 = drop_dups(data_clean4)
data_clean4 = drop_nans(data_clean4)

data_clean4
```

```
Dropped 67 rows that were outliers.
Dropped 233 rows that were duplicates.
Dropped 0 rows that had NaN values.
```

[36]:
```
     Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
0                  5                        1                         1
1                  5                        4                         4
```

```
2                  3                            1                            1
3                  6                            8                            8
4                  4                            1                            1
..               …                            …                            …
693                3                            1                            1
694                3                            1                            1
696                5                           10                           10
697                4                            8                            6
698                4                            8                            8

     Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  \
0                    1                            2          1.0
1                    5                            7         10.0
2                    1                            2          2.0
3                    1                            3          4.0
4                    3                            2          1.0
..                 …                            …          …
693                  1                            2          1.0
694                  1                            3          2.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               …                …      …
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[399 rows x 9 columns]
```

[37]:
```
# Drop duplicates, outliers, then nans
data_clean5 = drop_dups(data)
data_clean5 = drop_outliers(data_clean5)
data_clean5 = drop_nans(data_clean5)

data_clean5
```

```
Dropped 236 rows that were duplicates.
Dropped 28 rows that were outliers.
```

```
Dropped 0 rows that had NaN values.
```

[37]:
```
     Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
0                  5                        1                         1
1                  5                        4                         4
2                  3                        1                         1
3                  6                        8                         8
4                  4                        1                         1
..               ...                      ...                       ...
693                3                        1                         1
694                3                        1                         1
696                5                       10                        10
697                4                        8                         6
698                4                        8                         8

     Marginal Adhesion  Single Epithelial Cell Size  Bare Nuclei  \
0                    1                            2          1.0
1                    5                            7         10.0
2                    1                            2          2.0
3                    1                            3          4.0
4                    3                            2          1.0
..                 ...                          ...          ...
693                  1                            2          1.0
694                  1                            3          2.0
696                  3                            7          3.0
697                  4                            3          4.0
698                  5                            4          5.0

     Bland Chromatin  Normal Nucleoli  Mitoses
0                  3                1        1
1                  3                2        1
2                  3                1        1
3                  3                7        1
4                  3                1        1
..               ...              ...      ...
693                2                1        2
694                1                1        1
696                8               10        2
697               10                6        1
698               10                4        1

[435 rows x 9 columns]
```

[38]:
```
# Drop duplicates, nans, then outliers
data_clean6 = drop_dups(data)
data_clean6 = drop_nans(data_clean6)
data_clean6 = drop_outliers(data_clean6)
```

```
data_clean6
```

```
Dropped 236 rows that were duplicates.
Dropped 14 rows that had NaN values.
Dropped 14 rows that were outliers.
```

[38]:

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	\
0	5	1	1	
1	5	4	4	
2	3	1	1	
3	6	8	8	
4	4	1	1	
..	…	…	…	
693	3	1	1	
694	3	1	1	
696	5	10	10	
697	4	8	6	
698	4	8	8	

	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	\
0	1	2	1.0	
1	5	7	10.0	
2	1	2	2.0	
3	1	3	4.0	
4	3	2	1.0	
..	…	…	…	
693	1	2	1.0	
694	1	3	2.0	
696	3	7	3.0	
697	4	3	4.0	
698	5	4	5.0	

	Bland Chromatin	Normal Nucleoli	Mitoses
0	3	1	1
1	3	2	1
2	3	1	1
3	3	7	1
4	3	1	1
..	…	…	…
693	2	1	2
694	1	1	1
696	8	10	2
697	10	6	1
698	10	4	1

```
[435 rows x 9 columns]
```