Python Exploratory Data Analysis Cheat Sheet

LearnPython
• com

Load and Clean Your Data

Processing your data properly is an important first step, the approach here will depend on the type of data you have.

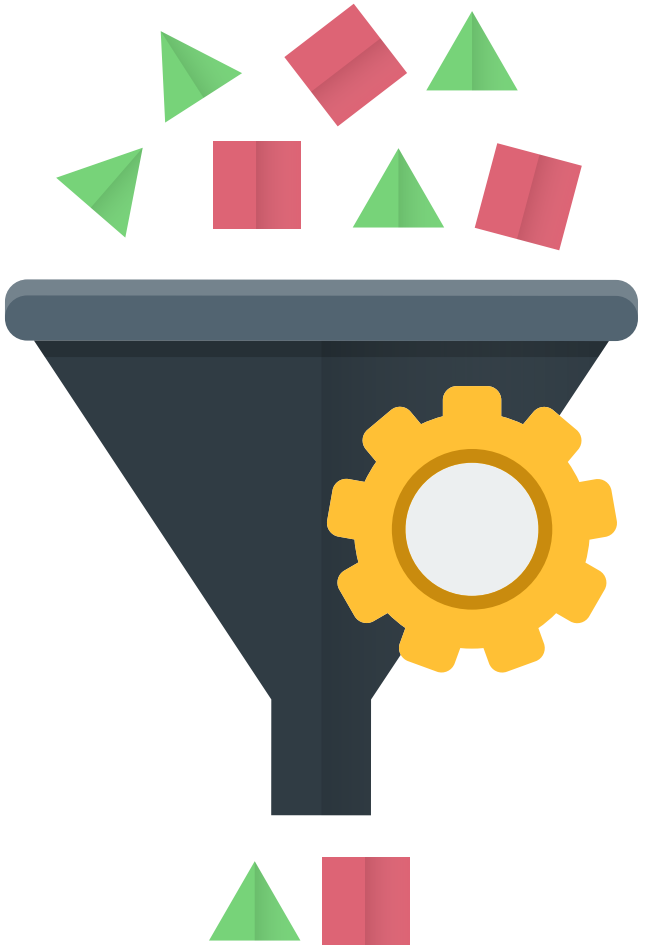
If you have data in CSV format, you can use Python's csv module to read it in. The article A GUIDE TO THE PYTHON CSV MODULE has more information and examples to help you out here. If your data is in an Excel spreadsheet, you'll need different libraries, which we discuss in BEST PYTHON PACKAGES FOR EXCEL.

Or perhaps you have data in the JSON format. In this case, you can use the json module to read it in. We have some useful examples in HOW TO CONVERT JSON TO CSV IN PYTHON. In that article, we also show you how the pandas library can make your life much easier when reading in data.

Part 1: Load Data and Remove Duplicates

Speaking of pandas, this library will come in handy for most parts of the EDA process. Start by importing your data into a pandas DataFrame called df.

To remove duplicate entries, use the df.drop_duplicates() function. The subset argument allows you to provide one or more column names to consider when dropping duplicates. The keep argument allows you to specify if you want to keep the first duplicated entry, the last, or none of them.



Part 2: Deal with Missing Data

When you're working with real data, it's common to have missing values. You can fill these with the df.fillna() method. The first argument specifies the value to use to fill in the missing values. With the second argument, you can choose to propagate the last or next valid observation forward or backwards.

See the article THE MOST HELPFUL PYTHON DATA CLEANING MODULES for more information and tips for cleaning data. Since this step in the EDA process involves manipulating data, here are 12 PYTHON TIPS AND TRICKS THAT EVERY DATA SCIENTIST SHOULD KNOW.

For the rest of this article, we'll be working with the famous IRIS FLOWER DATASET. It, along with many other interesting datasets, can be IMPORTED FROM SCIKIT-LEARN and then converted to a pandas DataFrame for convenience, as shown in the next slide:

Code

```
>>> import pandas as pd
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
>>> df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
>>> df.head()

   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  species
0                5.1               3.5  ...               0.2   setosa
1                4.9               3.0  ...               0.2   setosa
2                4.7               3.2  ...               0.2   setosa
3                4.6               3.1  ...               0.2   setosa
4                5.0               3.6  ...               0.2   setosa
```

Here we can see the variables in this dataset include measurements of the physical properties of different species of flowers.

Summarize Your Data

Pandas DataFrames have some useful built-in methods to help get a quick overview of your data. You can use df.shape to print the shape of the DataFrame.

The output is a tuple with the number of observations and the number of variables.

For the summary statistics of your dataset, use the df.describe() method. The output looks like this:

Code

```
>>> df.describe()

       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
count         150.000000        150.000000         150.000000        150.000000
mean            5.843333          3.057333           3.758000          1.199333
std             0.828066          0.435866           1.765298          0.762238
min             4.300000          2.000000           1.000000          0.100000
25%             5.100000          2.800000           1.600000          0.300000
50%             5.800000          3.000000           4.350000          1.300000
75%             6.400000          3.300000           5.100000          1.800000
max             7.900000          4.400000           6.900000          2.500000
```

This calculates the mean, standard deviation, minimum, and 25th, 50th (median), and 75th percentiles. It provides a nice quick overview of your variables.

Visualize Your Data

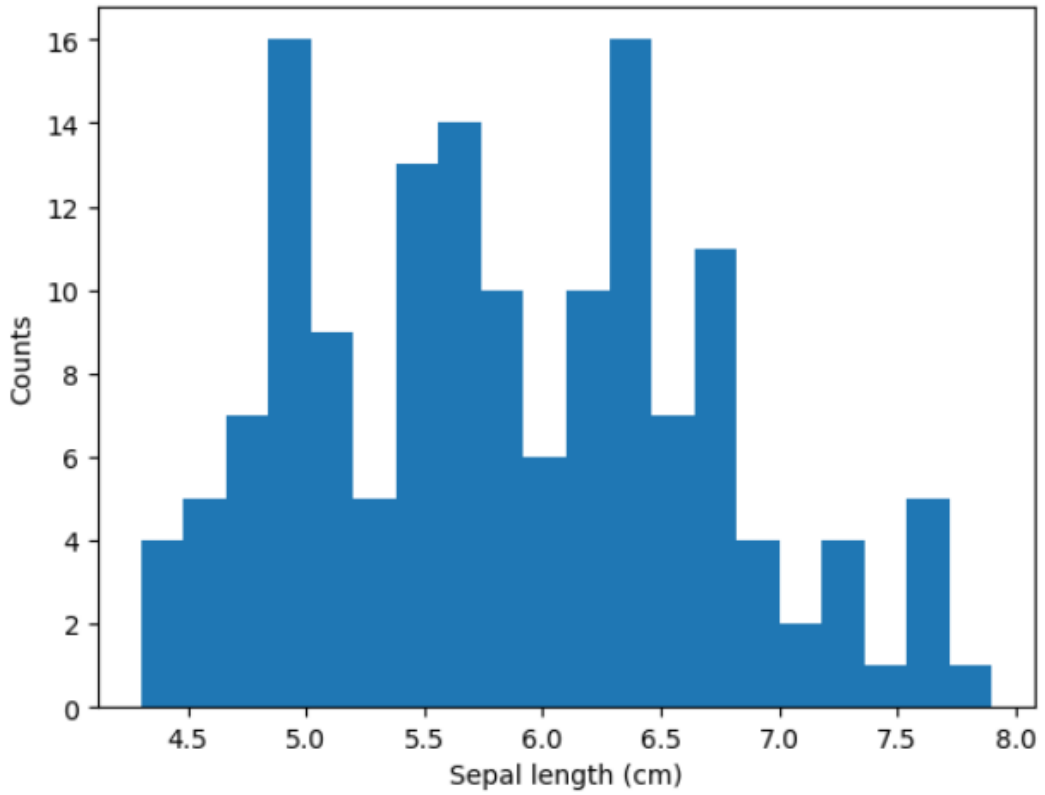
The next step in the EDA process is to start plotting your data to get an idea of the nature of the variables. The Python library Matplotlib – which is used to create static or interactive visualizations – is useful here.

The type of visualizations to consider plotting at this stage are histograms, box plots, bar plots, or density plots (amongst others). An example of using Matplotlib to plot a histogram of the sepal length is shown below:

Code

```
>>> import matplotlib.pyplot as plt
>>> plt.hist(df['sepal length (cm)'], bins=20)
>>> plt.ylabel('Counts')
>>> plt.xlabel('Sepal length (cm)')
>>> plt.show()
```

Running this code produces the diagram below. You can see the sepal length for all species varies between about 4.3 to 7.9 cm. Note also that this distribution doesn't look Gaussian; this could be formalized with a statistical test. This means, however, that some statistical quantities or tests that assume a normal distribution may not be valid.



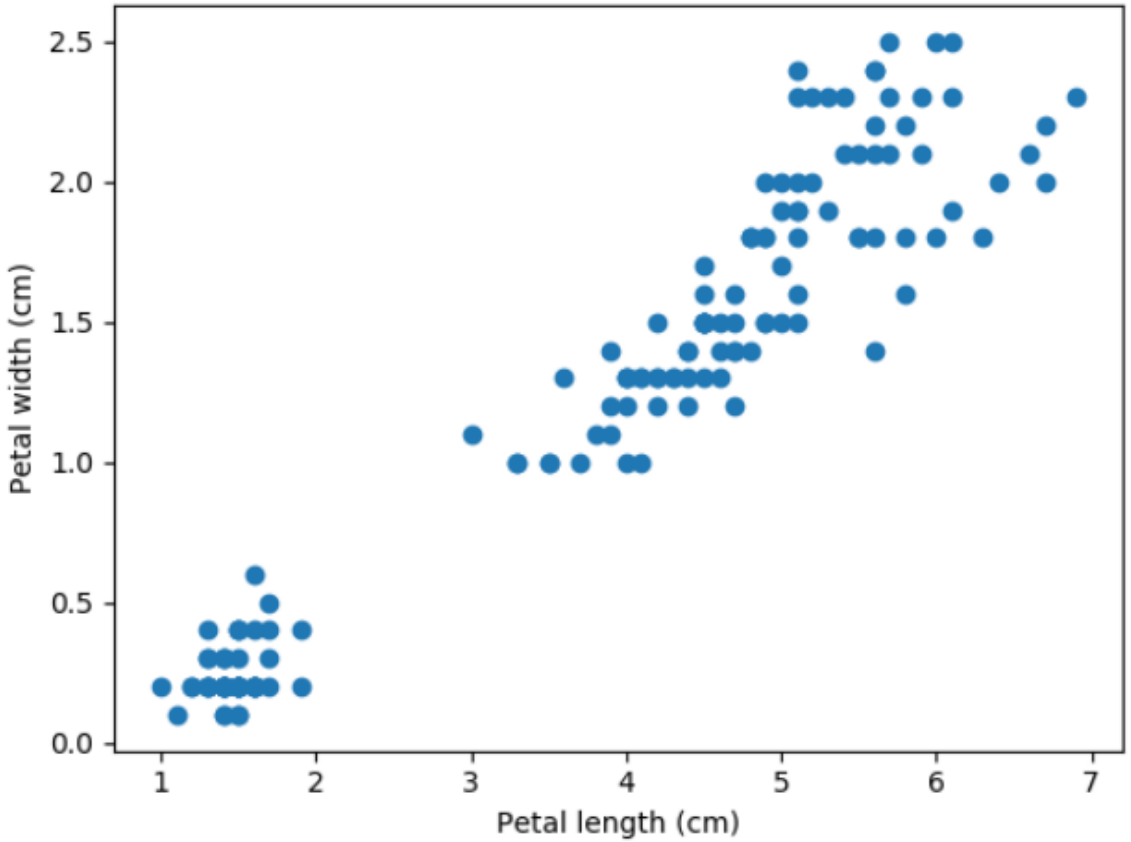
Next, you could consider plotting pairs of variables to see if there are any relationships between quantities.

For example, you can plot a scatter plot of petal length against petal width as follows:

Code

```
>>> plt.scatter(df['petal length (cm)'], df['petal width (cm)'])
>>> plt.ylabel('Petal width (cm)')
>>> plt.xlabel('Petal length (cm)')
>>> plt.show()
```

The resulting diagram is shown below. You can see a nice linear relationship between the two variables: as petal length increases, so does petal width. Interestingly there appears to be two clusters in the data. This is begging for an explanation!



At this point in the analysis, you shouldn't worry too much about making visually appealing plots; just use the style defaults to save time and effort. Check out the MATPLOTLIB GALLERY for detailed examples of the many different types of plots available.

Group Your Data

From the above visualization, we have a clue that there could be naturally occurring groups in the data. A good next step is to start to look at how the data could be grouped together. The built-in pandas function df.groupby() can come in handy here.

The groupby function must be used in conjunction with another function to produce a summary statistic of the group, for example df.max() or df.min(). Let's take a look at a concrete example:

Code

```
>>> df.groupby('species').mean()
            sepal length (cm)  ...  petal width (cm)
species                        ...
setosa                  5.006  ...             0.246
versicolor              5.936  ...             1.326
virginica               6.588  ...             2.026
```

In the above example, we group the data by the species column and calculate the mean of each variable for that species.

The results are printed to the console and show the species Virginica has the largest sepal length on average. Setosa, on the other hand, has quite clearly the smallest petal width compared to the other species. Perhaps this species could represent the lower left group from the above section?

From above we know the 50th percentile of the petal width is 1.3 cm. We can find out how many members of each species have petal widths greater than this value. Just run the following code:

Code

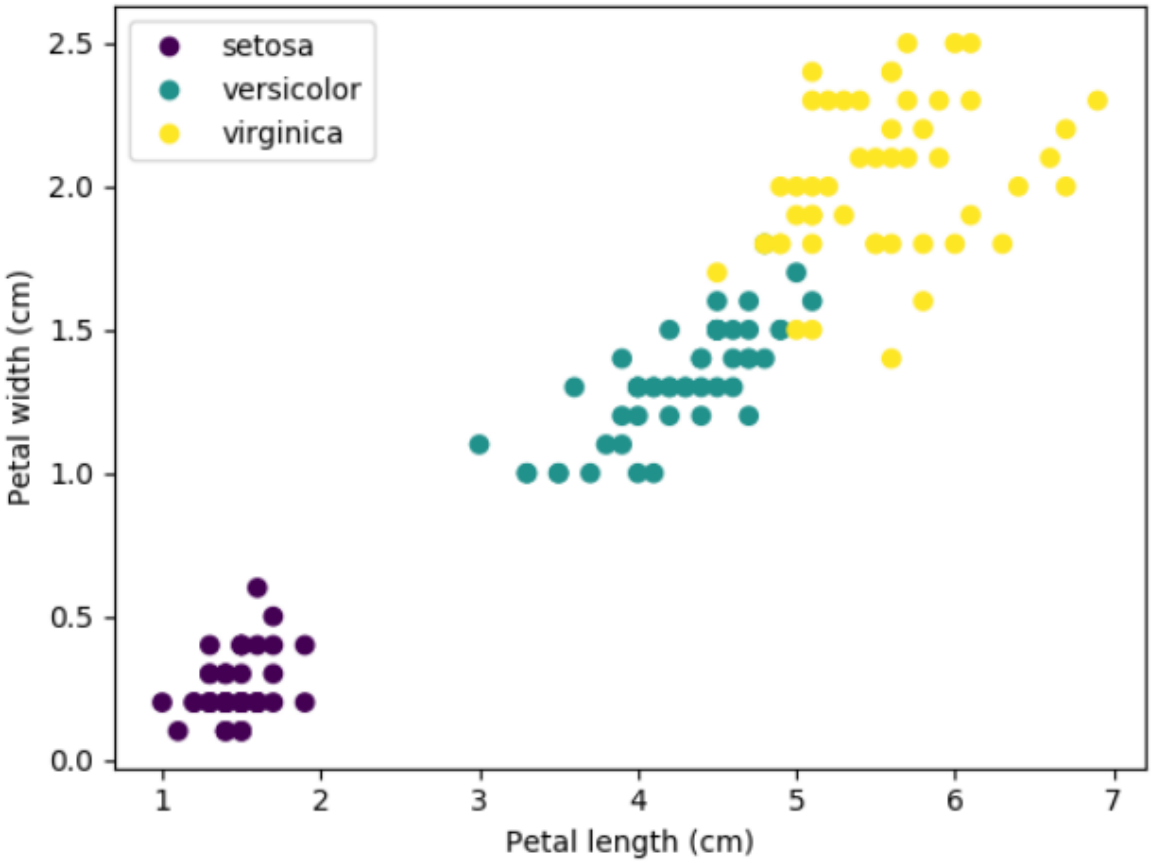
```
>>> df[df['petal width (cm)']>1.3]['species'].value_counts()
virginica     50
versicolor    22
setosa         0
```

Here, we're subsetting the DataFrame to return the species column for all observations which have a petal width greater than 1.3 cm. Then, using the df.value_counts() method, we count how many of each species we have. This reveals there are no examples of Setosa with a petal width greater than 1.3 cm. This suggests the lower left cluster represents this species. But what about the upper right cluster?

We can plot the same scatter diagram but colored by species. We just need to pass an array-like data structure containing an integer identifying the species to the c argument of the scatter function. To generate another scatter plot, use the following code:

Code

```
>>> scatter = plt.scatter(df['petal length (cm)'], df['petal width (cm)'],
c=iris.target)
>>> plt.legend(handles=scatter.legend_elements()[0],
labels=df['species'].unique())
>>> plt.ylabel('Petal width (cm)')
>>> plt.xlabel('Petal length (cm)')
>>> plt.show()
```



Grouping the data by species reveals that the upper right cluster contains two sub-clusters belonging to the Versicolor and Virginica species. And as we suspected, the lower left cluster belongs to the Setosa species.

How to Use This Cheat Sheet for Exploratory Data Analysis

In this guide, we demonstrated a typical process for EDA. We started with reading in and cleaning the data and proceeded to getting a quick understanding of the variables.

The next steps depend on the type of data you have, but they should involve visualizing each variable using histograms, box, and bar plots. Density plots and scatter diagrams are a good way to plot two variables to see if there are any interesting relationships worth further investigating.

Our findings motivated us to start to separate the data into groups and to try to explain why groupings occur. If you have tabular data, HOW TO PRETTY-PRINT TABLES IN PYTHON has some examples about producing nice tables in Python.

Now that you know what steps EDA includes, you can come back to this guide and use it as a cheat sheet to inspire you on how to best inspect your data.

Other techniques (such as statistical hypothesis testing) could provide some new insights, such as confirming if the distribution of sepal length (shown above) is statistically different for each species.

If you want some relevant learning material, the INTRODUCTION TO PYTHON FOR DATA SCIENCE course is aimed at beginner data scientists. For more in-depth learning material, consider taking the DATA PROCESSING WITH PYTHON track. It contains 5 interactive courses and is aimed at advanced users. Or you can start your own Python project.

Here are some PYTHON DATA SCIENCE PROJECT IDEAS to get you motivated to practice what you have learnt here.

If you found the material worthy to be shared, please tag LearnPython.com as a gesture of appreciation.

LearnPython
• com