CYBR 486 - Lab #2 – Plotting Iris Data

Picking up from Lab1, this lab will also work with the iris dataset. We'll be doing further work with the dataset, but this time focusing on plotting specific data into graphs for visual representation. The idea is to get used to generating these various plots, get familiar with the plotting library and its various offerings, and start to recognize characteristics and connections between bits of data in a dataset.

To start with we keep the same imports as before (though we won't always use all of them for each lab), but we add a sub-library of matplotlib so make sure you have it installed.

Terminal prompt: path/to/stuff> ***pip install matplotlib***

Terminal prompt: path/to/stuff> ***pip install seaborn***

Terminal prompt: path/to/stuff> ***pip install numpy***

Imports:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb
```

General seaborn documentation: https://seaborn.pydata.org/tutorial.html

Same for matplotlib: https://matplotlib.org/stable/

1.) This time around we’re going to start creating visualizations to represent data from datasets. It’s pretty hard to distill knowledge from a machine learning model without being able to visualize and present the model’s performance. We’ll start with a basic example of plotting 2 variables on the 2 axes of a simple graph.

```
def example1():
    time = [0, 1, 2, 3, 4, 5]
    distance = [0, 30, 60, 90, 120, 150]

    plt.plot(time, distance)
    plt.xlabel('Time (in hours)')
    plt.ylabel('Distance (in miles)')
    plt.show()
```

If you run this code, this will pop up a simple line graph with a labeled x and y axis using the data from the hard-coded lists for time and distance. Invoking the simple plot() method from matplot can have many arguments, but this is the most basic where the first argument is what you want for your x-axis, and the second argument is for your y-axis.

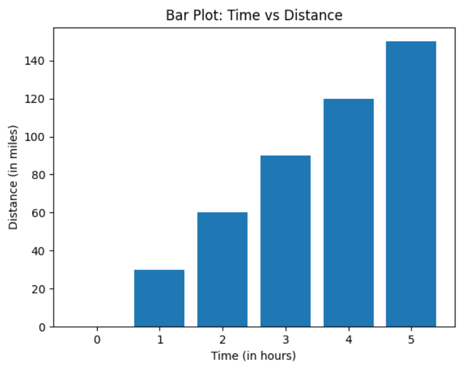
We can also invoke various other plot styles for when we want to visualize data differently. Different graphing styles are more suited for different datasets.

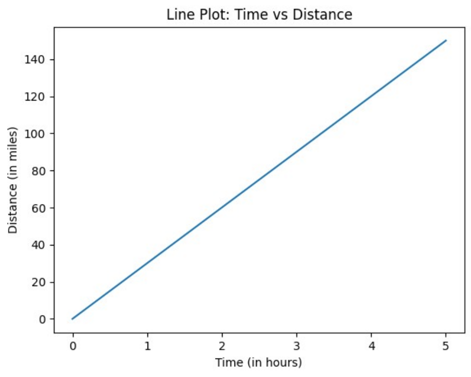
```python
# very basic example of matplot showing distance covered over time
def example1():
    time = [0, 1, 2, 3, 4, 5]
    distance = [0, 30, 60, 90, 120, 150]

    plt.plot(time, distance)
    plt.xlabel('Time (in hours)')
    plt.ylabel('Distance (in miles)')
    plt.show()

    plt.bar(time, distance)
    plt.show()

    plt.scatter(time, distance)
    plt.show()
```

This is the extended version of that first example. If you run this, it will create the line plot and display it. When you close that window, it'll create a similar plot but this time a bar graph. Closing that window will then display a scatter plot.

Matplot plotting documentation: https://matplotlib.org/stable/api/axes_api.html

Question 1a: using what you've seen, create 2 simple 2-axis graphs. You can use other methods to format the plots or dive into the documentation to play with various plotting methods. Make something you find interesting or funny. Imaginary bonus points for graphs that make me laugh. You can use the same two sets of data variables for both graphs but make 2 DIFFERENT styles of graphs. So like 1 bar plot and 1 line plot, or 1 pie chart and one histogram. Attach a screenshot of your graphs at runtime below.





Now we do the same, making simple plots using our iris dataset.

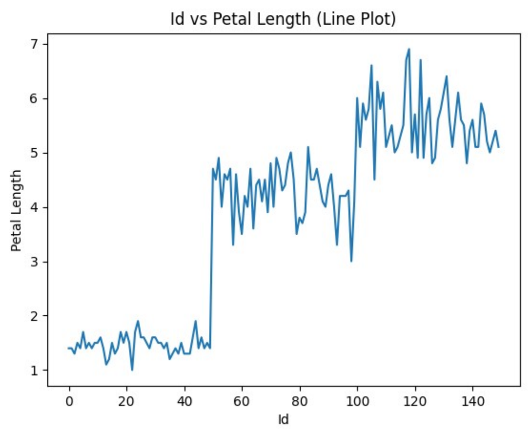
NOTE1: in order to get a single column of data from a dataset to use in a plot you use the following syntax, assuming the variable name of your data frame is data_frame:

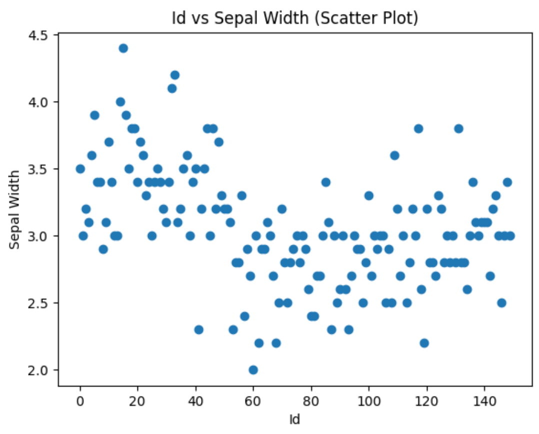
data_frame[“data_column_name”]   or

data_frame.Id (in the case of wanting to use the Id numbers as one of your axes

<u>NOTE2:</u> You will want to have your x-axis be the Id column so you can plot each individual data point. The second column can be any of the numeric columns not including Id, and obviously not the variety or species columns as they are strings, and we need to do some extra processing to not throw an exception with that

<u>Question 1b: Referencing what we did in lab 1, as well as the above example, read the iris.csv dataset into memory as a pandas data frame, then create 2 plots where the style is your choice. Take screenshots of your 2 plots running and attach them below. I risk making this too easy, but an example would be Id on the x-axis and petal length on the y-axis.</u>





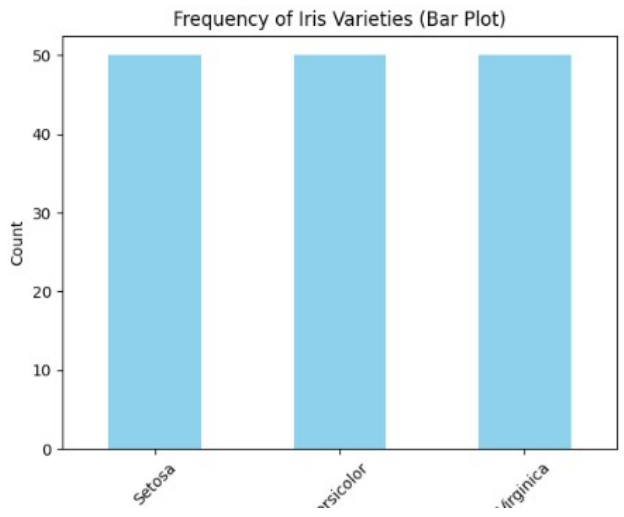
2.) Plotting data relevant to data analysis: Frequency

When looking over datasets to determine what model to use, or just to try to make some initial decisions about how the data relates to each other, a good first step is determining the frequency of recurring items in the dataset. Regarding our iris dataset, it would be good to know the frequency of each species, or in other words how many data points are there for each variety of iris.

As is usually the case in these situations, there already exists a nifty method we can call to count the number of occurrences of identical values in a column of a data frame, the aptly named method: value_counts() from pandas.
https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html

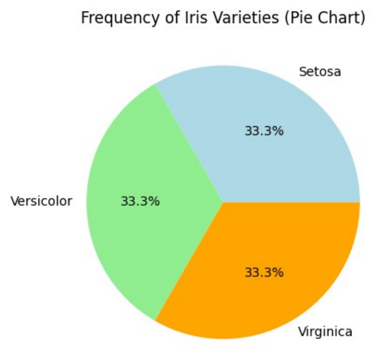
There are many things this method can do, it can provide absolute counts, it can sort after counting, it can ignore null or empty entries, or it can even provide a normalized count, where the numbers returned represent the percentage of the set for each entry.

Question 2a: Using the documentation of the value_counts() method, create a bar plot that will display the frequency of the different varieties of irises. Take a screenshot of your code and your plot running and attach below



Question 2b: Similar to above, but this time create a pie chart showing the frequency of the iris varieties. NOTE: documentation for pie plots: https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html

Take a screenshot of your code and the plot running and attach below



3.) Plotting data relevant to data analysis: Data Relationships

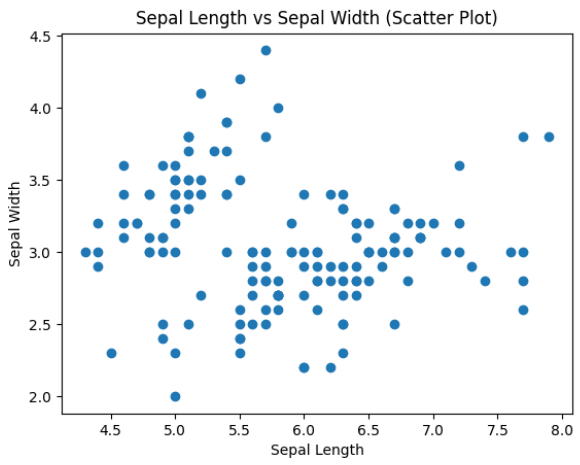
Another important use of dataset plotting is the ability to plot 2 variables in relation to each other in order to try to spot any relationships between those variables. This task is similar to 1b in this lab and is the reason for having you use Id as your x-axis for that part. For this part you will be relating an actual data variable like sepal length, to another data variable like sepal width.

Try running a plot using the values below:

```
    X = data["sepal.length"]
    y = data["sepal.width"]

    plt.plot(X, y)
    plt.show()
```

You’ll notice that this is pretty hard to read. The reason being that invoking plot() is making a line graph and looking at the dataset, you could (and do in this case) have multiple data points that overlap. Like having multiple entries where the sepal length is 4.5, but the sepal width is varying. In this case, you would be much better served with a ***scatter*** plot, which will place a dot on the graph for each entry where it’s sepal length and sepal width fall on the graph.

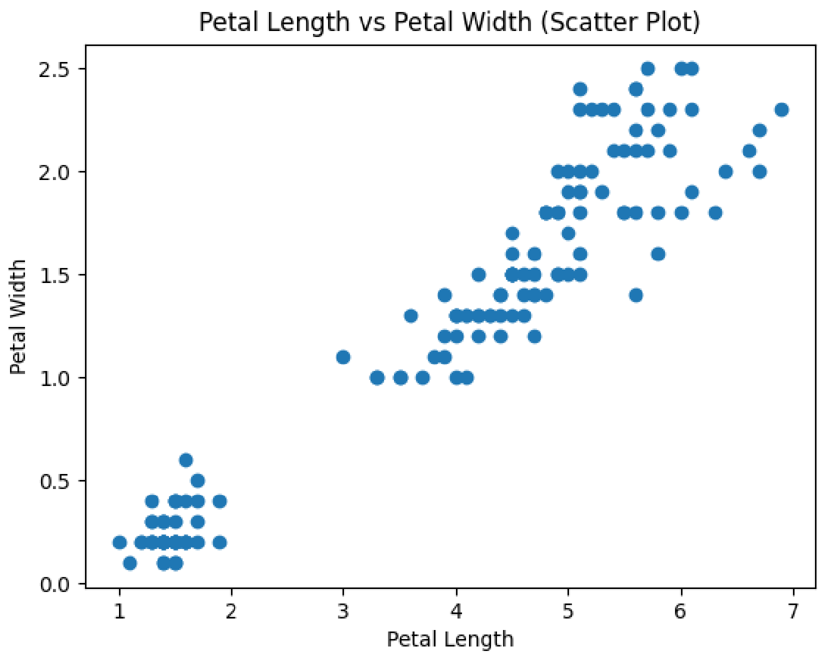
Question 3a.) Create the scatter plot discussed above comparing sepal length and sepal width. Are there any observations you can make at this point regarding the relationship between these variables? If so, what?



Observations:

From this scatter plot, we can observe a weak negative correlation between sepal length and sepal width, where larger sepal lengths tend to correspond to slightly narrower sepal widths. However, there is a noticeable spread of values, with no strong linear relationship. Different species of irises may account for this spread, with *Setosa* likely exhibiting smaller sepals compared to other species.

Question 3b.) Create a similar scatter plot but this time compare petal length to petal width. Similarly to the previous question, are there any observations you can make at this point regarding the relationship between these variables? If so, what?



In contrast to the sepal dimensions, the scatter plot of petal length versus petal width shows a much stronger **positive correlation**, where petal width increases as petal length increases. This relationship is clearer, indicating that petal dimensions are more closely linked, with less variability compared to sepals. The species clusters are also likely to be more distinct in this plot, reflecting clearer differentiation among the iris varieties based on petal size.

4.) Seaborn - Now we introduce another nifty library for plotting, seaborn.

Seaborn documentation: https://seaborn.pydata.org/tutorial.html

Seaborn adds some really nifty features to sort of streamline the things we've done manually with matplotlib so far. In particular if you glance at the various guides and tutorials in the documentation, you'll notice very colorful and organized results. This is good for visualization as it can very easily enhance our plots to make them more understandable. For example, we can create a similar scatterplot to the ones for 3a and 3b, but this time add some extra clarity through color differentiation.

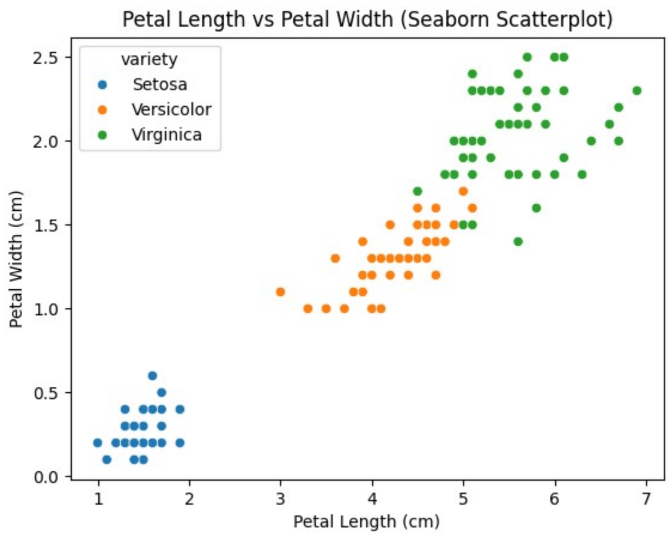
```
file = "iris.csv"
data_frame = pd.read_csv(file)
sb.scatterplot(data=data_frame, x='sepal.length', y='sepal.width',
hue='variety')
plt.show()
```

The code above will create a scatterplot similar to what we did before, except it adds a lot of neat things. Like intuitive labeling from the data frame itself just by designating the data frame as an argument, our x= and y= variable columns as arguments, and adding this hue= argument. The hue argument adds color to the data points based on the column you specify. In this case we’re coloring the points based on iris variety.

Question 4a.) Based on the seaborn scatterplot, are there any conclusions you can draw or any relationships between sepal length and sepal width? If so, what are they?

Based on the seaborn scatterplot of sepal length versus sepal width, we can observe that the three iris species form distinct clusters. The *Setosa* species has shorter sepal lengths and wider sepal widths, separating it clearly from the other species. There is a slight negative correlation between sepal length and sepal width, particularly for *Versicolor* and *Virginica*, where sepal width decreases as sepal length increases. *Versicolor* and *Virginica* show some overlap, but *Virginica* generally has longer sepals, while *Setosa* is more distinct in its dimensions.

Question 4b.) Create a seaborn scatterplot relating petal length to petal width adding hue based on the variety. Are there any conclusions or relationships you can spot? If so, what?



The seaborn scatterplot reveals a **strong positive correlation** between petal length and petal width across all iris species. The iris varieties form distinct clusters in the plot, with **Setosa** having much shorter and narrower petals, clearly separated from the other species. **Versicolor** and **Virginica** both show larger petal dimensions, but *Virginica* tends to have even longer and wider petals than *Versicolor*, which helps differentiate them. The distinct clustering indicates that petal dimensions are a strong feature for distinguishing between iris species.

4o

5.) Heatmaps and data correlation

Lastly for this lab we'll cover a common way to visualize data correlation. The goal here is to now create a new visualization that will display how variables relate to other variables in our iris dataset. We will do this using a seaborn heatmap.

For those unfamiliar a heatmap is a color or grayscale intensity graph where the higher or lower a value is, the intensity or hue of the color changes accordingly. For our purposes we want to create a heatmap displaying the correlation between the variables in our set.

While we're on the definition train I suppose a quick definition of correlation in this context is appropriate. It's basically a statistical measure that describes the relationship between 2 variables. Keep in mind that we're looking for correlation not causation, so the variables are likely not directly related to each other, but machine learning is all about looking for patterns.
Seaborn heatmap documentation:
https://seaborn.pydata.org/generated/seaborn.heatmap.html

The first thing we need to do is generate the data correlation values, and as luck would have it… there's a method for that too, from the pandas library: .corr()

```
    file = "iris.csv"

    data_frame = pd.read_csv(file)

    frame_without_variety = data_frame.drop(columns=['variety', 'Id'])

    sb.heatmap(frame_without_variety.corr(), annot=True)

    plt.show()
```

You'll notice a few additions in the code above. Firstly we're creating a new variable and dropping the variety and the Id columns as first we just want to see how the 4 main variables relate to each other independently of iris species. Secondly you can see the heatmap creation, which calls the corr() method for one of its arguments then adds the 'annot=True' argument, which is a Boolean argument that simply adds number annotations to the heatmap which can help people that struggle to see color better understand the heatmap.

Run the code and then look at the heatmap created. The scale on the right displays the color key for the correlation scale. 1.0 means highly correlated (basically if the correlation is 1.0, when one variable changes by x, the other variable changes by exactly x in the same way) The higher the number when positive, indicates how closely the variables move in the same direction. As the scales dips into the negative, this indicates the variables scale in opposite directions, meaning if one goes up, the other goes down and vice versa.

NOTE: You can ignore any points where a variable is compared with itself, because obviously a variable will scale at a 1:1 ratio when compared only to itself.

Question 5a.) What observations can you make looking at the variety agnostic heatmap?

From the variety-agnostic heatmap, we can observe the following:

1. Strong Positive Correlation: There is a strong positive correlation between petal length and petal width (correlation ~0.96). This means that as petal

length increases, petal width tends to increase in almost the same proportion.

2. Moderate Positive Correlation: There is a moderate positive correlation between sepal length and petal length (~0.87) as well as sepal length and petal width (~0.82), indicating that larger sepals often correspond to larger petals.
3. Weak Correlation: Sepal width has a weaker correlation with other variables. It has a very low negative correlation with sepal length (-0.12) and almost no correlation with petal width and petal length.

Finally, we add a few more lines to group the entries by their variety to see data correlation between the variables within each variety separately

```
grouped_data = (data_frame.groupby('variety'))
sb.heatmap(grouped_data.corr(), annot=True)
plt.show()
```

We can invoke the .groupby() method from pandas in order to create a sort-of bucket for each variety and plot them separately. The heatmap creation and plot display are the same as before.

<u>Question 5b.) What observations can you make of the correlation between variables for each variety?</u>

- Setosa: The sepal length and sepal width have a weak negative correlation, while petal length and petal width are not strongly correlated. Sepal dimensions are more loosely related to petal dimensions in Setosa compared to the other species.
- Versicolor and Virginica: In both Versicolor and Virginica, there are strong positive correlations between petal length and petal width, similar to the variety-agnostic heatmap. However, Virginica shows even stronger correlations between petal and sepal dimensions compared to Versicolor, suggesting a more direct relationship between these variables in this species.

<u>FINAL NOTE:</u> Apologies for adding this at the very end but in case anyone reaches the end of this lab sheet stressing out over not having an extensive background in statistics and data analysis, don't worry. I'm only looking for basic observations on these questions, and how well you can spot patterns. I won't be very strict on them as I want you to get the general idea of what we're doing here.