AML Hw0: Iris Flowers

Jiwon Jeong

September 2025

1 Iris Flowers Dataset Structure

The data set is 3 different iris plant classes of 50 instances each. This means that the data includes 3 different species, and 50 samples for each species.

For each instance, there are 4 attributes. They are numerical measurements to be used for prediction (sepal length, sepal width, petal length, petal width). There is also the label for each sample, being the actual Iris species.

2 Parsing the Dataset

The dataset was parsed for the 4 attributes using pandas, resulting in an $N \times p$ array. Where N is 150, the number of samples, and p is 4, the number of attributes.

```
attributes = pd.read_csv(file_path, usecols=[0,1,2,3],
        header=None, names=[#array of headers names])
```

```
%run index.py
✓ 3.2s                                                    Python
attributes:
     sepal length in cm  sepal width in cm  petal length in cm  \
0                   5.1                3.5                 1.4
1                   4.9                3.0                 1.4
2                   4.7                3.2                 1.3
3                   4.6                3.1                 1.5
4                   5.0                3.6                 1.4
..                  ...                ...                 ...
145                 6.7                3.0                 5.2
146                 6.3                2.5                 5.0
147                 6.5                3.0                 5.2
148                 6.2                3.4                 5.4
149                 5.9                3.0                 5.1

     petal width in cm
0                  0.2
1                  0.2
2                  0.2
3                  0.2
4                  0.2
..                 ...
145                2.3
146                1.9
147                2.0
148                2.3
149                1.8

[150 rows x 4 columns]
```

With a similar approach, the same data file was parsed for the N-dimensional label vector.

```
labels = pd.read_csv(file_path, usecols=[4],
        header =None, names=['species'])
```

```
label (species):
            species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 1 columns]
```

3 Visualizing the Dataset

With 4 attributes, but only 2D representations that can show 2 attributes at a time, a series of 2D plots were created. Each plot has the x and y dimensions as two different pairs of attributes. With 12 plots, all possible pairs of two attributes from the 4 attributes were represented.

In code, this was performed by the following nested loop algorithm to create all pairs of different attributes.

```
# create a plot with subplots
fig, axes = plt.subplots(4,4, constrained_layout=True, figsize=(20,20))
# nested loop algorithm begins here
    for x_index in range(4):
     for y_index in range(4):
         if (x_index != y_index):
             axes[x_index,y_index].scatter(attribute_arr[x_index],
                 attribute_arr[y_index], c=color_vector)
             # more code to add titles
```

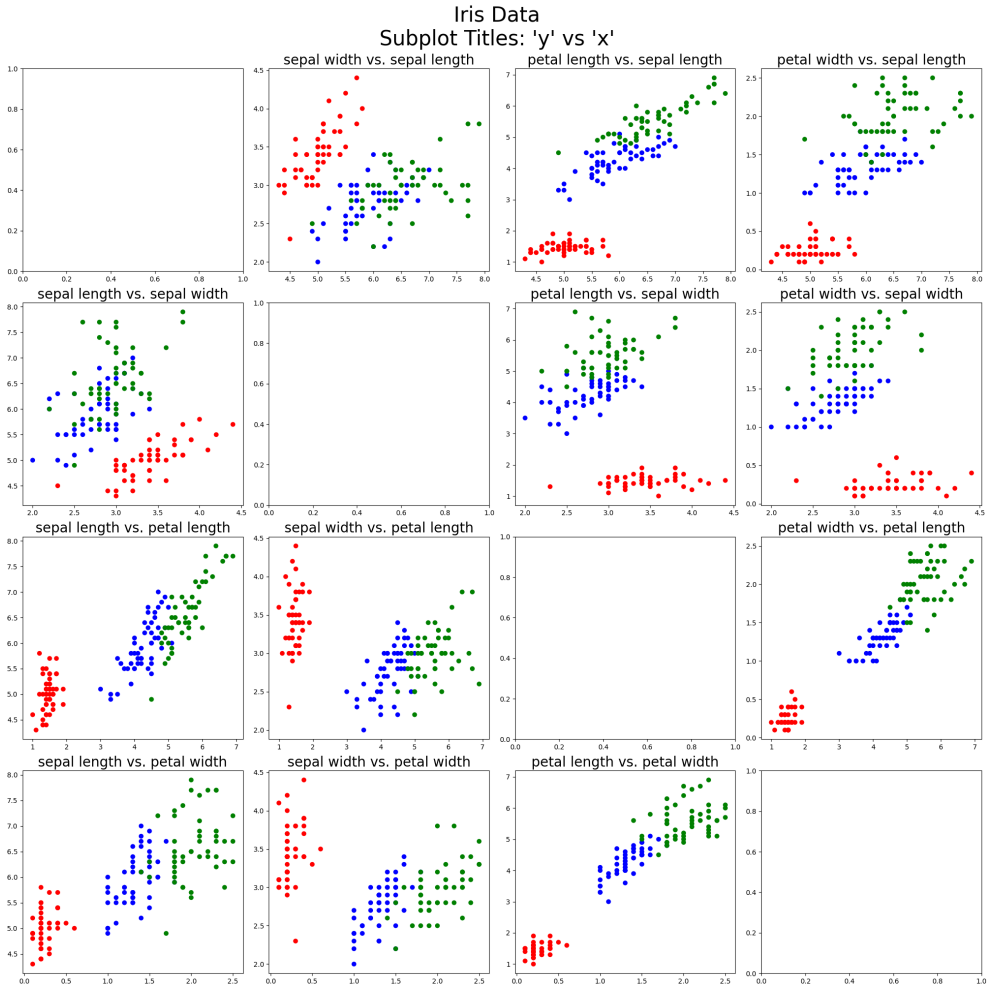
In the above code, axes[x_index, y_index] is responsible for creating a subplot on the position of (x_index, y_index). And the variable attribute_arr is an p×N array (constructed transpose of the parsed attributes data array), where the n-th element of attribute_arr will give the nth attribute's N-dimension vector. Lastly, color_vector is an N-dimension vector of assigned colors for each sample based on the label (species). The code for this is shown below.

```
    # get each attribute vector
```

```
attributes_matrix = attributes.to_numpy()
sepal_length = attributes_matrix[:,0:1]
sepal_width = attributes_matrix[:,1:2]
petal_length = attributes_matrix[:,2:3]
petal_width = attributes_matrix[:,3:4]

# create a graph using 2 attribute vector
# (for non-same attributes) for all possible pairs
# red is Iris-setosa, green is Iris-virginica, blue is Iris-versicolor
color_map = {
    'Iris-setosa': 'red',
    'Iris-virginica': 'green',
    'Iris-versicolor': 'blue'
}
labels_array = labels.to_numpy().tolist()
color_vector = [color_map[species[0]] for species in labels_array]
attribute_arr = [sepal_length, sepal_width, petal_length, petal_width]
attribute_category_strings = ["sepal length", "sepal width",
            "petal length", "petal width"]
```

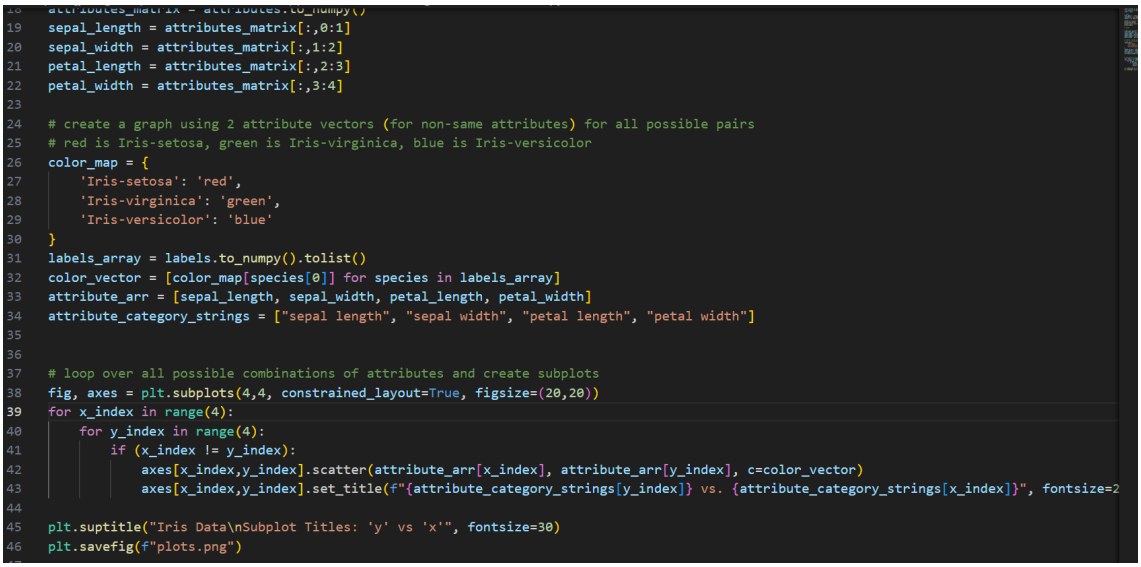
The final result is shown below. Red is Iris-setosa, green is Iris-virginica, and blue is Iris-versicolor.



As seen, the Iris dataset was properly visualized. Based on visuals, it seems that Iris-setosa is the distinguishable species from the other two species (separated well on most dimensions). And Iris-virginica and Iris-versicolor are poorly distinguishable (not well separated on most dimensions). This aligns with the description of the dataset by R.A. Fischer in the UC Irvine Machine Learning.

4 Source Code

The source code can be found at GitHub in the /hw-0 directory. In case link does not work, the source code for the plotting can be seen below.



```
attributes_matrix = attributes.to_numpy()
sepal_length = attributes_matrix[:,0:1]
sepal_width = attributes_matrix[:,1:2]
petal_length = attributes_matrix[:,2:3]
petal_width = attributes_matrix[:,3:4]

# create a graph using 2 attribute vectors (for non-same attributes) for all possible pairs
# red is Iris-setosa, green is Iris-virginica, blue is Iris-versicolor
color_map = {
    'Iris-setosa': 'red',
    'Iris-virginica': 'green',
    'Iris-versicolor': 'blue'
}
labels_array = labels.to_numpy().tolist()
color_vector = [color_map[species[0]] for species in labels_array]
attribute_arr = [sepal_length, sepal_width, petal_length, petal_width]
attribute_category_strings = ["sepal length", "sepal width", "petal length", "petal width"]


# loop over all possible combinations of attributes and create subplots
fig, axes = plt.subplots(4,4, constrained_layout=True, figsize=(20,20))
for x_index in range(4):
    for y_index in range(4):
        if (x_index != y_index):
            axes[x_index,y_index].scatter(attribute_arr[x_index], attribute_arr[y_index], c=color_vector)
            axes[x_index,y_index].set_title(f"{attribute_category_strings[y_index]} vs. {attribute_category_strings[x_index]}", fontsize=2

plt.suptitle("Iris Data\nSubplot Titles: 'y' vs 'x'", fontsize=30)
plt.savefig(f"plots.png")
```