Background:

The Iris Dataset contains four features (length and width of sepals and petals) of 50 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). These measures were used to create a linear discriminant model to classify the species. The dataset is often used in data mining, classification and clustering examples and to test algorithms.

Data Description

Petal Length - in cm

Petal Width - in cm

Sepal Length - in cm

Sepal Width - in cm

Species - Sentosa, Versicolour, and Virginica

Getting Started with Pandas:

In [1]:

```
import pandas as pd
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

Load the dataset

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. This is a very famous and widely used dataset by everyone trying to learn machine learning.

The dataset is available in the UCI Machine Learning repository. we will load this data from a library called seaborn.

In [2]:

```
import pandas as pd
import seaborn
data = seaborn.load_dataset("iris")
```

Check if the dataset has been loaded correctly

In [3]:

```
data.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [4]:

```
# Looking at the last few rows of the data frame
data.tail()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:

```
data.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
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

We can save this dataset to our local system as a csv file.

Export dataframe as csv

In [7]:

```
data.to_csv('iris.csv') # Saves the file in the same folder that contains the notebook
```

In [8]:

```
data.to_excel('iris_data.xlsx', index=False)
```

In [9]:

```
df = pd.read_csv('iris.csv')
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    150 non-null    int64
 1   sepal_length  150 non-null    float64
 2   sepal_width   150 non-null    float64
 3   petal_length  150 non-null    float64
 4   petal_width   150 non-null    float64
 5   species       150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Let us now look at the data itself

Displaying the number of rows randomly

In [11]:

```
data.sample(10)
```

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	species
126	6.2	2.8	4.8	1.8	virginica
86	6.7	3.1	4.7	1.5	versicolor
107	7.3	2.9	6.3	1.8	virginica

4	5.0	3.6	1.4	0.2	setosa
52	6.9	3.1	4.9	1.5	versicolor
35	5.0	3.2	1.2	0.2	setosa
57	4.9	2.4	3.3	1.0	versicolor
39	5.1	3.4	1.5	0.2	setosa
13	4.3	3.0	1.1	0.1	setosa
19	5.1	3.8	1.5	0.3	setosa

Check out the shape of the dataset

In [12]:

```
data.shape
```

Out[12]:

```
(150, 5)
```

The dataset has 150 rows of observations and 5 columns.

Slicing the rows

If you want to print or work upon a particular group of lines that is from say 10th row to 20th row.

In [13]:

```
# data[start:end]
# start is inclusive whereas end is exclusive
print(data[10:21])
# it will print the rows from 10 to 20.

# you can also save it in a variable for further use in analysis
sliced_data=data[10:21]
print(sliced_data)
```

```
    sepal_length  sepal_width  petal_length  petal_width species
10           5.4          3.7           1.5          0.2  setosa
11           4.8          3.4           1.6          0.2  setosa
12           4.8          3.0           1.4          0.1  setosa
13           4.3          3.0           1.1          0.1  setosa
14           5.8          4.0           1.2          0.2  setosa
15           5.7          4.4           1.5          0.4  setosa
16           5.4          3.9           1.3          0.4  setosa
17           5.1          3.5           1.4          0.3  setosa
18           5.7          3.8           1.7          0.3  setosa
19           5.1          3.8           1.5          0.3  setosa
20           5.4          3.4           1.7          0.2  setosa
    sepal_length  sepal_width  petal_length  petal_width species
10           5.4          3.7           1.5          0.2  setosa
11           4.8          3.4           1.6          0.2  setosa
12           4.8          3.0           1.4          0.1  setosa
13           4.3          3.0           1.1          0.1  setosa
14           5.8          4.0           1.2          0.2  setosa
15           5.7          4.4           1.5          0.4  setosa
16           5.4          3.9           1.3          0.4  setosa
17           5.1          3.5           1.4          0.3  setosa
18           5.7          3.8           1.7          0.3  setosa
19           5.1          3.8           1.5          0.3  setosa
20           5.4          3.4           1.7          0.2  setosa
```

Displaying only specific columns

In [14]:

```
# Select columns Petal Width and Species from iris data
# we will save it in a another variable named "specific_data"

specific_data = data[["petal_width","species"]]
# data[["column_name1","column_name2","column_name3"]]

# now we will print the first 10 columns of the specific_data dataframe.
print(specific_data.head(10))
```

```
   petal_width species
0          0.2  setosa
1          0.2  setosa
```

```
2          0.2  setosa
3          0.2  setosa
4          0.2  setosa
5          0.4  setosa
6          0.3  setosa
7          0.2  setosa
8          0.2  setosa
9          0.1  setosa
```

Calculating sum, mean, median and mode of a particular column

In [15]:
```
data.columns
```

Out[15]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [16]:
```
# data["column_name"].sum()

sum_data = data['sepal_length'].sum()
mean_data = data['sepal_length'].mean()
median_data = data['sepal_length'].median()
mode_data = data['sepal_length'].mode()

print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data, "\nMode:",mode_data)
```

```
Sum: 876.5
Mean: 5.843333333333335
Median: 5.8
Mode: 0    5.0
dtype: float64
```

Calculating sum, mean and mode of a particular Species

In [17]:
```
data.species.unique()
```

Out[17]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [18]:
```
data.sepal_length.unique()
```

Out[18]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [19]:
```
data['species'].unique()
```

Out[19]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [20]:
```
# Species == 'Iris-setosa'

sum_data_sentosa = data.loc[data['species'] == 'setosa', 'sepal_length'].sum()
mean_data_sentosa = data.loc[data['species'] == 'setosa', 'sepal_length'].mean()
median_data_sentosa = data.loc[data['species'] == 'setosa', 'sepal_length'].median()
mode_data_sentosa = data.loc[data['species'] == 'setosa', 'sepal_length'].mode()

print("Sum:",sum_data_sentosa, "\nMean:", mean_data_sentosa, "\nMedian:",median_data_sentosa, "\nMode:",mode_data
```

```
Sum: 250.3
Mean: 5.005999999999999
Median: 5.0
Mode: 0    5.0
1    5.1
dtype: float64
```

In [21]:
```
sum_data_versicolor = data.loc[data['species'] == 'versicolor', 'sepal_length'].sum()
mean_data_versicolor = data.loc[data['species'] == 'versicolor', 'sepal_length'].mean()
median_data_versicolor = data.loc[data['species'] == 'versicolor', 'sepal_length'].median()
mode_data_versicolor = data.loc[data['species'] == 'versicolor', 'sepal_length'].mode()
```

In [22]:
```
sum_data_virginica = data.loc[data['species'] == 'virginica', 'sepal_length'].sum()
mean_data_virginica = data.loc[data['species'] == 'virginica', 'sepal_length'].mean()
median_data_virginica = data.loc[data['species'] == 'virginica', 'sepal_length'].median()
mode_data_virginica = data.loc[data['species'] == 'virginica', 'sepal_length'].mode()
```

In [23]:
```
print(sum_data_virginica)
print(mean_data_virginica)
print(median_data_virginica)
print(mode_data_virginica)
```

```
329.3999999999999
6.587999999999998
6.5
0    6.3
dtype: float64
```

In [24]:
```
print(data.groupby(['species'])['sepal_length'].sum())
print(data.groupby(['species'])['sepal_length'].mean())
```

```
species
setosa        250.3
versicolor    296.8
virginica     329.4
Name: sepal_length, dtype: float64
species
setosa        5.006
versicolor    5.936
virginica     6.588
Name: sepal_length, dtype: float64
```

In [25]:
```
print("Sum:",sum_data_versicolor, "\nMean:", mean_data_versicolor, "\nMedian:",median_data_versicolor, "\nMode:",
```

```
Sum: 296.8
Mean: 5.936
Median: 5.9
Mode: 0    5.5
1    5.6
2    5.7
dtype: float64
```

groupby function is very helpful when we want to analyse such information in the data. Please try it on this dataset to practice.

We will discuss group by and several other data manipulation functions in the next session.

VISUALIZATION USING MATPLOTLIB & SEABORN

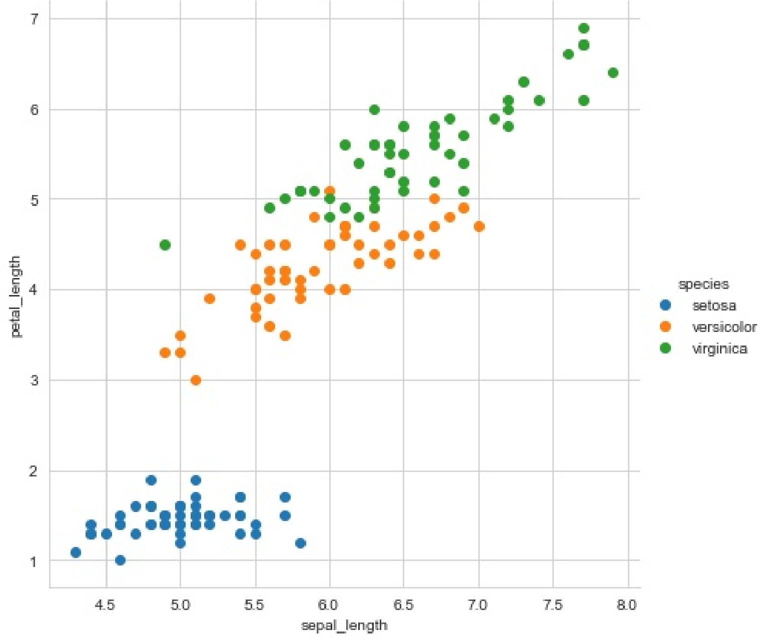
In [26]:
```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')

# style used as a theme of graph
# for example if we want black
# graph with grid then write "darkgrid"
sns.set_style("whitegrid")

# sepal_length, petal_length are iris
# feature data height used to define
# Height of graph whereas hue store the
# class of iris dataset.
sns.FacetGrid(iris, hue ="species",
              height = 6).map(plt.scatter,
                              'sepal_length',
                              'petal_length').add_legend()
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x7fd23928fdc0>



In [27]:
```
iris.corr()
```

Out[27]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

Based on the analysis of the Iris Dataset, here are five top conclusions:

Distinct Separation of Species Based on Features:

The Iris dataset shows clear separation between the three species (setosa, versicolor, virginica) based on sepal and petal dimensions. For instance, Iris setosa can be distinctly separated from the other two species using petal length and width, which are significantly smaller. Petal Dimensions as Key Differentiators:

Petal length and petal width are the most significant features for differentiating between species. The correlation matrix shows that petal length and petal width have strong positive correlations with each other (0.963) and with the species classification. These dimensions are crucial for building a discriminant model. Sepal Dimensions Show Lesser Variation:

While sepal length and sepal width do vary among species, the variation is not as distinct as petal dimensions. Sepal length has a moderate positive correlation with petal dimensions but is not as strong a differentiator between species on its own. Correlation Analysis:

There is a strong positive correlation between petal length and petal width (0.963), and both are strongly correlated with sepal length (0.872 and 0.818, respectively). Sepal width has a weak negative correlation with both petal length (-0.428) and petal width (-0.366), indicating that wider sepals do not necessarily correspond to larger petals. Species-Specific Summary Statistics:

Summary statistics show distinct differences in mean values of sepal and petal dimensions across species: Setosa: Sepal length mean of 5.01 cm, petal length mean of 1.46 cm. Versicolor: Sepal length mean of 5.94 cm, petal length mean of 4.26 cm. Virginica: Sepal length mean of 6.59 cm, petal length mean of 5.55 cm. These differences are significant and can be used to create effective classification models.

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js