ASSIGNMENT 5

Name: Aakanksha Bhondve

Grno: 21810939

Rollno: 322004

Comp B1

Aim –

Visualize the data and its outcome in Python by plotting the various graphs. Use Scatter plot, bar plot, Box plot and Histogram OR Perform the data visualization operations using Tableau for the given dataset and its prediction.

Theory –

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

Matplotlib - low level, provides lots of freedom

Pandas **-** Visualization: easy to use interface, built on Matplotlib

Seaborn - high-level interface, great default styles

ggplot - based on R’s ggplot2, uses Grammar of Graphics

Plotly - can create interactive plots

Code and Output –

In [3]:
```
import pandas as pd

import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", color_codes=True)

iris = pd.read_csv("Iris.csv")

iris.head()

```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

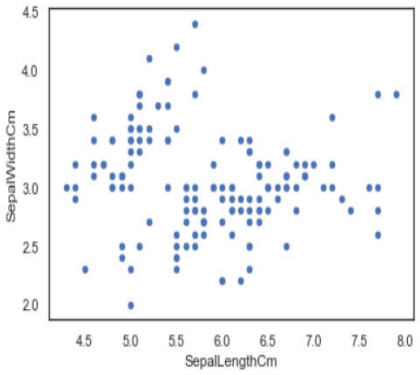
In [3]:
```
# Let's see how many examples we have of each species
iris["Species"].value_counts()
```

Out[3]:
```
Iris-setosa        50
Iris-virginica     50
Iris-versicolor    50
Name: Species, dtype: int64
```

In [5]:
```
# The first way we can plot things is using the .plot extension from Pandas dataframes
# We'll use this to make a scatterplot of the Iris features.
iris.plot(kind="scatter", x="SepalLengthCm", y="SepalWidthCm")
```

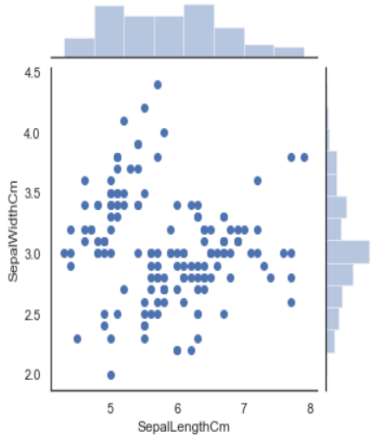
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a single row if you really want to specify the same RGB or RGBA value for all points.
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25818aa48c8>
```



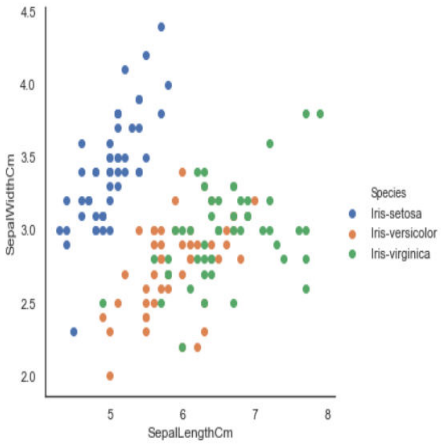
In [6]:
```
# We can also use the seaborn library to make a similar plot
# A seaborn jointplot shows bivariate scatterplots and univariate histograms in the same figure
sns.jointplot(x="SepalLengthCm", y="SepalWidthCm", data=iris, size=5)
```

Out[6]: <seaborn.axisgrid.JointGrid at 0x25818d80208>



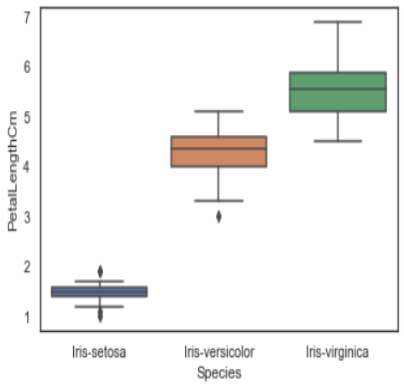
In [7]:
```
# One piece of information missing in the plots above is what species each plant is
# We'll use seaborn's FacetGrid to color the scatterplot by species
sns.FacetGrid(iris, hue="Species", size=5) \
   .map(plt.scatter, "SepalLengthCm", "SepalWidthCm") \
   .add_legend()
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x2581a799ec8>



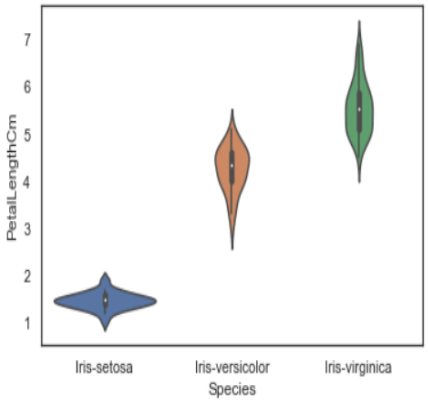
In [8]:
```
# We can look at an individual feature in Seaborn through a boxplot
sns.boxplot(x="Species", y="PetalLengthCm", data=iris)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2581a812908>



In [10]:
```
# A violin plot combines the benefits of the previous two plots and simplifies them
# Denser regions of the data are fatter, and sparser thiner in a violin plot
sns.violinplot(x="Species", y="PetalLengthCm", data=iris, size=6)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2581a8f2188>



In [11]:
```
# A final seaborn plot useful for looking at univariate relations is the kdeplot,
# which creates and visualizes a kernel density estimate of the underlying feature
sns.FacetGrid(iris, hue="Species", size=6) \
   .map(sns.kdeplot, "PetalLengthCm") \
   .add_legend()
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x2581a9dcac8>