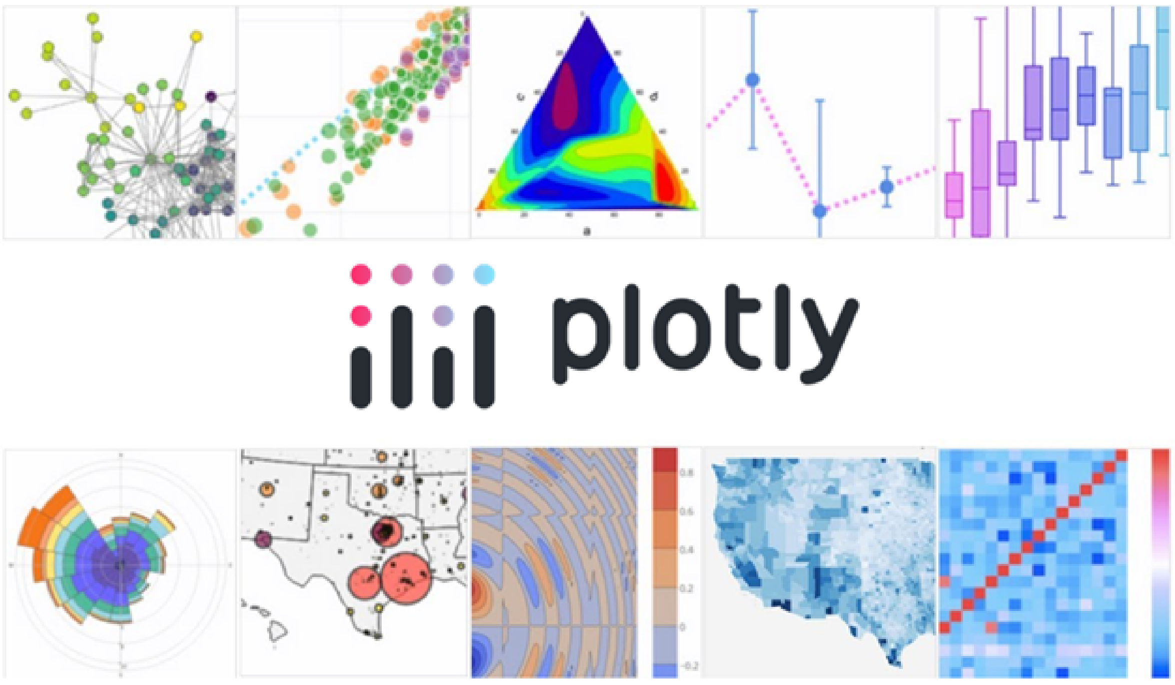
Lecture 23 & LAB 10: Data Visualization using **PLOTLY**



- Python Plotly Library is an open-source library that can be used for data visualization and effective understanding and representation of data
- Plotly supports various types of plots like line charts, scatter plots, histograms, box plots, etc.
- **Features**
  - Plotly has hover tool capabilities that allow in-depth analysis ans to detect any anomalies in a large number of data points.
  - It can be used to generate visually attractive plots

1. Installing Plotly

```
!pip install plotly
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (8.2.2)
```

2. Importing Plotly Library

```
import plotly.express as px
```

BASIC PLOTS USING PLOTLY

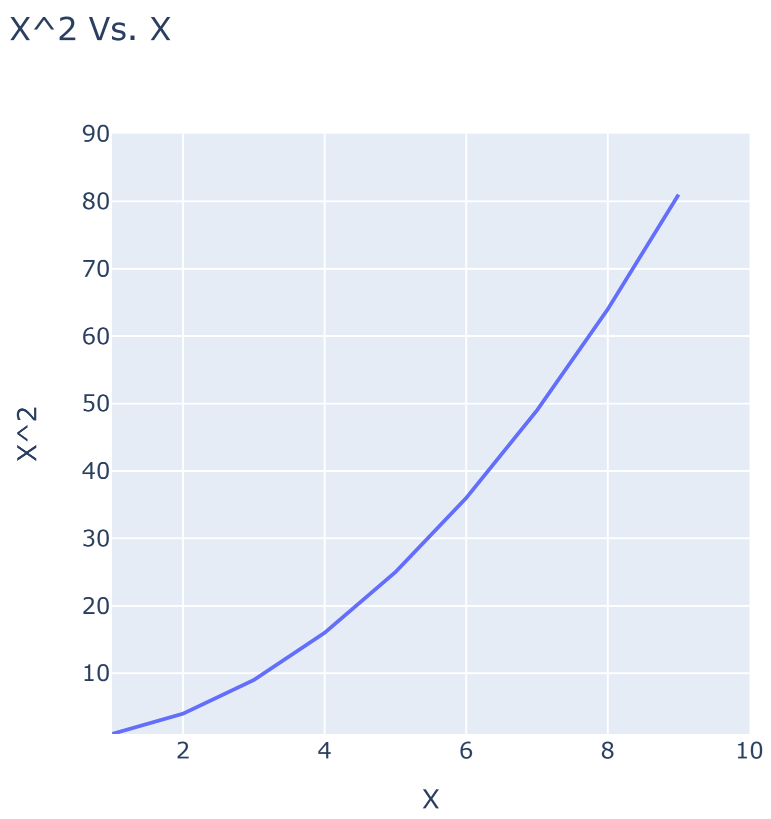
3. LINE PLOT

3A. LINE PLOT 1

X^2 Vs. X

- Basic plot using **px.line( )**
- using **'title'** parameter to add title to the plot
- using **'labels'** parameter to add x-axis and y-axis labels
- using **range_x() and range_y()** parameters to set minimum and maximum range of x-axis and y-axis
- using **height and width** parameters to adjust figure size
- using **.show()** parameter to display plot

```
import plotly.express as px

X = [m for m in range(1,10)]
Y = [m**2 for m in X]

Plot1 = px.line(x=X, y=Y,title='X^2 Vs. X',
                labels=dict(x='X',y='X^2'),
                range_x=(1,10), range_y=(1,90),
                height=500,width=500)

Plot1.show()
```

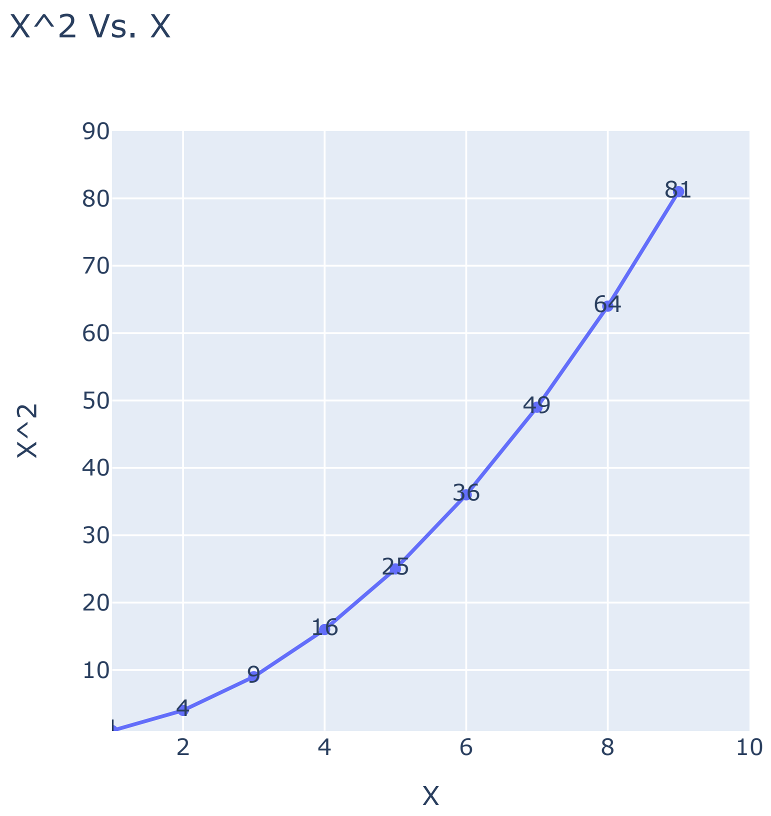


3B. LINE PLOT 2

X^2 Vs. X

- using '**markers = True**' parameter to display markers
- using **text** parameter to display values against marker datapoints

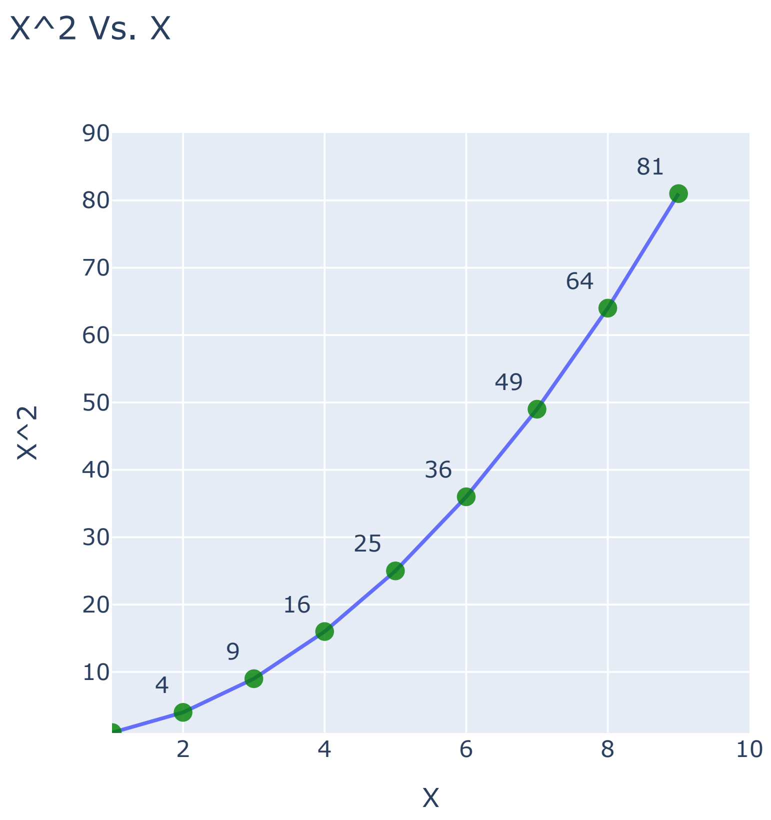
```
import plotly.express as px

X = [m for m in range(1,10)]
Y = [m**2 for m in X]

Plot1 = px.line(x=X, y=Y,title='X^2 Vs. X',
                labels=dict(x='X',y='X^2'),
                range_x=(1,10), range_y=(1,90),
                height=500,width=500,
                markers=True,
                text=Y)

Plot1.show()
```



3C. LINE PLOT 3

- Formatting marker datapoints and text labels

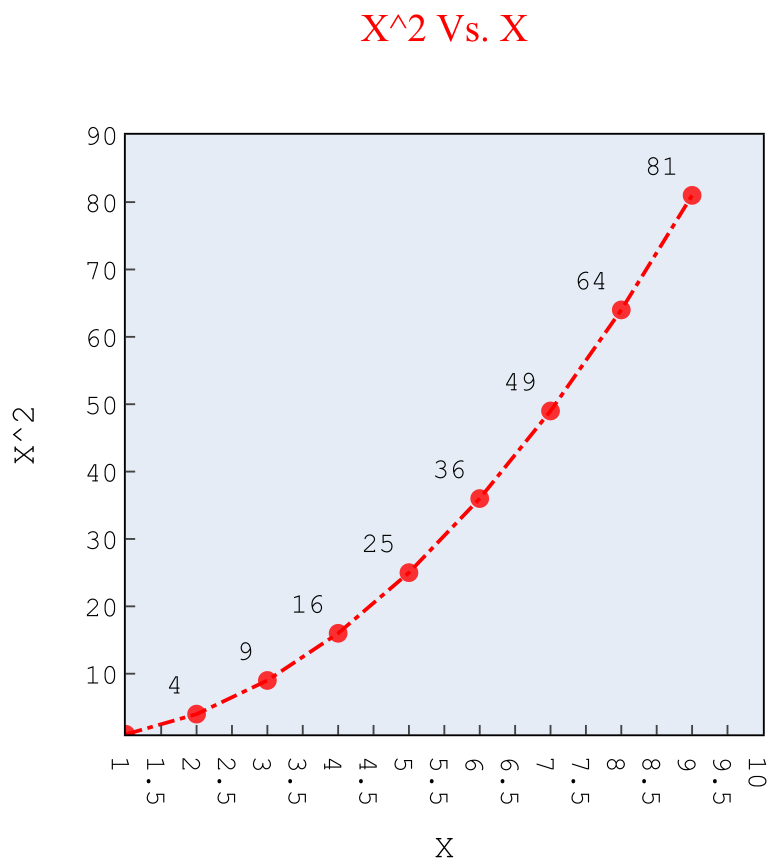
```
import plotly.express as px

X = [m for m in range(1,10)]
Y = [m**2 for m in X]

Plot1 = px.line(x=X, y=Y,title='X^2 Vs. X',
                labels=dict(x='X',y='X^2'),
                range_x=(1,10), range_y=(1,90),
                height=500,width=500,
                markers=True,
                text=Y)

Plot1.update_traces(marker = dict(size =10, opacity = 0.8,color='green'),
                    textposition='top left')

'''textposition parameter is used to adjust position of datapoint labels
   possible values are ['top left', 'top center', 'top right', 'middle left',
            'middle center', 'middle right', 'bottom left', 'bottom
            center', 'bottom right']'''

Plot1.show()
```



3D. LINE PLOT 4

- Formatting PLOT LINE
- Formatting figure layout

```
import plotly.express as px

X = [m for m in range(1,10)]
Y = [m**2 for m in X]

Plot1 = px.line(x=X, y=Y,title='X^2 Vs. X',
                labels=dict(x='X',y='X^2'),
                range_x=(1,10), range_y=(1,90),
                height=500,width=500,
                markers=True,
                text=Y)

Plot1.update_traces(textposition='top left',
                  marker = dict(size =10, opacity = 0.8,color='red'),
                  line = dict(dash = 'dashdot', width = 2,color='red'))

'''dash parameter is used to set line style
   Possible values: ['solid', 'dot', 'dash', 'longdash', 'dashdot', 'longdashdot']'''

Plot1.update_layout(plot_bgcolor='light grey',
                    paper_bgcolor='lightgreen',
                    xaxis_showgrid=False, yaxis_showgrid=False,
                    xaxis_showline=True, xaxis_linecolor='black',
                    yaxis_showline=True, yaxis_linecolor='black',
                    xaxis_mirror=True,yaxis_mirror=True,
                    xaxis_ticks='inside',yaxis_ticks='inside',
                    xaxis_tickmode='linear',xaxis_dtick=0.5,
                    yaxis_tickmode='linear',yaxis_dtick=10,
                    showlegend=True,
                    legend_title_font_color="black",
                    legend_title_text='X^2 Vs. X',
                    font_family="Courier New",
                    font_size=15,
                    font_color="black",
                    title_font_family="Times New Roman",
                    title_font_color="red",
                    title_x=0.5)
```

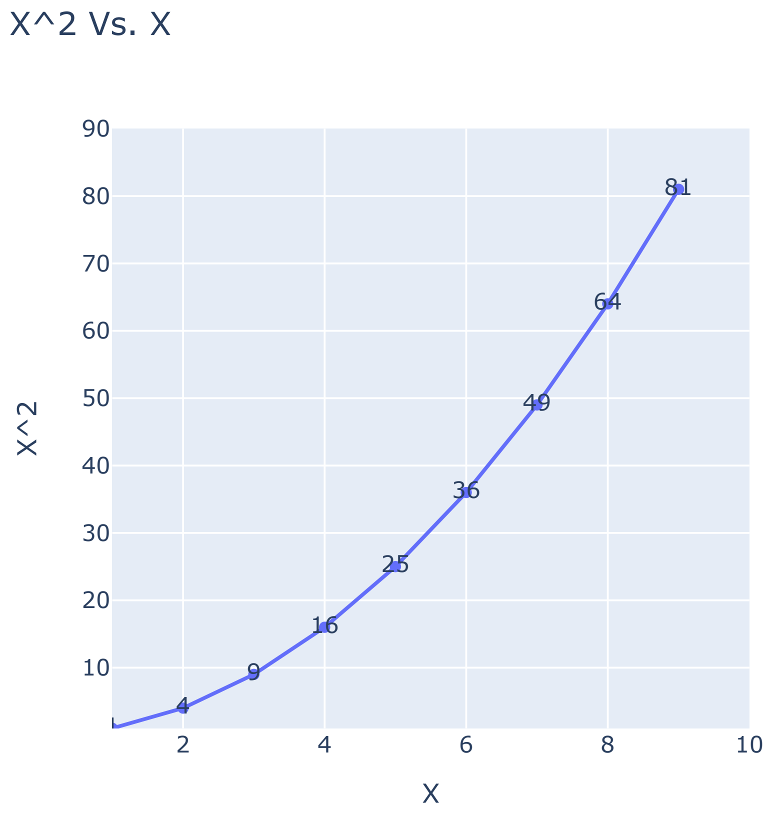


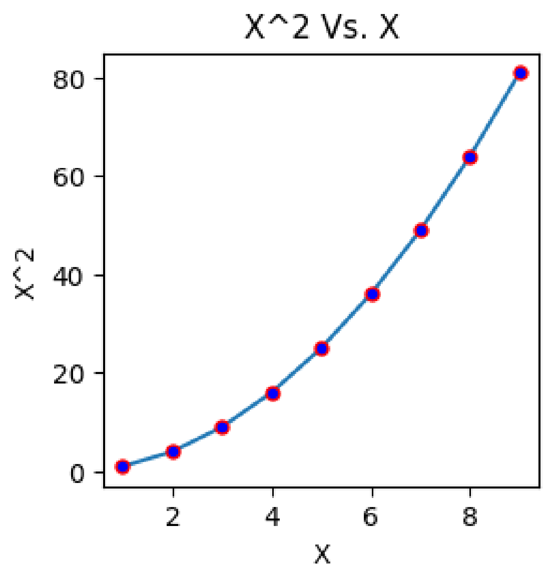
3E. LINE PLOT 5

- Comparing syntax for LINE PLOTs using Plotly, Matplotlib and Seaborn

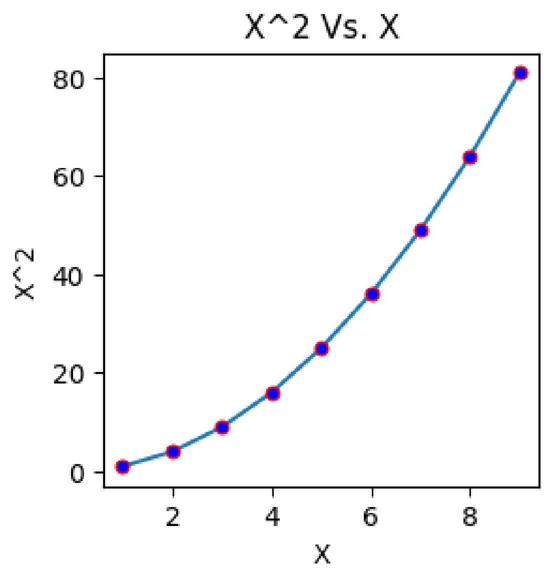
```
import plotly.express as px
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

X = [m for m in range(1,10)]
Y = [m**2 for m in X]

# Using Plotly

Plot1 = px.line(x=X, y=Y,title='X^2 Vs. X',
        labels=dict(x='X',y='X^2'),
        range_x=(1,10), range_y=(1,90),
        height=500,width=500,
        markers=True,
        text=Y)
```

```
Plot1.show()

# Using Matplotlib

plt.figure(figsize=(3,3))
plt.plot(X,Y,marker = 'o', ms = 5, mec = 'r',mfc='b')

plt.title('X^2 Vs. X')
plt.xlabel('X')
plt.ylabel('X^2')

plt.show()

# Using Seaborn

plt.figure(figsize=(3,3))

sns.lineplot(x=X, y=Y,marker='o', ms = 5, mec = 'r',mfc='b')

plt.title('X^2 Vs. X')
plt.xlabel('X')
plt.ylabel('X^2')
```





```
Text(0, 0.5, 'X^2')
```



3F. LINE PLOT 6

- Loading Dataset from an external file, For e.g. IRIS Dataset
- Classifying LINE plots according to 'Species'

```
# Loading dataset from an external .csv file

import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

display(IRIS_DATA)
```

	S.No.	Sepal Length	Sepal Width	Petal Length	Petal Width	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
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
# Loading an inbuilt example dataset file

import plotly.express as px

TIPS_DATA=px.data.tips()

display(TIPS_DATA)
```

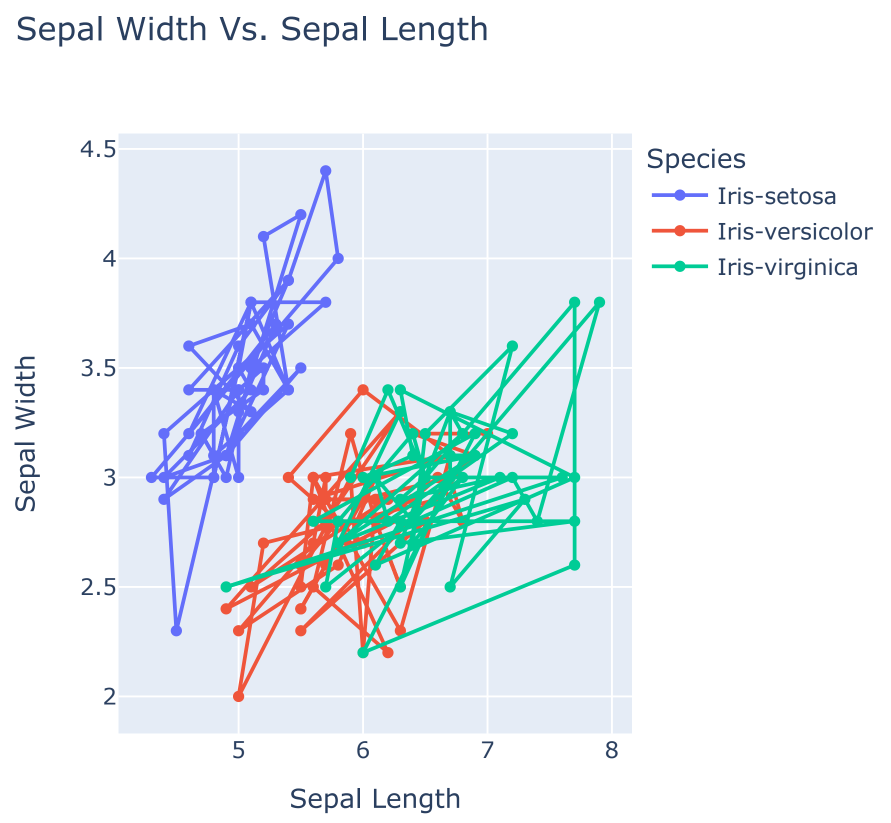
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

3F. LINE PLOT 6

- Loading Dataset from an external file, For e.g. IRIS Dataset
- Classifying LINE plots according to 'Species'

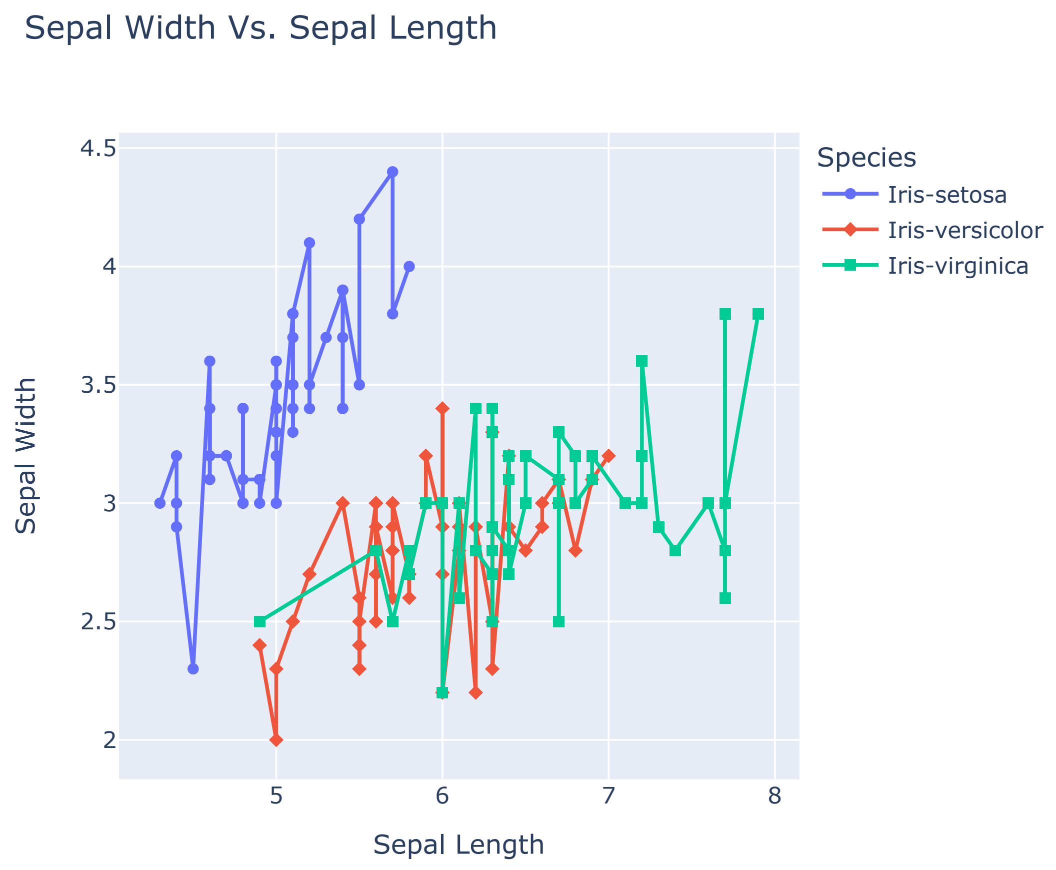
```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

Plot2 = px.line(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              markers=True,
              color='Species',
              title='Sepal Width Vs. Sepal Length',
              height=500,width=500)

Plot2.show()
```



3G. LINE PLOT 7

- Loading Dataset from an external file
- Reordering datapoints according to 'Sepal Length' in ascending order
- Displaying markers
- Using different 'marker symbols' for different 'Species'

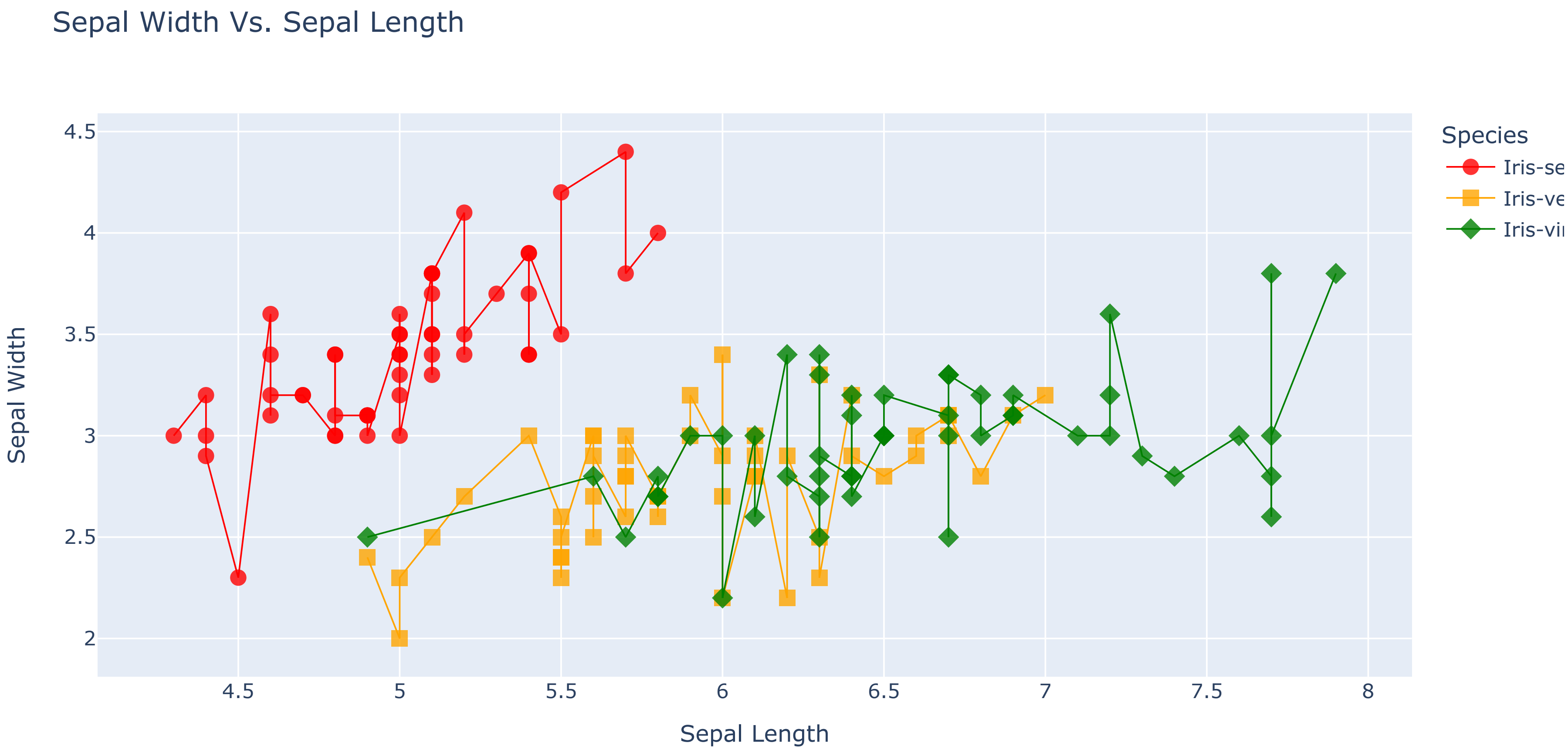
```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

IRIS_DATA = IRIS_DATA.sort_values(by='Sepal Length')

Plot2 = px.line(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',
              markers ='True', symbol='Species',
              title='Sepal Width Vs. Sepal Length')

Plot2.show()
```



3H. LINE PLOT 8

- Loading Dataset from an external file
- Reordering datapoints according to 'Sepal Length' in ascending order
- Displaying markers
- assigning colors to different plots
- Formatting markers

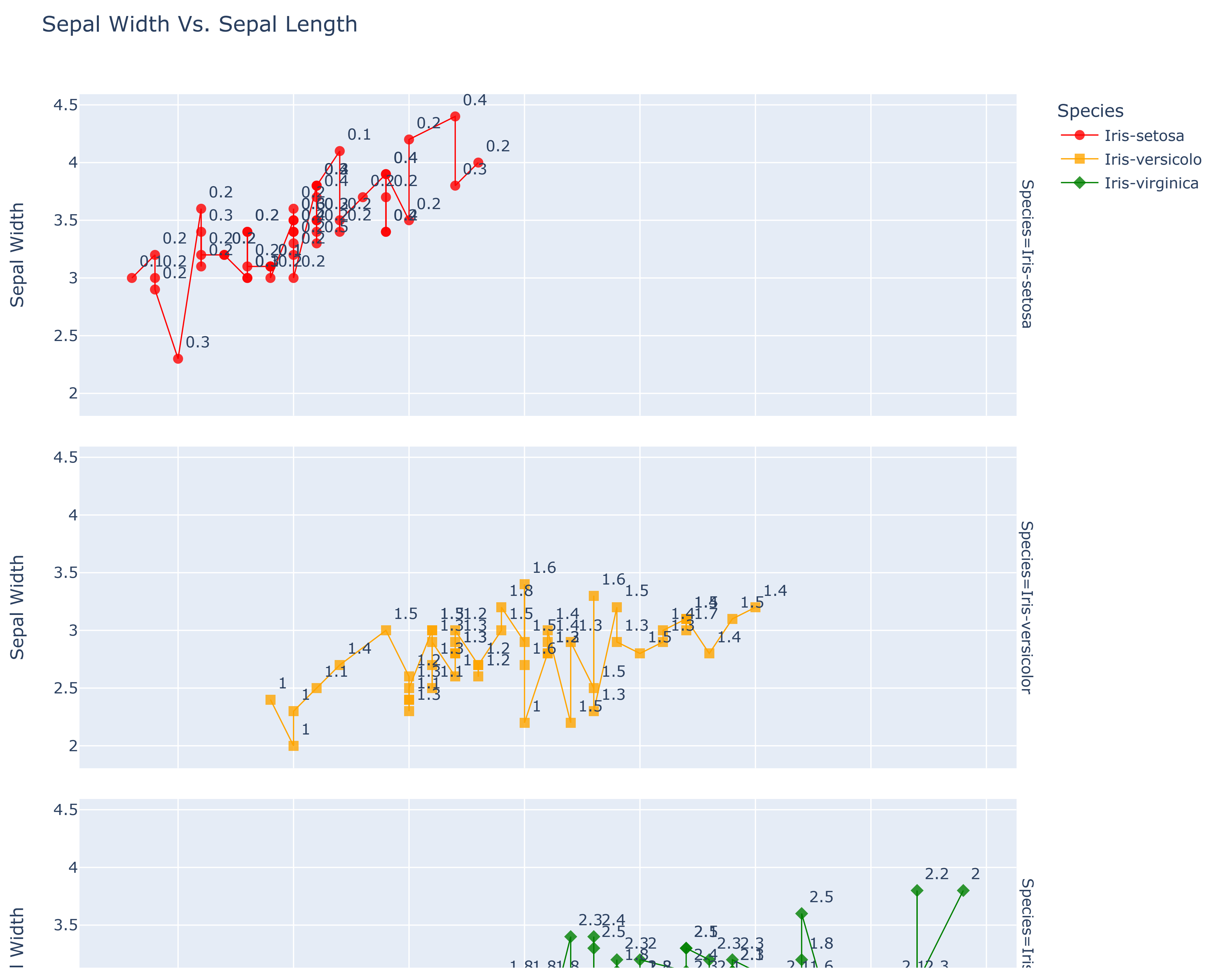
```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

IRIS_DATA = IRIS_DATA.sort_values(by='Sepal Length')

Plot2 = px.line(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',color_discrete_sequence = ['red','orange','green'],
              markers ='True',
              symbol='Species',symbol_sequence=['circle','square','diamond'],
              title='Sepal Width Vs. Sepal Length')

Plot2.update_traces(line = dict(dash = 'solid', width = 1),
                  marker = dict(size =10, opacity = 0.8))
Plot2.show()
```



MULTI-GRID (FACET) PLOTS

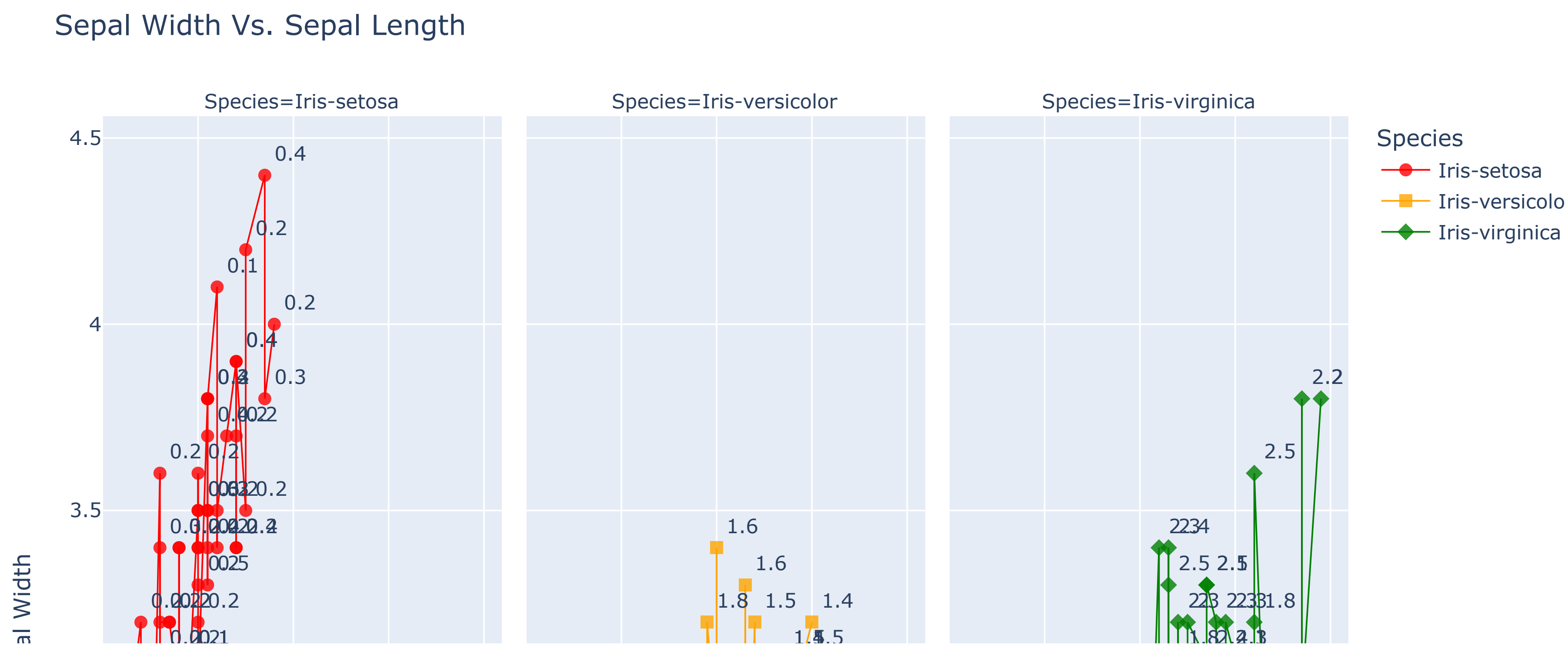
3I. LINE PLOT 9

```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

IRIS_DATA = IRIS_DATA.sort_values(by='Sepal Length')

Plot2 = px.line(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',color_discrete_sequence = ['red','orange','green'],
              markers ='True',
              symbol='Species',symbol_sequence=['circle','square','diamond'],
              title='Sepal Width Vs. Sepal Length',
              facet_row='Species',
              text='Petal Width',
              height=1000, width=1000)

Plot2.update_traces(line = dict(dash = 'solid', width = 1),
                  marker = dict(size =8, opacity = 0.8),
                  textposition='top right')
Plot2.show()
```



3J. LINE PLOT 10

```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

IRIS_DATA = IRIS_DATA.sort_values(by='Sepal Length')

fig = px.line(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',color_discrete_sequence = ['red','orange','green'],
              markers ='True',
              symbol='Species',symbol_sequence=['circle','square','diamond'],
              title='Sepal Width Vs. Sepal Length',
              facet_col='Species',
              text='Petal Width',
              height=800, width=1000)

fig.update_traces(line = dict(dash = 'solid', width = 1),
                  marker = dict(size =8, opacity = 0.8),
                  textposition='top right')
fig.show()
```

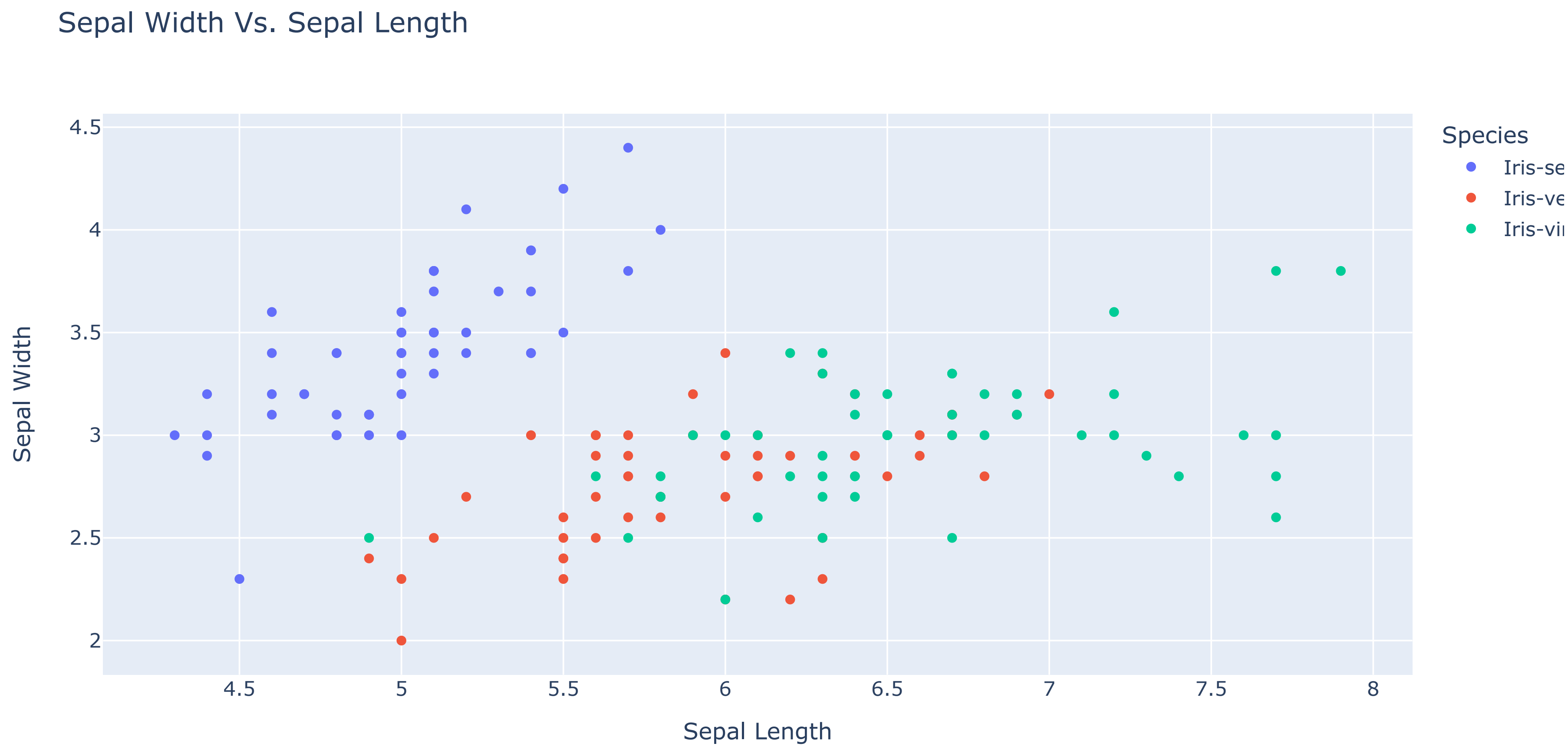


4. SCATTER PLOT

4A. SCATTER PLOT 1

- Loading IRIS Dataset from an external file
- Classifying scatter plots according to 'Species'

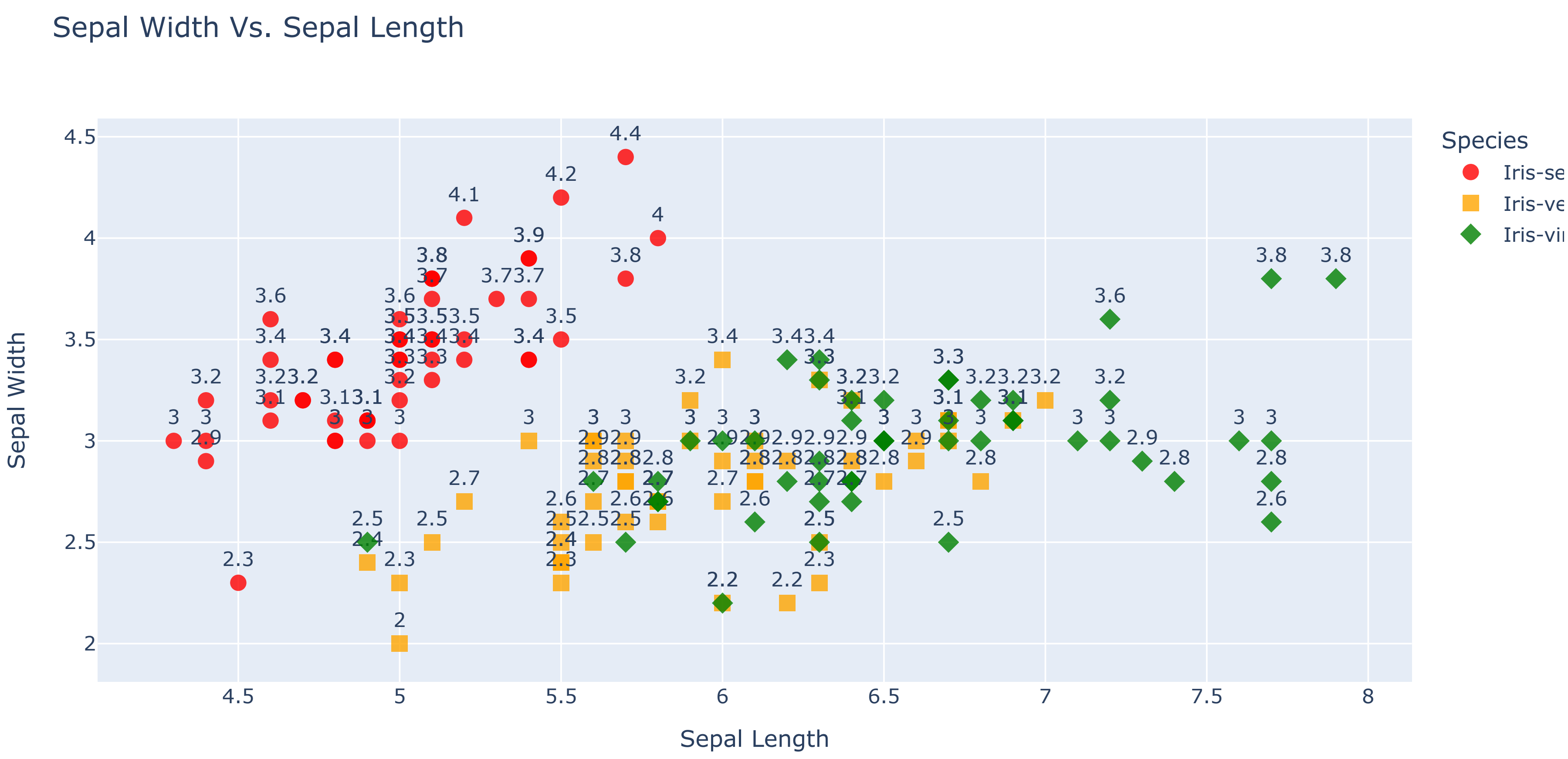
```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

Plot4 = px.scatter(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',
              title='Sepal Width Vs. Sepal Length')

Plot4.show()
```



4B. SCATTER PLOT 2

- Loading IRIS Dataset from an external file
- Formatting data point markers

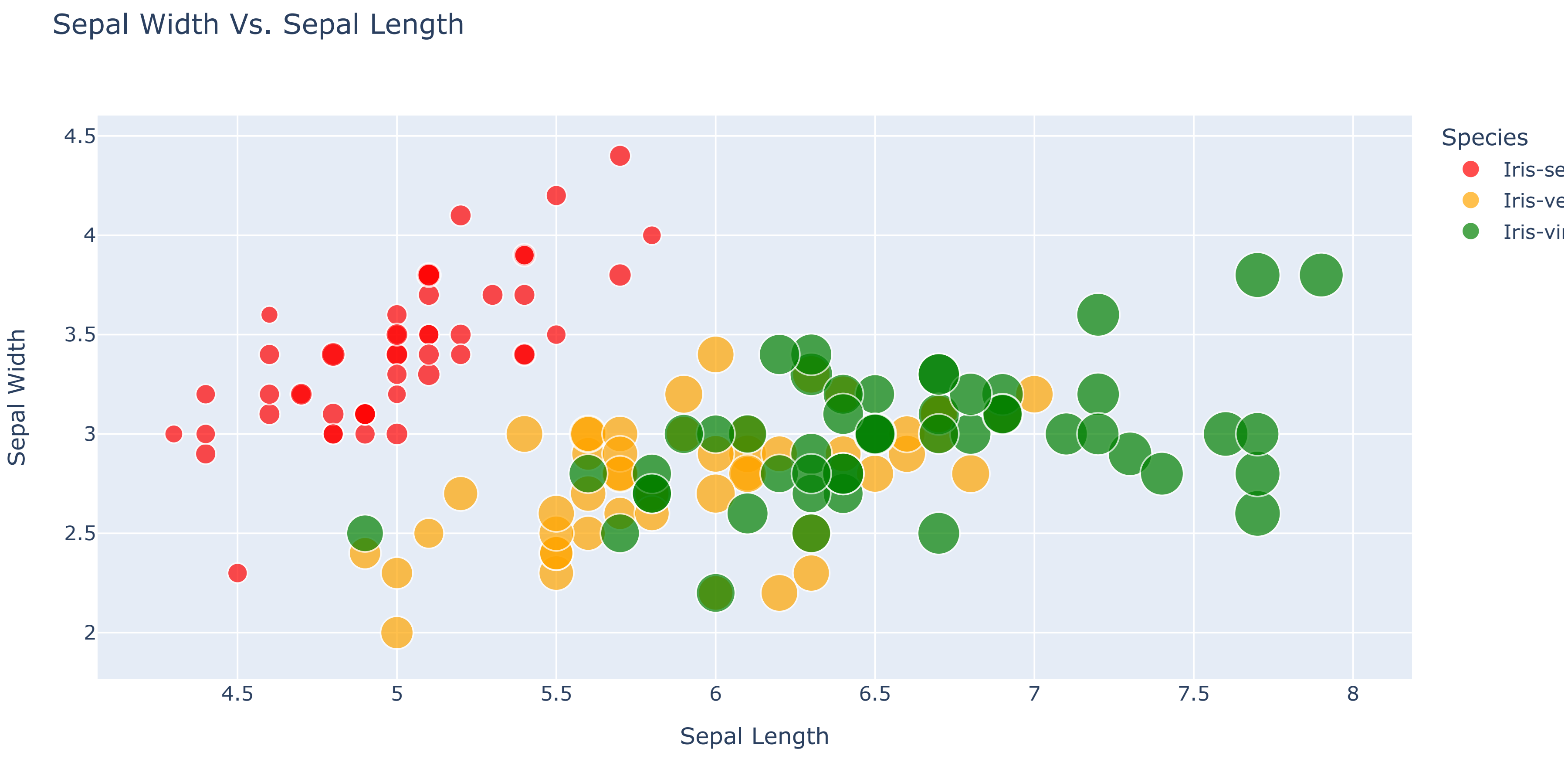
```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

Plot4 = px.scatter(IRIS_DATA, x='Sepal Length', y='Sepal Width',
              color='Species',color_discrete_sequence = ['red','orange','green'],
              symbol='Species',symbol_sequence=['circle','square','diamond'],
              title='Sepal Width Vs. Sepal Length',
              text='Sepal Width')

Plot4.update_traces(textposition='top center',
                  marker = dict(size =10, opacity = 0.8))
Plot4.show()
```



4C. SCATTER PLOT 3

- Loading IRIS Dataset from an external file
- Formatting data point markers
- Different Marker size for different 'Species'

```
import plotly.express as px
import pandas as pd

IRIS_DATA = pd.read_csv('Iris.csv')

Plot4 = px.scatter(IRIS_DATA, x='Sepal Length', y='Sepal Width',
                 color='Species',color_discrete_sequence = ['red','orange','green'],
                 size='Petal Length',
                 title='Sepal Width Vs. Sepal Length')

Plot4.show()
```


Sepal Width Vs. Sepal Length
Sepal Width
Sepal Length
Species
Iris-se
Iris-ve
Iris-vi