ASBD Lab 2

CED19I028

In [80]:
```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import pairwise_distances
import squarify

random.seed(100)
```

In [32]:
```
df = pd.read_csv('Iris.csv')
df
```

Out[32]:

	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
...	...	...	...	...	...	...
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

In [3]:
```
df['Species'].unique()
```

Out[3]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

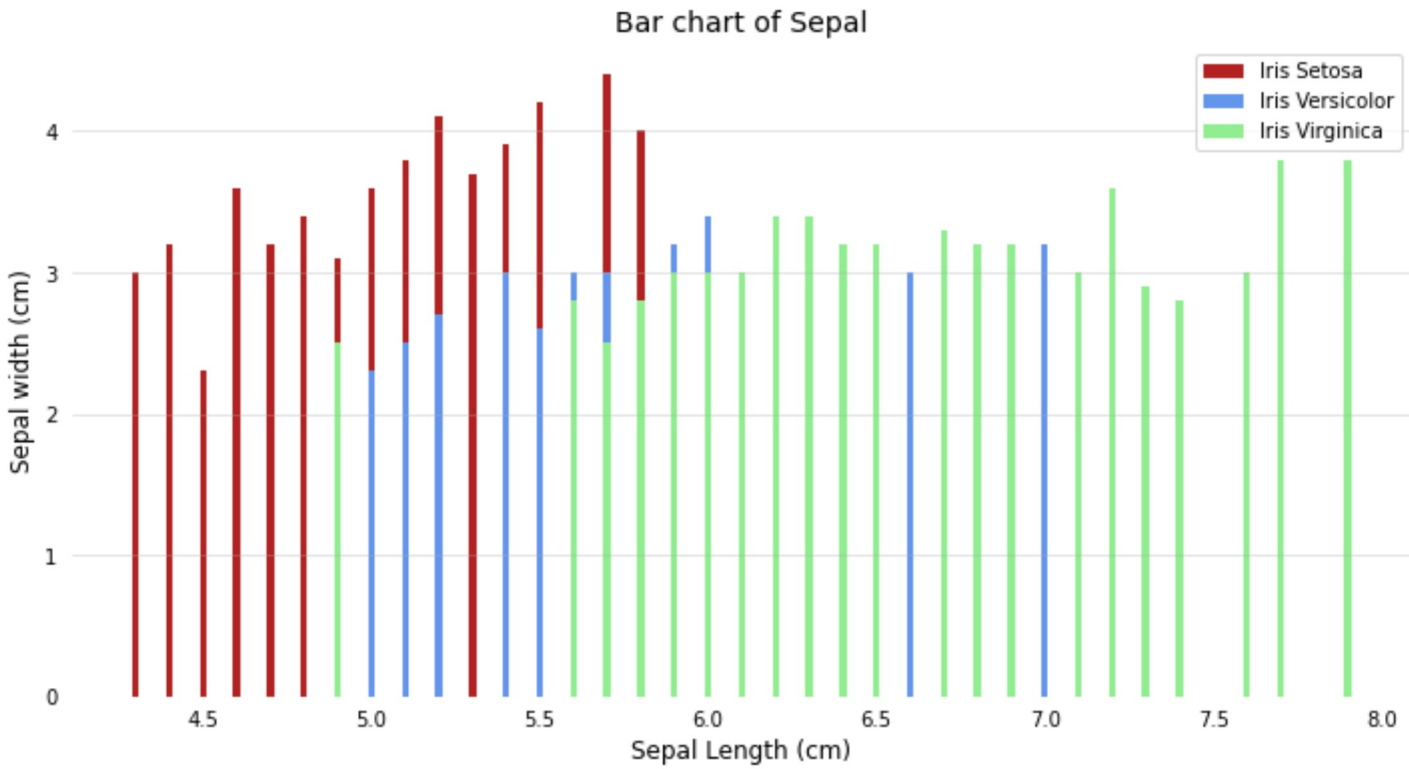
In [48]:
```
Iris_setosa = df[:50]
Iris_versicolor = df[50:100]
Iris_virginica = df[100:150]
```

1. Consider a subset of attributes in the given dataset and apply the below given visualization plots in any platform of your choice (Microsoft Excel / LibreOffice Calc / Python(preferred) / R).

1. Bar/Column Chart
2. Pie Chart
3. Doughnut chart
4. Pareto Chart
5. Scatter plot
6. Line Chart
7. Radar Chart
8. Area Chart
9. Histogram

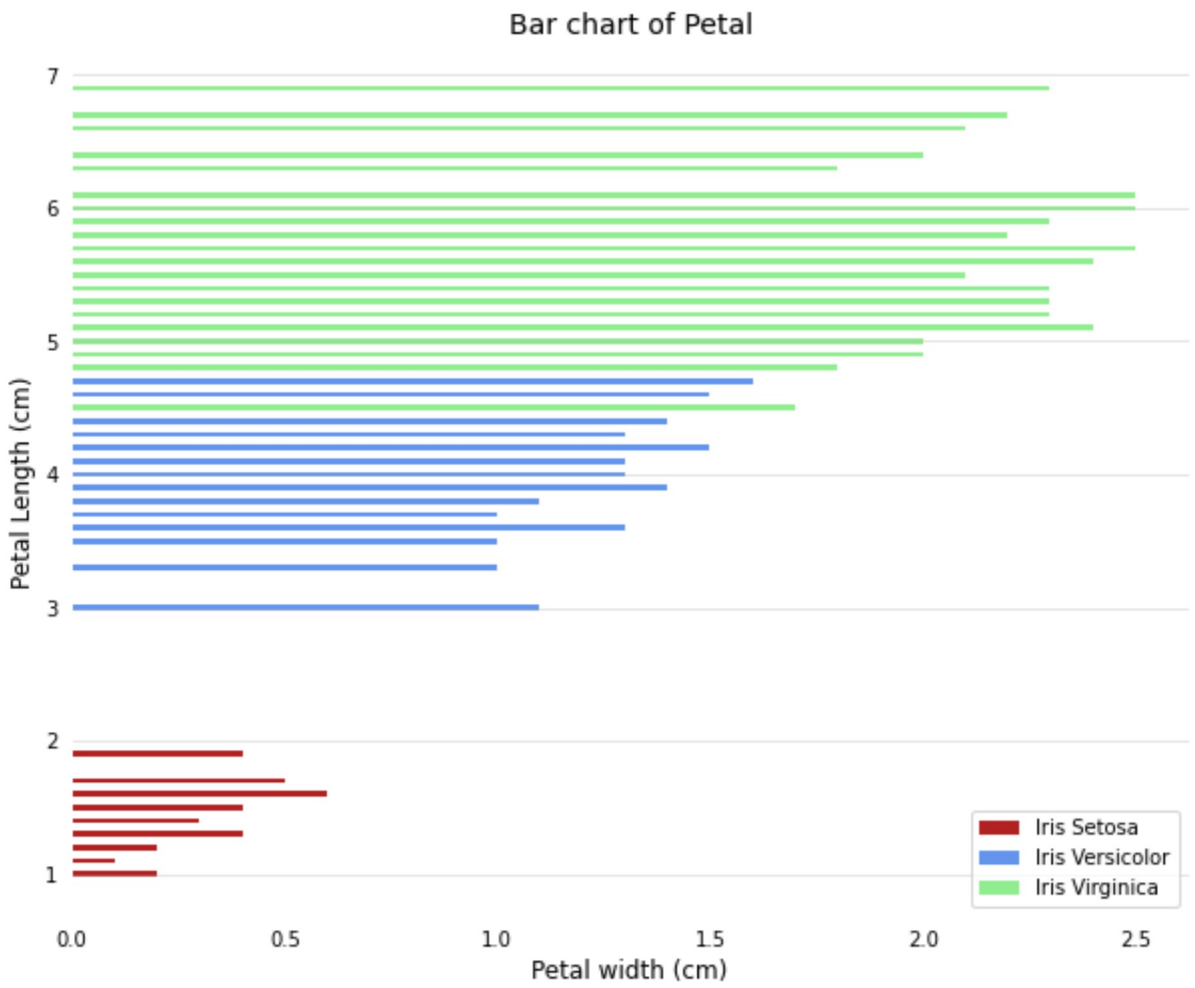
In [ ]:

In [5]:
```
Iris_setosa_sepal = Iris_setosa.iloc[:,1:3]
Iris_versicolor_sepal = Iris_versicolor.iloc[:,1:3]
Iris_virginica_sepal = Iris_virginica.iloc[:,1:3]

plt.figure(figsize=(12,6))
plt.bar(Iris_setosa_sepal.iloc[:,0].values, Iris_setosa_sepal.iloc[:,1].values, color='firebrick', label="Iris Setosa", width=
plt.bar(Iris_versicolor_sepal.iloc[:,0].values, Iris_versicolor_sepal.iloc[:,1].values, color='cornflowerblue', label="Iris Ve
plt.bar(Iris_virginica_sepal.iloc[:,0].values, Iris_virginica_sepal.iloc[:,1].values, color='lightgreen', label="Iris Virginic

plt.xlabel("Sepal Length (cm)", size=12)
plt.ylabel("Sepal width (cm)", size=12)

plt.grid(axis ='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)

for spine in plt.gca().spines.values():
    spine.set_visible(False)

plt.title("Bar chart of Sepal", size=14)
plt.legend()
plt.show()
```


Bar chart of Sepal
Iris Setosa
Iris Versicolor
Iris Virginica
Sepal width (cm)
0
1
2
3
4
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal Length (cm)

In [6]:

```
# horizontal... petal length

Iris_setosa_petal = Iris_setosa.iloc[:,3:5]
Iris_versicolor_petal = Iris_versicolor.iloc[:,3:5]
Iris_virginica_petal = Iris_virginica.iloc[:,3:5]

plt.figure(figsize=(10,8))
plt.barh(Iris_setosa_petal.iloc[:,0].values, Iris_setosa_petal.iloc[:,1].values, color='firebrick', label="Iris Setosa", align
plt.barh(Iris_versicolor_petal.iloc[:,0].values, Iris_versicolor_petal.iloc[:,1].values, color='cornflowerblue', label="Iris V
plt.barh(Iris_virginica_petal.iloc[:,0].values, Iris_virginica_petal.iloc[:,1].values, color='lightgreen', label="Iris Virgini

plt.ylabel("Petal Length (cm)", size=12)
plt.xlabel("Petal width (cm)", size=12)

plt.grid(axis ='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)

for spine in plt.gca().spines.values():
    spine.set_visible(False)

plt.title("Bar chart of Petal", size=14)
plt.legend()
plt.show()
```

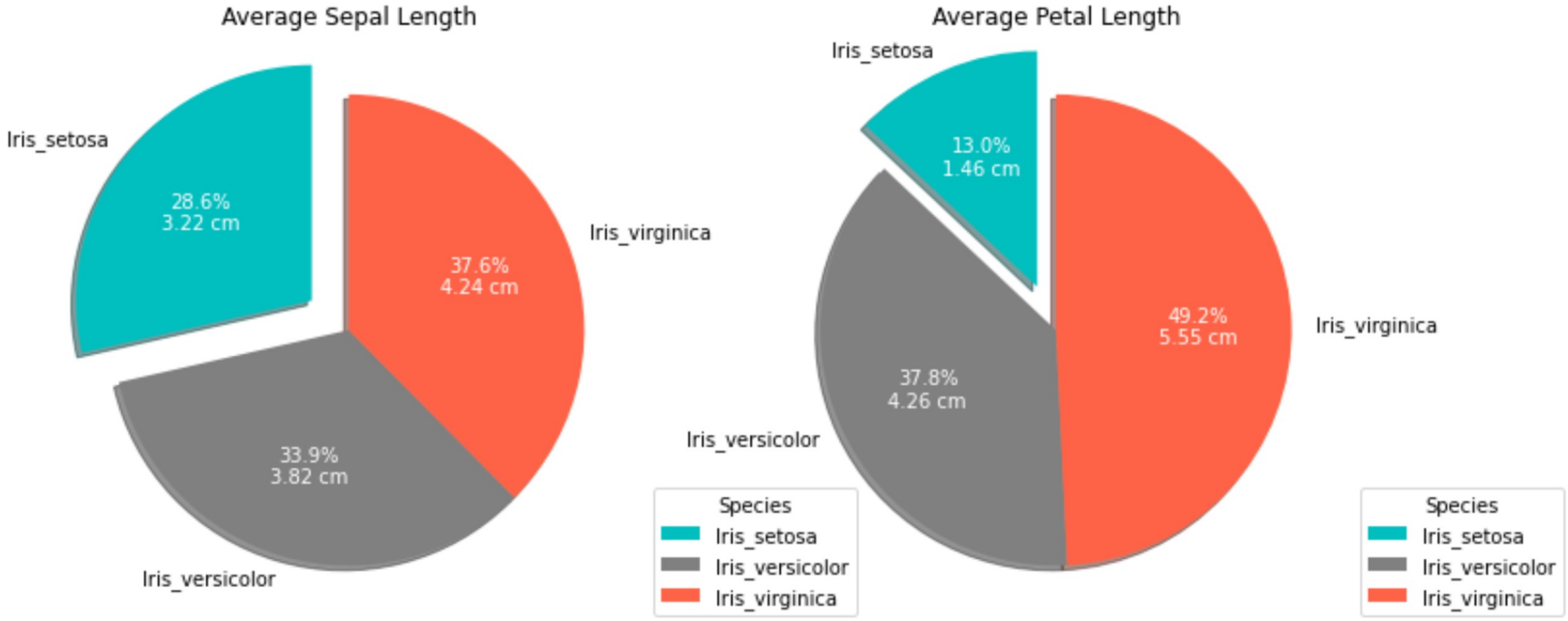

Bar chart of Petal
Petal Length (cm)
7
6
5
4
3
2
1
0.0
0.5
1.0
1.5
2.0
2.5
Petal width (cm)
Iris Setosa
Iris Versicolor
Iris Virginica

In [ ]:

In [9]:

```
petalAvg = (np.mean(Iris_setosa['PetalLengthCm'].values),
            np.mean(Iris_versicolor['PetalLengthCm'].values),
            np.mean(Iris_virginica['PetalLengthCm'].values))

def func(pct, ptalAvg):
    absolute = pct/100.*np.sum(ptalAvg)
    return "{:.1f}%\n{:.2f} cm".format(pct, absolute)

def plotGraph(title, data, names):
    explodes = (0.2, 0, 0)
    colors = ("c", "grey", "tomato")
    _, _, autotexts = plt.pie(data,
         labels = names,
         explode = explodes,
          shadow = True,
          colors = colors,
          startangle = 90,
          autopct=lambda pct: func(pct, petalAvg)
         )

    for autotext in autotexts:
        autotext.set_color('white')

    plt.legend(names,
             title="Species",
             loc="lower left",
             bbox_to_anchor=(1,0,0.5,1))

    plt.title(title)


names = ('Iris_setosa', 'Iris_versicolor', 'Iris_virginica')
sepalAvg = (np.mean(Iris_setosa['SepalLengthCm'].values),
            np.mean(Iris_versicolor['SepalLengthCm'].values),
            np.mean(Iris_virginica['SepalLengthCm'].values))


plt.figure(figsize=(12,8))
plt.subplot(1, 2, 1)
plotGraph('Average Sepal Length',sepalAvg, names)

plt.subplot(1, 2, 2)
```

```
plotGraph('Average Petal Length',petalAvg, names)
```

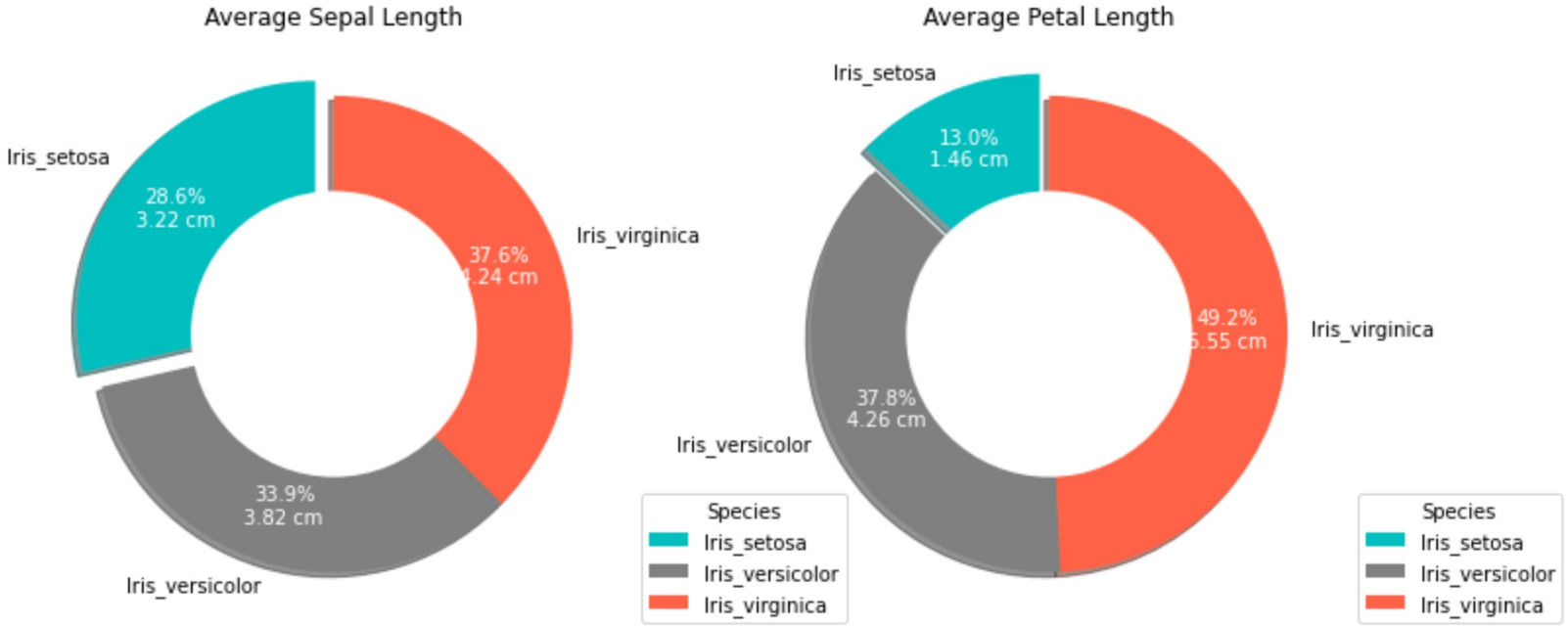


In [ ]:

In [12]:

```
petalAvg = (np.mean(Iris_setosa['PetalLengthCm'].values),
            np.mean(Iris_versicolor['PetalLengthCm'].values),
            np.mean(Iris_virginica['PetalLengthCm'].values))

def func(pct, ptalAvg):
    absolute = pct/100.*np.sum(ptalAvg)
    return "{:.1f}%\n{:.2f} cm".format(pct, absolute)

def plotGraph(title, data, names):
    explodes = (0.1, 0, 0)
    colors = ("c", "grey", "tomato")
    _, _, autotexts = plt.pie(data,
         labels = names,
         explode = explodes,
          shadow = True,
          colors = colors,
          startangle = 90,
          autopct=lambda pct: func(pct, petalAvg),
          pctdistance=0.75
         )

    # draw circle
    centre_circle = plt.Circle((0, 0), 0.60, fc='white')
    fig = plt.gcf()

    # Adding Circle in Pie chart
    fig.gca().add_artist(centre_circle)


    for autotext in autotexts:
        autotext.set_color('white')

    plt.legend(names,
             title="Species",
             loc="lower left",
             bbox_to_anchor=(1,0,0.5,1))

    plt.title(title)


names = ('Iris_setosa', 'Iris_versicolor', 'Iris_virginica')
sepalAvg = (np.mean(Iris_setosa['SepalLengthCm'].values),
```

```
            np.mean(Iris_versicolor['SepalLengthCm'].values),
            np.mean(Iris_virginica['SepalLengthCm'].values))


plt.figure(figsize=(12,8))
plt.subplot(1, 2, 1)
plotGraph('Average Sepal Length',sepalAvg, names)

petalAvg = (np.mean(Iris_setosa['PetalLengthCm'].values),
            np.mean(Iris_versicolor['PetalLengthCm'].values),
            np.mean(Iris_virginica['PetalLengthCm'].values))
plt.subplot(1, 2, 2)
plotGraph('Average Petal Length',petalAvg, names)
```



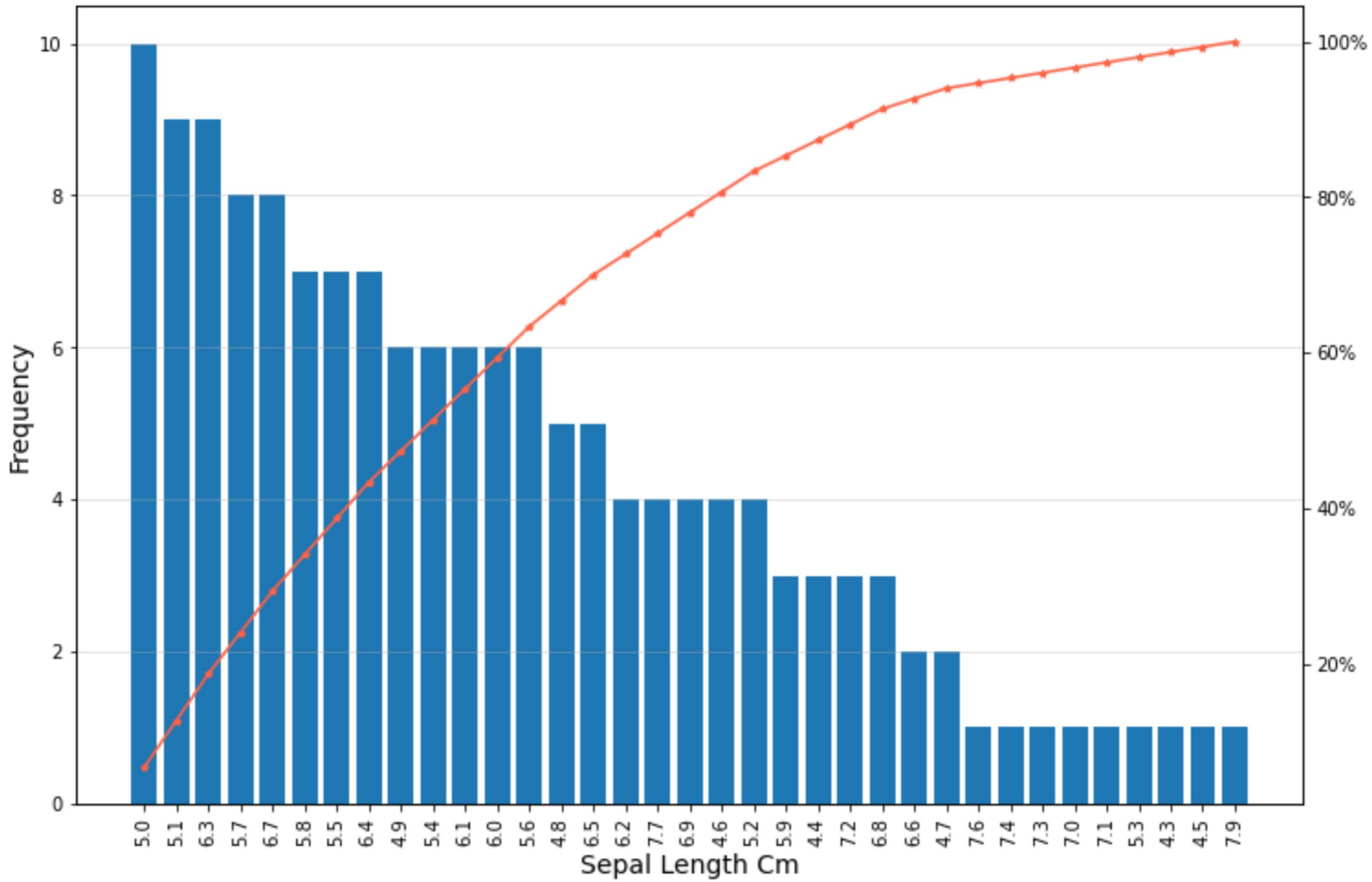
In [ ]:

In [ ]:

In [13]:

```
from matplotlib.ticker import PercentFormatter

a = df['SepalLengthCm'].value_counts()
x = []
y = []
for each in a.index:
    x.append(str(each));

for each in a:
    y.append(each);

#add column to display cumulative percentage
line = a.values.cumsum()/a.values.sum()*100

fig, ax = plt.subplots(figsize=(12,8))
plt.grid(axis ='y',alpha=0.4)

ax.bar(x, y)
ax.set_xticklabels(x,rotation=90)
ax.set_xlabel("Sepal Length Cm", size=14)
ax.set_ylabel("Frequency", size=14)

#add cumulative percentage line to plot
ax2 = ax.twinx()
ax2.plot(x, line, color="tomato", marker="*", ms=4)
ax2.yaxis.set_major_formatter(PercentFormatter())

plt.show()
```

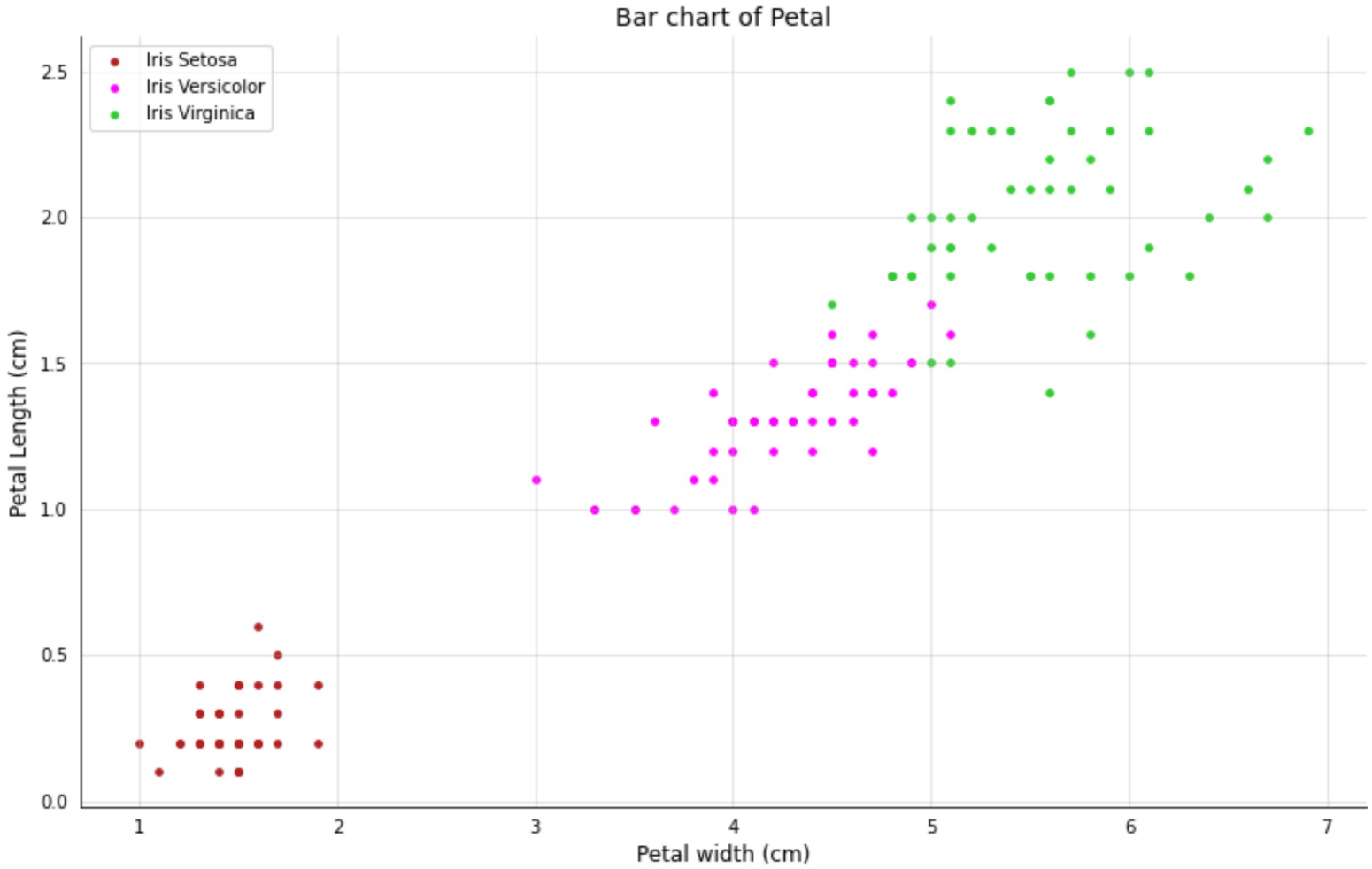
```
C:\Users\Sumit\AppData\Local\Temp\ipykernel_9100\3799624601.py:19: UserWarning: FixedFormatter should only be used together w
ith FixedLocator
  ax.set_xticklabels(x,rotation=90)
```


Frequency
Sepal Length Cm
0
2
4
6
8
10
20%
40%
60%
80%
100%
5.0
5.1
6.3
5.7
6.7
5.8
5.5
6.4
4.9
5.4
6.1
6.0
5.6
4.8
6.5
6.2
7.7
6.9
4.6
5.2
5.9
4.4
7.2
6.8
6.6
4.7
7.6
7.4
7.3
7.0
7.1
5.3
4.3
4.5
7.9

In [ ]:

In [18]:

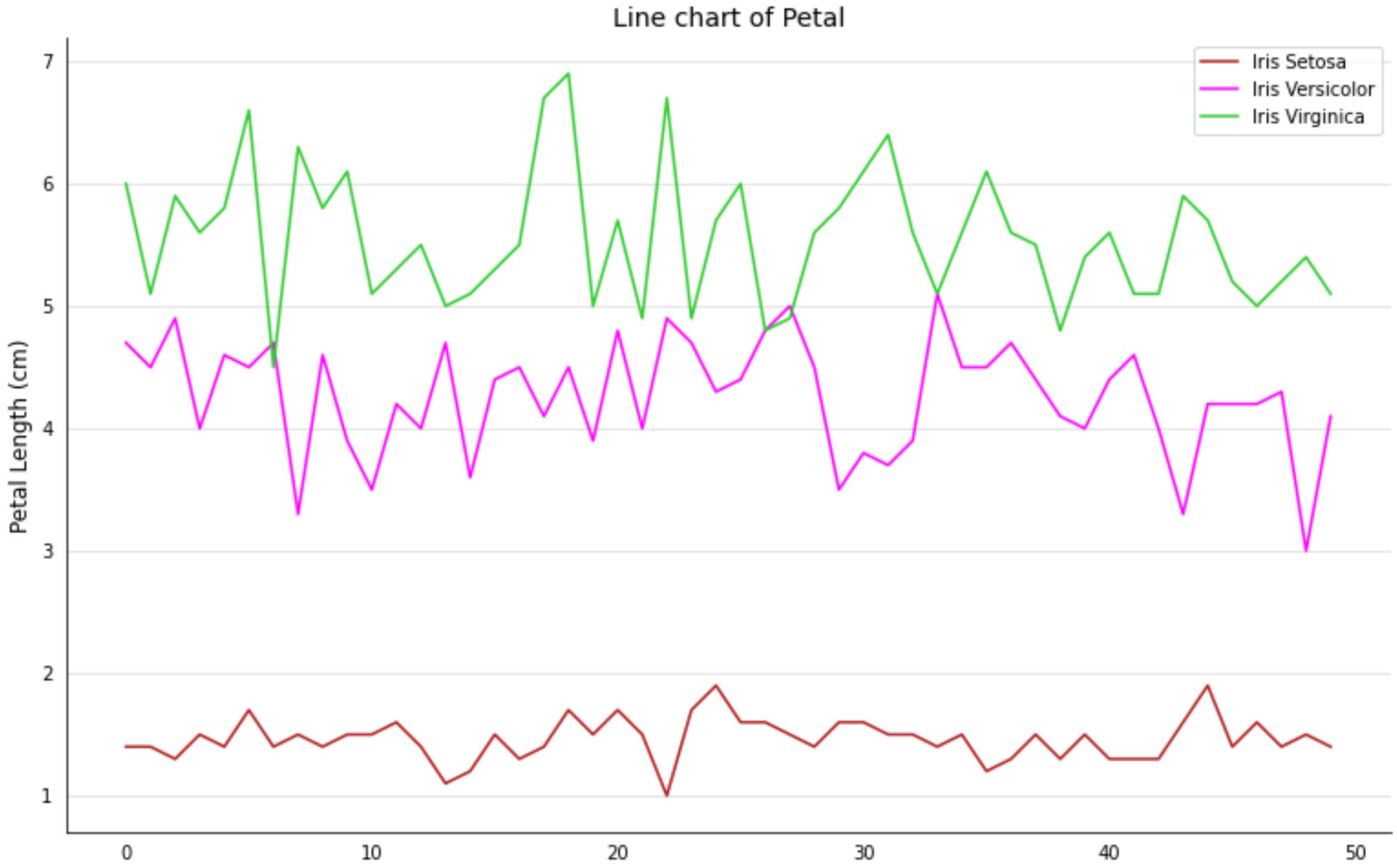
```
plt.figure(figsize=(13,8))
plt.scatter(Iris_setosa.iloc[:,3].values, Iris_setosa.iloc[:,4].values, color='firebrick', label="Iris Setosa", s=14)
plt.scatter(Iris_versicolor.iloc[:,3].values, Iris_versicolor.iloc[:,4].values, color='magenta', label="Iris Versicolor", s=14
plt.scatter(Iris_virginica.iloc[:,3].values, Iris_virginica.iloc[:,4].values, color='limegreen', label="Iris Virginica", s=14)

plt.ylabel("Petal Length (cm)", size=12)
plt.xlabel("Petal width (cm)", size=12)

plt.grid(alpha=0.4)
plt.tick_params(left = False, bottom=False)

count = 1;
for spine in plt.gca().spines.values():
    if(count&1):
        pass;
    else:
        spine.set_visible(False)
    count += 1

plt.title("Bar chart of Petal", size=14)
plt.legend()
plt.show()
```


Bar chart of Petal
Iris Setosa
Iris Versicolor
Iris Virginica
Petal Length (cm)
Petal width (cm)
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

In [ ]:

In [56]:

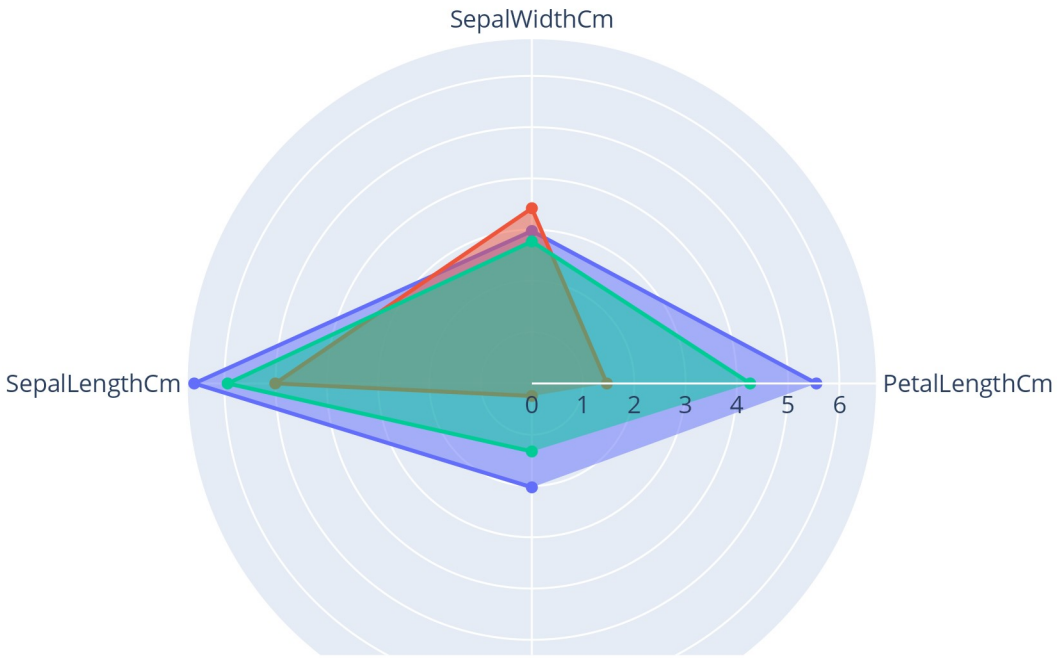
```
plt.figure(figsize=(13,8))
plt.plot(Iris_setosa.iloc[:,3].values, color='firebrick', label="Iris Setosa")
plt.plot(Iris_versicolor.iloc[:,3].values, color='magenta', label="Iris Versicolor")
plt.plot(Iris_virginica.iloc[:,3].values, color='limegreen', label="Iris Virginica")

plt.ylabel("Petal Length (cm)", size=12)

plt.grid(axis='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)

count = 1;
for spine in plt.gca().spines.values():
    if(count&1):
        pass;
    else:
        spine.set_visible(False)
    count += 1

plt.title("Line chart of Petal Length", size=14)
plt.legend()
plt.show()
```


Line chart of Petal
Iris Setosa
Iris Versicolor
Iris Virginica
Petal Length (cm)
7
6
5
4
3
2
1
0
10
20
30
40
50

In [ ]:

In [118…

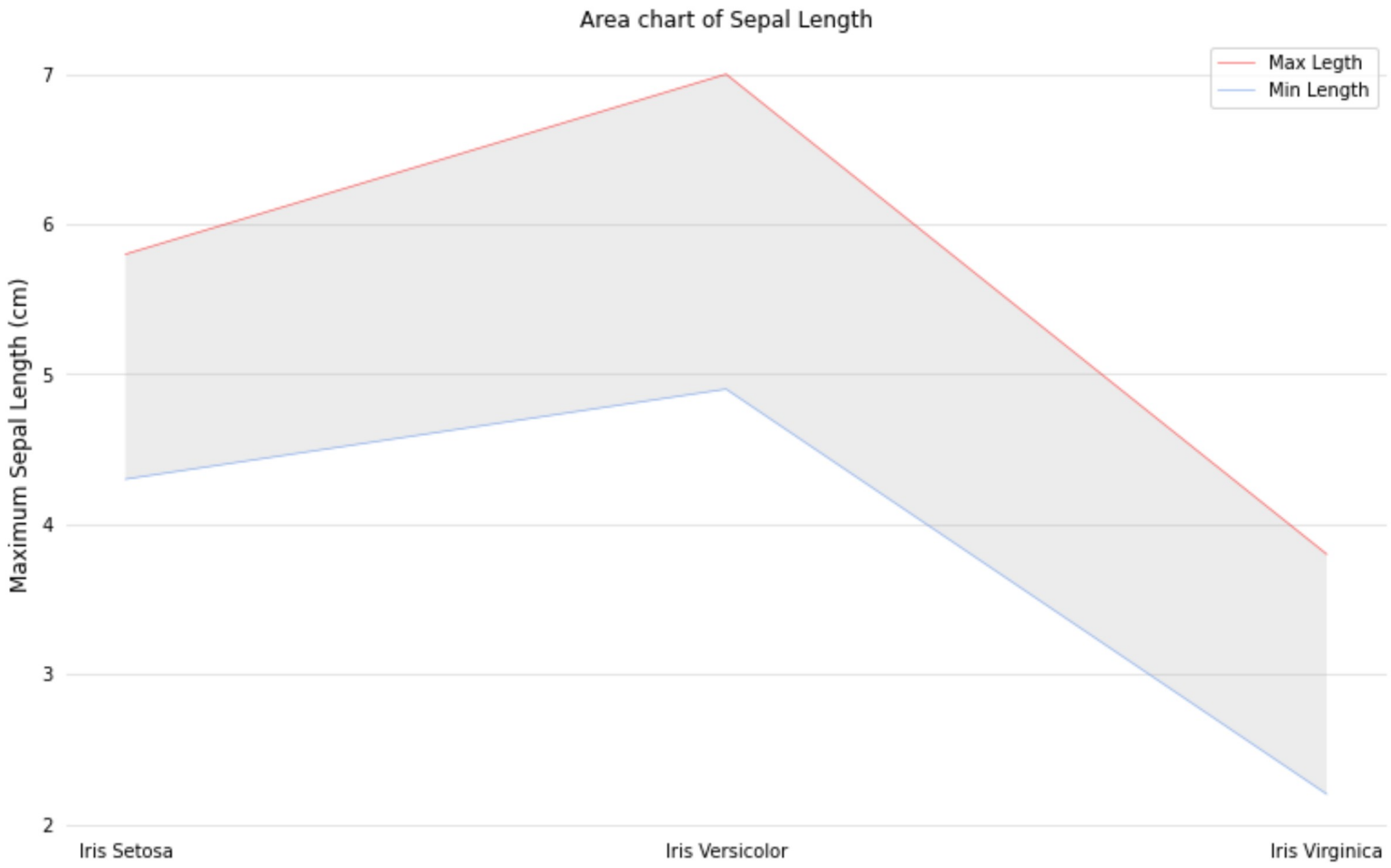
```
# radar chart...
import plotly.express as px
import plotly.graph_objects as go

# Iris_setosa = df[:50]
# Iris_versicolor = df[50:100]
# Iris_virginica = df[100:150]

virgi_mean = Iris_virginica[["PetalLengthCm","SepalWidthCm","SepalLengthCm","PetalWidthCm"]].mean().tolist()
setosa_mean= Iris_setosa[["PetalLengthCm","SepalWidthCm","SepalLengthCm","PetalWidthCm"]].mean().tolist()
versi_mean=  Iris_versicolor[["PetalLengthCm","SepalWidthCm","SepalLengthCm","PetalWidthCm"]].mean().tolist()

categories=["PetalLengthCm","SepalWidthCm","SepalLengthCm","PetalWidthCm",]
fig = go.Figure()
fig.add_trace(go.Scatterpolar(
    r = virgi_mean,
    theta=categories,
    # fill='toself',
    name='Iris-Virginica'
))

fig.add_trace(go.Scatterpolar(
    r = setosa_mean,
    theta = categories,
    # fill='toself',
    name='Iris-Setosa'
))

fig.add_trace(go.Scatterpolar(
    r = versi_mean,
    theta = categories,
    # fill='toself',
    name = 'Iris-Versicolor'
))

fig.update_traces(fill='toself')
fig.show()
```



In [ ]:

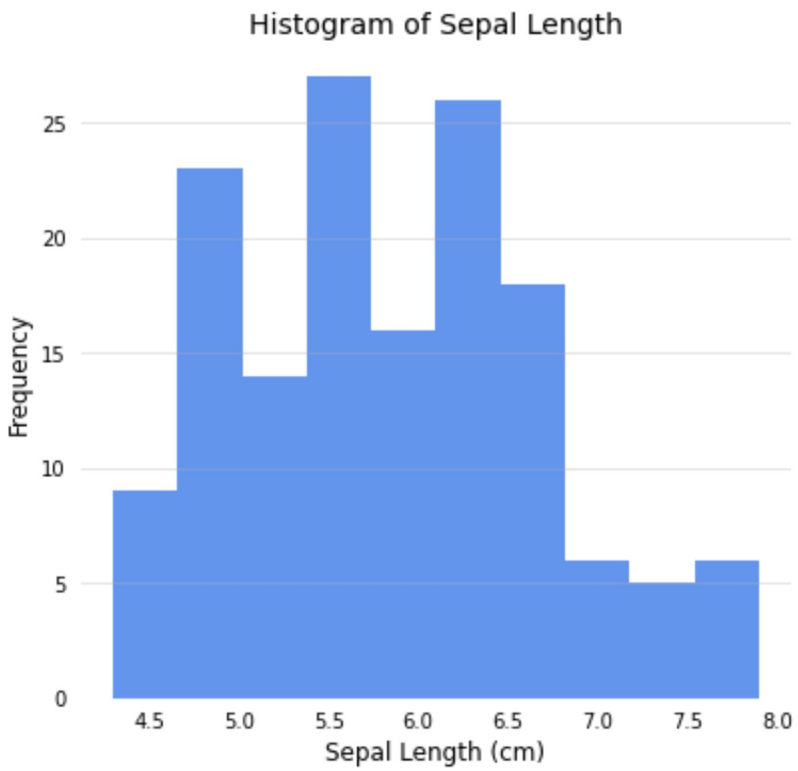
In [49]:

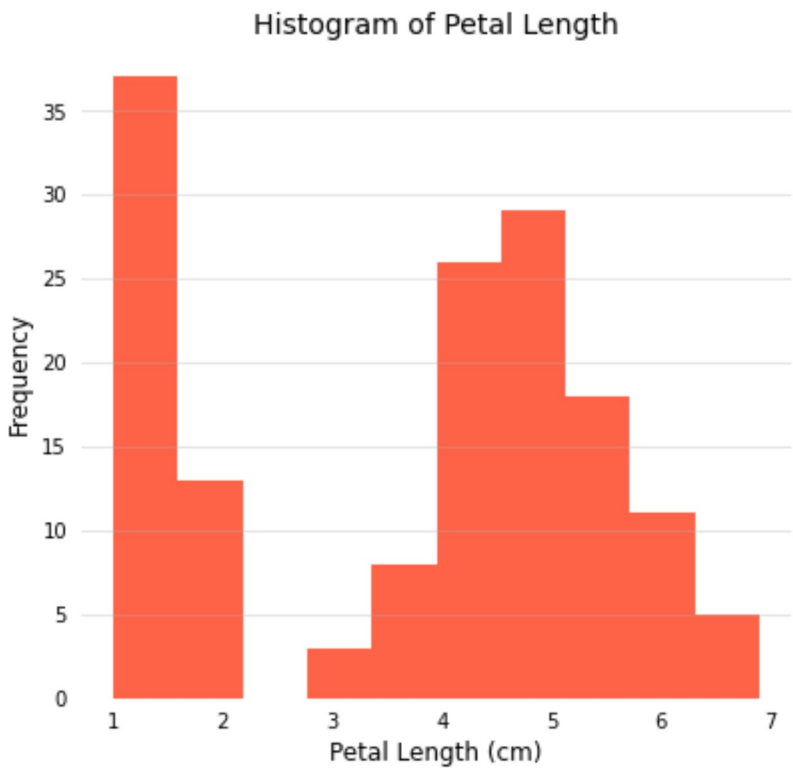
```
# Area Chart....

plt.figure(figsize=(13,8))

y_max = [Iris_setosa.iloc[:,1].values.max(), Iris_versicolor.iloc[:,1].values.max(), Iris_virginica.iloc[:,2].values.max()]
y_min = [Iris_setosa.iloc[:,1].values.min(), Iris_versicolor.iloc[:,1].values.min(), Iris_virginica.iloc[:,2].values.min()]

x = ["Iris Setosa", "Iris Versicolor", "Iris Virginica"]

plt.plot(x, y_max, c='red', alpha=0.6 , linewidth=0.7, label='Max Legth')
plt.plot(x, y_min, c='cornflowerblue', alpha=0.7, linewidth=0.7,label='Min Length')
plt.fill_between(range(len(x)), y_max, y_min, facecolor = 'gray', alpha = 0.15)

plt.title("Area chart of Sepal Length")
plt.ylabel("Maximum Sepal Length (cm)", size=12)

plt.grid(axis ='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)

for spine in plt.gca().spines.values():
    spine.set_visible(False)

plt.legend()
plt.show()
```


Area chart of Sepal Length
Max Legth
Min Length
Maximum Sepal Length (cm)
7
6
5
4
3
2
Iris Setosa
Iris Versicolor
Iris Virginica

In [ ]:

In [78]:
```
# histogram...


def plotGraph(index, name, color):
    plt.hist(df.iloc[:,index].values, color=color)
    plt.xlabel(f"{name} Length (cm)", size=12)
    plt.ylabel("Frequency", size=12)

    plt.grid(axis ='y',alpha=0.4)
    plt.tick_params(left = False, bottom=False)

    for spine in plt.gca().spines.values():
        spine.set_visible(False)

    plt.title(f"Histogram of {name} Length", size=14)




plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
plotGraph(1, "Sepal", 'cornflowerblue');

plt.subplot(1,2,2)
plotGraph(3, "Petal", 'tomato');
```



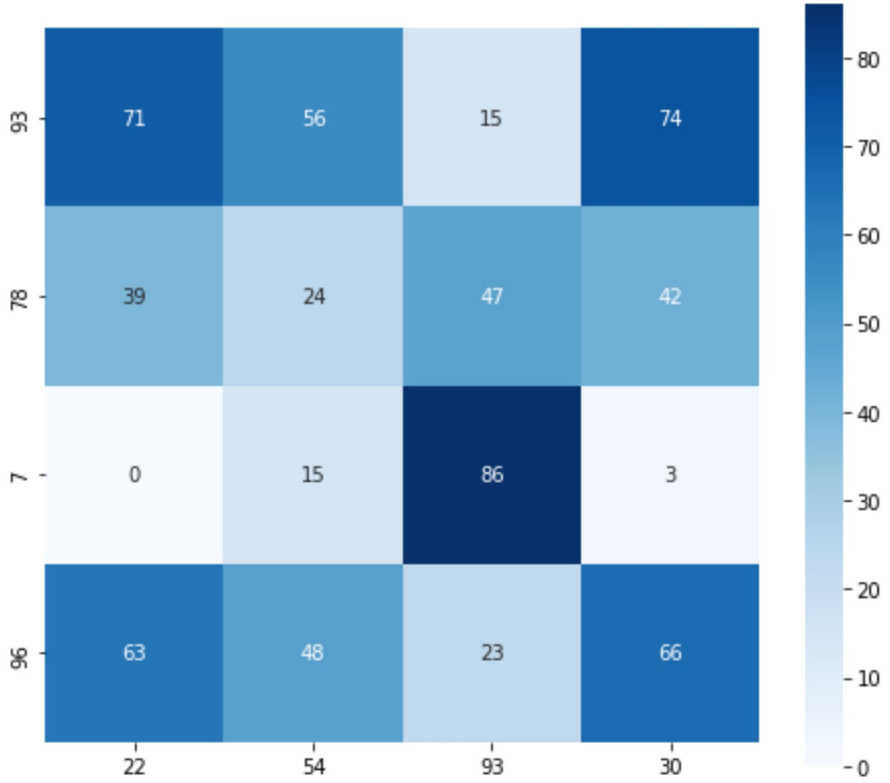


In [ ]:

In [ ]:

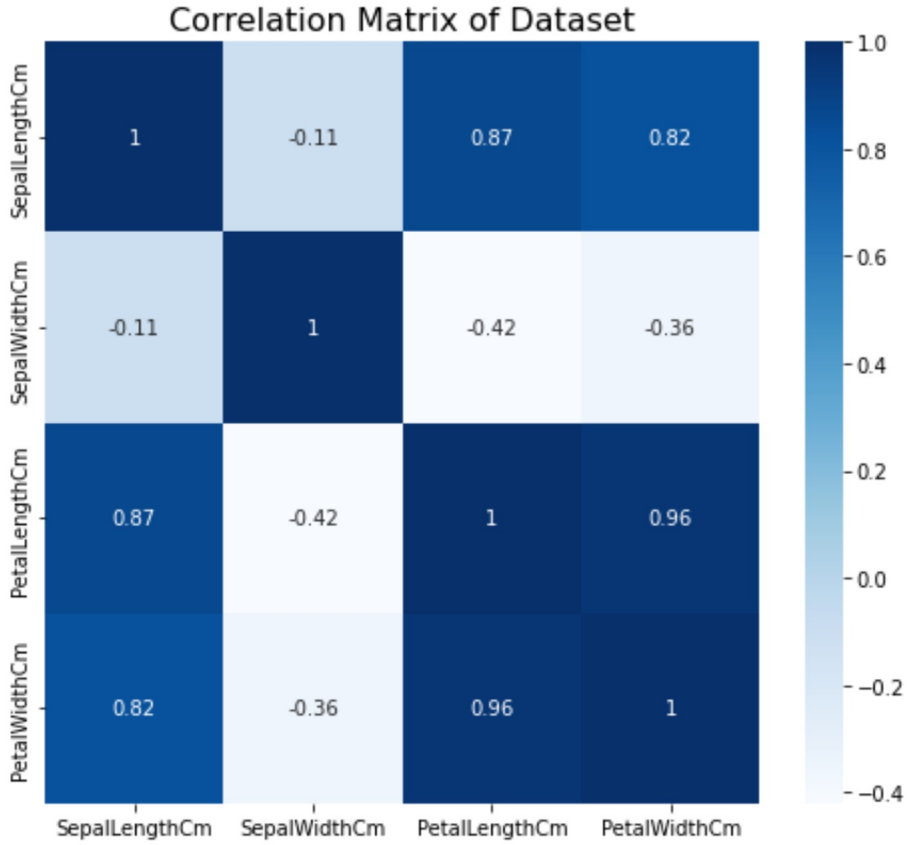
2. Given two arrays of numeric values, identify the suitable visualization mechanisms like Heat Maps to draw relationship between the given two sets. Also extend the same for the IRIS dataset for possible attribute subset of your choice.

In [46]:
```
# random Heapmap example...

a=np.random.randint(1, 101, 4)
b=np.random.randint(1, 101, 4)

plt.figure(figsize=(8,7))
distances = pairwise_distances(X=a.reshape(-1, 1),Y=b.reshape(-1, 1))
sns.heatmap(distances, square=True, xticklabels = a, yticklabels=b, annot=True, cbar=True, cmap="Blues");
```



In [ ]:
```

```

In [47]:
```
# heatap of Iris Dataset..
plt.figure(figsize=(8,7))
plt.title("Correlation Matrix of Dataset", size=16)
sns.heatmap(df.drop('Id',axis=1).corr(),cmap="Blues",annot=True)
plt.show()
```

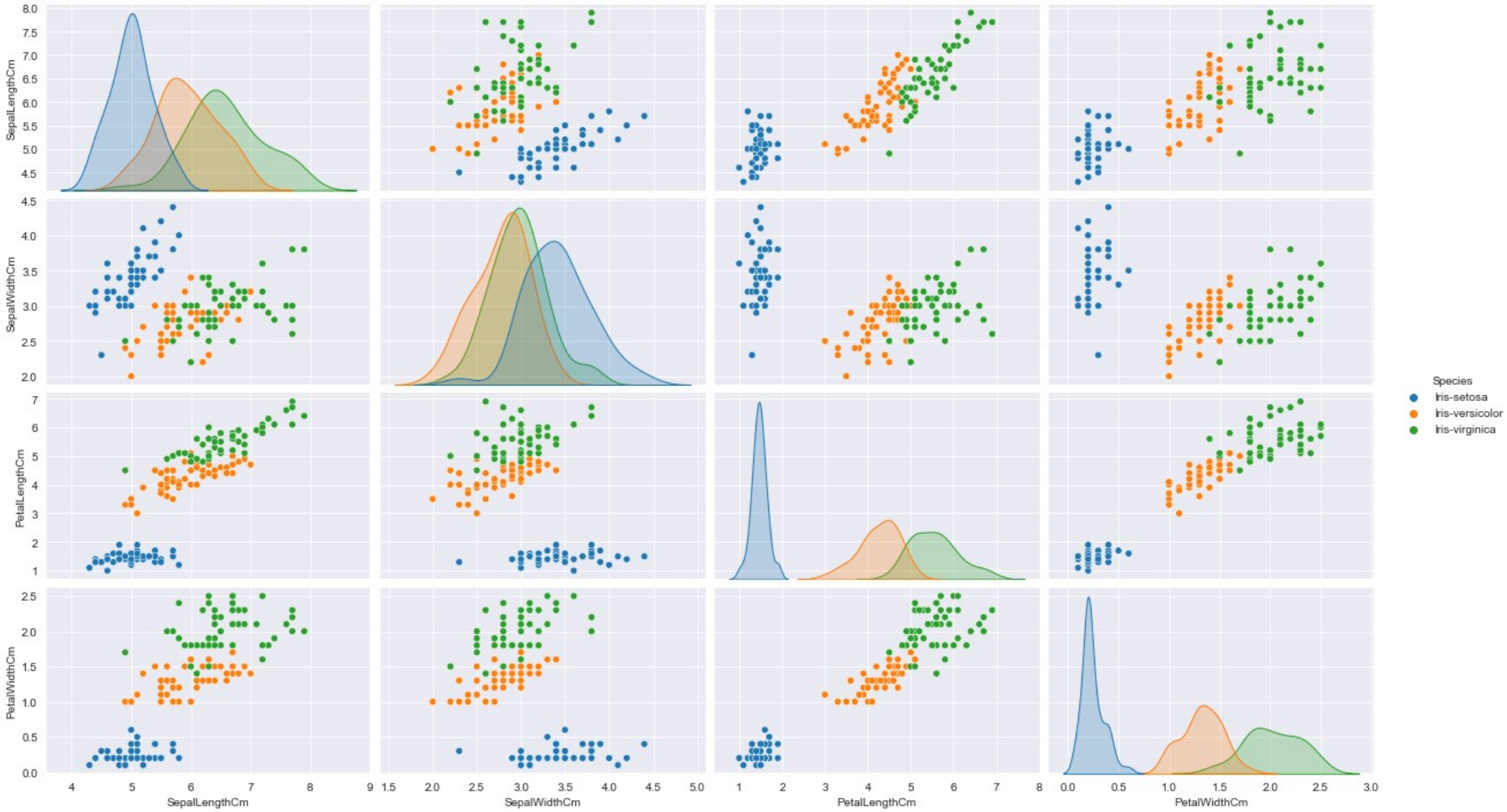


In [ ]:

In [ ]:

3. Explore the usage of correlogram using the in-built packages for IRIS dataset of your choice and list the inferences drawn from the plots.

In [76]:
```
# plt.figure(figsize=(10,7))
sns.set_style('darkgrid')
sns.pairplot(df.iloc[:,1:], diag_kind='kde', height=2.5, aspect=1.7, hue='Species')
plt.show()
```



Inferences

-> From the above Graph we can see if we take petalLength and petalWidth then all the species are lineary separable.

-> We can also see petalLength and petalWidth are strongly positve correlated.

-> SepalWidth is poorly coorelated with any of the columns.

-> PetalLength and PetalWidth is also lineary separable.

In [ ]:

In [ ]:

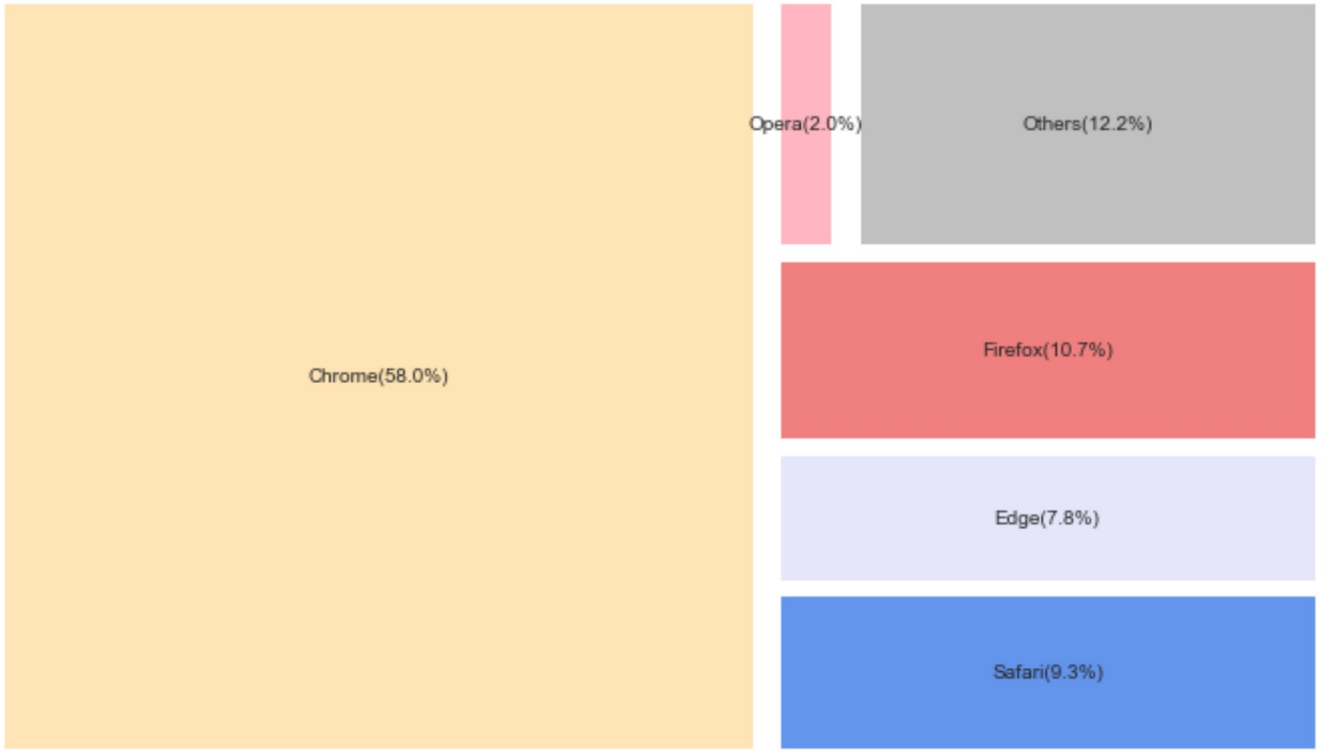
4. Test Drive the hierarchical data visualization techniques like TreeMap for the below given data.

In [78]:
```
values=[58,9.3,7.8,10.7,2,12.2]
labels=["Chrome","Safari","Edge","Firefox","Opera","Others"]

df=pd.DataFrame({"Browsers":labels,"Market Share(%)":values})
df
```

Out[78]:

	Browsers	Market Share(%)
0	Chrome	58.0
1	Safari	9.3
2	Edge	7.8
3	Firefox	10.7
4	Opera	2.0
5	Others	12.2

In [117…
```
plt.figure(figsize=(12,7))

colors = ('moccasin', 'cornflowerblue', 'lavender', 'lightcoral', 'lightpink', 'silver')
graphlabels=df["Browsers"].to_list()
for i in range(len(graphlabels)):
    graphlabels[i]=f"{graphlabels[i]}({str(df['Market Share(%)'].to_list()[i])}%)"

norms = squarify.normalize_sizes(df['Market Share(%)'].to_list(),  dx=6, dy=6)
squarify.plot(sizes=norms, label=graphlabels, color=colors, alpha=1, pad=True)
plt.axis('off')
plt.show()
```



In [ ]:

In [ ]:

In [ ]: