In [1]:

**import** numpy **as** np *# linear algebra*

**import** pandas **as** pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

In [3]:

iris**=** pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\6\Iris.csv")

In [4]:

iris.head()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Out[4]: |  | | | | | | |
|  |  | **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 [5]:

iris['Species'].unique()

Out[5]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

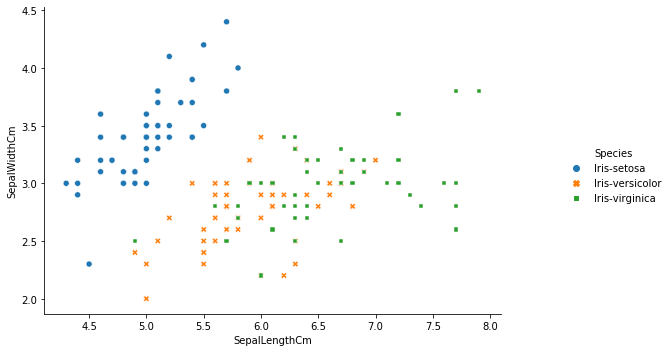
In [6]:

iris.drop(columns**=**"Id",inplace**=True**)

In [7]:

g**=**sns.relplot(x**=**'SepalLengthCm',y**=**'SepalWidthCm',data**=**iris,hue**=**'Species',style**=**'Species g.fig.set\_size\_inches(10,5)

plt.show()



In [8]:

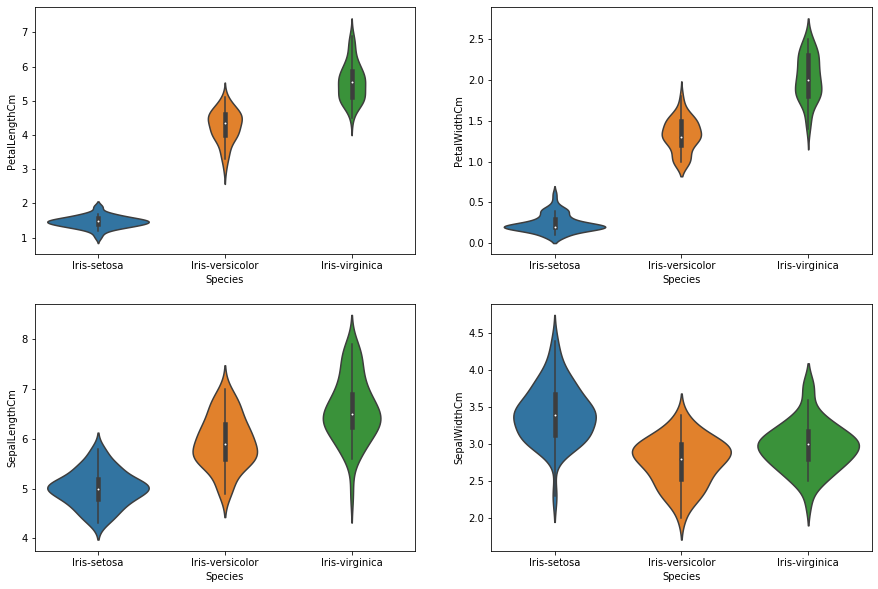
plt.figure(figsize**=**(15,10)) plt.subplot(2,2,1)

sns.violinplot(x**=**'Species',y**=**'PetalLengthCm',data**=**iris) plt.subplot(2,2,2)

sns.violinplot(x**=**'Species',y**=**'PetalWidthCm',data**=**iris) plt.subplot(2,2,3)

sns.violinplot(x**=**'Species',y**=**'SepalLengthCm',data**=**iris) plt.subplot(2,2,4)

sns.violinplot(x**=**'Species',y**=**'SepalWidthCm',data**=**iris) plt.show()



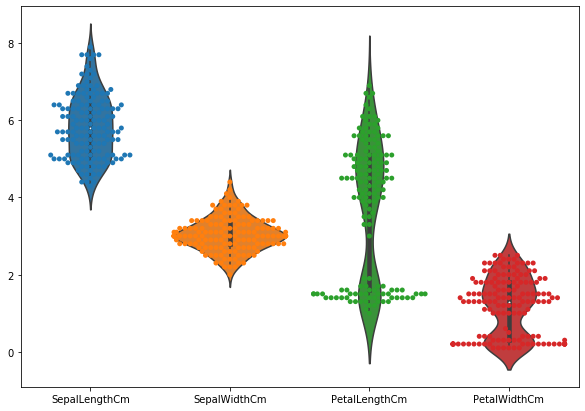
In [9]:

plt.subplots(figsize**=**(10,7)) sns.violinplot(data**=**iris)

sns.swarmplot( data**=**iris) plt.show()

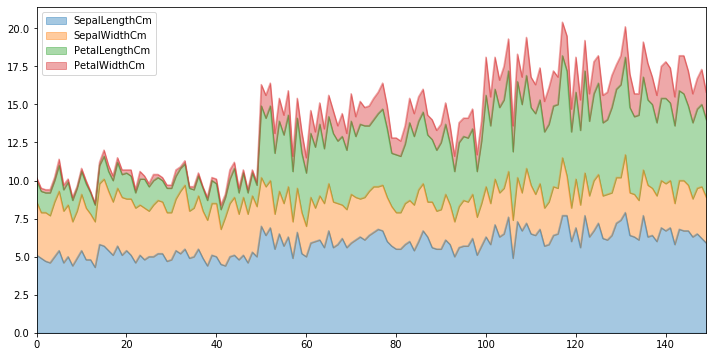
C:\Users\HP\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 9.3% of the points cannot be placed; you may want to decrease the size of the markers or us e stripplot.

warnings.warn(msg, UserWarning)



In [10]:

iris.plot.area(y**=**['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'],alpha**=**



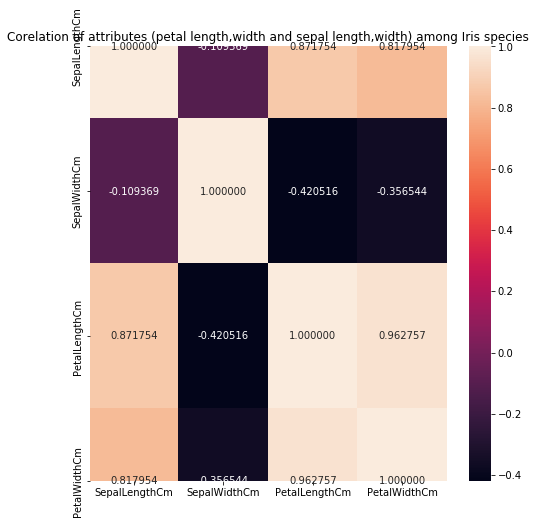
In [11]:

iris.corr()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[11]: |  | | | | |
|  |  | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** |
|  | SepalLengthCm | 1.000000 | -0.109369 | 0.871754 | 0.817954 |
|  | SepalWidthCm | -0.109369 | 1.000000 | -0.420516 | -0.356544 |
|  | PetalLengthCm | 0.871754 | -0.420516 | 1.000000 | 0.962757 |
|  | PetalWidthCm | 0.817954 | -0.356544 | 0.962757 | 1.000000 |

In [12]:

|  |  |  |
| --- | --- | --- |
| plt.subplots(figsize **=** (8,8))  sns.heatmap(iris.corr(),annot**=True**,fmt**=**"f").set\_title("Corelation of attributes (petal plt.show() | | |
|  |  |  |



In [13]:

X**=**iris.iloc[:,0:4].values y**=**iris.iloc[:,4].values

In [14]:

**from** sklearn.preprocessing **import** LabelEncoder le **=** LabelEncoder()

y **=** le.fit\_transform(y)

In [18]:

*#Metrics*

**from** sklearn.metrics **import** make\_scorer, accuracy\_score,precision\_score

**from** sklearn.metrics **import** classification\_report

**from** sklearn.metrics **import** confusion\_matrix

**from** sklearn.metrics **import** accuracy\_score ,precision\_score,recall\_score,f1\_score

*#Model Select*

**from** sklearn.naive\_bayes **import** GaussianNB

In [21]:

**from** sklearn.model\_selection **import** train\_test\_split

In [22]:

*#Train and Test split*

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.3,random\_state**=**0)

In [23]:

gaussian **=** GaussianNB()

gaussian.fit(X\_train, y\_train)

Y\_pred **=** gaussian.predict(X\_test)

accuracy\_nb**=**round(accuracy\_score(y\_test,Y\_pred)**\*** 100, 2)

acc\_gaussian **=** round(gaussian.score(X\_train, y\_train) **\*** 100, 2)

cm **=** confusion\_matrix(y\_test, Y\_pred)

accuracy **=** accuracy\_score(y\_test,Y\_pred)

precision **=**precision\_score(y\_test, Y\_pred,average**=**'micro') recall **=** recall\_score(y\_test, Y\_pred,average**=**'micro')

f1 **=** f1\_score(y\_test,Y\_pred,average**=**'micro')

print('Confusion matrix for Naive Bayes\n',cm) print('accuracy\_Naive Bayes: %.3f' **%**accuracy) print('precision\_Naive Bayes: %.3f' **%**precision) print('recall\_Naive Bayes: %.3f' **%**recall)

print('f1-score\_Naive Bayes : %.3f' **%**f1)

Confusion matrix for Naive Bayes [[16 0 0]

[ 0 18 0]

[ 0 0 11]]

accuracy\_Naive Bayes: 1.000

precision\_Naive Bayes: 1.000

recall\_Naive Bayes: 1.000

f1-score\_Naive Bayes : 1.000

In [ ]: