In [1]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


import warnings
warnings.filterwarnings('ignore')


plt.style.use("fivethirtyeight")
%matplotlib inline
```

In [2]:
```
df=pd.read_csv('Iris.csv')
df.head()
```

Out[2]:

	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]:
```
#information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [4]:
```
#describing about the dataset
df.describe()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [5]:
```
df.shape
```

Out[5]:
```
(150, 6)
```

In [6]:
```
df.drop('Id',axis=1,inplace=True)
```

In [7]:
```
df.head()
```

Out[7]:

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

In [8]:
```
#count the value
df['Species'].value_counts()
```

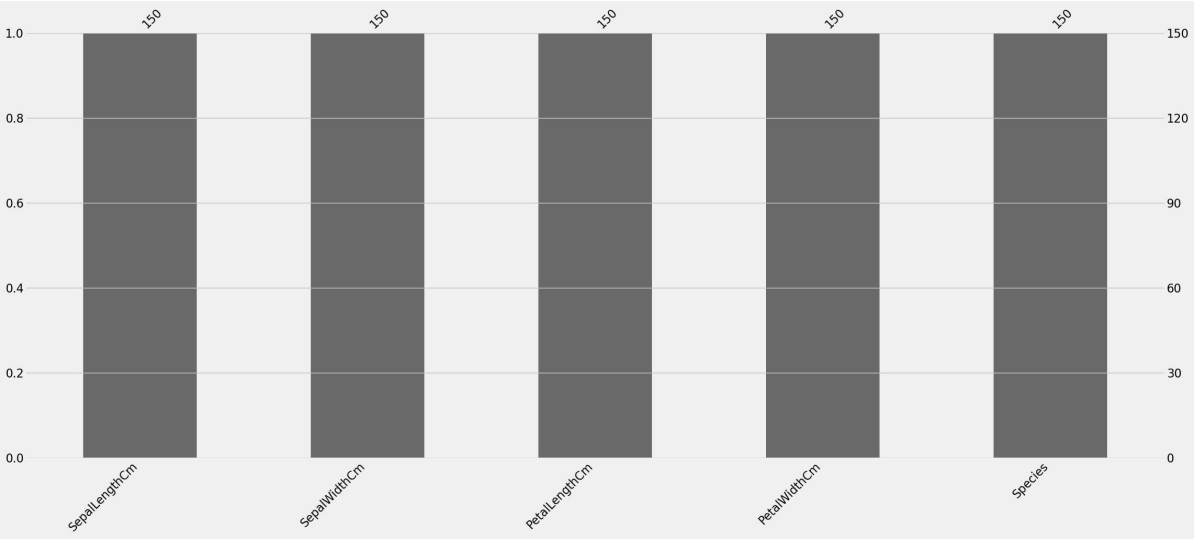
Out[8]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [9]:
```
#finding the null value
df.isnull().sum()
```

Out[9]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [12]:
```
import missingno as msno
msno.bar(df)
```

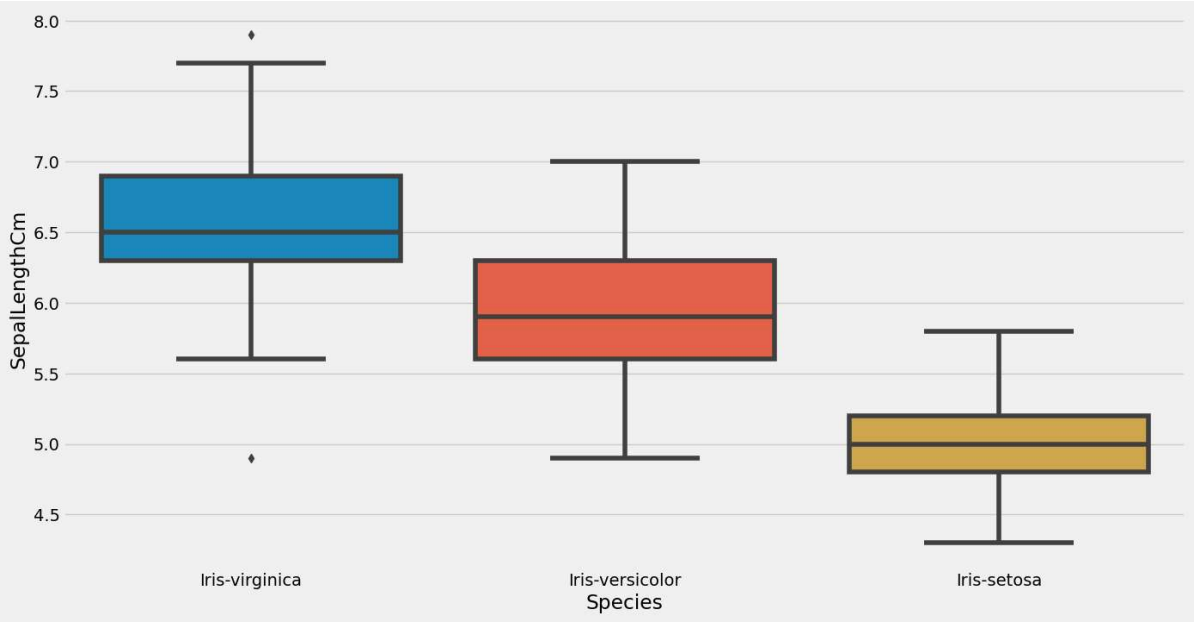
Out[12]:
```
<Axes: >
```



In [13]:
```
df.drop_duplicates(inplace=True)
```

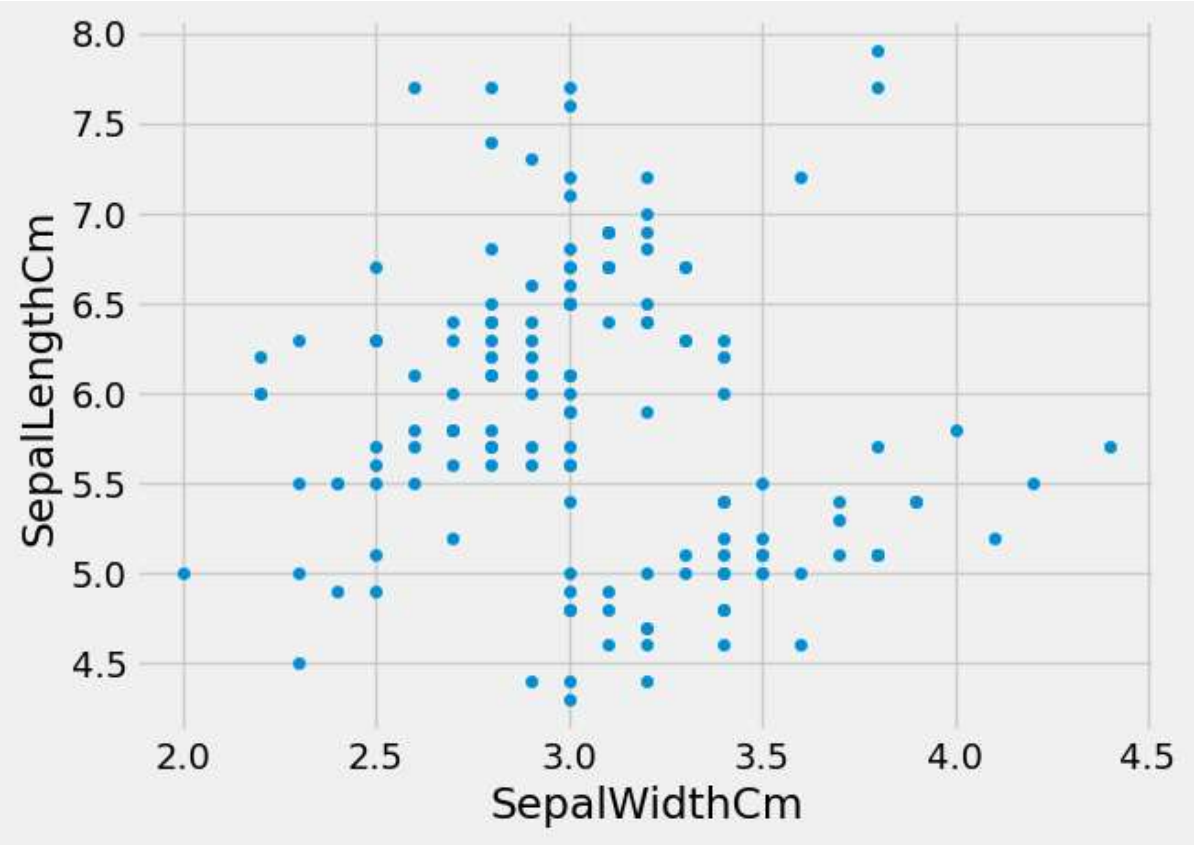
In [14]:
```
plt.figure(figsize=(15,8))
sns.boxplot(x='Species',y='SepalLengthCm',data=df.sort_values('SepalLengthCm',asce
```

Out[14]:
```
<Axes: xlabel='Species', ylabel='SepalLengthCm'>
```



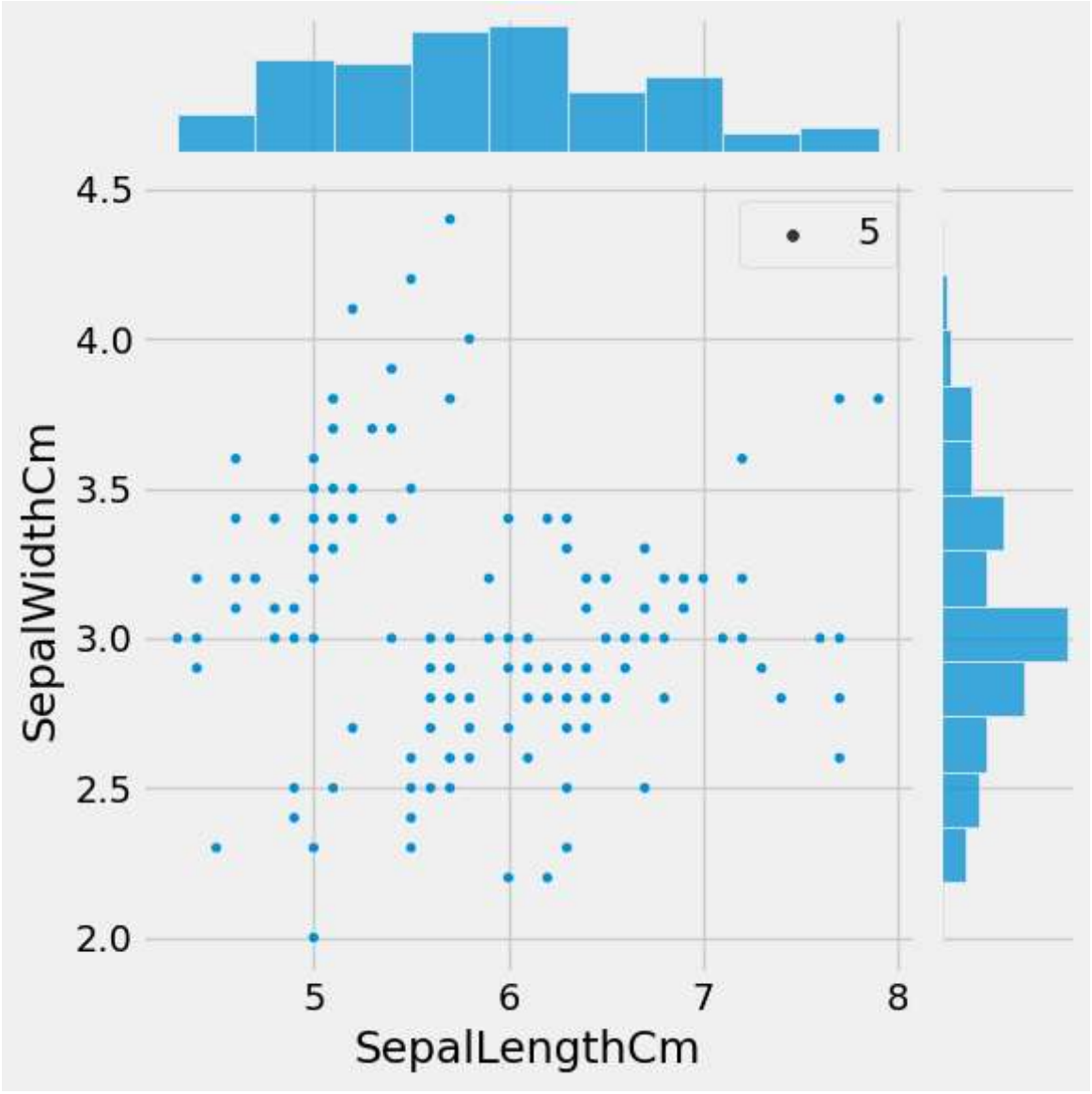
In [15]:
```
df.plot(kind='scatter',x='SepalWidthCm',y='SepalLengthCm')
```

Out[15]:
```
<Axes: xlabel='SepalWidthCm', ylabel='SepalLengthCm'>
```



In [16]:
```
sns.jointplot(x="SepalLengthCm", y="SepalWidthCm", data=df, size=5)
```

Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x2918f0b6390>
```

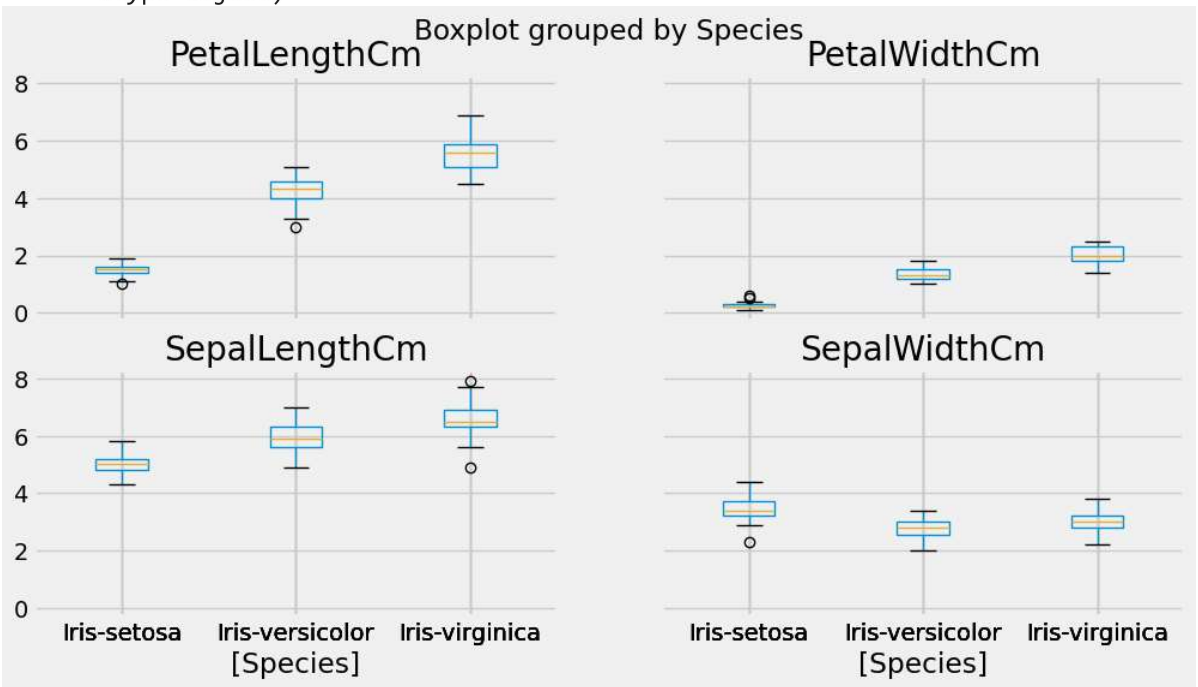


In [17]:
```
sns.pairplot(df, hue="Species", size=3)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x2918f1f4ad0>
```

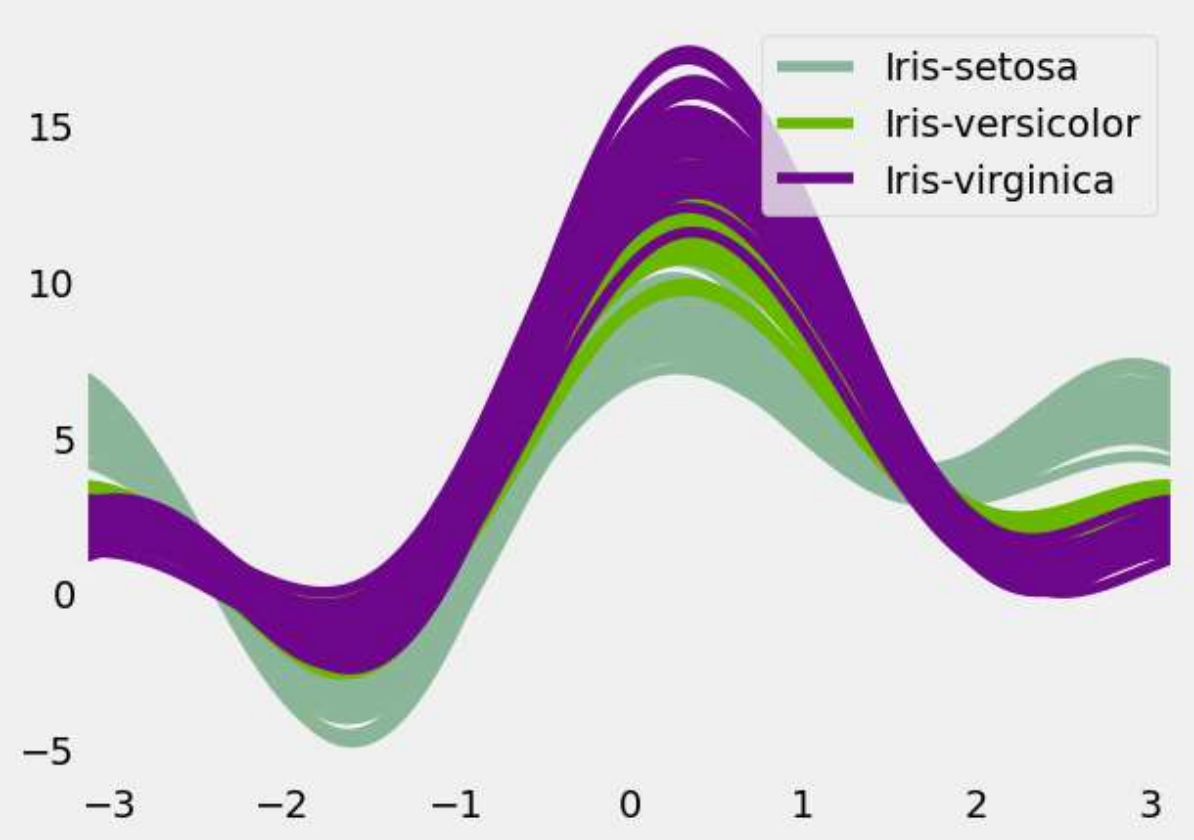
In [18]:
```
df.boxplot(by="Species", figsize=(12, 6))
```

Out[18]:
```
array([[<Axes: title={'center': 'PetalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'PetalWidthCm'}, xlabel='[Species]'>],
       [<Axes: title={'center': 'SepalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



In [19]:
```
import pandas.plotting
from pandas.plotting import andrews_curves
andrews_curves(df, "Species")
```

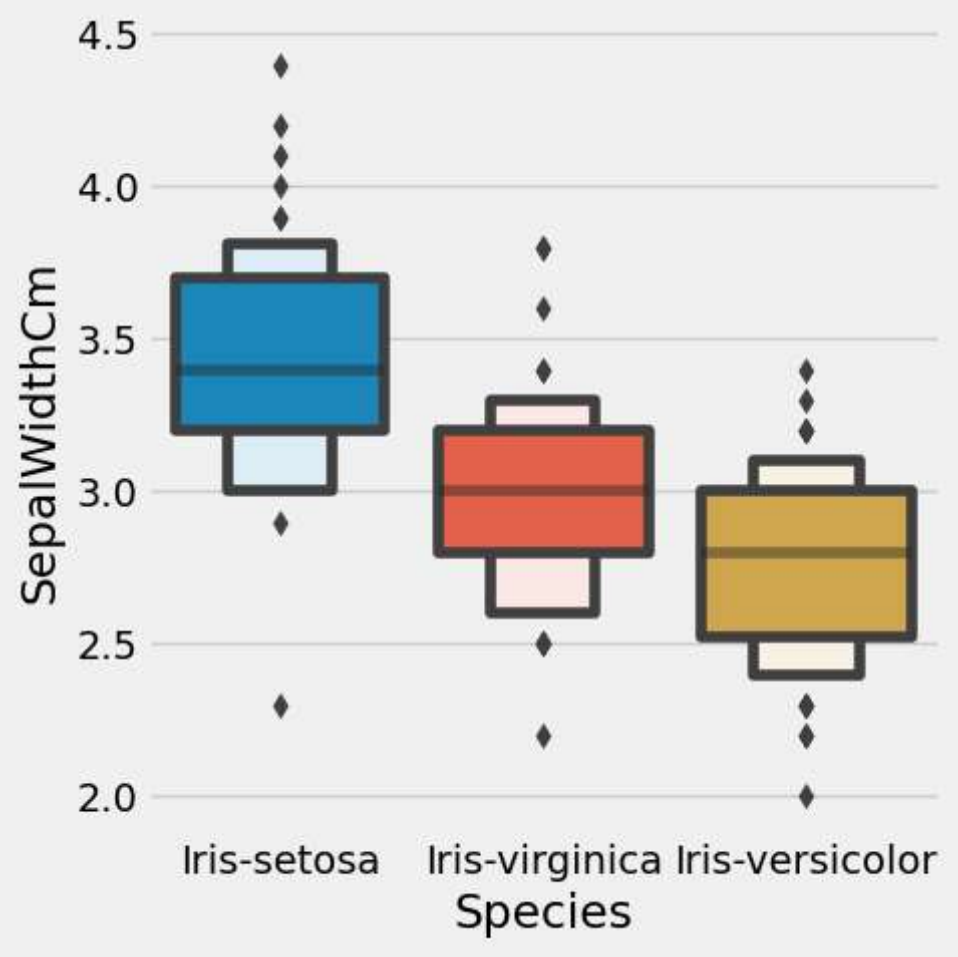
Out[19]:
```
<Axes: >
```



In [20]:
```
plt.figure(figsize=(15,15))
sns.catplot(x='Species',y='SepalWidthCm',data=df.sort_values('SepalWidthCm',ascend
```

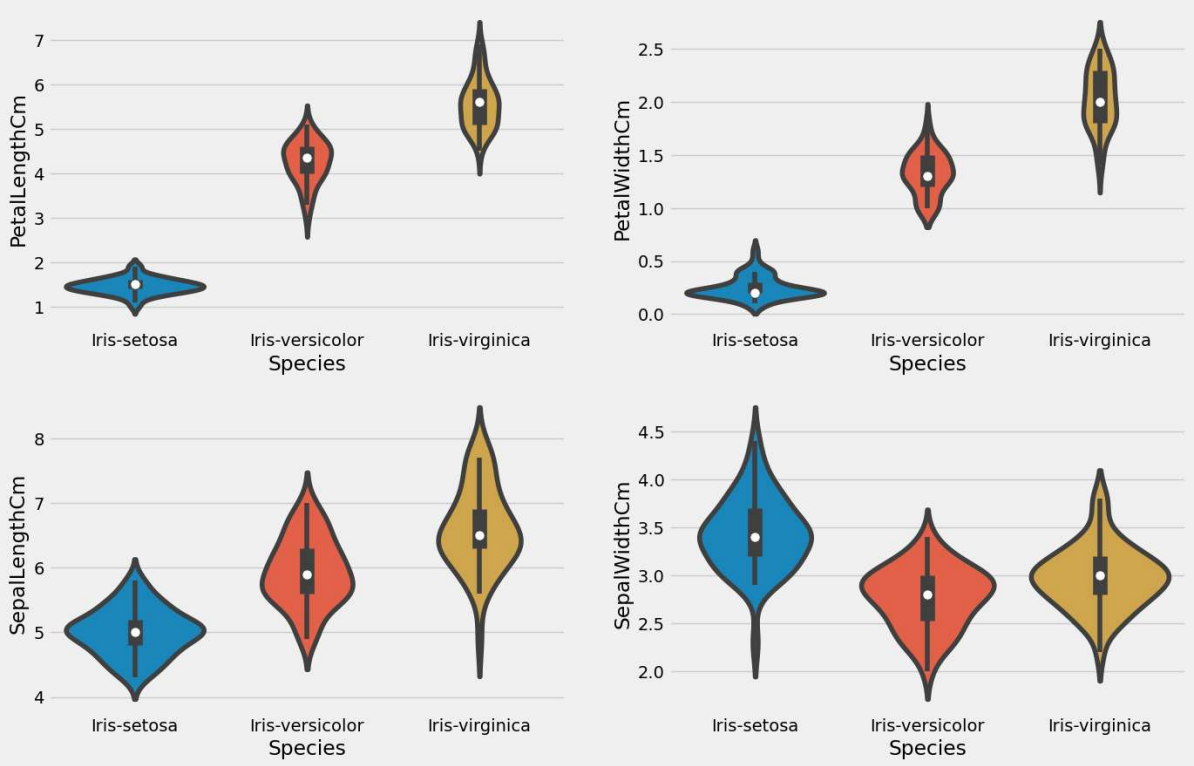
Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x29191dfda50>

<Figure size 1500x1500 with 0 Axes>
```



In [21]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=df)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=df)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=df)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=df)
```

Out[21]:
```
<Axes: xlabel='Species', ylabel='SepalWidthCm'>
```



In [22]:
```
X=df.drop('Species',axis=1)
y=df['Species']
```

In [27]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
```

In [28]:
```
df['Species'] = pd.Categorical(df.Species)
df['Species'] = df.Species.cat.codes
# Turn response variable into one-hot response vectory = to_categorical(df.response
y = to_categorical(df.Species)
```

In [29]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.30,stratify=y,ran
```

In [30]:
```
model=Sequential()
model.add(Dense(100,activation='relu',input_shape=(4,)))

model.add(Dense(3,activation='softmax'))
```

In [31]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'
```

In [39]:
```
history=model.fit(X_train,y_train,epochs=45,validation_data=(X_test, y_test))
```

```
Epoch 1/45
4/4 [==============================] - 0s 51ms/step - loss: 0.0649 - accuracy: 0.9
804 - val_loss: 0.0900 - val_accuracy: 0.9556
Epoch 2/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0648 - accuracy: 0.9
804 - val_loss: 0.0892 - val_accuracy: 0.9556
Epoch 3/45
4/4 [==============================] - 0s 21ms/step - loss: 0.0645 - accuracy: 0.9
804 - val_loss: 0.0884 - val_accuracy: 0.9556
Epoch 4/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0643 - accuracy: 0.9
804 - val_loss: 0.0881 - val_accuracy: 0.9556
Epoch 5/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0651 - accuracy: 0.9
804 - val_loss: 0.0885 - val_accuracy: 0.9778
Epoch 6/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0650 - accuracy: 0.9
804 - val_loss: 0.0887 - val_accuracy: 0.9778
Epoch 7/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0657 - accuracy: 0.9
902 - val_loss: 0.0894 - val_accuracy: 0.9778
Epoch 8/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0646 - accuracy: 0.9
902 - val_loss: 0.0878 - val_accuracy: 0.9556
Epoch 9/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0647 - accuracy: 0.9
804 - val_loss: 0.0929 - val_accuracy: 0.9556
Epoch 10/45
4/4 [==============================] - 0s 21ms/step - loss: 0.0658 - accuracy: 0.9
804 - val_loss: 0.0937 - val_accuracy: 0.9556
Epoch 11/45
4/4 [==============================] - 0s 25ms/step - loss: 0.0655 - accuracy: 0.9
804 - val_loss: 0.0906 - val_accuracy: 0.9556
Epoch 12/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0643 - accuracy: 0.9
804 - val_loss: 0.0879 - val_accuracy: 0.9556
Epoch 13/45
4/4 [==============================] - 0s 27ms/step - loss: 0.0635 - accuracy: 0.9
804 - val_loss: 0.0874 - val_accuracy: 0.9778
Epoch 14/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0630 - accuracy: 0.9
902 - val_loss: 0.0910 - val_accuracy: 0.9778
Epoch 15/45
4/4 [==============================] - 0s 27ms/step - loss: 0.0699 - accuracy: 0.9
804 - val_loss: 0.0957 - val_accuracy: 0.9778
Epoch 16/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0696 - accuracy: 0.9
804 - val_loss: 0.0880 - val_accuracy: 0.9778
Epoch 17/45
4/4 [==============================] - 0s 27ms/step - loss: 0.0637 - accuracy: 0.9
804 - val_loss: 0.0872 - val_accuracy: 0.9556
Epoch 18/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0636 - accuracy: 0.9
804 - val_loss: 0.0881 - val_accuracy: 0.9778
Epoch 19/45
4/4 [==============================] - 0s 25ms/step - loss: 0.0640 - accuracy: 0.9
902 - val_loss: 0.0879 - val_accuracy: 0.9778
Epoch 20/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0635 - accuracy: 0.9
902 - val_loss: 0.0870 - val_accuracy: 0.9778
Epoch 21/45
4/4 [==============================] - 0s 26ms/step - loss: 0.0643 - accuracy: 0.9
804 - val_loss: 0.0879 - val_accuracy: 0.9556
Epoch 22/45
```

```
4/4 [==============================] - 0s 22ms/step - loss: 0.0622 - accuracy: 0.9
804 - val_loss: 0.0875 - val_accuracy: 0.9778
Epoch 23/45
4/4 [==============================] - 0s 19ms/step - loss: 0.0645 - accuracy: 0.9
804 - val_loss: 0.0875 - val_accuracy: 0.9778
Epoch 24/45
4/4 [==============================] - 0s 23ms/step - loss: 0.0644 - accuracy: 0.9
902 - val_loss: 0.0918 - val_accuracy: 0.9778
Epoch 25/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0669 - accuracy: 0.9
804 - val_loss: 0.0888 - val_accuracy: 0.9778
Epoch 26/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0620 - accuracy: 0.9
902 - val_loss: 0.0879 - val_accuracy: 0.9556
Epoch 27/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0616 - accuracy: 0.9
804 - val_loss: 0.0905 - val_accuracy: 0.9556
Epoch 28/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0633 - accuracy: 0.9
804 - val_loss: 0.0914 - val_accuracy: 0.9556
Epoch 29/45
4/4 [==============================] - 0s 28ms/step - loss: 0.0641 - accuracy: 0.9
804 - val_loss: 0.0896 - val_accuracy: 0.9556
Epoch 30/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0628 - accuracy: 0.9
804 - val_loss: 0.0896 - val_accuracy: 0.9556
Epoch 31/45
4/4 [==============================] - 0s 21ms/step - loss: 0.0626 - accuracy: 0.9
804 - val_loss: 0.0889 - val_accuracy: 0.9556
Epoch 32/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0644 - accuracy: 0.9
804 - val_loss: 0.0917 - val_accuracy: 0.9556
Epoch 33/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0637 - accuracy: 0.9
804 - val_loss: 0.0889 - val_accuracy: 0.9556
Epoch 34/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0619 - accuracy: 0.9
804 - val_loss: 0.0883 - val_accuracy: 0.9333
Epoch 35/45
4/4 [==============================] - 0s 30ms/step - loss: 0.0615 - accuracy: 0.9
804 - val_loss: 0.0874 - val_accuracy: 0.9556
Epoch 36/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0614 - accuracy: 0.9
804 - val_loss: 0.0865 - val_accuracy: 0.9556
Epoch 37/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0602 - accuracy: 0.9
804 - val_loss: 0.0865 - val_accuracy: 0.9778
Epoch 38/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0614 - accuracy: 0.9
804 - val_loss: 0.0861 - val_accuracy: 0.9778
Epoch 39/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0631 - accuracy: 0.9
804 - val_loss: 0.0872 - val_accuracy: 0.9556
Epoch 40/45
4/4 [==============================] - 0s 21ms/step - loss: 0.0609 - accuracy: 0.9
804 - val_loss: 0.0879 - val_accuracy: 0.9333
Epoch 41/45
4/4 [==============================] - 0s 22ms/step - loss: 0.0616 - accuracy: 0.9
804 - val_loss: 0.0887 - val_accuracy: 0.9556
Epoch 42/45
4/4 [==============================] - 0s 21ms/step - loss: 0.0620 - accuracy: 0.9
804 - val_loss: 0.0888 - val_accuracy: 0.9556
Epoch 43/45
4/4 [==============================] - 0s 20ms/step - loss: 0.0635 - accuracy: 0.9
```

```
804 - val_loss: 0.0983 - val_accuracy: 0.9556
Epoch 44/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0677 - accuracy: 0.9
804 - val_loss: 0.1007 - val_accuracy: 0.9556
Epoch 45/45
4/4 [==============================] - 0s 24ms/step - loss: 0.0669 - accuracy: 0.9
804 - val_loss: 0.0927 - val_accuracy: 0.9556
```

In [40]:
```
model.evaluate(X_test,y_test)
```

```
2/2 [==============================] - 0s 8ms/step - loss: 0.0927 - accuracy: 0.95
56
```

Out[40]:
```
[0.09265806525945663, 0.9555555582046509]
```

In [41]:
```
pred = model.predict(X_test[:10])
print(pred)
```

```
1/1 [==============================] - 0s 168ms/step
[[9.7774034e-08 4.4938782e-03 9.9550605e-01]
 [6.2056365e-08 7.4188537e-03 9.9258101e-01]
 [1.6938194e-03 9.8962337e-01 8.6827455e-03]
 [4.4397949e-04 9.4897318e-01 5.0582740e-02]
 [9.9925882e-01 7.4121240e-04 4.4876602e-12]
 [3.6019078e-04 9.7398305e-01 2.5656831e-02]
 [2.9136688e-08 9.0456055e-04 9.9909544e-01]
 [4.8440320e-08 1.4234002e-03 9.9857652e-01]
 [9.9925512e-01 7.4486999e-04 2.6795003e-12]
 [5.6502458e-06 5.0783023e-02 9.4921136e-01]]
```

In [42]:
```
p=np.argmax(pred,axis=1)
print(p)
print(y_test[:10])
```

```
[2 2 1 1 0 1 2 2 0 2]
[[0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]]
```

In [43]:
```
history.history['accuracy']
```

Out[43]:
```
[0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9901960492134094,
 0.9901960492134094,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9901960492134094,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9901960492134094,
 0.9901960492134094,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9901960492134094,
 0.9803921580314636,
 0.9901960492134094,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636,
 0.9803921580314636]
```

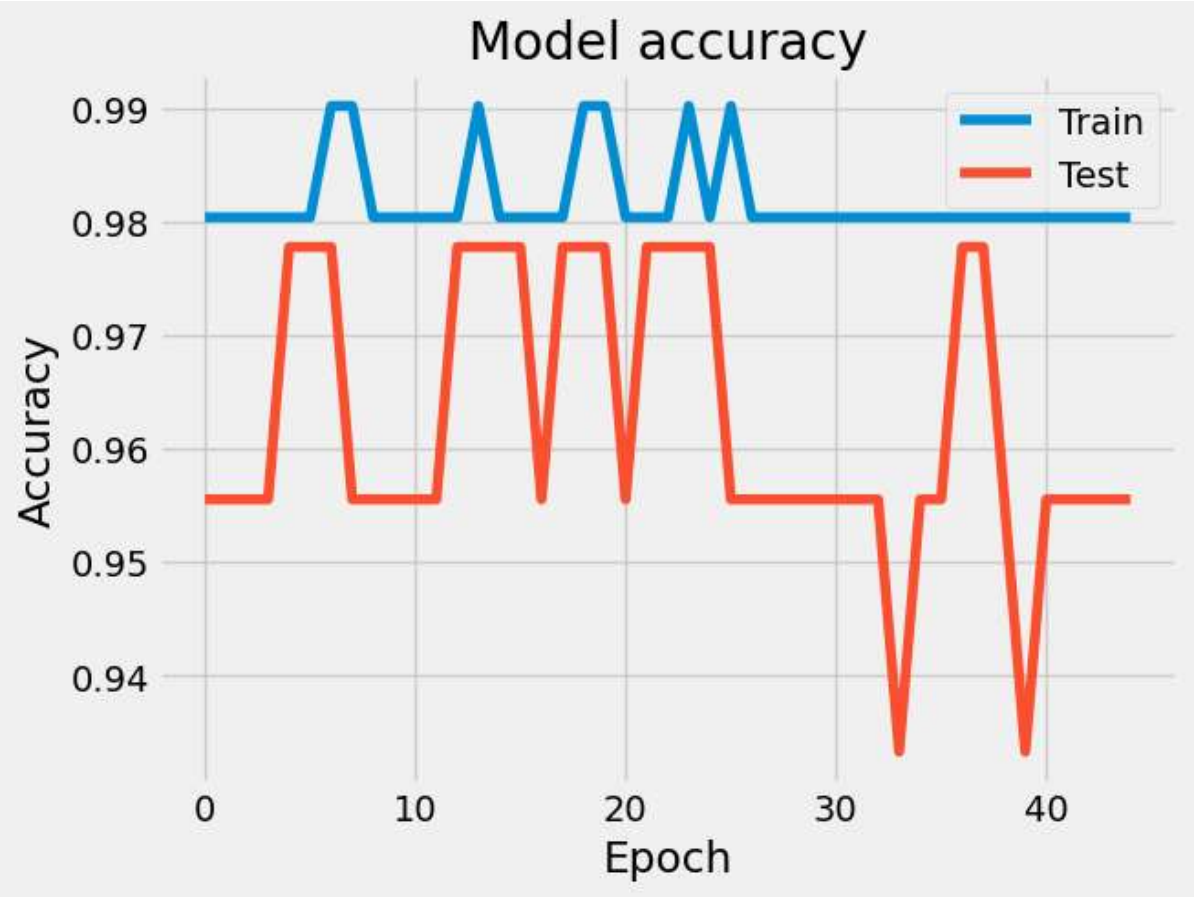
In [44]:
```
history.history['val_accuracy']
```

Out[44]:
```
[0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9555555582046509,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9555555582046509,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9777777791023254,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9333333373069763,
 0.9555555582046509,
 0.9555555582046509,
 0.9777777791023254,
 0.9777777791023254,
 0.9555555582046509,
 0.9333333373069763,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509,
 0.9555555582046509]
```

In [45]:
```
plt.figure()


plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'])
plt.show()
```


Model accuracy
Train
Test
Accuracy
0.99
0.98
0.97
0.96
0.95
0.94
0
10
20
30
40
Epoch

In [ ]: