Task-3 Iris Flower Classifcation

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


from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [2]:
```
iris=pd.read_csv("iris (1).csv")
print(iris)
```

```
     Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
\
0             1           5.1          3.5           1.4          0.2
1             2           4.9          3.0           1.4          0.2
2             3           4.7          3.2           1.3          0.2
3             4           4.6          3.1           1.5          0.2
4             5           5.0          3.6           1.4          0.2
..          ...           ...          ...           ...          ...
145         146           6.7          3.0           5.2          2.3
146         147           6.3          2.5           5.0          1.9
147         148           6.5          3.0           5.2          2.0
148         149           6.2          3.4           5.4          2.3
149         150           5.9          3.0           5.1          1.8

       Species
0       setosa
1       setosa
2       setosa
3       setosa
```

In [3]:
```
print(iris.shape)
```

```
(150, 6)
```

In [4]:
```
print(iris.describe())
```

```
       Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count  150.000000    150.000000   150.000000    150.000000   150.000000
mean    75.500000      5.843333     3.057333      3.758000     1.199333
std     43.445368      0.828066     0.435866      1.765298     0.762238
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]:
```
#Checking for null values
print(iris.isna().sum())
print(iris.describe())
```

```
Unnamed: 0      0
Sepal.Length    0
Sepal.Width     0
Petal.Length    0
Petal.Width     0
Species         0
dtype: int64
       Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count  150.000000    150.000000   150.000000    150.000000   150.000000
mean    75.500000      5.843333     3.057333      3.758000     1.199333
std     43.445368      0.828066     0.435866      1.765298     0.762238
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 [6]:
```
iris.head()
```

Out[6]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa

In [7]:
```
iris.head(150)
```

Out[7]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	5.9	3.0	5.1	1.8	virginica

150 rows × 6 columns

In [8]:
```
iris.tail(100)
```

Out[8]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
50	51	7.0	3.2	4.7	1.4	versicolor
51	52	6.4	3.2	4.5	1.5	versicolor
52	53	6.9	3.1	4.9	1.5	versicolor
53	54	5.5	2.3	4.0	1.3	versicolor
54	55	6.5	2.8	4.6	1.5	versicolor
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	5.9	3.0	5.1	1.8	virginica

100 rows × 6 columns

In [9]:
```
n = len(iris[iris['Species'] == 'versicolor'])
print("No of Versicolor in Dataset:",n)
```

```
No of Versicolor in Dataset: 50
```

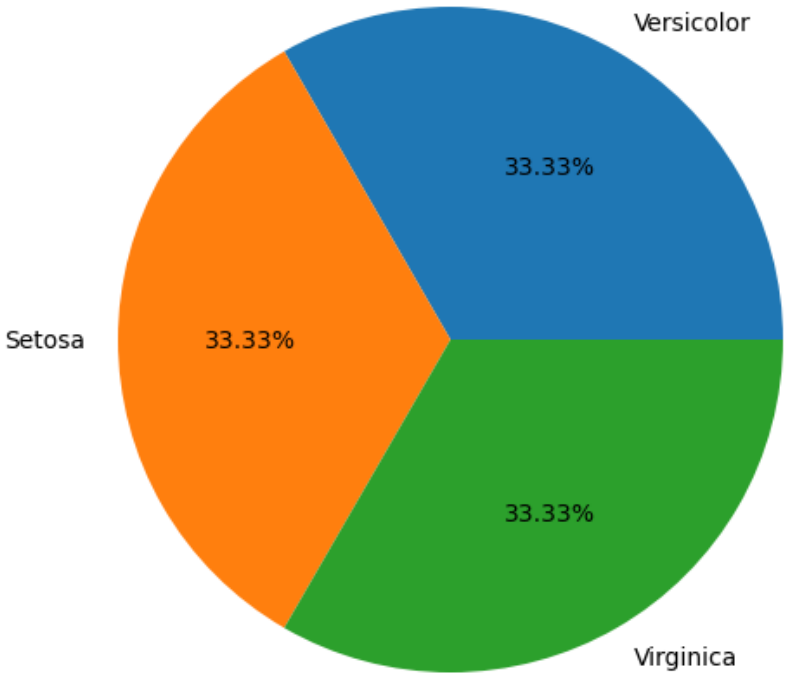
In [10]:
```
n1 = len(iris[iris['Species'] == 'virginica'])
print("No of Virginica in Dataset:",n1)
```

```
No of Virginica in Dataset: 50
```

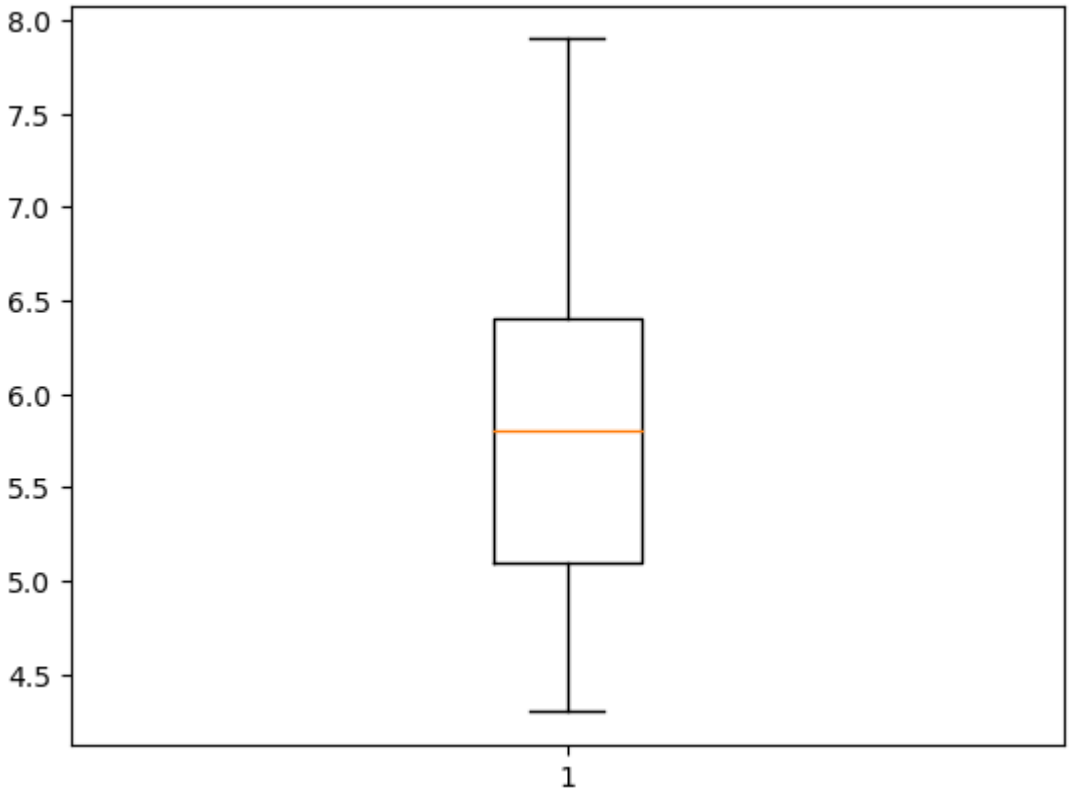
In [11]:
```
n2 = len(iris[iris['Species'] == 'setosa'])
print("No of Setosa in Dataset:",n2)
```

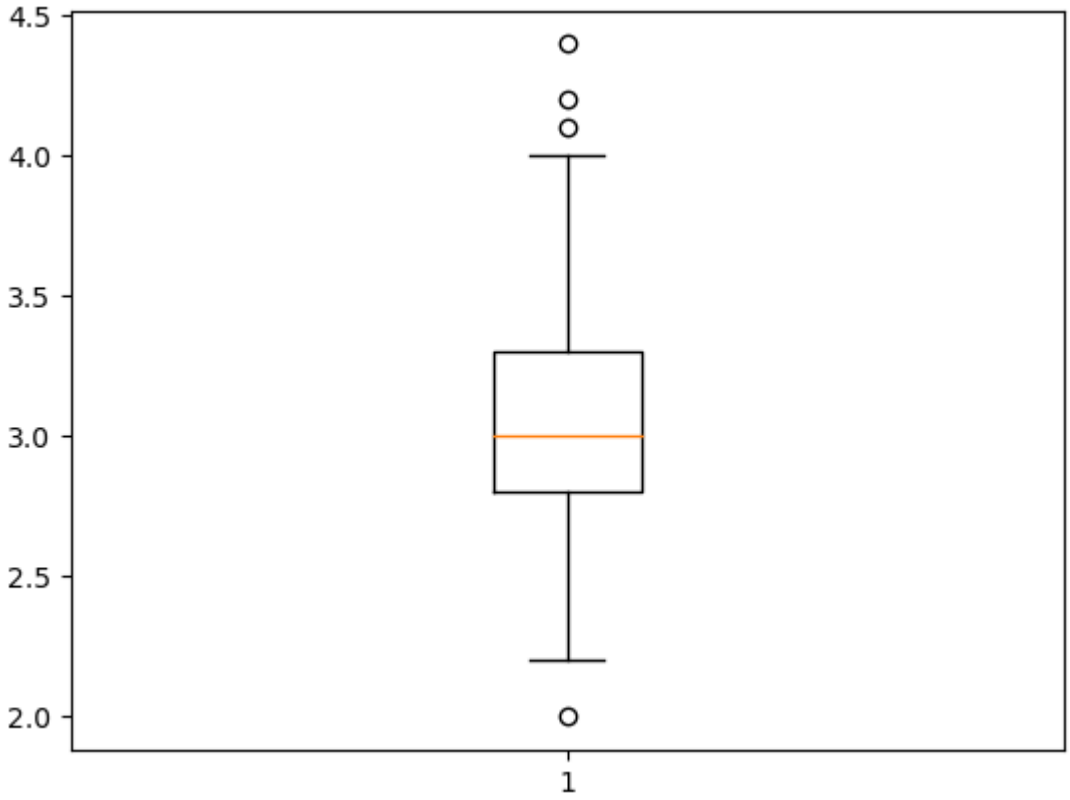
```
No of Setosa in Dataset: 50
```

In [12]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [50,50,50]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```

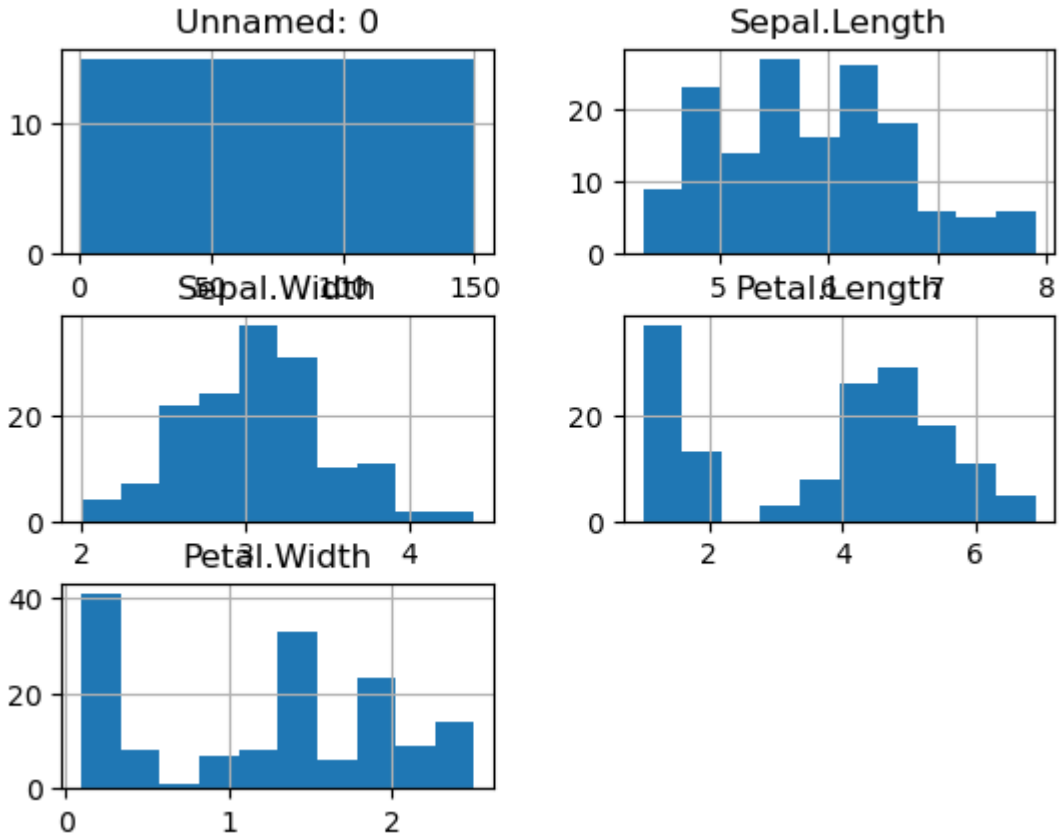


In [13]:
```
#Checking for outliars
import matplotlib.pyplot as plt
plt.figure(1)
plt.boxplot([iris['Sepal.Length']])
plt.figure(2)
plt.boxplot([iris['Sepal.Width']])
plt.show()
```



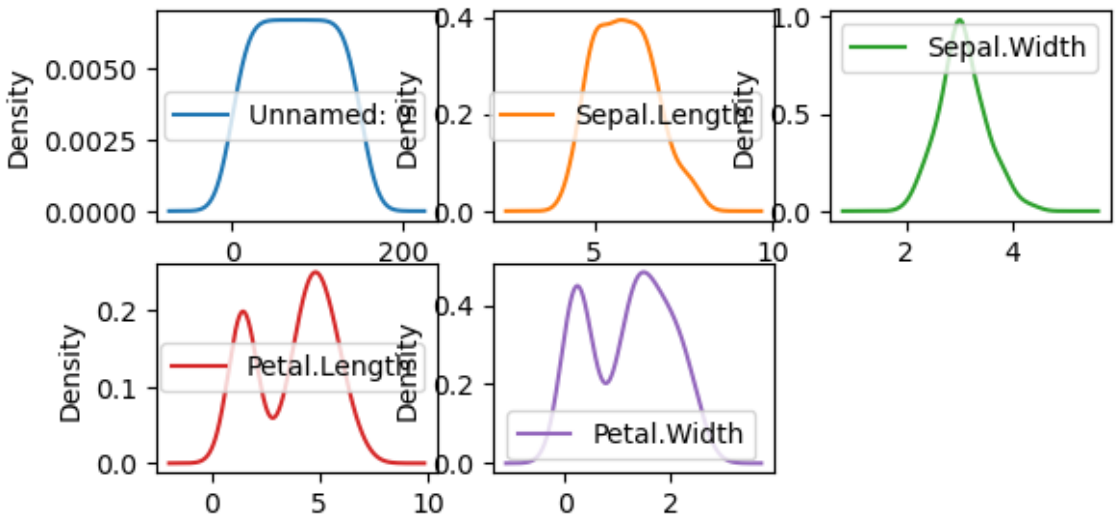


In [14]:
```
iris.hist()
plt.show()
```



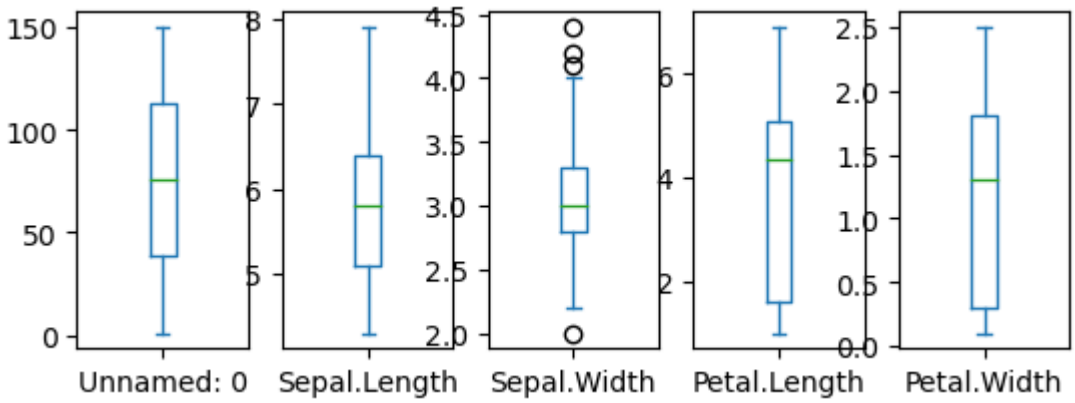
In [15]:
```
iris.plot(kind ='density',subplots = True, layout =(3,3),sharex = False)
```

Out[15]:
```
array([[<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
        <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
        <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
        <AxesSubplot:ylabel='Density'>]], dtype=object)
```



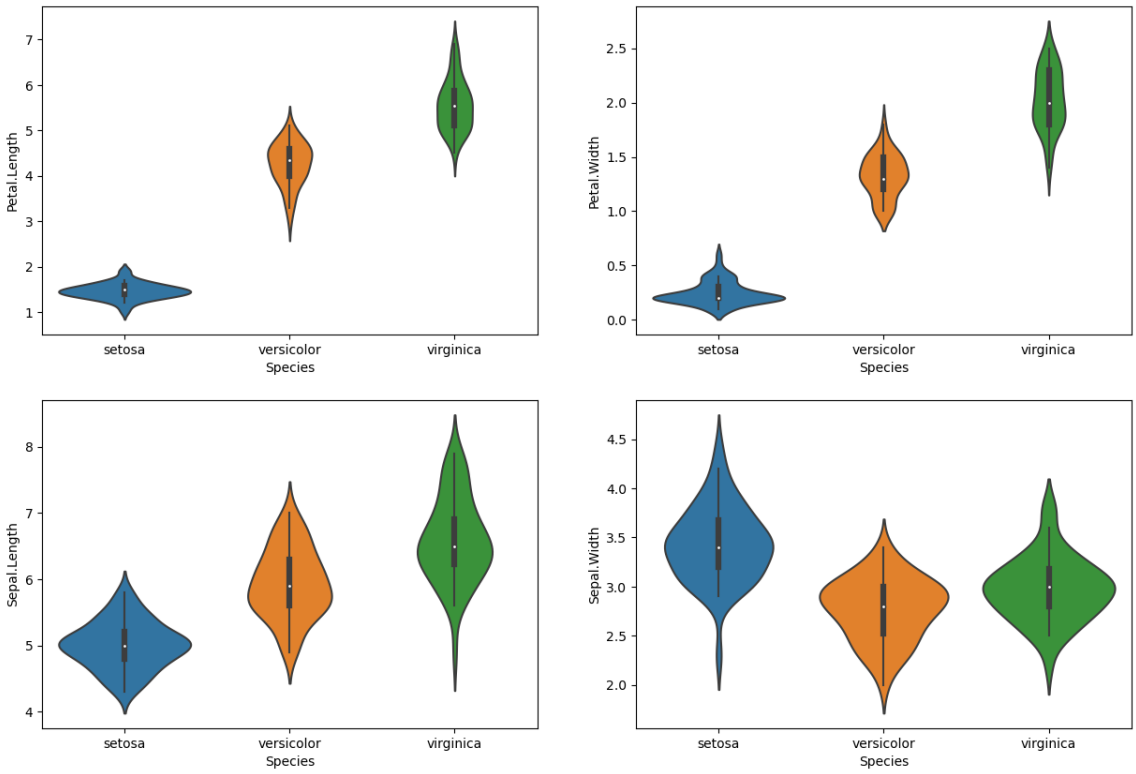
In [16]:
```
iris.plot(kind ='box',subplots = True, layout =(2,5),sharex = False)
```

Out[16]:
```
Unnamed: 0         AxesSubplot(0.125,0.53;0.133621x0.35)
Sepal.Length    AxesSubplot(0.285345,0.53;0.133621x0.35)
Sepal.Width      AxesSubplot(0.44569,0.53;0.133621x0.35)
Petal.Length    AxesSubplot(0.606034,0.53;0.133621x0.35)
Petal.Width     AxesSubplot(0.766379,0.53;0.133621x0.35)
dtype: object
```

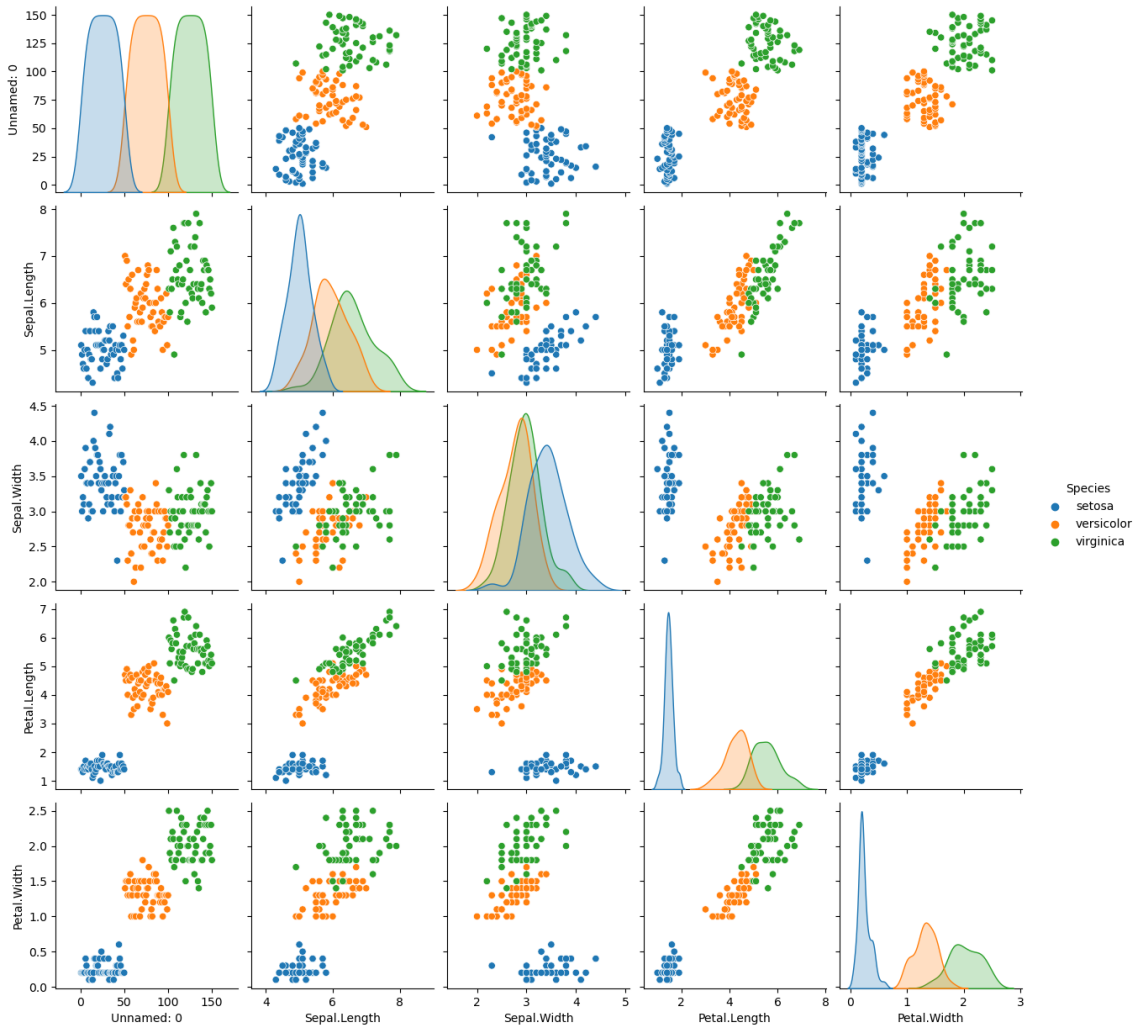


In [17]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='Petal.Length',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='Petal.Width',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='Sepal.Length',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='Sepal.Width',data=iris)
```

Out[17]:
```
<AxesSubplot:xlabel='Species', ylabel='Sepal.Width'>
```



In [18]:
```
sns.pairplot(iris,hue='Species');
```



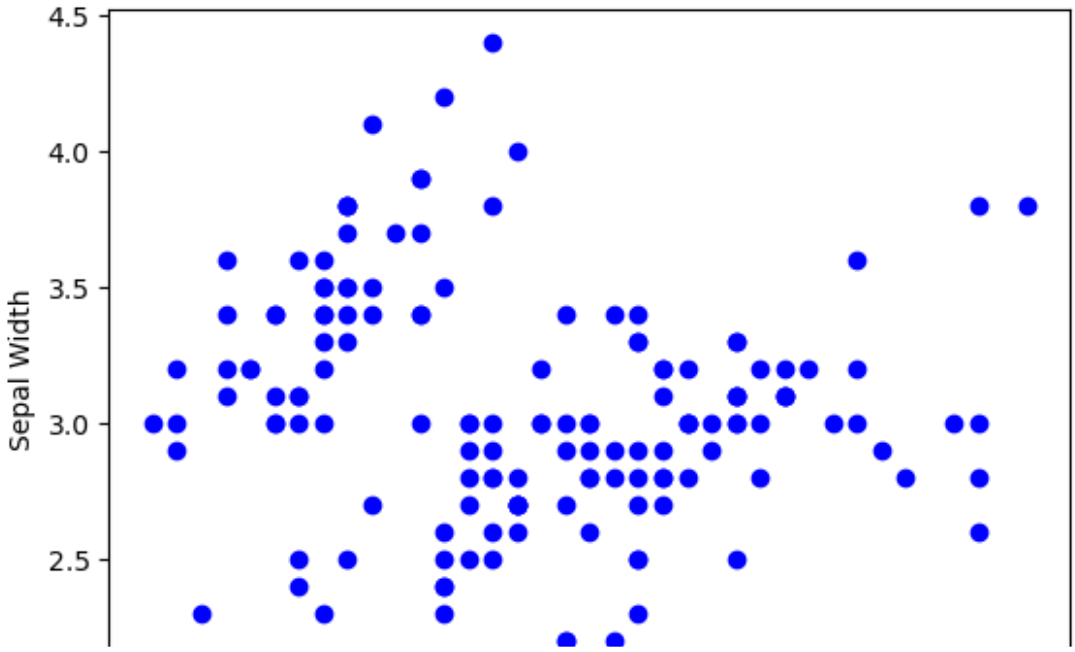
In [20]:
```
X = iris['Sepal.Length'].values.reshape(-1,1)
print(X)
```

```
[[5.1]
 [4.9]
 [4.7]
 [4.6]
 [5. ]
 [5.4]
 [4.6]
 [5. ]
 [4.4]
 [4.9]
 [5.4]
 [4.8]
 [4.8]
 [4.3]
 [5.8]
 [5.7]
 [5.4]
 [5.1]
 [5.7]
```

In [21]:
```
Y = iris['Sepal.Width'].values.reshape(-1,1)
print(Y)
```

```
[[3.5]
 [3. ]
 [3.2]
 [3.1]
 [3.6]
 [3.9]
 [3.4]
 [3.4]
 [2.9]
 [3.1]
 [3.7]
 [3.4]
 [3. ]
 [3. ]
 [4. ]
 [4.4]
 [3.9]
 [3.5]
 [3.8]
```

In [22]:
```
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(X,Y,color='b')
plt.show()
```



In [23]:
```
#Correlation
corr_mat = iris.corr()
print(corr_mat)
```

```
              Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.W
idth
Unnamed: 0      1.000000      0.716676    -0.402301      0.882637     0.90
0027
Sepal.Length    0.716676      1.000000    -0.117570      0.871754     0.81
7941
Sepal.Width    -0.402301     -0.117570     1.000000     -0.428440    -0.36
6126
Petal.Length    0.882637      0.871754    -0.428440      1.000000     0.96
2865
Petal.Width     0.900027      0.817941    -0.366126      0.962865     1.00
0000
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
```

In [25]:
```
train, test = train_test_split(iris, test_size = 0.25)
print(train.shape)
print(test.shape)
```

```
(112, 6)
(38, 6)
```

In [26]:
```
train_X = train[['Sepal.Length', 'Sepal.Width', 'Petal.Length',
                 'Petal.Width']]
train_y = train.Species

test_X = test[['Sepal.Length', 'Sepal.Width', 'Petal.Length',
                 'Petal.Width']]
test_y = test.Species
```

In [27]:
```
train_X.head()
```

Out[27]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
22	4.6	3.6	1.0	0.2
99	5.7	2.8	4.1	1.3
49	5.0	3.3	1.4	0.2
34	4.9	3.1	1.5	0.2
121	5.6	2.8	4.9	2.0

In [28]:
```
test_y.head()
```

Out[28]:
```
28         setosa
58     versicolor
38         setosa
72     versicolor
110     virginica
Name: Species, dtype: object
```

In [29]:
```
test_y.head()
```

Out[29]:
```
28         setosa
58     versicolor
38         setosa
72     versicolor
110     virginica
Name: Species, dtype: object
```

In [30]:
```
#Using LogisticRegression
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('Accuracy:',metrics.accuracy_score(prediction,test_y))
```

```
Accuracy: 0.9473684210526315
```

In [31]:
```
#Confusion matrix
from sklearn.metrics import confusion_matrix,classification_report
confusion_mat = confusion_matrix(test_y,prediction)
print("Confusion matrix: \n",confusion_mat)
print(classification_report(test_y,prediction))
```

```
Confusion matrix:
 [[10  0  0]
 [ 0 14  0]
 [ 0  2 12]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.88      1.00      0.93        14
   virginica       1.00      0.86      0.92        14

    accuracy                           0.95        38
   macro avg       0.96      0.95      0.95        38
weighted avg       0.95      0.95      0.95        38
```

In [32]:
```
#Using Support Vector
from sklearn.svm import SVC
model1 = SVC()
model1.fit(train_X,train_y)

pred_y = model1.predict(test_X)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(test_y,pred_y))
```

```
Acc= 0.9473684210526315
```

In [33]:
```
#Using KNN Neighbors
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(train_X,train_y)
y_pred2 = model2.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred2))
```

```
Accuracy Score: 0.9473684210526315
```

In [34]:
```
#Using GaussianNB
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(train_X,train_y)
y_pred3 = model3.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred3))
```

```
Accuracy Score: 0.9473684210526315
```

In [35]:
```
#Using Decision Tree
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(train_X,train_y)
y_pred4 = model4.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred4))
```

```
Accuracy Score: 0.9473684210526315
```

In [36]:
```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes
    'Score': [0.947,0.947,0.947,0.947,0.921]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[36]:

Score	Model
0.947	Logistic Regression
0.947	Support Vector Machines
0.947	Naive Bayes
0.947	KNN
0.921	Decision Tree

In [ ]: