```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


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

iris=pd.read_csv("iris.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
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica
```

```
[150 rows x 6 columns]
```

```python
print(iris.shape)
```

```
(150, 6)
```

```python
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
```

```python
#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
```

```
iris.head()
```

```
   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
```

```
iris.head(150)
```

```
     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
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 6 columns]
```

```
iris.tail(100)
```

```
     Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
\
50           51           7.0          3.2           4.7          1.4

51           52           6.4          3.2           4.5          1.5

52           53           6.9          3.1           4.9          1.5

53           54           5.5          2.3           4.0          1.3

54           55           6.5          2.8           4.6          1.5

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

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
50   versicolor
51   versicolor
52   versicolor
53   versicolor
54   versicolor
..          ...
145   virginica
```

```
146   virginica
147   virginica
148   virginica
149   virginica

[100 rows x 6 columns]
```

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

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

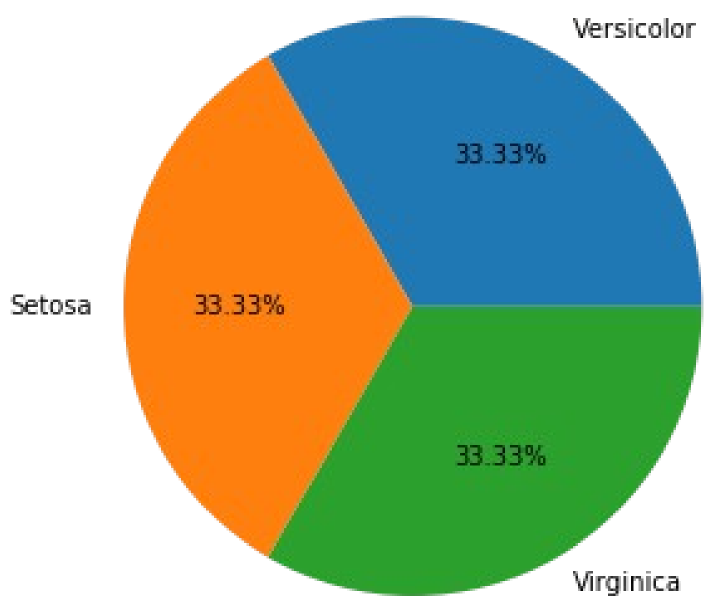
```
n1 = len(iris[iris['Species'] == 'virginica'])
print("No of Virginica in Dataset:",n1)
```

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

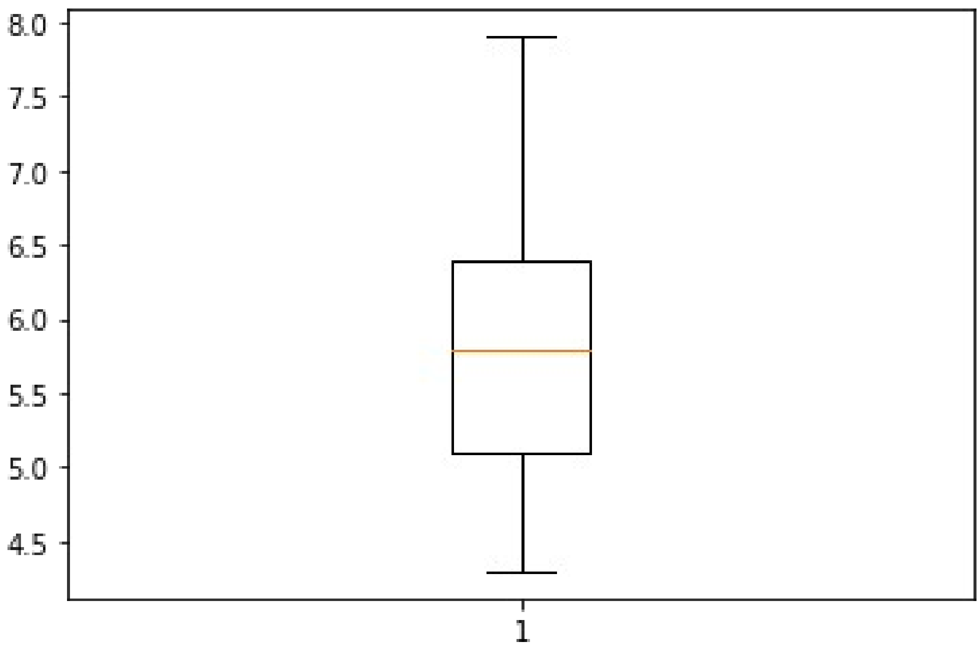
```
n2 = len(iris[iris['Species'] == 'setosa'])
print("No of Setosa in Dataset:",n2)
```

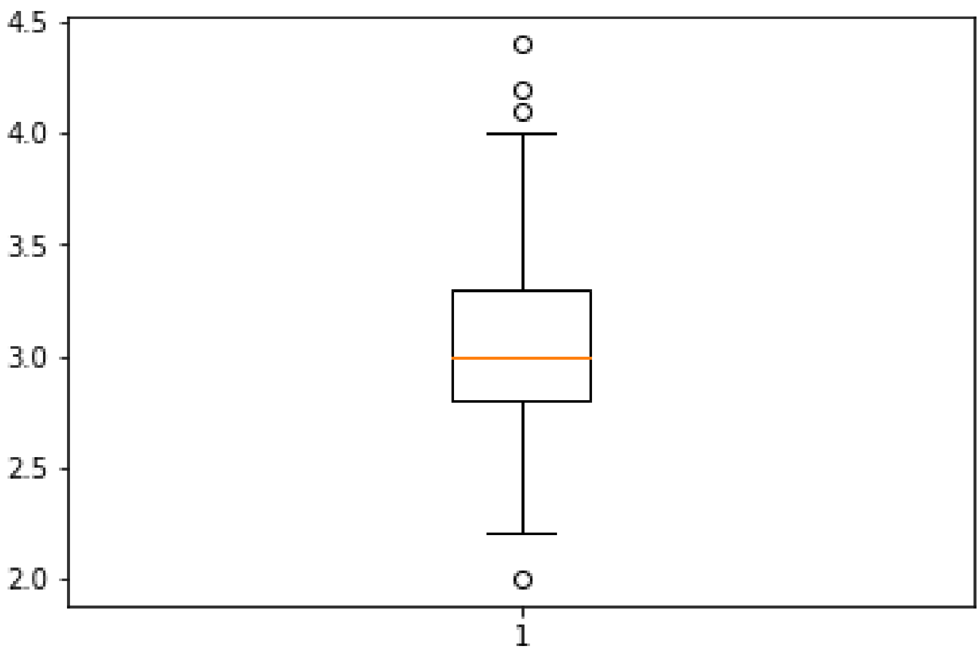
```
No of Setosa in Dataset: 50
```

```
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()
```

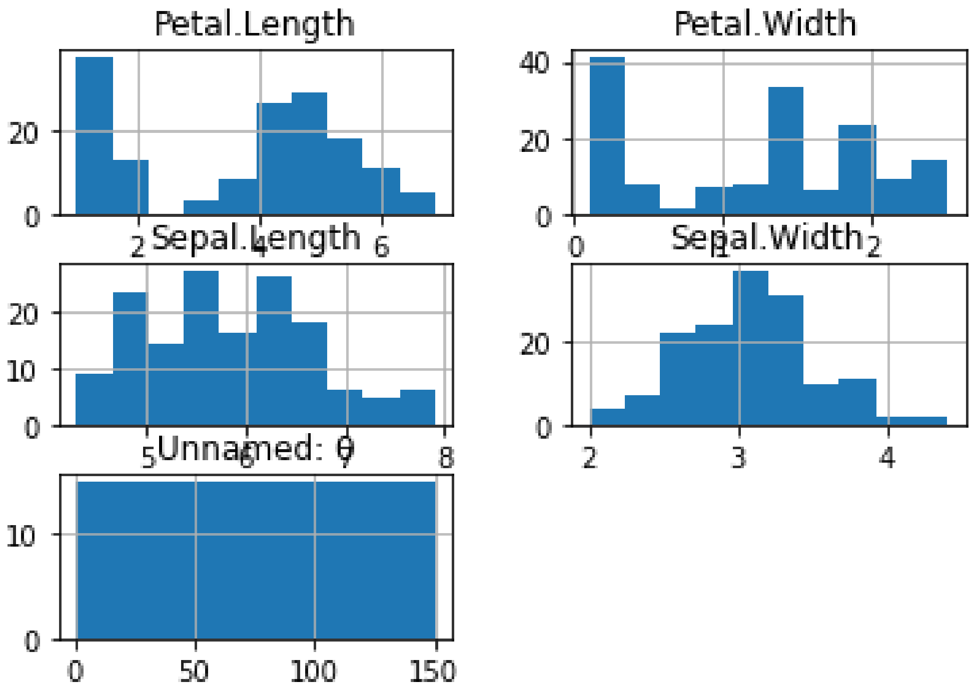


```python
#Checking for outliars
import matplotlib.pyplot as plt
plt.figure(1)
plt.boxplot([iris['Sepal.Length']])
plt.figure(2)
plt.boxplot([iris['Sepal.Width']])
plt.show()
```



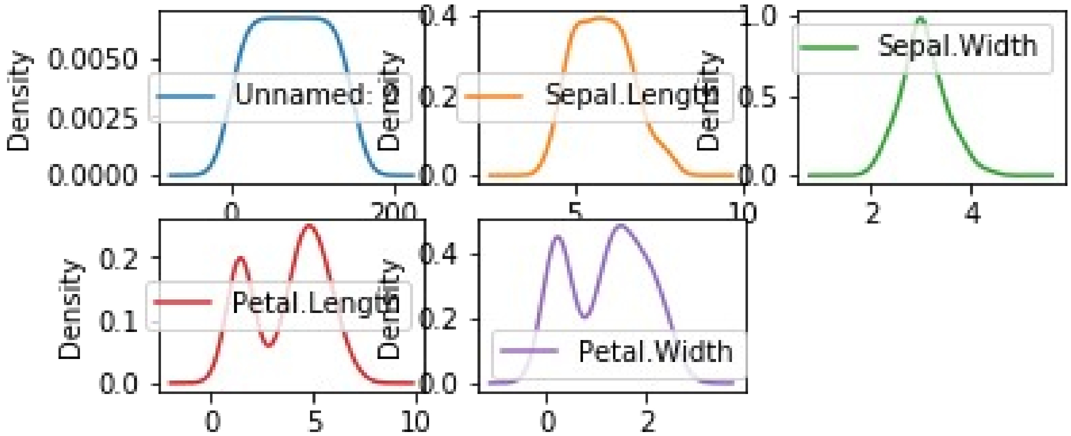


```
iris.hist()
plt.show()
```



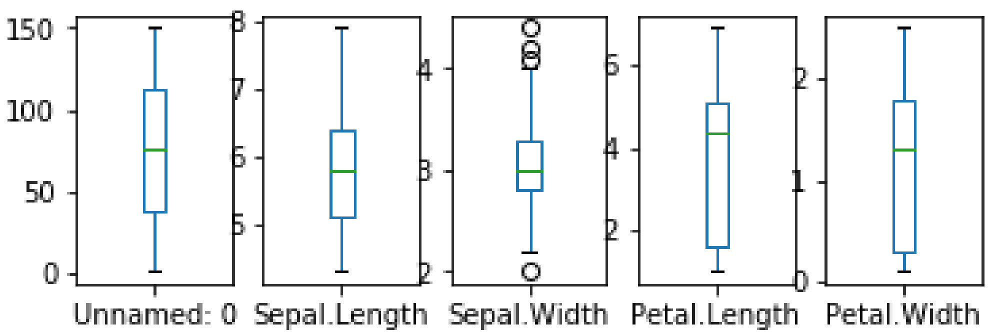

```
iris.plot(kind ='density',subplots = True, layout =(3,3),sharex =
False)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x0000028CA9FCC448>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA02DC88>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA05F388>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA094D88>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA0CE788>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA109188>],
       [<matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA142248>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA17B408>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000028CAA181F88>]],
      dtype=object)
```



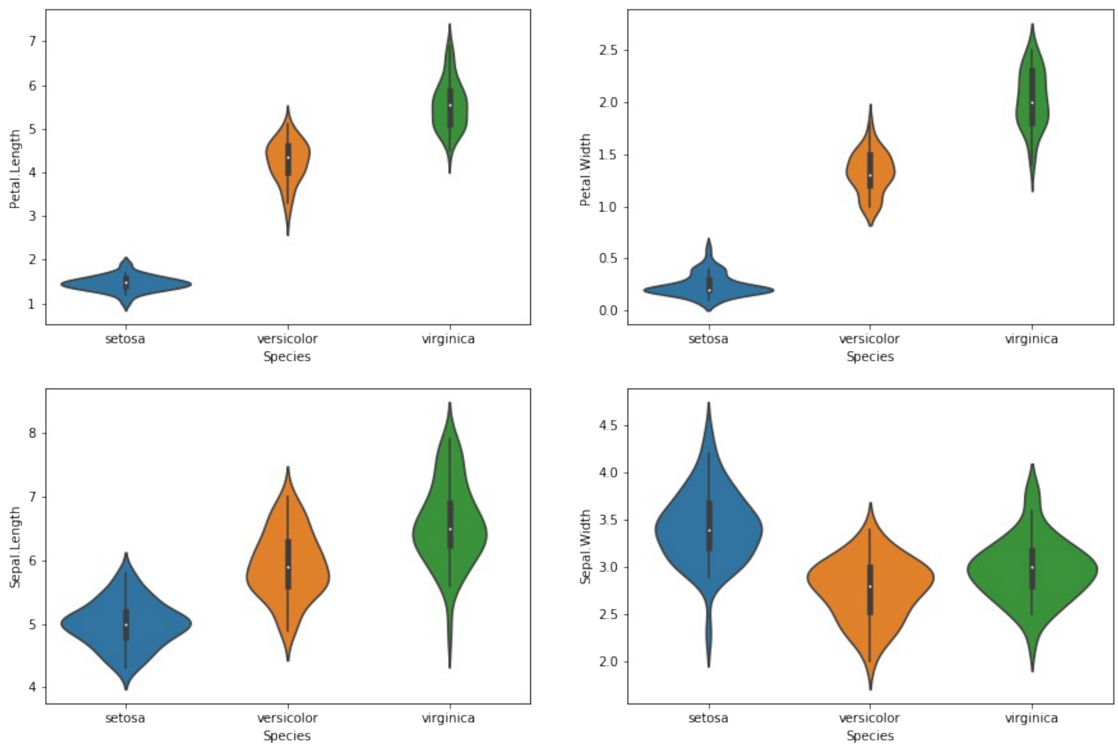

```
iris.plot(kind ='box',subplots = True, layout =(2,5),sharex = False)
```

```
Unnamed: 0         AxesSubplot(0.125,0.536818;0.133621x0.343182)
Sepal.Length    AxesSubplot(0.285345,0.536818;0.133621x0.343182)
Sepal.Width      AxesSubplot(0.44569,0.536818;0.133621x0.343182)
Petal.Length    AxesSubplot(0.606034,0.536818;0.133621x0.343182)
Petal.Width     AxesSubplot(0.766379,0.536818;0.133621x0.343182)
dtype: object
```

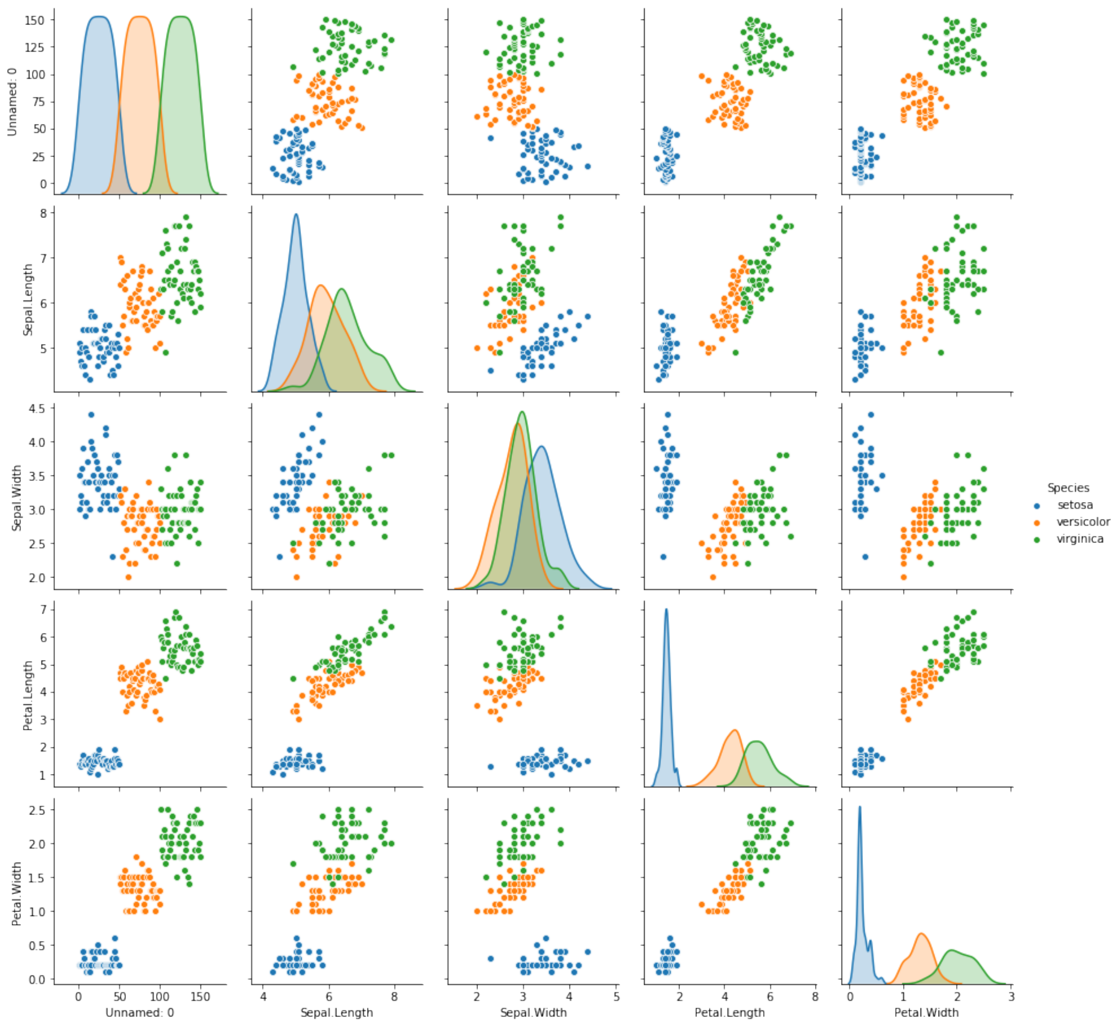



```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='Petal.Length',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='Petal.Width',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='Sepal.Length',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='Sepal.Width',data=iris)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

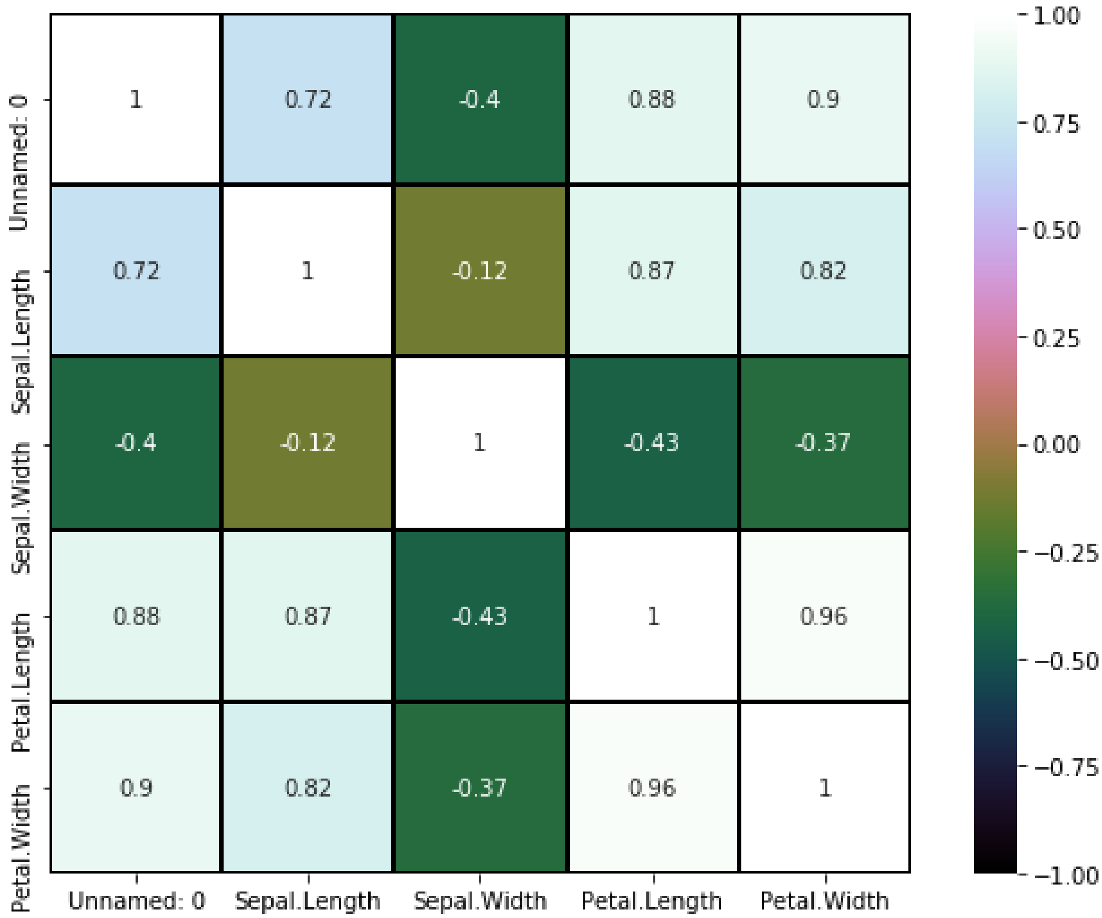
```
<matplotlib.axes._subplots.AxesSubplot at 0x28caa3f9688>
```




```
sns.pairplot(iris,hue='Species');
```




```
#Heat Maps
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.heatmap(iris.corr(),annot=True,cmap='cubehelix',linewidths=1,l
inecolor='k',square=True,mask=False, vmin=-1,
vmax=1,cbar_kws={"orientation": "vertical"},cbar=True)
```




```
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]
```

[5.1]
[5.4]
[5.1]
[4.6]
[5.1]
[4.8]
[5.]
[5.]
[5.2]
[5.2]
[4.7]
[4.8]
[5.4]
[5.2]
[5.5]
[4.9]
[5.]
[5.5]
[4.9]
[4.4]
[5.1]
[5.]
[4.5]
[4.4]
[5.]
[5.1]
[4.8]
[5.1]
[4.6]
[5.3]
[5.]
[7.]
[6.4]
[6.9]
[5.5]
[6.5]
[5.7]
[6.3]
[4.9]
[6.6]
[5.2]
[5.]
[5.9]
[6.]
[6.1]
[5.6]
[6.7]
[5.6]
[5.8]
[6.2]

[5.6]
[5.9]
[6.1]
[6.3]
[6.1]
[6.4]
[6.6]
[6.8]
[6.7]
[6.]
[5.7]
[5.5]
[5.5]
[5.8]
[6.]
[5.4]
[6.]
[6.7]
[6.3]
[5.6]
[5.5]
[5.5]
[6.1]
[5.8]
[5.]
[5.6]
[5.7]
[5.7]
[6.2]
[5.1]
[5.7]
[6.3]
[5.8]
[7.1]
[6.3]
[6.5]
[7.6]
[4.9]
[7.3]
[6.7]
[7.2]
[6.5]
[6.4]
[6.8]
[5.7]
[5.8]
[6.4]
[6.5]
[7.7]
[7.7]

```
 [6. ]
 [6.9]
 [5.6]
 [7.7]
 [6.3]
 [6.7]
 [7.2]
 [6.2]
 [6.1]
 [6.4]
 [7.2]
 [7.4]
 [7.9]
 [6.4]
 [6.3]
 [6.1]
 [7.7]
 [6.3]
 [6.4]
 [6. ]
 [6.9]
 [6.7]
 [6.9]
 [5.8]
 [6.8]
 [6.7]
 [6.7]
 [6.3]
 [6.5]
 [6.2]
 [5.9]]
```

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

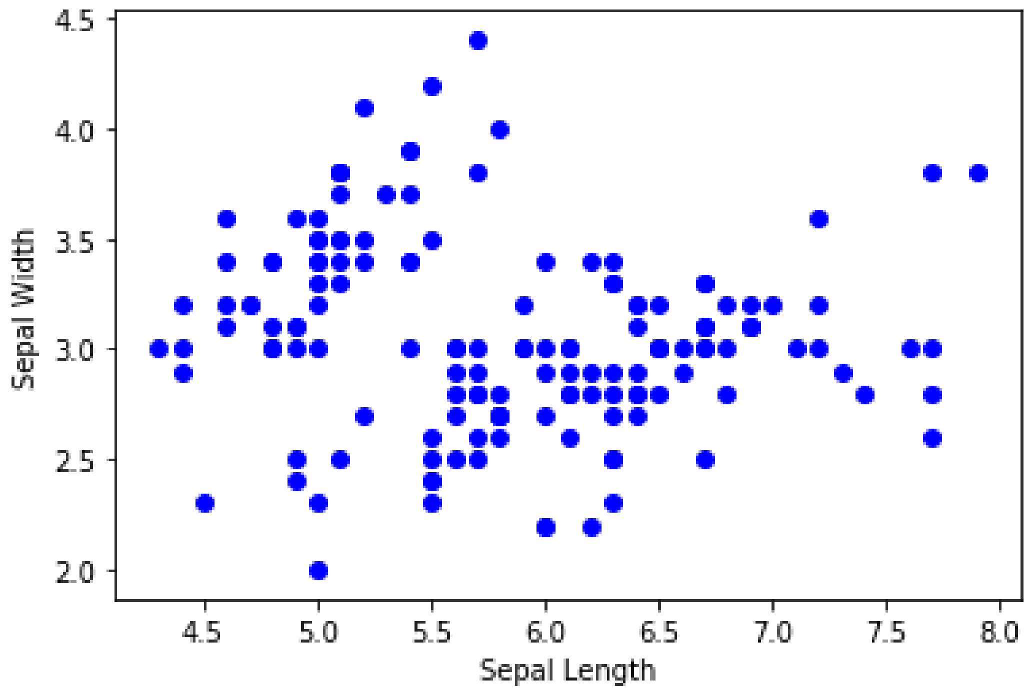
```
[[3.5]
 [3. ]
 [3.2]
 [3.1]
 [3.6]
 [3.9]
 [3.4]
 [3.4]
 [2.9]
 [3.1]
 [3.7]
 [3.4]
 [3. ]
 [3. ]
 [4. ]
 [4.4]
```

```
[3.9]
[3.5]
[3.8]
[3.8]
[3.4]
[3.7]
[3.6]
[3.3]
[3.4]
[3. ]
[3.4]
[3.5]
[3.4]
[3.2]
[3.1]
[3.4]
[4.1]
[4.2]
[3.1]
[3.2]
[3.5]
[3.6]
[3. ]
[3.4]
[3.5]
[2.3]
[3.2]
[3.5]
[3.8]
[3. ]
[3.8]
[3.2]
[3.7]
[3.3]
[3.2]
[3.2]
[3.1]
[2.3]
[2.8]
[2.8]
[3.3]
[2.4]
[2.9]
[2.7]
[2. ]
[3. ]
[2.2]
[2.9]
[2.9]
[3.1]
```

```
[3. ]
[2.7]
[2.2]
[2.5]
[3.2]
[2.8]
[2.5]
[2.8]
[2.9]
[3. ]
[2.8]
[3. ]
[2.9]
[2.6]
[2.4]
[2.4]
[2.7]
[2.7]
[3. ]
[3.4]
[3.1]
[2.3]
[3. ]
[2.5]
[2.6]
[3. ]
[2.6]
[2.3]
[2.7]
[3. ]
[2.9]
[2.9]
[2.5]
[2.8]
[3.3]
[2.7]
[3. ]
[2.9]
[3. ]
[3. ]
[2.5]
[2.9]
[2.5]
[3.6]
[3.2]
[2.7]
[3. ]
[2.5]
[2.8]
[3.2]
```

```
 [3. ]
 [3.8]
 [2.6]
 [2.2]
 [3.2]
 [2.8]
 [2.8]
 [2.7]
 [3.3]
 [3.2]
 [2.8]
 [3. ]
 [2.8]
 [3. ]
 [2.8]
 [3.8]
 [2.8]
 [2.8]
 [2.6]
 [3. ]
 [3.4]
 [3.1]
 [3. ]
 [3.1]
 [3.1]
 [3.1]
 [2.7]
 [3.2]
 [3.3]
 [3. ]
 [2.5]
 [3. ]
 [3.4]
 [3. ]]
```

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




```
#Correlation
corr_mat = iris.corr()
print(corr_mat)
```

```
              Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length
Petal.Width
Unnamed: 0      1.000000      0.716676    -0.402301      0.882637
0.900027
Sepal.Length    0.716676      1.000000    -0.117570      0.871754
0.817941
Sepal.Width    -0.402301     -0.117570     1.000000     -0.428440    -
0.366126
Petal.Length    0.882637      0.871754    -0.428440      1.000000
0.962865
Petal.Width     0.900027      0.817941    -0.366126      0.962865
1.000000
```

```
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

train, test = train_test_split(iris, test_size = 0.25)
print(train.shape)
print(test.shape)
```

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

```
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
```

```
train_X.head()
```

```
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
68           6.2          2.2           4.5          1.5
42           4.4          3.2           1.3          0.2
29           4.7          3.2           1.6          0.2
53           5.5          2.3           4.0          1.3
61           5.9          3.0           4.2          1.5
```

```
test_y.head()
```

```
32        setosa
63    versicolor
44        setosa
49        setosa
96    versicolor
Name: Species, dtype: object
```

```
test_y.head()
```

```
32        setosa
63    versicolor
44        setosa
49        setosa
96    versicolor
Name: Species, dtype: object
```

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

```
Accuracy: 0.9736842105263158
```

```
#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:
 [[16  0  0]
 [ 0 14  1]
 [ 0  0  7]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       1.00      0.93      0.97        15
   virginica       0.88      1.00      0.93         7

    accuracy                           0.97        38
   macro avg       0.96      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

```
#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= 0.9736842105263158
```

```
#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: 0.9736842105263158
```

```
#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: 1.0
```

```
#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: 0.9736842105263158
```

```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive
Bayes','KNN' ,'Decision Tree'],
    '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)
```

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

```
#Hence I will use Naive Bayes algorithms for training my model.
```