Importing Libraries

In [75]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

import warnings
warnings.filterwarnings('ignore')
```

In [76]:
```
iris = pd.read_csv("iris.csv")
print(iris)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
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       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [77]:
```
iris.head()
```

Out[77]:

	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 [78]:
```
iris.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 [79]:
```
iris.describe()
```

Out[79]:

	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 [80]:
```
iris.isnull().sum()
```

Out[80]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [81]:
```
iris.columns
```

Out[81]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [82]:
```
n = len(iris[iris['Species'] == 'versicolor'])
print("No of Versicolor in Dataset:",n)
```

```
No of Versicolor in Dataset: 0
```

In [83]:
```
n1 = len(iris[iris['Species'] == 'virginica'])
print("No of Virginica in Dataset:",n1)
```

```
No of Virginica in Dataset: 0
```

In [84]:
```
n2 = len(iris[iris['Species'] == 'virginica'])
print("No of Virginica in Dataset:",n2)
```

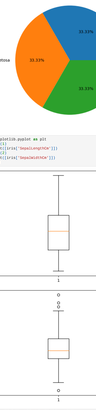
```
No of Virginica in Dataset: 0
```

Data Visualization

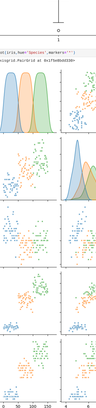
In [85]:
```
iris['Species'].value_counts()
```

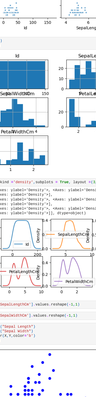
Out[85]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [86]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [34,33,33]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```

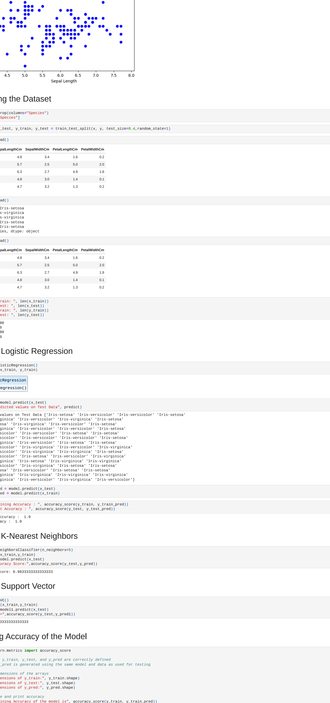


In [87]:
```
import matplotlib.pyplot as plt
plt.figure(1)
plt.boxplot(iris['SepalLengthCm'])
plt.figure(2)
plt.boxplot(iris['SepalWidthCm'])
plt.show()
```





In [88]:
```
sns.pairplot(iris,hue='Species',markers='*')
```



In [89]:
```
iris.hist()
plt.show()
```

In [90]:
```
iris.plot(kind ='density',subplots = True, layout =(3,3),sharex = False)
```


In [91]:
```
X = iris['SepalLengthCm'].values.reshape(-1,1)
```

In [92]:
```
Y = iris['SepalWidthCm'].values.reshape(-1,1)
```

In [93]:
```
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(X,Y,color='b')
plt.show()
```


Splitting the Dataset

In [94]:
```
X = iris.drop(columns="Species")
y = iris['Species']
```

In [95]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,random_state=1)
```

In [96]:
```
X_train.head()
```

Out[96]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
16	17	5.4	3.9	1.3	0.4
113	114	5.7	2.5	5.0	2.0
120	121	6.9	3.2	5.7	2.3
18	19	5.7	3.8	1.7	0.3
4	5	5.0	3.6	1.4	0.2

In [97]:
```
y_train.head()
```

Out[97]:
```
16     Iris-setosa
113    Iris-setosa
120    Iris-setosa
18     Iris-setosa
4      Iris-setosa
Name: Species, dtype: object
```

In [98]:
```
X_test.head()
```

Out[98]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
31	32	5.4	3.4	1.5	0.4
113	114	5.7	2.5	5.0	2.0
120	121	6.9	3.2	5.7	2.3
18	19	5.7	3.8	1.7	0.3
4	5	5.0	3.6	1.4	0.2

In [99]:
```
print("X_train:",len(X_train))
print("X_test:",len(X_test))
print("y_train:",len(y_train))
print("y_test:",len(y_test))
```

```
X_train:  90
X_test:  60
y_train:  90
y_test:  60
```

Using Logistic Regression

In [100]:
```
model= LogisticRegression()
model.fit(X_train, y_train)
```

Out[100]:
```
LogisticRegression()
```

In [101]:
```
LogisticRegression()
```

In [102]:
```
predict = model.predict(X_test)
print("Predicted values on Test Data",predict)
```

```
Predicted values on Test Data ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor']
```

In [103]:
```
y_pred_prob = model.predict(X_test)
y_train_pred = model.predict(X_train)
```

In [104]:
```
print("Training Accuracy  : ", accuracy_score(y_train, y_train_pred))
print("Test Accuracy  : ", accuracy_score(y_test, y_test_pred))
```

```
Training Accuracy  :  1.0
Test Accuracy  :  1.0
```

Using K-Nearest Neighbors

In [105]:
```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy Score: 0.9833333333333333
```

Using Support Vector

In [106]:
```
model = SVC()
model.fit(x_train,y_train)
y_pred1 = model.predict(x_test)
print("Acc=",accuracy_score(y_test,y_pred1))
```

```
Acc= 0.9833333333333333
```

Testing Accuracy of the Model

In [108]:
```
from sklearn.metrics import accuracy_score

# Assuming y_train, y_test, and y_pred are correctly defined
# Ensure y_pred is generated using the same model and data as used for testing

# Check dimensions of the arrays
print("Dimensions of y_train: {}".format(y_train.shape))
print("Dimensions of y_test: {}".format(y_test.shape))
print("Dimensions of y_pred: {}".format(y_pred.shape))

# Calculate and print accuracy
print("Training Accuracy of the model is", accuracy_score(y_train, y_train_pred))
print("Testing Accuracy of the model is", accuracy_score(y_test, y_pred))
```

```
Dimensions of y_train: (90,)
Dimensions of y_test: (60,)
Dimensions of y_pred: (60,)
Training Accuracy of the model is 1.0
Testing Accuracy of the model is 0.9833333333333333
```

In [ ]:
```

```