LetsGrowMore

Priyanshi Badaya

Data Science Intern

Prediction using Decision Tree Algorithm

Data Set Link - https://bit.ly/3kXTdox

Importing Libraries

In [1]:
```
# Importing the required Libraries

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
import sklearn.metrics as sm

import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
from IPython.display import Image

from sklearn.metrics import classification_report, plot_confusion_matrix,accuracy_score
from sklearn.tree import plot_tree
```

Loading the Dataset

In [2]:
```
iris = load_iris()
X=iris.data[:,:]
y=iris.target
```

Exploratory Data Analysis

In [3]:
```
data=pd.DataFrame(iris['data'],columns=["Petal length","Petal Width","Sepal Length","Sepal Width"])
data['Species']=iris['target']
data['Species']=data['Species'].apply(lambda x: iris['target_names'][x])

data.head()
```

Out[3]:

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

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Petal length  150 non-null    float64
 1   Petal Width   150 non-null    float64
 2   Sepal Length  150 non-null    float64
 3   Sepal Width   150 non-null    float64
 4   Species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:
```
data.describe()
```

Out[6]:

	Petal length	Petal Width	Sepal Length	Sepal Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
Petal length    0
Petal Width     0
Sepal Length    0
Sepal Width     0
Species         0
dtype: int64
```

In [8]:
```
data.corr()
```

Out[8]:

	Petal length	Petal Width	Sepal Length	Sepal Width
Petal length	1.000000	-0.117570	0.871754	0.817941
Petal Width	-0.117570	1.000000	-0.428440	-0.366126
Sepal Length	0.871754	-0.428440	1.000000	0.962865
Sepal Width	0.817941	-0.366126	0.962865	1.000000

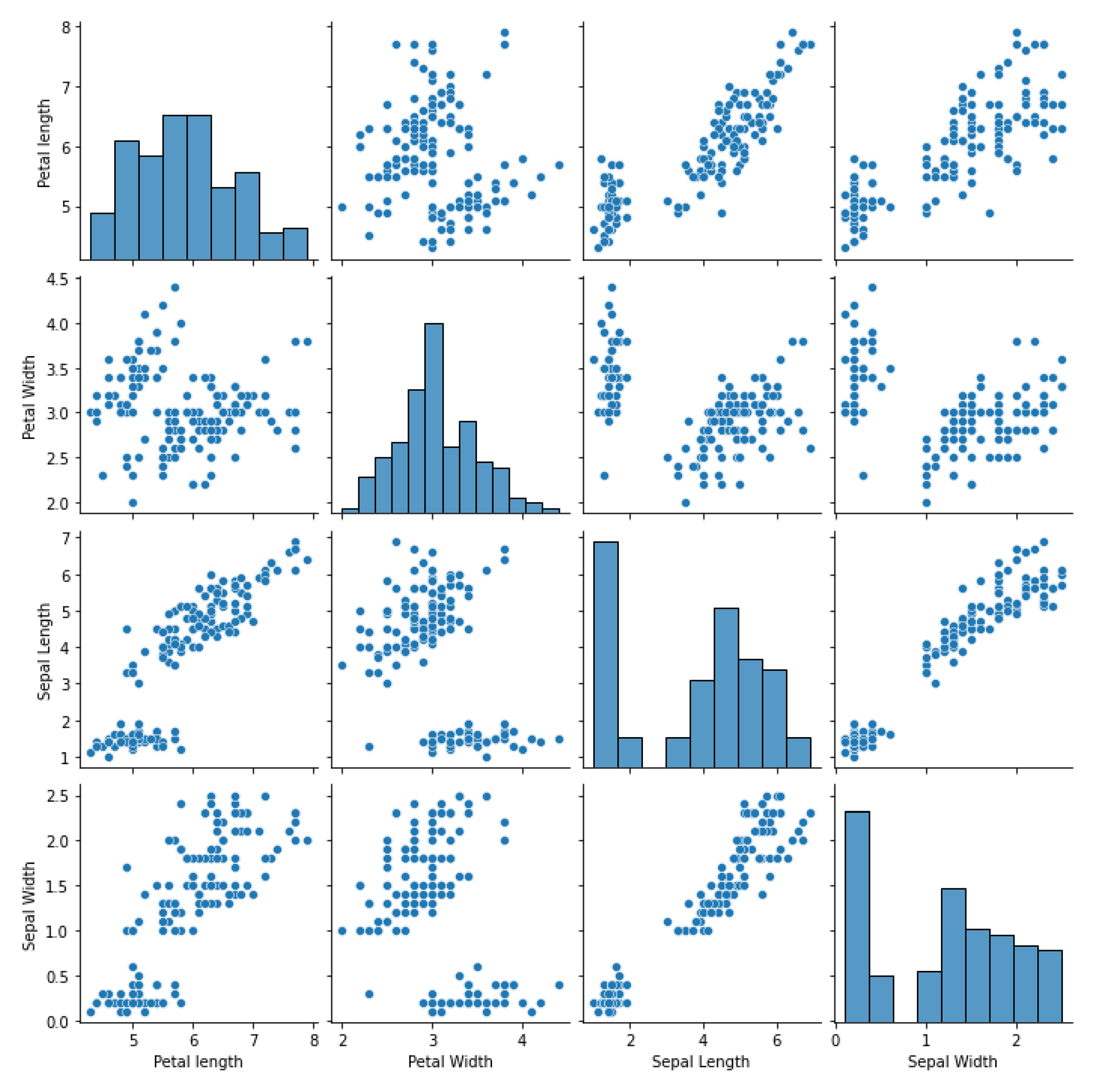
**From this correlation table, we can make the following inferences

1.The sepal length has a very high correlation with petal length and petal width 2.In the same vein, petal length is highly correlated with petal width

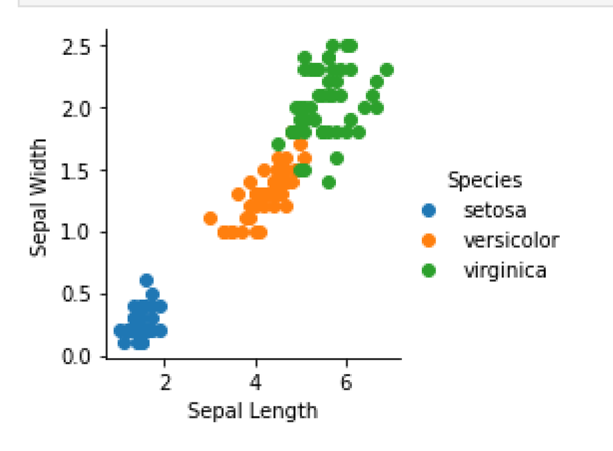
**Data Visualization comparing various features

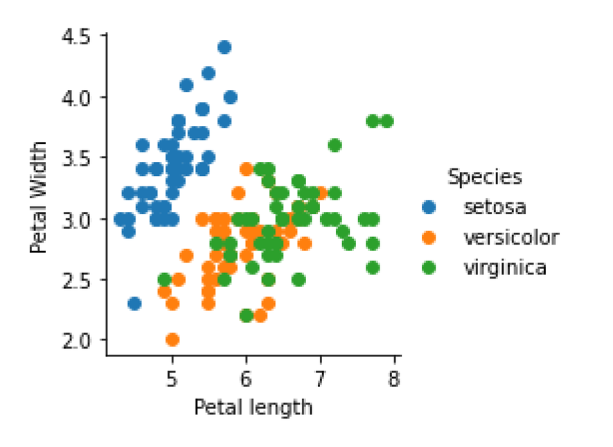
In [9]:
```
sns.pairplot(data)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1f04451c400>
```



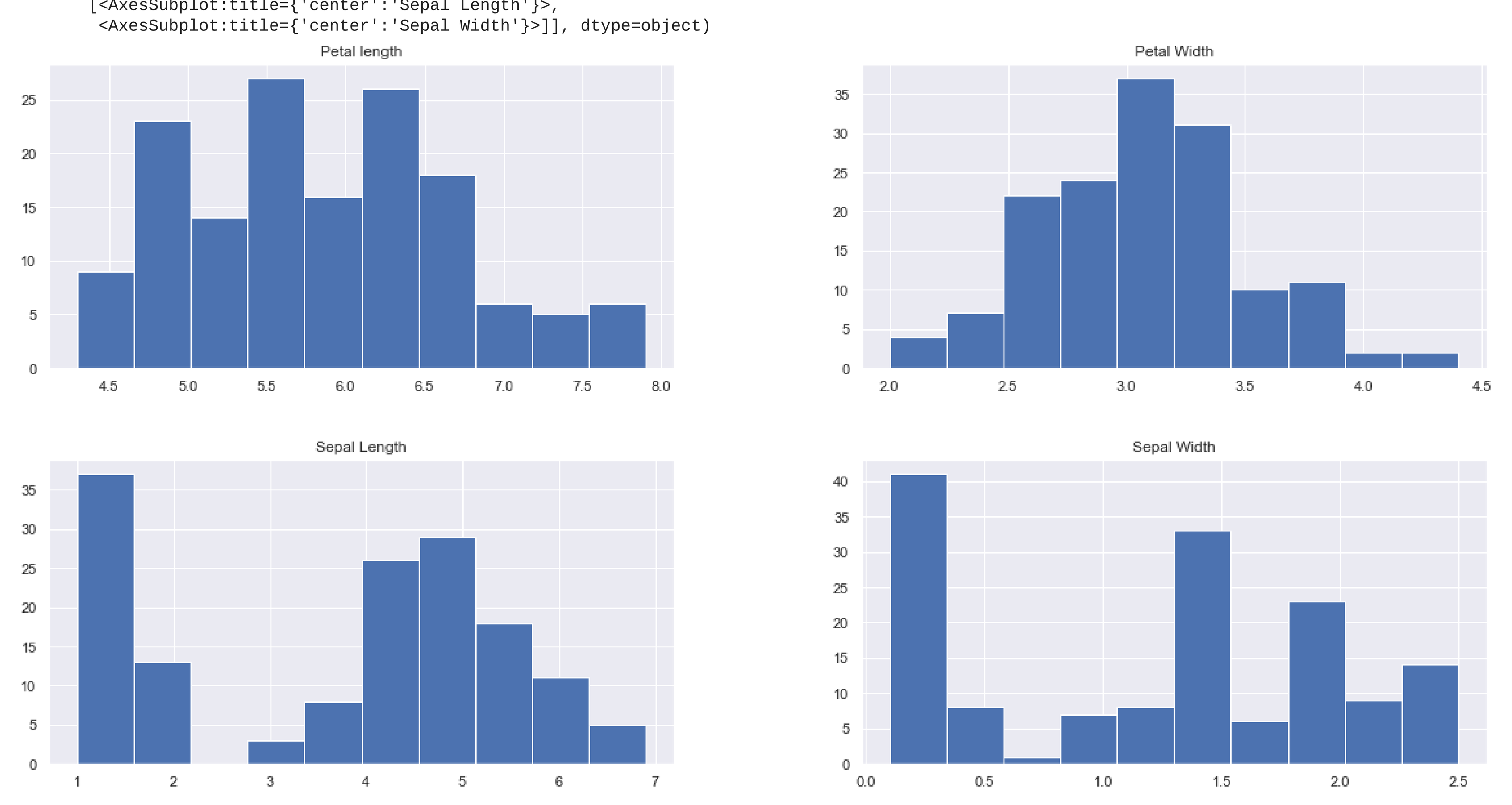
In [10]:
```
# Scatter plot of data based on Sepal Length and Width features
sns.FacetGrid(data,hue='Species').map(plt.scatter,'Sepal Length','Sepal Width').add_legend()
plt.show()

# Scatter plot of data based on Petal Length and Width features
sns.FacetGrid(data,hue='Species').map(plt.scatter,'Petal length','Petal Width').add_legend()
plt.show()
```





In [11]:
```
sns.set(rc={'figure.figsize':(20,10)})
data.hist()
```

Out[11]:
```
array([[<AxesSubplot:title={'center':'Petal length'}>,
        <AxesSubplot:title={'center':'Petal Width'}>],
       [<AxesSubplot:title={'center':'Sepal Length'}>,
        <AxesSubplot:title={'center':'Sepal Width'}>]], dtype=object)
```



In [12]:
```
#Importing the required library
from sklearn.tree import DecisionTreeClassifier
```

In [13]:
```
#Importing necessary libraries for data preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [14]:
```
# Importing libraries in Python
import sklearn.datasets as datasets
import pandas as pd

# Loading the iris dataset
iris=datasets.load_iris()

# Forming the iris dataframe
df=pd.DataFrame(iris.data, columns=iris.feature_names)
print(df.head(5))

y=iris.target
print(y)
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Decision Tree Model Training

In [16]:
```
# Model Training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
tree_classifier = DecisionTreeClassifier()
tree_classifier.fit(X_train,y_train)
print("Training Complete.")
y_pred = tree_classifier.predict(X_test)
```

```
Training Complete.
```

Visualizing the Trained Model

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)


print('Decision Tree Classifer Created Successfully')
```

```
Decision Tree Classifer Created Successfully
```

In [18]:
```
import matplotlib.pyplot as plt
from sklearn import tree


a=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
b=['setosa','versicolor','virginica']

fig, axes = plt.subplots(nrows = 1, ncols = 1, figsize = (4,4), dpi = 300)

tree.plot_tree(dtree, feature_names = a, class_names = b, filled = True)
```

Out[18]:
```
[Text(0.5416666666666666, 0.9285714285714286, 'petal length (cm) <= 2.45\ngini = 0.666\nsamples = 100\nvalue = [31, 35, 34]\nclass = versicolor'),
 Text(0.4583333333333333, 0.7857142857142857, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]\nclass = setosa'),
 Text(0.625, 0.7857142857142857, 'petal width (cm) <= 1.75\ngini = 0.5\nsamples = 69\nvalue = [0, 35, 34]\nclass = versicolor'),
 Text(0.4166666666666667, 0.6428571428571429, 'petal length (cm) <= 5.35\ngini = 0.188\nsamples = 38\nvalue = [0, 34, 4]\nclass = versicolor'),
 Text(0.3333333333333333, 0.5, 'petal width (cm) <= 1.65\ngini = 0.105\nsamples = 36\nvalue = [0, 34, 2]\nclass = versicolor'),
 Text(0.16666666666666666, 0.35714285714285715, 'petal length (cm) <= 4.95\ngini = 0.057\nsamples = 34\nvalue = [0, 33, 1]\nclass = versicolor'),
 Text(0.08333333333333333, 0.21428571428571427, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]\nclass = versicolor'),
 Text(0.25, 0.21428571428571427, 'petal width (cm) <= 1.55\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]\nclass = versicolor'),
 Text(0.16666666666666666, 0.07142857142857142, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = virginica'),
 Text(0.3333333333333333, 0.07142857142857142, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = versicolor'),
 Text(0.5, 0.35714285714285715, 'petal length (cm) <= 4.75\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]\nclass = versicolor'),
 Text(0.4166666666666667, 0.21428571428571427, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = virginica'),
 Text(0.5833333333333334, 0.21428571428571427, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = versicolor'),
 Text(0.5, 0.5, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]\nclass = virginica'),
 Text(0.8333333333333334, 0.6428571428571429, 'petal length (cm) <= 4.85\ngini = 0.062\nsamples = 31\nvalue = [0, 1, 30]\nclass = virginica'),
 Text(0.75, 0.5, 'sepal width (cm) <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]\nclass = virginica'),
 Text(0.6666666666666666, 0.35714285714285715, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]\nclass = virginica'),
 Text(0.8333333333333334, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = versicolor'),
 Text(0.9166666666666666, 0.5, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]\nclass = virginica')]
```

petal length (cm) <= 2.45
gini = 0.666
samples = 100
value = [31, 35, 34]
class = versicolor
gini = 0.0
samples = 31
value = [31, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 69
value = [0, 35, 34]
class = versicolor
petal length (cm) <= 5.35
gini = 0.188
samples = 38
value = [0, 34, 4]
class = versicolor
petal length (cm) <= 4.85
gini = 0.062
samples = 31
value = [0, 1, 30]
class = virginica
petal width (cm) <= 1.65
gini = 0.105
samples = 36
value = [0, 34, 2]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 28
value = [0, 0, 28]
class = virginica
petal length (cm) <= 4.95
gini = 0.057
samples = 34
value = [0, 33, 1]
class = versicolor
petal length (cm) <= 4.75
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 32
value = [0, 32, 0]
class = versicolor
petal width (cm) <= 1.55
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor

Predicting the class output for some random values of petal and sepal length and width

Calculating the Model Accuracy

In [20]:
```
# Model Accuracy
print("Accuracy:",sm.accuracy_score(y_test, y_pred))
```

```
Accuracy: 1.0
```