Perform Data cleaning, processing (Using data visualisation),classification/clustering/associa mining and performance evaluation.

Prediction using Decision Tree Algorithm on Dirty iris dataset

Importing required libraries.

In [1]:

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import seaborn as sns
```

Data cleaning and preprocessing

In [10]:

```
# Loading the dataset
iris = datasets.load_iris()
data=pd.DataFrame(iris['data'],columns=["Petal length","Petal Width","Sepal Length"
data['Species']=iris['target']
data['Species']=data['Species'].apply(lambda x: iris['target_names'][x])
data.head()
```

Out[10]:

	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 [11]:

```
# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

In [12]:

```
data.tail()
```

Out[12]:

	Petal length	Petal Width	Sepal Length	Sepal Width	Species
145	7.7	2.8	6.7	2.0	virginica
146	5.2	3.5	1.5	0.2	setosa
147	5.5	2.6	4.4	1.2	versicolor
148	6.4	3.2	4.5	1.5	versicolor
149	5.2	2.7	3.9	1.4	versicolor

In [13]:

```
data.describe()
```

Out[13]:

	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 [17]:

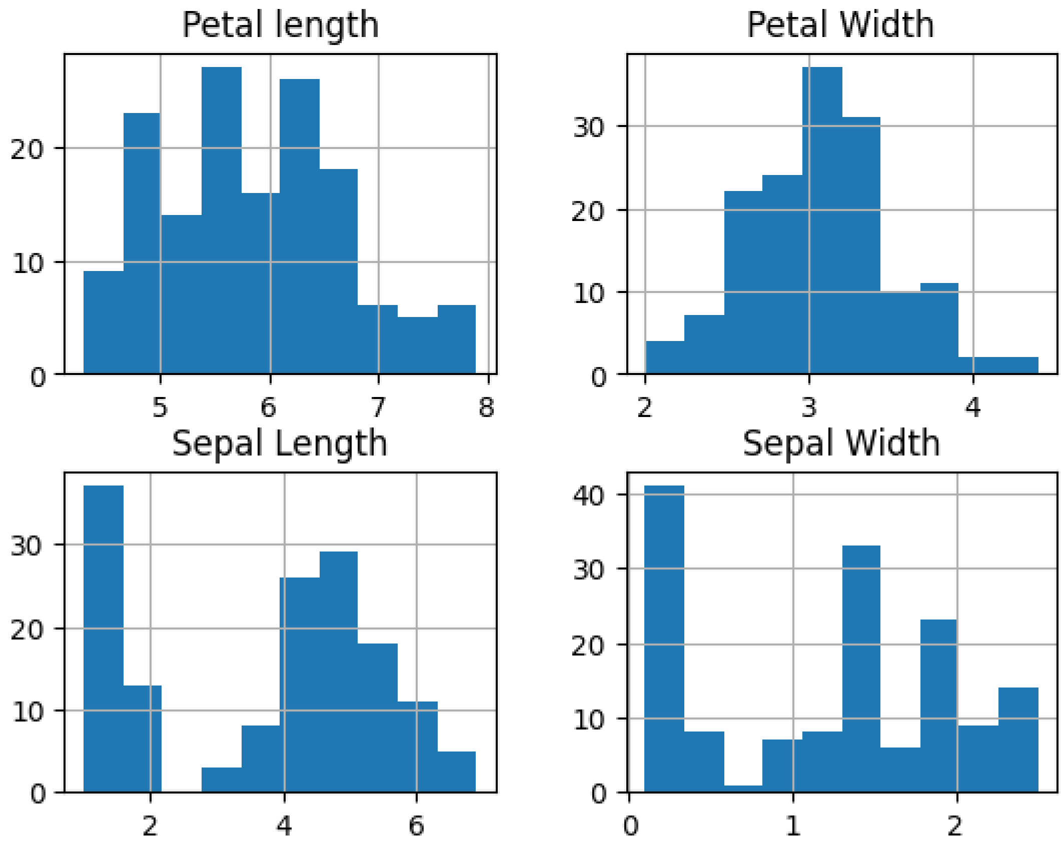
```
data.info()
```

```
<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
```

Data Visualisation

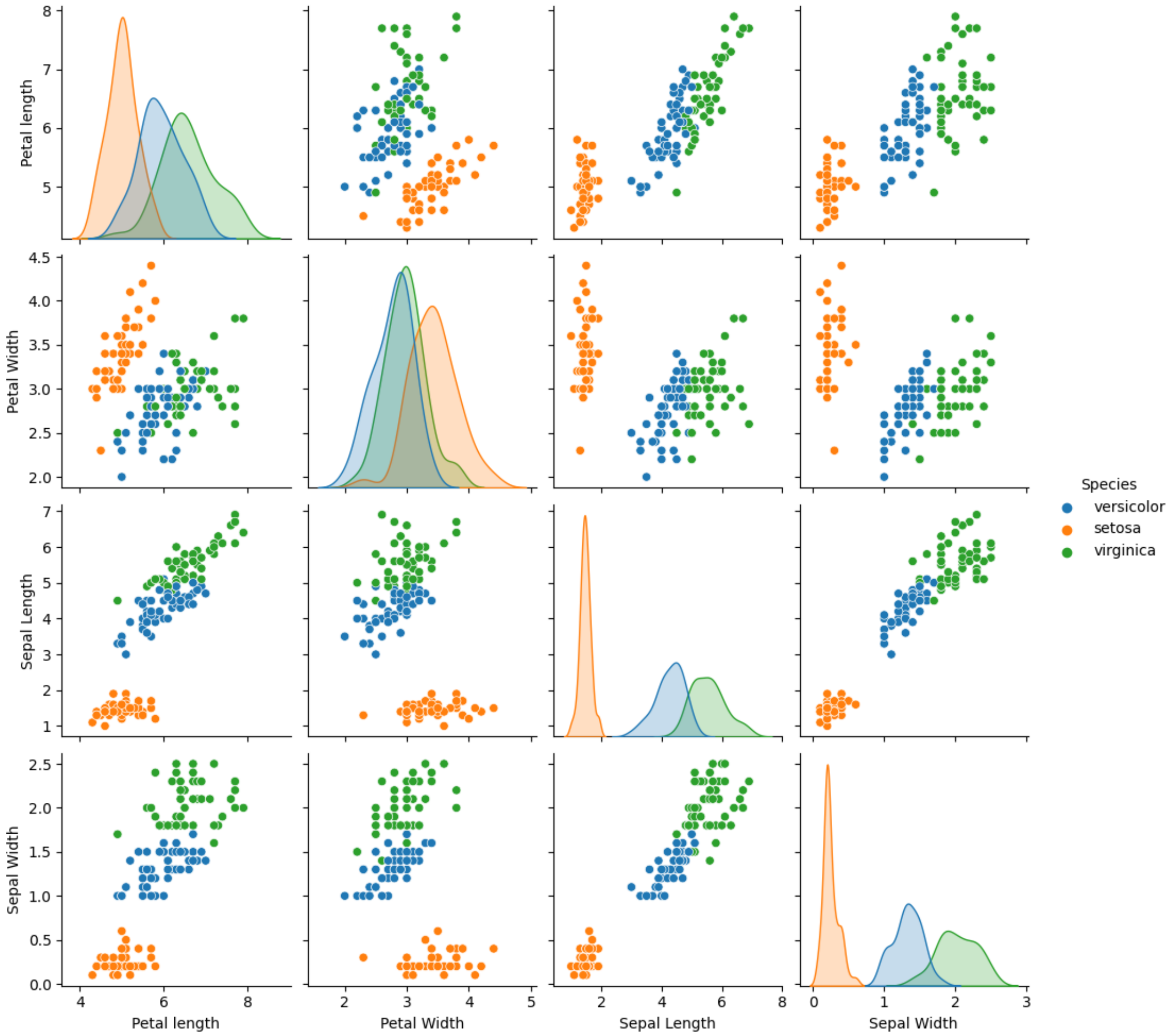
In [20]:

```
data.hist()
plt.show()
```



In [15]:

```
sns.pairplot(data, hue = 'Species')
plt.show()
```

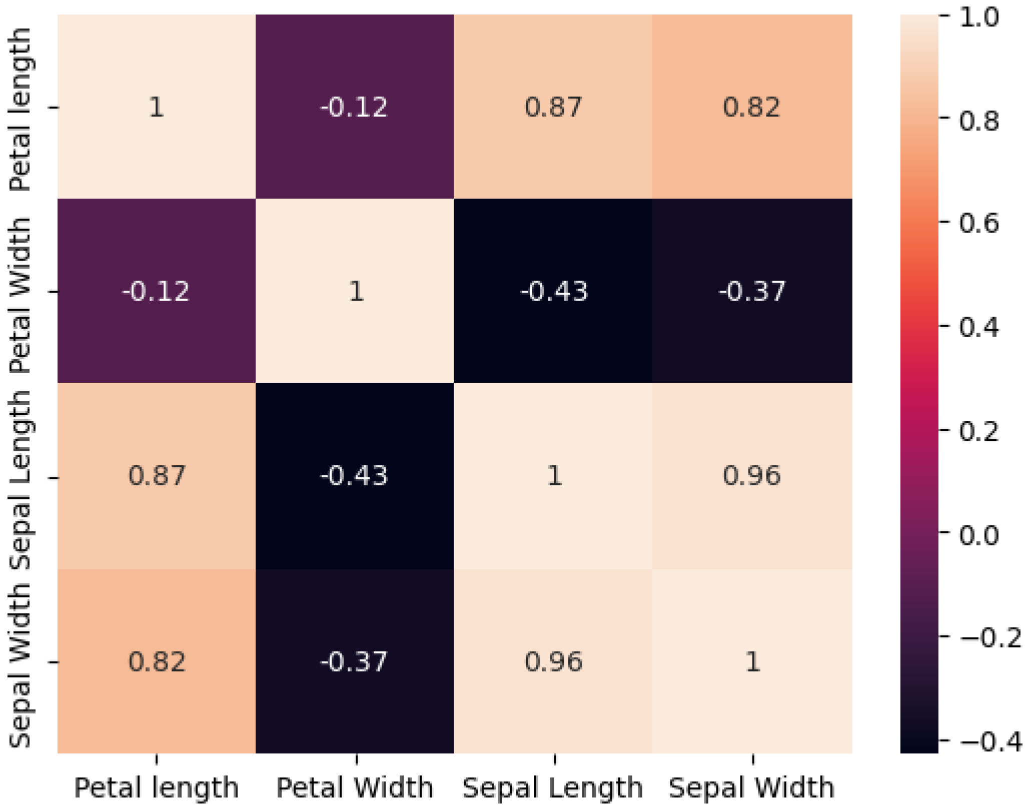


```
In [18]: print(data.corr())
         sns.heatmap(data.corr(), annot = True)
```

```
                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
```

```
<ipython-input-18-d851783e3452>:1: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Sele
ct only valid columns or specify the value of numeric_only to silence this warning.
  print(data.corr())
<ipython-input-18-d851783e3452>:2: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Sele
ct only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data.corr(), annot = True)
```

```
Out[18]: <Axes: >
```



Data Prepration

In [39]:

```
from sklearn.model_selection import train_test_split
X = data.drop('Species', axis = 1)
Y = data['Species']
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.2)
print(len(X_train),len(X_test),len(y_train),len(y_test))
```

```
120 30 120 30
```

Training the Model

In [41]:

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(random_state=12)
model = DT.fit(X_train, y_train)
```

In [42]:

```
y_pred = DT.predict(X_test)
```

Model Evaluation

In [43]:

```
DT.score(X_test, y_test)
```

Out[43]:

```
0.9
```

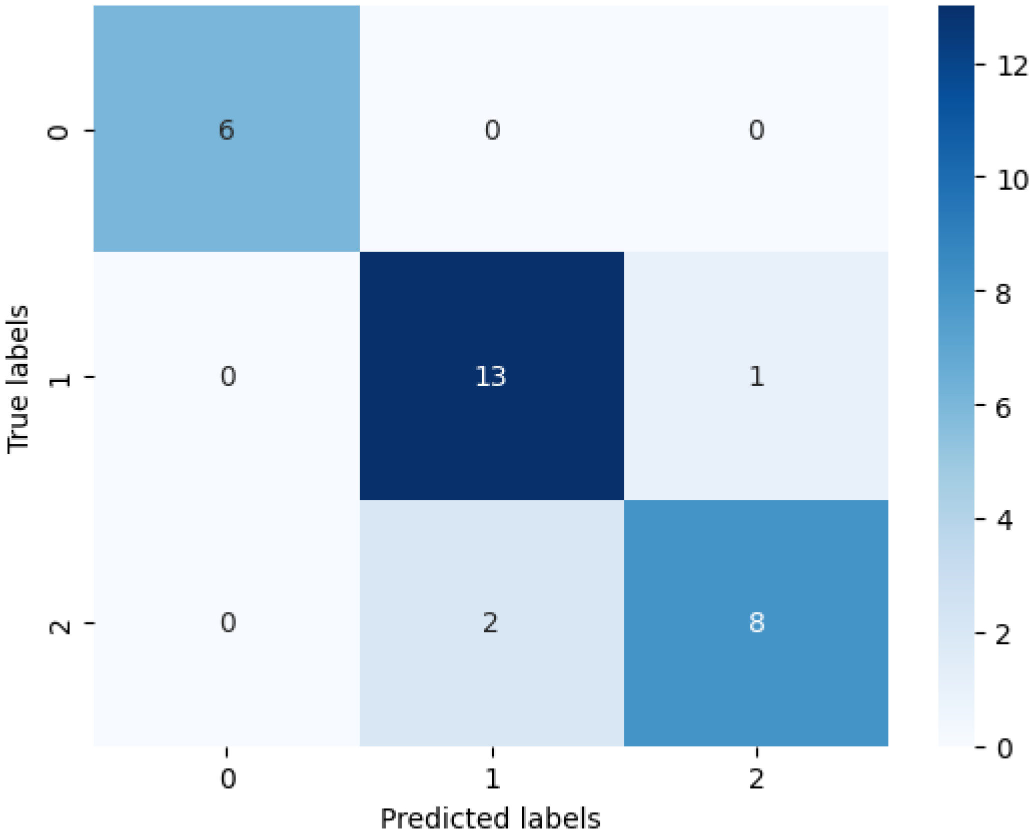
In [44]:

```
#Accuracy
from sklearn import metrics
print('Accuracy Score:', metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy Score: 0.9
```

In [45]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Blues", fmt='g')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.show()
```



Visualize the Decision Tree Classifier algorithm graph

In [30]:

```
!pip install pydotplus
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
public/simple/
Requirement already satisfied: pydotplus in /usr/local/lib/python3.10/dist-packages
(2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.10/dist-pa
ckages (from pydotplus) (3.0.9)
```

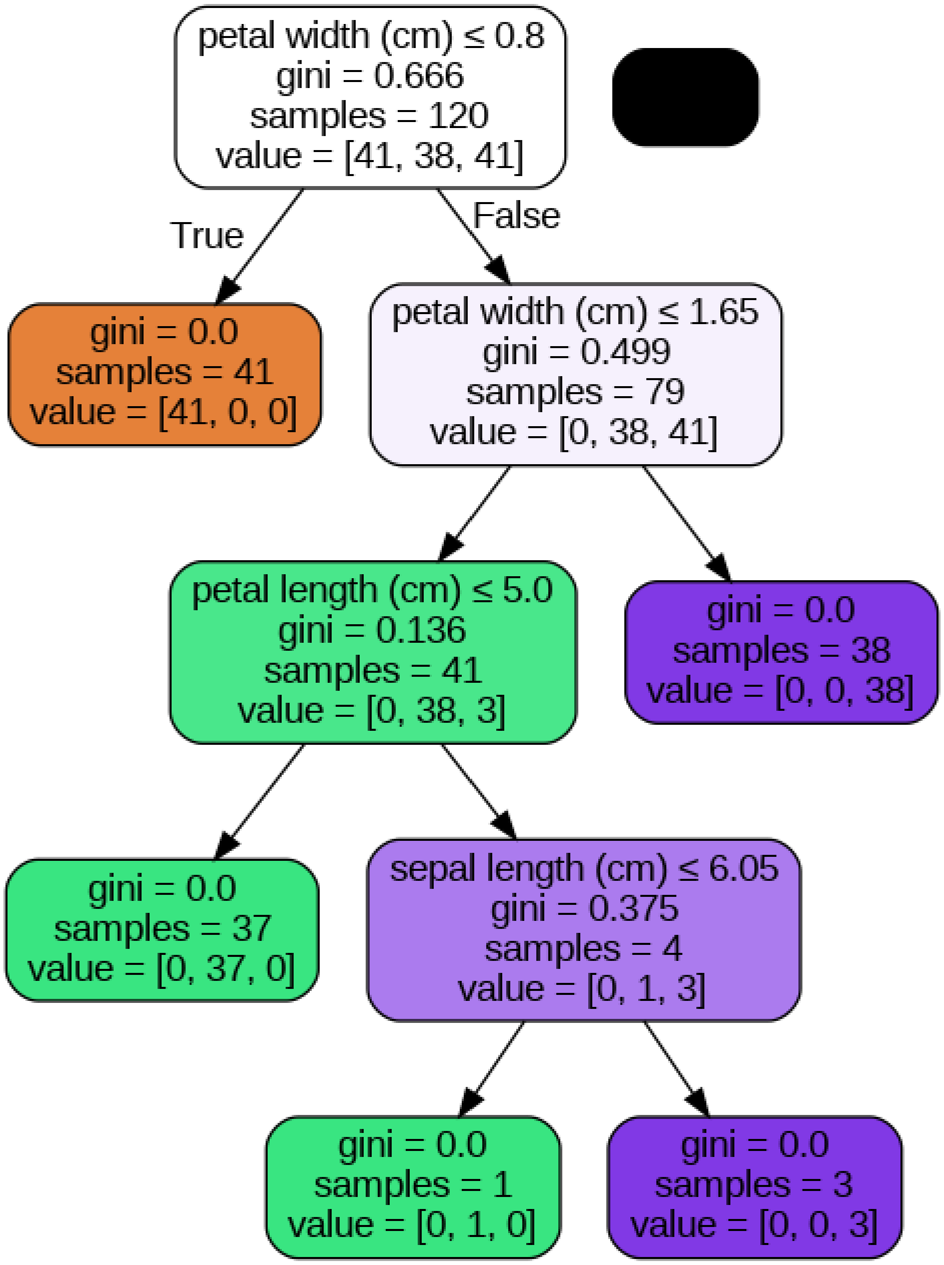
In [35]:

```
# Import necessary libraries for graph viz
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
```

```python
import pydotplus

# Visualize the graph
dot_data = StringIO()
export_graphviz(DT, out_file=dot_data, feature_names=iris.feature_names,
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[35]:



We got 90% accuracy using Decision Tree model.