Iris Flowers Classification ML Project:

Importing libraries and csv file in jupyter

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import scikitplot as skplt
import os
```

In [24]:
```
pwd
```

Out[24]: 'C:\\Users\\Azaan'

In [34]:
```
df= pd.read_csv("C:/Users/Azaan/Downloads/iris.csv")
```

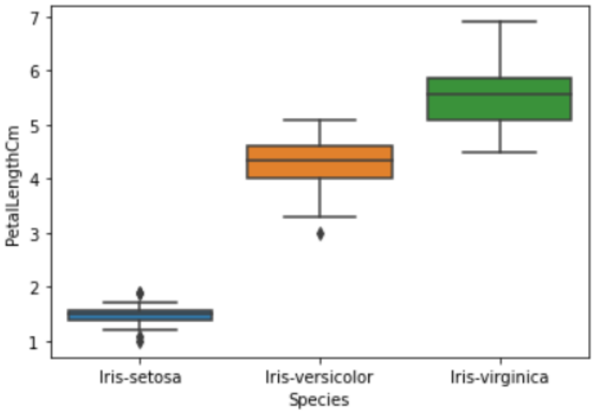
In [35]:
```
df.head()
```

Out[35]:

	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

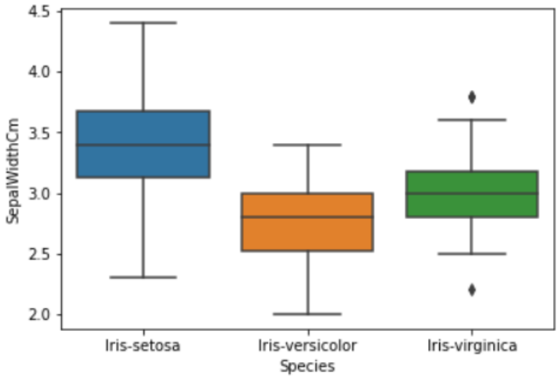
Visualization

In [48]:
```
sns.boxplot(x="Species", y='PetalLengthCm', data=df)
plt.show()
```



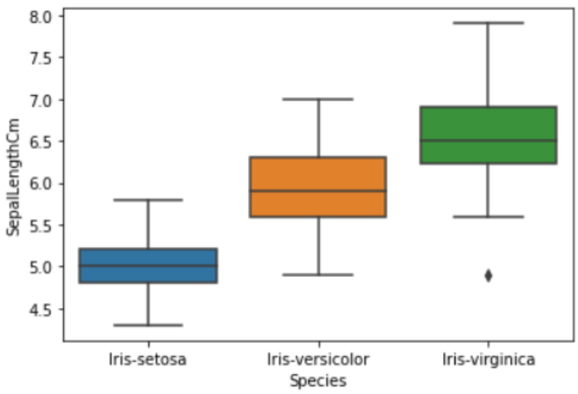
```
In [49]: sns.boxplot(x = "Species", y = "SepalWidthCm", data = df)
```

```
Out[49]: <AxesSubplot:xlabel='Species', ylabel='SepalWidthCm'>
```



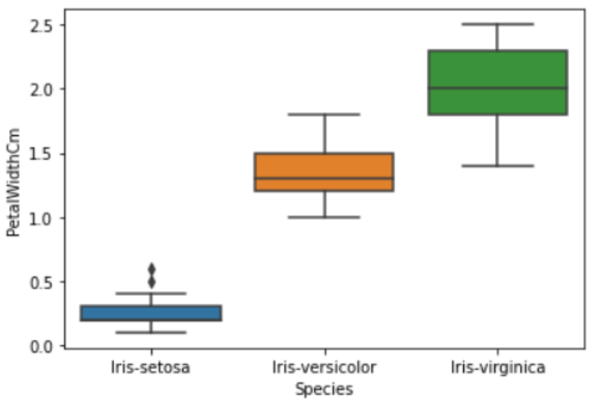

```
In [50]: sns.boxplot(x = "Species", y = "SepalLengthCm", data = df)
```

```
Out[50]: <AxesSubplot:xlabel='Species', ylabel='SepalLengthCm'>
```

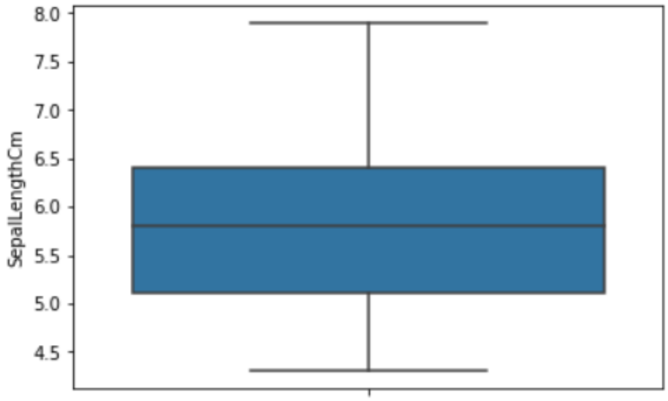



```
In [51]: sns.boxplot(x = "Species", y = "PetalWidthCm", data = df)
```

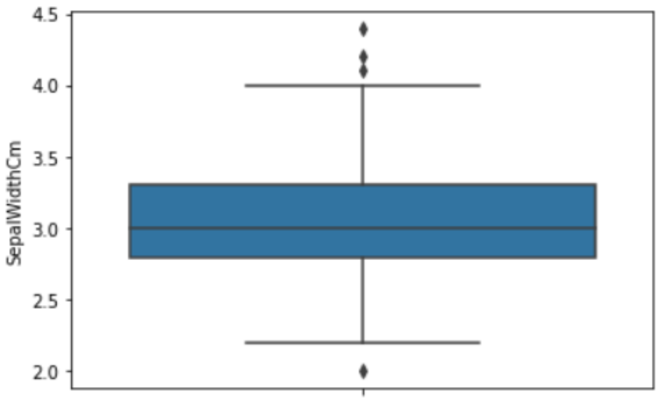
```
Out[51]: <AxesSubplot:xlabel='Species', ylabel='PetalWidthCm'>
```



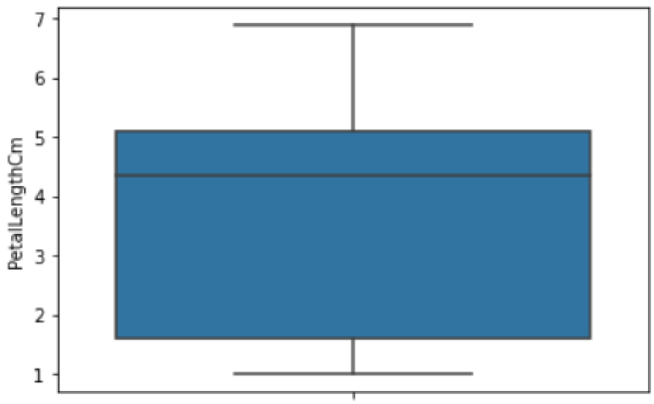
In [52]:
```
sns.boxplot( y="SepalLengthCm" , data=df);
plt.show()
```



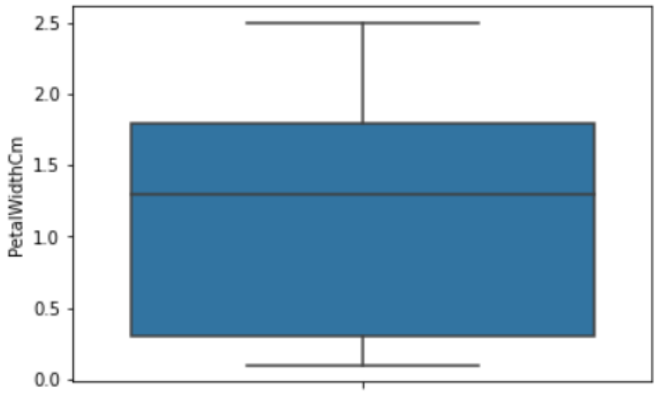
In [53]:
```
sns.boxplot( y="SepalWidthCm" , data=df);
plt.show()
```



In [54]:
```
sns.boxplot( y="PetalLengthCm" , data=df);
plt.show()
```

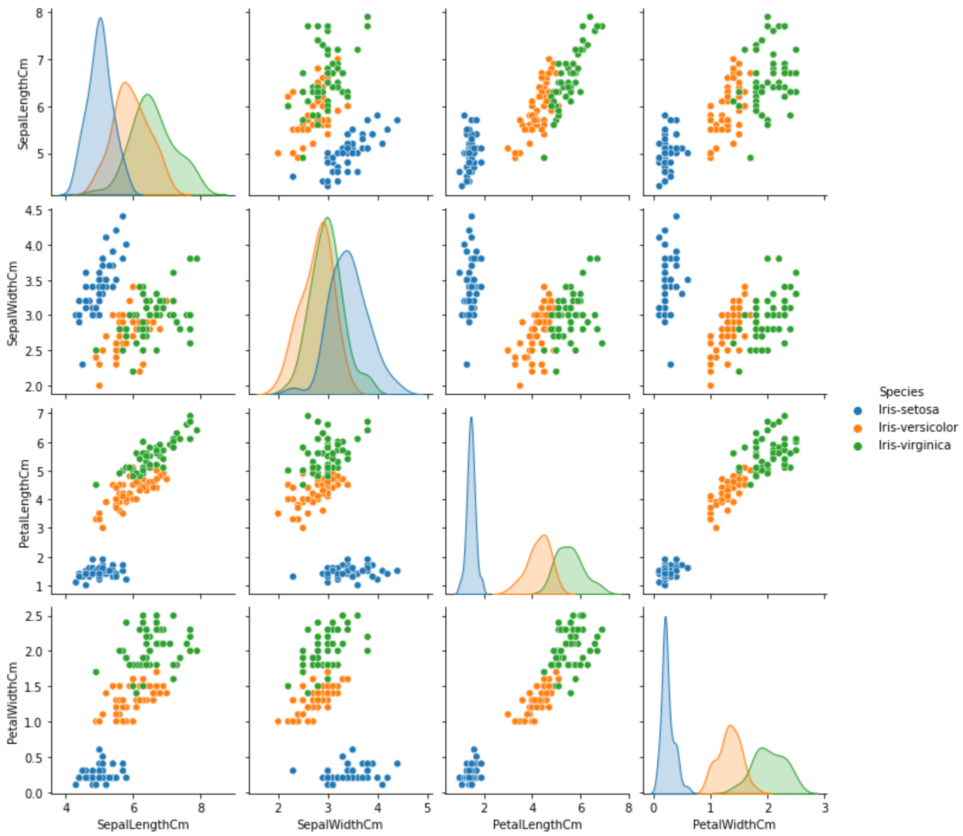


In [55]:
```
sns.boxplot( y="PetalWidthCm" , data=df);
plt.show()
```



In [56]:
```
sns.pairplot(df,hue = 'Species')
```

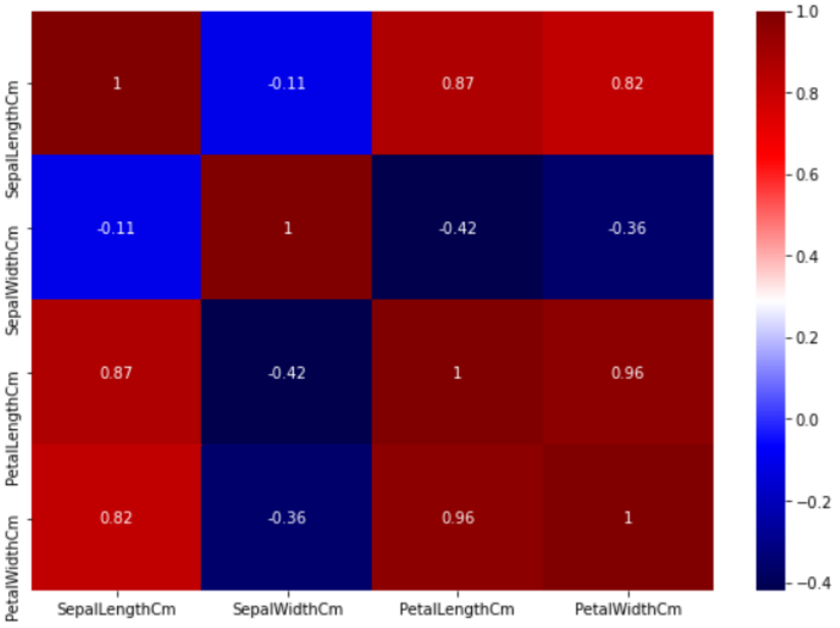
Out[56]: <seaborn.axisgrid.PairGrid at 0x1e9dc899df0>



Data Preprocessing and Correlation Matrix

 # heatmap uses to show 2D data in graphical format. Each data value represents in a matrix and it has a special colour.

In [57]:
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True,cmap="seismic")
plt.show()
```



Label Encoder

In [58]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [59]:
```
df['Species'] = le.fit_transform(df['Species'])  # fit_transform: Fit label encoder and return encoded labels.
df.head()
```

Out[59]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [60]:
```
X = df.drop(columns=['Species']) # Drop column
y = df['Species']
X[:5]
```

Out[60]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

Splitting the Dataset into the Training set and Test set

In [62]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 1)
```

Selecting the Models and Metrics (Supervised Machine Learning Models)

In [63]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [64]:
```
lr = LogisticRegression()
knn = KNeighborsClassifier()
svm = SVC()
nb = GaussianNB()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
```

Training and Evaluating the Models

In [65]:
```
models = [lr, knn, svm, nb, dt, rf]
scores = []

for model in models:
  model.fit(X_train, y_train) # LogisticRegression.fit(X_train, y_train) # Fitting Support Vector Classifer to the Training set
  y_pred = model.predict(X_test) #LogisticRegression.predict(X_test) # Predicting the Test set results
  scores.append(accuracy_score(y_test, y_pred)) # Accuracy on the Test set results
  print("Accuracy of " + type(model).__name__ + " is", accuracy_score(y_test, y_pred))
```

```
Accuracy of LogisticRegression is 0.9777777777777777
Accuracy of KNeighborsClassifier is 0.9777777777777777
Accuracy of SVC is 0.9777777777777777
Accuracy of GaussianNB is 0.9333333333333333
Accuracy of DecisionTreeClassifier is 0.9555555555555556
Accuracy of RandomForestClassifier is 0.9555555555555556
```

In [66]:
```
results = pd.DataFrame({
    'Models': ['Logistic Regression', 'K-Nearest Neighbors', 'Support Vector Machine', 'Naive Bayes', 'Decision Tree',
               'Random Forest'],'Accuracy': scores})

results = results.sort_values(by='Accuracy', ascending=False)
print(results)
```

```
                   Models  Accuracy
0     Logistic Regression  0.977778
1     K-Nearest Neighbors  0.977778
2  Support Vector Machine  0.977778
4           Decision Tree  0.955556
5           Random Forest  0.955556
3             Naive Bayes  0.933333
```

Prediction using Decision Tree Algorithm:
Importing Libraries and dataset in jupyter

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

In [2]:
```
dataset= pd.read_csv("C:/Users/Azaan/Downloads/iris.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

In [3]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

In [4]:
```
print(X_train)
```

```
[[6.20e+01 5.90e+00 3.00e+00 4.20e+00 1.50e+00]
 [9.30e+01 5.80e+00 2.60e+00 4.00e+00 1.20e+00]
 [1.13e+02 6.80e+00 3.00e+00 5.50e+00 2.10e+00]
 [3.00e+00 4.70e+00 3.20e+00 1.30e+00 2.00e-01]
 [1.42e+02 6.90e+00 3.10e+00 5.10e+00 2.30e+00]
 [4.40e+01 5.00e+00 3.50e+00 1.60e+00 6.00e-01]
 [1.10e+01 5.40e+00 3.70e+00 1.50e+00 2.00e-01]
 [6.10e+01 5.00e+00 2.00e+00 3.50e+00 1.00e+00]
 [1.17e+02 6.50e+00 3.00e+00 5.50e+00 1.80e+00]
 [1.45e+02 6.70e+00 3.30e+00 5.70e+00 2.50e+00]
 [1.20e+02 6.00e+00 2.20e+00 5.00e+00 1.50e+00]
 [1.09e+02 6.70e+00 2.50e+00 5.80e+00 1.80e+00]
 [7.00e+01 5.60e+00 2.50e+00 3.90e+00 1.10e+00]
 [1.36e+02 7.70e+00 3.00e+00 6.10e+00 2.30e+00]
 [5.70e+01 6.30e+00 3.30e+00 4.70e+00 1.60e+00]
 [8.10e+01 5.50e+00 2.40e+00 3.80e+00 1.10e+00]
 [1.24e+02 6.30e+00 2.70e+00 4.90e+00 1.80e+00]
 [1.34e+02 6.30e+00 2.80e+00 5.10e+00 1.50e+00]
 [1.07e+02 4.90e+00 2.50e+00 4.50e+00 1.70e+00]
```

In [5]:
```
print(y_train)
```

```
['Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa']
```

In [6]:
```
print(X_test)
```

```
[[1.15e+02 5.80e+00 2.80e+00 5.10e+00 2.40e+00]
 [6.30e+01 6.00e+00 2.20e+00 4.00e+00 1.00e+00]
 [3.40e+01 5.50e+00 4.20e+00 1.40e+00 2.00e-01]
 [1.08e+02 7.30e+00 2.90e+00 6.30e+00 1.80e+00]
 [8.00e+00 5.00e+00 3.40e+00 1.50e+00 2.00e-01]
 [1.01e+02 6.30e+00 3.30e+00 6.00e+00 2.50e+00]
 [4.10e+01 5.00e+00 3.50e+00 1.30e+00 3.00e-01]
 [8.70e+01 6.70e+00 3.10e+00 4.70e+00 1.50e+00]
 [7.70e+01 6.80e+00 2.80e+00 4.80e+00 1.40e+00]
 [7.20e+01 6.10e+00 2.80e+00 4.00e+00 1.30e+00]
 [1.35e+02 6.10e+00 2.60e+00 5.60e+00 1.40e+00]
 [5.20e+01 6.40e+00 3.20e+00 4.50e+00 1.50e+00]
 [7.40e+01 6.10e+00 2.80e+00 4.70e+00 1.20e+00]
 [5.50e+01 6.50e+00 2.80e+00 4.60e+00 1.50e+00]
 [6.40e+01 6.10e+00 2.90e+00 4.70e+00 1.40e+00]
 [3.80e+01 4.90e+00 3.10e+00 1.50e+00 1.00e-01]
 [7.90e+01 6.00e+00 2.90e+00 4.50e+00 1.50e+00]
 [9.10e+01 5.50e+00 2.60e+00 4.40e+00 1.20e+00]
 [4.60e+01 4.80e+00 3.00e+00 1.40e+00 3.00e-01]
 [1.70e+01 5.40e+00 3.90e+00 1.30e+00 4.00e-01]
 [1.22e+02 5.60e+00 2.80e+00 4.90e+00 2.00e+00]
 [6.70e+01 5.60e+00 3.00e+00 4.50e+00 1.50e+00]
 [2.50e+01 4.80e+00 3.40e+00 1.90e+00 2.00e-01]
 [9.00e+00 4.40e+00 2.90e+00 1.40e+00 2.00e-01]
 [1.27e+02 6.20e+00 2.80e+00 4.80e+00 1.80e+00]
 [2.30e+01 4.60e+00 3.60e+00 1.00e+00 2.00e-01]
 [4.50e+01 5.10e+00 3.80e+00 1.90e+00 4.00e-01]
 [9.80e+01 6.20e+00 2.90e+00 4.30e+00 1.30e+00]
```

In [7]:
```
print(y_test)
```

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor']
```

Feature Scaling

In [8]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [9]:
```
print(X_train)
```

```
[[-3.41283725e-01  1.54399532e-02 -1.19254753e-01  2.25126850e-01
   3.56381749e-01]
 [ 3.50852428e-01 -9.98450310e-02 -1.04039491e+00  1.13559562e-01
  -2.86480506e-02]
 [ 7.97391882e-01  1.05300481e+00 -1.19254753e-01  9.50314227e-01
   1.12644135e+00]
 [-1.65857511e+00 -1.36797986e+00  3.41315328e-01 -1.39259884e+00
  -1.31208072e+00]
 [ 1.44487409e+00  1.16828980e+00  1.11030287e-01  7.27179649e-01
   1.38312788e+00]
 [-7.43169234e-01 -1.02212490e+00  1.03217045e+00 -1.22524790e+00
  -7.98707650e-01]
 [-1.47995933e+00 -5.60984968e-01  1.49274053e+00 -1.28103155e+00
  -1.31208072e+00]
 [-3.63610698e-01 -1.02212490e+00 -2.42210516e+00 -1.65358660e-01
  -2.85334584e-01]
 [ 8.86699773e-01  7.07149859e-01 -1.19254753e-01  9.50314227e-01
   7.41411549e-01]
 [ 1.51185501e+00  9.37719827e-01  5.71600368e-01  1.06188152e+00
```

In [10]:
```
print(X_test)
```

```
[[ 0.84204583 -0.09984503 -0.57982483  0.72717965  1.51147115]
 [-0.31895675  0.13072494 -1.96153508  0.11355956 -0.28533458]
 [-0.96643896 -0.44569998  2.64416573 -1.33681519 -1.31208072]
 [ 0.68575702  1.62942973 -0.34953979  1.39658338  0.74141155]
 [-1.54694025 -1.0221249   0.80188541 -1.28103155 -1.31208072]
 [ 0.52946821  0.47657989  0.57160037  1.22923245  1.63981441]
 [-0.81015015 -1.0221249   1.03217045 -1.39259884 -1.18373745]
 [ 0.21689059  0.93771983  0.11103029  0.50404507  0.35638175]
 [-0.00637914  1.05300481 -0.57982483  0.55982872  0.22803848]
 [-0.118014    0.24600992 -0.57982483  0.11355956  0.09969522]
 [ 1.28858528  0.24600992 -1.04039491  1.00609787  0.22803848]
 [-0.56455345  0.59186487  0.34131533  0.39247778  0.35638175]
 [-0.07336005  0.24600992 -0.57982483  0.50404507 -0.02864805]
 [-0.49757253  0.70714986 -0.57982483  0.44826143  0.35638175]
 [-0.29662978  0.24600992 -0.34953979  0.50404507  0.22803848]
 [-0.87713107 -1.13740989  0.11103029 -1.28103155 -1.44042398]
 [ 0.03827481  0.13072494 -0.34953979  0.39247778  0.35638175]
 [ 0.30619848 -0.44569998 -1.04039491  0.33669414 -0.02864805]
 [-0.69851529 -1.25269487 -0.11925475 -1.33681519 -1.18373745]
 [-1.3459975  -0.56098497  1.95331061 -1.39259884 -1.05539418]
 [ 0.99833464 -0.330415   -0.57982483  0.61561236  0.99809808]
 [-0.22964886 -0.330415   -0.11925475  0.39247778  0.35638175]
 [-1.16738172 -1.25269487  0.80188541 -1.05789697 -1.31208072]
 [-1.52461328 -1.71383481 -0.34953979 -1.33681519 -1.31208072]
 [ 1.1099695   0.36129491 -0.57982483  0.55982872  0.74141155]
 [-1.21203566 -1.48326484  1.26245549 -1.55994977 -1.31208072]
```

Training the Decision Tree Classification model on the Training set

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

Predicting the Test set results

In [12]:
```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
 ['Iris-versicolor' 'Iris-versicolor']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-setosa' 'Iris-setosa']
 ['Iris-virginica' 'Iris-virginica']
```

Making the Confusion Matrix

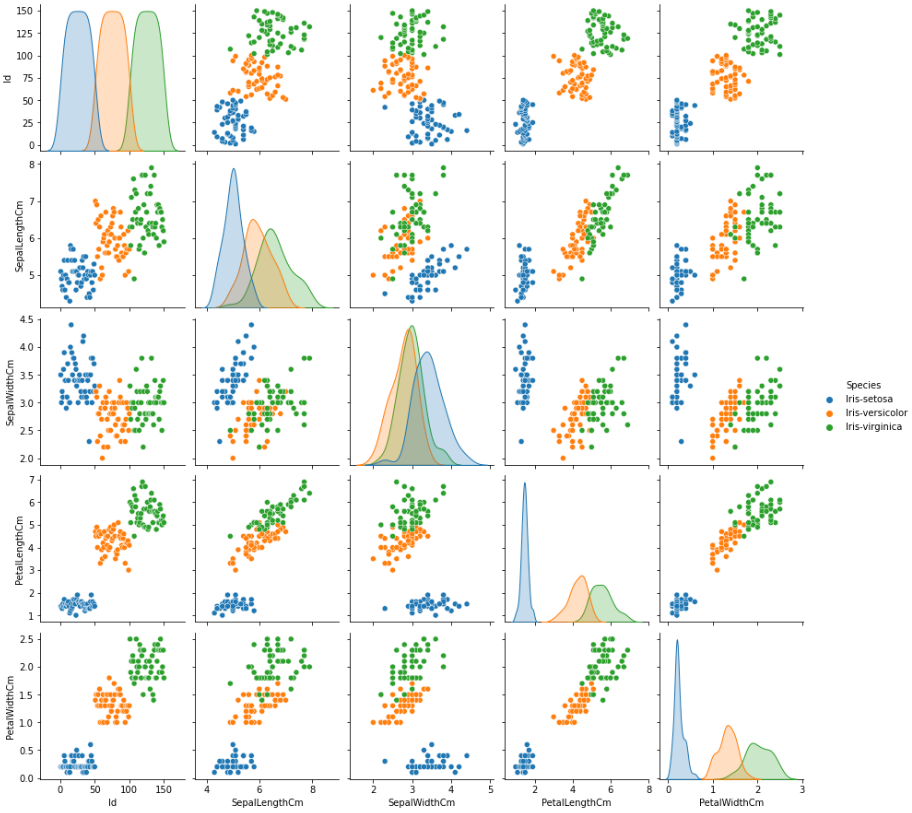
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]
1.0
```

Accuracy is 100%

Visualising things

```
In [15]: import seaborn as sns
         sns.pairplot(data=dataset, hue='Species')
         plt.show()
```



In [16]:
```
col = dataset.columns[:-1]
classes = dataset['Species'].unique().tolist()
from sklearn.tree import plot_tree
plt.figure(figsize=(16,10))
plot_tree(classifier, feature_names=col, class_names=classes,  filled=True)
```

Out[16]:
```
[Text(535.6800000000001, 453.0, 'Id <= 0.529\nentropy = 1.581\nsamples = 112\nvalue = [37, 34, 41]\nclass = Iris-virginica'),
 Text(357.12, 271.8, 'PetalWidthCm <= -0.542\nentropy = 0.999\nsamples = 71\nvalue = [37, 34, 0]\nclass = Iris-setosa'),
 Text(178.56, 90.59999999999997, 'entropy = 0.0\nsamples = 37\nvalue = [37, 0, 0]\nclass = Iris-setosa'),
 Text(535.6800000000001, 90.59999999999997, 'entropy = 0.0\nsamples = 34\nvalue = [0, 34, 0]\nclass = Iris-versicolor'),
 Text(714.24, 271.8, 'entropy = 0.0\nsamples = 41\nvalue = [0, 0, 41]\nclass = Iris-virginica')]
```