Prediction using Decision Tree Algorithm (Level - Intermediate) The Spark Foundation

Semih Bugra Sezer

In [173]:

```
# Importing libraries in Python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import pandas as pd

# Loading the iris dataset
iris = pd.read_csv('/content/Iris.csv')

# Attribute values
A = iris.iloc[:, :-1]

# Target values
b = iris.iloc[:, -1]

# Label Encoder
le = preprocessing.LabelEncoder()
b = le.fit_transform(b)

print(A.head())
print(b)
```

```
   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
[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]
```

In [176]:

```
A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 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
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

In [134]:

```
A.describe()
```

Out[134]:

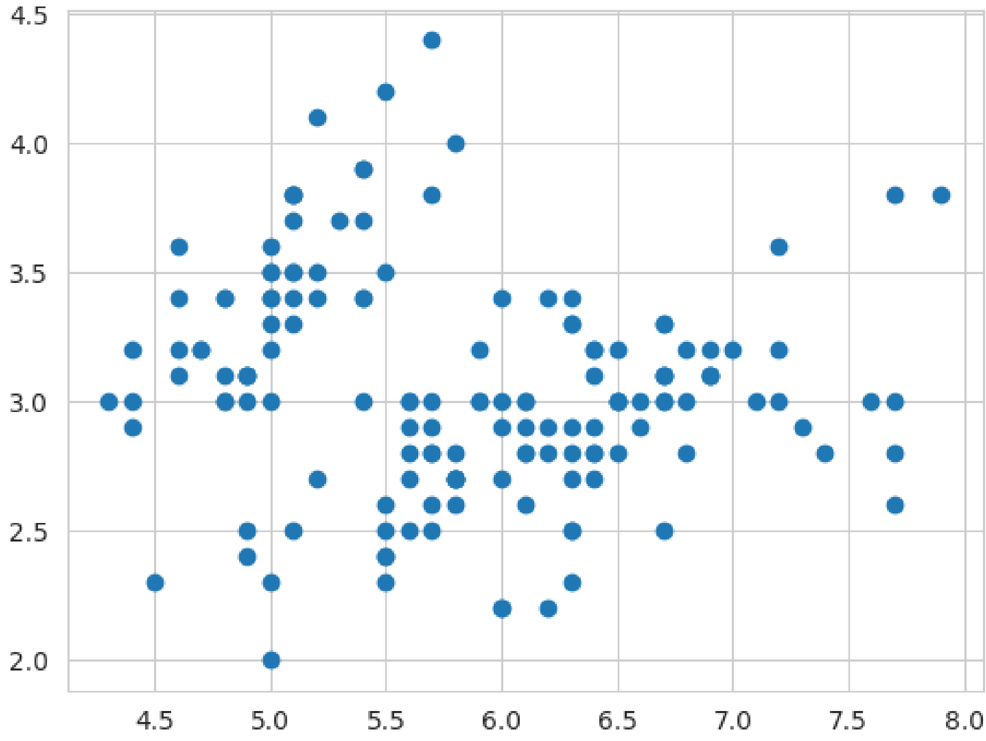
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm

	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

Visualizing Iris Data

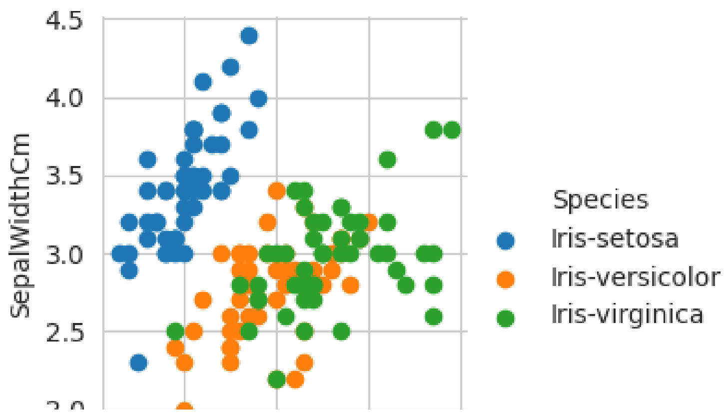
In [135]:

```
plt.scatter(iris['SepalLengthCm'],iris['SepalWidthCm'])
plt.show()
```



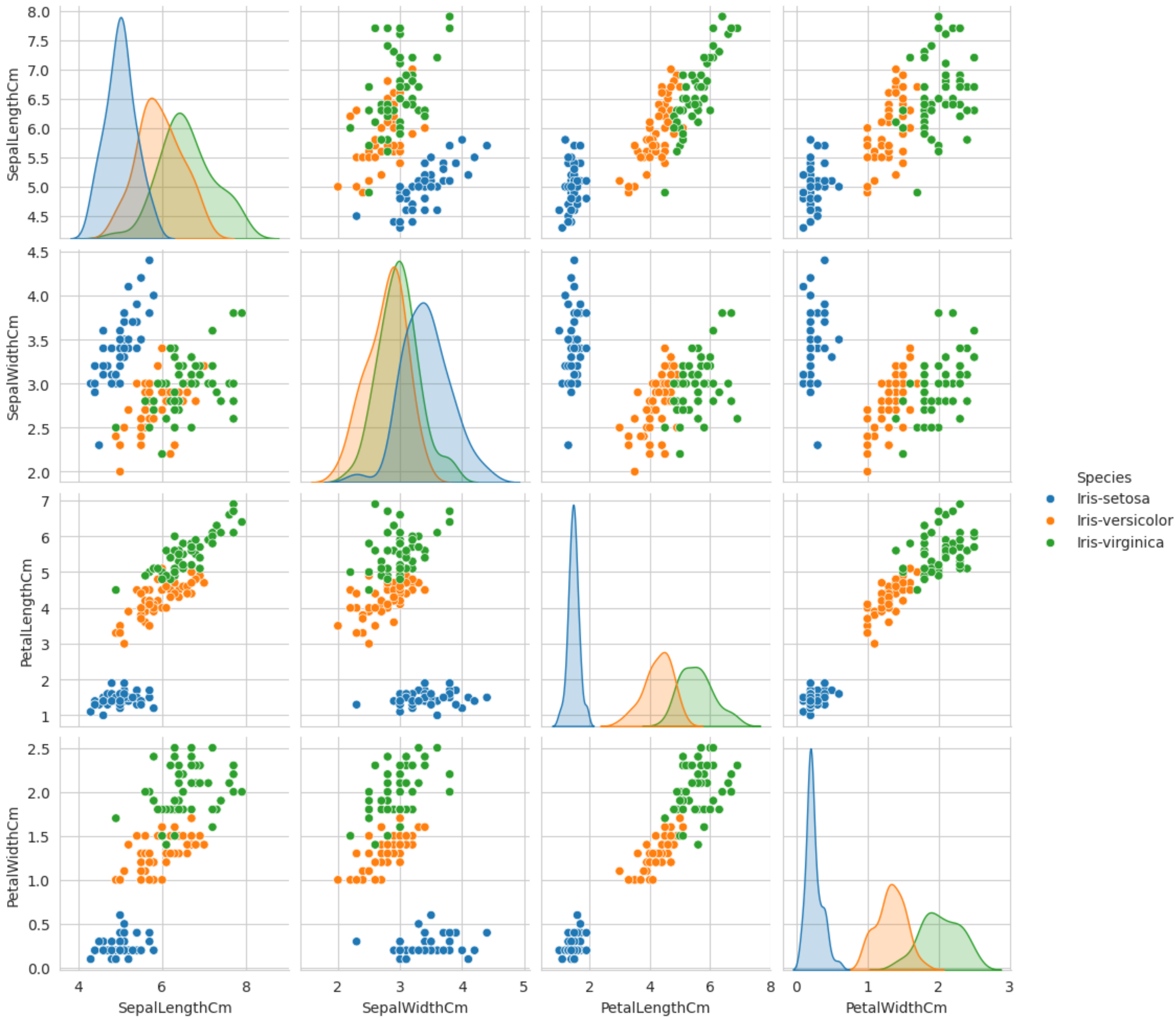
In [136]:

```
#Using Seaborn lib to visualized 2 features based on target variable.

sns.set_style('whitegrid')
sns.FacetGrid(iris, hue = 'Species') \
   .map(plt.scatter, 'SepalLengthCm','SepalWidthCm') \
   .add_legend()

plt.show()
```



In [137]:

```
#Pair plot gives the relationship b/w all features distribution with each other..

sns.pairplot(iris.drop(['Id'],axis=1), hue='Species')
plt.show()
```



Exploring Some New Features

In [138]:

```
#Just trying to explore some new feature using the given data...

iris['Sepal_diff'] = iris['SepalLengthCm']-iris['SepalWidthCm']
iris['petal_diff'] = iris['PetalLengthCm']-iris['PetalWidthCm']
iris
```

Out[138]:

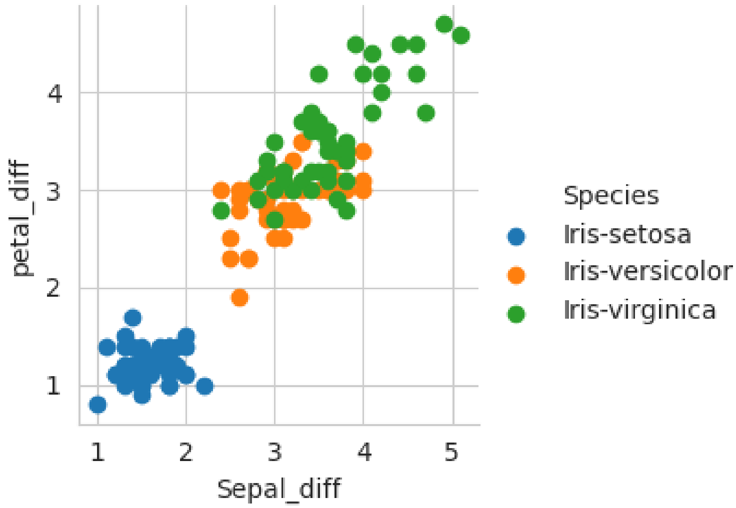
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal_diff	petal_diff
0	1	5.1	3.5	1.4	0.2	Iris-setosa	1.6	1.2
1	2	4.9	3.0	1.4	0.2	Iris-setosa	1.9	1.2
2	3	4.7	3.2	1.3	0.2	Iris-setosa	1.5	1.1

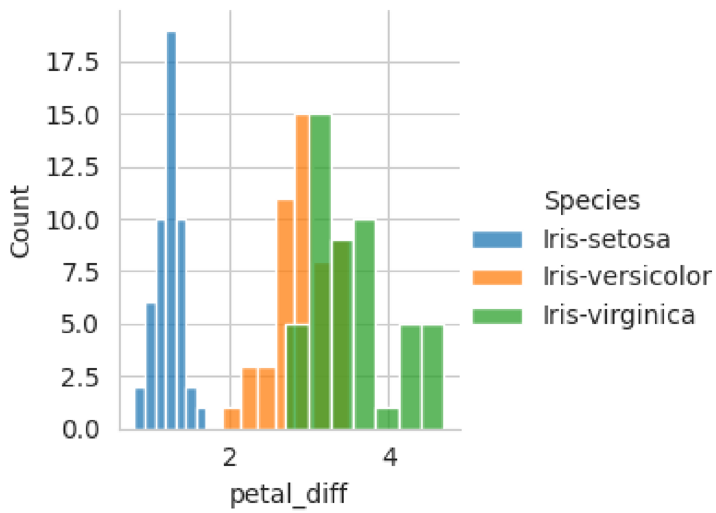
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal_diff	petal_diff
3	4	4.6	3.1	1.5	0.2	Iris-setosa	1.5	1.3
4	5	5.0	3.6	1.4	0.2	Iris-setosa	1.4	1.2
...	...	...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica	3.7	2.9
146	147	6.3	2.5	5.0	1.9	Iris-virginica	3.8	3.1
147	148	6.5	3.0	5.2	2.0	Iris-virginica	3.5	3.2
148	149	6.2	3.4	5.4	2.3	Iris-virginica	2.8	3.1
149	150	5.9	3.0	5.1	1.8	Iris-virginica	2.9	3.3

150 rows × 8 columns

In [139]:

```
#Analysed new feature to get some more infomation apart form existing ones...

sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(plt.scatter, 'Sepal_diff', 'petal_diff') \
    .add_legend()
plt.show()


sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(sns.histplot, 'petal_diff') \
    .add_legend()
plt.show()
```





In [140]:

```
iris['Sepal_petal_len_diff'] = iris['SepalLengthCm']-iris['PetalLengthCm']
iris['Sepal_petal_width_diff'] = iris['SepalWidthCm']-iris['PetalWidthCm']
iris
```

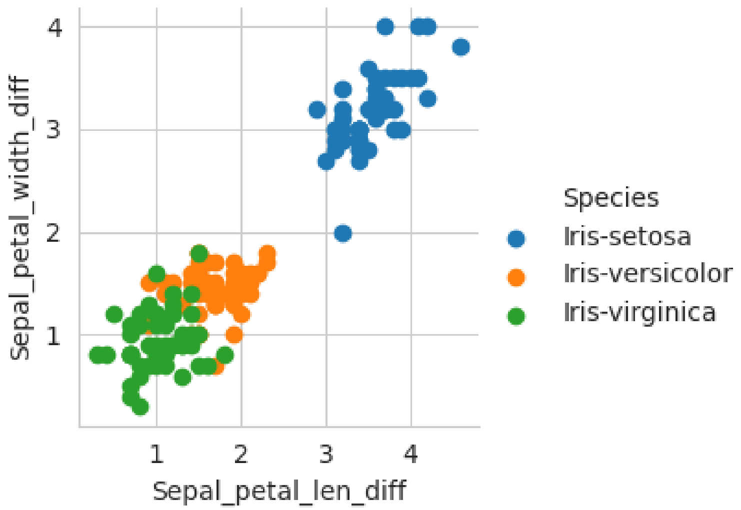
Out[140]:

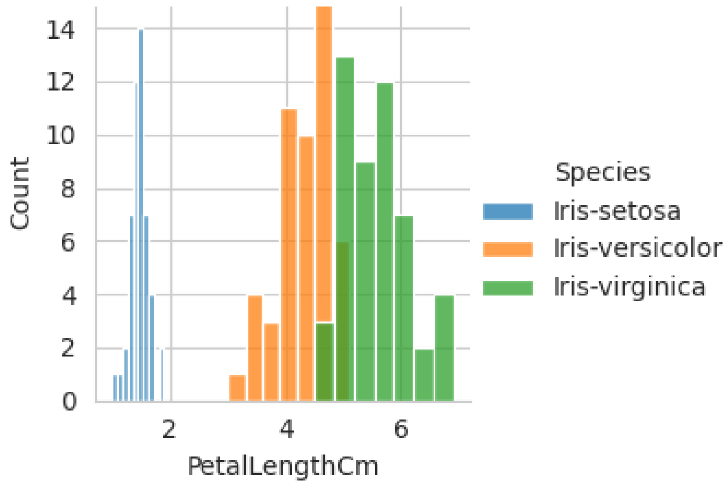
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal_diff	petal_diff	Sepal_petal_len_diff
0	1	5.1	3.5	1.4	0.2	Iris-setosa	1.6	1.2	3.7
1	2	4.9	3.0	1.4	0.2	Iris-setosa	1.9	1.2	3.5
2	3	4.7	3.2	1.3	0.2	Iris-setosa	1.5	1.1	3.4
3	4	4.6	3.1	1.5	0.2	Iris-setosa	1.5	1.3	3.1
4	5	5.0	3.6	1.4	0.2	Iris-setosa	1.4	1.2	3.6
...	...	...	...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica	3.7	2.9	1.5
146	147	6.3	2.5	5.0	1.9	Iris-virginica	3.8	3.1	1.3
147	148	6.5	3.0	5.2	2.0	Iris-virginica	3.5	3.2	1.3
148	149	6.2	3.4	5.4	2.3	Iris-virginica	2.8	3.1	0.8
149	150	5.9	3.0	5.1	1.8	Iris-virginica	2.9	3.3	0.8

150 rows × 10 columns

In [141]:

```
sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(plt.scatter, 'Sepal_petal_len_diff', 'Sepal_petal_width_diff') \
    .add_legend()
plt.show()

sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(sns.histplot, 'PetalLengthCm') \
    .add_legend()
plt.show()
```





In [142]:

```
iris['Sepal_petal_len_wid_diff'] = iris['SepalLengthCm']-iris['PetalWidthCm']
iris['Sepal_petal_wid_len_diff'] = iris['SepalWidthCm']-iris['PetalLengthCm']
iris
```

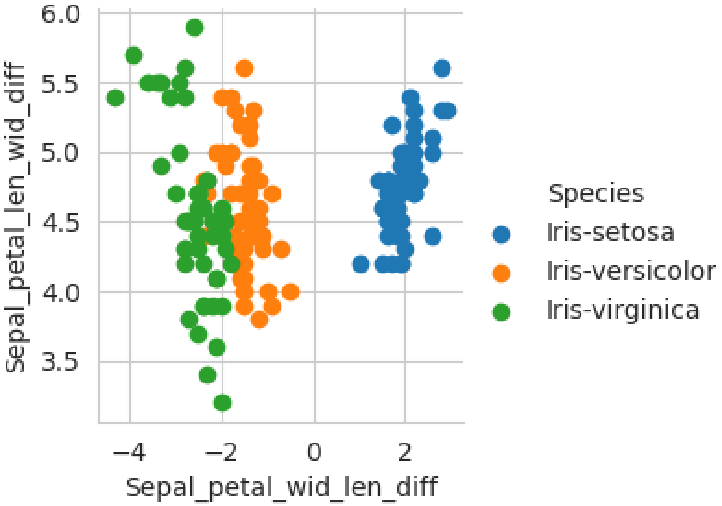
Out[142]:

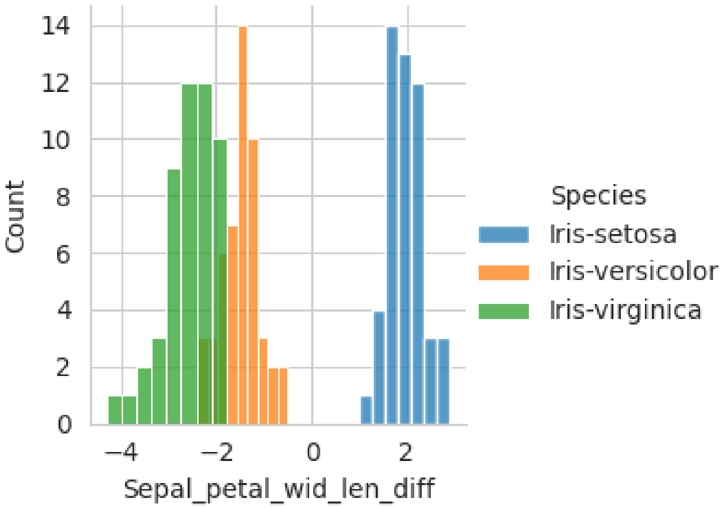
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal_diff	petal_diff	Sepal_petal_len_diff
0	1	5.1	3.5	1.4	0.2	Iris-setosa	1.6	1.2	3.7
1	2	4.9	3.0	1.4	0.2	Iris-setosa	1.9	1.2	3.5
2	3	4.7	3.2	1.3	0.2	Iris-setosa	1.5	1.1	3.4
3	4	4.6	3.1	1.5	0.2	Iris-setosa	1.5	1.3	3.1
4	5	5.0	3.6	1.4	0.2	Iris-setosa	1.4	1.2	3.6
...	...	...	...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica	3.7	2.9	1.5
146	147	6.3	2.5	5.0	1.9	Iris-virginica	3.8	3.1	1.3
147	148	6.5	3.0	5.2	2.0	Iris-virginica	3.5	3.2	1.3
148	149	6.2	3.4	5.4	2.3	Iris-virginica	2.8	3.1	0.8
149	150	5.9	3.0	5.1	1.8	Iris-virginica	2.9	3.3	0.8

150 rows × 12 columns

In [143]:

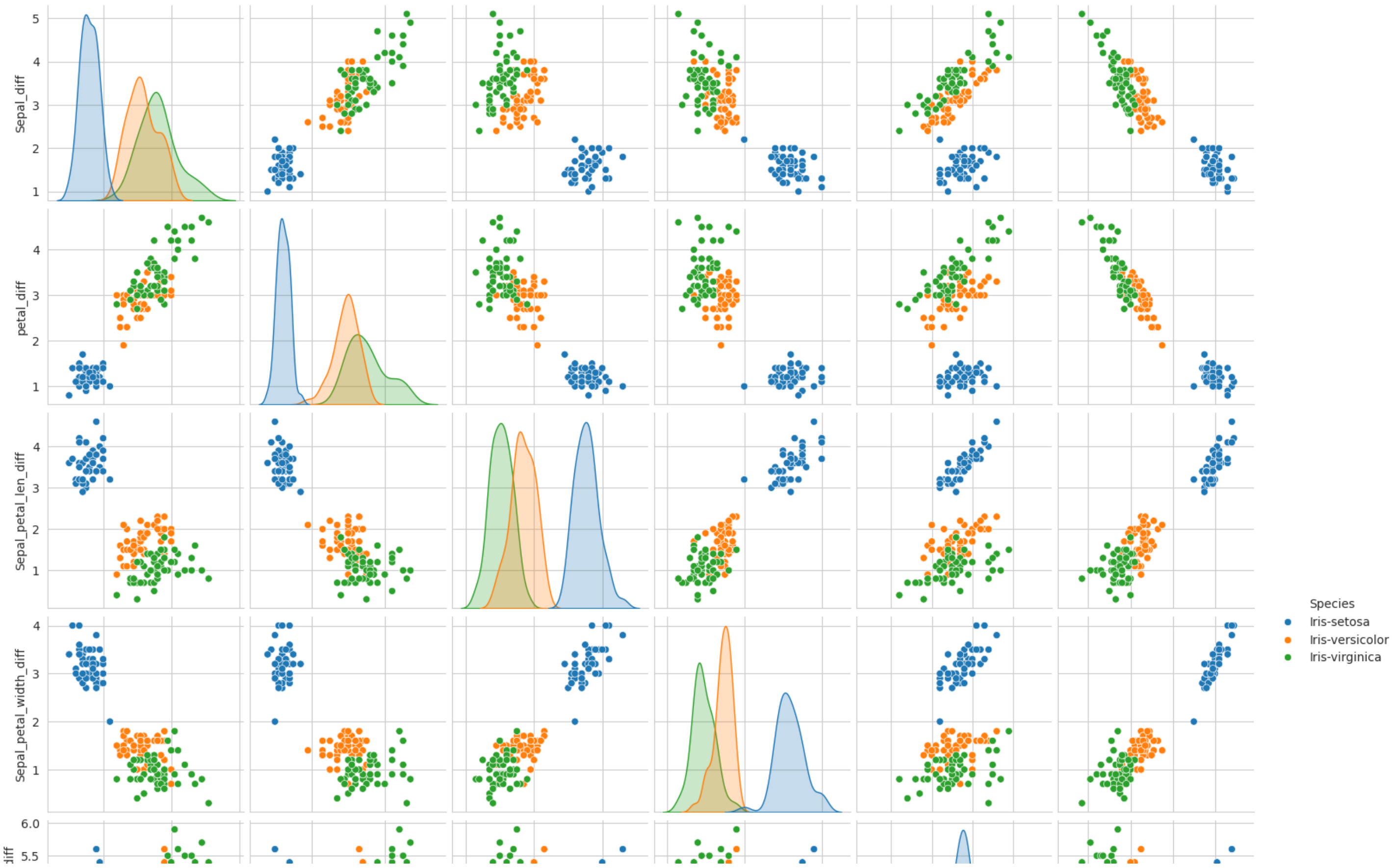
```
sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(plt.scatter, 'Sepal_petal_wid_len_diff', 'Sepal_petal_len_wid_diff') \
    .add_legend()
plt.show()

sns.set_style('whitegrid')
sns.FacetGrid(iris, hue='Species') \
    .map(sns.histplot, 'Sepal_petal_wid_len_diff') \
    .add_legend()
plt.show()
```

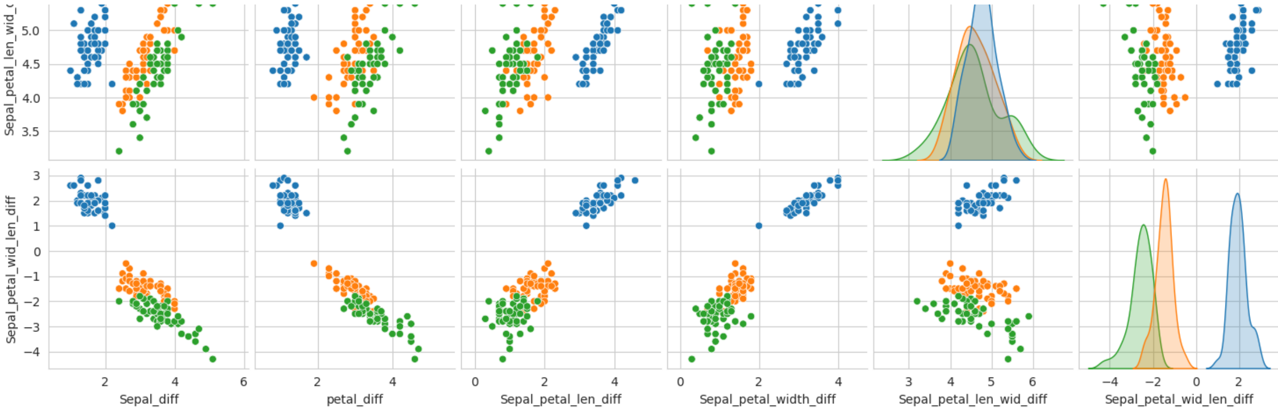




In [144]:

```
sns.pairplot(iris[['Species', 'Sepal_diff', 'petal_diff', 'Sepal_petal_len_diff',\
        'Sepal_petal_width_diff', 'Sepal_petal_len_wid_diff',\
        'Sepal_petal_wid_len_diff']], hue='Species')
plt.show()
```



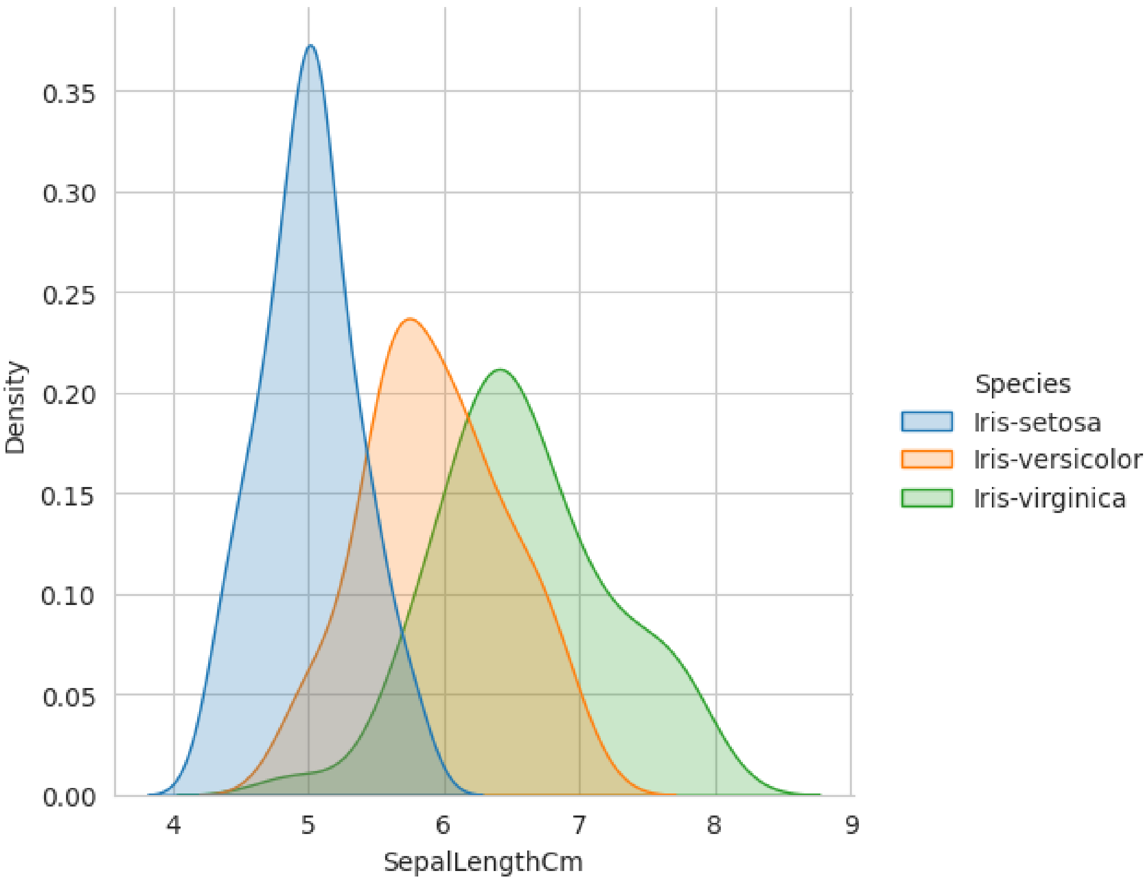


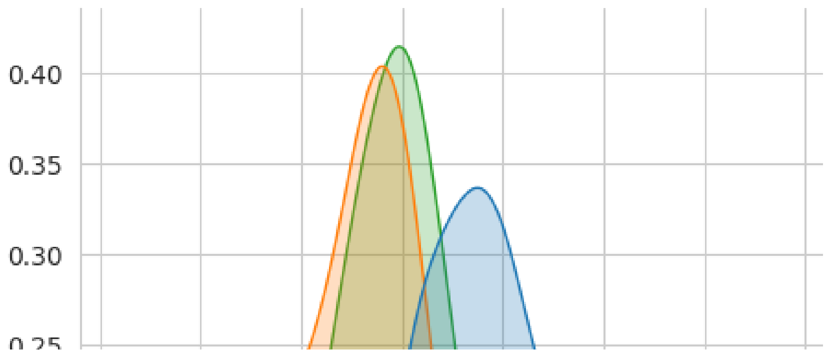
In [145]:

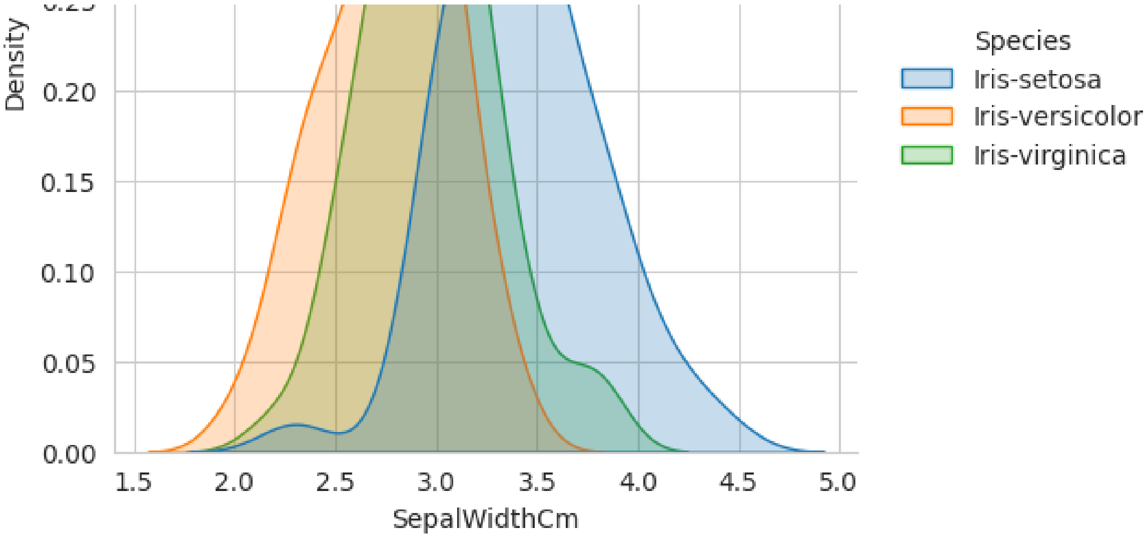
```
#Droping Id column as it is of no use in classifing the class labels..

iris.drop(['Id'],axis=1,inplace=True)
```

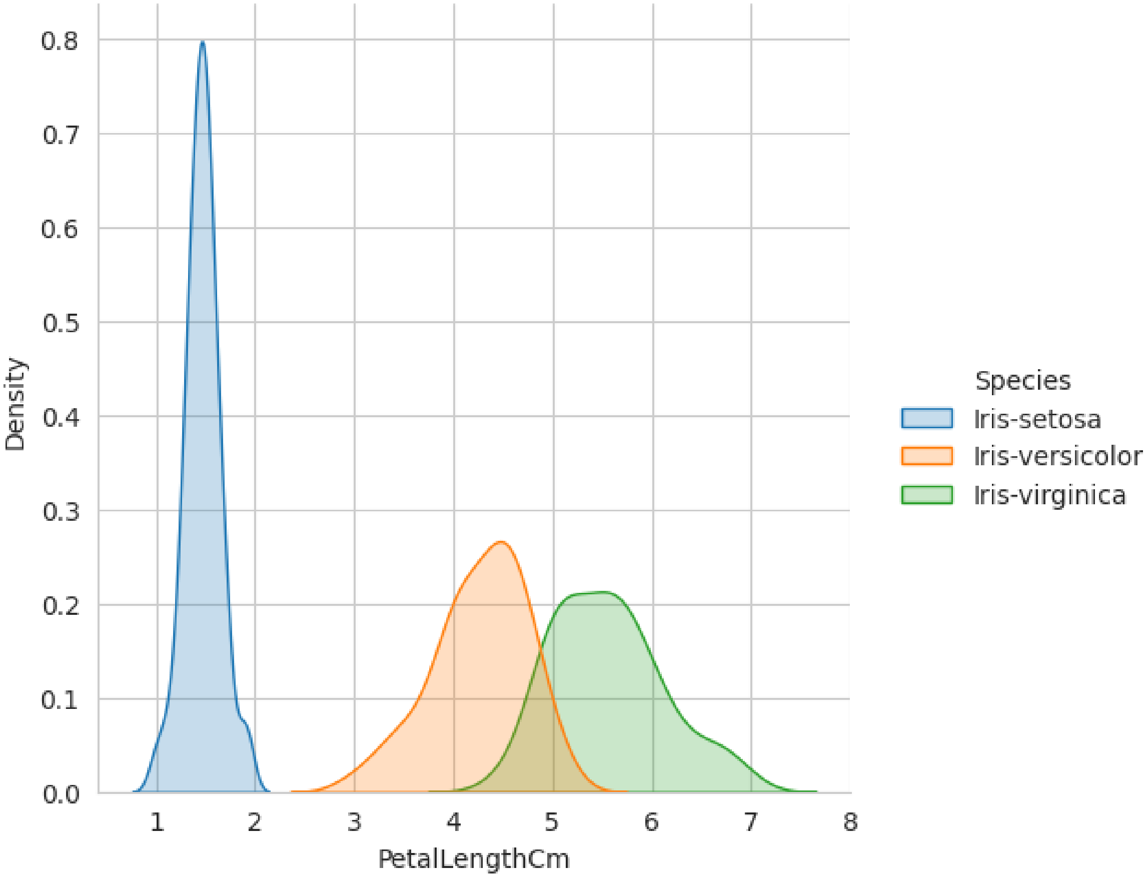
In [146]:

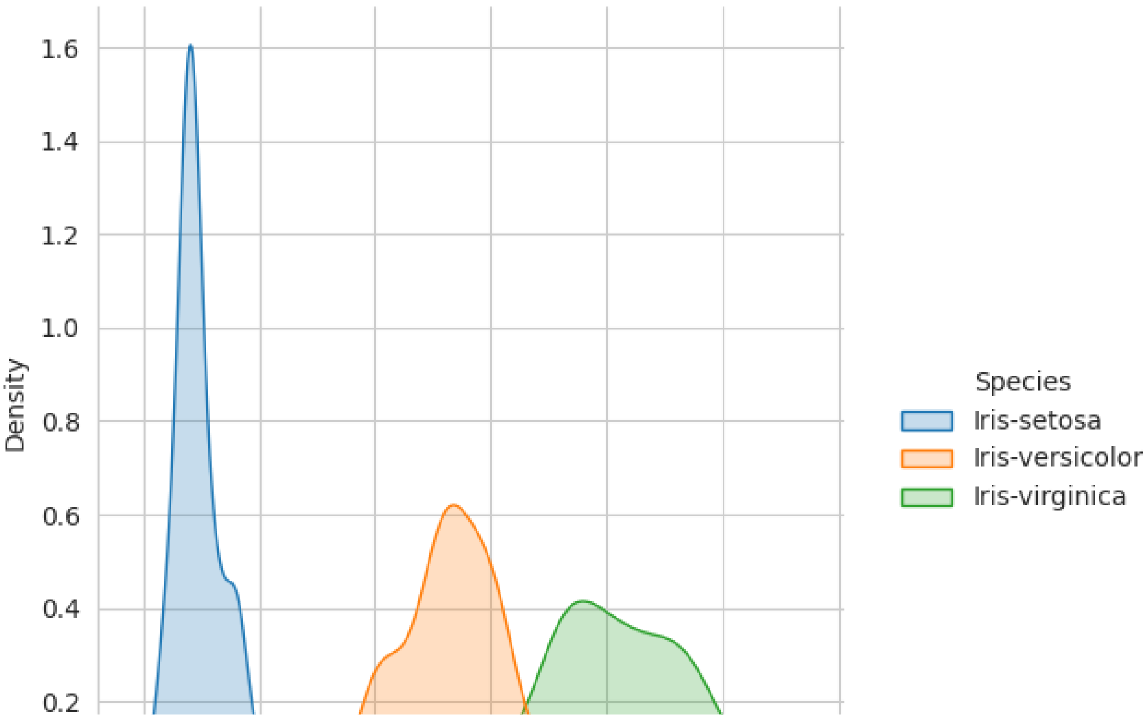
```
# Exploring distribution plot for all features
for i in iris.columns:
    if i == 'Species':
        continue
    sns.set_style('whitegrid')
    sns.displot(iris, x=i, hue='Species', kind='kde', fill=True)
    plt.show()
```

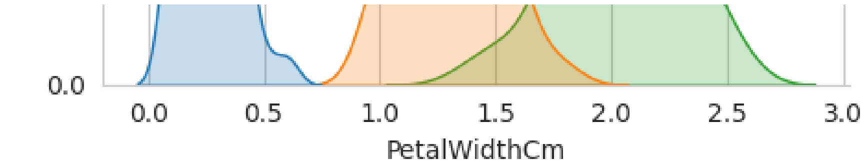


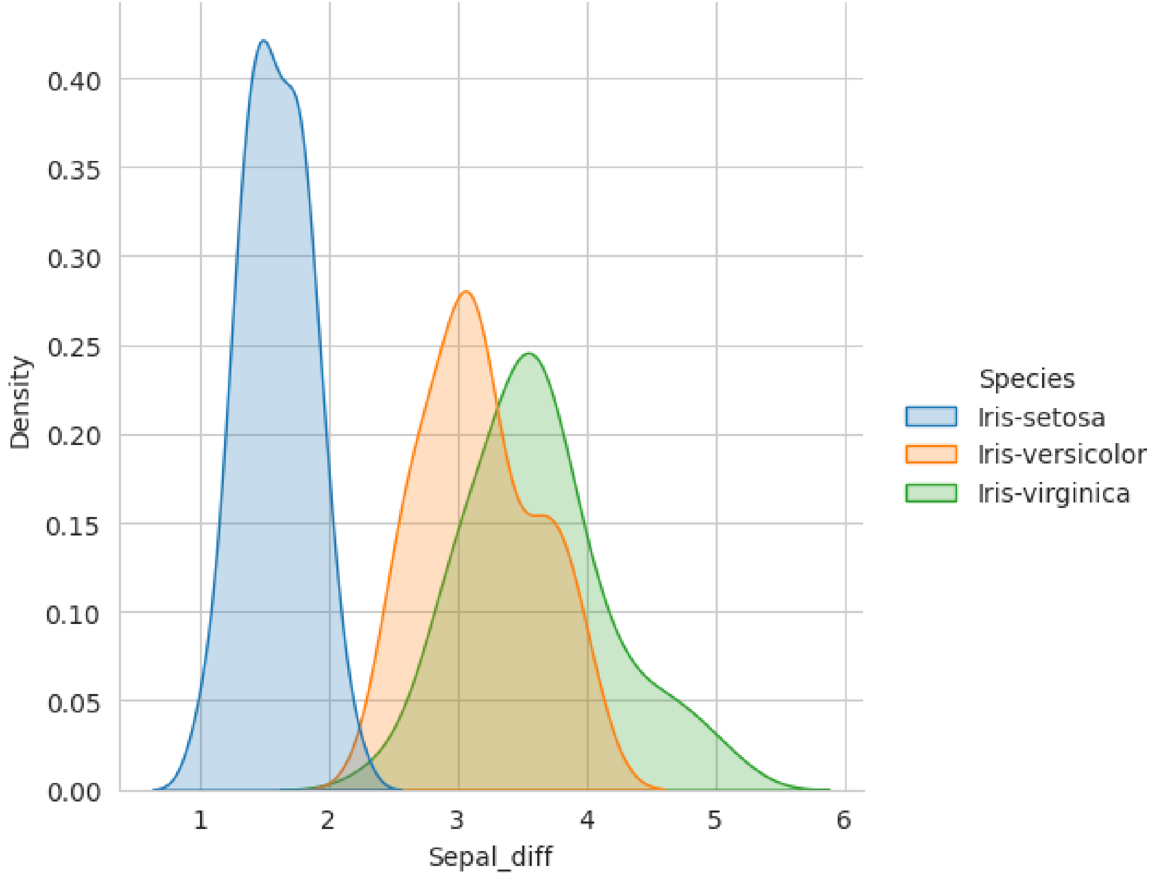


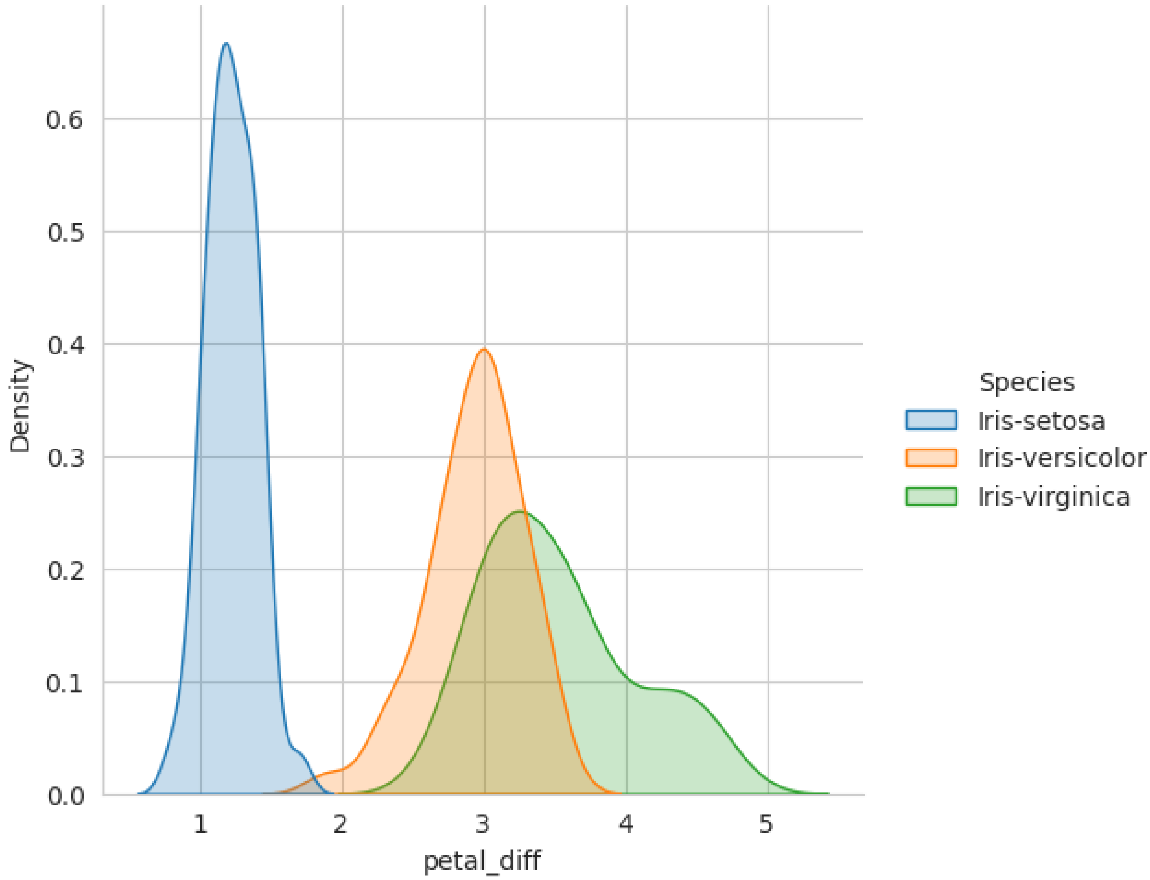

Density
0.20
0.15
0.10
0.05
0.00
1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
SepalWidthCm
Species
Iris-setosa
Iris-versicolor
Iris-virginica


Density
0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0
1 2 3 4 5 6 7 8
PetalLengthCm
Species
Iris-setosa
Iris-versicolor
Iris-virginica

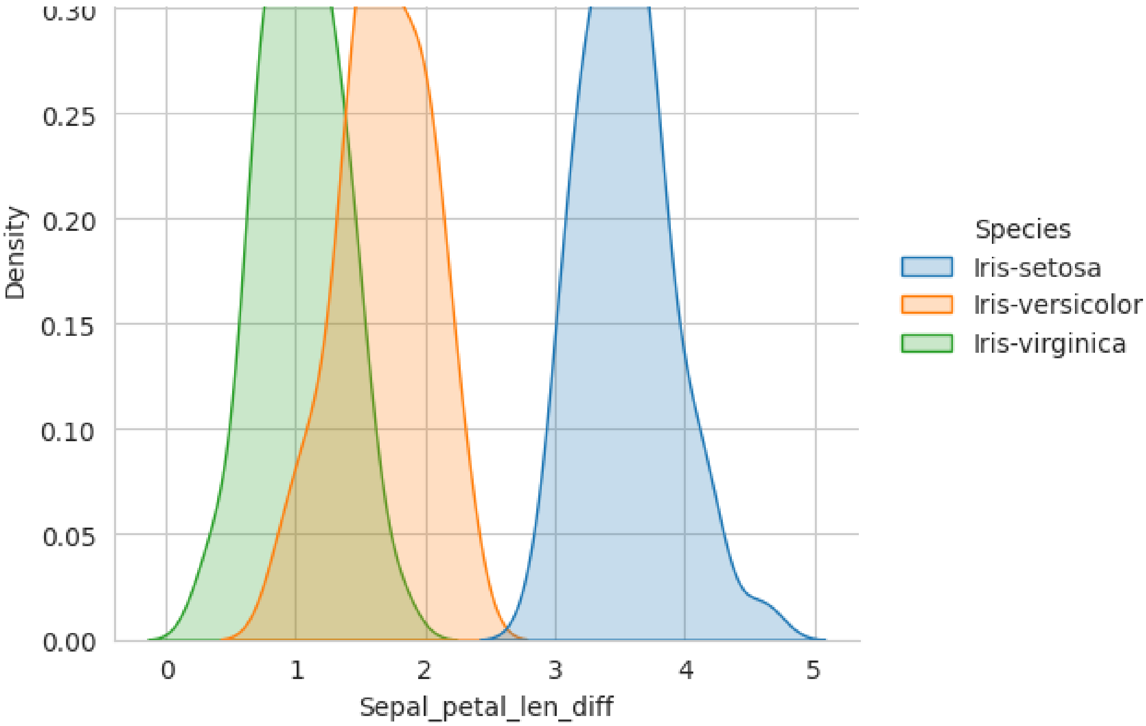

Density
1.6 1.4 1.2 1.0 0.8 0.6 0.4 0.2
Species
Iris-setosa
Iris-versicolor
Iris-virginica

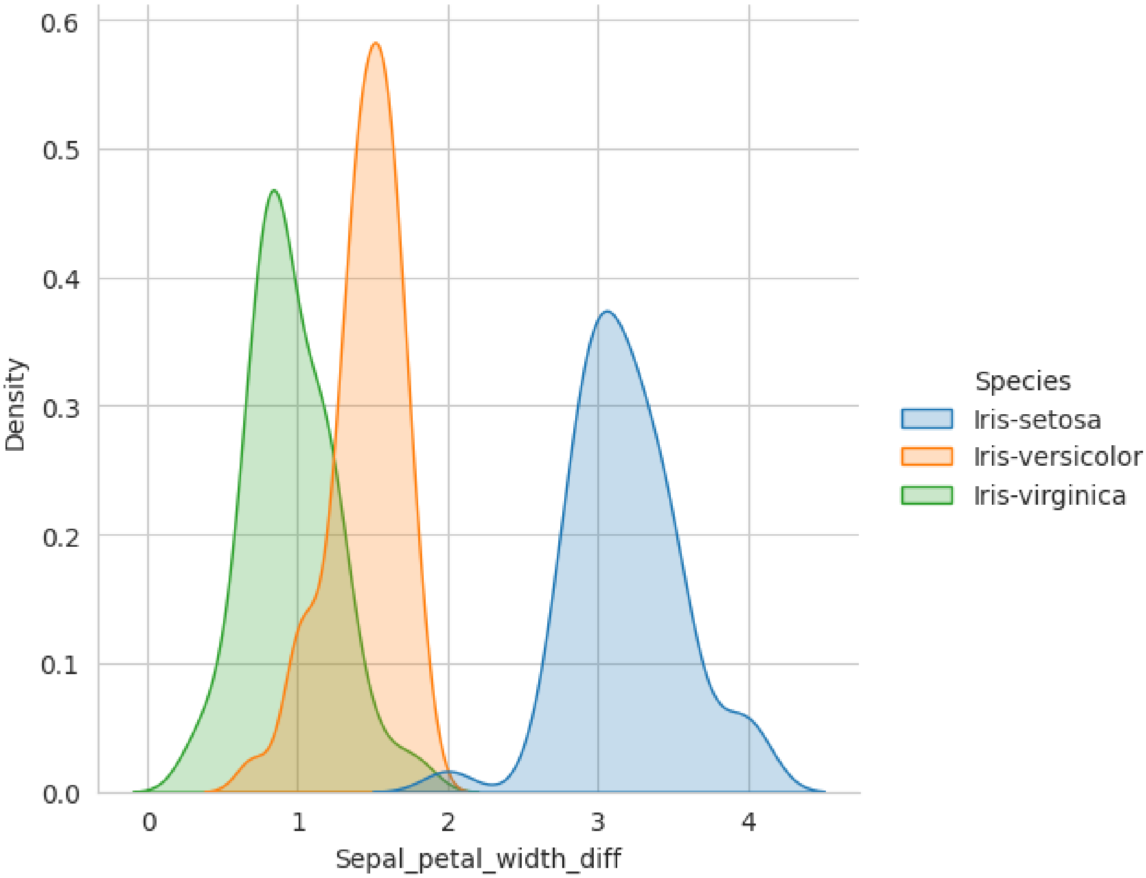

0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
PetalWidthCm

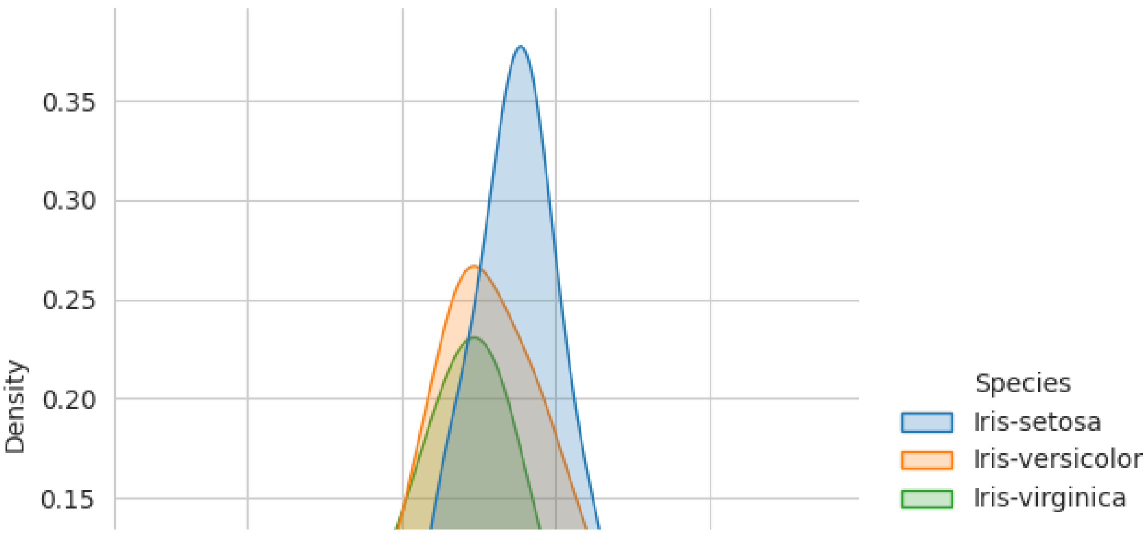

Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6
Sepal_diff
Species
Iris-setosa
Iris-versicolor
Iris-virginica

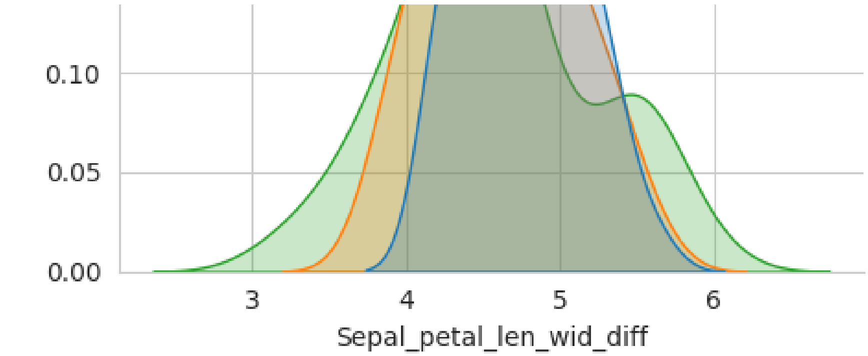

Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
petal_diff
Species
Iris-setosa
Iris-versicolor
Iris-virginica

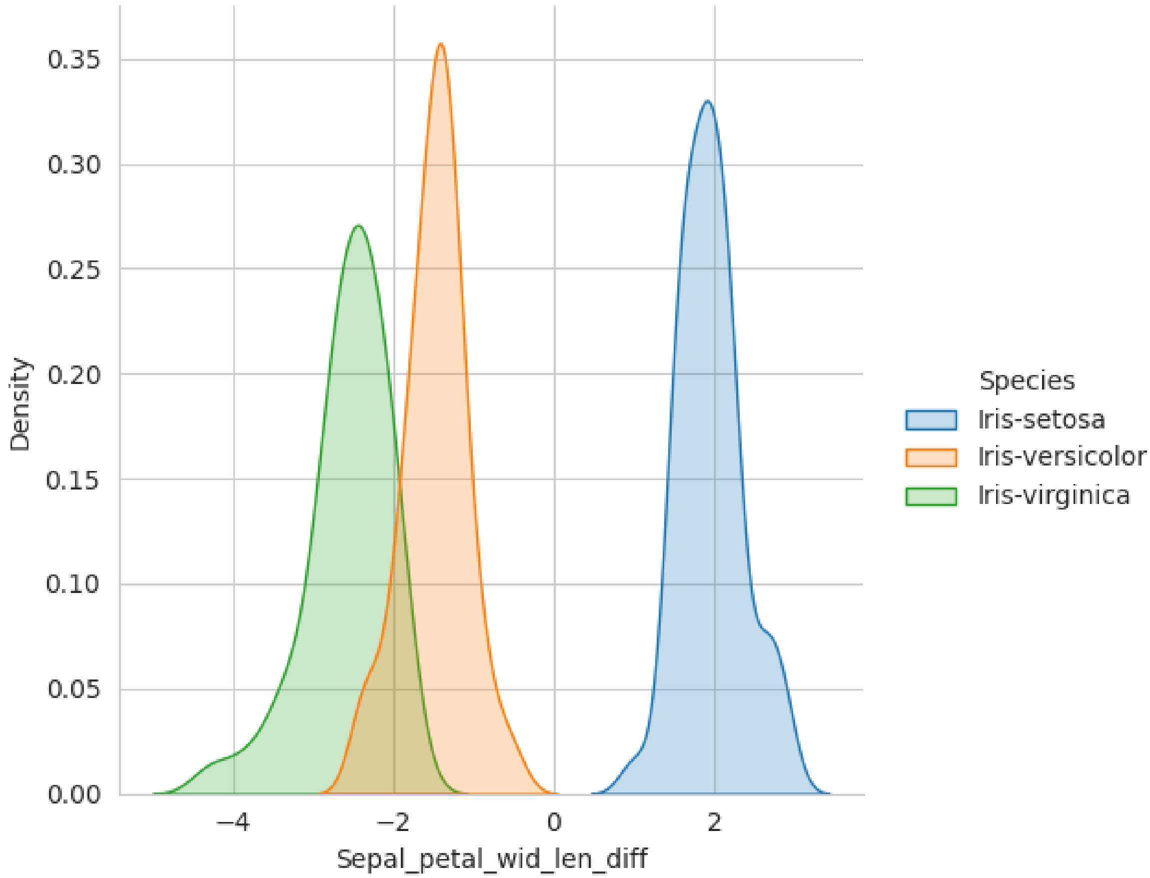

0.35


0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0
1
2
3
4
5
Sepal_petal_len_diff
Species
Iris-setosa
Iris-versicolor
Iris-virginica


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
0
1
2
3
4
Sepal_petal_width_diff
Species
Iris-setosa
Iris-versicolor
Iris-virginica


0.35
0.30
0.25
0.20
0.15
Density
Species
Iris-setosa
Iris-versicolor
Iris-virginica





Building Classification Model

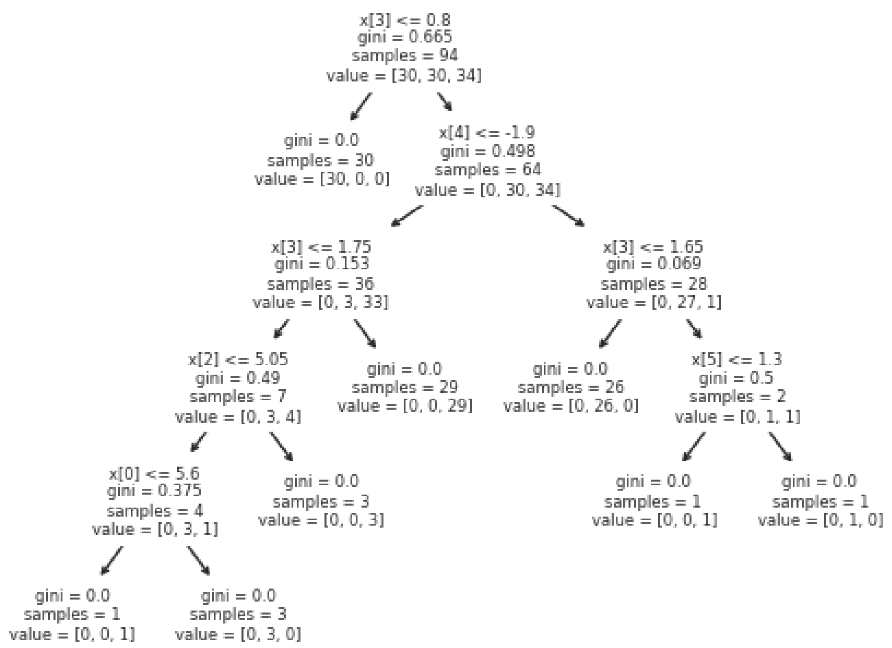
In [147]:

```
from sklearn import tree
import graphviz
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score

X = iris[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm','Sepal_petal_wi
d_len_diff','Sepal_petal_width_diff']]
y = iris['Species']


#Before training the model we have split our data into Actual Train and Actual Test Datas
et for training and validating purpose...

Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.30, random_state=42)

#spliting data into validation train and validation test
Xt, Xcv, Yt, Ycv = train_test_split(Xtrain, Ytrain, test_size=0.10, random_state=42)

'''Now we have create a Decision tree classifier and trained it with training dataset.'''


Iris_clf = DecisionTreeClassifier(criterion='gini',min_samples_split=2)
Iris_clf.fit(Xt, Yt)
```

```
#Visualized the Tree which is formed on train dataset

tree.plot_tree(Iris_clf)
```

Out[147]:

```
[Text(0.45454545454545453, 0.9166666666666666, 'x[3] <= 0.8\ngini = 0.665\nsamples = 94\n
value = [30, 30, 34]'),
 Text(0.36363636363636365, 0.75, 'gini = 0.0\nsamples = 30\nvalue = [30, 0, 0]'),
 Text(0.5454545454545454, 0.75, 'x[4] <= -1.9\ngini = 0.498\nsamples = 64\nvalue = [0, 30
, 34]'),
 Text(0.36363636363636365, 0.5833333333333334, 'x[3] <= 1.75\ngini = 0.153\nsamples = 36\
nvalue = [0, 3, 33]'),
 Text(0.2727272727272727, 0.4166666666666667, 'x[2] <= 5.05\ngini = 0.49\nsamples = 7\nva
lue = [0, 3, 4]'),
 Text(0.18181818181818182, 0.25, 'x[0] <= 5.6\ngini = 0.375\nsamples = 4\nvalue = [0, 3,
1]'),
 Text(0.09090909090909091, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(0.2727272727272727, 0.08333333333333333, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0
]'),
 Text(0.36363636363636365, 0.25, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.45454545454545453, 0.4166666666666667, 'gini = 0.0\nsamples = 29\nvalue = [0, 0,
29]'),
 Text(0.7272727272727273, 0.5833333333333334, 'x[3] <= 1.65\ngini = 0.069\nsamples = 28\n
value = [0, 27, 1]'),
 Text(0.6363636363636364, 0.4166666666666667, 'gini = 0.0\nsamples = 26\nvalue = [0, 26,
0]'),
 Text(0.8181818181818182, 0.4166666666666667, 'x[5] <= 1.3\ngini = 0.5\nsamples = 2\nvalu
e = [0, 1, 1]'),
 Text(0.7272727272727273, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.9090909090909091, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]')]
```



In [148]:

```
#Visualizing Decision Tree using graphviz library

dot_data = tree.export_graphviz(Iris_clf, out_file=None)

graph = graphviz.Source(dot_data)
graph
```

Out[148]:

In [149]:

```
# As our model has been trained....
#Now we can validate our Decision tree using cross validation method to get the accuracy
or performance score of our model.

print('Accuracy score is:',cross_val_score(Iris_clf, Xt, Yt, cv=3, scoring='accuracy').m
ean())
```

```
Accuracy score is: 0.9254032258064516
```

In [150]:

```
#Checking validation test data on our trained model and getting performance metrices

from sklearn.metrics import multilabel_confusion_matrix, accuracy_score

Y_hat = Iris_clf.predict(Xcv)


print('Accuracy score for validation test data is:',accuracy_score(Ycv, Y_hat))
multilabel_confusion_matrix(Ycv , Y_hat)
```

```
Accuracy score for validation test data is: 0.8181818181818182
```

Out[150]:

```
array([[[10,  0],
        [ 0,  1]],

       [[ 3,  1],
        [ 1,  6]],

       [[ 7,  1],
        [ 1,  2]]])
```

In [151]:

```
#Checking our model performance on actual unseen test data..
YT_hat = Iris_clf.predict(Xtest)
YT_hat

print('Model Accuracy Score on totally unseen data(Xtest) is:',accuracy_score(Ytest, YT_h
at)*100,'%')
multilabel_confusion_matrix(Ytest , YT_hat)
```

```
Model Accuracy Score on totally unseen data(Xtest) is: 100.0 %
```

Out[151]:

```
array([[[26,  0],
        [ 0, 19]],

       [[32,  0],
        [ 0, 13]],

       [[32,  0],
        [ 0, 13]]])
```

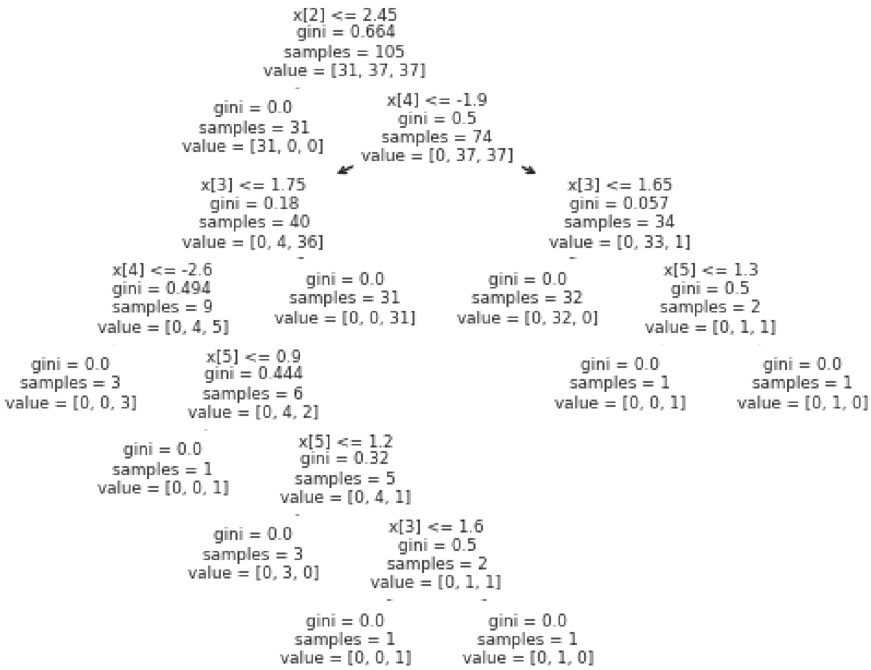
In [152]:

```
'''Training model on Actual train data... '''
Iris_Fclf = DecisionTreeClassifier(criterion='gini',min_samples_split=2)
Iris_Fclf.fit(Xtrain, Ytrain)

#Visualize tree structure..
tree.plot_tree(Iris_Fclf)
```

Out[152]:

```
[Text(0.4, 0.9375, 'x[2] <= 2.45\ngini = 0.664\nsamples = 105\nvalue = [31, 37, 37]'),
 Text(0.3, 0.8125, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(0.5, 0.8125, 'x[4] <= -1.9\ngini = 0.5\nsamples = 74\nvalue = [0, 37, 37]'),
 Text(0.3, 0.6875, 'x[3] <= 1.75\ngini = 0.18\nsamples = 40\nvalue = [0, 4, 36]'),
 Text(0.2, 0.5625, 'x[4] <= -2.6\ngini = 0.494\nsamples = 9\nvalue = [0, 4, 5]'),
```

```
 Text(0.1, 0.4375, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.3, 0.4375, 'x[5] <= 0.9\ngini = 0.444\nsamples = 6\nvalue = [0, 4, 2]'),
 Text(0.2, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.4, 0.3125, 'x[5] <= 1.2\ngini = 0.32\nsamples = 5\nvalue = [0, 4, 1]'),
 Text(0.3, 0.1875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0]'),
 Text(0.5, 0.1875, 'x[3] <= 1.6\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.4, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.6, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.4, 0.5625, 'gini = 0.0\nsamples = 31\nvalue = [0, 0, 31]'),
 Text(0.7, 0.6875, 'x[3] <= 1.65\ngini = 0.057\nsamples = 34\nvalue = [0, 33, 1]'),
 Text(0.6, 0.5625, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(0.8, 0.5625, 'x[5] <= 1.3\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.7, 0.4375, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.9, 0.4375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]')]
```



In [153]:

```
#Final Decision tree build for deploying in real world cases....

dot_data = tree.export_graphviz(Iris_Fclf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

Out[153]:

In [154]:

```
#Checking the performance of model on Actual Test data...

YT_Fhat = Iris_Fclf.predict(Xtest)
YT_Fhat

print('Model Accuracy Score on totally unseen data(Xtest) is:',accuracy_score(Ytest, YT_F
hat)*100,'%')
multilabel_confusion_matrix(Ytest , YT_Fhat)
```

```
Model Accuracy Score on totally unseen data(Xtest) is: 93.33333333333333 %
```

Out[154]:

```
array([[[26,  0],
        [ 0, 19]],

       [[32,  0],
        [ 3, 10]],
```

```
        [[29,  3],
         [ 0, 13]]])
```

In [175]:

```python
#Testing for New points except from Dataset

Test_point = [[5.4,3.0,4.5,1.5,-1.5,1.5],
              [6.5,2.8,4.6,1.5,-1.8,1.3],
              [5.1,2.5,3.0,1.1,-0.5,1.4],
              [5.1,3.3,1.7,0.5,1.6,2.8],
              [6.0,2.7,5.1,1.6,-2.4,1.1],
              [6.0,2.2,5.0,1.5,-2.8,0.7]]

print(Iris_Fclf.predict(Test_point))
```

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

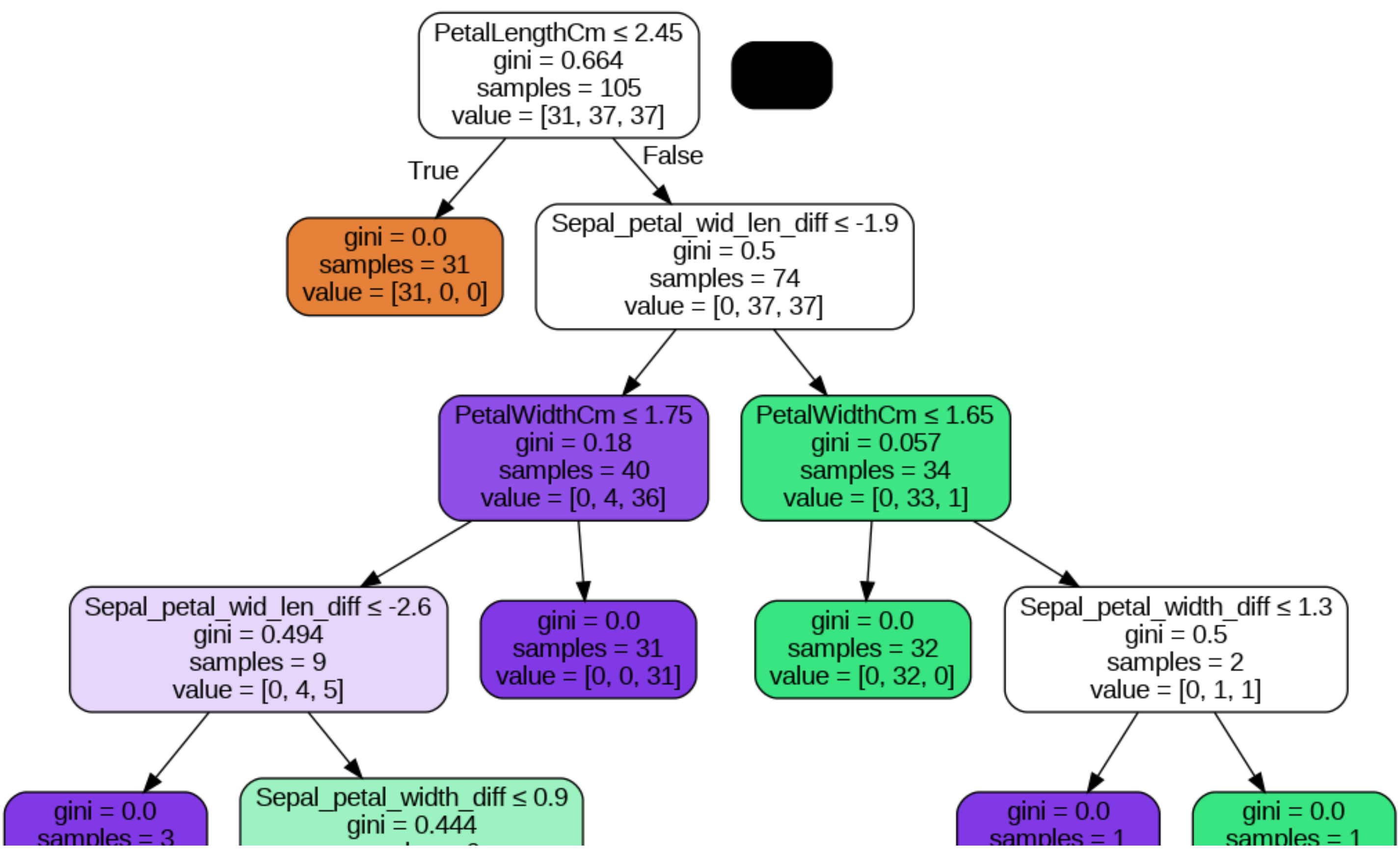
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have
valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

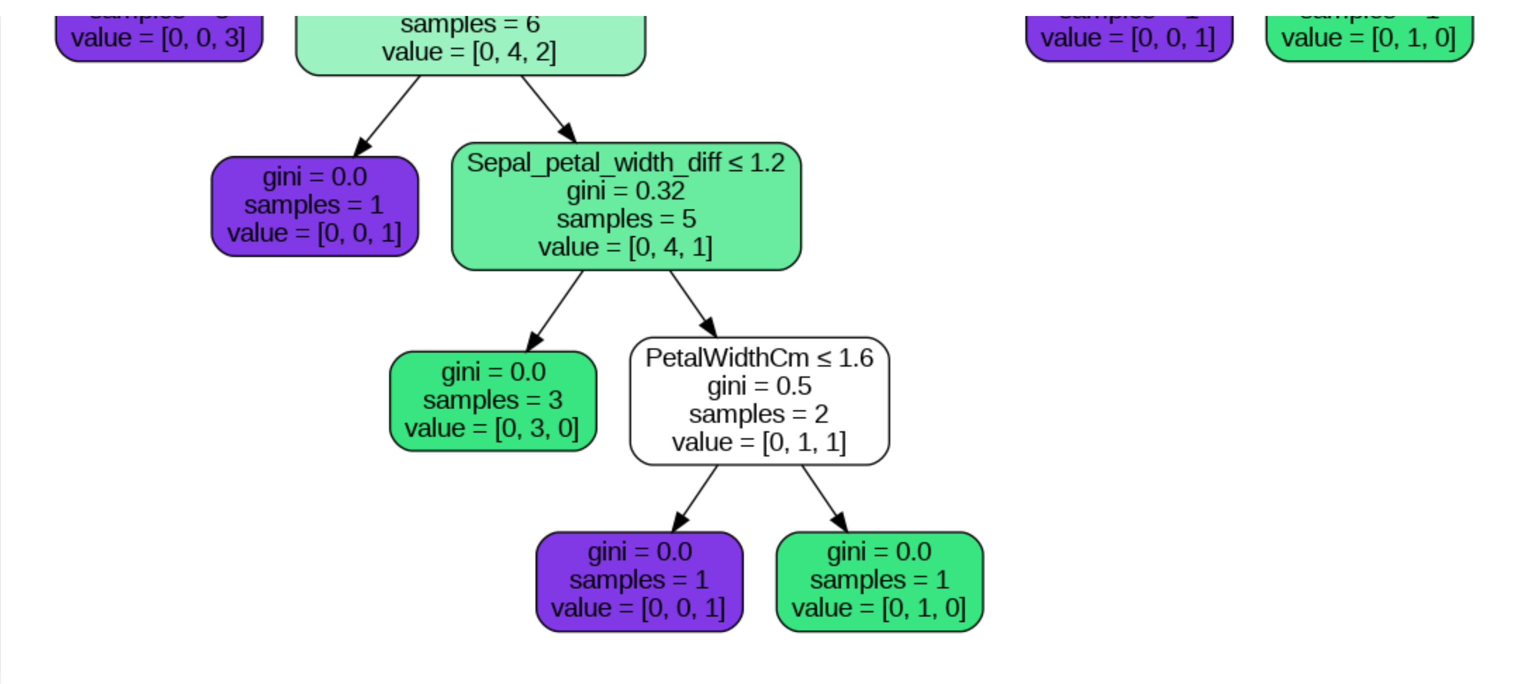
Let us visualize the Decision Tree to understand it better.

In [174]:

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

# Visualize the graph
dot_data = StringIO()
export_graphviz(Iris_Fclf, out_file=dot_data, feature_names=X.columns,
                filled=True, rounded=True,
                special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[174]:




samples = 3
value = [0, 0, 3]
samples = 6
value = [0, 4, 2]
samples = 1
value = [0, 0, 1]
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]
Sepal_petal_width_diff ≤ 1.2
gini = 0.32
samples = 5
value = [0, 4, 1]
gini = 0.0
samples = 3
value = [0, 3, 0]
PetalWidthCm ≤ 1.6
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 1
value = [0, 0, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]