·•·⊰·○☽ GRIP: The Sparks Foundation ☾○·⊱·•·

Task 2 :Prediction using Decision Tree Algorithm

Objectives:

Create the Decision Tree classifier and visualize it graphically.

---

1) Decision trees in machine learning provide an effective method for making decisions because they lay out the problem and all the possible outcomes. 2) Decision trees are used to solve classification problems and categorize objects depending on their learning features.

The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

Author: Neha Parameshwar Sonavane

In [1]:
```
# Importing required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# First we load the dataset
data = pd.read_csv(r"C:\Users\DELL\Downloads\Iris (1).csv")
data.head(5)              # showing first 5 records of the dataset
```

Out[2]:

	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 [3]:
```
data.tail(5)              #which shows last 5 records of the given dataset
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

Exploratory Data Analysis

In [4]:
```
data.shape              # Nnumber of records and columns
```

Out[4]: (150, 6)

In [5]:
```
# Drop the feature which is not used for the analysis
df=data.drop(['Id'],axis=1)
df
```

Out[5]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [6]:
```
df.info()            # Information about the data
```

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

In [7]:
```
data.isna().sum()                 # Null values
```

Out[7]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

There are no missing values present in our data. The data is cleaned and ready for analysis.

In [8]:
```
df.describe()   # summary statistics
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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 [9]:
```
df.columns          # what features are available in our data
```

Out[9]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [10]:
```
df.nunique()     # unique values
```

Out[10]:
```
SepalLengthCm    35
SepalWidthCm     23
PetalLengthCm    43
PetalWidthCm     22
Species           3
dtype: int64
```

In [11]:
```
df.corr()       # correlation between the features
```

Out[11]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [12]:
```
df['Species'].value_counts()
```

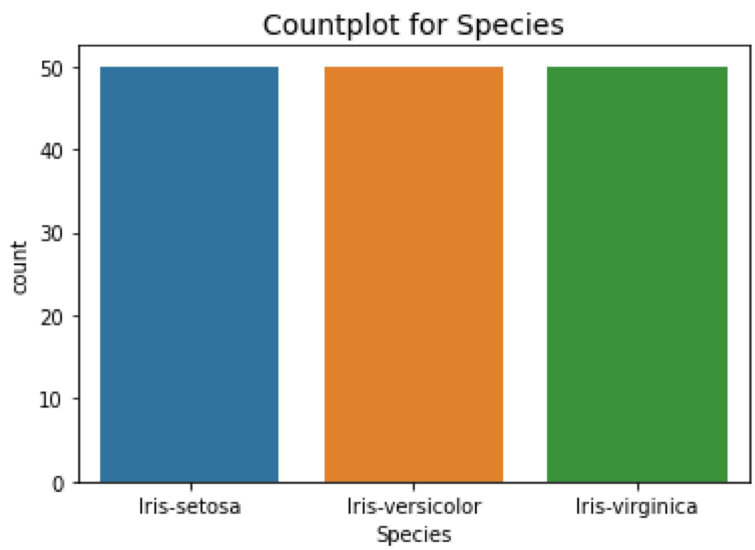
Out[12]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

We observe that the data is balanced.

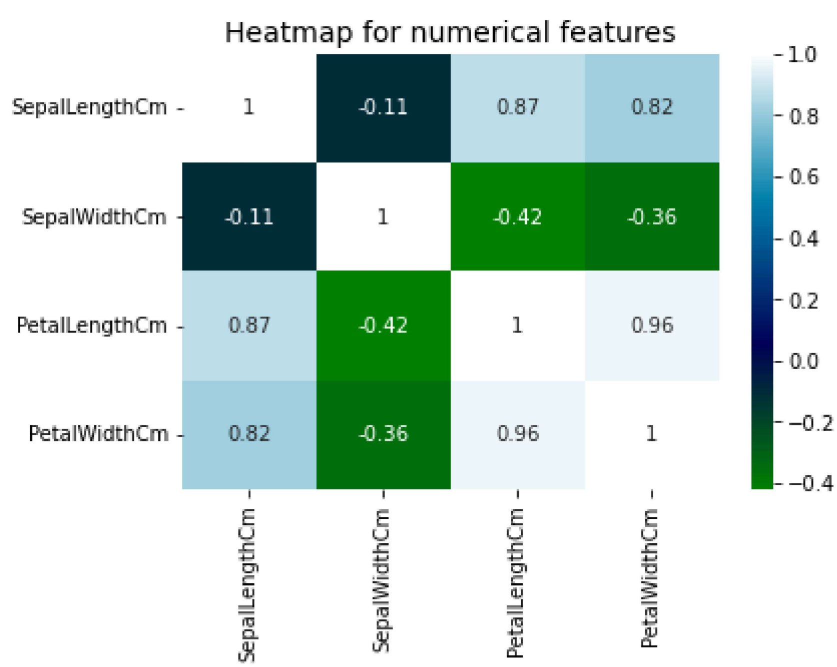
Data Visualization

In [13]:
```
c=sns.countplot(df['Species'])
c.set_title('Countplot for Species',fontsize=14)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

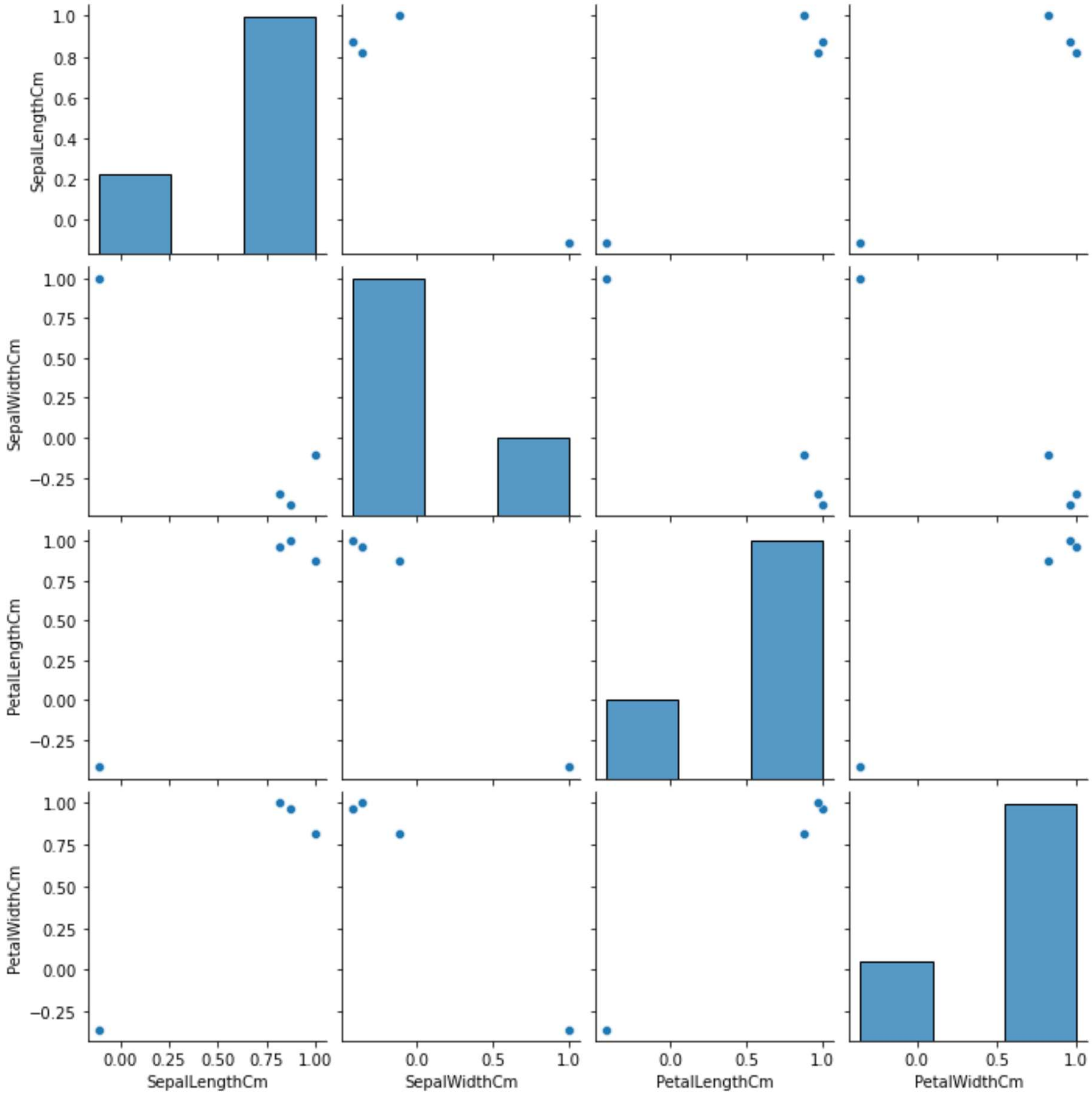


In [14]:
```
corr=sns.heatmap(df.corr(),annot=True,cmap='ocean')
corr.set_title('Heatmap for numerical features',fontsize=14)
plt.show()
```

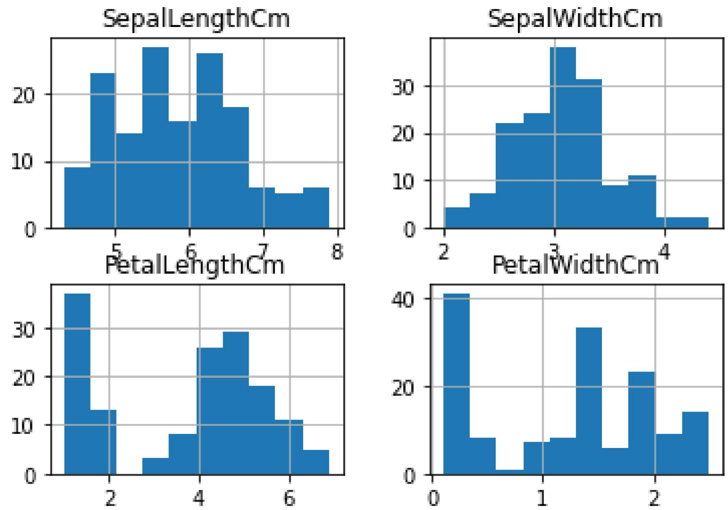


The correlation plot shows that there is high correlation between sepalength and petallength, sepallength and petalwidth, petallength and petalwidth.

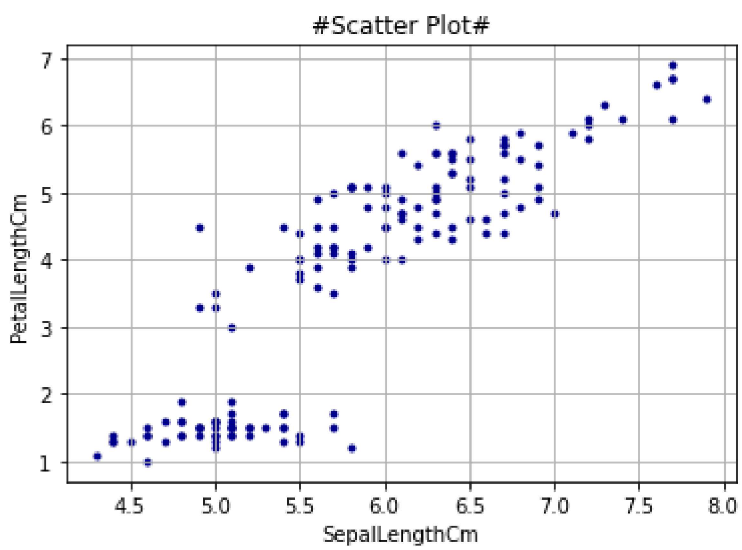
In [15]:
```
sns.pairplot(df.corr())
plt.show()
```



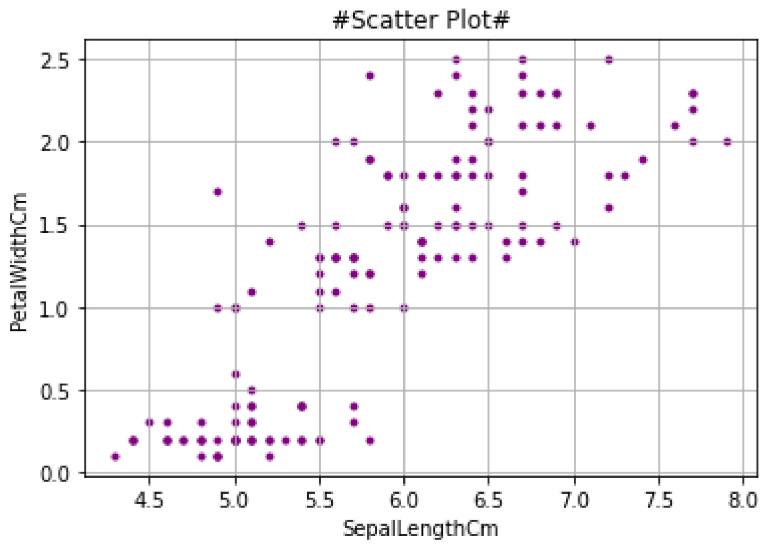
In [16]:
```
df.hist()
plt.show()
```



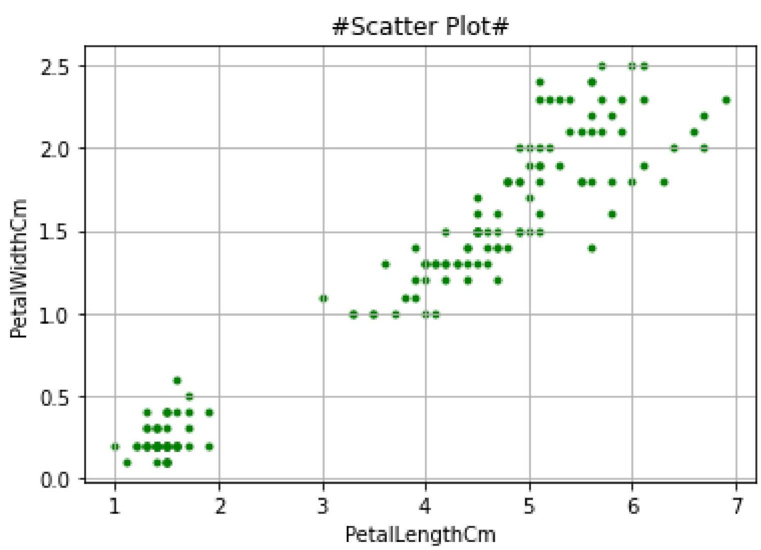
In [17]:
```
plt.scatter(df['SepalLengthCm'], df['PetalLengthCm'],marker='.',color='darkblu
plt.title('#Scatter Plot#')
plt.xlabel('SepalLengthCm')
plt.ylabel('PetalLengthCm')
plt.grid()
plt.show()
```



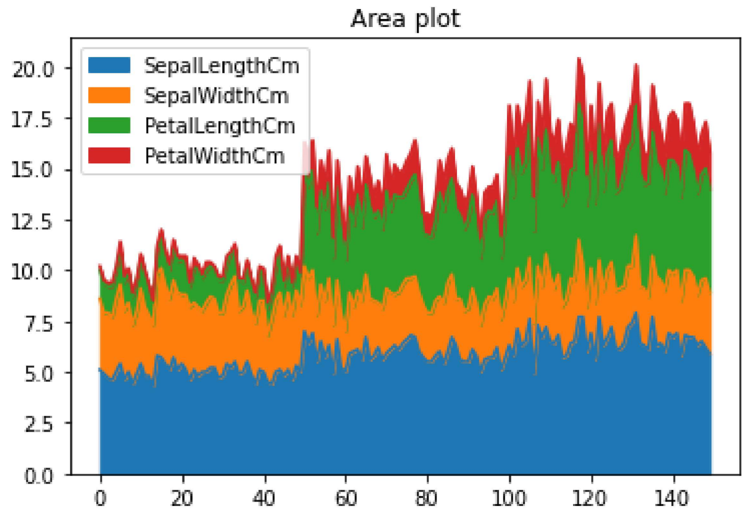
In [18]:
```
plt.scatter(df['SepalLengthCm'], df['PetalWidthCm'],marker='.',color='purple')
plt.title('#Scatter Plot#')
plt.xlabel('SepalLengthCm')
plt.ylabel('PetalWidthCm')
plt.grid()
plt.show()
```



In [19]:
```
plt.scatter(df['PetalLengthCm'], df['PetalWidthCm'],marker='.',color='green')
plt.title('#Scatter Plot#')
plt.xlabel('PetalLengthCm')
plt.ylabel('PetalWidthCm')
plt.grid()
plt.show()
```



In [20]:
```
df.plot.area()
plt.title('Area plot')
plt.show()
```



Converting categorical variable species into numerical variable by label encoding

In [21]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['Species']= label_encoder.fit_transform(df['Species'])

df['Species'].unique()
```

Out[21]:
```
array([0, 1, 2])
```

In [22]:
```
df['Species']
```

Out[22]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int32
```

In [23]:
```
# Splitting dependent and independent variable
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [24]:
```
# train test split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =train_test_split(x,y,test_size=0.2,random_st
```

Model building

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
dtree= DecisionTreeClassifier()
dtree.fit(x_train,y_train)       #fitting Decision tree classifier to the dat
```

Out[25]:
```
DecisionTreeClassifier()
```

prediction

In [26]:
```
y_pred= dtree.predict(x_test)
y_pred
```

Out[26]:
```
array([2, 0, 1, 1, 2, 0, 2, 1, 2, 1, 0, 2, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1,
       0, 1, 0, 0, 2, 2, 2, 1])
```

confusion matrix, precision of the model

In [27]:
```
from sklearn.metrics import confusion_matrix,precision_score,recall_score,accu
```

In [28]:
```
c_m=confusion_matrix(y_test,y_pred)
pre_score=precision_score(y_test,y_pred,average='weighted')
recall_score=recall_score(y_test,y_pred,average='weighted')
acc_score=accuracy_score(y_test,y_pred)
f_score=f1_score(y_test,y_pred,average='weighted')
```

In [29]:
```
print(f'precision score: {pre_score}')
print(f'recall_score: {recall_score}')
print(f'accuracy_score: {acc_score}')
print(f'f1_score: {f_score}')
```

```
precision score: 0.9692307692307692
recall_score: 0.9666666666666667
accuracy_score: 0.9666666666666667
f1_score: 0.9666086956521741
```

In [30]:
```
print(f'confusion matrix: {c_m}')
```

```
confusion matrix: [[ 6  0  0]
 [ 0 11  1]
 [ 0  0 12]]
```

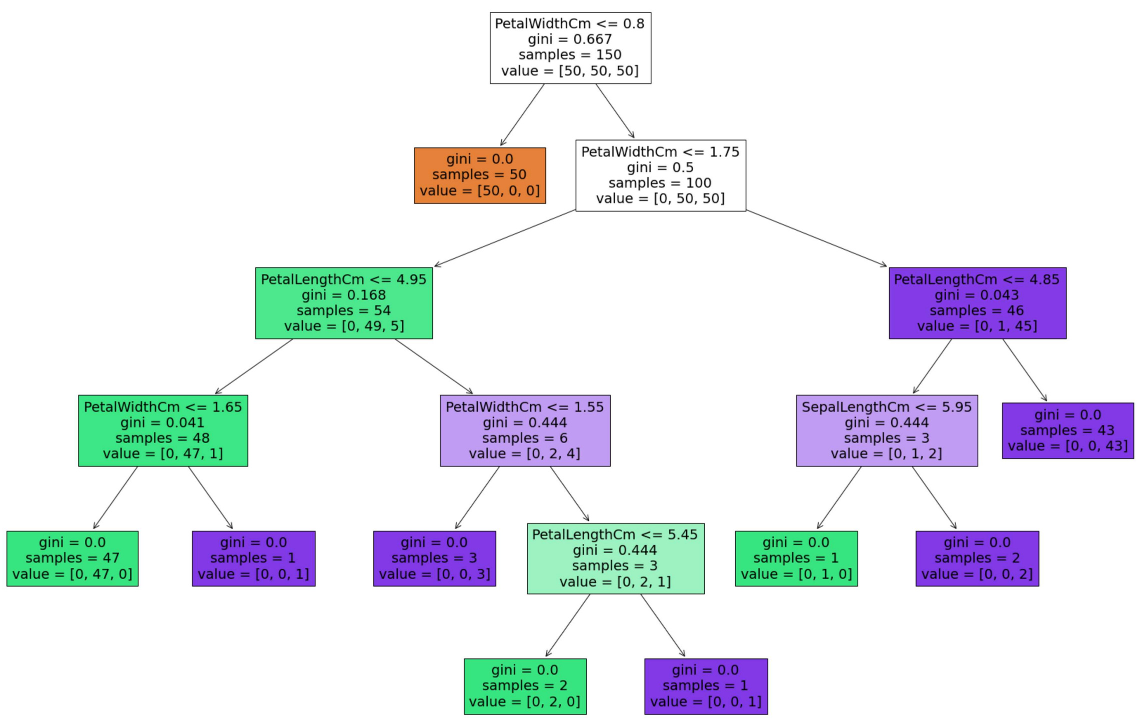
Data visualization of tree

In [31]:
```
from sklearn import tree
features = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']

x = df[features]
y = df['Species']

dtree = DecisionTreeClassifier()
dtree = dtree.fit(x, y)

plt.figure(figsize=(30,20))
tree.plot_tree(dtree, feature_names=features,filled=True)
plt.show()
```



😊 Thank You 😊

In [ ]: