>Intermediate Level Task...

TASK -5 Prediction using Decision Tree Algorithm :

Create the Decision Tree classifier and visualize it graphically.

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

dtaset: https://bit.ly/2TK5Xn5

Name: Manish Singh

1. Importing Required Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
print(" All required packages included successfully!")
```

```
All required packages included successfully!
```

2. Importing the Dataset

In [3]:
```
dataset = pd.read_csv('D:\Data_Set\Iris.csv')
dataset.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	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

3. Data Exploration

In [5]:
```
# Shape of Dataset
dataset.shape
```

Out[5]:
```
(150, 5)
```

In [6]:
```
# Dataset Columns
dataset.columns
```

Out[6]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [10]:
```
# To display basic data
dataset.info()
```

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

In [9]:
```
# to display stats about data
dataset.describe()
```

Out[9]:

	sepal_length	sepal_width	petal_length	petal_width
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 [11]:
```
#Checking Null Values
dataset.isnull().sum()
```

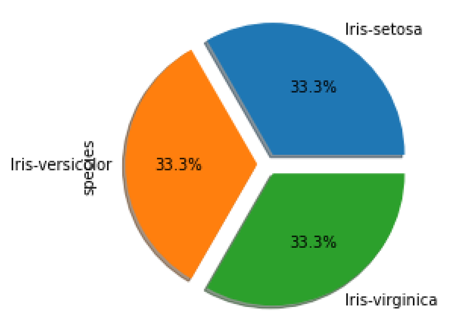
Out[11]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [21]:
```
#Checking columns count of "Species"
dataset['species'].value_counts()
```

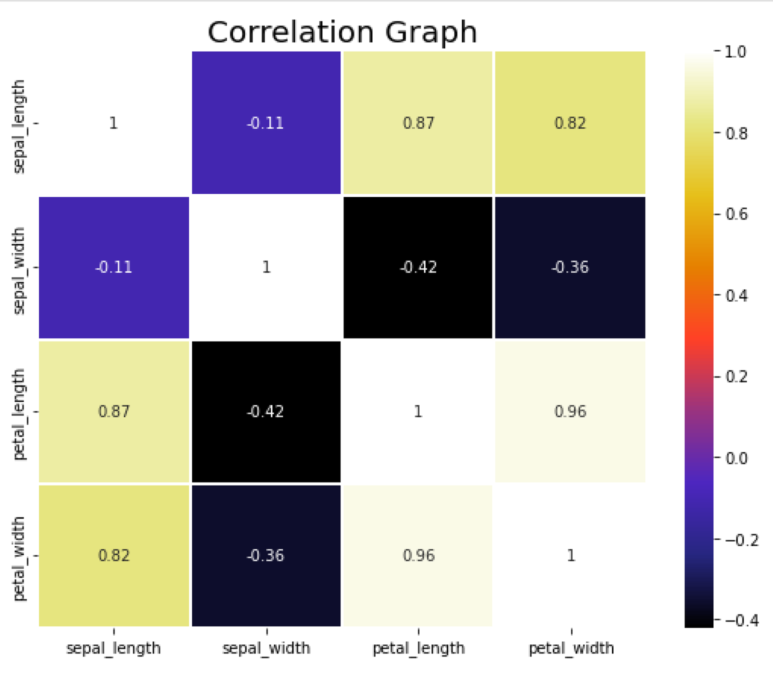
Out[21]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

In [23]:
```
#Pie plot to show the overall types of Iris classifications
dataset['species'].value_counts().plot(kind = 'pie',  autopct = '%1.1f%%', shadow = True, explode = [0.08,0.08,0.08])
```

Out[23]:
```
<AxesSubplot:ylabel='species'>
```



In [24]:
```
#Correlation Heatmap
plt.figure(figsize=(9,7))
sns.heatmap(dataset.corr(),cmap='CMRmap',annot=True,linewidths=2)
plt.title("Correlation Graph",size=20)
plt.show()
```



4. Defining dependent and independent variables

In [31]:
```
features = ['sepal_length','sepal_width','petal_length','petal_width']
X = dataset.loc[:, features].values   #defining the feature matrix
y = dataset.species
```

5. Splitting the dataset into training and test sets

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,random_state=0)
```

6. Defining the decision tree classifier and fitting the training set

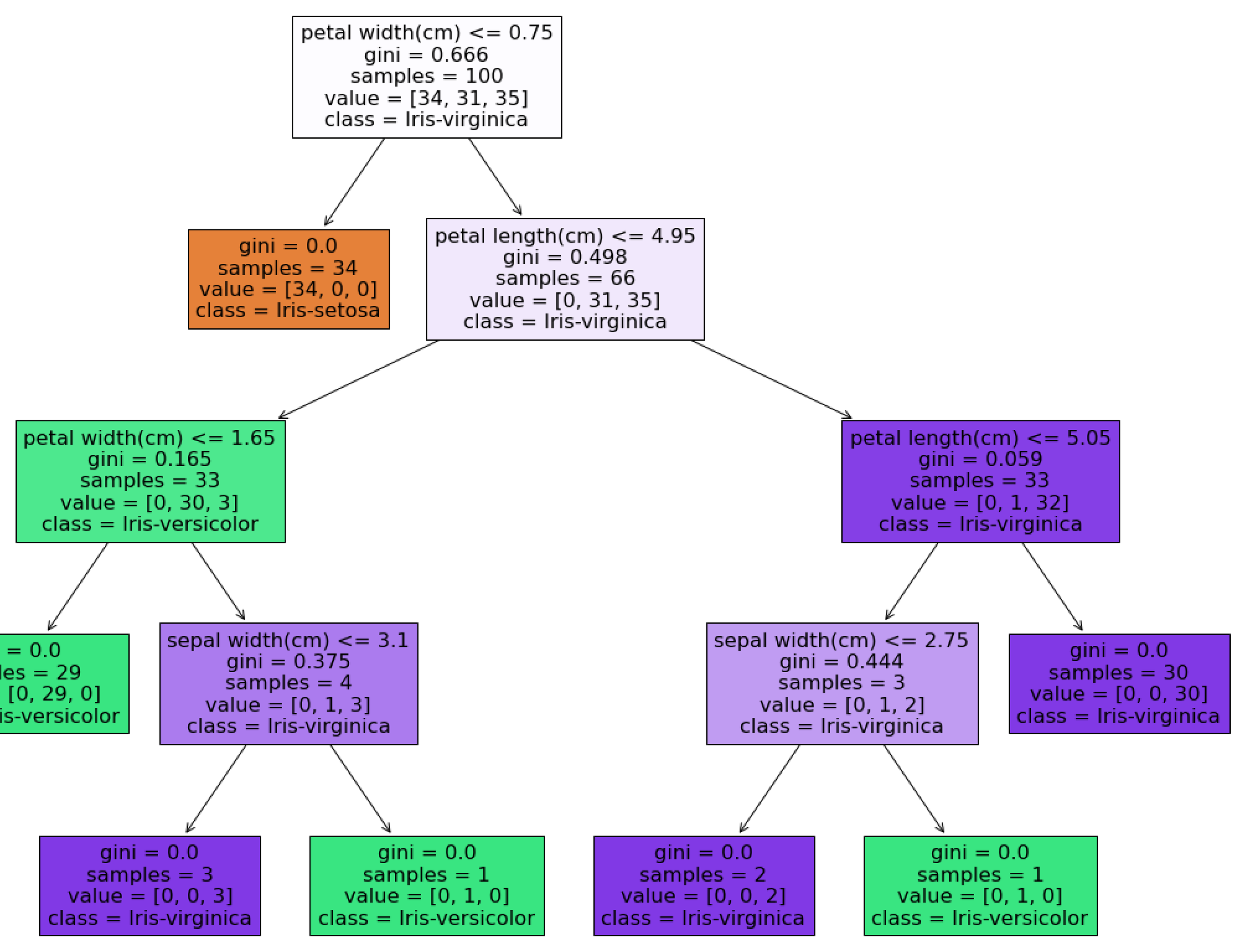
In [33]:
```
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

Out[33]:
```
DecisionTreeClassifier()
```

7. Visualizing of Decision tree

In [37]:
```
from sklearn import tree
feature_name =  ['sepal length(cm)','sepal width(cm)','petal length(cm)','petal width(cm)']
class_name= dataset.species.unique()
plt.figure(figsize=(20,15))
tree.plot_tree(dtree, filled = True, feature_names = feature_name, class_names= class_name)
```

Out[37]:
```
[Text(446.4, 733.86, 'petal width(cm) <= 0.75\ngini = 0.666\nsamples = 100\nvalue = [34, 31, 35]\nclass = Iris-virginica'),
 Text(334.79999999999995, 570.78, 'gini = 0.0\nsamples = 34\nvalue = [34, 0, 0]\nclass = Iris-setosa'),
 Text(558.0, 570.78, 'petal length(cm) <= 4.95\ngini = 0.498\nsamples = 66\nvalue = [0, 31, 35]\nclass = Iris-virginica'),
 Text(223.2, 407.70000000000005, 'petal width(cm) <= 1.65\ngini = 0.165\nsamples = 33\nvalue = [0, 30, 3]\nclass = Iris-versicolor'),
 Text(111.6, 244.62, 'gini = 0.0\nsamples = 29\nvalue = [0, 29, 0]\nclass = Iris-versicolor'),
 Text(334.79999999999995, 244.62, 'sepal width(cm) <= 3.1\ngini = 0.375\nsamples = 4\nvalue = [0, 1, 3]\nclass = Iris-virginica'),
 Text(223.2, 81.54000000000008, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]\nclass = Iris-virginica'),
 Text(446.4, 81.54000000000008, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = Iris-versicolor'),
 Text(892.8, 407.70000000000005, 'petal length(cm) <= 5.05\ngini = 0.059\nsamples = 33\nvalue = [0, 1, 32]\nclass = Iris-virginica'),
 Text(781.1999999999999, 244.62, 'sepal width(cm) <= 2.75\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]\nclass = Iris-virginica'),
 Text(669.5999999999999, 81.54000000000008, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]\nclass = Iris-virginica'),
 Text(892.8, 81.54000000000008, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = Iris-versicolor'),
 Text(1004.4, 244.62, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]\nclass = Iris-virginica')]
```



8. Prediction on Dataset.

In [41]:
```
y_dataset = dtree.predict(X_test)
y_dataset
```

Out[41]:
```
array(['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-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica'], dtype=object)
```

9. To check the accuracy of the model..

In [42]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.98
```

In [43]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       1.00      0.95      0.97        19
 Iris-virginica       0.94      1.00      0.97        15

       accuracy                           0.98        50
      macro avg       0.98      0.98      0.98        50
   weighted avg       0.98      0.98      0.98        50
```

In [44]:
```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_pred)
```

Out[44]:
```
array([[16,  0,  0],
       [ 0, 18,  1],
       [ 0,  0, 15]], dtype=int64)
```

10. Prediction the output class for random values for petal and sepal length and width

Predict the flower type for a flower with sepal length, sepal width, petal length, petal width as 5cm, 3.6cm, 1.4cm and 0.2cm respectively

In [45]:
```
dtree.predict([[5, 3.6, 1.4 , 0.2]])
```

Out[45]:
```
array(['Iris-setosa'], dtype=object)
```

Predict the flower type for a flower with sepal length, sepal width, petal length, petal width as 9cm, 3.1cm, 5cm and 1.5cm respectively

In [46]:
```
dtree.predict([[9, 3.1, 5, 1.5]])
```

Out[46]:
```
array(['Iris-versicolor'], dtype=object)
```

Predict the flower type for a flower with sepal length, sepal width, petal length, petal width as 4.1cm, 3cm, 5.1cm and 1.8cm respectively

In [47]:
```
dtree.predict([[4.1, 3.0, 5.1, 1.8]])
```

Out[47]:
```
array(['Iris-virginica'], dtype=object)
```

THANK YOU !