DECISION TREE (iris Flower classifications)

1. Import Necessary libraries

In [2]:

```
import seaborn as sns
iris_data=sns.load_dataset('iris')
```

In [3]:

```
sns.get_dataset_names()
```

Out[3]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'taxis',
 'tips',
 'titanic']
```

In [4]:

```
iris_data
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

3.Data Understanding

In [5]:

```
iris_data.shape
```

Out[5]:

```
(150, 5)
```

In [6]:

```
iris_data.dtypes
```

Out[6]:

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [7]:

```
iris_data.isna().sum()
```

Out[7]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

4.Data Preprocessing

In [8]:

```
from sklearn.preprocessing import LabelEncoder
le_encoder=LabelEncoder()
iris_data['species']=le_encoder.fit_transform(iris_data['species'])
iris_data
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [32]:

```
iris_data.dtypes()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13484/3836864080.py in <module>
----> 1 iris_data.dtypes()

TypeError: 'Series' object is not callable
```

5. Model Building

In [10]:

```
x=iris_data.drop(['species'],axis=1)
y=iris_data[['species']]
```

In [11]:

```
x.shape,y.shape
```

Out[11]:

```
((150, 4), (150, 1))
```

In [12]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.15,random_state=12)
```

In [13]:

```
x_train.shape,y_train.shape
```

Out[13]:

```
((127, 4), (127, 1))
```

In [14]:

```
x_test.shape,y_test.shape
```

Out[14]:

```
((23, 4), (23, 1))
```

6.Model Training

In [15]:

```
from sklearn.tree import DecisionTreeClassifier

plt.figure(figsize=(10,7))
dt_model=DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[15]:

```
DecisionTreeClassifier()
```

```
<Figure size 720x504 with 0 Axes>
```

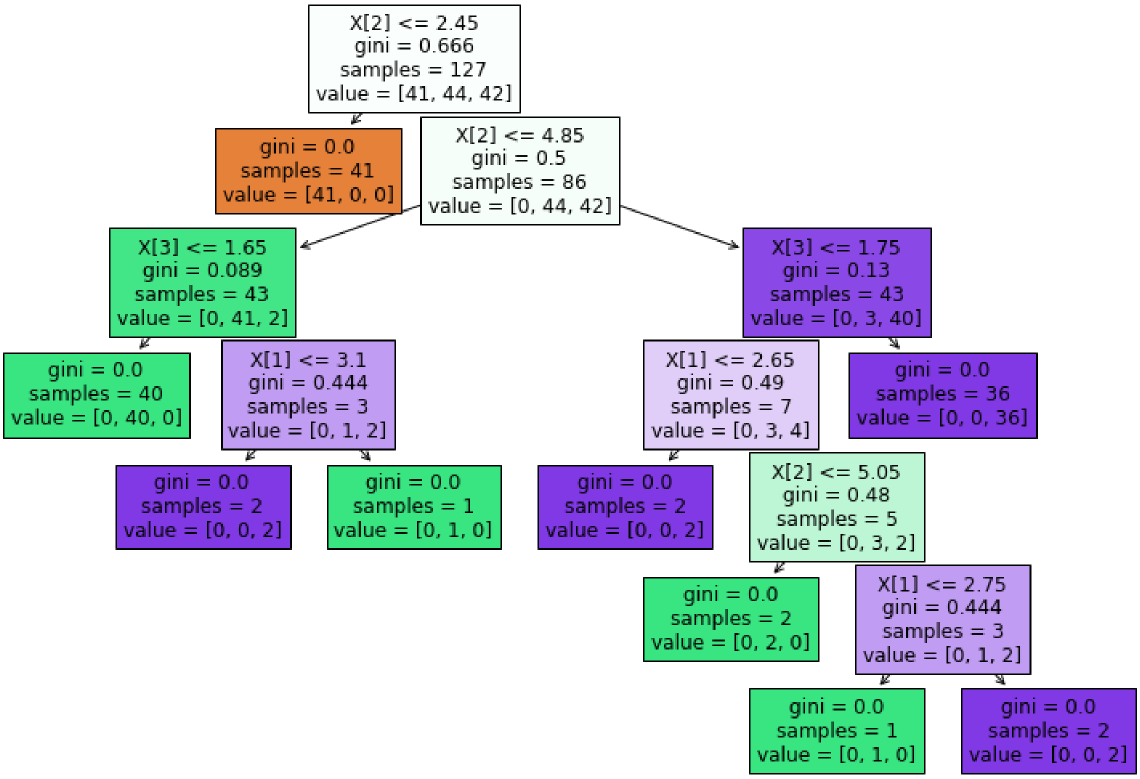
Plot The Tree

In [16]:

```
from sklearn.tree import plot_tree
from matplotlib import pyplot as plt
plt.figure(figsize=(13,9))
plot_tree(decision_tree=dt_model,filled=True)
plt.show
```

Out[16]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:

```

```

7.Model Testing || 8. Model Evaluation

Train data

In [19]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [27]:

```
y_train_pred=dt_model.predict(x_train)
accuracy_score(y_train,y_train_pred)
```

Out[27]:

```
1.0
```

In [28]:

```
confusion_matrix(y_train,y_train_pred)
```

Out[28]:

```
array([[41,  0,  0],
       [ 0, 44,  0],
       [ 0,  0, 42]], dtype=int64)
```

In [ ]:

Test data

In [29]:

```
y_test_pred=dt_model.predict(x_test)
```

In [30]:

```
accuracy_score(y_test,y_test_pred)
```

Out[30]:

```
0.9565217391304348
```

In [31]:

```
confusion_matrix(y_test,y_test_pred)
```

Out[31]:

```
array([[9, 0, 0],
       [0, 5, 1],
       [0, 0, 8]], dtype=int64)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: