LetsGrowMore Task-2

Prediction using Decision Tree Algorithm

import the necesaary libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
df=pd.read_csv("Iris.csv")
df
```

Out[3]:

	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
...	...	...	...	...	...	...
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

150 rows × 6 columns

In [4]:

```
df.head()
```

Out[4]:

	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 [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 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
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:

```
df.tail()
```

Out[6]:

	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

In [7]:

```
df.describe()
```

Out[7]:

	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

In [8]:

```
df.shape
```

Out[8]:

```
(150, 6)
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [10]:

```
df.value_counts()
```

Out[10]:

```
Id   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
1    5.1            3.5           1.4            0.2           Iris-setos
a        1
95   5.6            2.7           4.2            1.3           Iris-versi
color    1
97   5.7            2.9           4.2            1.3           Iris-versi
color    1
98   6.2            2.9           4.3            1.3           Iris-versi
color    1
99   5.1            2.5           3.0            1.1           Iris-versi
color    1

..
51   7.0            3.2           4.7            1.4           Iris-versi
color    1
52   6.4            3.2           4.5            1.5           Iris-versi
color    1
53   6.9            3.1           4.9            1.5           Iris-versi
color    1
54   5.5            2.3           4.0            1.3           Iris-versi
color    1
150  5.9            3.0           5.1            1.8           Iris-virgi
nica     1
Length: 150, dtype: int64
```

In [11]:

```
df.columns
```

Out[11]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidt
hCm',
       'Species'],
      dtype='object')
```

In [12]:

```
df.dtypes
```

Out[12]:

```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [13]:

```
df.corr()
```

Out[13]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

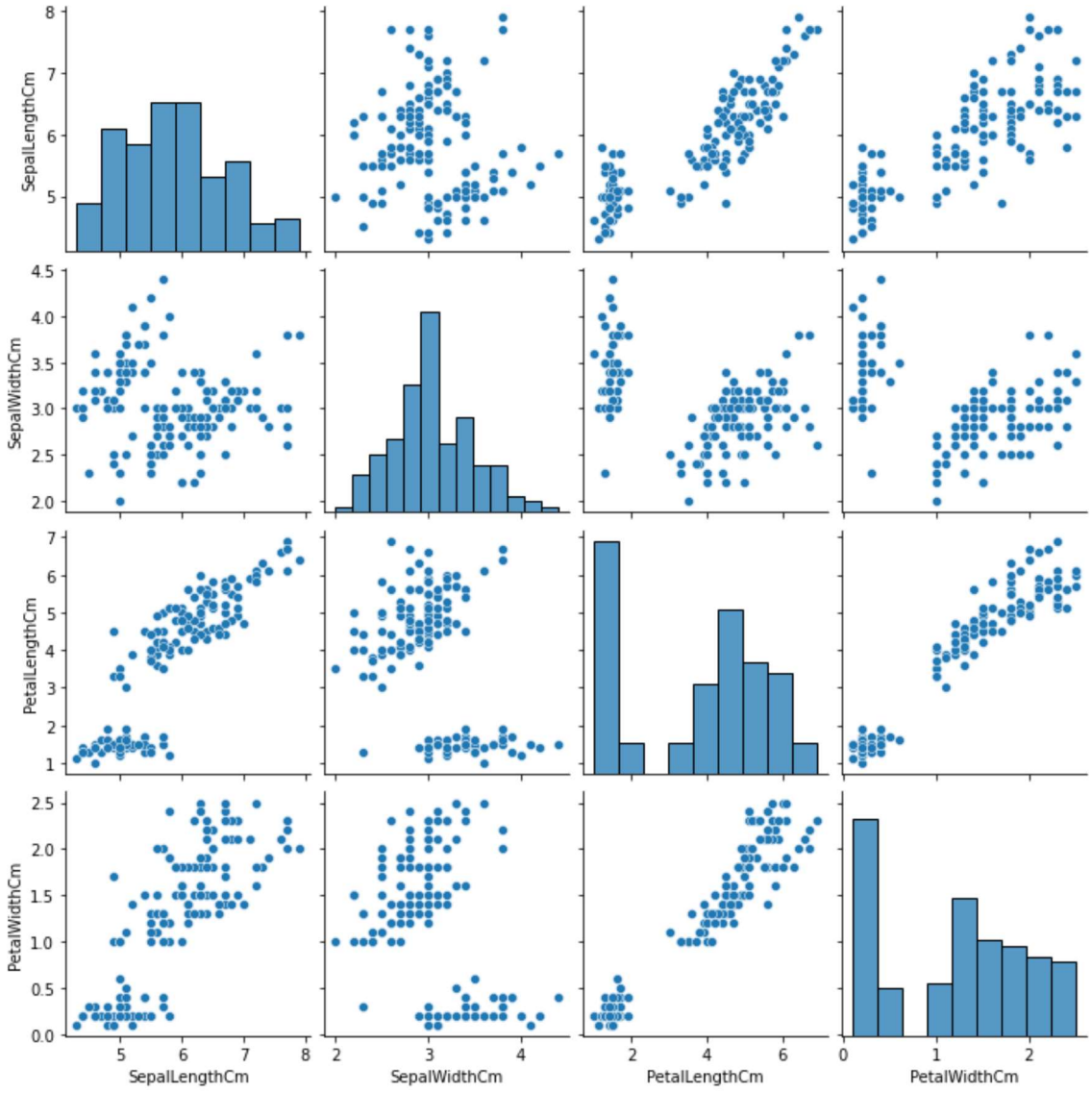
DATA VISUALISATION

In [14]:

```
sns.pairplot(df.iloc[:,1:])
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x1dc2c6a4340>
```

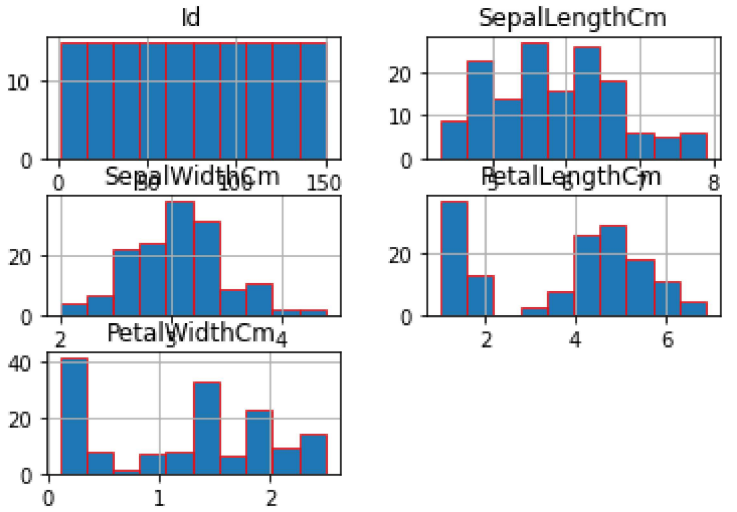


In [15]:

```
df.hist(edgecolor="red",linewidth=0.75)
```

Out[15]:

```
array([[<AxesSubplot:title={'center':'Id'}>,
        <AxesSubplot:title={'center':'SepalLengthCm'}>],
       [<AxesSubplot:title={'center':'SepalWidthCm'}>,
        <AxesSubplot:title={'center':'PetalLengthCm'}>],
       [<AxesSubplot:title={'center':'PetalWidthCm'}>, <AxesSubplot:>]],
      dtype=object)
```

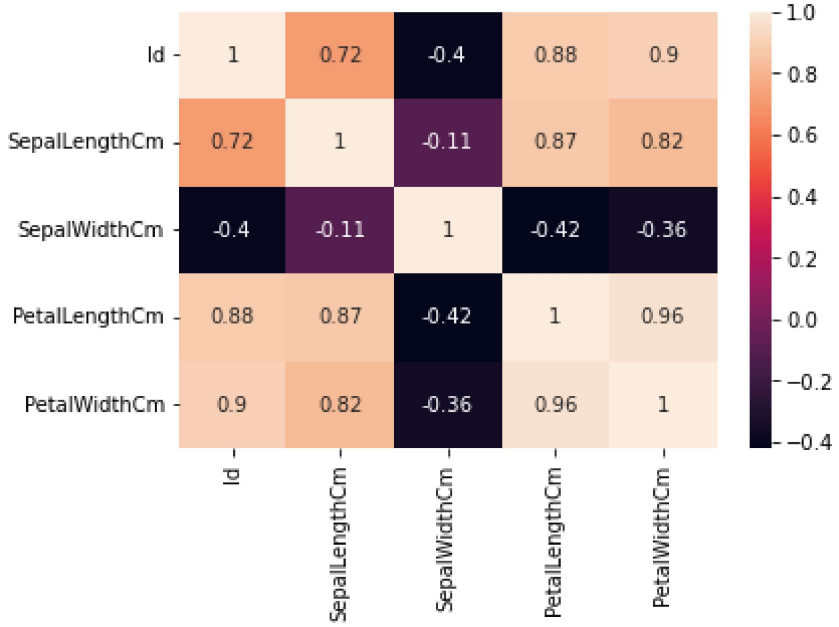


In [16]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[16]:

```
<AxesSubplot:>
```



In [17]:

```
df.dtypes
```

Out[17]:

```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [18]:

```
x=df.iloc[:,:4]
x
```

Out[18]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4
...	...	...	...	...
145	146	6.7	3.0	5.2
146	147	6.3	2.5	5.0
147	148	6.5	3.0	5.2
148	149	6.2	3.4	5.4
149	150	5.9	3.0	5.1

150 rows × 4 columns

In [19]:

```
y=df.iloc[:,-1]
y
```

Out[19]:

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [20]:

```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [21]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.30,random_state=1)
```

In [22]:

```
from sklearn.metrics import classification_report
```

In [23]:

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

```

In [24]:

```
dt=DecisionTreeClassifier()
dt=dt.fit(xtrain,ytrain)
ypred=dt.predict(xtest)
```

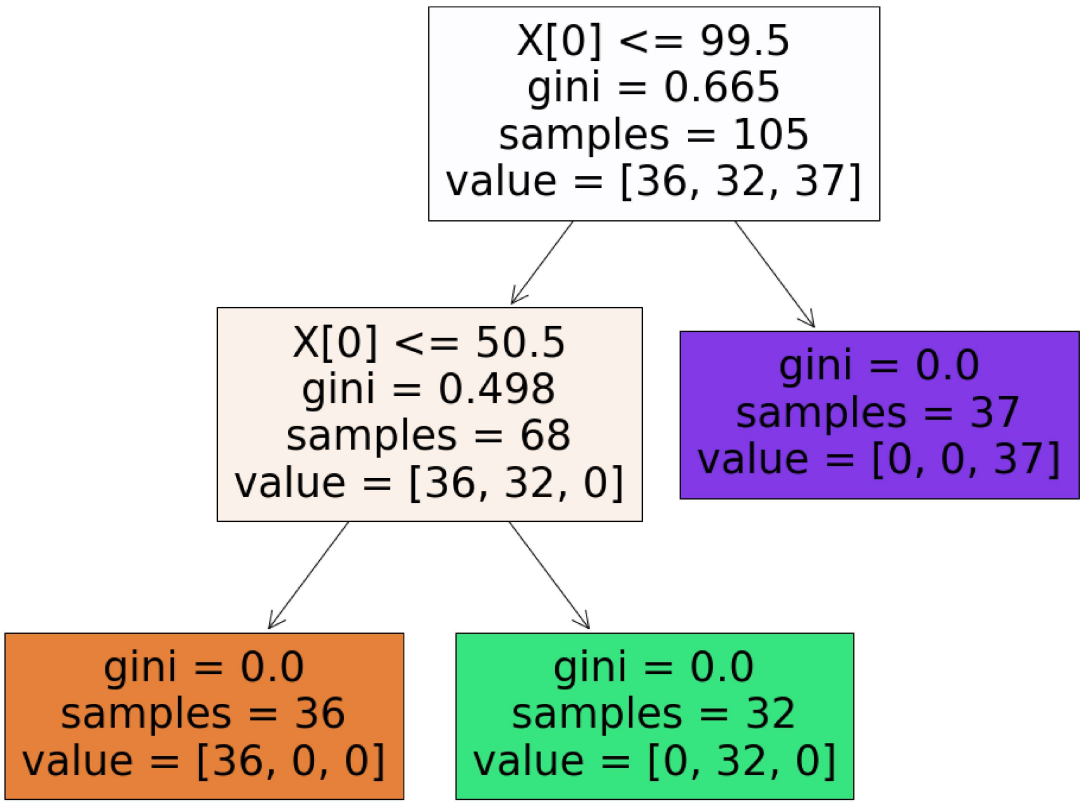
In [25]:

```
result=confusion_matrix(ytest,ypred)
print("Confusion Metrix:")
print(result)
result1=classification_report(ytest,ypred)
print("Classification Report:")
print(result1)
result2=accuracy_score(ytest,ypred)
print("Accuracy:",result2)
```

```
Confusion Metrix:
[[14  0  0]
 [ 0 17  1]
 [ 0  0 13]]
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        18
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45

Accuracy: 0.9777777777777777
```

In [27]:

```
features=df.columns[:-1]
from sklearn.tree import plot_tree
plt.figure(figsize=(17,14))
plot_tree(dt.fit(xtrain,ytrain),filled=True)
plt.show()
```



In [ ]:

```
```

In [ ]:

```
```