LETSGROWMORE

TASK 2 - Prediction using Decision Tree Algorithm

NITIN SINGH RATHORE

```
import pandas as pd import matplotlib.pyplot as plt from sklearn.datasets import load_iris from sklearn.model_selection import train_test_split from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import accuracy_score from sklearn.tree import plot_tree from sklearn import tree
```

In [14]:
```
df = pd.read_csv("D:\\totach\iris.csv")
```

In [15]:
```
df.head()
```

Out[15]:

	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 [16]:
```
df.drop('Id',axis=1,inplace = True)
df.head()
```

Out[16]:

	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

In [17]:
```
df.shape
```

Out[17]:
```
(150, 5)
```

In [18]:
```
df.dtypes
```

Out[18]:
```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [19]:
```
df.isnull().values.any()
```

Out[19]:
```
False
```

In [20]:
```
df.describe()
```

Out[20]:

	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 [21]:
```
df.corr()
```

Out[21]:

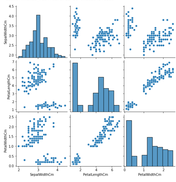
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [22]:
```
df['Species'].value_counts()
```

Out[22]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

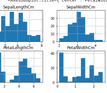
In [23]:
```
sns.pairplot(df.iloc[:,1:])
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x1a09424c100>
```

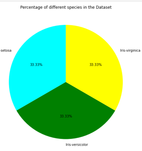


In [24]:
```
df.hist()
```

Out[24]:
```
array([[<AxesSubplot:title={'center':'SepalLengthCm'}>,
        <AxesSubplot:title={'center':'SepalWidthCm'}>],
       [<AxesSubplot:title={'center':'PetalLengthCm'}>,
        <AxesSubplot:title={'center':'PetalWidthCm'}>]], dtype=object)
```



In [25]:
```
fig = plt.figure(figsize=(5, 5))
ax = fig.add_axes([0, 0, 1, 1])
ax.axis('equal')
colors = ['cyan', 'green', 'yellow']
sp = df['Species'].unique()
ct = df['Species'].value_counts().tolist()
ax.pie(ct, labels = sp, autopct='%1.2f%%', colors=colors, startangle=90)
plt.title('Percentage of different species in the Dataset')
plt.show()
```



In [26]:
```
df.skew()
```

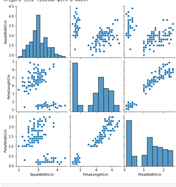
Out[26]:
```
SepalLengthCm    0.314911
SepalWidthCm     0.334053
PetalLengthCm   -0.274464
PetalWidthCm    -0.104997
dtype: float64
```

In [27]:
```
df.duplicated()
```

Out[27]:
```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

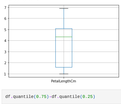
In [28]:
```
plt.figure(figsize=(10,5))
sns.pairplot(df.iloc[:,1:])
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0x1a0943a3d00>
<Figure size 720x360 with 0 Axes>
```



In [29]:
```
df.boxplot(column='PetalLengthCm')
```

Out[29]:
```
<AxesSubplot:>
```



In [30]:
```
df.quantile(0.75)-df.quantile(0.25)
```

Out[30]:
```
SepalLengthCm    1.3
SepalWidthCm     0.5
PetalLengthCm    3.5
PetalWidthCm     1.5
dtype: float64
```

In [31]:
```
df.head()
```

Out[31]:

	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

In [32]:
```
x = df.drop('Species', axis=1)
y = df['Species']
```

In [33]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state=1)
```

In [34]:
```
from sklearn.tree import DecisionTreeClassifier

dTree = DecisionTreeClassifier(criterion = 'entropy', max_depth=5)
dTree.fit(X_train, y_train)
```

Out[34]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [35]:
```
print("Accuracy:", dTree.score(X_test,y_test) * 100)
```

```
Accuracy: 95.55555555555556
```

In [36]:
```
y_pred = dTree.predict(X_test)
print(y_pred)
```

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

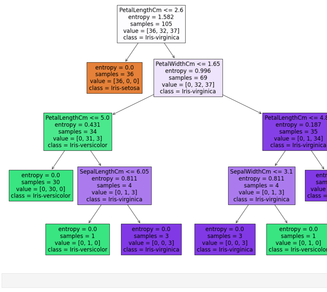
In [37]:
```
features = df.columns[:-1]
classes = df['Species'].unique().tolist()

from sklearn.tree import plot_tree
output = plot_tree(dTree, feature_names=features, class_names=classes, filled=True)
for o in output:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('black')
        arrow.set_linewidth(1)
```

In [38]:
```
dTree.fit(X_train, y_train)
```

Out[38]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [39]:
```
plt.figure(figsize=(20, 15))
output = plot_tree(dTree, feature_names=features, class_names=classes, filled=True)
for o in output:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('black')
        arrow.set_linewidth(1)
```



In [ ]:
```
```