LETSGROWMORE DATA SCIENCE

PREDICTION USING DECESION TREE

PRANJAL JAIN

TASK 1

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
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 [2]:
```
df = pd.read_csv('D:\\terrorism\data2.csv')
```

In [3]:
```
df.head()
```

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

In [4]:
```
df.drop('Id',axis=1,inplace = True)
df.head()
```

Out[4]:

	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 [5]:
```
df.shape
```

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

In [6]:
```
df.dtypes
```

Out[6]:
```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [7]:
```
df.isnull().values.any()
```

Out[7]:
```
False
```

In [8]:
```
df.corr()
```

Out[8]:

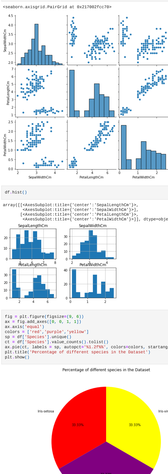
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [9]:
```
df['Species'].value_counts()
```

Out[9]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

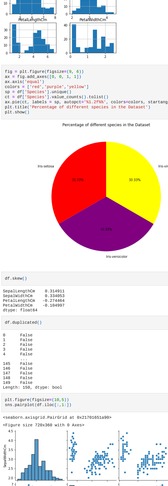
In [10]:
```
sns.pairplot(df.iloc[:,1:])
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x217982fcc70>
```



In [12]:
```
df.hist()
```

Out[13]:
```
array([[<AxesSubplot:title={'center':'SepalLengthCm'}>,
        <AxesSubplot:title={'center':'SepalWidthCm'}>],
       [<AxesSubplot:title={'center':'PetalLengthCm'}>,
        <AxesSubplot:title={'center':'PetalWidthCm'}>]], dtype=object)
```



In [14]:
```
fig = plt.figure(figsize=(6, 6))
ax = fig.add_axes([0, 0, 1, 1])
ax.axis('equal')
colors = ['red','purple','yellow']
sp = df['Species'].unique()
ax.pie(df['Species'].value_counts().tolist()
ax.pie(df['Species'].value_counts(), labels = sp, autopct='%1.2f%%', colors=colors, startangle=90)
plt.title('Percentage of different species in the Dataset')
plt.show()
```


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

Out[15]:
```
SepalLengthCm    0.314911
SepalWidthCm     0.334053
PetalLengthCm   -0.274464
PetalWidthCm    -0.104997
dtype: float64
```

In [16]:
```
df.duplicated()
```

Out[16]:
```
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 [17]:
```
plt.figure(figsize=(10,5))
sns.pairplot(df.iloc[:,1:])
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x217916b1a90>
```


Observation

Sepal Length and Sepal Width are Normally Distributed

Petal Length and Petal Width both are rightly Skewed

In [18]:
```
df.boxplot(column='PetalLengthCm')
```

Out[18]:
```
<AxesSubplot:>
```

Observation

Q1 = 1.7

Q2 = 4.4

Q3 = 5.1

In [19]:
```
df.quantile(0.75)-df.quantile(0.25)
```

Out[19]:
```
SepalLength    1.3
SepalWidth     0.5
PetalLength    3.5
PetalWidth     1.5
dtype: float64
```

Decision Tree

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

Out[20]:

	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 [21]:
```
x = df.drop('Species', axis=1)
y = df['Species']
```

In [22]:
```
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)
```

Build Decision Tree Model

In [23]:
```
from sklearn.tree import DecisionTreeClassifier

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

Out[23]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [24]:
```
print("Accuracy:", dtree.score(X_test,y_test) * 100)
```

```
Accuracy: 95.55555555555556
```

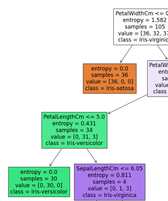
In [25]:
```
y_pred = dtree.predict(X_test)
print(y_pred)
```

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

Plotting the Decision Tree

In [26]:
```
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 [27]:
```
dtree.fit(X_train, y_train)
```

Out[27]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

In [28]:
```
fig = plt.figure(figsize=(26, 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 [ ]:
```

```