Author : Yatam Sumalatha

Task 3 : Prediction using Decision Tree Algorithm

GRIP @ The Sparks Foundation

Step 0: Importing Libraries needed to perform task

In [1]:
```
 #Importing the required Libraries
import numpy as np
import pandas as pd
import sklearn.metrics as sm
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
```

Step 1 : Loading and Reading The Data Set

In [4]:
```
data=pd.read_csv(r'C:\Users\ADMIN\Desktop\Sparks Foundation Internship\Datasets\Iris.csv')
data.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]:
```
data.tail()
```

Out[5]:

	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 [6]:
```
data.columns
```

Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
```

```
PetalWidthCm     0
Species          0
dtype: int64
```

Step 2 : Checking the dataset's information

In [9]:
```
data.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 [11]:
```
data.nunique()
```

Out[11]:
```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

In [12]:
```
data.describe()
```

Out[12]:

	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

Now, let's check for unique classes in the dataset.

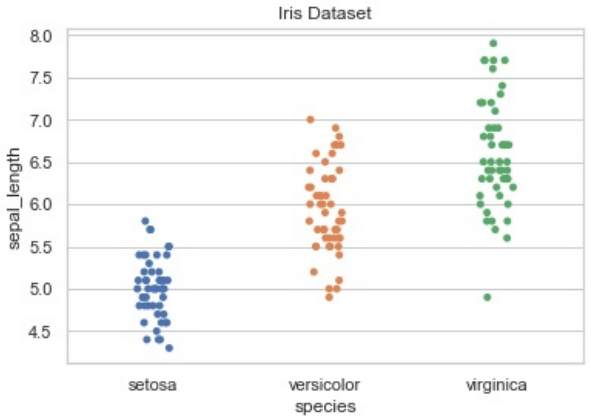
In [13]:
```
print(data.Species.nunique())
print(data.Species.value_counts())
```

```
3
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

Step 3 : Input Data Visualization

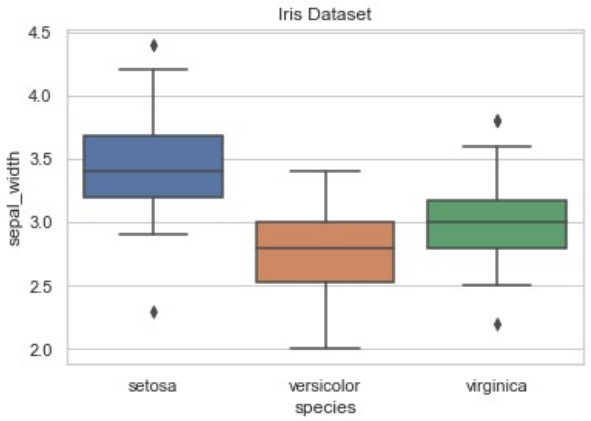
In [14]:
```
sns.set(style = 'whitegrid')
iris = sns.load_dataset('iris');
ax = sns.stripplot(x ='species',y = 'sepal_length',data = iris);
plt.title('Iris Dataset')
```

```
plt.show()
```



In [15]:

```
sns.boxplot(x='species',y='sepal_width',data=iris)
plt.title("Iris Dataset")
plt.show()
```

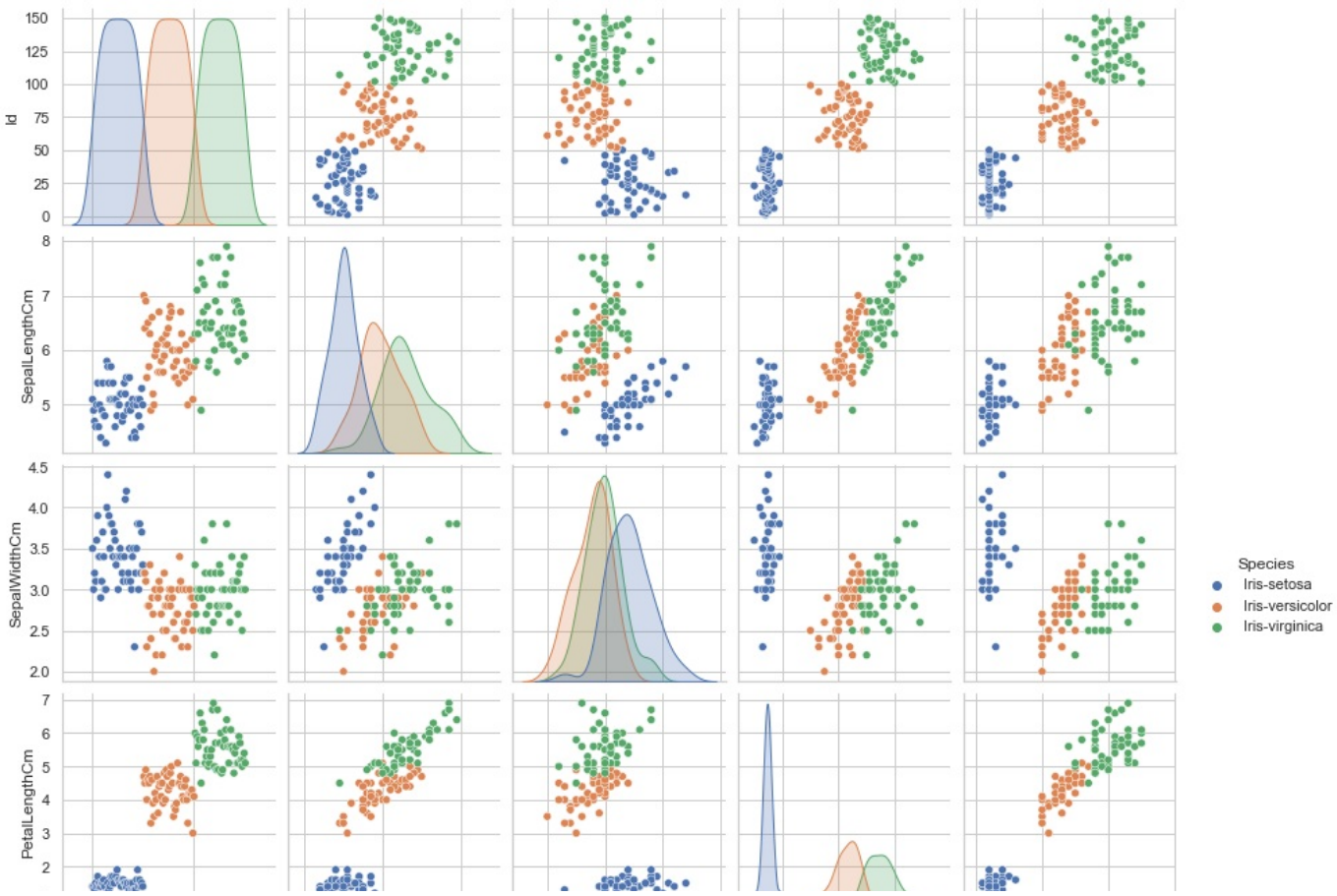


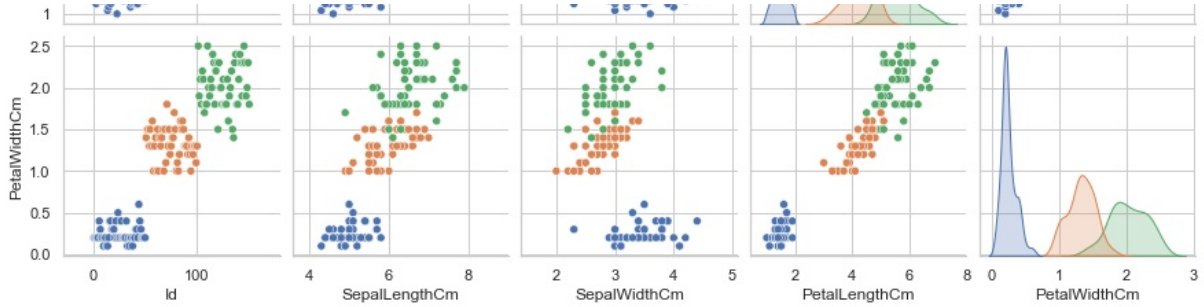
In [16]:

```
sns.pairplot(data, hue='Species')
```

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x2b92adac550>
```





We can observe that speciesv "Iris Setosa" makes a distinctive cluster in every parameter, while other two species overlap a bit each other.

Step 4 : Finding the correlation matrix

In [17]:
```
data.corr()
```

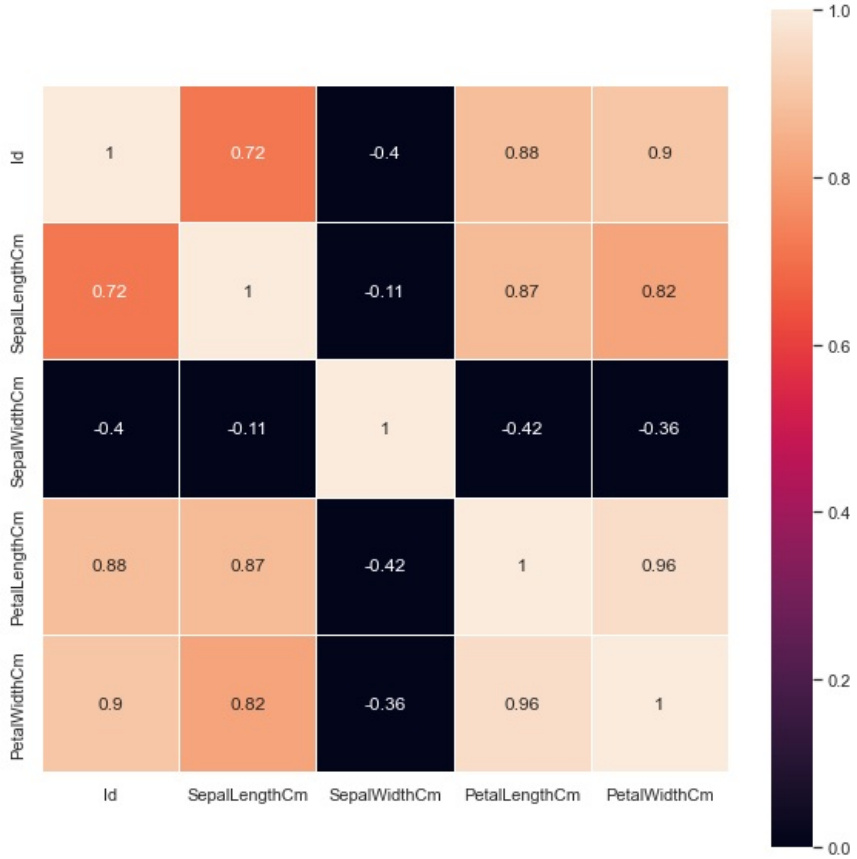
Out[17]:

	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

In next step, using heatmap to visulaize data

In [18]:
```
iris1 = data.corr() #finding correlation between variables of iris dataset
fig,ax=plt.subplots(figsize=(10,10))
sns.heatmap(iris1,vmin=0,vmax=1,square=True,annot=True,linewidth=1)
```

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

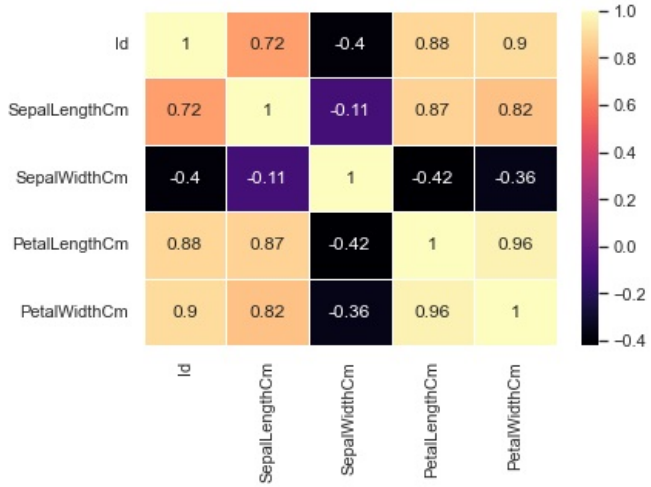


In [19]:
```
sns.heatmap(data.corr(), annot=True, linewidth=1, cmap='magma')
```

```
sns.heatmap(data.corr(), annot=True, linewidth=1, cmap= 'magma' )
```

Out[19]:

```
<AxesSubplot:>
```



We observed that:

Petal length is highly related to petal width. Sepal length is not related to sepal width. Negative correlation of Sepal width with Petal length and Petal Width.

Step 5 : Data preprocessing

In [20]:

```
target=data['Species']
df=data.copy()
df=df.drop('Species', axis=1)
df.shape
```

Out[20]:

```
(150, 5)
```

In [21]:

```
#defining the attributes and labels
X=data.iloc[:, [0,1,2,3]].values
le=LabelEncoder()
data['Species']=le.fit_transform(data['Species'])
y=data['Species'].values
data.shape
```

Out[21]:

```
(150, 6)
```

Step 6 : Trainig the model

We will now split the data into test and train.

In [22]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
print("Traingin split:",X_train.shape)
print("Testin spllit:",X_test.shape)
```

```
Traingin split: (120, 4)
Testin spllit: (30, 4)
```

In [24]:

```
#Defining Decision Tree Algorithm

dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print("Decision Tree Classifier created!")
```

```
Decision Tree Classifier created!
```

Step 7 : Classification Report and Confusion Matrix

In [25]:
```
y_pred=dtree.predict(X_test)
print("Classification report:\n",classification_report(y_test,y_pred))
```

```
Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        11
           2       1.00      1.00      1.00         8

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [26]:
```
acc = sm.accuracy_score(y_test,y_pred)
print("The Accuracy is: {0}%".format(acc*100))
```

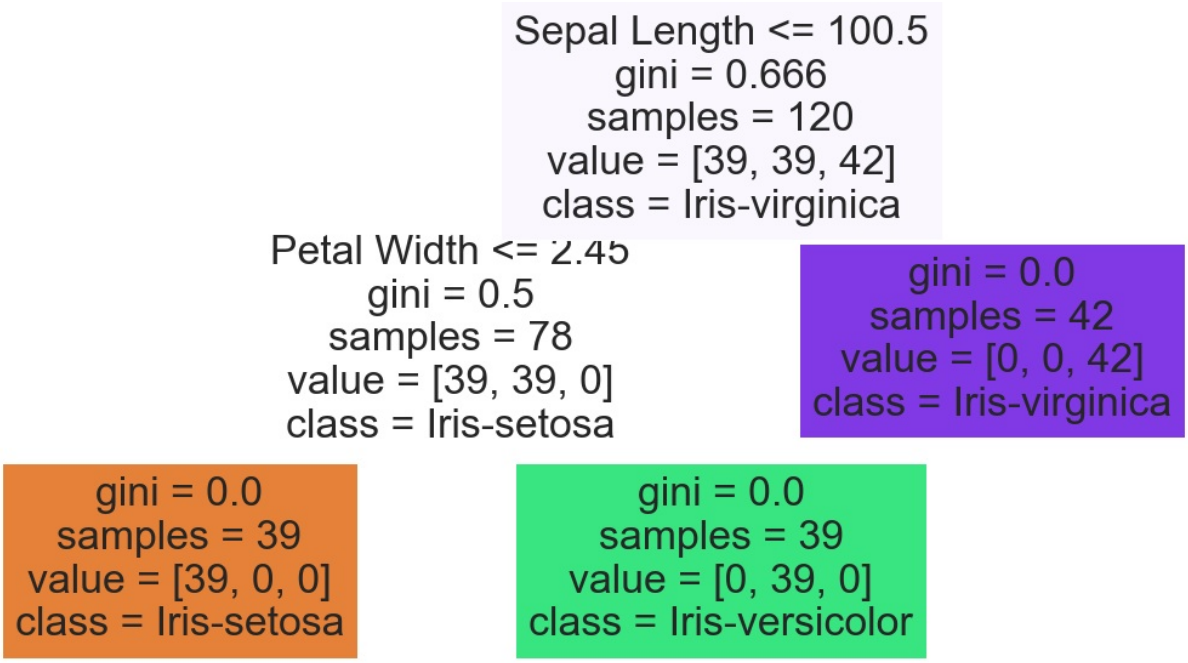
```
The Accuracy is: 100.0%
```

In [27]:
```
#confusion matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[27]:
```
array([[11,  0,  0],
       [ 0, 11,  0],
       [ 0,  0,  8]], dtype=int64)
```

Step 8 : Visualization of Trained Model

In [32]:
```
#visualizing the graph
plt.figure(figsize=(20,10))
# tree=plot_tree(dtree,feature_names=df.columns,precision=2,rounded=True,filled=True,class_names=['Iris-setosa',
feature=['Sepal Length','Sepal Width','Petal Length','Petal Width']
class_name =['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
plot_tree(dtree, filled = True,class_names=class_name,feature_names=feature);
plt.show()
```



Testing for New points except from Dataset

In [29]:

```
Test_point = [[5.4,3.0,4.5,1.5],
              [6.5,2.8,4.6,1.5],
              [5.1,2.5,3.0,1.1],
              [5.1,3.3,1.7,0.5],
              [6.0,2.7,5.1,1.6],
              [6.0,2.2,5.0,1.5]]

print(dtree.predict(Test_point))
```

```
[0 0 0 0 0 0]
```

The Descision Tree Classifier is created and is visaulized graphically. Also the prediction was calculated using decision tree algorithm and accuracy of the model was evaluated.

Thank You

In [ ]: