

Exploring Supervised Machine Learning with Decision Tree

In 'Iris' dataset we have to create the Decision Tree classifier and visualize it graphically. The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

In [1]:

```
from IPython.display import Image
Image(filename='Iris.jpg', height=450, width=400)
```

Out[1]:



In [2]:

```
# Importing libraries in Python

import pandas as pd
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import seaborn as sns
```

In [3]:

```
#Filtering the warnings

import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [4]:

```
iris = datasets.load_iris()
```

In [5]:

```
data=pd.DataFrame(iris['data'],columns=["Petal length","Petal Width","Sepal Length","Sepal Width"])
```

Creating a new column "Species" with target values from the dataset, I am using lambda function to apply a function so as to convert the target values which are 0,1,2 to the corresponding target values("setosa","versicolor","virginica")

In [6]:

```
data['Species']=iris['target']
```

```
In [7]:
```

```
data['Species']=data['Species'].apply(lambda x: iris['target_names'][x])
```

```
In [8]:
```

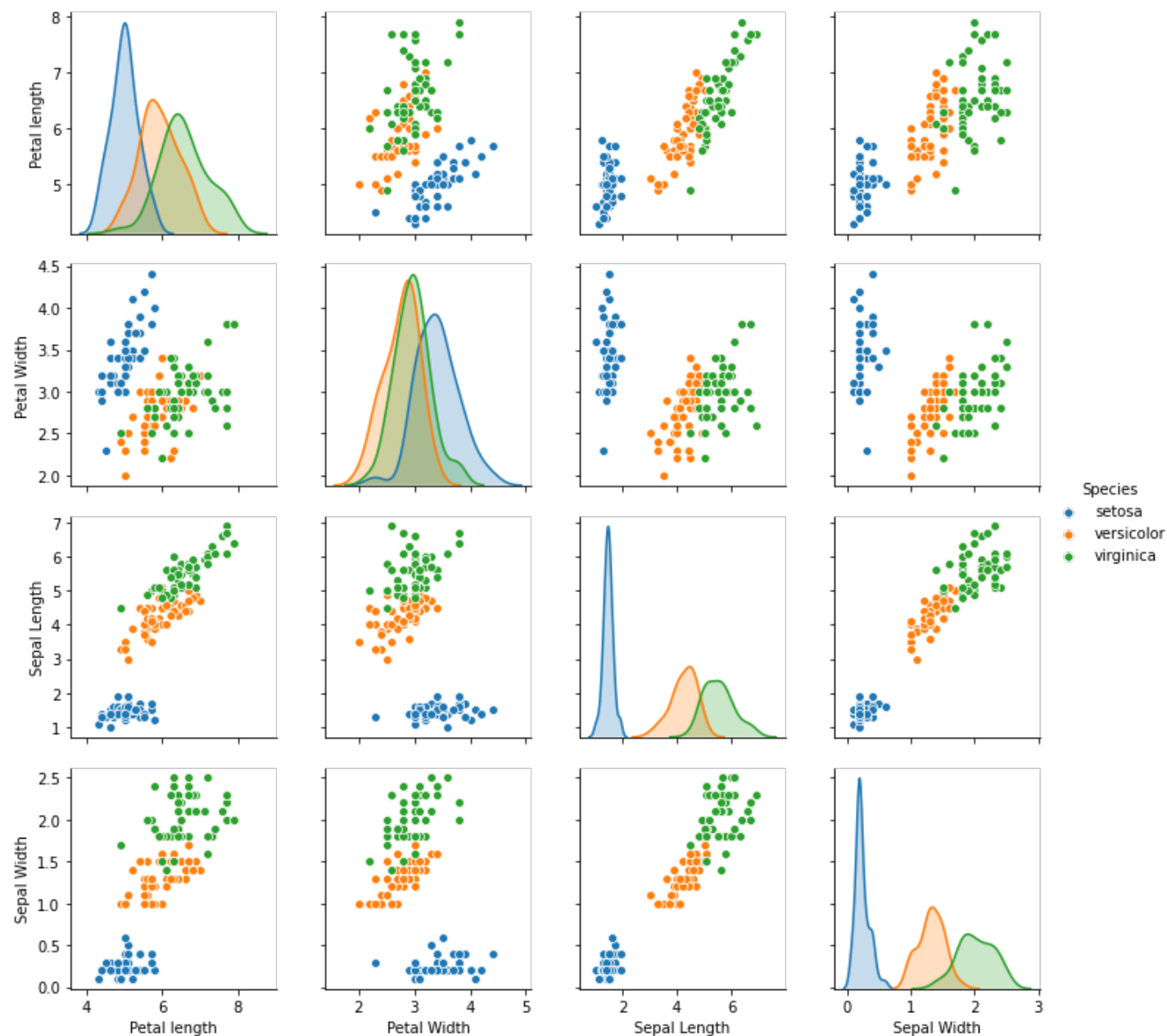
```
data.head()
```

```
Out[8]:
```

| | Petal length | Petal Width | Sepal Length | Sepal 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 |

```
In [9]:
```

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



We can easily observe that "iris-setosa" makes a distinctive cluster in every parameter, while the other two species are overlapping a bit on each other

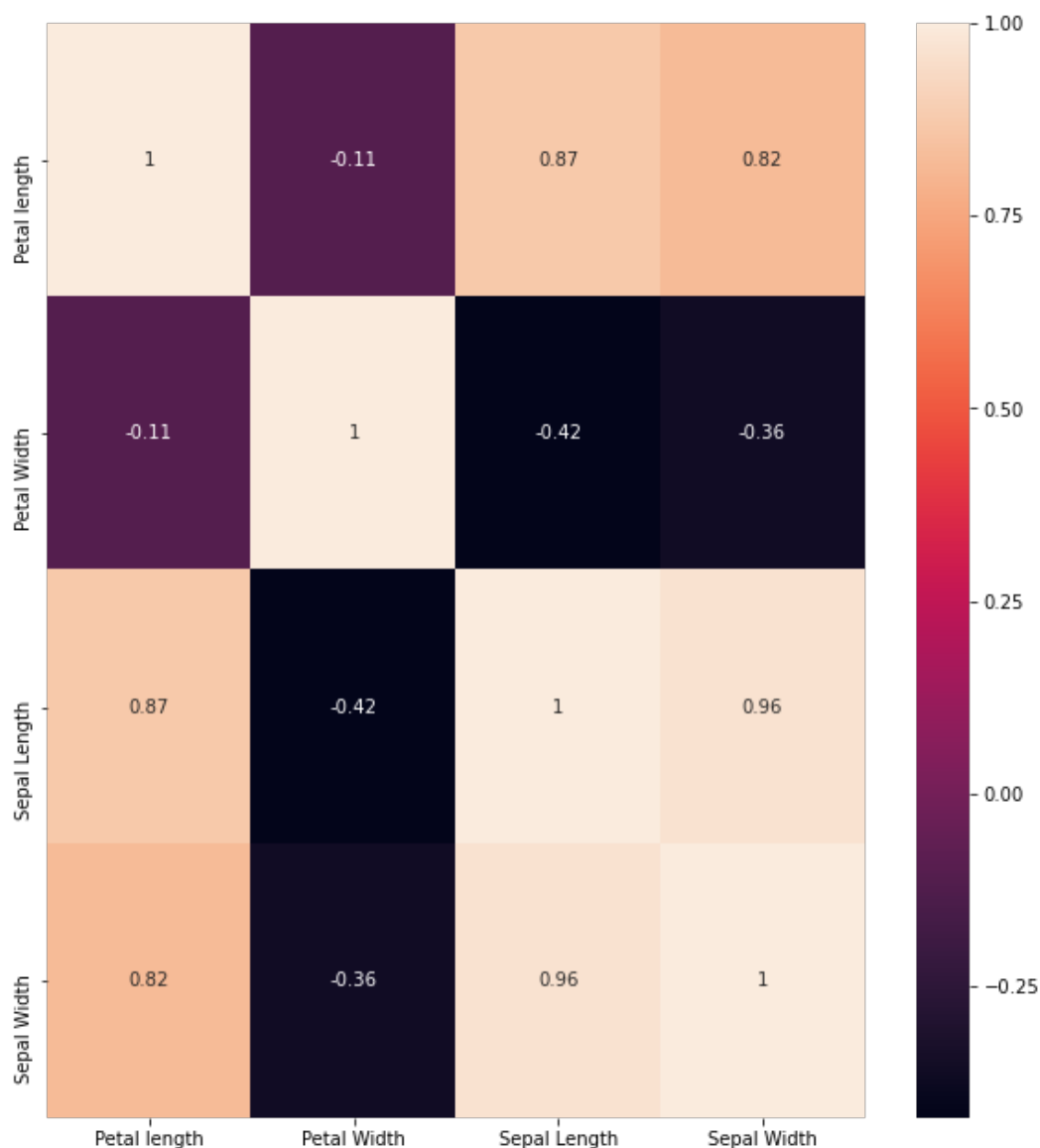
Let's check correlation using heatmap

In [10]:

```
plt.figure(figsize=(10,11))
sns.heatmap(data.corr(),annot=True)
plt.plot()
```

Out[10]:

[]



Observations made -

1. Petal length is highly related to petal width.
2. Sepal length is not related to sepal width

Let's see that how the categorical feature “Species” is distributed with all other four input variables.

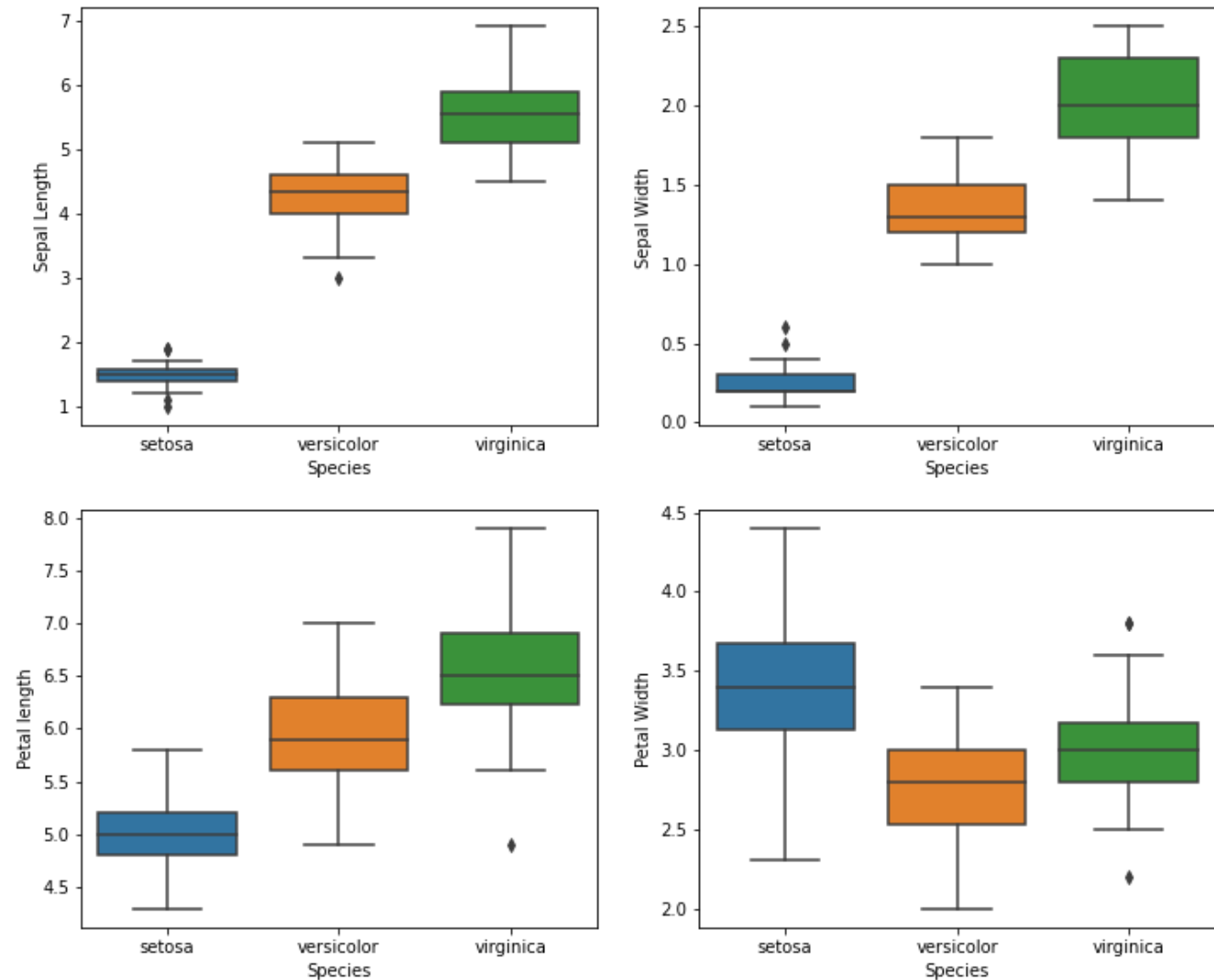
In [11]:

```
plt.figure(figsize=(12,10))
plt.subplot(2,2,1)
sns.boxplot(x="Species",y="Sepal Length",data=data)
plt.subplot(2,2,2)
sns.boxplot(x="Species",y="Sepal Width",data=data)
```

```
plt.subplot(2,2,3)
sns.boxplot(x="Species",y="Petal length",data=data)
plt.subplot(2,2,4)
sns.boxplot(x="Species",y="Petal Width",data=data)
```

Out[11]:

<matplotlib.axes._subplots.AxesSubplot at 0x1ab1ef22160>



Preparing the data for Model Building

In [12]:

```
from sklearn.model_selection import train_test_split

train,test=train_test_split(data,test_size=0.3)
```

In [13]:

```
train_X=train[['Sepal Length',"Sepal Width","Petal length","Petal Width"]]
train_y=train.Species
```

In [14]:

```
test_X=test[['Sepal Length',"Sepal Width","Petal length","Petal Width"]]
test_y=test.Species
```

Importing Decision Tree from Sklearn to build a classification model

In [15]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [16]:

```
DT = DecisionTreeClassifier(random_state=12)
model = DT.fit(train_X, train_y)
```

In [17]:

```
DT.predict(test_X)
```

Out[17]:

```
array(['setosa', 'virginica', 'virginica', 'versicolor', 'setosa',
       'virginica', 'versicolor', 'virginica', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'virginica', 'virginica',
       'setosa', 'versicolor', 'setosa', 'virginica', 'versicolor',
       'setosa', 'setosa', 'setosa', 'setosa', 'virginica', 'setosa',
       'setosa', 'virginica', 'setosa', 'virginica', 'versicolor',
       'versicolor', 'versicolor', 'setosa', 'virginica', 'versicolor',
       'virginica', 'virginica', 'virginica', 'versicolor', 'virginica',
       'versicolor', 'setosa', 'versicolor', 'virginica'], dtype=object)
```

Let's check accuracy of the model build

In [18]:

```
DT.score(test_X, test_y)
```

Out[18]:

```
0.9111111111111111
```

In [19]:

```
#prediction
y_pred = DT.predict(test_X)
```

In [20]:

```
#Accuracy
from sklearn import metrics
print('Accuracy Score:', metrics.accuracy_score(test_y, y_pred))
```

```
Accuracy Score: 0.9111111111111111
```

As the model is now ready, moving ahead for the visualization of the Decision Tree

In [21]:

```
!pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\users\shunbham\anaconda3\lib\site-packages (2.0.2)
```

```
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\shunbham\anaconda3\lib\site-packages (from pydotplus) (2.2.0)
```

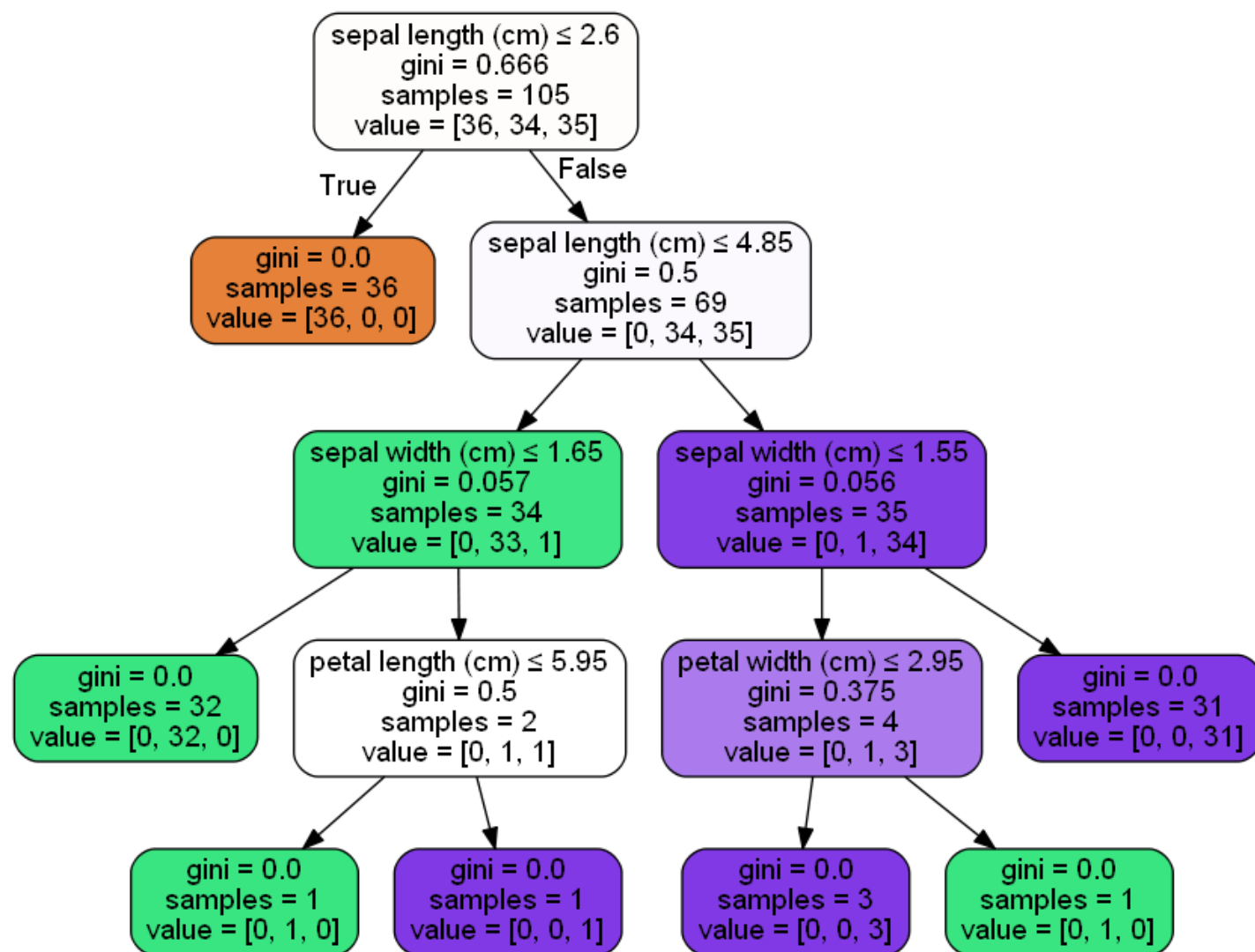
In [22]:

```
# Import necessary libraries for graph viz
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

# Visualize the graph
dot_data = StringIO()
export_graphviz(DT, out_file=dot_data, feature_names=iris.feature_names,
               filled=True, rounded=True,
```

```
special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[22]:



Now feeding any new/test data to this classifier and it would be able to predict the right class accordingly.

In [23]:

```
DT.predict([[3.5, 1.9, 5.2, 2.4]])
```

Out[23]:

```
array(['versicolor'], dtype=object)
```

In [24]:

```
DT.predict([[4.6, 2.9, 3.6, 5.9]])
```

Out[24]:

```
array(['versicolor'], dtype=object)
```

In []: