In [1]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
dataset = sns.load_dataset('iris')
```

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

Out[3]:

	sepal_length	sepal_width	petal_length	petal_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 [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
dataset['species'].value_counts()
```

Out[5]:
```
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

In [6]:
```
x = dataset.iloc[:,0:4]
y = dataset.iloc[:,4]
```

In [7]:
```
x.head()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [8]:
```
y.head()
```

Out[8]:
```
0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: species, dtype: object
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test = train_test_split(x, y ,test_size = 0.25, ran
```

In [10]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(112, 4)
(38, 4)
```

In [11]:
```
print(y_train.shape)
print(y_test.shape)
```

```
(112,)
(38,)
```

Naive Bayes Theorem

In [12]:
```
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()
nb_model.fit(x_train,y_train)
```

Out[12]:
```
GaussianNB()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:
```
y_pred_train = nb_model.predict(x_train)
y_pred_test = nb_model.predict(x_test)
```

Evaluate the model

In [14]:
```
from sklearn.metrics import accuracy_score
```

In [15]:
```
print(accuracy_score(y_train,y_pred_train))
```

```
0.9553571428571429
```

In [16]:
```
print(accuracy_score(y_test,y_pred_test))
```

```
0.9736842105263158
```

Building Decision Tree Model

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:
```
y_pred_dt_train = dt_model.predict(x_train)
y_pred_dt_test = dt_model.predict(x_test)
```

In [22]:
```
print(accuracy_score(y_train,y_pred_dt_train))
print(accuracy_score(y_test,y_pred_dt_test))
```

```
1.0
0.9736842105263158
```

Building Random Forest Model

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(x_train,y_train)
```

Out[23]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
y_pred_rf_train = rf_model.predict(x_train)
y_pred_rf_test = rf_model.predict(x_test)
```

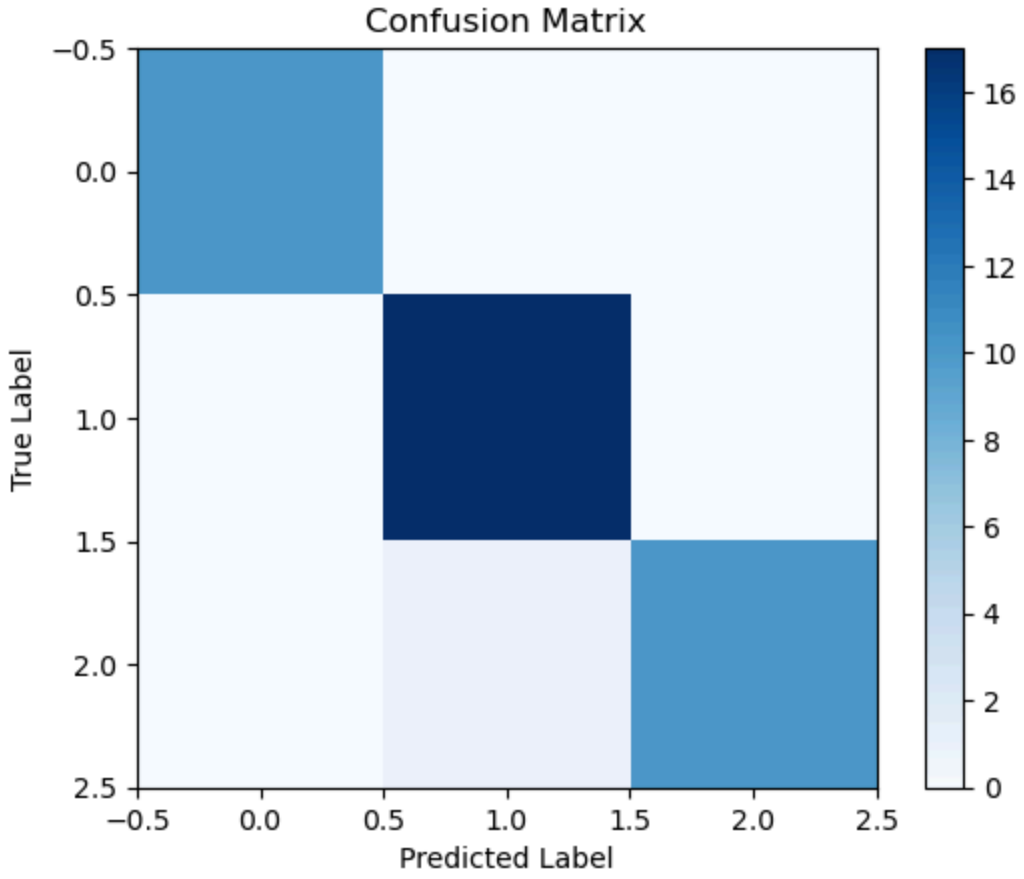
In [25]:
```
print(accuracy_score(y_train,y_pred_rf_train))
print(accuracy_score(y_test,y_pred_rf_test))
```

```
1.0
0.9473684210526315
```

Visualization of Decision Tree Model

In [29]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, dt_model.predict(x_test))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [30]:
```
#Finding important features of the model
dt_model.feature_importances_
#Sepal width of 0 importance, this shows that we can even drop it
#In multilclass we can go for finding out imp features
```

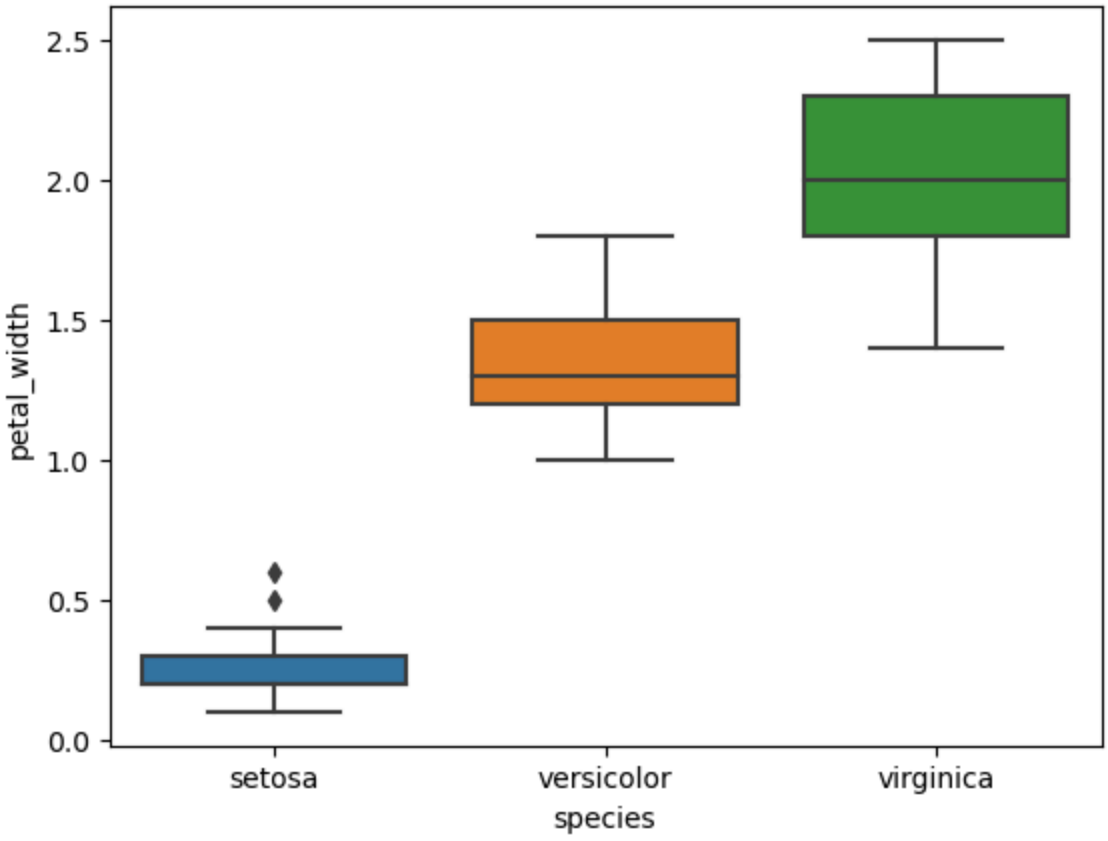
Out[30]:
```
array([0.01791857, 0.        , 0.02906628, 0.95301515])
```

In [31]:
```
pd.DataFrame(index = x.columns,data = dt_model.feature_importances_,columns =
#We can explain imp features to the client looking at the data
```

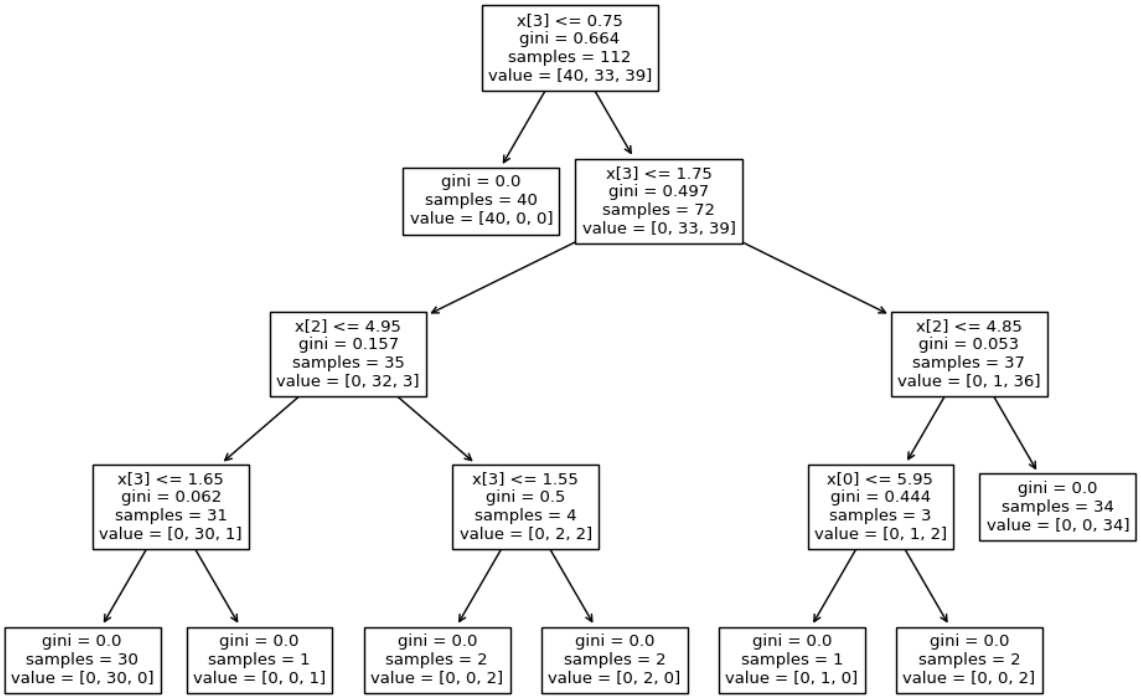
Out[31]:

	Feature Importances
sepal_length	0.017919
sepal_width	0.000000
petal_length	0.029066
petal_width	0.953015

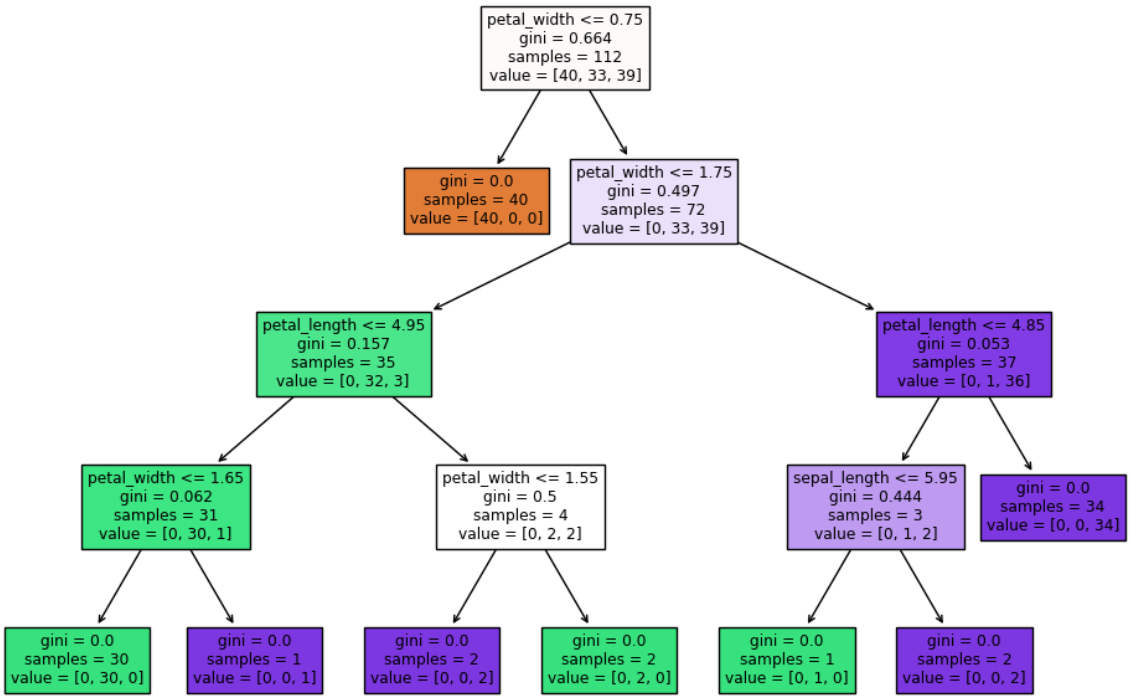
In [32]:
```
sns.boxplot(x = 'species', y  = 'petal_width', data = dataset)
plt.show()
```



In [33]:
```
from sklearn.tree import plot_tree
plt.figure(figsize = (12,8))
plot_tree(dt_model)
plt.show()
#Whenever we do Decision tree, it is imp to do this
```



In [34]:
```
from sklearn.tree import plot_tree
plt.figure(figsize = (12,8))
plot_tree(dt_model, filled = True, feature_names = x.columns)
plt.show()
#colourful Decision tree
```



In [ ]:
```
```