In [ ]:

```
# Maha Ebrahim mohammed al juhdali
# 4051350
# IA8G
# lab 9 : Classification using Decision Trees & Naïve Bayes
```

In [1]:

```
pip install Graphviz
```

```
Collecting Graphviz
  Downloading graphviz-0.19.1-py3-none-any.whl (46 kB)
Installing collected packages: Graphviz
Successfully installed Graphviz-0.19.1
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
import pandas as pd
import graphviz
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
from sklearn .metrics import accuracy_score
seed = 10
```

In [18]:

```
from sklearn import datasets
iris = datasets.load_iris()
df = pd.read_csv (r'C:\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\iris.csv',
                  delimiter =   ',' , header =0 , names = ['sepal length (cm)', 'sepal widt
                                                          'petal length (cm)', 'petal widt

df.head()
```

Out[18]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [19]:

```
le = LabelEncoder()
le.fit(df['Species'].values)
y = le.transform(df['Species'].values)
X = df.drop('Species', axis=1).values
X_train, X_test, y_train ,y_test = train_test_split(X, y, test_size =0.34,stratify=y,random

X_train
```

Out[19]:

```
array([[4.9, 2.5, 4.5, 1.7],
       [6.6, 3. , 4.4, 1.4],
       [5.5, 2.4, 3.8, 1.1],
       [5. , 3.5, 1.6, 0.6],
       [6.9, 3.1, 5.4, 2.1],
       [4.9, 3. , 1.4, 0.2],
       [5.1, 3.8, 1.6, 0.2],
       [5.8, 2.7, 4.1, 1. ],
       [7.7, 2.8, 6.7, 2. ],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3.2, 4.8, 1.8],
       [4.6, 3.6, 1. , 0.2],
       [5.8, 2.7, 3.9, 1.2],
       [5.1, 3.5, 1.4, 0.3],
       [6. , 2.9, 4.5, 1.5],
       [7.7, 2.6, 6.9, 2.3],
       [5.2, 2.7, 3.9, 1.4],
       [6.5, 3.2, 5.1, 2. ],
       [4.9, 3.6, 1.4, 0.1],
       [7. , 3.2, 4.7, 1.4],
       [4.6, 3.4, 1.4, 0.3],
       [7.7, 3. , 6.1, 2.3],
       [5.1, 3.7, 1.5, 0.4],
       [6.6, 2.9, 4.6, 1.3],
       [5. , 3.6, 1.4, 0.2],
       [6.3, 2.3, 4.4, 1.3],
       [5.7, 2.5, 5. , 2. ],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3.7, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [5.7, 2.6, 3.5, 1. ],
       [5. , 3.4, 1.5, 0.2],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3. , 5. , 1.7],
       [4.5, 2.3, 1.3, 0.3],
       [7.1, 3. , 5.9, 2.1],
       [5.2, 3.5, 1.5, 0.2],
       [6.3, 2.7, 4.9, 1.8],
       [4.6, 3.1, 1.5, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5.6, 2.8, 4.9, 2. ],
       [5. , 3.4, 1.6, 0.4],
       [6.3, 2.8, 5.1, 1.5],
       [6.8, 2.8, 4.8, 1.4],
       [5.7, 3.8, 1.7, 0.3],
       [5.4, 3.9, 1.7, 0.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.7, 3.1, 5.6, 2.4],
       [5.6, 2.9, 3.6, 1.3],
       [4.4, 3. , 1.3, 0.2],
       [5.5, 2.3, 4. , 1.3],
       [5.9, 3. , 5.1, 1.8],
       [5.6, 3. , 4.1, 1.3],
       [5. , 3.2, 1.2, 0.2],
       [5.7, 3. , 4.2, 1.2],
       [6.1, 3. , 4.9, 1.8],
       [6.3, 3.4, 5.6, 2.4],
       [7.2, 3.6, 6.1, 2.5],
       [5.5, 2.5, 4. , 1.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [5.6, 3. , 4.5, 1.5],
       [5.7, 2.9, 4.2, 1.3],
       [6.4, 2.8, 5.6, 2.1],
       [6.2, 2.9, 4.3, 1.3],
       [5.4, 3.4, 1.7, 0.2],
       [6.7, 2.5, 5.8, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [6.5, 3. , 5.8, 2.2],
       [4.8, 3. , 1.4, 0.3],
       [6.4, 3.2, 5.3, 2.3],
       [5.8, 2.8, 5.1, 2.4],
       [7.2, 3. , 5.8, 1.6],
       [6.9, 3.2, 5.7, 2.3],
       [7.3, 2.9, 6.3, 1.8],
       [6.2, 2.2, 4.5, 1.5],
       [6. , 3. , 4.8, 1.8],
       [5.1, 3.3, 1.7, 0.5],
       [5.8, 4. , 1.2, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [5.4, 3.4, 1.5, 0.4],
       [5.3, 3.7, 1.5, 0.2],
       [6.7, 3.1, 4.7, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.6, 2.5, 3.9, 1.1],
       [4.7, 3.2, 1.3, 0.2],
       [5.8, 2.6, 4. , 1.2],
       [6.4, 2.7, 5.3, 1.9],
       [7.6, 3. , 6.6, 2.1],
       [5.8, 2.7, 5.1, 1.9],
       [4.8, 3.1, 1.6, 0.2],
       [6.3, 2.9, 5.6, 1.8],
       [5.1, 3.8, 1.5, 0.3],
       [4.3, 3. , 1.1, 0.1],
       [6.7, 3. , 5.2, 2.3]])
```

In [20]:

```
le = LabelEncoder()
le.fit(df['Species'].values)
y = le.transform(df['Species'].values)
X = df.drop('Species', axis=1).values
X_train, X_test, y_train ,y_test = train_test_split(X, y, test_size =0.34,stratify=y,random

y_train
```

Out[20]:

```
array([2, 1, 1, 0, 2, 0, 0, 1, 2, 1, 1, 0, 1, 0, 1, 2, 1, 2, 0, 1, 0, 2,
       0, 1, 0, 1, 2, 1, 0, 0, 1, 0, 1, 1, 0, 2, 0, 2, 0, 0, 2, 0, 2, 1,
       0, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1,
       0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 1, 2, 0, 0, 0, 0, 0, 1, 2, 0, 1, 1,
       0, 1, 2, 2, 2, 0, 2, 0, 0, 2], dtype=int64)
```

In [16]:

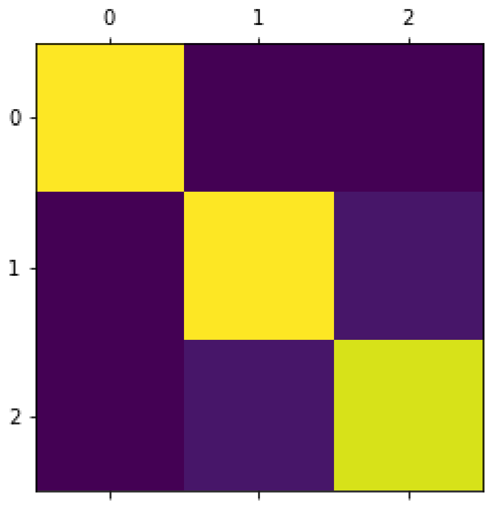
```
tree = DecisionTreeClassifier(criterion='gini',
                              min_samples_leaf=5,
                              min_samples_split=5,
                              max_depth=None,
                              random_state=seed)
tree.fit(X_train, y_train)
y_pred=tree.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
print('DecisionTreeClassifier accuracy score: {}'.format(accuracy))
```

```
DecisionTreeClassifier accuracy score: 0.9615384615384616
```

In [17]:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
print('Confusion Matrix is')
print(confusion_matrix(y_test, y_pred))
cm=confusion_matrix(y_test, y_pred)
plt.matshow(cm)
plt. show()
```

```
Confusion Matrix is
[[17  0  0]
 [ 0 17  1]
 [ 0  1 16]]
```



In [21]:

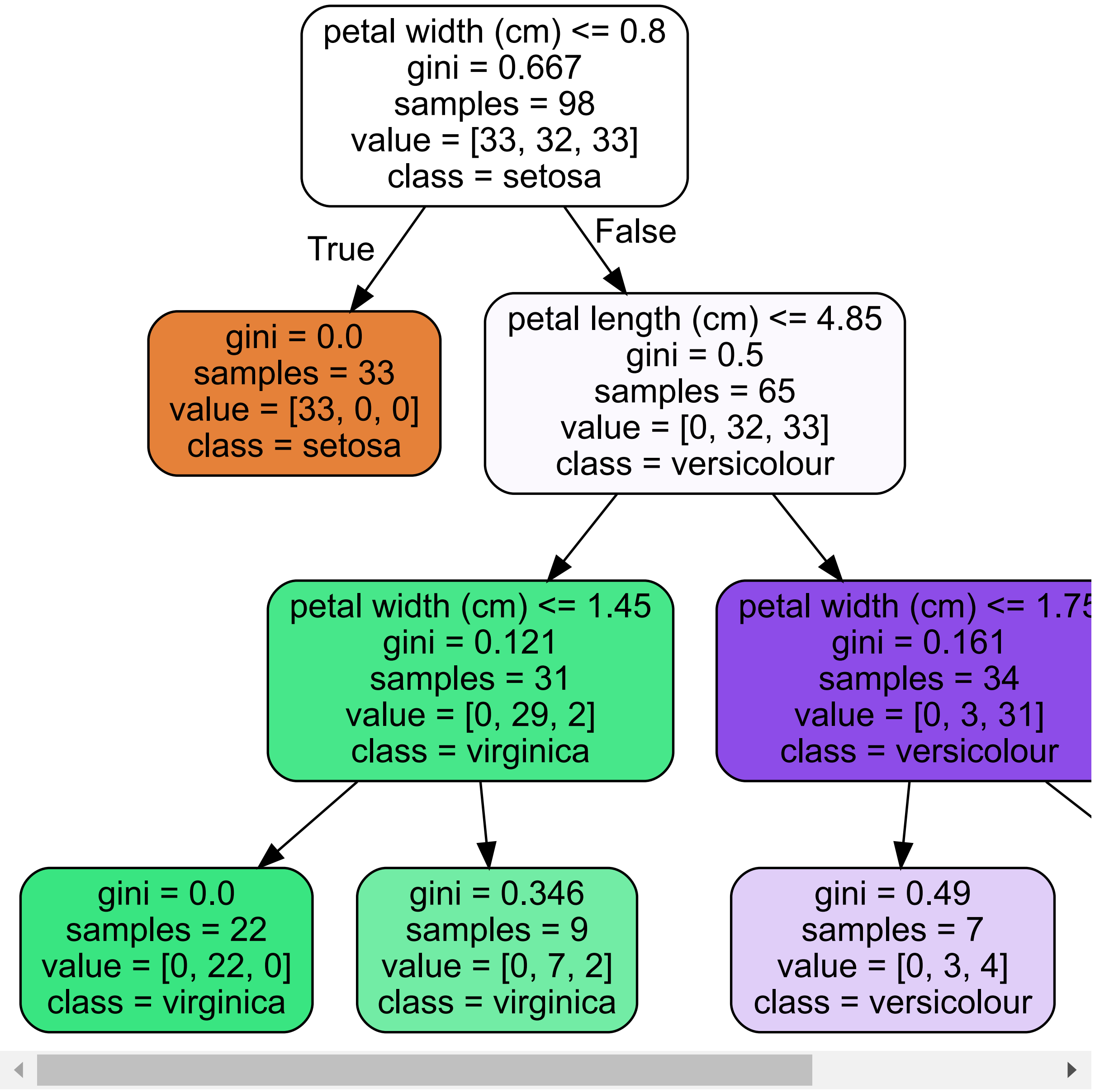
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, labels=df['Species'].unique()))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.94      0.94      0.94        18
           2       0.94      0.94      0.94        17

    accuracy                           0.96        52
   macro avg       0.96      0.96      0.96        52
weighted avg       0.96      0.96      0.96        52
```

In [31]:

```
import os # operating system
os.environ["PATH"]+=os.pathsep +"C:/Program Files/Graphviz/bin/"

def plot_tree(tree, dataframe, label_col, label_encoder, plot_title) :
    label_names=['setosa', 'virginica', 'versicolour']
    graph_data=export_graphviz(tree,
                               feature_names=dataframe.drop(label_col,axis=1).columns,
                               class_names=label_names,
                               filled=True,
                               rounded=True,
                               out_file=None)
    graph=graphviz.Source(graph_data)
    graph.render(plot_title)
    return graph
tree_graph=plot_tree(tree,df, 'Species', le, 'Iris')
tree_graph
```

Out[31]:



In [32]:

```
from sklearn.naive_bayes import GaussianNB, BernoulliNB
```

In [33]:

```
dataframe =[('Rainy','hot','high','false','no'),
            ('Rainy','hot','high','true','no'),
            ('Overcast','hot','high','false','yes'),
            ('sunny','mild','high','false','yes')
           ]
df =pd.DataFrame(dataframe, columns=['outlook','temp','humidity','windy','play golf'])
df
```

Out[33]:

	outlook	temp	humidity	windy	play golf
0	Rainy	hot	high	false	no
1	Rainy	hot	high	true	no
2	Overcast	hot	high	false	yes
3	sunny	mild	high	false	yes

In [73]:

```
outlook_encoded=le.fit_transform(df['outlook'])
print("outlook encoded: ",outlook_encoded)
temp_encoded=le.fit_transform(df['temp'])
print("temp encoded: ",temp_encoded)
play_encoded=le.fit_transform(data['play'])
print("play encoded: ",play_encoded)
```

```
outlook encoded:  [1 1 0 2]
temp encoded:  [0 0 0 1]
play encoded:  [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

In [74]:

```
features=zip(outlook_encoded, temp_encoded)
features_ls = list(features)
print(features_ls)
```

```
[(1, 0), (1, 0), (0, 0), (2, 1)]
```

In [61]:

```
data={'weather': ['Rainy', 'Rainy','Overcast','Sunny','Sunny','Sunny',
                  'Overcast','Rainy','Rainy','Sunny','Rainy','Overcast',
                  'Overcast','Sunny'],
      'temp':['Hot','Hot','Hot','Mild','Cool','Cool','Cool',
              'Mild','Cool','Mild','Mild','Mild','Hot','Mild'],
      'play':['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No

data=pd.DataFrame(data)
data
```

Out[61]:

	weather	temp	play
0	Rainy	Hot	No
1	Rainy	Hot	No
2	Overcast	Hot	Yes
3	Sunny	Mild	Yes
4	Sunny	Cool	Yes
5	Sunny	Cool	No
6	Overcast	Cool	Yes
7	Rainy	Mild	No
8	Rainy	Cool	Yes
9	Sunny	Mild	Yes
10	Rainy	Mild	Yes
11	Overcast	Mild	Yes
12	Overcast	Hot	Yes
13	Sunny	Mild	No

In [68]:

```
weather_encoded=le.fit_transform(data['weather'])
print("weather encoded: ",weather_encoded)
tempp_encoded=le.fit_transform(data['temp'])
print("temp encoded: ",tempp_encoded)
pplay_encoded=le.fit_transform(data['play'])
print("play encoded: ",pplay_encoded)
```

```
weather encoded:  [1 1 0 2 2 2 0 1 1 2 1 0 0 2]
temp encoded:  [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
play encoded:  [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

In [69]:

```
features=zip(weather_encoded, pplay_encoded)
features_lss = list(features)
print(features_lss)
```

```
[(1, 0), (1, 0), (0, 1), (2, 1), (2, 1), (2, 0), (0, 1), (1, 0), (1, 1), (2,
1), (1, 1), (0, 1), (0, 1), (2, 0)]
```

In [71]:

```
from sklearn.naive_bayes import GaussianNB, BernoulliNB
model = BernoulliNB()
model.fit(features_lss, play_encoded)
predicted=model.predict([[0,2]])
print("predicted Value:",predicted)
```

```
predicted Value: [1]
```

In [72]:

```
from sklearn.naive_bayes import GaussianNB, BernoulliNB
model = GaussianNB()
model.fit(features_lss,play_encoded)
predicted=model.predict([[0,2]])
print("predicted Value:",predicted)
```

```
predicted Value: [1]
```

In [ ]: