21BDS0340 - Abhinav Dinesh Srivatsa

In [ ]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [ ]:
```
data = pd.read_csv("drug_data.csv")
data
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [ ]:
```
test = {
    "Age": "31.0 < x <= 45.0",
    "Sex": "M",
    "BP": "LOW",
    "Cholesterol": "HIGH",
    "Na_to_K": "13.9365 < x <= 19.38"
}
test
```

Out[ ]:
```
{'Age': '31.0 < x <= 45.0',
 'Sex': 'M',
 'BP': 'LOW',
 'Cholesterol': 'HIGH',
 'Na_to_K': '13.9365 < x <= 19.38'}
```

In [ ]:
```
def numeric_to_categorical(numeric_data):
    """
    Method to convert a numerical Series to a categorical ranged list. Ranges are deci
    """
```

```
    quartiles = numeric_data.quantile([0.25, 0.5, 0.75])
    q1, q2, q3 = quartiles[0.25], quartiles[0.5], quartiles[0.75]

    new_numeric_data = []
    for row in numeric_data:
        if row <= q1:
            new_numeric_data.append(f"x <= {q1}")
        elif row <= q2:
            new_numeric_data.append(f"{q1} < x <= {q2}")
        elif row <= q3:
            new_numeric_data.append(f"{q2} < x <= {q3}")
        else:
            new_numeric_data.append(f"{q3} < x")

    return new_numeric_data
```

In [ ]:
```
# converting the numerical columns to categorical
data["Age"] = numeric_to_categorical(data["Age"])
data["Na_to_K"] = numeric_to_categorical(data["Na_to_K"])
data
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	x <= 31.0	F	HIGH	HIGH	19.38 < x	drugY
1	45.0 < x <= 58.0	M	LOW	HIGH	10.4455 < x <= 13.9365	drugC
2	45.0 < x <= 58.0	M	LOW	HIGH	x <= 10.4455	drugC
3	x <= 31.0	F	NORMAL	HIGH	x <= 10.4455	drugX
4	58.0 < x	F	LOW	HIGH	13.9365 < x <= 19.38	drugY
...	...	...	...	...	...	...
195	45.0 < x <= 58.0	F	LOW	HIGH	10.4455 < x <= 13.9365	drugC
196	x <= 31.0	M	LOW	HIGH	10.4455 < x <= 13.9365	drugC
197	45.0 < x <= 58.0	M	NORMAL	HIGH	x <= 10.4455	drugX
198	x <= 31.0	M	NORMAL	NORMAL	13.9365 < x <= 19.38	drugX
199	31.0 < x <= 45.0	F	LOW	NORMAL	10.4455 < x <= 13.9365	drugX

200 rows × 6 columns

In [ ]:
```
les = {column: LabelEncoder().fit(data[column]) for column in data.columns}
les
```

Out[ ]:
```
{'Age': LabelEncoder(),
 'Sex': LabelEncoder(),
 'BP': LabelEncoder(),
 'Cholesterol': LabelEncoder(),
 'Na_to_K': LabelEncoder(),
 'Drug': LabelEncoder()}
```

In [ ]:
```
data_transformed = pd.DataFrame()
for column, enc in les.items():
    data_transformed[column] = enc.transform(data[column])
data_transformed
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	3	0	0	0	2	4
1	1	1	1	0	0	2
2	1	1	1	0	3	2
3	3	0	2	0	3	3
4	2	0	1	0	1	4
...	...	...	...	...	...	...
195	1	0	1	0	0	2
196	3	1	1	0	0	2
197	1	1	2	0	3	3
198	3	1	2	1	1	3
199	0	0	1	1	0	3

200 rows × 6 columns

In [ ]:
```
test_transformed = pd.DataFrame()
for column, enc in les.items():
    if column == "Drug":
        continue
    test_transformed[column] = enc.transform([test[column]])
test_transformed
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K
0	0	1	1	0	1

In [ ]:
```
X = data_transformed.drop("Drug", axis=1)
y = data_transformed["Drug"]
X
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K
0	3	0	0	0	2
1	1	1	1	0	0
2	1	1	1	0	3
3	3	0	2	0	3
4	2	0	1	0	1
...	...	...	...	...	...
195	1	0	1	0	0
196	3	1	1	0	0
197	1	1	2	0	3
198	3	1	2	1	1
199	0	0	1	1	0

200 rows × 5 columns

In [ ]:
```
drug_classes = len(les["Drug"].classes_)
drug_classes
```

Out[ ]: 5

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=
X_train
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K
124	1	0	0	1	0
16	2	1	1	1	0
148	2	0	1	1	3
93	0	0	1	1	2
65	2	0	2	1	2
...	...	...	...	...	...
106	3	1	2	0	0
14	1	0	2	0	0
92	3	0	0	0	2
179	2	0	2	0	1
102	3	0	1	0	0

180 rows × 5 columns

In [ ]:
```
def confusion_matrix(y_pred, y_test, classes):
    matrix = np.zeros((classes, classes))
```

```
    for i in range(len(y_pred)):
        pred = y_pred[i]
        real = y_test[i]
        matrix[pred][real] += 1
    sns.heatmap(matrix, cmap="mako", annot=True)
```

In [ ]:

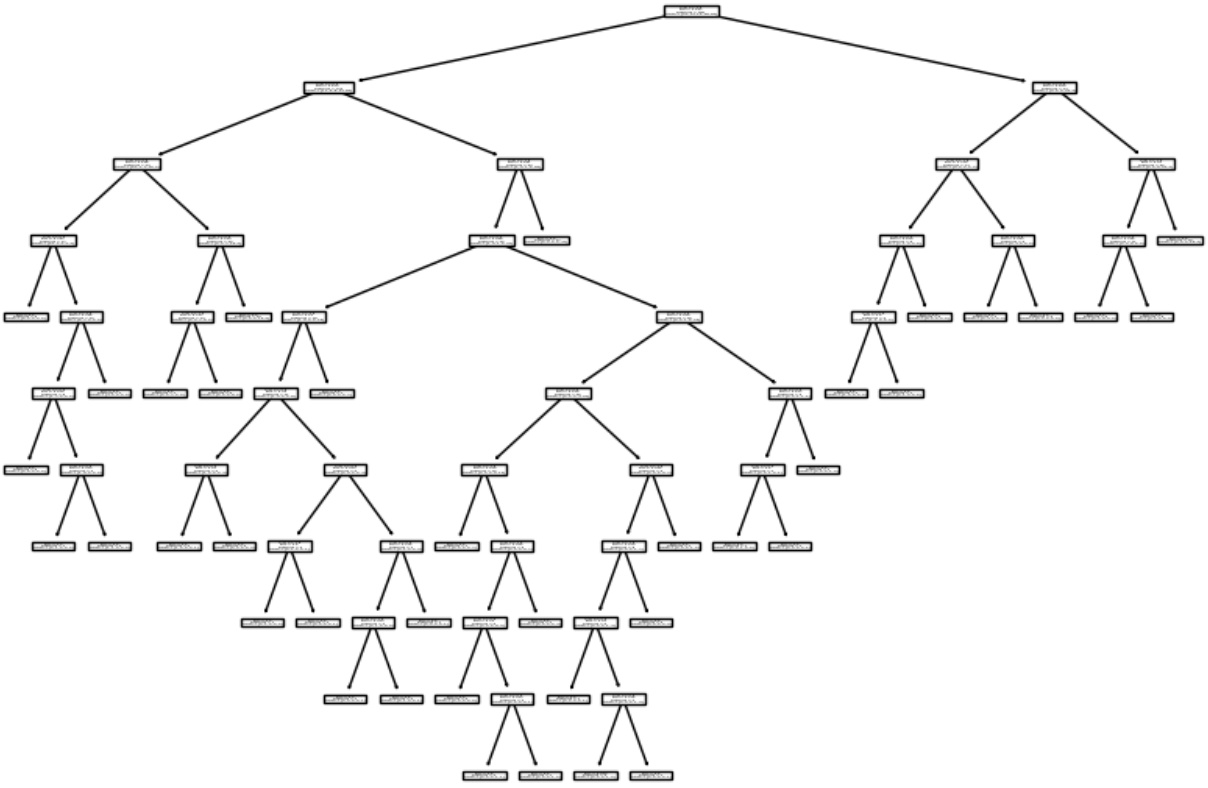
```
# training decision tree classifier
dtc = DecisionTreeClassifier()
dtc = dtc.fit(X_train, y_train)

plt.figure(figsize=(10, 7))
plot_tree(dtc)
```

Out[ ]:
```
[Text(0.5732758620689655, 0.9545454545454546, 'X[4] <= 2.5\ngini = 0.687\nsamples = 1
80\nvalue = [20, 14, 14, 48, 84]'),
 Text(0.27298850574712646, 0.8636363636363636, 'X[4] <= 0.5\ngini = 0.573\nsamples =
136\nvalue = [11, 9, 8, 24, 84]'),
 Text(0.11494252873563218, 0.7727272727272727, 'X[2] <= 0.5\ngini = 0.698\nsamples =
43\nvalue = [10, 7, 7, 19, 0]'),
 Text(0.04597701149425287, 0.6818181818181818, 'X[0] <= 0.5\ngini = 0.484\nsamples =
17\nvalue = [10, 7, 0, 0, 0]'),
 Text(0.022988505747126436, 0.5909090909090909, 'gini = 0.0\nsamples = 5\nvalue = [5,
0, 0, 0, 0]'),
 Text(0.06896551724137931, 0.5909090909090909, 'X[0] <= 2.5\ngini = 0.486\nsamples =
12\nvalue = [5, 7, 0, 0, 0]'),
 Text(0.04597701149425287, 0.5, 'X[3] <= 0.5\ngini = 0.219\nsamples = 8\nvalue = [1,
7, 0, 0, 0]'),
 Text(0.022988505747126436, 0.4090909090909091, 'gini = 0.0\nsamples = 5\nvalue = [0,
5, 0, 0, 0]'),
 Text(0.06896551724137931, 0.4090909090909091, 'X[1] <= 0.5\ngini = 0.444\nsamples =
3\nvalue = [1, 2, 0, 0, 0]'),
 Text(0.04597701149425287, 0.3181818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0,
1, 0, 0, 0]'),
 Text(0.09195402298850575, 0.3181818181818182, 'gini = 0.5\nsamples = 2\nvalue = [1,
1, 0, 0, 0]'),
 Text(0.09195402298850575, 0.5, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0, 0]'),
 Text(0.1839080459770115, 0.6818181818181818, 'X[3] <= 0.5\ngini = 0.393\nsamples = 2
6\nvalue = [0, 0, 7, 19, 0]'),
 Text(0.16091954022988506, 0.5909090909090909, 'X[2] <= 1.5\ngini = 0.497\nsamples =
13\nvalue = [0, 0, 7, 6, 0]'),
 Text(0.13793103448275862, 0.5, 'gini = 0.0\nsamples = 7\nvalue = [0, 0, 7, 0, 0]'),
 Text(0.1839080459770115, 0.5, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0]'),
 Text(0.20689655172413793, 0.5909090909090909, 'gini = 0.0\nsamples = 13\nvalue = [0,
0, 0, 13, 0]'),
 Text(0.43103448275862066, 0.7727272727272727, 'X[4] <= 1.5\ngini = 0.181\nsamples =
93\nvalue = [1, 2, 1, 5, 84]'),
 Text(0.40804597701149425, 0.6818181818181818, 'X[2] <= 0.5\ngini = 0.338\nsamples =
46\nvalue = [1, 2, 1, 5, 37]'),
 Text(0.25287356321839083, 0.5909090909090909, 'X[0] <= 2.5\ngini = 0.29\nsamples = 1
8\nvalue = [1, 2, 0, 0, 15]'),
 Text(0.22988505747126436, 0.5, 'X[0] <= 0.5\ngini = 0.43\nsamples = 11\nvalue = [1,
2, 0, 0, 8]'),
 Text(0.1724137931034483, 0.4090909090909091, 'X[3] <= 0.5\ngini = 0.32\nsamples = 5
\nvalue = [1, 0, 0, 0, 4]'),
 Text(0.14942528735632185, 0.3181818181818182, 'gini = 0.5\nsamples = 2\nvalue = [1,
0, 0, 0, 1]'),
 Text(0.19540229885057472, 0.3181818181818182, 'gini = 0.0\nsamples = 3\nvalue = [0,
0, 0, 0, 3]'),
 Text(0.28735632183908044, 0.4090909090909091, 'X[1] <= 0.5\ngini = 0.444\nsamples =
6\nvalue = [0, 2, 0, 0, 4]'),
 Text(0.2413793103448276, 0.3181818181818182, 'X[0] <= 1.5\ngini = 0.5\nsamples = 2\n
value = [0, 1, 0, 0, 1]'),
 Text(0.21839080459770116, 0.22727272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0,
1, 0, 0, 0]'),
 Text(0.26436781609195403, 0.22727272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.3333333333333333, 0.3181818181818182, 'X[3] <= 0.5\ngini = 0.375\nsamples = 4
\nvalue = [0, 1, 0, 0, 3]'),
 Text(0.3103448275862069, 0.22727272727272727, 'X[0] <= 1.5\ngini = 0.444\nsamples =
3\nvalue = [0, 1, 0, 0, 2]'),
 Text(0.28735632183908044, 0.13636363636363635, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.3333333333333333, 0.13636363636363635, 'gini = 0.5\nsamples = 2\nvalue = [0,
```

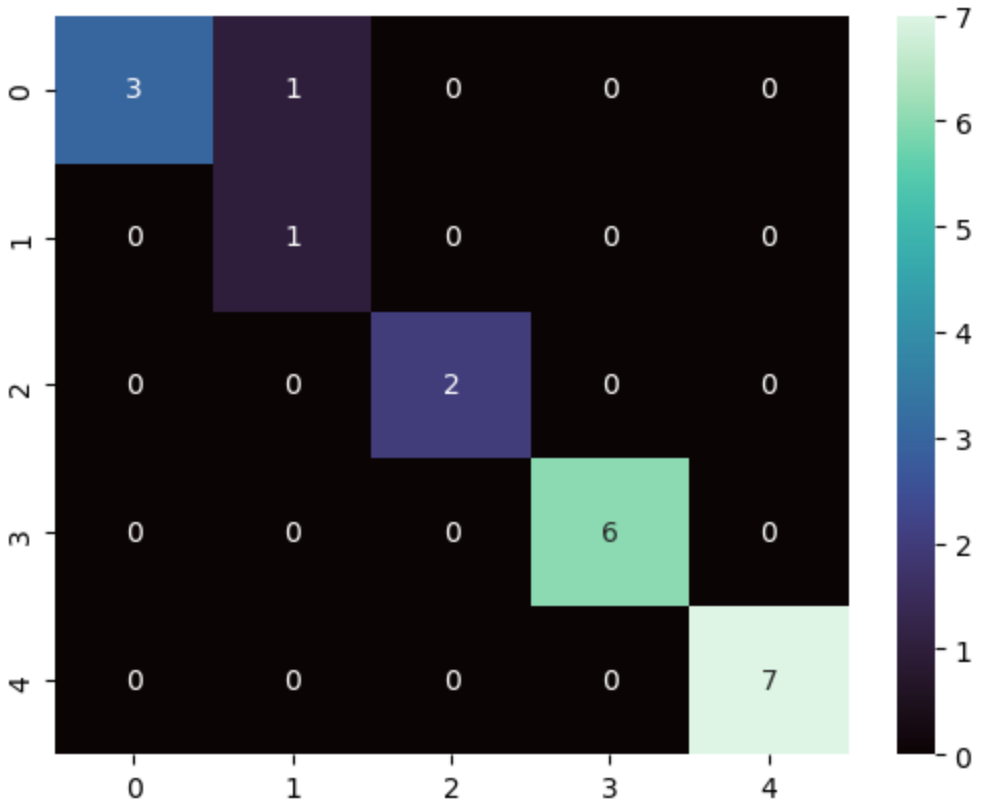
```
1, 0, 0, 1]'),
 Text(0.3563218390804598, 0.22727272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.27586206896551724, 0.5, 'gini = 0.0\nsamples = 7\nvalue = [0, 0, 0, 0, 7]'),
 Text(0.5632183908045977, 0.5909090909090909, 'X[0] <= 2.5\ngini = 0.349\nsamples = 2
8\nvalue = [0, 0, 1, 5, 22]'),
 Text(0.47126436781609193, 0.5, 'X[3] <= 0.5\ngini = 0.269\nsamples = 25\nvalue = [0,
0, 0, 4, 21]'),
 Text(0.40229885057471265, 0.4090909090909091, 'X[2] <= 1.5\ngini = 0.219\nsamples =
16\nvalue = [0, 0, 0, 2, 14]'),
 Text(0.3793103448275862, 0.3181818181818182, 'gini = 0.0\nsamples = 7\nvalue = [0,
0, 0, 0, 7]'),
 Text(0.42528735632183906, 0.3181818181818182, 'X[0] <= 1.5\ngini = 0.346\nsamples =
9\nvalue = [0, 0, 0, 2, 7]'),
 Text(0.40229885057471265, 0.22727272727272727, 'X[0] <= 0.5\ngini = 0.48\nsamples =
5\nvalue = [0, 0, 0, 2, 3]'),
 Text(0.3793103448275862, 0.13636363636363635, 'gini = 0.0\nsamples = 2\nvalue = [0,
0, 0, 0, 2]'),
 Text(0.42528735632183906, 0.13636363636363635, 'X[1] <= 0.5\ngini = 0.444\nsamples =
3\nvalue = [0, 0, 0, 2, 1]'),
 Text(0.40229885057471265, 0.045454545454545456, 'gini = 0.5\nsamples = 2\nvalue =
[0, 0, 0, 1, 1]'),
 Text(0.4482758620689655, 0.045454545454545456, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 1, 0]'),
 Text(0.4482758620689655, 0.22727272727272727, 'gini = 0.0\nsamples = 4\nvalue = [0,
0, 0, 0, 4]'),
 Text(0.5402298850574713, 0.4090909090909091, 'X[0] <= 1.5\ngini = 0.346\nsamples = 9
\nvalue = [0, 0, 0, 2, 7]'),
 Text(0.5172413793103449, 0.3181818181818182, 'X[2] <= 1.5\ngini = 0.219\nsamples = 8
\nvalue = [0, 0, 0, 1, 7]'),
 Text(0.4942528735632184, 0.22727272727272727, 'X[1] <= 0.5\ngini = 0.32\nsamples = 5
\nvalue = [0, 0, 0, 1, 4]'),
 Text(0.47126436781609193, 0.13636363636363635, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.5172413793103449, 0.13636363636363635, 'X[0] <= 0.5\ngini = 0.375\nsamples =
4\nvalue = [0, 0, 0, 1, 3]'),
 Text(0.4942528735632184, 0.045454545454545456, 'gini = 0.444\nsamples = 3\nvalue =
[0, 0, 0, 1, 2]'),
 Text(0.5402298850574713, 0.045454545454545456, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.5402298850574713, 0.22727272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0,
0, 0, 0, 3]'),
 Text(0.5632183908045977, 0.3181818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 1, 0]'),
 Text(0.6551724137931034, 0.5, 'X[3] <= 0.5\ngini = 0.667\nsamples = 3\nvalue = [0,
0, 1, 1, 1]'),
 Text(0.632183908045977, 0.4090909090909091, 'X[1] <= 0.5\ngini = 0.5\nsamples = 2\nv
alue = [0, 0, 1, 0, 1]'),
 Text(0.6091954022988506, 0.3181818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 1, 0, 0]'),
 Text(0.6551724137931034, 0.3181818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 0, 1]'),
 Text(0.6781609195402298, 0.4090909090909091, 'gini = 0.0\nsamples = 1\nvalue = [0,
0, 0, 1, 0]'),
 Text(0.4540229885057471, 0.6818181818181818, 'gini = 0.0\nsamples = 47\nvalue = [0,
0, 0, 0, 47]'),
 Text(0.8735632183908046, 0.8636363636363636, 'X[2] <= 0.5\ngini = 0.629\nsamples = 4
4\nvalue = [9, 5, 6, 24, 0]'),
 Text(0.7931034482758621, 0.7727272727272727, 'X[0] <= 1.5\ngini = 0.459\nsamples = 1
4\nvalue = [9, 5, 0, 0, 0]'),
```

```
 Text(0.7471264367816092, 0.6818181818181818, 'X[1] <= 0.5\ngini = 0.219\nsamples = 8
\nvalue = [7, 1, 0, 0, 0]'),
 Text(0.7241379310344828, 0.5909090909090909, 'X[0] <= 0.5\ngini = 0.5\nsamples = 2\n
value = [1, 1, 0, 0, 0]'),
 Text(0.7011494252873564, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0, 0]'),
 Text(0.7471264367816092, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0, 0]'),
 Text(0.7701149425287356, 0.5909090909090909, 'gini = 0.0\nsamples = 6\nvalue = [6,
0, 0, 0, 0]'),
 Text(0.8390804597701149, 0.6818181818181818, 'X[0] <= 2.5\ngini = 0.444\nsamples = 6
\nvalue = [2, 4, 0, 0, 0]'),
 Text(0.8160919540229885, 0.5909090909090909, 'gini = 0.0\nsamples = 4\nvalue = [0,
4, 0, 0, 0]'),
 Text(0.8620689655172413, 0.5909090909090909, 'gini = 0.0\nsamples = 2\nvalue = [2,
0, 0, 0, 0]'),
 Text(0.9540229885057471, 0.7727272727272727, 'X[2] <= 1.5\ngini = 0.32\nsamples = 30
\nvalue = [0, 0, 6, 24, 0]'),
 Text(0.9310344827586207, 0.6818181818181818, 'X[3] <= 0.5\ngini = 0.496\nsamples = 1
1\nvalue = [0, 0, 6, 5, 0]'),
 Text(0.9080459770114943, 0.5909090909090909, 'gini = 0.0\nsamples = 6\nvalue = [0,
0, 6, 0, 0]'),
 Text(0.9540229885057471, 0.5909090909090909, 'gini = 0.0\nsamples = 5\nvalue = [0,
0, 0, 5, 0]'),
 Text(0.9770114942528736, 0.6818181818181818, 'gini = 0.0\nsamples = 19\nvalue = [0,
0, 0, 19, 0]')]
```



In [ ]:
```
# checking decision tree accuracy
predicted_test = dtc.predict(X_test)
print(f"Score: {dtc.score(X_test, y_test)}")
confusion_matrix(predicted_test, y_test.values, drug_classes)
```

```
Score: 0.95
```



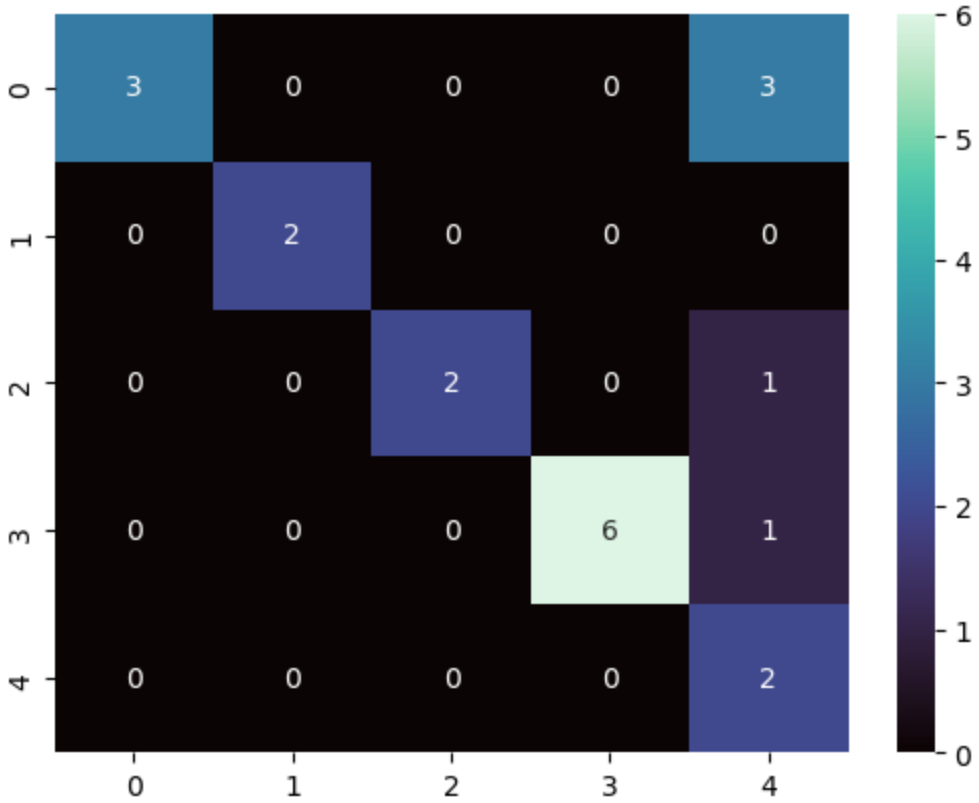
In [ ]:
```
# prediction of non labelled data
predicted_enc = dtc.predict(test_transformed)
predicted = les["Drug"].inverse_transform(predicted_enc)[0]
predicted
```

Out[ ]:
```
'drugY'
```

In [ ]:
```
# training naive bayes classifier
gnb = GaussianNB()
gnb = gnb.fit(X_train, y_train)
```

In [ ]:
```
# checking naive bayes classifier accuracy
predicted_test = gnb.predict(X_test)
print(f"Score: {gnb.score(X_test, y_test)}")
confusion_matrix(predicted_test, y_test.values, drug_classes)
```

```
Score: 0.75
```



In [ ]:
```
print(f"Score of Decision Tree: {dtc.score(X_test, y_test)}")
print(f"Score of Gaussian Naive Bayes: {gnb.score(X_test, y_test)}")
```

```
Score of Decision Tree: 0.95
Score of Gaussian Naive Bayes: 0.75
```