codes with explanation

```
In [1]:
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
import pandas as pd
import seaborn as sns
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
from sklearn.metrics import confusion_matrix
```

1- import required libraries for processing pandas, pre-processing, and plotting graphs.

```
In [2]:
df = pd.read_csv (r'C:\Users\hp\Downloads\drug200.csv')
print (df)
```

| | 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 x 6 columns]

2- we upload our data  which is the drugs dataset  to read the dataset for testing and analysis of the problem.

[7]:
```
df.shape
```

[7]:
```
(200, 6)
```

[8]:
```
df.head()
```

[8]:

| | 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 |

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

3-we use pandas for the shape of the dataset which return the number of rows and columns. df.head() to return the first 5 rows of the dataset, df.info() to print information.

```
In [12]: df.isnull()
```

Out[12]:

| | Age | Sex | BP | Cholesterol | Na_to_K | Drug |
|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False |

4- replace values with Boolean type True for Null values and False with otherwise to check and manage Null values.

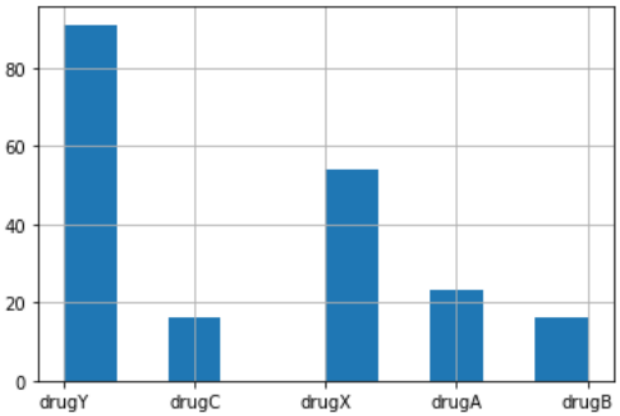
```
In [14]: # Descriptive Statistics
df.describe(include="all")
```

Out[14]:

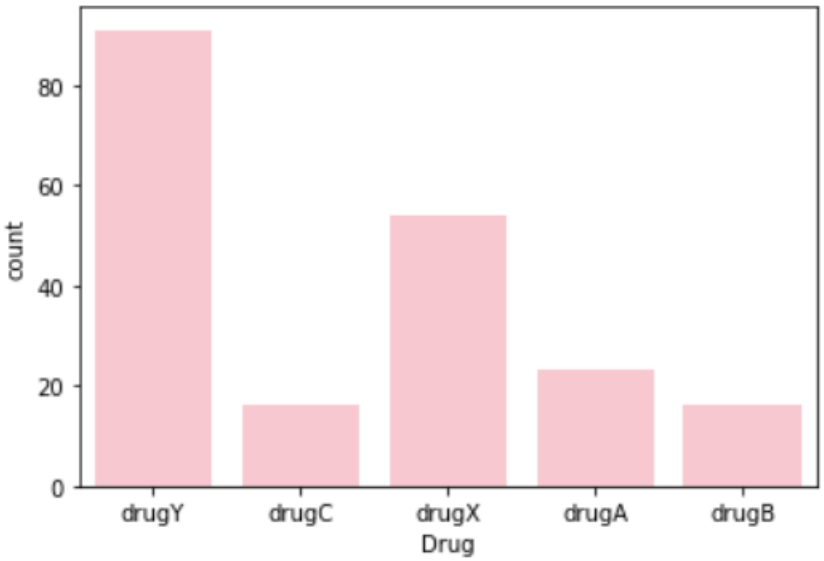
| | Age | Sex | BP | Cholesterol | Na_to_K | Drug |
|---|---|---|---|---|---|---|
| **count** | 200.000000 | 200 | 200 | 200 | 200.000000 | 200 |
| **unique** | NaN | 2 | 3 | 2 | NaN | 5 |
| **top** | NaN | M | HIGH | HIGH | NaN | drugY |
| **freq** | NaN | 104 | 77 | 103 | NaN | 91 |
| **mean** | 44.315000 | NaN | NaN | NaN | 16.084485 | NaN |
| **std** | 16.544315 | NaN | NaN | NaN | 7.223956 | NaN |
| **min** | 15.000000 | NaN | NaN | NaN | 6.269000 | NaN |
| **25%** | 31.000000 | NaN | NaN | NaN | 10.445500 | NaN |
| **50%** | 45.000000 | NaN | NaN | NaN | 13.936500 | NaN |
| **75%** | 58.000000 | NaN | NaN | NaN | 19.380000 | NaN |
| **max** | 74.000000 | NaN | NaN | NaN | 38.247000 | NaN |

5- returns a description of summary statistics. contains this information for each column: count - The number of not-empty values. mean - The average (mean) value. std - The standard deviation.

```
n [15]: print(df.shape)
        print(df.columns)
        df['Drug'].hist()

        (200, 6)
        Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
ut[15]: <AxesSubplot:>
```




```
In [17]: #countplot
         sns.countplot(x = df['Drug'], color= 'pink')

Out[17]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```

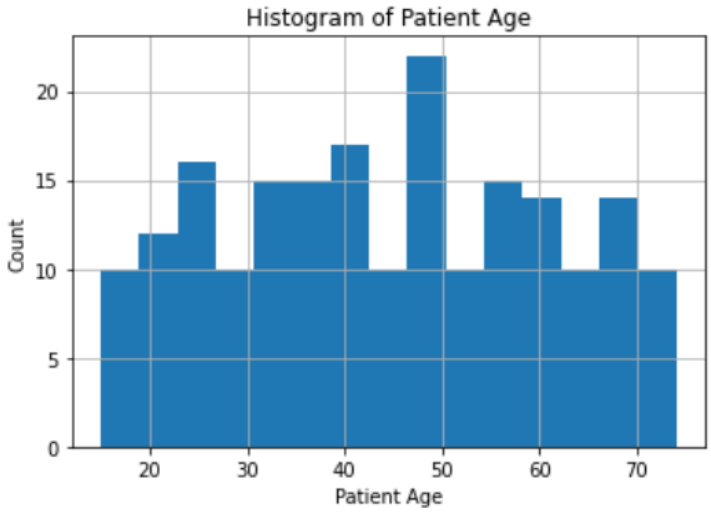



6-Plot the target field's histogram to describe each class with show information about it

This means the drugY has high records =95 and the least records for drugB

In [16]:

```
#histogram
plt.hist(df["Age"], bins=15)
plt.xlabel("Patient Age")
plt.ylabel("Count")
plt.title("Histogram of Patient Age")
plt.grid(True)
plt.show()
```

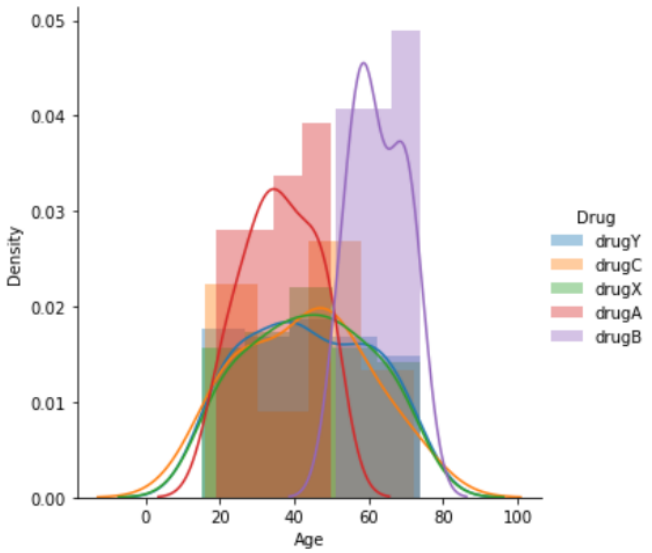



7- plot histogram to describe the count of patient Ages in the dataset.

This means most patients in the dataset have 50 years

In [18]:

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
for ojha, feature in enumerate(list(df.columns)[:1]):
    fg = sns.FacetGrid(df, hue='Drug',height=5)
    fg.map(sns.distplot, feature).add_legend()
    plt.show()
```




8- Distribution plots are used to visually analyze the distribution of data points in terms of frequency.

It shows the range between patient ages who can use a specific drug. Such as those who have age between 50 - 65 years can take drugB.

In [19]:
```
r= df.iloc[:, [0,1,2,3]].values
r
```

Out[19]:
```
array([[23, 'F', 'HIGH', 'HIGH'],
       [47, 'M', 'LOW', 'HIGH'],
       [47, 'M', 'LOW', 'HIGH'],
       [28, 'F', 'NORMAL', 'HIGH'],
       [61, 'F', 'LOW', 'HIGH'],
       [22, 'F', 'NORMAL', 'HIGH'],
       [49, 'F', 'NORMAL', 'HIGH'],
       [41, 'M', 'LOW', 'HIGH'],
       [60, 'M', 'NORMAL', 'HIGH'],
       [43, 'M', 'LOW', 'NORMAL'],
       [47, 'F', 'LOW', 'HIGH'],
       [34, 'F', 'HIGH', 'NORMAL'],
       [43, 'M', 'LOW', 'HIGH'],
       [74, 'F', 'LOW', 'HIGH'],
       [50, 'F', 'NORMAL', 'HIGH'],
       [16, 'F', 'HIGH', 'NORMAL'],
       [69, 'M', 'LOW', 'NORMAL'],
       [43, 'M', 'HIGH', 'HIGH'],
       [23, 'M', 'LOW', 'HIGH'],
       [32, 'F', 'HIGH', 'NORMAL'],
       [57, 'M', 'LOW', 'NORMAL'],
       [63, 'M', 'NORMAL', 'HIGH'],
       [47, 'M', 'LOW', 'NORMAL'],
       [48, 'F', 'LOW', 'HIGH'],
       [33, 'F', 'LOW', 'HIGH'],
```

In [25]:
```
y = df['Drug']
x =  df.drop(['Drug'], axis=1)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=42)
```

In [26]:
```
import category_encoders as tr
cols_drug=['Sex', 'BP', 'Cholesterol']
encoder = tr.OrdinalEncoder(cols_drug)
X_train = encoder.fit_transform(x_train)
X_test = encoder.transform(x_test)
```

9- Dealing with a dataset, extracting its values and labels, and splitting them into training sets have 67%  and testing sets have 33%. import libraries to convert features to ordinal integers.

In [27]:

```
# import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# instantiate the DecisionTreeClassifier model with criterion gini index
clf_tree = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)
clf_tree.fit(X_train, y_train)
y_pred = clf_tree.predict(X_test)
accuracy = accuracy_score(y_test,y_pred)
print('DecisionTreeClassifier accuracy score : {}'.format(accuracy))
```

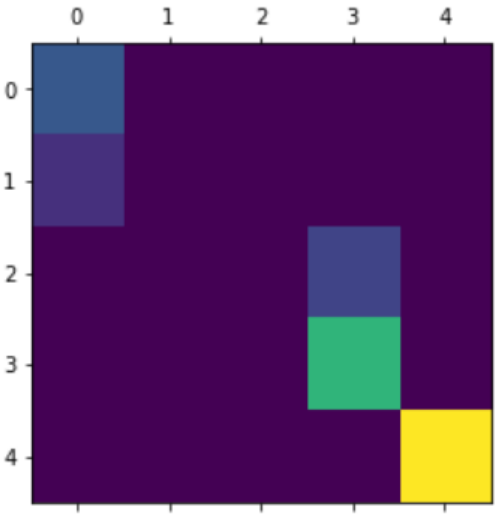
```
DecisionTreeClassifier accuracy score : 0.8484848484848485
```

10-We create the model and fit the data within it and calculate the accuracy.

n [29]:

```
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
[[ 8  0  0  0  0]
 [ 4  0  0  0  0]
 [ 0  0  0  6  0]
 [ 0  0  0 19  0]
 [ 0  0  0  0 29]]
```




11-Creating a confusion matrix to get a quick overview of the training results

[32]:

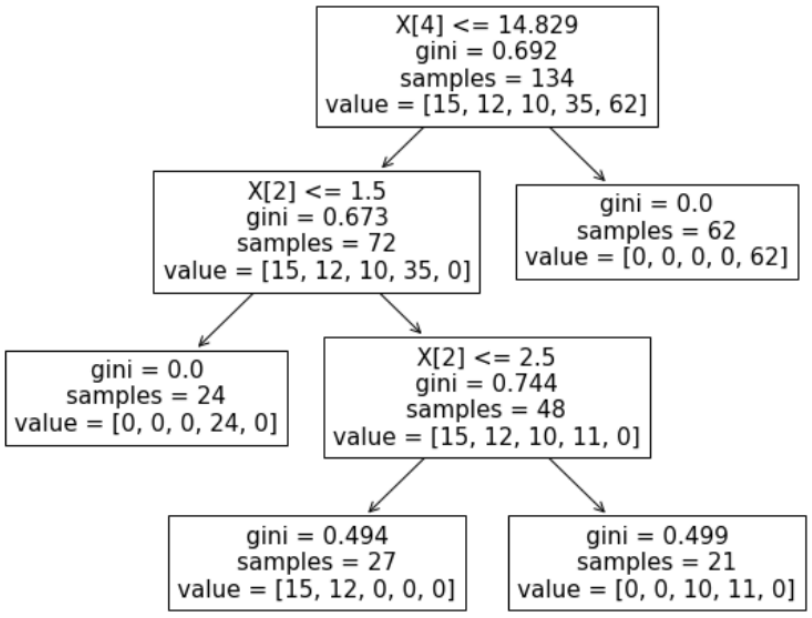
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, labels=df['Drug'].unique()))
```

```
              precision    recall  f1-score   support

       drugY       1.00      1.00      1.00        29
       drugC       1.00      1.00      1.00         6
       drugX       1.00      1.00      1.00        19
       drugA       1.00      1.00      1.00         8
       drugB       1.00      1.00      1.00         4

    accuracy                           1.00        66
   macro avg       1.00      1.00      1.00        66
weighted avg       1.00      1.00      1.00        66
```

12- We print the report that calculates precision, recall, and score report.

In [28]:

```
from sklearn import tree
plt.figure(figsize=(10,8))
tree.plot_tree(clf_tree.fit(X_train, y_train))
plt.show()
```




13-Finally, we plot the obtained tree to visualize the rules extracted from the dataset.

And as shown X[4] most affected parameter