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
```

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]
```

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

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

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

Out[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

In [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
```

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
...	...	...	...	...	...	...
195	False	False	False	False	False	False
196	False	False	False	False	False	False
197	False	False	False	False	False	False
198	False	False	False	False	False	False
199	False	False	False	False	False	False

200 rows × 6 columns

In [13]:
```
# Descriptive Statistics
df.describe
```

Out[13]:
```
<bound method NDFrame.describe of      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]>
```

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

Out[14]:

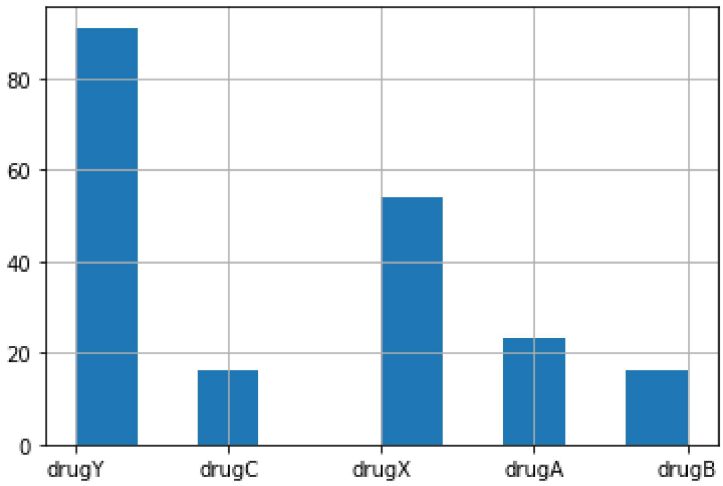
	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200	200	200	200.000000	200
unique	NaN	2	3	2	NaN	5
top	NaN	M	HIGH	HIGH	NaN	drugY

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
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

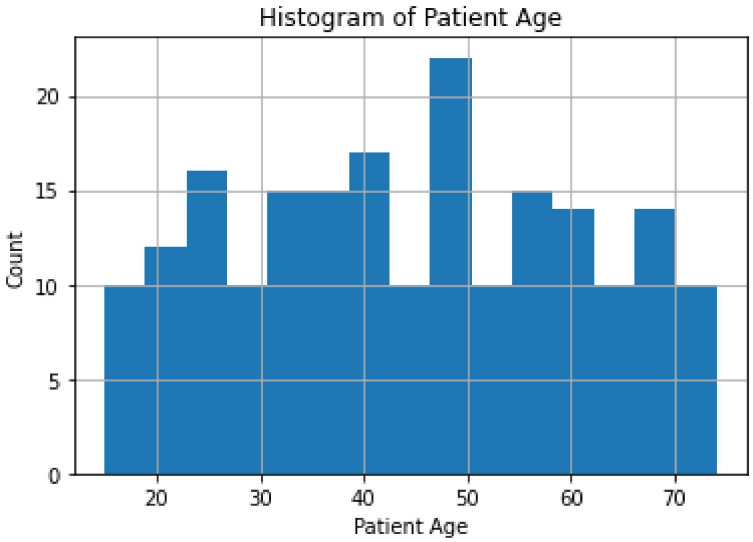
In [15]:
```
print(df.shape)
print(df.columns)
df['Drug'].hist()
```

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

Out[15]:

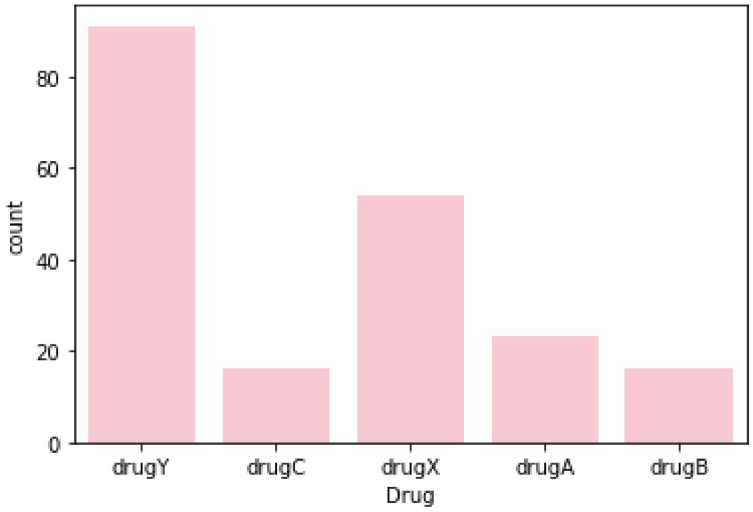


In [16]:
```
#histogram
plt.hist(df["Age"], bins=15)
plt.xlabel("Patient Age")
plt.ylabel("Count")
plt.title("Histogram of Patient Age")
plt.grid(True)
plt.show()
```

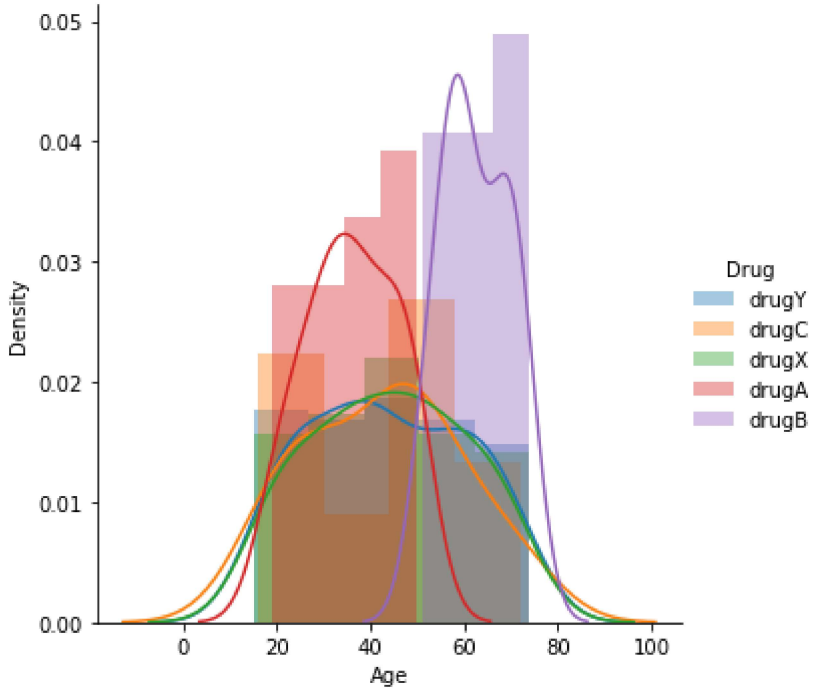


In [17]:
```
#countplot
sns.countplot(x = df['Drug'], color= 'pink')
```

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



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()
```



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'],
       [28, 'F', 'HIGH', 'NORMAL'],
       [31, 'M', 'HIGH', 'HIGH'],
       [49, 'F', 'NORMAL', 'NORMAL'],
       [39, 'F', 'LOW', 'NORMAL'],
       [45, 'M', 'LOW', 'HIGH'],
       [18, 'F', 'NORMAL', 'NORMAL'],
       [74, 'M', 'HIGH', 'HIGH'],
       [49, 'M', 'LOW', 'NORMAL'],
       [65, 'F', 'HIGH', 'NORMAL'],
       [53, 'M', 'NORMAL', 'HIGH'],
       [46, 'M', 'NORMAL', 'NORMAL'],
       [32, 'M', 'HIGH', 'NORMAL'],
```

```
[39, 'M', 'LOW', 'NORMAL'],
[39, 'F', 'NORMAL', 'NORMAL'],
[15, 'M', 'NORMAL', 'HIGH'],
[73, 'F', 'NORMAL', 'HIGH'],
[58, 'F', 'HIGH', 'NORMAL'],
[50, 'M', 'NORMAL', 'NORMAL'],
[23, 'M', 'NORMAL', 'HIGH'],
[50, 'F', 'NORMAL', 'NORMAL'],
[66, 'F', 'NORMAL', 'NORMAL'],
[37, 'F', 'HIGH', 'HIGH'],
[68, 'M', 'LOW', 'HIGH'],
[23, 'M', 'NORMAL', 'HIGH'],
[28, 'F', 'LOW', 'HIGH'],
[58, 'F', 'HIGH', 'HIGH'],
[67, 'M', 'NORMAL', 'NORMAL'],
[62, 'M', 'LOW', 'NORMAL'],
[24, 'F', 'HIGH', 'NORMAL'],
[68, 'F', 'HIGH', 'NORMAL'],
[26, 'F', 'LOW', 'HIGH'],
[65, 'M', 'HIGH', 'NORMAL'],
[40, 'M', 'HIGH', 'HIGH'],
[60, 'M', 'NORMAL', 'NORMAL'],
[34, 'M', 'HIGH', 'HIGH'],
[38, 'F', 'LOW', 'NORMAL'],
[24, 'M', 'HIGH', 'NORMAL'],
[67, 'M', 'LOW', 'NORMAL'],
[45, 'M', 'LOW', 'NORMAL'],
[60, 'F', 'HIGH', 'HIGH'],
[68, 'F', 'NORMAL', 'NORMAL'],
[29, 'M', 'HIGH', 'HIGH'],
[17, 'M', 'NORMAL', 'NORMAL'],
[54, 'M', 'NORMAL', 'HIGH'],
[18, 'F', 'HIGH', 'NORMAL'],
[70, 'M', 'HIGH', 'HIGH'],
[28, 'F', 'NORMAL', 'HIGH'],
[24, 'F', 'NORMAL', 'HIGH'],
[41, 'F', 'NORMAL', 'NORMAL'],
[31, 'M', 'HIGH', 'NORMAL'],
[26, 'M', 'LOW', 'NORMAL'],
[36, 'F', 'HIGH', 'HIGH'],
[26, 'F', 'HIGH', 'NORMAL'],
[19, 'F', 'HIGH', 'HIGH'],
[32, 'F', 'LOW', 'NORMAL'],
[60, 'M', 'HIGH', 'HIGH'],
[64, 'M', 'NORMAL', 'HIGH'],
[32, 'F', 'LOW', 'HIGH'],
[38, 'F', 'HIGH', 'NORMAL'],
[47, 'F', 'LOW', 'HIGH'],
[59, 'M', 'HIGH', 'HIGH'],
[51, 'F', 'NORMAL', 'HIGH'],
[69, 'M', 'LOW', 'HIGH'],
[37, 'F', 'HIGH', 'NORMAL'],
[50, 'F', 'NORMAL', 'NORMAL'],
[62, 'M', 'NORMAL', 'HIGH'],
[41, 'M', 'HIGH', 'NORMAL'],
[29, 'F', 'HIGH', 'HIGH'],
[42, 'F', 'LOW', 'NORMAL'],
[56, 'M', 'LOW', 'HIGH'],
[36, 'M', 'LOW', 'NORMAL'],
[58, 'F', 'LOW', 'HIGH'],
[56, 'F', 'HIGH', 'HIGH'],
[20, 'M', 'HIGH', 'NORMAL'],
[15, 'F', 'HIGH', 'NORMAL'],
[31, 'M', 'HIGH', 'NORMAL'],
```

```
[45, 'F', 'HIGH', 'HIGH'],
[28, 'F', 'LOW', 'HIGH'],
[56, 'M', 'NORMAL', 'HIGH'],
[22, 'M', 'HIGH', 'NORMAL'],
[37, 'M', 'LOW', 'NORMAL'],
[22, 'M', 'NORMAL', 'HIGH'],
[42, 'M', 'LOW', 'HIGH'],
[72, 'M', 'HIGH', 'NORMAL'],
[23, 'M', 'NORMAL', 'HIGH'],
[50, 'M', 'HIGH', 'HIGH'],
[47, 'F', 'NORMAL', 'NORMAL'],
[35, 'M', 'LOW', 'NORMAL'],
[65, 'F', 'LOW', 'NORMAL'],
[20, 'F', 'NORMAL', 'NORMAL'],
[51, 'M', 'HIGH', 'HIGH'],
[67, 'M', 'NORMAL', 'NORMAL'],
[40, 'F', 'NORMAL', 'HIGH'],
[32, 'F', 'HIGH', 'NORMAL'],
[61, 'F', 'HIGH', 'HIGH'],
[28, 'M', 'NORMAL', 'HIGH'],
[15, 'M', 'HIGH', 'NORMAL'],
[34, 'M', 'NORMAL', 'HIGH'],
[36, 'F', 'NORMAL', 'HIGH'],
[53, 'F', 'HIGH', 'NORMAL'],
[19, 'F', 'HIGH', 'NORMAL'],
[66, 'M', 'HIGH', 'HIGH'],
[35, 'M', 'NORMAL', 'NORMAL'],
[47, 'M', 'LOW', 'NORMAL'],
[32, 'F', 'NORMAL', 'HIGH'],
[70, 'F', 'NORMAL', 'HIGH'],
[52, 'M', 'LOW', 'NORMAL'],
[49, 'M', 'LOW', 'NORMAL'],
[24, 'M', 'NORMAL', 'HIGH'],
[42, 'F', 'HIGH', 'HIGH'],
[74, 'M', 'LOW', 'NORMAL'],
[55, 'F', 'HIGH', 'HIGH'],
[35, 'F', 'HIGH', 'HIGH'],
[51, 'M', 'HIGH', 'NORMAL'],
[69, 'F', 'NORMAL', 'HIGH'],
[49, 'M', 'HIGH', 'NORMAL'],
[64, 'F', 'LOW', 'NORMAL'],
[60, 'M', 'HIGH', 'NORMAL'],
[74, 'M', 'HIGH', 'NORMAL'],
[39, 'M', 'HIGH', 'HIGH'],
[61, 'M', 'NORMAL', 'HIGH'],
[37, 'F', 'LOW', 'NORMAL'],
[26, 'F', 'HIGH', 'NORMAL'],
[61, 'F', 'LOW', 'NORMAL'],
[22, 'M', 'LOW', 'HIGH'],
[49, 'M', 'HIGH', 'NORMAL'],
[68, 'M', 'HIGH', 'HIGH'],
[55, 'M', 'NORMAL', 'NORMAL'],
[72, 'F', 'LOW', 'NORMAL'],
[37, 'M', 'LOW', 'NORMAL'],
[49, 'M', 'LOW', 'HIGH'],
[31, 'M', 'HIGH', 'NORMAL'],
[53, 'M', 'LOW', 'HIGH'],
[59, 'F', 'LOW', 'HIGH'],
[34, 'F', 'LOW', 'NORMAL'],
[30, 'F', 'NORMAL', 'HIGH'],
[57, 'F', 'HIGH', 'NORMAL'],
[43, 'M', 'NORMAL', 'NORMAL'],
[21, 'F', 'HIGH', 'NORMAL'],
[16, 'M', 'HIGH', 'NORMAL'],
```

```
       [38, 'M', 'LOW', 'HIGH'],
       [58, 'F', 'LOW', 'HIGH'],
       [57, 'F', 'NORMAL', 'HIGH'],
       [51, 'F', 'LOW', 'NORMAL'],
       [20, 'F', 'HIGH', 'HIGH'],
       [28, 'F', 'NORMAL', 'HIGH'],
       [45, 'M', 'LOW', 'NORMAL'],
       [39, 'F', 'NORMAL', 'NORMAL'],
       [41, 'F', 'LOW', 'NORMAL'],
       [42, 'M', 'HIGH', 'NORMAL'],
       [73, 'F', 'HIGH', 'HIGH'],
       [48, 'M', 'HIGH', 'NORMAL'],
       [25, 'M', 'NORMAL', 'HIGH'],
       [39, 'M', 'NORMAL', 'HIGH'],
       [67, 'F', 'NORMAL', 'HIGH'],
       [22, 'F', 'HIGH', 'NORMAL'],
       [59, 'F', 'NORMAL', 'HIGH'],
       [20, 'F', 'LOW', 'NORMAL'],
       [36, 'F', 'HIGH', 'NORMAL'],
       [18, 'F', 'HIGH', 'HIGH'],
       [57, 'F', 'NORMAL', 'NORMAL'],
       [70, 'M', 'HIGH', 'HIGH'],
       [47, 'M', 'HIGH', 'HIGH'],
       [65, 'M', 'HIGH', 'NORMAL'],
       [64, 'M', 'HIGH', 'NORMAL'],
       [58, 'M', 'HIGH', 'HIGH'],
       [23, 'M', 'HIGH', 'HIGH'],
       [72, 'M', 'LOW', 'HIGH'],
       [72, 'M', 'LOW', 'HIGH'],
       [46, 'F', 'HIGH', 'HIGH'],
       [56, 'F', 'LOW', 'HIGH'],
       [16, 'M', 'LOW', 'HIGH'],
       [52, 'M', 'NORMAL', 'HIGH'],
       [23, 'M', 'NORMAL', 'NORMAL'],
       [40, 'F', 'LOW', 'NORMAL']], dtype=object)
```

In [23]:
```
pip install Graphviz
```

```
Requirement already satisfied: Graphviz in c:\anaconda3\lib\site-packages (0.19.1)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
```

In [ ]:
```
```

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)
```

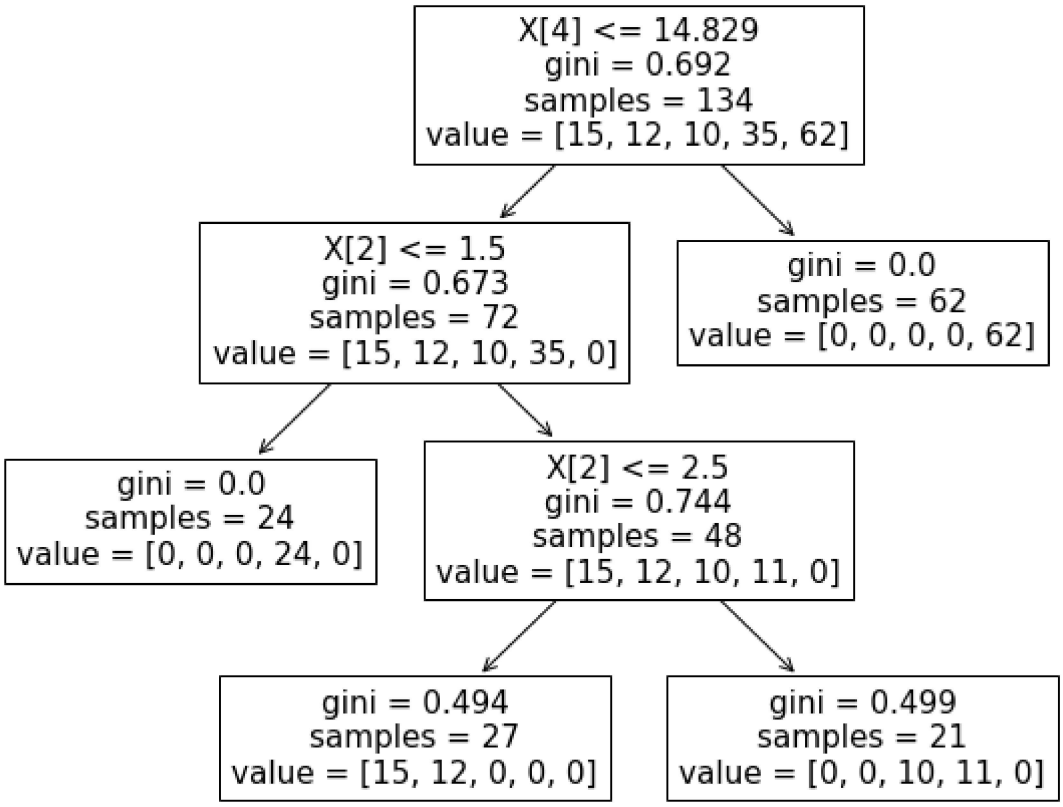
In [27]:
```
# import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
# 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
```

In [28]:

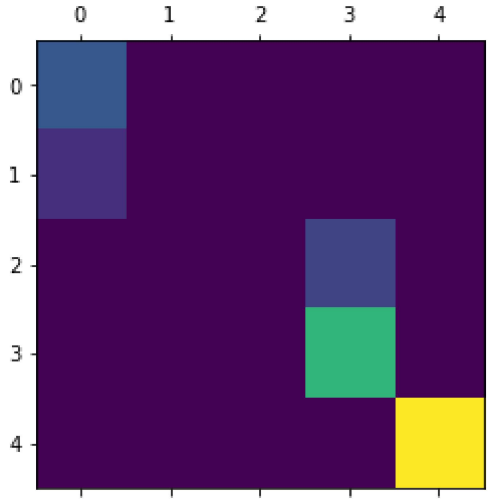
```
from sklearn import tree
plt.figure(figsize=(10,8))
tree.plot_tree(clf_tree.fit(X_train, y_train))
plt.show()
```



In [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]]
```



In [31]:

```
tree = DecisionTreeClassifier(criterion='gini',
min_samples_leaf=5,
min_samples_split=5,
max_depth=None,
random_state=42)
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:1.0
```

In [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
```

In [ ]: