Drug Classification

this notebook purpose is to explore the dataset Drug to predictthe drug that are being used by a person

the procedure that are needed for step 1 are listed below as follows:

1. Exploration of Data

- 1.1 Importing Libraries

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

- 1.2. Viewing The Csv File via DataFrame

In [2]:
```
df = pd.read_csv('Drug.csv');
df
```

Out[2]:

	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 [3]:
```
df.shape
```

Out[3]:
```
(200, 6)
```

we can see here that we have 200 rows and 6 columns(Age,Sex,BP,Cholesterol,Na_to_K,Drug)

we can also view the columns by using df.columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

here are the columns

2. Data Pre-processing

- 2.1 Dropping duplicates

In [5]:
```
df = df.drop_duplicates()
```

In [6]:
```
df.shape
```

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

here we drop the duplicates by drop_duplicates function but as we can see there are no duplicates

- 2.2 filling null/missing values

we should check our missing values and fill it with appropriate values to make our dataset more clean and accurate in the modelling

In [7]:
```
df.isna().sum() #checking all the missing or null values and summing it up per
```

Out[7]:
```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

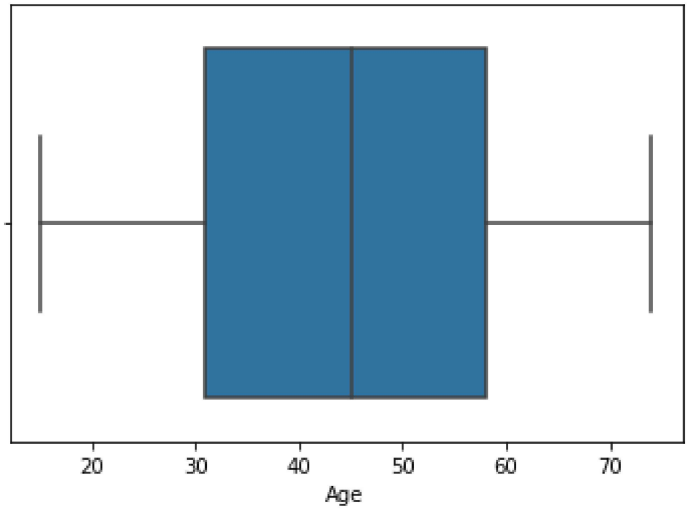
we can see here that our dataset have no missing values so we will skip the filling out null/missing values

- 2.3 Removing extreme values/outliers

here we will check for outliers so our dataset will be more cleaner so our model will have better results

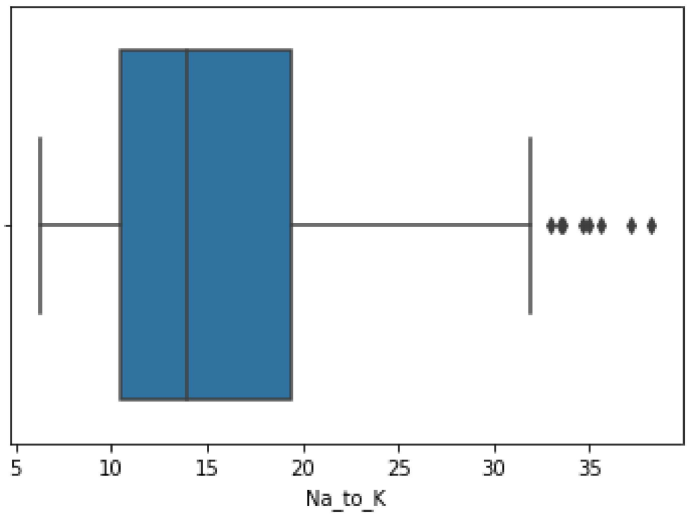
In [8]:
```
sns.boxplot(df['Age']); # checking the outliers for Age column
```

```
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```



In [9]:
```
sns.boxplot(df['Na_to_K']); # checking the outliers for Na_to_K column
```

```
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```



Here, we can see that the Na_to_K is the only numerical column that have an outliers

we will now remove the outliers using the iqr method

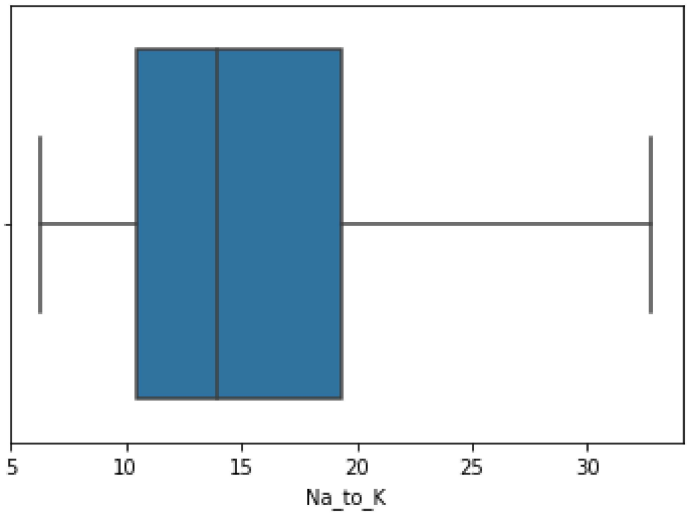
In [10]:
```
cols = ['Na_to_K'];

for c in cols:
    q3 = df[c].quantile(.75);
    q1 = df[c].quantile(.25);
    IQR = q3 - q1;
    _max = q3 + (IQR * 1.5);
    _min = q1 - (IQR * 1.5);
    df.loc[df[c] < _min,c] = _min;
    df.loc[df[c] > _max,c] = _max;
```

we loop every row in Na_to_K and change the values of those rows that are lower than *min and set their values to min_value and the row that have higher values than the max* value will be set to have the max_value

In [11]:
```
sns.boxplot(df['Na_to_K'])
```

```
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[11]:
```
<AxesSubplot:xlabel='Na_to_K'>
```



- 2.3 Converting catergorical features into Numerical Features

we can see above that there columns that are not numerical values, we need to convert it to numercial values

we will use the get_dummies() from the pandas libraries

but first we need to copy the current df to another variable so the df will not be alter because we

In [12]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder();
df_new = df.copy();
df_new
```

Out[12]:

	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 [13]:
```
for col in df_new.columns:
    if(col != "Drug"):
        df_new[col] = encoder.fit_transform(df_new[col])
```

Here, we encode all the categorical values that are not the Drug column and fit transform it

In [14]:
```
df_new['Drug'] = encoder.fit_transform(df_new['Drug'])
```

Here we one hot encoding the drug column

In [15]:
```
df_new
```

Out[15]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	8	0	0	0	167	0
1	30	1	1	0	89	3
2	30	1	1	0	43	3
3	12	0	2	0	10	4
4	44	0	1	0	133	0
...	...	...	...	...	...	...
195	39	0	1	0	69	3
196	1	1	1	0	75	3
197	35	1	2	0	36	4
198	8	1	2	1	102	4
199	24	0	1	1	66	4

200 rows × 6 columns

In [16]:
```
df_new.dtypes
```

Out[16]:
```
Age            int64
Sex            int32
BP             int32
Cholesterol    int32
Na_to_K        int64
Drug           int32
dtype: object
```

- 2.3 Data Normalization/ features scaling

Here we will now normalize our dataset using MinMaxScaler

In [17]:
```
from sklearn.preprocessing import MinMaxScaler

df_scaled = df_new.copy();
scaler = MinMaxScaler();
```

we create a new dataframe by copying the df_new value to the new dataframe df_scaled

we also assign the MinMaxScaler to a variable scaler

In [18]:
```
for col in df_scaled.columns:
    if(col != "Drug"):
        df_scaled[col] = scaler.fit_transform(df_scaled[col].values.reshape(-1
```

we loop on our df_scaled column and fit_transform our df_scaled columns and have them a value of ranging to -1 to 1 only

In [19]:
```
df_scaled
```

Out[19]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	0.142857	0.0	0.0	0.0	0.878947	0
1	0.535714	1.0	0.5	0.0	0.468421	3
2	0.535714	1.0	0.5	0.0	0.226316	3
3	0.214286	0.0	1.0	0.0	0.052632	4
4	0.785714	0.0	0.5	0.0	0.700000	0
...	...	...	...	...	...	...
195	0.696429	0.0	0.5	0.0	0.363158	3
196	0.017857	1.0	0.5	0.0	0.394737	3
197	0.625000	1.0	1.0	0.0	0.189474	4
198	0.142857	1.0	1.0	1.0	0.536842	4
199	0.428571	0.0	0.5	1.0	0.347368	4

200 rows × 6 columns

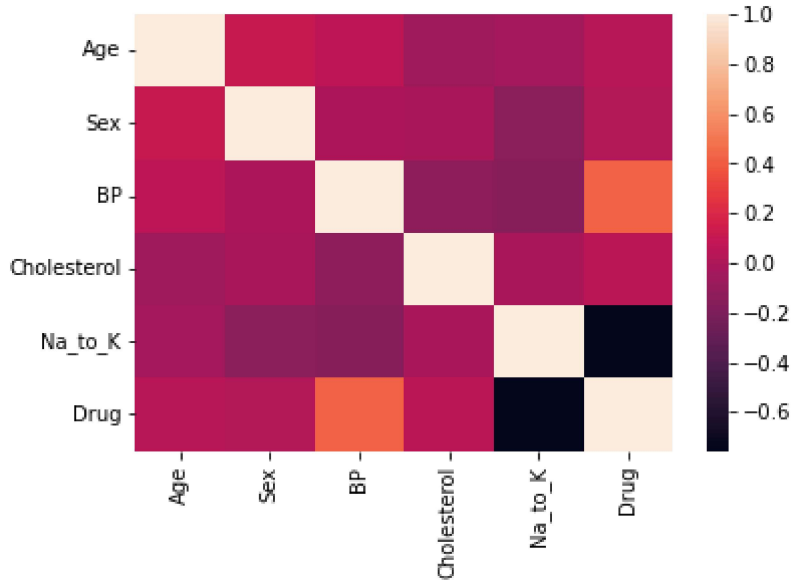
this is our new dataframe

- 2.4 Feature Selection / Correlation Analysis

we are finished normalizing our dataset so will now have to find its correlations to each other

In [20]:
```
sns.heatmap(df_scaled.corr())
```

Out[20]: <AxesSubplot:>



Here we can see there are no correlated features so we will not drop any feature

3. Modelling

out dataset are now cleaned, we can start modelling now

- 3.1. split target variable to feature variables

we will split our target variable to the rest of our features

In [44]:
```
X = df_scaled.drop(['Drug'],axis=1); #dropping the 'Drug' column or target var
y = df_scaled['Drug']; #getting the target variable
```

- 3.2. Split our dataset to training and test

we will split our dataset to training and test data

In [45]:
```
from sklearn.model_selection import train_test_split #import the library
```

In [46]:
```
# stratify will balance the partition of the dataset for training and testing
# in the train data and the edible and poisonous data will also be 60 - 40
#the partition of the datasets are 80 percent in training data and 20 percent
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=42,train_siz
```

In [47]:
```
print(X_train.shape);
print(X_test.shape);
```

```
(160, 5)
(40, 5)
```

we have a 160 row and 5 columns in our train data and 40 and 5 columns in test data

- 3.3 choose the best model

now we will create and choose our best model but first we need to create a function that will evaluate our models

In [48]:
```
df['Drug'].unique()
```

Out[48]:
```
array(['DrugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

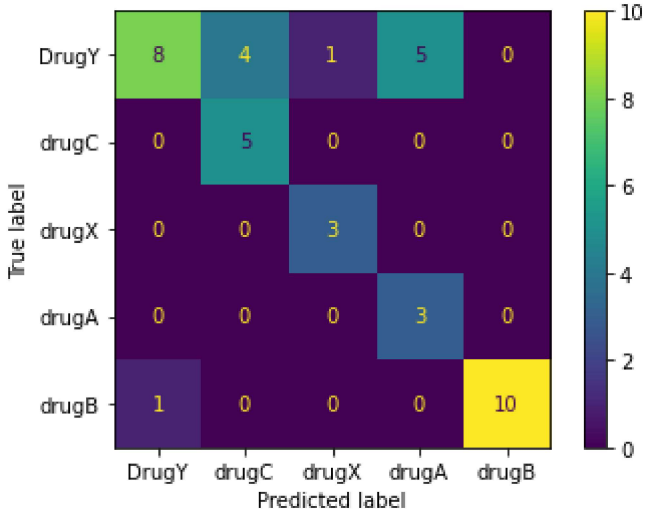
In [49]:
```
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import classification_report


def evaluate_model(model):
    print(classification_report(y_test, model.predict(X_test), target_names=['
    plot_confusion_matrix(model, X_test, y_test, display_labels=['DrugY','drug
    print('Training Score:', model.score(X_train,y_train))
    print('Test Score:', model.score(X_test,y_test))
```

This is the function that will evaluate our models

- 3.3.1 Gaussian Naive Bayes

In [50]:
```
from sklearn.naive_bayes import GaussianNB


gnb = GaussianNB()
gnb.fit(X_train,y_train);
gnb.score(X_test,y_test);


evaluate_model(gnb)
```

```
              precision    recall  f1-score   support

       DrugY       0.89      0.44      0.59        18
       drugC       0.56      1.00      0.71         5
       drugX       0.75      1.00      0.86         3
       drugA       0.38      1.00      0.55         3
       drugB       1.00      0.91      0.95        11

    accuracy                           0.73        40
   macro avg       0.71      0.87      0.73        40
weighted avg       0.83      0.72      0.72        40

Training Score: 0.74375
Test Score: 0.725

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



- 3.3.2 Bernoulli Naive Bayes

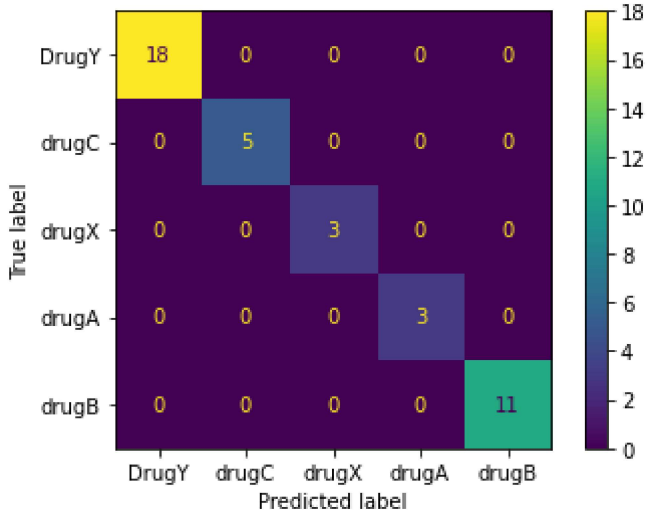
In [51]:
```
from sklearn.naive_bayes import BernoulliNB


bnb = BernoulliNB()
bnb.fit(X_train,y_train);
bnb.score(X_test,y_test);
evaluate_model(bnb)
```

```
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\metrics\_classif
ication.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defi
ned and being set to 0.0 in labels with no predicted samples. Use `zero_div
ision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\metrics\_classif
ication.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defi
ned and being set to 0.0 in labels with no predicted samples. Use `zero_div
ision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\metrics\_classif
ication.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defi
ned and being set to 0.0 in labels with no predicted samples. Use `zero_div
ision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Funct
ion `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.
2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or
```

- 3.3.3 Decision Tree Classifier

In [52]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier();
dt.fit(X_train,y_train);
dt.score(X_test,y_test);

evaluate_model(dt)
```

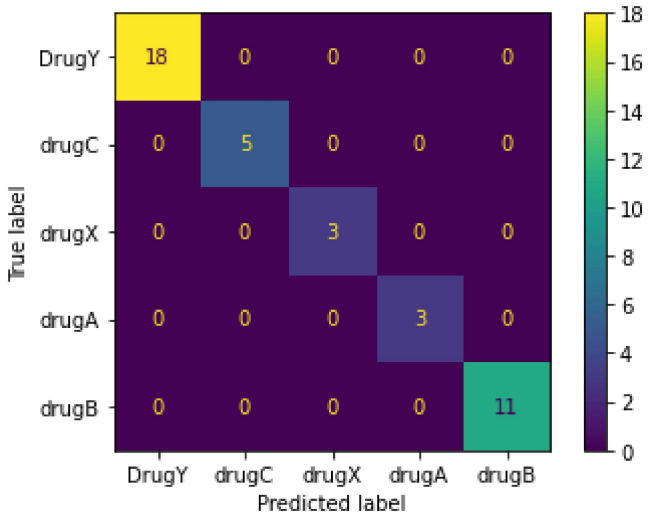
```
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        18
       drugC       1.00      1.00      1.00         5
       drugX       1.00      1.00      1.00         3
       drugA       1.00      1.00      1.00         3
       drugB       1.00      1.00      1.00        11

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Training Score: 1.0
Test Score: 1.0

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



- 3.3.4 Random Forest Clasifier

In [53]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train,y_train)
rf.score(X_test,y_test)

evaluate_model(rf)
```

```
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        18
       drugC       1.00      1.00      1.00         5
       drugX       1.00      1.00      1.00         3
       drugA       1.00      1.00      1.00         3
       drugB       1.00      1.00      1.00        11

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Training Score: 1.0
Test Score: 1.0

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



- 3.3.5 KNN Classifier

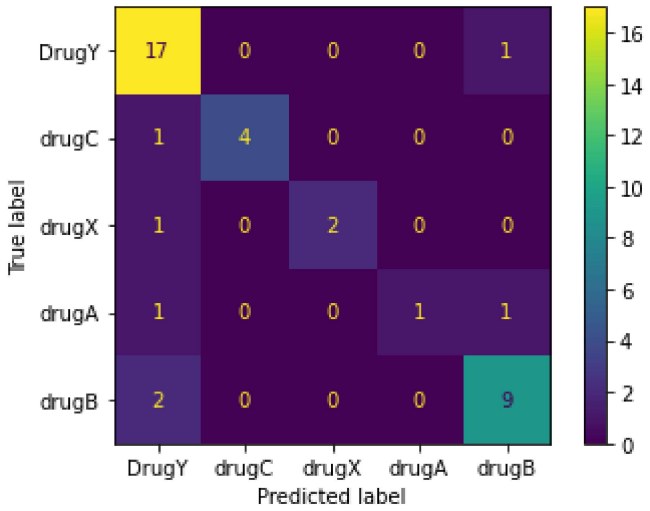
In [54]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier();
knn.fit(X_train,y_train);
knn.score(X_test,y_test);

evaluate_model(knn)
```

```
              precision    recall  f1-score   support

       DrugY       0.62      0.72      0.67        18
       drugC       1.00      0.40      0.57         5
       drugX       0.67      0.67      0.67         3
       drugA       1.00      0.33      0.50         3
       drugB       0.62      0.73      0.67        11

    accuracy                           0.65        40
   macro avg       0.78      0.57      0.61        40
weighted avg       0.70      0.65      0.64        40

Training Score: 0.91875
Test Score: 0.65

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Funct
ion `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.
2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator
```

- 3.3.6 Logistic Regression

In [55]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression();
lr.fit(X_train,y_train);
lr.score(X_test,y_test);

evaluate_model(lr)
```

```
              precision    recall  f1-score   support

       DrugY       0.77      0.94      0.85        18
       drugC       1.00      0.80      0.89         5
       drugX       1.00      0.67      0.80         3
       drugA       1.00      0.33      0.50         3
       drugB       0.82      0.82      0.82        11

    accuracy                           0.82        40
   macro avg       0.92      0.71      0.77        40
weighted avg       0.85      0.82      0.82        40

Training Score: 0.9125
Test Score: 0.825

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



4. Hyper Parameter Tuning / Cross Validation

To tune our model into a better model

in our models we saw that the random forest and decision tree models are the best models but if the dataset get bigger the prediction will be inaccurate so we need to tune or Hyperparameter tuning our models

first we need to import the libraries for RandomizedSearchCV for cross validation

In [56]:
```
from sklearn.model_selection import RandomizedSearchCV
```

- 4.1 Decition Tree Classifier

Here we have the parameters for Decision tree

In [66]:
```
dt.get_params()
```

Out[66]:
```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

doing the cross validation

In [57]:
```
params = {
    'criterion': ['gini','entropy'],
    'splitter': ['best','random'],
    'max_depth': [5,10,15,20]
}

rsearch = RandomizedSearchCV(DecisionTreeClassifier(),params,n_iter=15,cv=10)
rsearch.fit(X_train,y_train)
```

Out[57]:
```
RandomizedSearchCV(cv=10, estimator=DecisionTreeClassifier(), n_iter=15,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [5, 10, 15, 20],
                                        'splitter': ['best', 'random']})
```

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.

and here is the best parameters

In [58]:
```
rsearch.best_params_
```

Out[58]:
```
{'splitter': 'best', 'max_depth': 5, 'criterion': 'gini'}
```

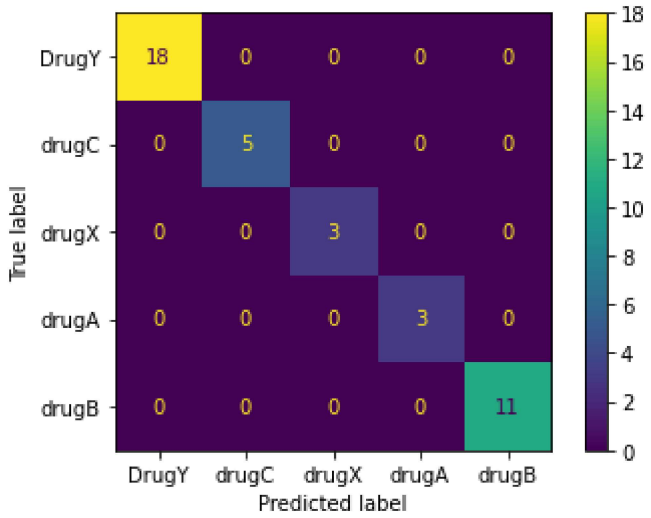
we can see that the parameters that we get from the cross validations are the following above

In [59]:
```
dt_tuned = DecisionTreeClassifier(splitter='best',max_depth = 5,criterion='gin
dt_tuned.fit(X_train,y_train);
dt_tuned.score(X_test,y_test);
```

we fit our model again with our best parameters

In [60]:
```
evaluate_model(dt_tuned)
```

```
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        18
       drugC       1.00      1.00      1.00         5
       drugX       1.00      1.00      1.00         3
       drugA       1.00      1.00      1.00         3
       drugB       1.00      1.00      1.00        11

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Training Score: 1.0
Test Score: 1.0

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



we evaluated our model and we see that the score are still the same

- 4.2 Random Forest

Here are the parameters RandomForest

In [61]:
```
rf.get_params()
```

Out[61]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Here, We do the cross validation

In [62]:
```
params = {
    'criterion': ['gini','entropy','log_loss'],
    'max_depth': [5,10,15,20],
    'n_estimators': [20,30,40,50,60,70,80,90,100]
}

rsearch = RandomizedSearchCV(RandomForestClassifier(),params,cv=10)
rsearch.fit(X_train,y_train)
```

Out[62]:
```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(),
                   param_distributions={'criterion': ['gini', 'entropy',
                                                      'log_loss'],
                                        'max_depth': [5, 10, 15, 20],
                                        'n_estimators': [20, 30, 40, 50, 60,
70,
                                                         80, 90, 100]})
```

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.

Here are the best parameters for random forest according to the CV

In [63]:
```
rsearch.best_params_
```

Out[63]:
```
{'n_estimators': 80, 'max_depth': 15, 'criterion': 'gini'}
```

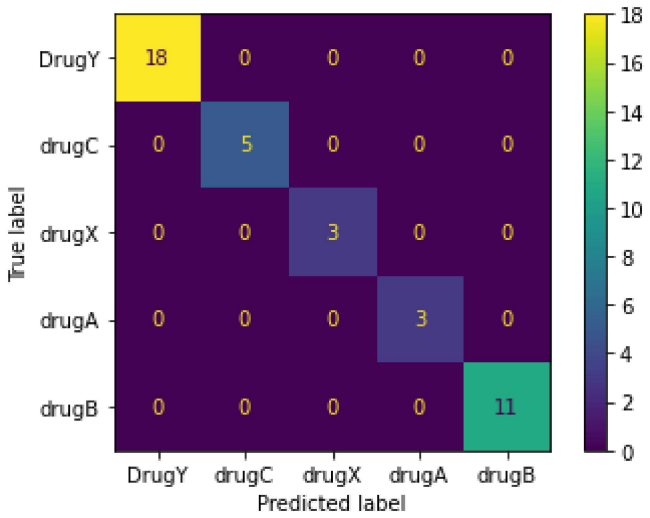
Here we fit our model again with our best parameters

In [64]:
```
rf_tuned = RandomForestClassifier(n_estimators = 80,max_depth = 15,criterion='
rf_tuned.fit(X_train,y_train);
rf_tuned.score(X_test,y_test);
```

Here, we evaluate our model again

In [65]:
```
evaluate_model(dt_tuned)
```

```
              precision    recall  f1-score   support

       DrugY       1.00      1.00      1.00        18
       drugC       1.00      1.00      1.00         5
       drugX       1.00      1.00      1.00         3
       drugA       1.00      1.00      1.00         3
       drugB       1.00      1.00      1.00        11

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Training Score: 1.0
Test Score: 1.0

C:\Users\melvin\Desktop\Anaconda\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use
one of the class methods: ConfusionMatrixDisplay.from_predictions or Confusio
nMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



5. Choosing a model

we can say that the best model are the Decision tree and the random forest models because the models have a 100% accuracy

In [ ]: