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

Reading in the Data

In [ ]:
```
pwd = os.getcwd()
filepath = os.path.join(pwd, 'drug200.csv')
```

In [ ]:
```
drug_data = pd.read_csv(filepath)
drug_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

Exploring the Data

In [ ]:
```
drug_data.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 [ ]:
```
### Are there any null values? ###
```

```
drug_data.isnull().sum()
```

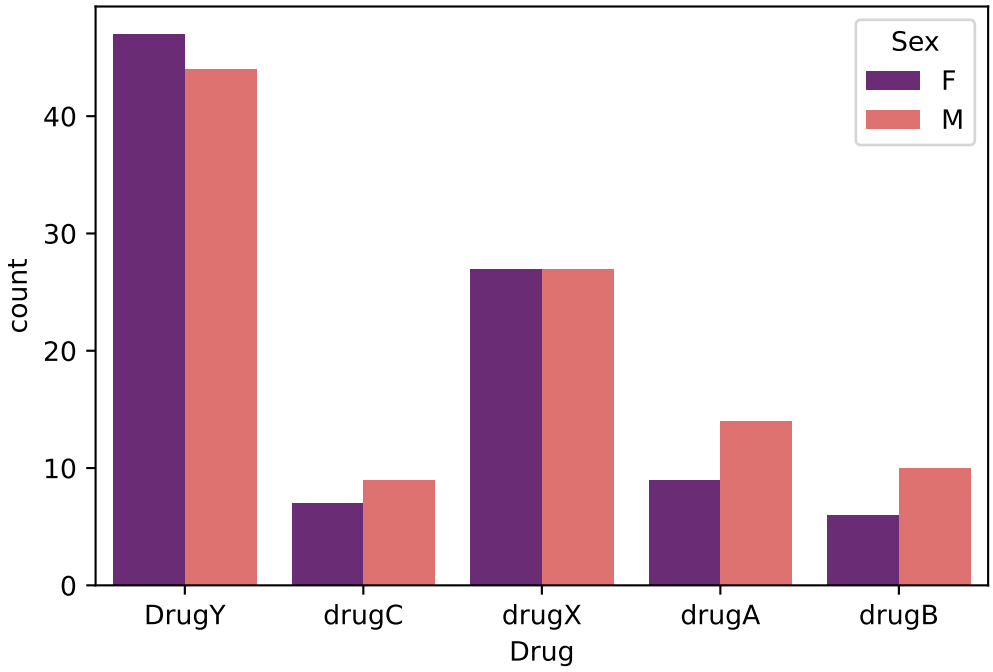
Out[ ]:
```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

Visualizing the Data

In [ ]:
```
drug = drug_data.copy()
```

In [ ]:
```
sns.countplot(x = 'Drug', data = drug, palette="magma", hue='Sex')
```

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

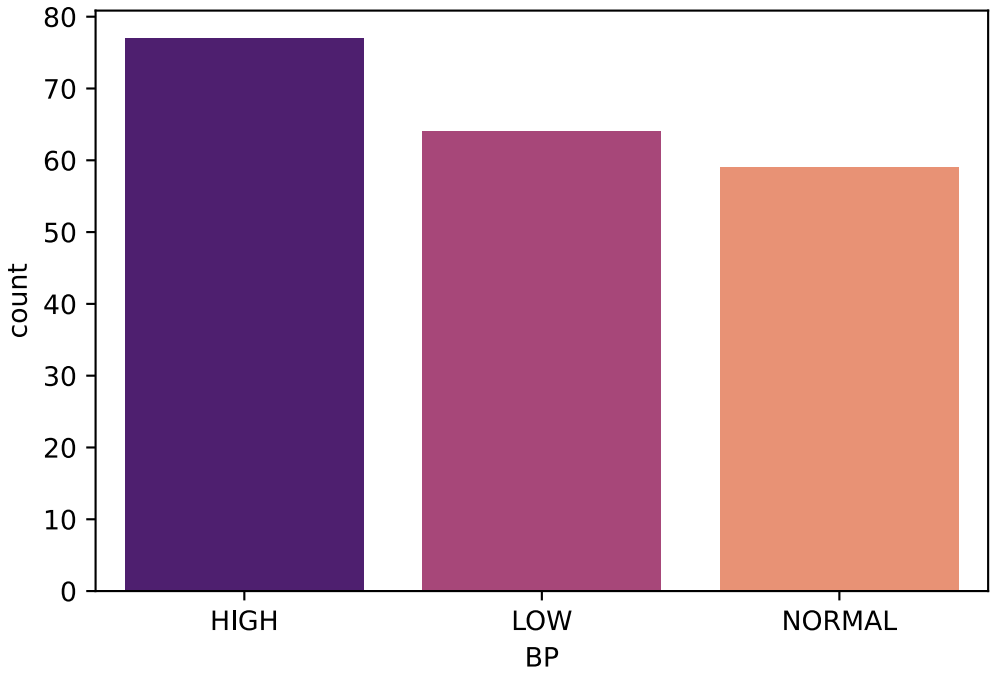


DrugY has the most frequency, which means it will be the highest predicted drug in our model.

No relationship between sex and drug use.

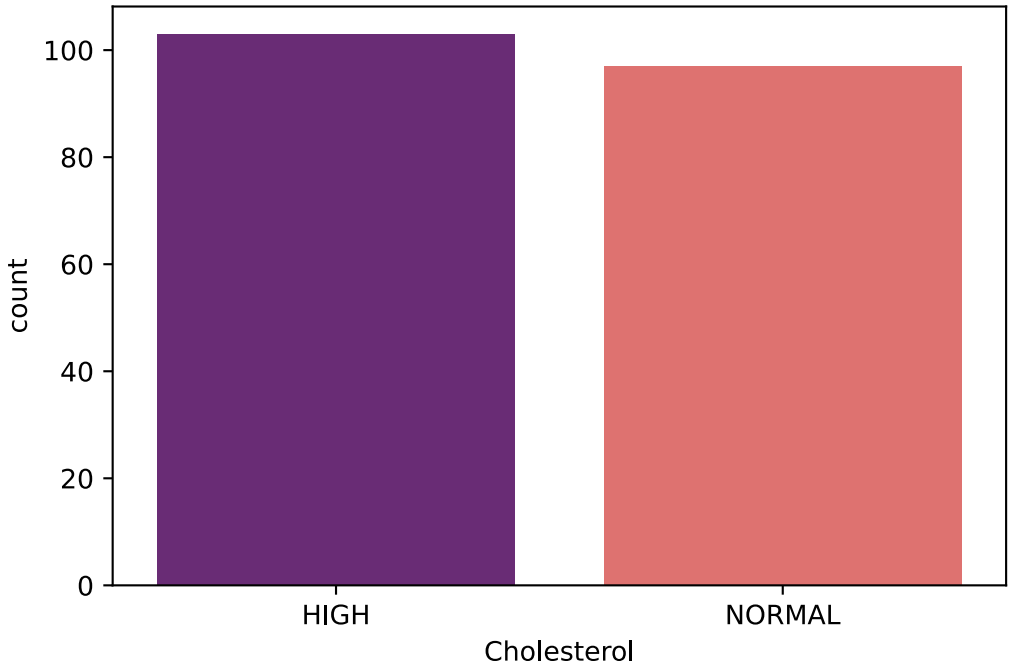
In [ ]:
```
sns.countplot(x = 'BP', data = drug, palette="magma")
```

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



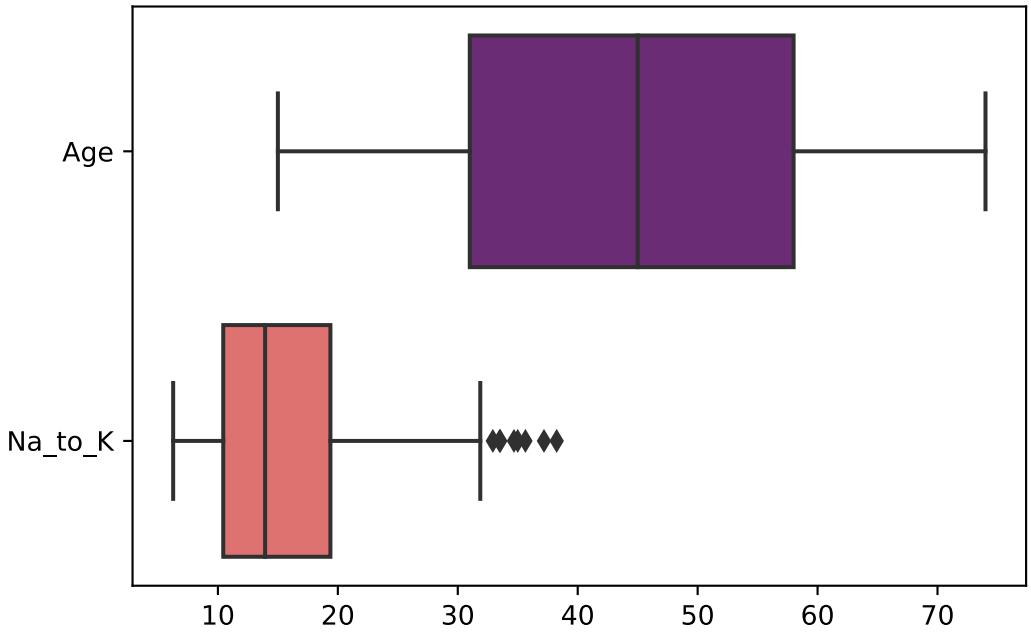
In [ ]:
```
sns.countplot(x = 'Cholesterol', data = drug, palette="magma")
```

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



In [ ]:
```
sns.boxplot(data=drug, palette="magma", orient="h")
```

Out[ ]:
```
<AxesSubplot:>
```



We can see that there are a few outliers in the Na_to_K column

In [ ]:
```
### Removing the outliers using IQR ###
IQR1= np.quantile(drug['Na_to_K'],0.25)
IQR3= np.quantile(drug['Na_to_K'],0.75)
IQR=IQR3-IQR1
IQR
```

Out[ ]: 8.934499999999998

In [ ]:
```
outliers = drug[drug['Na_to_K'] > (IQR3+1.5*IQR)]
outliers
```

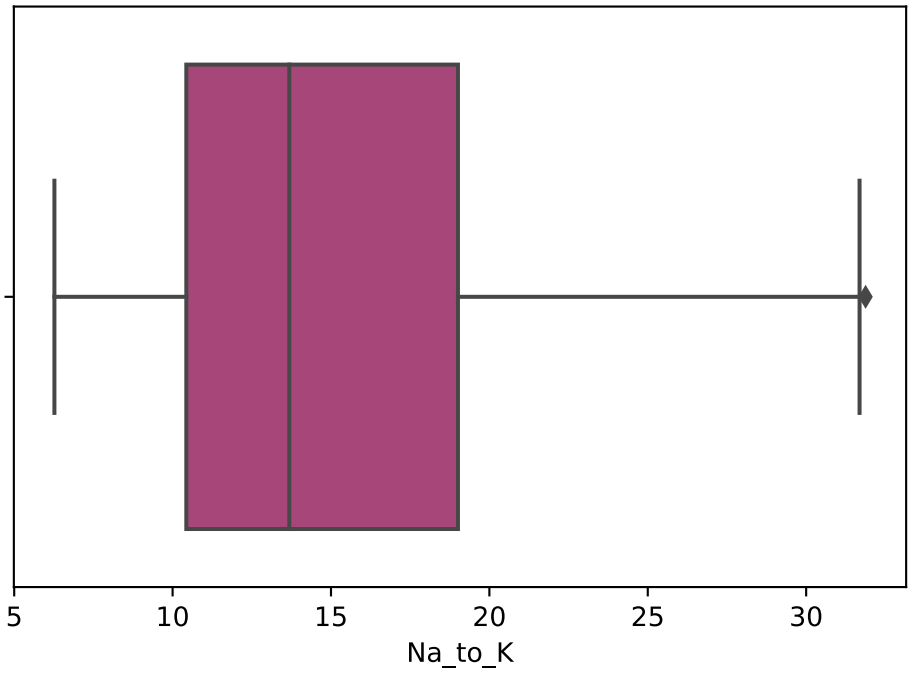
Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
24	33	F	LOW	HIGH	33.486	DrugY
96	58	F	LOW	HIGH	38.247	DrugY
98	20	M	HIGH	NORMAL	35.639	DrugY
128	47	M	LOW	NORMAL	33.542	DrugY
131	52	M	LOW	NORMAL	32.922	DrugY
184	18	F	HIGH	HIGH	37.188	DrugY
188	65	M	HIGH	NORMAL	34.997	DrugY
194	46	F	HIGH	HIGH	34.686	DrugY

In [ ]:
```
drug.drop(outliers.index.tolist(),axis=0,inplace=True)
```

In [ ]:
```
sns.boxplot(x = 'Na_to_K', data=drug, palette="magma", orient="h")
```

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



Outliers have been removed

Feature Engineering

Encoding categorical variables

In [ ]:
```
{column: len(drug[column].unique()) for column in drug.columns}
```

Out[ ]:
```
{'Age': 56, 'Sex': 2, 'BP': 3, 'Cholesterol': 2, 'Na_to_K': 190, 'Drug': 5}
```

In [ ]:
```
### Using binary encoding on sex and cholesterol columms because both of them consist o
def binary_encode(data, col, value):
    drugc = drug.copy()
    drugc[col] = drugc[col].apply(lambda x: 1 if x == value else 0)
    return drugc
```

In [ ]:
```
drug = binary_encode(drug, 'Sex', 'M')
drug = binary_encode(drug, 'Cholesterol', 'HIGH')
```

In [ ]:
```
### The categorical features are ordinal in this dataset, so I decided to use label enc
from sklearn import preprocessing
label_e = preprocessing.LabelEncoder()
drug['BP'] = label_e.fit_transform(drug['BP'])
```

Now our dataset have all numeric values, and we can start our model building

In [ ]:
```
scaled_data = drug.copy()
scaled_data.head(10)
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	0	0	1	25.355	DrugY
1	47	1	1	1	13.093	drugC

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
2	47	1	1	1	10.114	drugC
3	28	0	2	1	7.798	drugX
4	61	0	1	1	18.043	DrugY
5	22	0	2	1	8.607	drugX
6	49	0	2	1	16.275	DrugY
7	41	1	1	1	11.037	drugC
8	60	1	2	1	15.171	DrugY
9	43	1	1	0	19.368	DrugY

Building and Testing the Model

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

Dataset splitted into training and testing

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
X_train
```

Out[ ]:

	Age	Sex	BP	Cholesterol	Na_to_K
17	43	1	0	1	13.972
143	74	1	0	0	15.436
80	60	1	0	1	13.934
106	22	1	2	1	11.953
130	70	0	2	1	20.489
...	...	...	...	...	...
72	24	0	2	1	10.605
42	50	1	2	0	15.790
34	53	1	2	1	14.133
25	28	0	0	0	18.809
56	65	1	0	0	11.340

153 rows × 5 columns

In [ ]:
```
###Function to print accuracy, precision, and recall score of differnet models###
def algo_accuracy(ytest, pred):
    acc = accuracy_score(ytest, pred)
    prec = precision_score(ytest, pred, average='macro')
    rec = recall_score(ytest, pred, average='macro')
```

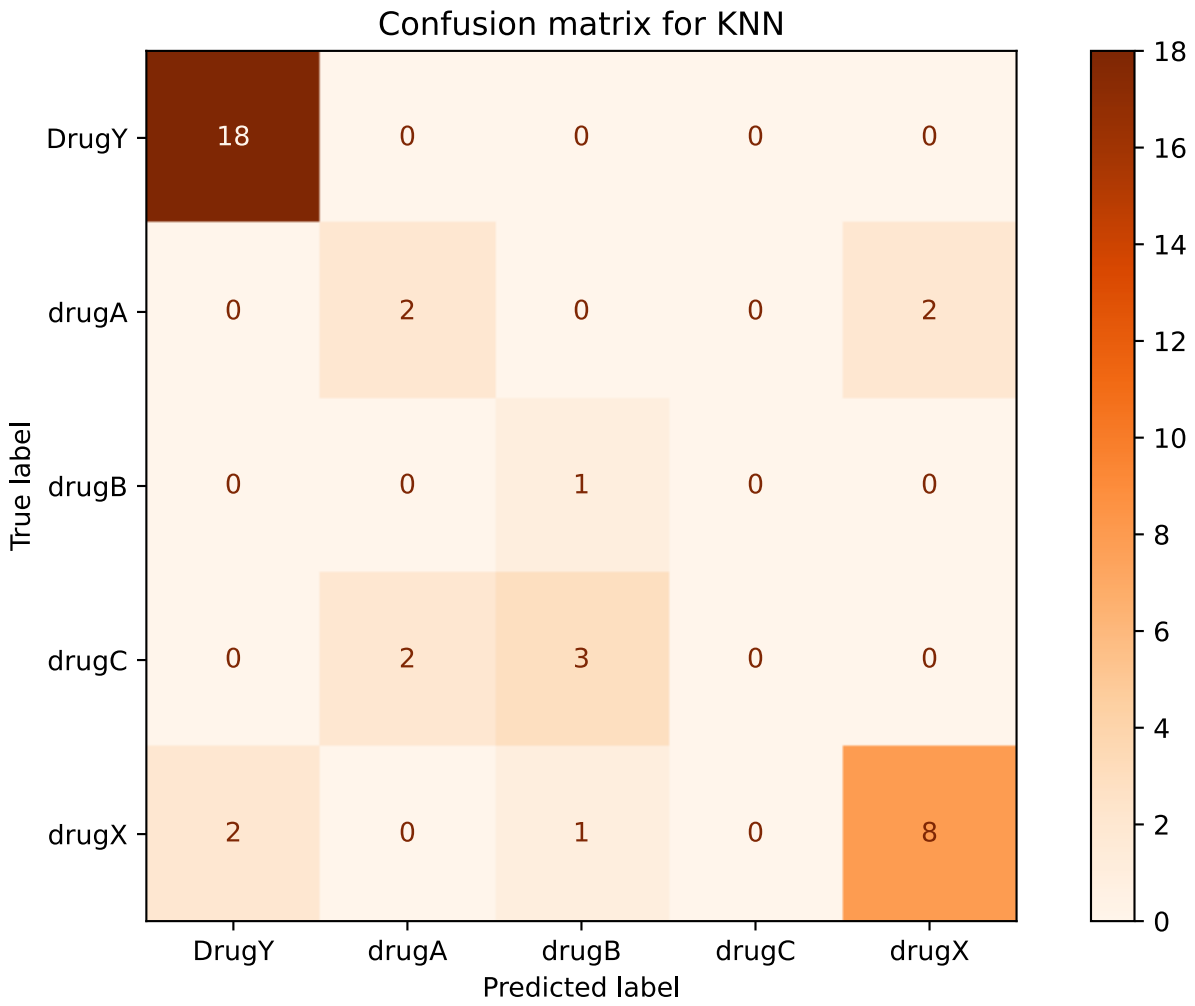
```
    return print('Accuracy score: ', acc*100,
        '\nPrecision score: ', prec*100,
        '\nRecall score: ', rec*100)
```

In [ ]:
```
from sklearn.metrics import accuracy_score, recall_score, precision_score
from sklearn.neighbors import KNeighborsClassifier
knc = KNeighborsClassifier(n_neighbors=3)
knc.fit(X_train, y_train)
y_knn_pred = knc.predict(X_test)

print("Kneighbor Classifier: ")
algo_accuracy(y_test, y_knn_pred)
```

```
Kneighbor Classifier:
Accuracy score:  74.35897435897436
Precision score:  48.0
Recall score:  64.54545454545455
C:\Users\mzuba\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: Unde
finedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```
from sklearn.metrics import plot_confusion_matrix
matrix = plot_confusion_matrix(knc, X_test, y_test, cmap=plt.cm.Oranges)
matrix.ax_.set_title("Confusion matrix for KNN", color='black')
plt.xlabel('Predicted label', color = 'black')
plt.ylabel('True label', color = 'black')
plt.gcf().axes[0].tick_params(colors='black')
plt.gcf().axes[1].tick_params(colors='black')
plt.gcf().set_size_inches(10,6)
plt.show()
```

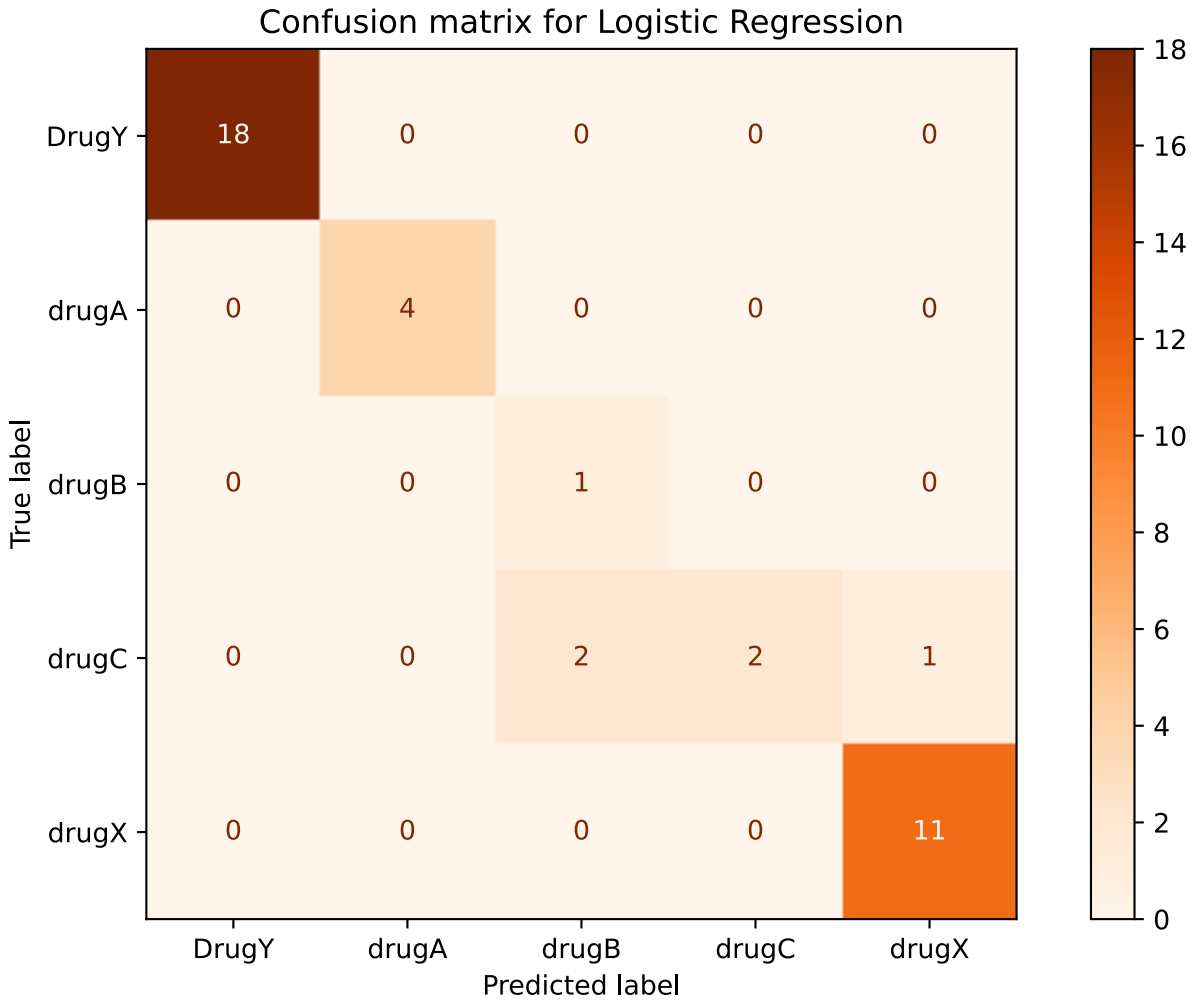


In [ ]:
```
from sklearn import linear_model
log_reg = linear_model.LogisticRegression(max_iter = 5000)
log_reg.fit(X_train, y_train)
y_log_pred = log_reg.predict(X_test)

print("Logistic Regression:")
algo_accuracy(y_test, y_log_pred)
```

```
Logistic Regression:
Accuracy score:  92.3076923076923
Precision score:  85.0
Recall score:  88.00000000000001
```

In [ ]:
```
matrix = plot_confusion_matrix(log_reg, X_test, y_test, cmap=plt.cm.Oranges)
matrix.ax_.set_title("Confusion matrix for Logistic Regression", color='black')
plt.xlabel('Predicted label', color = 'black')
plt.ylabel('True label', color = 'black')
plt.gcf().axes[0].tick_params(colors='black')
plt.gcf().axes[1].tick_params(colors='black')
plt.gcf().set_size_inches(10,6)
plt.show()
```



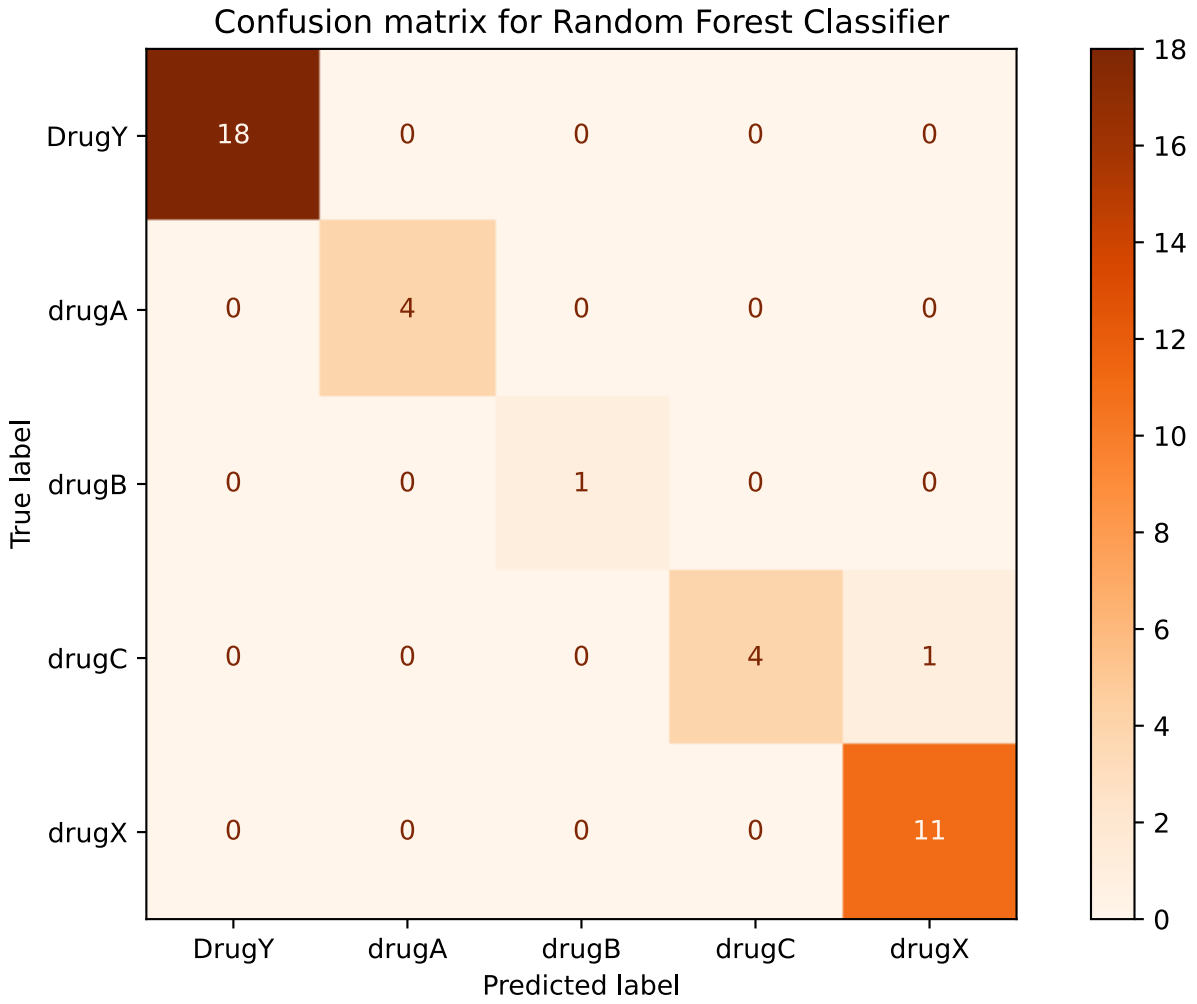
In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(random_state = 42)
rfc.fit(X_train,y_train)
y_rfc_pred = rfc.predict(X_test)

print("Random Forest Classifier:")
algo_accuracy(y_test, y_rfc_pred)
```

```
Random Forest Classifier:
Accuracy score:  97.43589743589743
Precision score:  98.33333333333334
Recall score:  96.0
```

In [ ]:
```
matrix = plot_confusion_matrix(rfc, X_test, y_test, cmap=plt.cm.Oranges)
matrix.ax_.set_title("Confusion matrix for Random Forest Classifier", color='black')
plt.xlabel('Predicted label', color = 'black')
plt.ylabel('True label', color = 'black')
plt.gcf().axes[0].tick_params(colors='black')
plt.gcf().axes[1].tick_params(colors='black')
plt.gcf().set_size_inches(10,6)
plt.show()
```


Confusion matrix for Random Forest Classifier
DrugY
drugA
drugB
drugC
drugX
True label
18 0 0 0 0
0 4 0 0 0
0 0 1 0 0
0 0 0 4 1
0 0 0 0 11
DrugY
drugA
drugB
drugC
drugX
Predicted label
18
16
14
12
10
8
6
4
2
0