Classification algorithms for classifying drugs

Which is more accurate ? Let's see !!!

```
# Import all required packages for data analysis and visualization
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

# Import the drug dataset
drug_data = pd.read_csv('drug200.csv')
drug_data.head()
```

```
   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
```

```
# Statistical measures of the drug dataset
drug_data.describe()
```

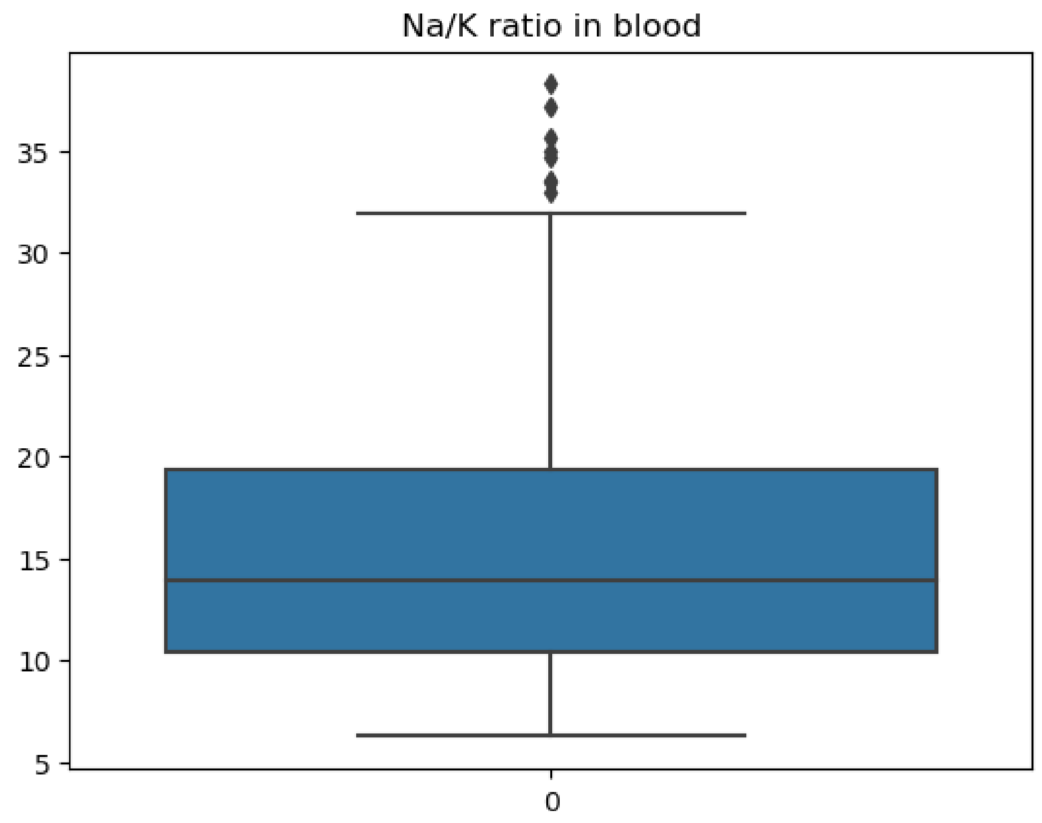
```
              Age     Na_to_K
count  200.000000  200.000000
mean    44.315000   16.084485
std     16.544315    7.223956
min     15.000000    6.269000
25%     31.000000   10.445500
50%     45.000000   13.936500
75%     58.000000   19.380000
max     74.000000   38.247000
```

```
# Concise summary of the drug dataset
drug_data.info()
```

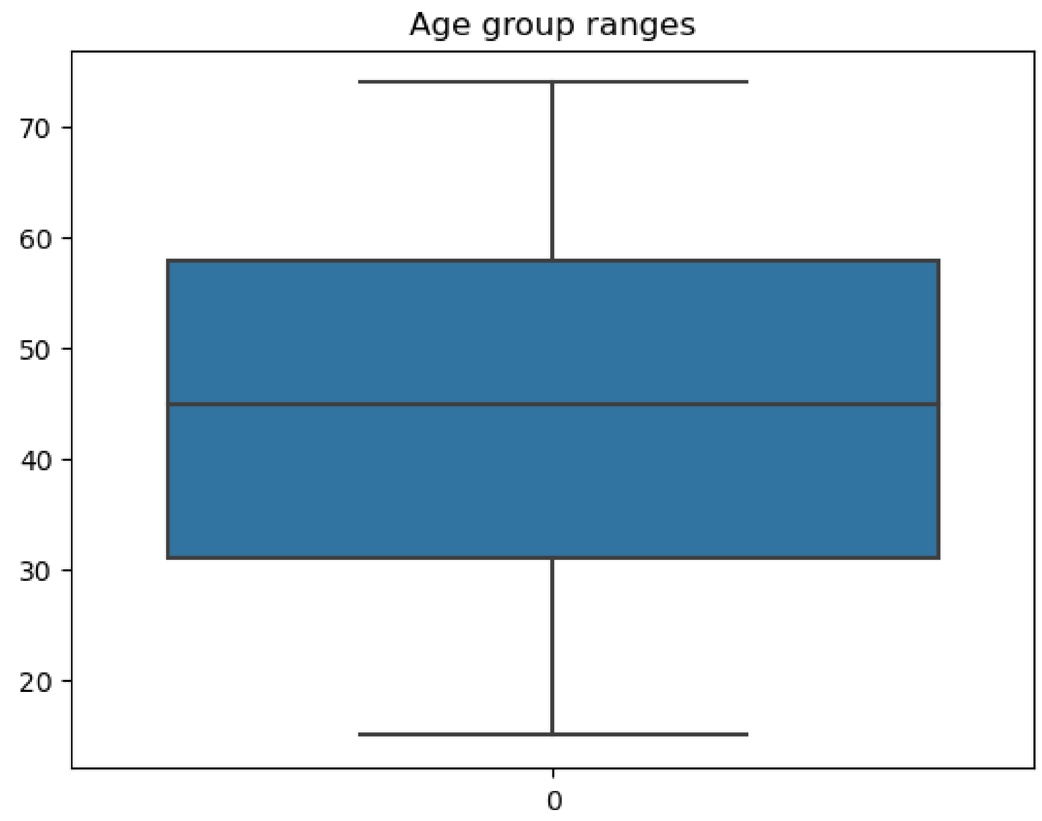
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null    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

# Check for duplicate sum in the drug_data
drug_data.duplicated().sum()

0

# Check for duplicates - column wise
drug_data.isna().sum()

Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64

# Displays the datatypes of the columns in a drug dataset
drug_data.dtypes

Age              int64
Sex             object
BP              object
Cholesterol     object
Na_to_K        float64
Drug            object
dtype: object

# Number of records in the drug dataset
len(drug_data)

200

count = drug_data['Cholesterol'].value_counts()
count

HIGH      103
NORMAL     97
Name: Cholesterol, dtype: int64

count = drug_data['BP'].value_counts()
count

HIGH      77
LOW       64
NORMAL    59
Name: BP, dtype: int64

count = drug_data['Sex'].value_counts()
count
```

```
M    104
F     96
Name: Sex, dtype: int64

# Range of Na/K ratio of patients in drug_data
ax = sns.boxplot(drug_data['Na_to_K'])
ax.set_title(" Na/K ratio in blood ")
plt.show(ax)
```



```
# Range of patients with age
ax = sns.boxplot(drug_data['Age'])
ax.set_title(" Age group ranges ")
plt.show(ax)
```



```
#Extract X and y from the dataframe , income column is the target
column, rest columns are features
X = drug_data.loc[:,['Age','Sex','BP','Cholesterol','Na_to_K']]
y = drug_data.loc[:,'Drug']

X

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

[200 rows x 5 columns]

y
```

```
0      DrugY
1      drugC
2      drugC
3      drugX
4      DrugY
       ...
195    drugC
196    drugC
197    drugX
198    drugX
199    drugX
Name: Drug, Length: 200, dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
y = LabelEncoder().fit_transform(y)
y = pd.DataFrame(y)
y.head()
```

```
   0
0  0
1  3
2  3
3  4
4  0
```

```
#First identify caterogical features and numeric features
numeric_features = X.select_dtypes('number')
categorical_features = X.select_dtypes('object')
categorical_features
```

```
    Sex      BP Cholesterol
0     F    HIGH        HIGH
1     M     LOW        HIGH
2     M     LOW        HIGH
3     F  NORMAL        HIGH
4     F     LOW        HIGH
..   ..     ...         ...
195   F     LOW        HIGH
196   M     LOW        HIGH
197   M  NORMAL        HIGH
198   M  NORMAL      NORMAL
199   F     LOW      NORMAL

[200 rows x 3 columns]
```

```
numeric_features
```

```
     Age  Na_to_K
0     23   25.355
1     47   13.093
2     47   10.114
```

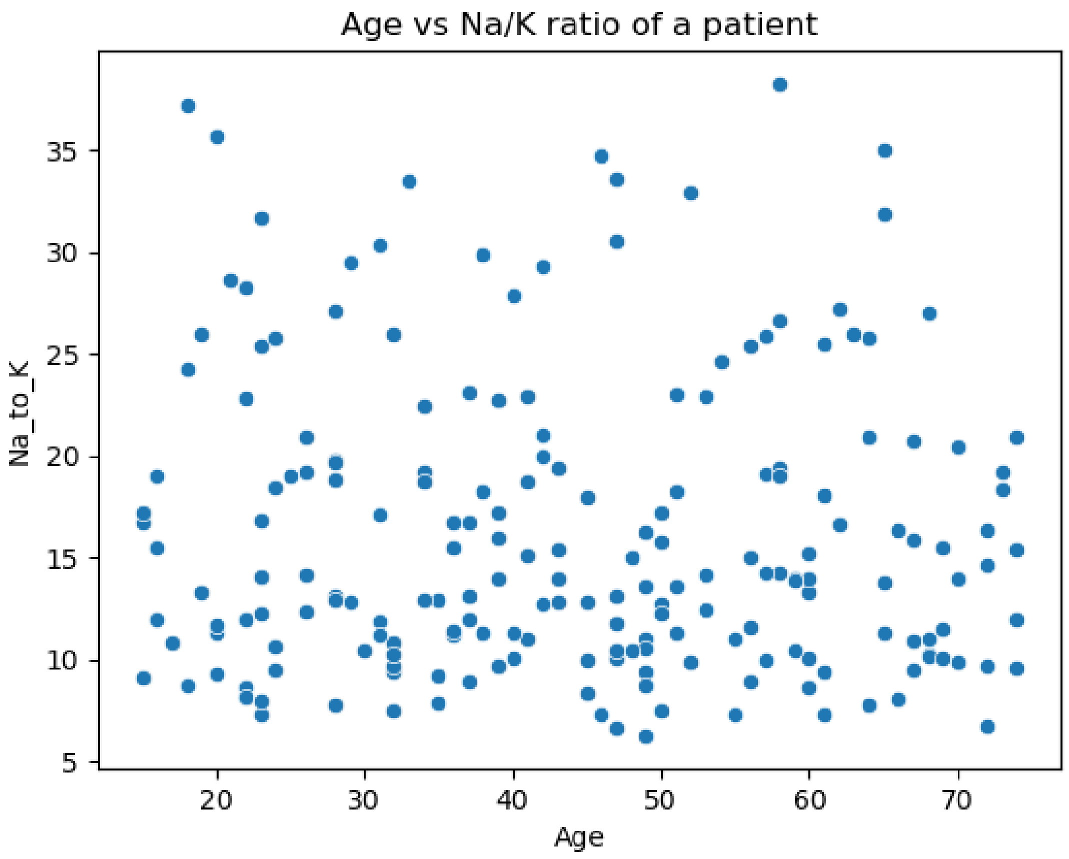
```
3     28    7.798
4     61   18.043
..   ...      ...
195   56   11.567
196   16   12.006
197   52    9.894
198   23   14.020
199   40   11.349

[200 rows x 2 columns]

#Method 1: Convert categorical features into numeric
converted_categorical_features = pd.get_dummies(categorical_features)
converted_categorical_features.shape
converted_categorical_features

     Sex_F  Sex_M  BP_HIGH  BP_LOW  BP_NORMAL  Cholesterol_HIGH  \
0        1      0        1       0          0                 1
1        0      1        0       1          0                 1
2        0      1        0       1          0                 1
3        1      0        0       0          1                 1
4        1      0        0       1          0                 1
..     ...    ...      ...     ...        ...               ...
195      1      0        0       1          0                 1
196      0      1        0       1          0                 1
197      0      1        0       0          1                 1
198      0      1        0       0          1                 0
199      1      0        0       1          0                 0

     Cholesterol_NORMAL
0                     0
1                     0
2                     0
3                     0
4                     0
..                  ...
195                   0
196                   0
197                   0
198                   1
199                   1

[200 rows x 7 columns]

#combine the converted categorical features and the numeric features
together into a new dataframe called "newX"
all_features = [converted_categorical_features, numeric_features]
newX = pd.concat(all_features,axis=1, join='inner')
newX.shape
```

```
(200, 9)
```

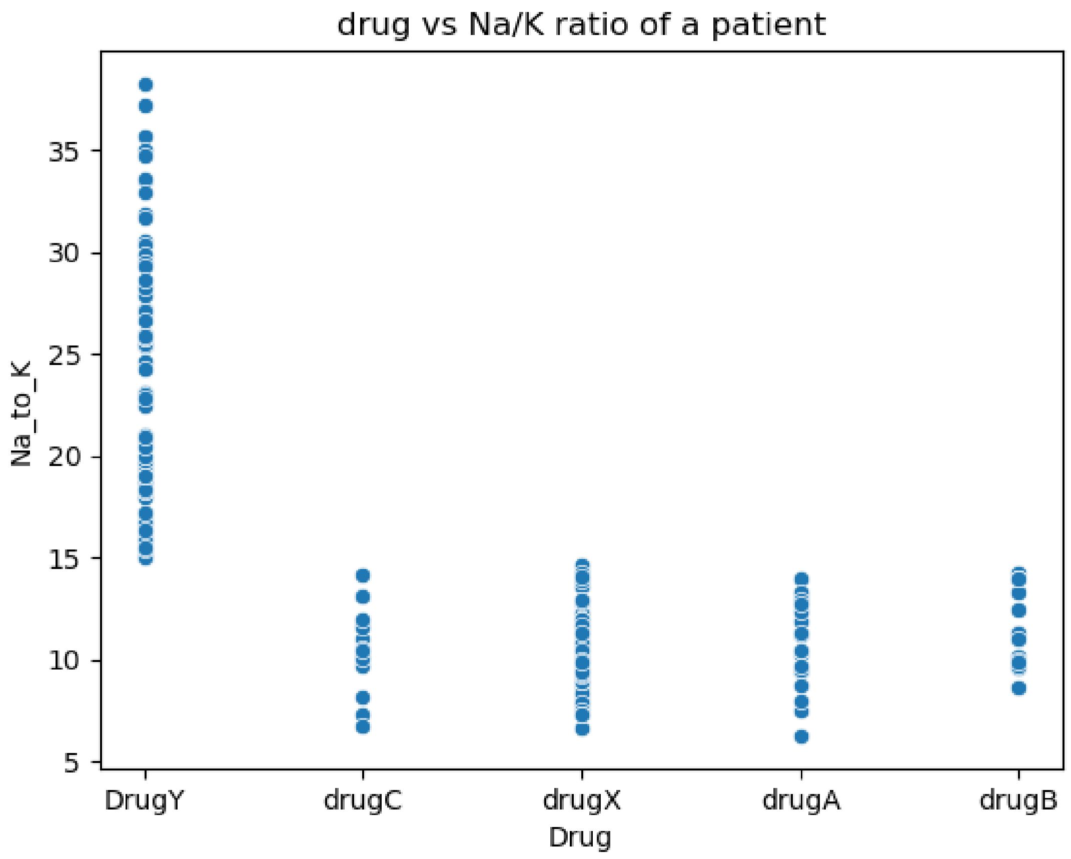
```
# Method 2:
# converting categorical drug_data into numerical drug_data
# from sklearn.preprocessing import LabelEncoder
# labelencoder = LabelEncoder()
# drug_data.iloc[:,5] =
labelencoder.fit_transform(drug_data.iloc[:,5])
# drug_data["Sex"] = drug_data["Sex"].replace({"M": 1, "F": 0})
# drug_data["BP"] = drug_data["BP"].replace({"HIGH": 1,"LOW": 0,
"NORMAL":0.5})
# drug_data["Cholesterol"] = drug_data["Cholesterol"].replace({"HIGH":
1, "LOW": 0, "NORMAL": 0.5})
# drug_data

# # Check the data
# drug_data.head()

newX.columns
```

```
Index(['Sex_F', 'Sex_M', 'BP_HIGH', 'BP_LOW', 'BP_NORMAL',
'Cholesterol_HIGH',
       'Cholesterol_NORMAL', 'Age', 'Na_to_K'],
      dtype='object')
```

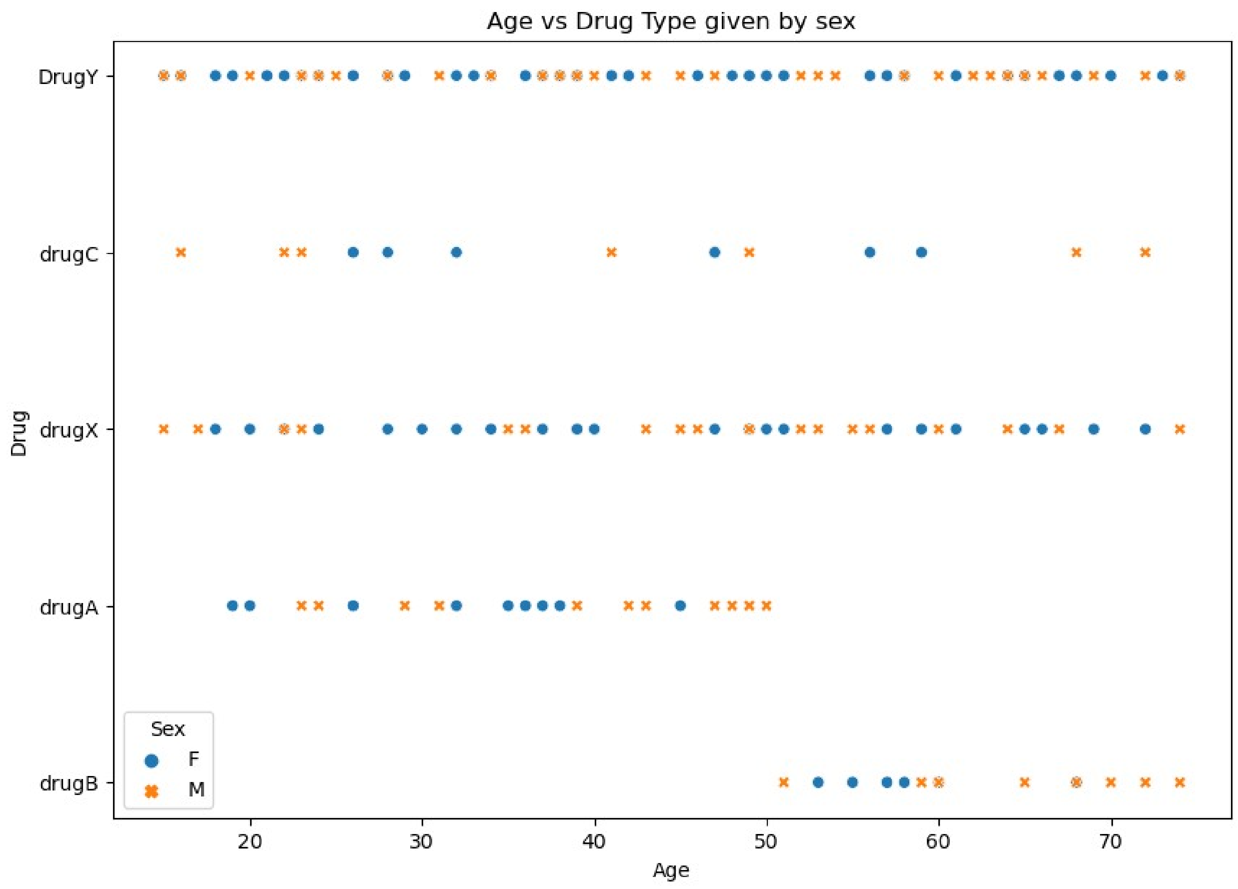
```
# Some of the visualizations regarding patient features for drug
analysis

ax = sns.scatterplot(x = 'Age', y='Na_to_K', data = drug_data)
ax.set_title('Age vs Na/K ratio of a patient')
plt.show(ax)
```



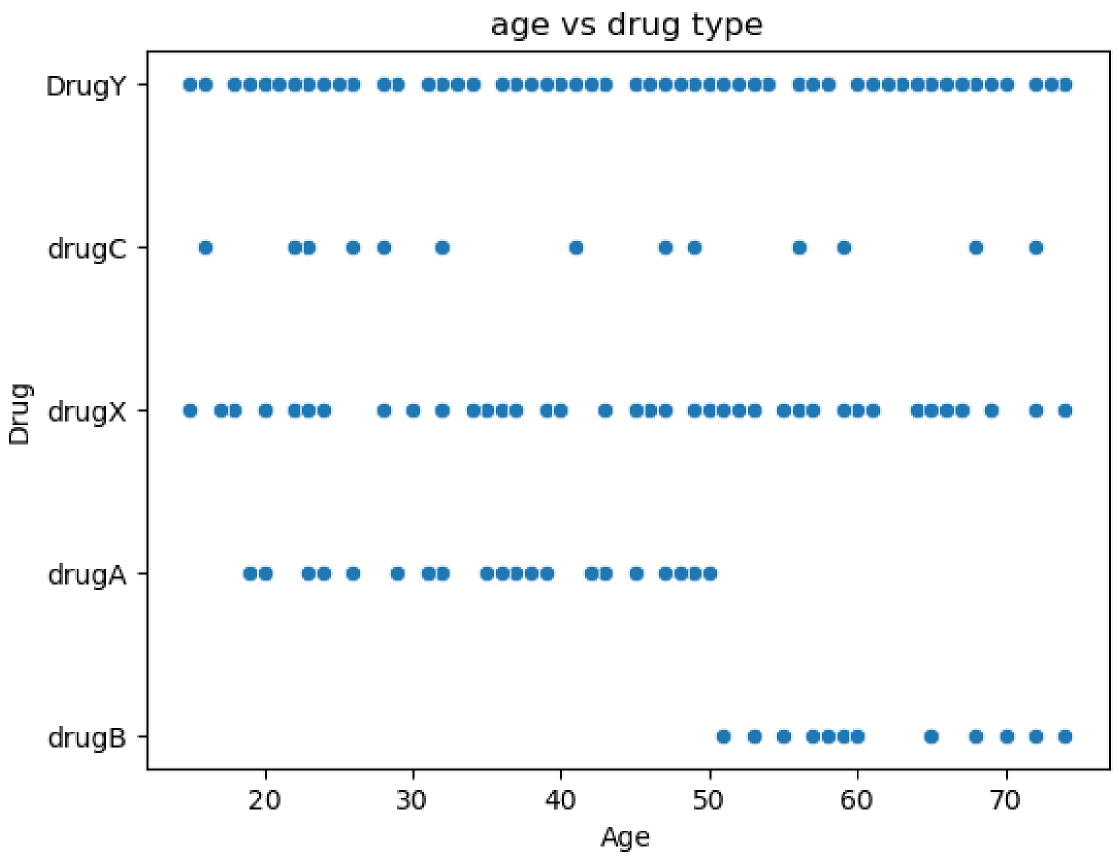
```
ax = sns.scatterplot(x = 'Drug', y='Na_to_K', data = drug_data)
ax.set_title('drug vs Na/K ratio of a patient')
plt.show(ax)
```



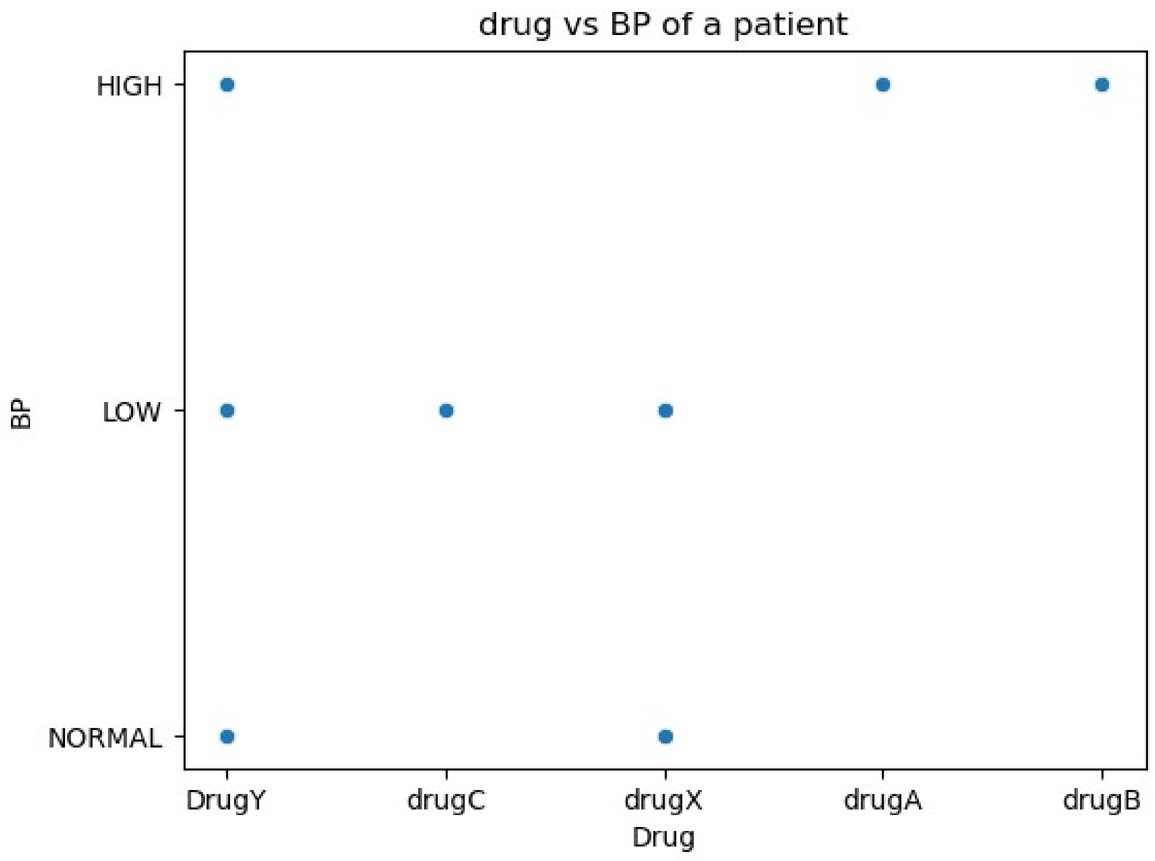
```
#Age with gender does not influence drug type given to a patient.
plt.figure(figsize=(10, 7))
ax = sns.scatterplot(x = 'Age', y='Drug',
                     hue='Sex', style='Sex', data=drug_data)
ax.set_title("Age vs Drug Type given by sex")
plt.show(ax)
```



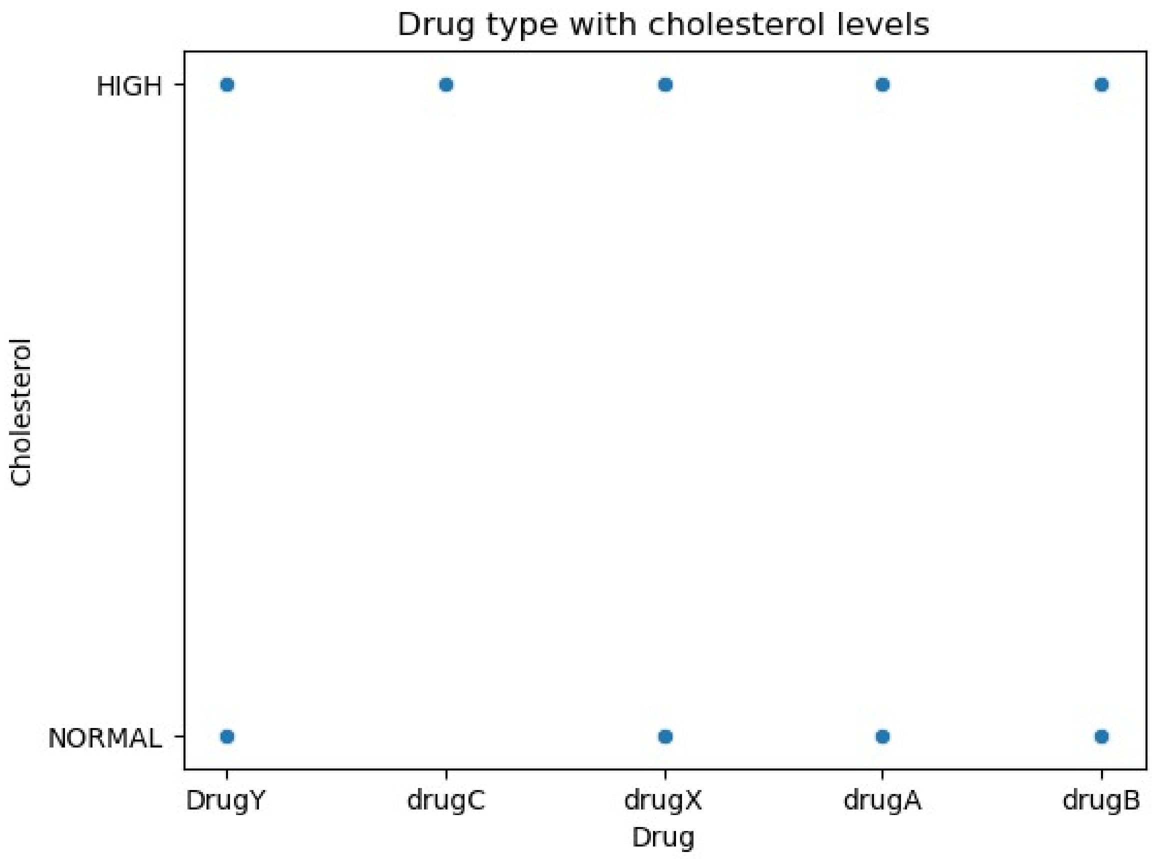
```
ax = sns.scatterplot(x = 'Age', y='Drug', data = drug_data)
ax.set_title('age vs drug type')
plt.show(ax)
```



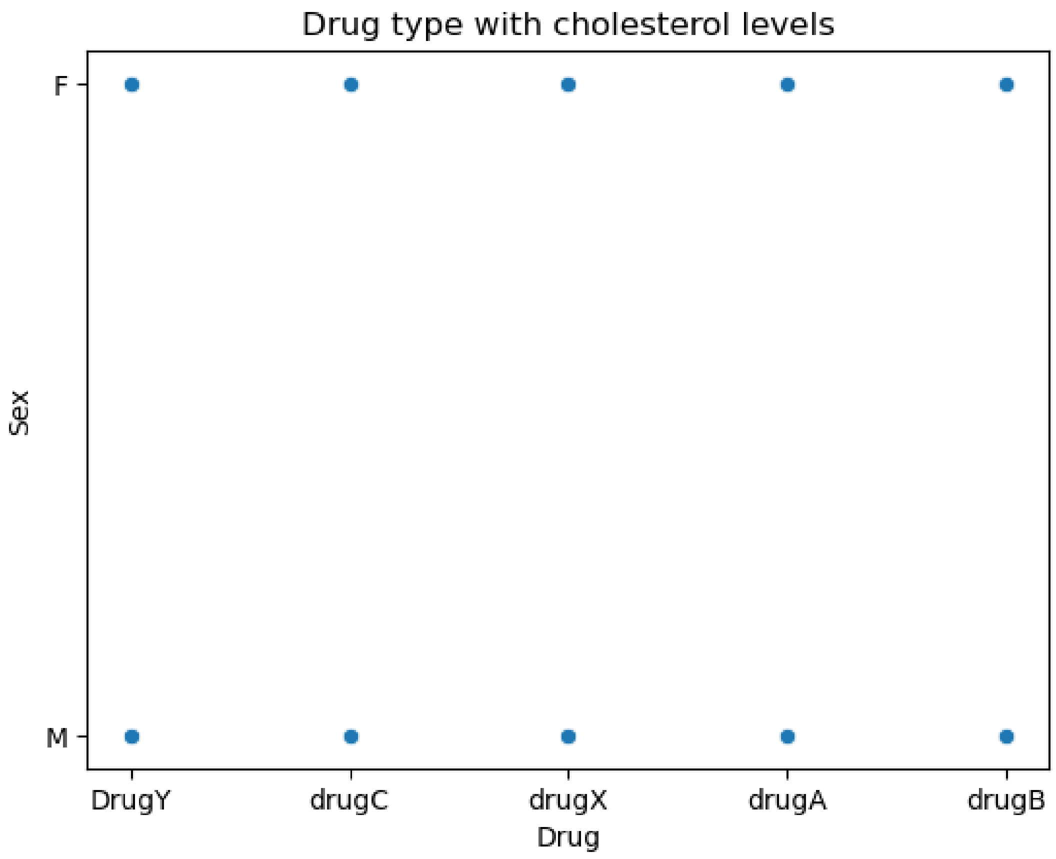
```
#It tells us that only people who have high BP are recommended
#to take drugA and drugB, similarly drugC with Low BP.
ax = sns.scatterplot(x = 'Drug', y='BP', data = drug_data)
ax.set_title('drug vs BP of a patient')
plt.show(ax)
```



```
# It is suggested that patients with normal cholesterol
#are not recommended with drugC.
ax = sns.scatterplot(x = 'Drug', y='Cholesterol', data = drug_data)
ax.set_title('Drug type with cholesterol levels')
plt.show(ax)
```

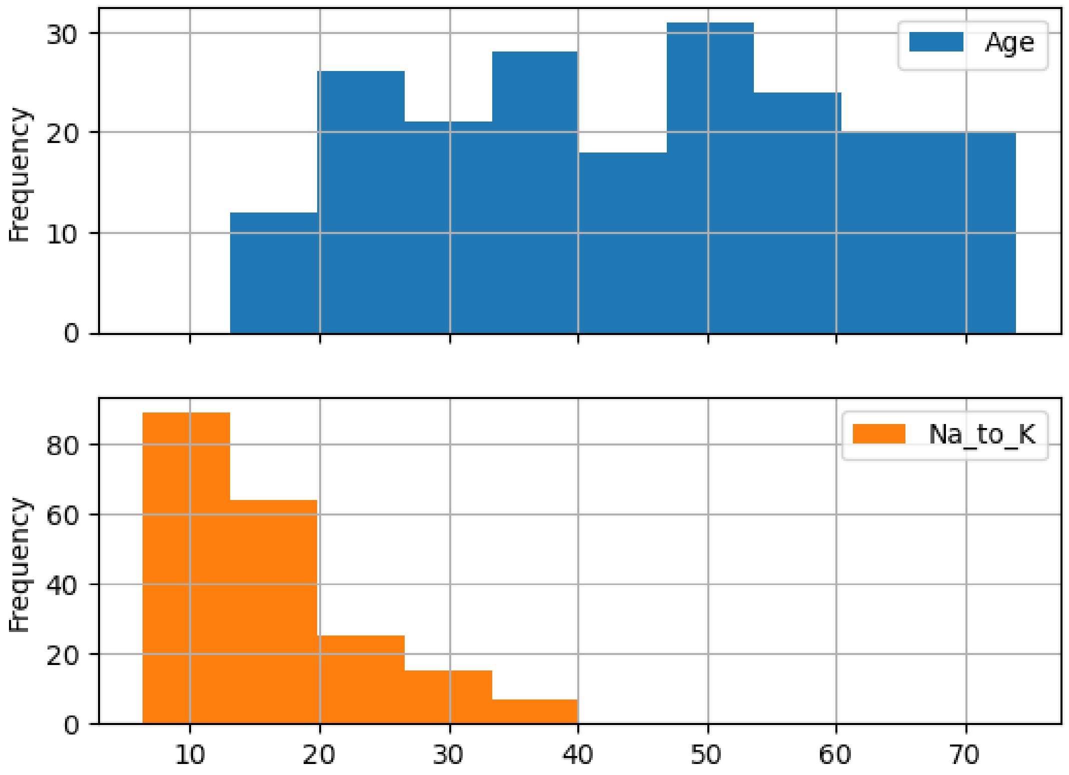


```
# Irrespective of gender the drug types are classified
ax = sns.scatterplot(x = 'Drug', y='Sex', data = drug_data)
ax.set_title('Drug type with cholesterol levels')
plt.show(ax)
```

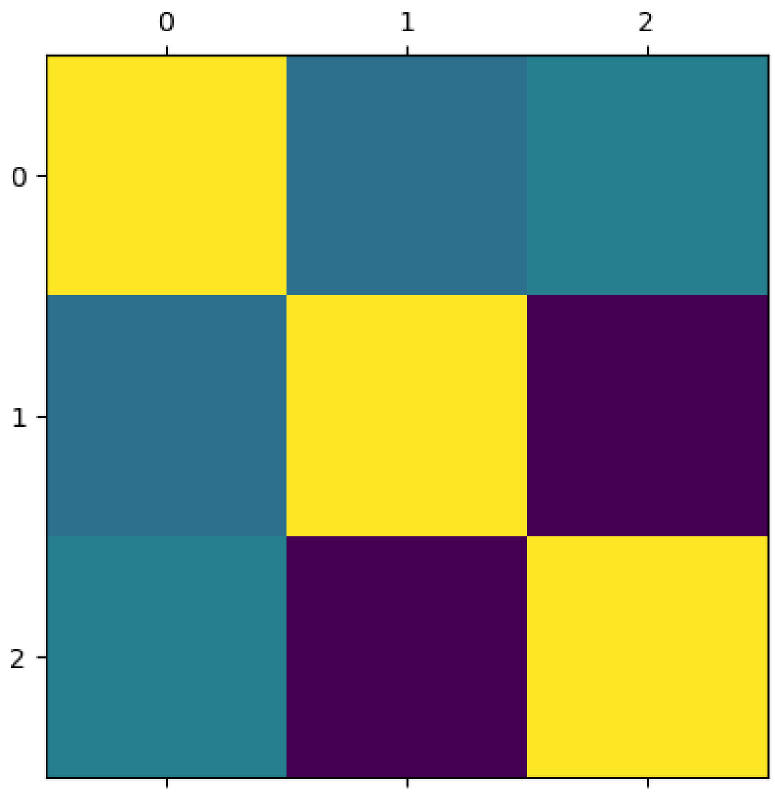


```
# By this analysis both Age and Na/K ratio are not correlated
drug_data.plot.hist( subplots = True, grid = True)
```

```
array([<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
      dtype=object)
```



```
import warnings
warnings.filterwarnings('ignore')

plt.matshow(drug_data.corr())
plt.show()
```



```python
# Splitting the drug dataset
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(newX, y,
test_size=0.2)

# RandomForestClassifier for drug classification
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
1.0
```

```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
1.0
```

```python
from sklearn.naive_bayes import GaussianNB

naive_bayes = GaussianNB()
```

```
naive_bayes.fit(X_train , y_train)
y_predicted = naive_bayes.predict(X_test)

#Import metrics class from sklearn
from sklearn import metrics

metrics.accuracy_score(y_predicted , y_test)

0.725
```

We can conclude that RandomForestClassifier and DecisionTreeClassifier are the best fit algorithms for classifying drugs with 100% accuracy for the drug dataset taken above.