```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import  train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import
confusion_matrix,roc_auc_score,classification_report

#importing datasets
df= pd.read_csv('drug200.csv')

df.head()
```

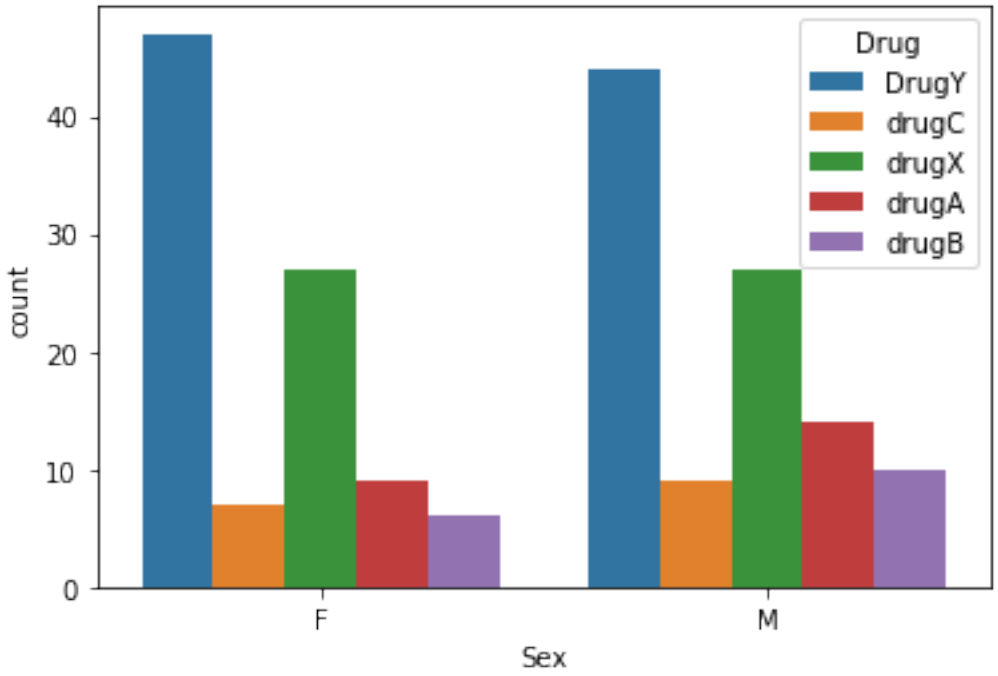
	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

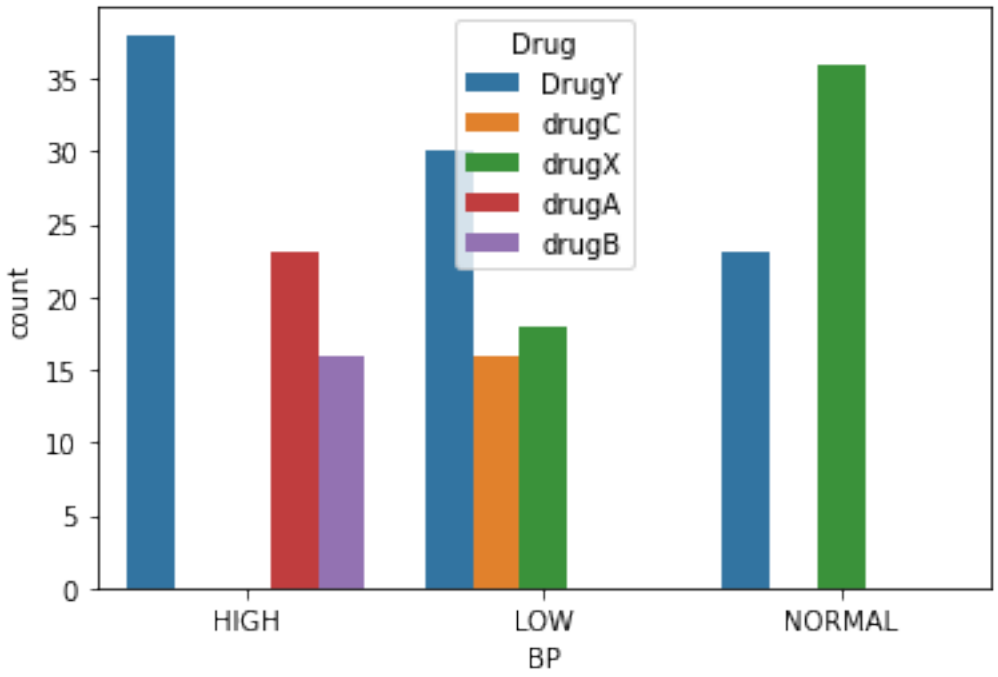
```
df.info()
```

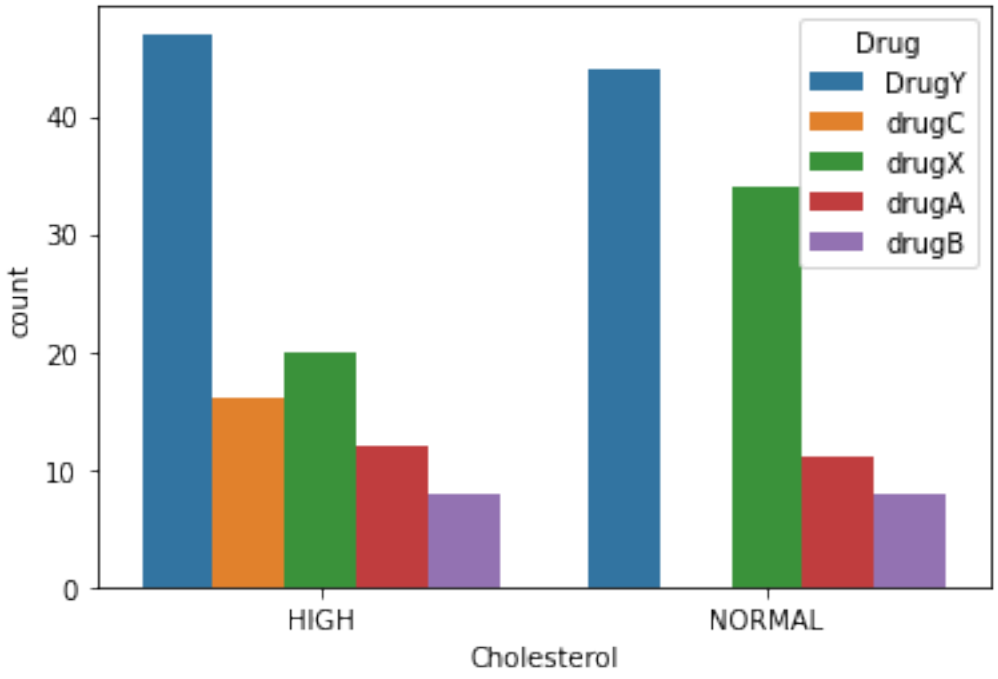
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          200 non-null    int64
 1   Sex          200 non-null    object
 2   BP           200 non-null 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

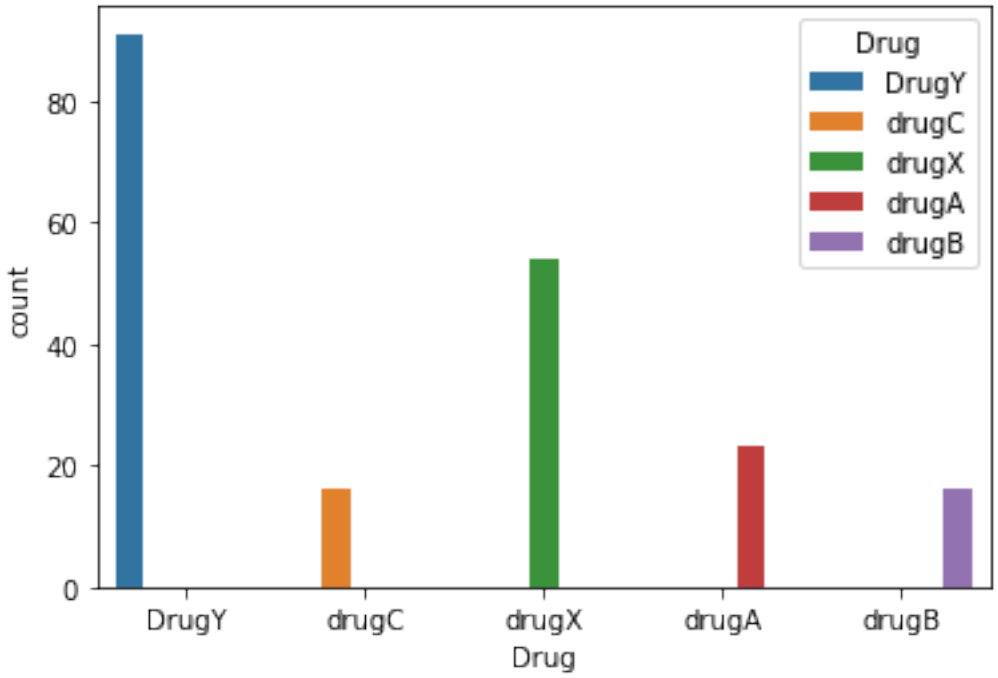
```
#categorical columns
categ_column=[column for column in df.columns if
df[column].dtype=='object']
#numerical columns
numeric_column=df.drop(categ_column,axis=1).columns

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

```
for column in categ_column:
    sns.countplot(df[column],hue=df['Drug'])
    plt.show()
```

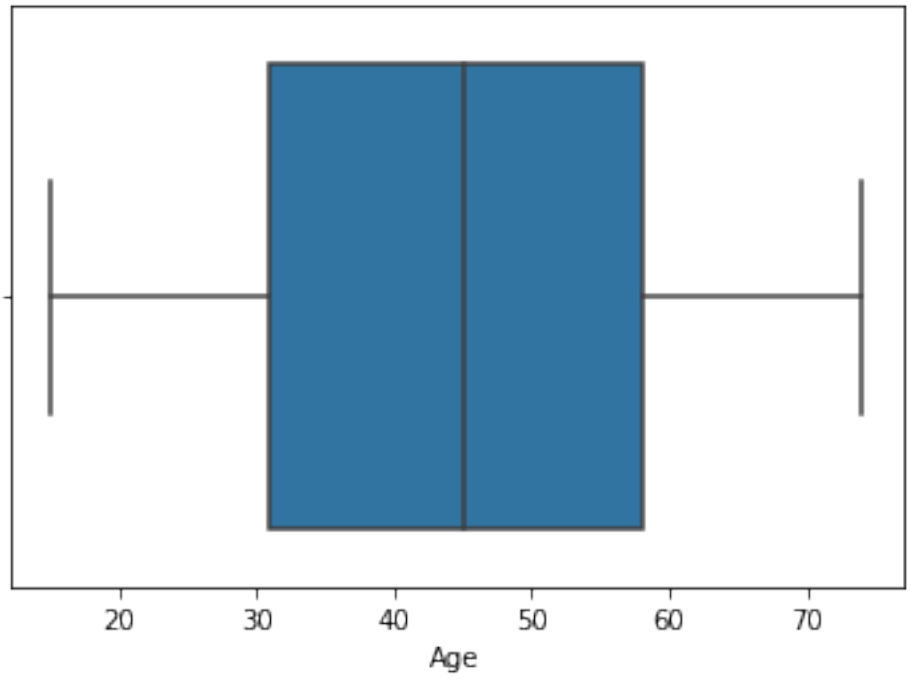


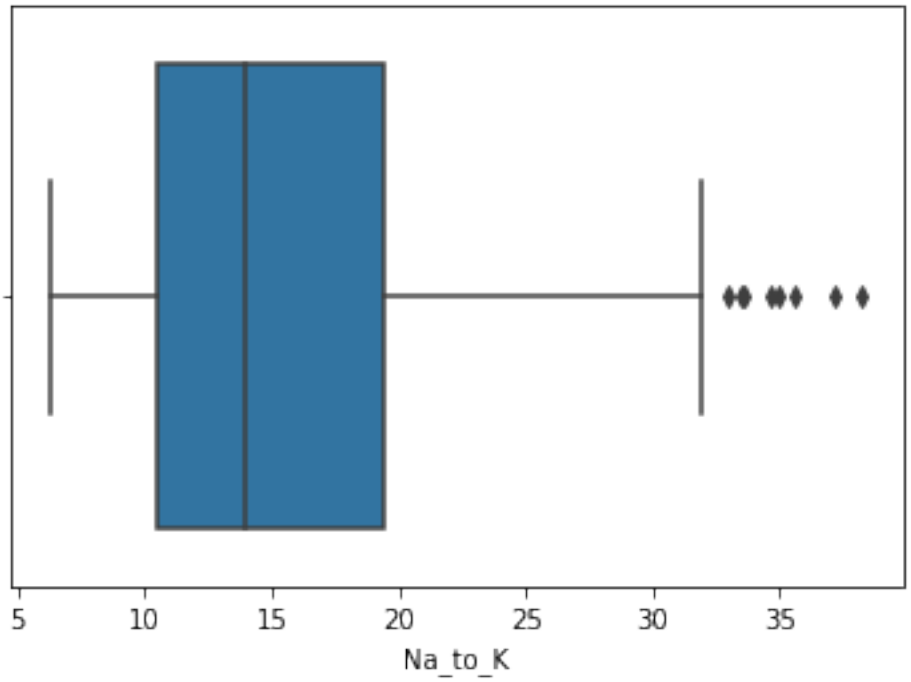







```
for column in numeric_column:
    sns.boxplot(df[column])
    plt.show()
```

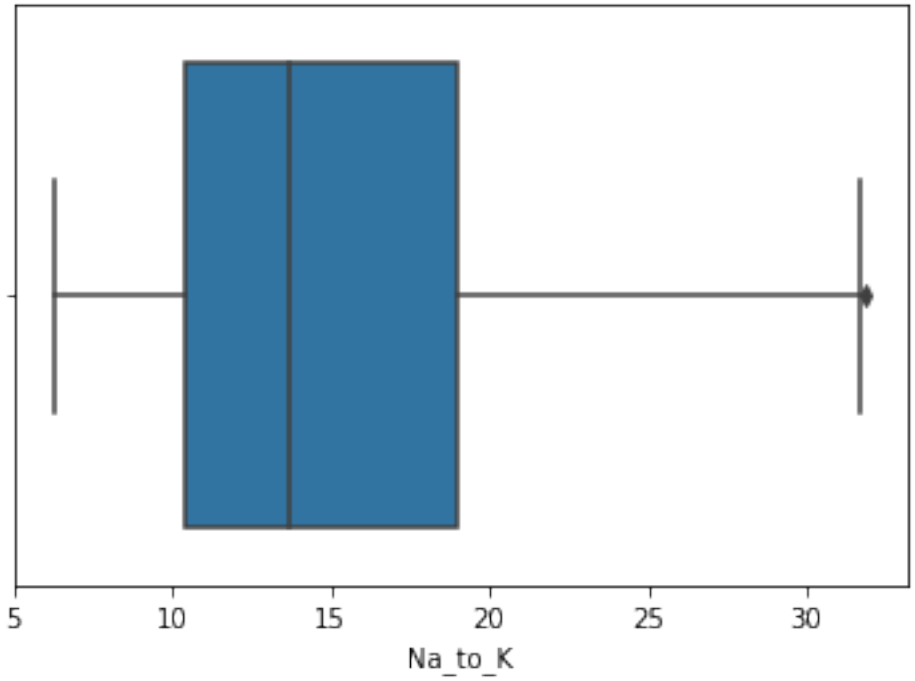





```
#first quartile
Q1=df['Na_to_K'].quantile(0.25)
# 3rd quartile
Q3=df['Na_to_K'].quantile(0.75)
#Inter quartile range
IQR=Q3-Q1
#upper fence = Q3+1.5*IQR
upper=Q3+ 1.5*IQR
```

```
# lower fence= Q1-1.5*IQR
lower= Q1 - 1.5*IQR

#putting conditions
df=df[(df['Na_to_K']<upper) & (df['Na_to_K']>lower)]

sns.boxplot(df['Na_to_K'])
```

```
<AxesSubplot:xlabel='Na_to_K'>
```




```
for column in categ_column:
    le=LabelEncoder()
    df[column]=le.fit_transform(df[column])

df.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	0	0	0	25.355	0
1	47	1	1	0	13.093	3
2	47	1	1	0	10.114	3
3	28	0	2	0	7.798	4
4	61	0	1	0	18.043	0

```
df.shape
```

```
(192, 6)
```

```
x = df.iloc[:,:-1].values
y = df.iloc[:, 1].values
```

```
print(x)
```

```
[[23.     0.     0.     0.    25.355]
 [47.     1.     1.     0.    13.093]
 [47.     1.     1.     0.    10.114]
 [28.     0.     2.     0.     7.798]
 [61.     0.     1.     0.    18.043]
 [22.     0.     2.     0.     8.607]
 [49.     0.     2.     0.    16.275]
 [41.     1.     1.     0.    11.037]
 [60.     1.     2.     0.    15.171]
 [43.     1.     1.     1.    19.368]
 [47.     0.     1.     0.    11.767]
 [34.     0.     0.     1.    19.199]
 [43.     1.     1.     0.    15.376]
 [74.     0.     1.     0.    20.942]
 [50.     0.     2.     0.    12.703]
 [16.     0.     0.     1.    15.516]
 [69.     1.     1.     1.    11.455]
 [43.     1.     0.     0.    13.972]
 [23.     1.     1.     0.     7.298]
 [32.     0.     0.     1.    25.974]
 [57.     1.     1.     1.    19.128]
 [63.     1.     2.     0.    25.917]
 [47.     1.     1.     1.    30.568]
 [48.     0.     1.     0.    15.036]
 [28.     0.     0.     1.    18.809]
 [31.     1.     0.     0.    30.366]
 [49.     0.     2.     1.     9.381]
 [39.     0.     1.     1.    22.697]
 [45.     1.     1.     0.    17.951]
 [18.     0.     2.     1.     8.75 ]
 [74.     1.     0.     0.     9.567]
 [49.     1.     1.     1.    11.014]
 [65.     0.     0.     1.    31.876]
 [53.     1.     2.     0.    14.133]
 [46.     1.     2.     1.     7.285]
 [32.     1.     0.     1.     9.445]
 [39.     1.     1.     1.    13.938]
 [39.     0.     2.     1.     9.709]
 [15.     1.     2.     0.     9.084]
 [73.     0.     2.     0.    19.221]
 [58.     0.     0.     1.    14.239]
 [50.     1.     2.     1.    15.79 ]
 [23.     1.     2.     0.    12.26 ]
 [50.     0.     2.     1.    12.295]
 [66.     0.     2.     1.     8.107]
 [37.     0.     0.     0.    13.091]
 [68.     1.     1.     0.    10.291]
 [23.     1.     2.     0.    31.686]
```

```
[28.     0.     1.     0.    19.796]
[58.     0.     0.     0.    19.416]
[67.     1.     2.     1.    10.898]
[62.     1.     1.     1.    27.183]
[24.     0.     0.     1.    18.457]
[68.     0.     0.     1.    10.189]
[26.     0.     1.     0.    14.16 ]
[65.     1.     0.     1.    11.34 ]
[40.     1.     0.     0.    27.826]
[60.     1.     2.     1.    10.091]
[34.     1.     0.     0.    18.703]
[38.     0.     1.     1.    29.875]
[24.     1.     0.     1.     9.475]
[67.     1.     1.     1.    20.693]
[45.     1.     1.     1.     8.37 ]
[60.     0.     0.     0.    13.303]
[68.     0.     2.     1.    27.05 ]
[29.     1.     0.     0.    12.856]
[17.     1.     2.     1.    10.832]
[54.     1.     2.     0.    24.658]
[18.     0.     0.     1.    24.276]
[70.     1.     0.     0.    13.967]
[28.     0.     2.     0.    19.675]
[24.     0.     2.     0.    10.605]
[41.     0.     2.     1.    22.905]
[31.     1.     0.     1.    17.069]
[26.     1.     1.     1.    20.909]
[36.     0.     0.     0.    11.198]
[26.     0.     0.     1.    19.161]
[19.     0.     0.     0.    13.313]
[32.     0.     1.     1.    10.84 ]
[60.     1.     0.     0.    13.934]
[64.     1.     2.     0.     7.761]
[32.     0.     1.     0.     9.712]
[38.     0.     0.     1.    11.326]
[47.     0.     1.     0.    10.067]
[59.     1.     0.     0.    13.935]
[51.     0.     2.     0.    13.597]
[69.     1.     1.     0.    15.478]
[37.     0.     0.     1.    23.091]
[50.     0.     2.     1.    17.211]
[62.     1.     2.     0.    16.594]
[41.     1.     0.     1.    15.156]
[29.     0.     0.     0.    29.45 ]
[42.     0.     1.     1.    29.271]
[56.     1.     1.     0.    15.015]
[36.     1.     1.     1.    11.424]
[56.     0.     0.     0.    25.395]
[15.     0.     0.     1.    16.725]
[31.     1.     0.     1.    11.871]
```

```
[45.     0.     0.     0.    12.854]
[28.     0.     1.     0.    13.127]
[56.     1.     2.     0.     8.966]
[22.     1.     0.     1.    28.294]
[37.     1.     1.     1.     8.968]
[22.     1.     2.     0.    11.953]
[42.     1.     1.     0.    20.013]
[72.     1.     0.     1.     9.677]
[23.     1.     2.     0.    16.85 ]
[50.     1.     0.     0.     7.49 ]
[47.     0.     2.     1.     6.683]
[35.     1.     1.     1.     9.17 ]
[65.     0.     1.     1.    13.769]
[20.     0.     2.     1.     9.281]
[51.     1.     0.     0.    18.295]
[67.     1.     2.     1.     9.514]
[40.     0.     2.     0.    10.103]
[32.     0.     0.     1.    10.292]
[61.     0.     0.     0.    25.475]
[28.     1.     2.     0.    27.064]
[15.     1.     0.     1.    17.206]
[34.     1.     2.     0.    22.456]
[36.     0.     2.     0.    16.753]
[53.     0.     0.     1.    12.495]
[19.     0.     0.     1.    25.969]
[66.     1.     0.     0.    16.347]
[35.     1.     2.     1.     7.845]
[32.     0.     2.     0.     7.477]
[70.     0.     2.     0.    20.489]
[49.     1.     1.     1.    13.598]
[24.     1.     2.     0.    25.786]
[42.     0.     0.     0.    21.036]
[74.     1.     1.     1.    11.939]
[55.     0.     0.     0.    10.977]
[35.     0.     0.     0.    12.894]
[51.     1.     0.     1.    11.343]
[69.     0.     2.     0.    10.065]
[49.     1.     0.     1.     6.269]
[64.     0.     1.     1.    25.741]
[60.     1.     0.     1.     8.621]
[74.     1.     0.     1.    15.436]
[39.     1.     0.     0.     9.664]
[61.     1.     2.     0.     9.443]
[37.     0.     1.     1.    12.006]
[26.     0.     0.     1.    12.307]
[61.     0.     1.     1.     7.34 ]
[22.     1.     1.     0.     8.151]
[49.     1.     0.     1.     8.7  ]
[68.     1.     0.     0.    11.009]
[55.     1.     2.     1.     7.261]
```

```
 [72.     0.     1.     1.    14.642]
 [37.     1.     1.     1.    16.724]
 [49.     1.     1.     0.    10.537]
 [31.     1.     0.     1.    11.227]
 [53.     1.     1.     0.    22.963]
 [59.     0.     1.     0.    10.444]
 [34.     0.     1.     1.    12.923]
 [30.     0.     2.     0.    10.443]
 [57.     0.     0.     1.     9.945]
 [43.     1.     2.     1.    12.859]
 [21.     0.     0.     1.    28.632]
 [16.     1.     0.     1.    19.007]
 [38.     1.     1.     0.    18.295]
 [58.     0.     1.     0.    26.645]
 [57.     0.     2.     0.    14.216]
 [51.     0.     1.     1.    23.003]
 [20.     0.     0.     0.    11.262]
 [28.     0.     2.     0.    12.879]
 [45.     1.     1.     1.    10.017]
 [39.     0.     2.     1.    17.225]
 [41.     0.     1.     1.    18.739]
 [42.     1.     0.     1.    12.766]
 [73.     0.     0.     0.    18.348]
 [48.     1.     0.     1.    10.446]
 [25.     1.     2.     0.    19.011]
 [39.     1.     2.     0.    15.969]
 [67.     0.     2.     0.    15.891]
 [22.     0.     0.     1.    22.818]
 [59.     0.     2.     0.    13.884]
 [20.     0.     1.     1.    11.686]
 [36.     0.     0.     1.    15.49 ]
 [57.     0.     2.     1.    25.893]
 [70.     1.     0.     0.     9.849]
 [47.     1.     0.     0.    10.403]
 [64.     1.     0.     1.    20.932]
 [58.     1.     0.     0.    18.991]
 [23.     1.     0.     0.     8.011]
 [72.     1.     1.     0.    16.31 ]
 [72.     1.     1.     0.     6.769]
 [56.     0.     1.     0.    11.567]
 [16.     1.     1.     0.    12.006]
 [52.     1.     2.     0.     9.894]
 [23.     1.     2.     1.    14.02 ]
 [40.     0.     1.     1.    11.349]]

print(y)

[0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 1 0 0 1 0 0 1 0 1 1 0 1 1
1 1
 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 0 0
0 1
```

```
 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0
1 0
 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1
1 1
 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 1
1 1
 1 1 0 1 1 1 0]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
```

```
print(x_train)
```

```
[[37.     1.     1.     1.     8.968]
 [49.     1.     1.     0.    10.537]
 [45.     1.     1.     1.     8.37 ]
 [31.     1.     0.     1.    11.871]
 [60.     1.     0.     1.     8.621]
 [61.     0.     1.     1.     7.34 ]
 [39.     1.     0.     0.     9.664]
 [23.     1.     2.     0.    12.26 ]
 [42.     1.     0.     1.    12.766]
 [24.     0.     2.     0.    10.605]
 [21.     0.     0.     1.    28.632]
 [17.     1.     2.     1.    10.832]
 [62.     1.     1.     1.    27.183]
 [51.     0.     2.     0.    13.597]
 [20.     0.     2.     1.     9.281]
 [37.     0.     0.     0.    13.091]
 [32.     0.     0.     1.    25.974]
 [39.     1.     2.     0.    15.969]
 [28.     0.     2.     0.     7.798]
 [24.     1.     2.     0.    25.786]
 [54.     1.     2.     0.    24.658]
 [72.     1.     1.     0.     6.769]
 [18.     0.     2.     1.     8.75 ]
 [20.     0.     1.     1.    11.686]
 [62.     1.     2.     0.    16.594]
 [49.     0.     2.     1.     9.381]
 [37.     0.     1.     1.    12.006]
 [31.     1.     0.     1.    17.069]
 [53.     1.     2.     0.    14.133]
 [58.     0.     1.     0.    26.645]
 [30.     0.     2.     0.    10.443]
 [60.     0.     0.     0.    13.303]
 [57.     1.     1.     1.    19.128]
 [55.     1.     2.     1.     7.261]
 [56.     0.     1.     0.    11.567]
 [47.     1.     0.     0.    10.403]
 [69.     0.     2.     0.    10.065]
 [47.     1.     1.     0.    13.093]
```

```
[36.     0.     0.     1.    15.49 ]
[67.     1.     2.     1.    10.898]
[39.     1.     1.     1.    13.938]
[15.     1.     2.     0.     9.084]
[22.     0.     2.     0.     8.607]
[23.     1.     2.     0.    31.686]
[23.     1.     2.     1.    14.02 ]
[34.     0.     0.     1.    19.199]
[26.     0.     0.     1.    19.161]
[61.     0.     0.     0.    25.475]
[58.     0.     0.     0.    19.416]
[55.     0.     0.     0.    10.977]
[69.     1.     1.     1.    11.455]
[28.     0.     2.     0.    19.675]
[25.     1.     2.     0.    19.011]
[22.     0.     0.     1.    22.818]
[32.     1.     0.     1.     9.445]
[23.     1.     1.     0.     7.298]
[49.     0.     2.     0.    16.275]
[16.     1.     0.     1.    19.007]
[50.     0.     2.     1.    17.211]
[57.     0.     2.     0.    14.216]
[23.     1.     2.     0.    16.85 ]
[48.     1.     0.     1.    10.446]
[68.     0.     2.     1.    27.05 ]
[74.     1.     1.     1.    11.939]
[43.     1.     1.     1.    19.368]
[61.     1.     2.     0.     9.443]
[15.     0.     0.     1.    16.725]
[49.     1.     1.     1.    11.014]
[47.     1.     1.     0.    10.114]
[56.     1.     1.     0.    15.015]
[19.     0.     0.     1.    25.969]
[40.     0.     2.     0.    10.103]
[22.     1.     1.     0.     8.151]
[59.     0.     2.     0.    13.884]
[67.     0.     2.     0.    15.891]
[63.     1.     2.     0.    25.917]
[32.     0.     1.     0.     9.712]
[50.     1.     0.     0.     7.49 ]
[50.     0.     2.     1.    12.295]
[38.     0.     1.     1.    29.875]
[45.     0.     0.     0.    12.854]
[42.     0.     1.     1.    29.271]
[68.     0.     0.     1.    10.189]
[40.     1.     0.     0.    27.826]
[38.     0.     0.     1.    11.326]
[53.     1.     1.     0.    22.963]
[32.     0.     1.     1.    10.84 ]
[22.     1.     2.     0.    11.953]
```

```
[60.     1.     0.     0.    13.934]
[39.     0.     2.     1.    17.225]
[28.     0.     0.     1.    18.809]
[31.     1.     0.     1.    11.227]
[64.     0.     1.     1.    25.741]
[28.     1.     2.     0.    27.064]
[34.     1.     2.     0.    22.456]
[28.     0.     1.     0.    13.127]
[38.     1.     1.     0.    18.295]
[41.     0.     1.     1.    18.739]
[70.     0.     2.     0.    20.489]
[68.     1.     0.     0.    11.009]
[65.     0.     0.     1.    31.876]
[73.     0.     0.     0.    18.348]
[24.     0.     0.     1.    18.457]
[43.     1.     1.     0.    15.376]
[42.     0.     0.     0.    21.036]
[23.     1.     0.     0.     8.011]
[57.     0.     2.     1.    25.893]
[50.     1.     2.     1.    15.79 ]
[57.     0.     0.     1.     9.945]
[41.     0.     2.     1.    22.905]
[26.     0.     0.     1.    12.307]
[67.     1.     2.     1.     9.514]
[41.     1.     1.     0.    11.037]
[49.     1.     0.     1.     8.7  ]
[60.     1.     2.     1.    10.091]
[43.     1.     2.     1.    12.859]
[58.     1.     0.     0.    18.991]
[41.     1.     0.     1.    15.156]
[70.     1.     0.     0.     9.849]
[22.     1.     0.     1.    28.294]
[36.     0.     0.     0.    11.198]
[48.     0.     1.     0.    15.036]
[37.     0.     0.     1.    23.091]
[53.     0.     0.     1.    12.495]
[42.     1.     1.     0.    20.013]
[72.     0.     1.     1.    14.642]
[49.     1.     1.     1.    13.598]
[65.     1.     0.     1.    11.34 ]
[50.     0.     2.     0.    12.703]
[46.     1.     2.     1.     7.285]
[47.     0.     2.     1.     6.683]
[47.     0.     1.     0.    11.767]
[64.     1.     0.     1.    20.932]
[66.     0.     2.     1.     8.107]
[60.     1.     2.     0.    15.171]
[35.     1.     1.     1.     9.17 ]
[24.     1.     0.     1.     9.475]
[74.     1.     0.     0.     9.567]
```

```
 [72.     1.     1.     0.    16.31 ]
 [59.     1.     0.     0.    13.935]
 [28.     0.     2.     0.    12.879]
 [70.     1.     0.     0.    13.967]
 [69.     1.     1.     0.    15.478]
 [43.     1.     0.     0.    13.972]]

print(y_train)

[1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 1 1 0
1 0
 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0 0
1 0
 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 1 0
0 0
 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1]

print(x_test)

[[16.     1.     1.     0.    12.006]
 [65.     0.     1.     1.    13.769]
 [58.     0.     0.     1.    14.239]
 [37.     1.     1.     1.    16.724]
 [31.     1.     0.     0.    30.366]
 [56.     1.     2.     0.     8.966]
 [18.     0.     0.     1.    24.276]
 [35.     1.     2.     1.     7.845]
 [19.     0.     0.     0.    13.313]
 [36.     1.     1.     1.    11.424]
 [28.     0.     1.     0.    19.796]
 [73.     0.     2.     0.    19.221]
 [23.     0.     0.     0.    25.355]
 [51.     1.     0.     0.    18.295]
 [66.     1.     0.     0.    16.347]
 [32.     0.     2.     0.     7.477]
 [52.     1.     2.     0.     9.894]
 [45.     1.     1.     1.    10.017]
 [59.     0.     1.     0.    10.444]
 [51.     0.     1.     1.    23.003]
 [39.     0.     1.     1.    22.697]
 [29.     1.     0.     0.    12.856]
 [34.     1.     0.     0.    18.703]
 [72.     1.     0.     1.     9.677]
 [34.     0.     1.     1.    12.923]
 [56.     0.     0.     0.    25.395]
 [32.     0.     0.     1.    10.292]
 [64.     1.     2.     0.     7.761]
 [49.     1.     0.     1.     6.269]
 [68.     1.     1.     0.    10.291]
 [36.     0.     2.     0.    16.753]
 [16.     0.     0.     1.    15.516]
```

```
 [67.     1.     1.     1.    20.693]
 [26.     0.     1.     0.    14.16 ]
 [26.     1.     1.     1.    20.909]
 [51.     1.     0.     1.    11.343]
 [47.     0.     1.     0.    10.067]
 [74.     0.     1.     0.    20.942]
 [61.     0.     1.     0.    18.043]
 [47.     1.     1.     1.    30.568]
 [74.     1.     0.     1.    15.436]
 [39.     0.     2.     1.     9.709]
 [40.     0.     1.     1.    11.349]
 [15.     1.     0.     1.    17.206]
 [45.     1.     1.     0.    17.951]
 [35.     0.     0.     0.    12.894]
 [29.     0.     0.     0.    29.45 ]
 [20.     0.     0.     0.    11.262]]

print(y_test)

[1 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 0 1
1 0
 0 0 1 1 0 0 1 1 0 0 0]

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)

LinearRegression()

y_pred=model.predict(x_test)
y_pred

array([ 1.00000000e+00, -4.97054830e-16, -6.47220551e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00, -1.95589616e-15,
1.00000000e+00,
       -1.37999663e-18,  1.00000000e+00, -8.66617878e-16, -
8.40435507e-16,
       -1.76193617e-15,  1.00000000e+00,  1.00000000e+00,
1.01245868e-15,
        1.00000000e+00,  1.00000000e+00,  3.75713393e-16, -
1.78455462e-15,
       -1.69622936e-15,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00,
       -2.60834604e-16, -1.88862559e-15,  2.02798046e-17,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00, -3.47070483e-16, -
6.78505646e-16,
        1.00000000e+00, -4.21588063e-17,  1.00000000e+00,
1.00000000e+00,
        4.74332550e-16, -1.20128346e-15, -7.33350169e-16,
```

```
1.00000000e+00,
        1.00000000e+00,  2.94495138e-16, -5.46090758e-17,
1.00000000e+00,
        1.00000000e+00,  7.55136032e-19, -2.37762753e-15,
2.92319791e-16])

x_pred = model.predict(x_train)
x_pred

array([ 1.00000000e+00,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00,
        1.00000000e+00,  4.49703586e-16,  1.00000000e+00,
1.00000000e+00,
        1.00000000e+00,  5.88253978e-16, -2.59843846e-15,
1.00000000e+00,
        1.00000000e+00,  5.55504447e-17,  4.26151845e-16, -
3.51334715e-17,
       -2.25336646e-15,  1.00000000e+00,  9.80572011e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00,  5.10465261e-16, -
3.02021442e-17,
        1.00000000e+00,  3.05416693e-16, -1.38873860e-16,
1.00000000e+00,
        1.00000000e+00, -1.96951647e-15,  5.89761452e-16, -
1.50126874e-16,
        1.00000000e+00,  1.00000000e+00,  2.23885851e-16,
1.00000000e+00,
        5.01695524e-16,  1.00000000e+00, -7.48002744e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00,  8.85259593e-16,
1.00000000e+00,
        1.00000000e+00, -1.27842343e-15, -1.24360734e-15, -
1.91854102e-15,
       -1.02909019e-15,  2.05423647e-16,  1.00000000e+00, -
7.41408390e-16,
        1.00000000e+00, -1.75916216e-15,  1.00000000e+00,
1.00000000e+00,
       -3.25391263e-16,  1.00000000e+00, -8.33474944e-16, -
5.61749426e-17,
        1.00000000e+00,  1.00000000e+00, -2.32591704e-15,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00, -8.50128520e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00, -2.20501818e-15,
6.02422821e-16,
        1.00000000e+00, -1.53667721e-17, -3.35657256e-16,
1.00000000e+00,
        5.80752052e-16,  1.00000000e+00, -1.20731351e-16, -
2.73326446e-15,
       -3.00788325e-17, -2.66034719e-15, -9.66668451e-17,
1.00000000e+00,
```

```
       -1.51614133e-16,  1.00000000e+00,  4.84946905e-17,
1.00000000e+00,
        1.00000000e+00, -7.95208044e-16, -1.19989948e-15,
1.00000000e+00,
       -2.22914542e-15,  1.00000000e+00,  1.00000000e+00,
1.00283469e-16,
        1.00000000e+00, -1.12970755e-15, -1.01328580e-15,
1.00000000e+00,
       -3.22995477e-15, -9.29196813e-16, -1.13421161e-15,
1.00000000e+00,
       -1.20535166e-15,  1.00000000e+00, -2.11787335e-15,
1.00000000e+00,
       -2.09939559e-17, -1.62604643e-15, -2.49883869e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00,
        2.42985441e-16, -2.49758554e-16, -1.85369294e-15, -
3.76050984e-16,
        1.00000000e+00, -6.49269604e-16,  1.00000000e+00,
1.00000000e+00,
        1.88829884e-16,  1.00000000e+00,  7.03911428e-16,
2.27858972e-16,
        1.00000000e+00,  4.27850202e-16,  1.00000000e+00,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00,
        2.43905983e-16,  1.00000000e+00,  1.00000000e+00,
1.00000000e+00])

y_test[:6]

array([1, 0, 0, 1, 1, 1])

model.predict(x_test[:6])

array([ 1.00000000e+00, -4.97054830e-16, -6.47220551e-16,
1.00000000e+00,
        1.00000000e+00,  1.00000000e+00])

model.score(x_test,y_test)

1.0

X_train, X_test, y_train,
y_test=train_test_split(df.drop('Drug',axis=1),df['Drug'],test_size=0.
2,stratify=df['Drug'])

#logistic regression
model = LogisticRegression()
model.fit(X_train,y_train)
```

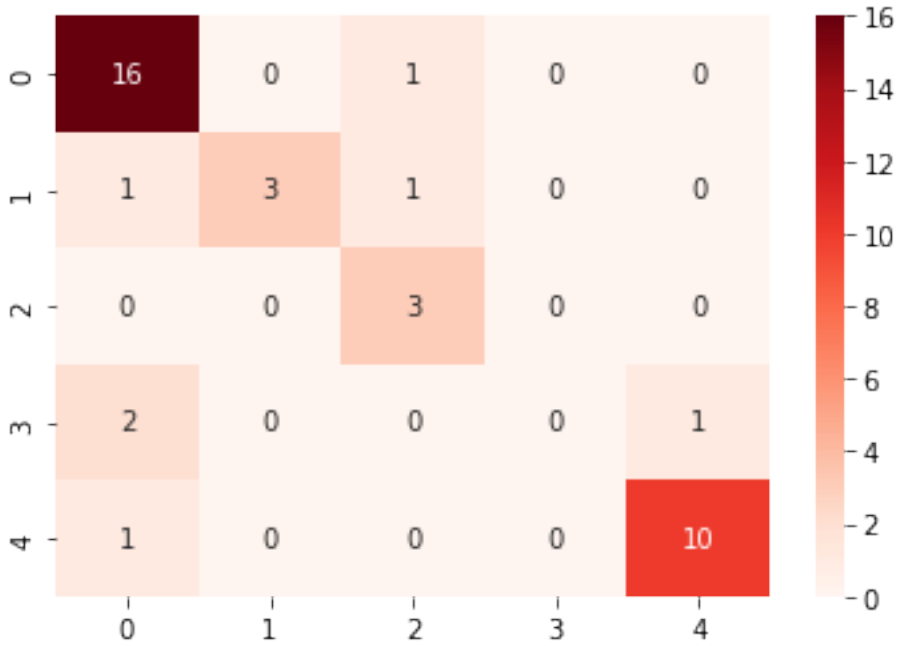
```
LogisticRegression()
```

```
print(f"Accuracy score for logistic regression model is
{model.score(X_test,y_test)*100}")
```

```
Accuracy score for logistic regression model is 82.05128205128204
```

```
# Plotting the confusion matrix for this model
predict = model.predict(X_test)
cf = confusion_matrix(y_test,predict)
sns.heatmap(cf, cmap='Reds',annot=True)
```

```
<AxesSubplot:>
```




```
# Classification report for Logistic Regression
from sklearn import metrics
print(metrics.classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

           0       0.80      0.94      0.86        17
           1       1.00      0.60      0.75         5
           2       0.60      1.00      0.75         3
           3       0.00      0.00      0.00         3
           4       0.91      0.91      0.91        11

    accuracy                           0.82        39
   macro avg       0.66      0.69      0.65        39
weighted avg       0.78      0.82      0.79        39
```

```
model = RandomForestClassifier()
model.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
print(classification_report(y_test,ypred_dt))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         3
           4       1.00      1.00      1.00        11

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39
```

```
print(f"Accuracy score for Random Forest model is
{model.score(X_test,y_test)*100}")
```

```
Accuracy score for Random Forest model is 97.43589743589743
```

```
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

```
print(f"Accuracy score for Decision Tree Classifier is
{model.score(X_test,y_test)*100}")
```

```
Accuracy score for Decision Tree Classifier is 100.0
```

```
scores = {}
```

```
from sklearn.model_selection import GridSearchCV
grid = {
    'C':[0.01,0.1,1,10],
    'kernel' : ["linear","poly","rbf","sigmoid"],
    'degree' : [1,3,5,7],
    'gamma' : [0.01,1]
}
```

```
svm  = SVC ();
svm_cv = GridSearchCV(svm, grid, cv = 5)
svm_cv.fit(X_train,y_train)
print("Best Parameters:",svm_cv.best_params_)
print("Train Score:",svm_cv.best_score_)
print("Test Score:",svm_cv.score(X_test,y_test))
```

```
Best Parameters: {'C': 10, 'degree': 1, 'gamma': 0.01, 'kernel':
'linear'}
Train Score: 1.0
Test Score: 1.0
```

```
from sklearn.linear_model import Lasso, LassoCV

lasso_cv = LassoCV(alphas = None, cv = 10, max_iter = 100000,
normalize = True)
lasso_cv.fit(X_train, y_train)
```

```
LassoCV(cv=10, max_iter=100000, normalize=True)
```

```
# best alpha parameter

alpha = lasso_cv.alpha_
alpha
```

```
0.003243748941942202
```

```
lasso = Lasso(alpha = lasso_cv.alpha_)
lasso.fit(X_train, y_train)
```

```
Lasso(alpha=0.003243748941942202)
```

```
lasso.score(X_train, y_train)
```

```
0.6299201075742547
```

```
lasso.score(X_test, y_test)
```

```
0.6409608228247142
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import *
from sklearn.metrics import r2_score

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.8, random_state=1)

rid = Ridge()

rid.fit(x_train,y_train)
```

```
Ridge()
```

```
x_test_pred_rd = rid.predict(x_test)
r2_score(y_test, x_test_pred_rd)
```

```
0.9839109320556412
```