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

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\4_drug200 - 4_drug200.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.info()
```

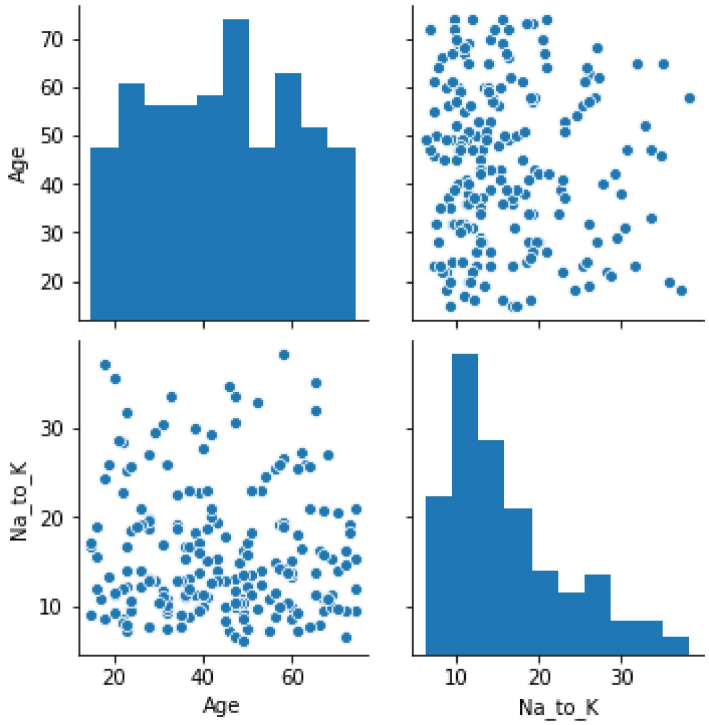
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    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 [4]:
```
df.columns
```

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

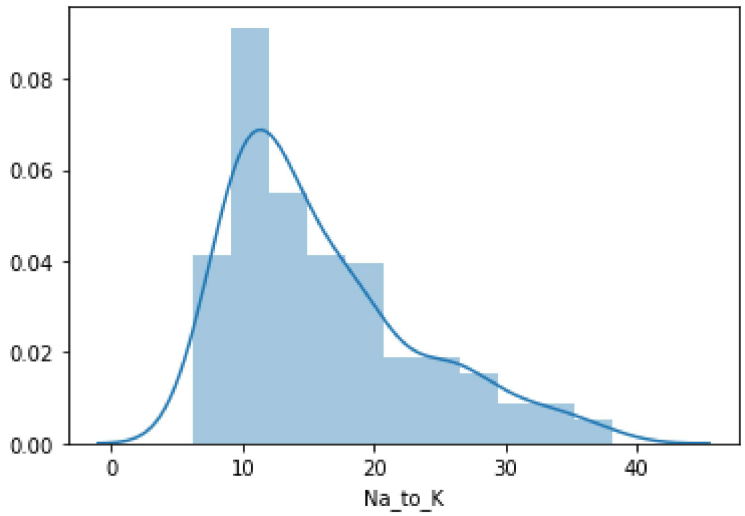
In [5]:
```
sns.pairplot(df)
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x18f1e122130>
```



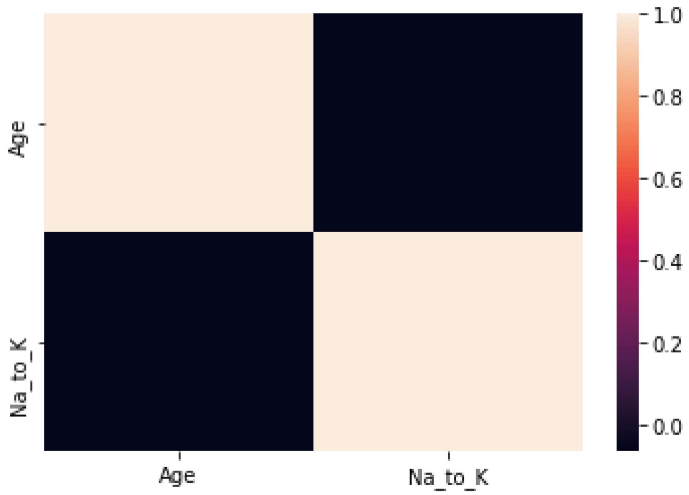
In [6]:
```
sns.distplot(df['Na_to_K'])
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x18f1e3573a0>
```



In [7]:
```
sns.heatmap(df.corr())
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x18f1e42d850>



In [8]:
```
x=df[['Age']]
y=df['Na_to_K']
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [10]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

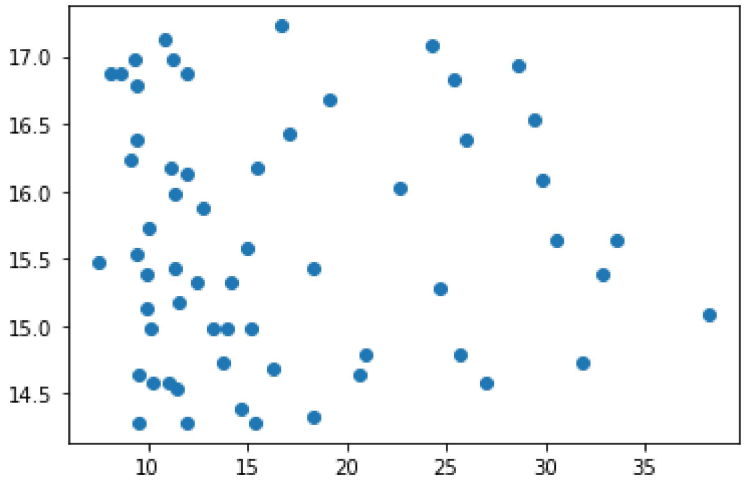
Out[10]: LinearRegression()

In [11]:
```
print(lr.intercept_)
```

```
17.976713998736777
```

In [12]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[12]: <matplotlib.collections.PathCollection at 0x18f1e87ad90>



In [13]:
```
print(lr.score(x_test,y_test))
```

```
-0.033514240720123656
```

In [14]:
```
print(lr.score(x_train,y_train))
```

```
0.01294370415328594
```

In [15]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [16]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[16]: Ridge(alpha=10)

In [17]:
```
rr.score(x_test,y_test)
```

Out[17]: -0.033504093143870595

In [18]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[18]: Lasso(alpha=10)

In [19]:
```
la.score(x_test,y_test)
```

Out[19]: -0.013873753091794061

In [20]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[20]:
```
ElasticNet()
```

In [21]:
```
print(en.intercept_)
```

```
17.885946121323627
```

In [22]:
```
print(en.predict(x_test))
```

```
[14.39801876 14.68469772 16.0703127  15.44917495 14.63691789 16.16587235
 15.59251443 15.73585391 14.63691789 15.3536153  16.93034958 14.63691789
 16.7870101  14.8280372  14.78025737 15.87919339 15.16249599 14.35023893
 15.11471616 17.16924872 16.83478993 16.21365218 15.40139512 15.97475305
 14.44579859 15.49695478 15.01915651 15.40139512 15.01915651 16.35699166
 16.02253287 15.64029426 17.02590924 16.88256976 16.50033114 15.44917495
 16.73923028 14.58913807 15.30583547 14.35023893 16.16587235 14.73247755
 16.83478993 15.01915651 16.93034958 17.07368906 15.21027582 14.35023893
 14.68469772 15.5447346  14.78025737 16.40477149 16.11809253 15.01915651
 16.64367062 15.3536153  14.8280372  16.83478993 16.35699166 15.64029426]
```

In [23]:
```
print(en.score(x_test,y_test))
```

```
-0.032046076154058856
```

Evaluation

In [24]:
```
from sklearn import metrics
```

In [25]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 6.2497185971993305
```

In [26]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 63.18006644989919
```

In [27]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean Squared Error: 7.948588959677006
```

In [ ]:
```

```