# D3

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\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.describe()
```

Out[4]:

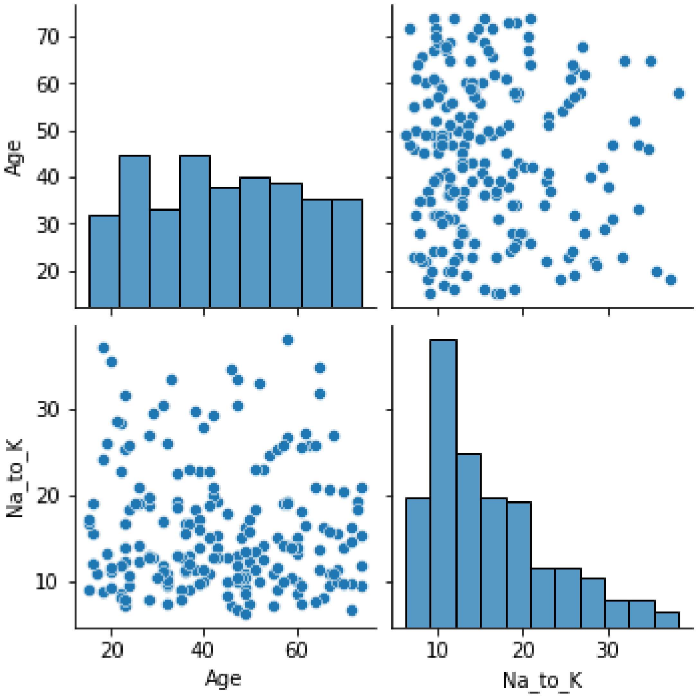
| | Age | Na_to_K |
|---|---|---|
| **count** | 200.000000 | 200.000000 |
| **mean** | 44.315000 | 16.084485 |
| **std** | 16.544315 | 7.223956 |
| **min** | 15.000000 | 6.269000 |
| **25%** | 31.000000 | 10.445500 |
| **50%** | 45.000000 | 13.936500 |
| **75%** | 58.000000 | 19.380000 |
| **max** |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

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [6]:
```
sns.pairplot(df)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1598e26d430>
```



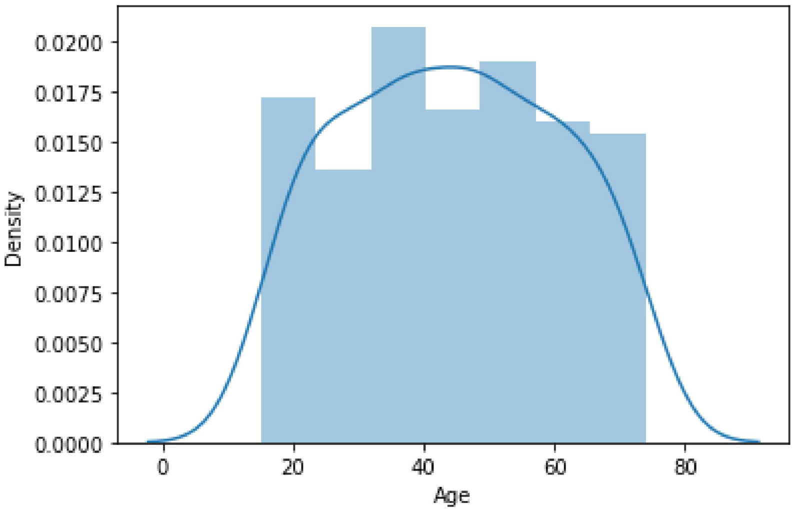
In [7]:

```
sns.distplot(df["Age"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[7]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



In [8]:

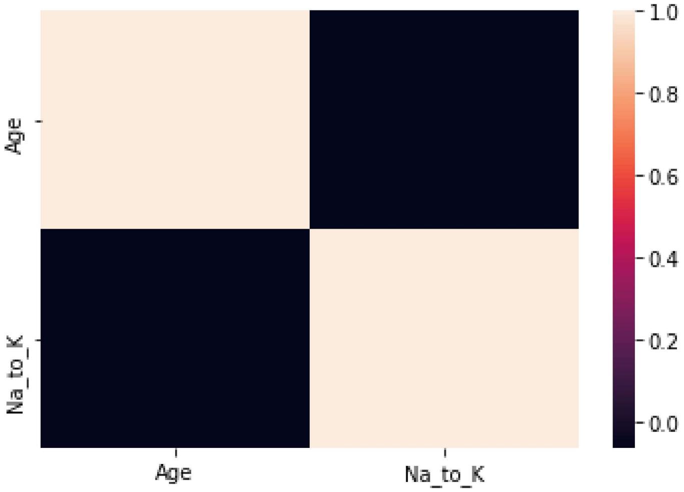
```
df1=df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
```

In [9]:

```
sns.heatmap(df1.corr())
```

Out[9]:

```
<AxesSubplot:>
```



In [10]:

```
x=df1[['Age']]
y=df1['Na_to_K']
```

In [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
17.10771115845836
```

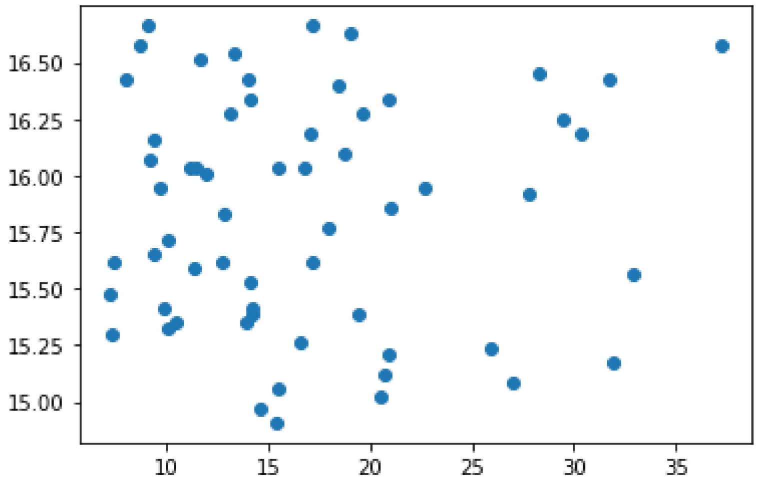
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

| | Co-efficient |
|---|---|
| **Age** | -0.029715 |

In [15]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x15990511e20>
```



In [16]:
```
print(lr.score(x_test,y_test))
```

```
-0.01722911502858282
```

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

Out[18]:
```
Ridge(alpha=10)
```

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

Out[19]:
```
-0.017228719109628976
```

In [20]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[20]:
```
Lasso(alpha=10)
```

In [22]:
```
la.score(x_test,y_test)
```

Out[22]:
```
-0.020687052718400212
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[-0.02776591]
```

In [25]:
```
print(en.intercept_)
```

```
17.020012335318512
```

In [26]:
```
print(en.predict(x_train))
```

```
[15.15969609 16.32586448 15.82607803 14.96533469 16.18703491 16.13150308
 16.60352362 15.13193018 16.04820534 16.3536304  15.68724846 15.65948254
 16.40916223 15.35405749 15.29852566 16.38139631 16.5479918  16.24256674
 15.93714169 16.38139631 15.85384394 15.40958932 15.68724846 15.07639835
 15.88160986 15.10416426 15.99267351 16.24256674 15.99267351 15.93714169
 15.88160986 16.159269   15.9649076  16.159269   15.3818234  15.71501437
 16.46469405 16.40916223 15.63171663 16.57575771 15.52065297 16.38139631
 15.93714169 16.13150308 15.7705462  15.46512115 16.07597126 16.46469405
 15.15969609 15.43735523 16.04820534 15.71501437 16.29809857 15.02086652
 15.88160986 16.43692814 15.21522792 15.60395072 15.18746201 15.74278029
 15.65948254 15.35405749 14.96533469 15.02086652 15.49288706 16.13150308
 15.82607803 15.15969609 15.60395072 16.57575771 15.82607803 14.96533469
 15.54841889 15.90937577 15.63171663 16.13150308 16.40916223 15.32629158
 14.99310061 15.32629158 16.38139631 16.10373717 15.90937577 15.21522792
 16.07597126 16.21480083 15.21522792 15.13193018 15.7705462  15.88160986
 15.60395072 15.54841889 15.10416426 15.74278029 15.32629158 16.40916223
 15.9649076  15.13193018 15.24299383 15.24299383 16.29809857 15.43735523
 15.46512115 15.65948254 15.85384394 15.35405749 16.07597126 15.9649076
 15.71501437 15.71501437 15.46512115 15.99267351 15.7705462  16.3536304
 15.71501437 15.35405749 16.24256674 15.65948254 16.52022588 15.40958932
 15.40958932 16.13150308 15.93714169 15.85384394 15.65948254 15.5761848
 15.02086652 15.07639835 16.24256674 16.49245997 15.46512115 14.99310061
 16.46469405 15.99267351 15.71501437 15.18746201 16.24256674 15.65948254
 15.71501437 16.3536304 ]
```

In [27]:

```
print(en.score(x_train,y_train))
```

```
0.004524680678691162
```

In [28]:

```
from sklearn import metrics
```

In [29]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 5.675777671689483
```

In [30]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 53.711773888821966
```

In [31]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 2.3823890680763045
```

In [ ]: