kaviyadevi 20106064

In [8]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\4_drug200 - 4_drug200.csv")
data
```

Out[9]:

| | 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 [10]:
```
data.head()
```

Out[10]:

| | 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 |

# DATA CLEANING AND PREPROCESSING

In [11]:
```
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 [12]:
```
data.describe()
```

Out[12]:

| | 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 |

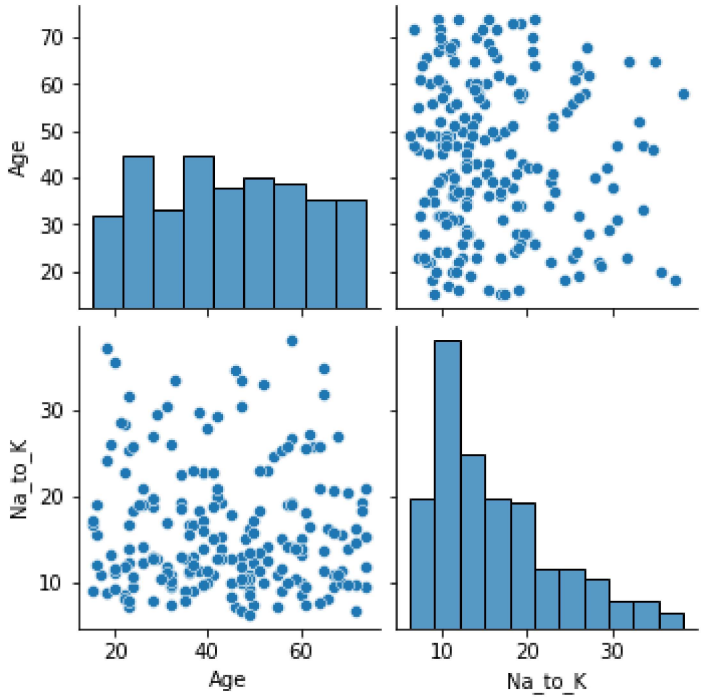
In [13]:
```
data.columns
```

Out[13]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

# EDA and DATA VISUALIZATION

In [14]:
```
sns.pairplot(data)
```

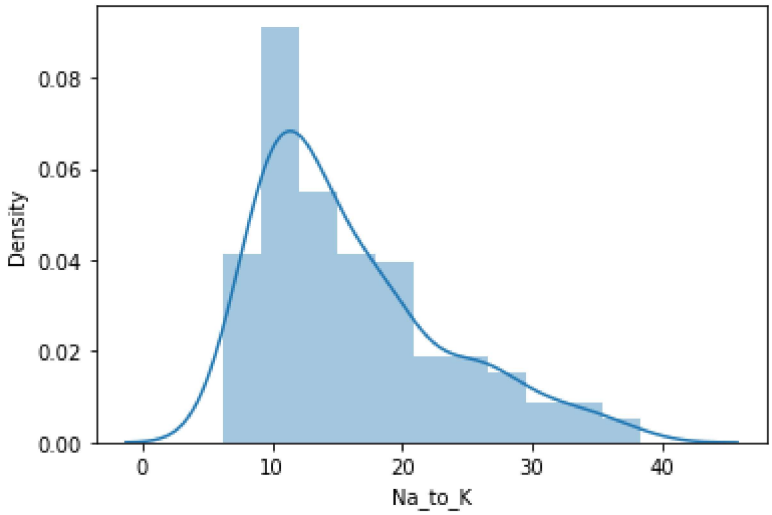
Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x28d057fa3d0>
```



In [15]:
```
sns.distplot(data['Na_to_K'])
```

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

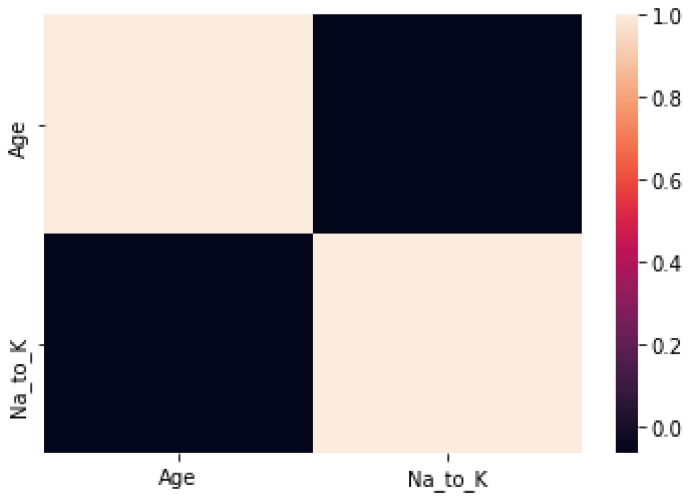
Out[15]: <AxesSubplot:xlabel='Na_to_K', ylabel='Density'>




In [16]:
```
df=data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
```

In [17]:
```
sns.heatmap(df.corr())
```

Out[17]: <AxesSubplot:>



# TRAINNING MODEL

In [18]:
```
x=df[['Age']]
y=df[['Na_to_K']]
```

In [19]:
```
#to split my dataset into trainning and test

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 [20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

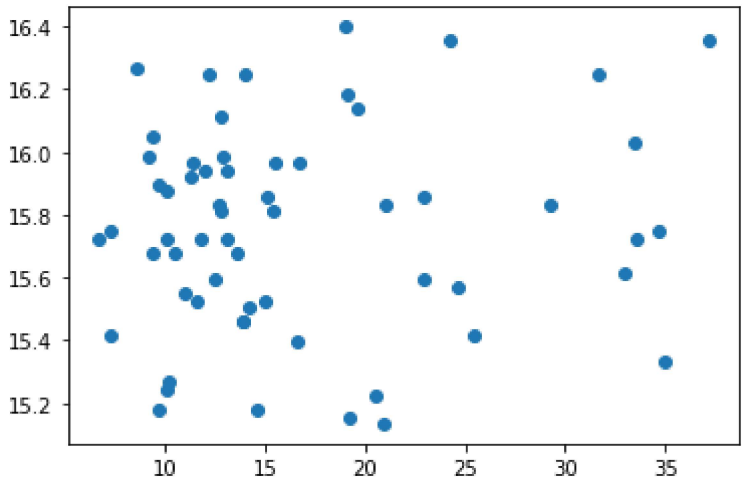
Out[20]:
```
LinearRegression()
```

In [21]:
```
#to find intercept
print(lr.intercept_)
```

```
[16.74885176]
```

In [22]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x28d05c025b0>
```




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

```
-0.009955722953167712
```

# RIDGE AND LASSO REGRESSION

In [24]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]:
```
Ridge(alpha=10)
```

In [26]:
```
rr.score(x_test,y_test)
```

Out[26]:
```
-0.009956517886995808
```

In [27]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[27]:
```
Lasso(alpha=10)
```

In [28]:
```
la.score(x_test,y_test)
```

Out[28]:
```
-0.01491531607825447
```

In [29]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:
```
ElasticNet()
```

In [30]:
```
print(en.coef_)
```

```
[-0.02006718]
```

In [31]:
```
print(en.predict(x_test))
```

```
[15.80946899 16.21081255 15.72920028 15.72920028 16.09040948 15.82953617
 15.68906592 15.56866285 16.31114844 16.15061101 16.23087973 15.60879721
 15.84960334 15.86967052 15.72920028 15.54859567 16.03020795 15.48839414
 15.60879721 15.44825978 15.92987206 15.30778954 15.72920028 16.01014077
 15.28772236 15.36799107 15.5285285  15.90980488 15.22752082 15.58873003
 15.74926745 15.62886439 15.44825978 15.20745365 15.94993923 15.82953617
 15.82953617 16.21081255 15.68906592 16.11047666 15.92987206 15.8897377
 16.21081255 15.54859567 16.31114844 15.68906592 15.94993923 15.18738647
 15.84960334 15.26765518 15.80946899 15.42819261 15.94993923 15.22752082
 15.74926745 15.72920028 15.48839414 15.97000641 15.97000641 16.35128279]
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
-0.010226915185498786
```

# Evaluation metrics

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 6.1202398832739195
```

In [35]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 63.86700838972528
```

In [36]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 7.991683701806853
```