mk 31/07/23

In [141]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [142]:
```
x=pd.read_csv(r"C:\Users\user\Downloads\4_drug200.csv")
x
```

Out[142]:

	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 [143]:
```
x.head(10)
```

Out[143]:

	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
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

In [144]:
```
x.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 [145]:
```
x.columns
```

Out[145]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

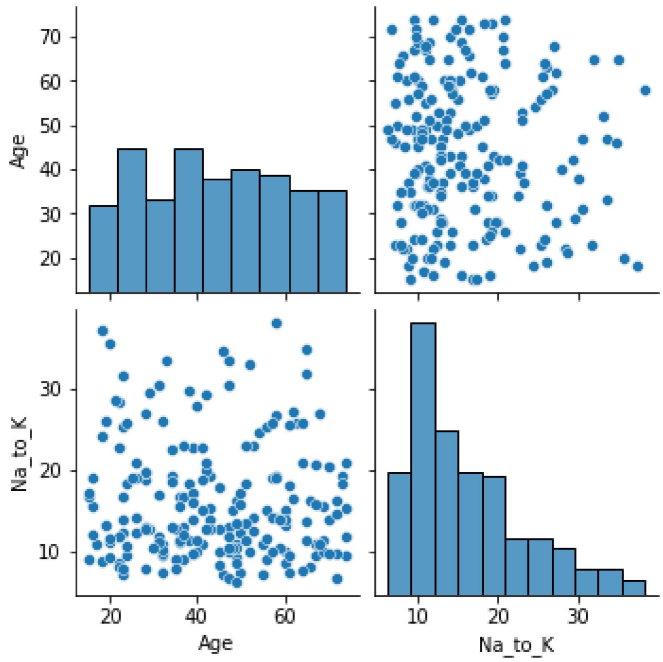
In [146]:
```
x.describe()
```

Out[146]:

	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 [147]:
```
sns.pairplot(x)
```

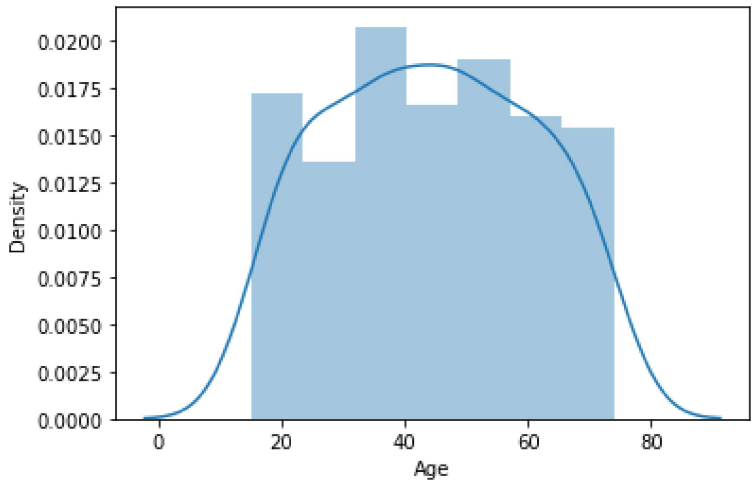
Out[147]:
```
<seaborn.axisgrid.PairGrid at 0x110f25b39a0>
```



In [150]:
```
sns.distplot(x['Age'])
```

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

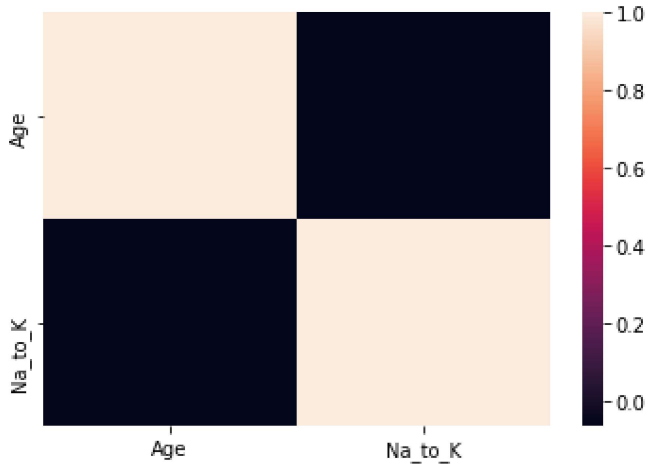
Out[150]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



In [152]:
```
x1=x[['Age','Na_to_K']]
```

In [153]:
```
sns.heatmap(x1.corr())
```

Out[153]: <AxesSubplot:>



In [154]:
```
a=x1[['Age']]
b=x1['Na_to_K']
```

In [155]:
```
from sklearn.model_selection import train_test_split
a_train,a_test, b_train, b_test=train_test_split(a,b,test_size=0.3)
```

In [156]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(a_train,b_train)
```

Out[156]: LinearRegression()

In [157]:
```
print(lr.intercept_)
```

```
17.554827577508803
```

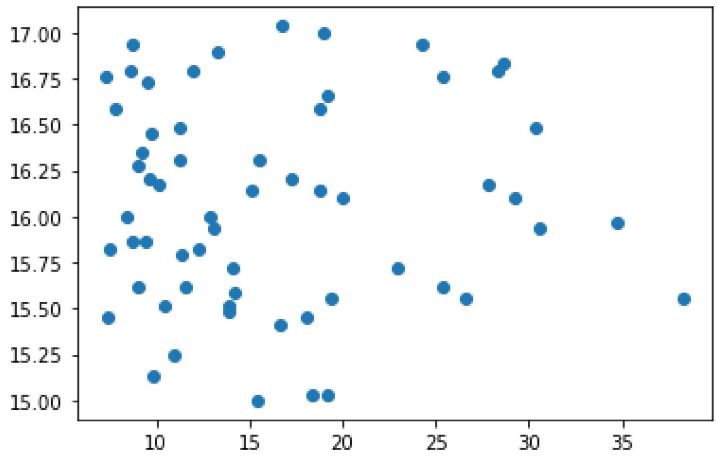
In [158]:
```
coeff=pd.DataFrame(lr.coef_,a.columns,columns=['Co-efficient'])
coeff
```

Out[158]:

	Co-efficient
Age	-0.034542

In [159]:
```
prediction=lr.predict(a_test)
plt.scatter(b_test,prediction)
```

Out[159]:
```
<matplotlib.collections.PathCollection at 0x110f5a81fa0>
```



In [160]:
```
lr.score(a_test,b_test)
```

Out[160]:
```
-0.003219817632408084
```

In [161]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [162]:
```
rr=Ridge(alpha=10)
rr.fit(a_train,b_train)
```

Out[162]:
```
Ridge(alpha=10)
```

In [163]:
```
rr.score(a_test,b_test)
```

Out[163]:
```
-0.003217911883337088
```

In [164]:
```
la=Lasso(alpha=10)
la.fit(a_train,b_train)
```

Out[164]:
```
Lasso(alpha=10)
```

In [165]:
```
la.score(a_test,b_test)
```

Out[165]:
```
-0.001144687022455404
```

In [166]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(a_train,b_train)
```

Out[166]:
```
ElasticNet()
```

In [167]:
```
en.coef_
```

Out[167]:
```
array([-0.03268358])
```

In [168]:
```
en.predict(a_test)
```

Out[168]:
```
array([16.6871733 , 16.16423606, 16.78522403, 15.64129881, 15.54324808,
       16.85059119, 15.57593166, 16.88327477, 15.28177946, 16.55643899,
       15.44519735, 15.60861523, 16.55643899, 15.87008386, 16.75254045,
       16.45838825, 15.73934955, 16.71985688, 15.57593166, 15.93545101,
       16.19691963, 16.0988689 , 16.88327477, 16.13155248, 15.57593166,
       16.26228679, 16.94864192, 15.8047167 , 15.93545101, 16.75254045,
       15.08567799, 15.47788092, 15.18372872, 16.0988689 , 16.32765394,
       16.75254045, 15.47788092, 15.73934955, 16.29497037, 16.42570468,
       16.00081817, 16.19691963, 15.64129881, 15.87008386, 15.08567799,
       16.00081817, 15.64129881, 15.5105645 , 15.05299441, 16.9813255 ,
       15.96813459, 15.83740028, 16.62180614, 15.83740028, 16.71985688,
       16.29497037, 16.16423606, 15.54324808, 16.13155248, 16.45838825])
```

In [169]:
```
en.intercept_
```

Out[169]:
```
17.47157916366664
```

In [170]:
```
en.score(a_test,b_test)
```

Out[170]:
```
-0.002836267719833252
```

In [171]:
```
from sklearn import metrics
```

In [172]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(b_test,prediction))
```

```
Mean Absolute Error 6.196695881765147
```

In [173]:
```
print("Mean Squared Error",metrics.mean_squared_error(b_test,prediction))
```

```
Mean Squared Error 58.28272579059952
```

In [174]:
```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(b_test,prediction)))
```

```
Root Mean Squared Error 7.634312398022465
```

In [ ]:
```

```