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

In [188]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\4_drug200 - 4_drug200.csv")
a
```

Out[188]:

	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 [189]:
```
a=a.head(10)
a
```

Out[189]:

	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 [190]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          10 non-null     int64  
 1   Sex          10 non-null     object 
 2   BP           10 non-null     object 
 3   Cholesterol  10 non-null     object 
 4   Na_to_K      10 non-null     float64
 5   Drug         10 non-null     object 
dtypes: float64(1), int64(1), object(4)
memory usage: 608.0+ bytes
```

In [191]:
```
a.columns
```

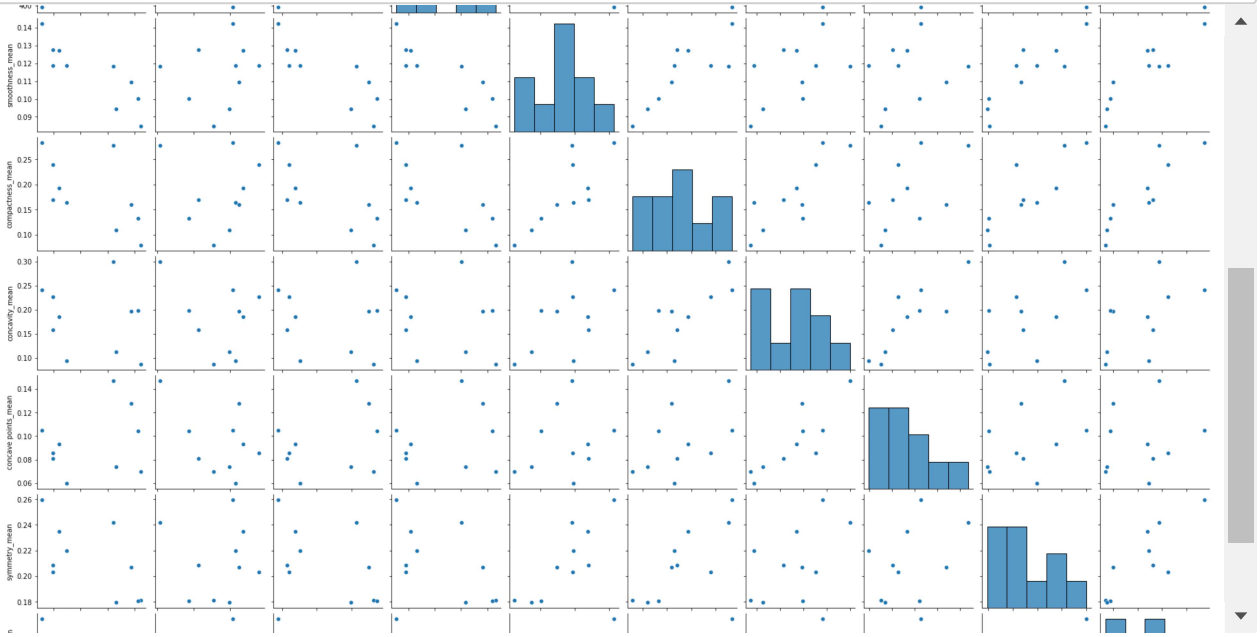
Out[191]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [192]:
```
a.describe()
```

Out[192]:

	Age	Na_to_K
count	10.000000	10.000000
mean	42.100000	14.486100
std	13.916018	5.482634
min	22.000000	7.798000
25%	31.250000	10.344750
50%	45.000000	14.132000
75%	48.500000	17.601000
max	61.000000	25.355000

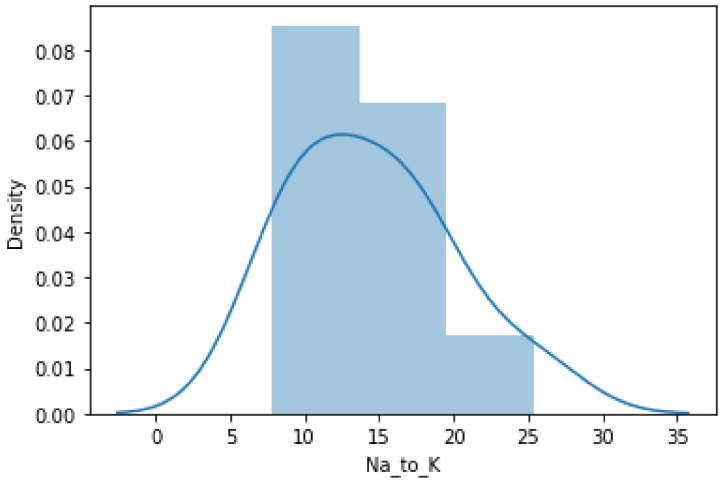
In [193]:
```
sns.pairplot(d)
```



In [194]:
```
sns.distplot(a['Na_to_K'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[194]: <AxesSubplot:xlabel='Na_to_K', ylabel='Density'>



In [202]:
```
x1=a[['Age']]
```

In [203]:
```
sns.heatmap(x1.corr())
```

Out[203]: <AxesSubplot:>

In [205]:
```
x=a[['Age']]
y=a['Na_to_K']
```

In [206]:
```
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 [207]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[207]:
```
LinearRegression()
```

In [208]:
```
print(lr.intercept_)
```

```
12.73248780487805
```

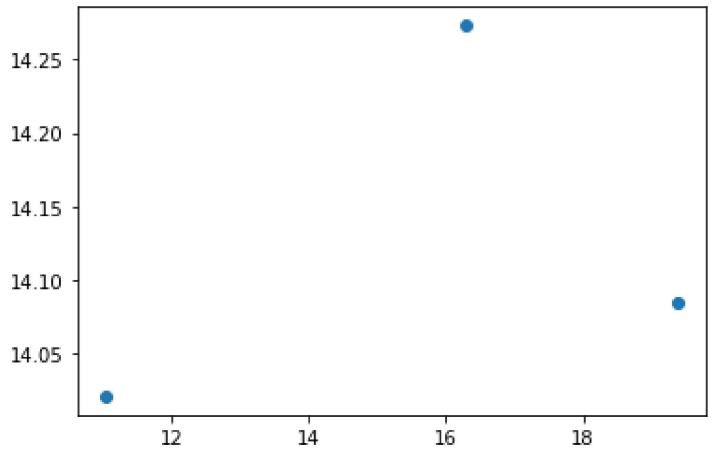
In [209]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[209]:

	Co-efficient
Age	0.031436

In [210]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[210]:
```
<matplotlib.collections.PathCollection at 0x190b82d3280>
```



In [211]:
```
print(lr.score(x_test,y_test))
```

```
-0.15121522622507233
```

In [212]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [213]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[213]:
```
Ridge(alpha=10)
```

In [214]:
```
rr.score(x_test,y_test)
```

Out[214]:
```
-0.15148657774410612
```

In [215]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[215]:
```
Lasso(alpha=10)
```

In [216]:
```
la.score(x_test,y_test)
```

Out[216]:
```
-0.1990656600352263
```

In [217]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[217]:
```
ElasticNet()
```

In [218]:
```
print(en.coef_)
```

```
[0.0293004]
```

In [219]:
```
print(en.intercept_)
```

```
12.82035491231777
```

In [220]:
```
print(en.predict(x_test))
```

```
[14.02167137 14.25607458 14.08027217]
```

In [221]:
```
en.score(x_test,y_test)
```

Out[221]:
```
-0.15435098837496364
```

In [222]:
```
from sklearn import metrics
```

In [223]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 3.4234243733062315
```

In [224]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 13.611054769995626
```

In [225]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 3.6893163011587427
```