LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\4_drug200.csv")
df
```

Out[3]:

	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

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	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

data cleaning

In [5]:

```
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 [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
df.columns
```

Out[7]:

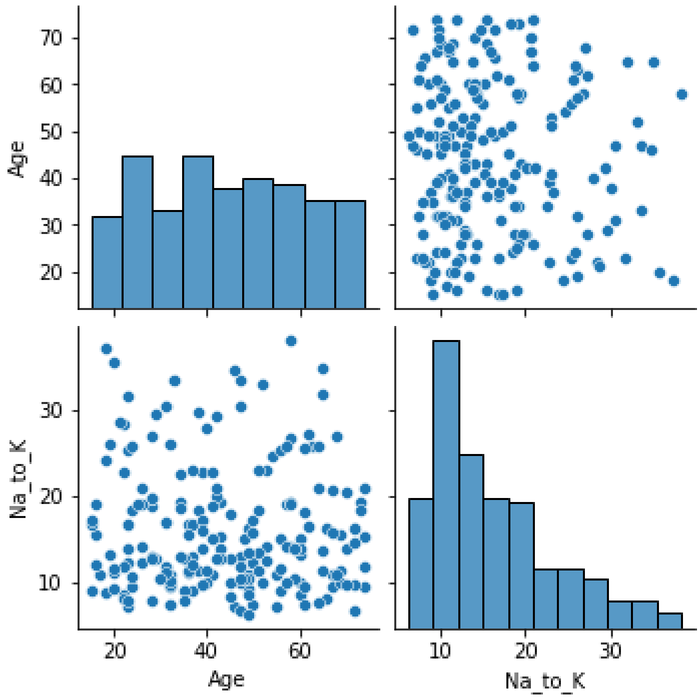
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='objec
t')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x1f5543c0520>
```



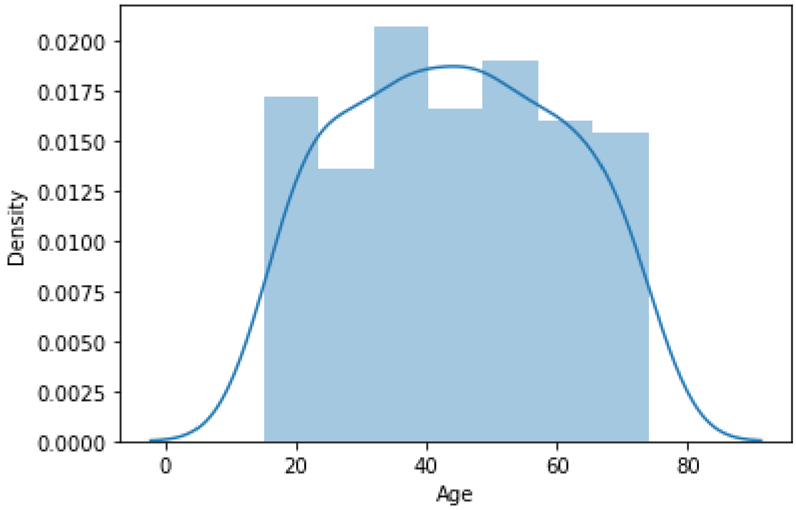
In [9]:

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

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

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



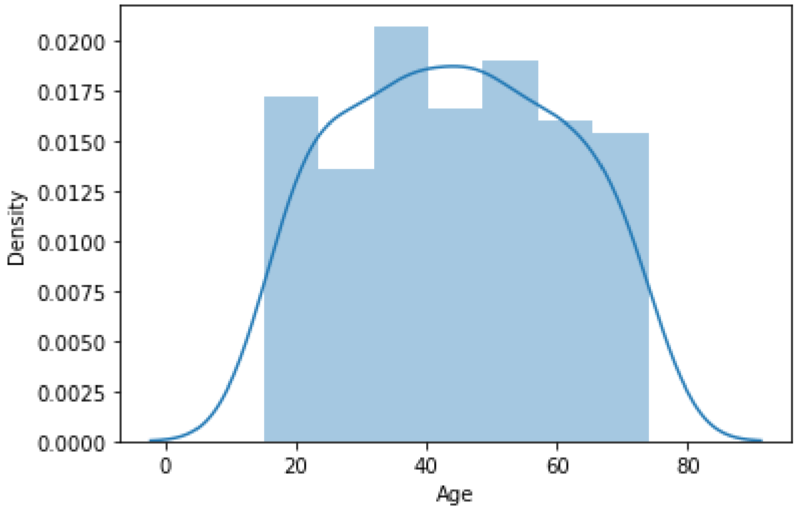
In [10]:

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

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

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



In [12]:

```
df1=df[['Age', 'Na_to_K']]
df1
```

Out[12]:

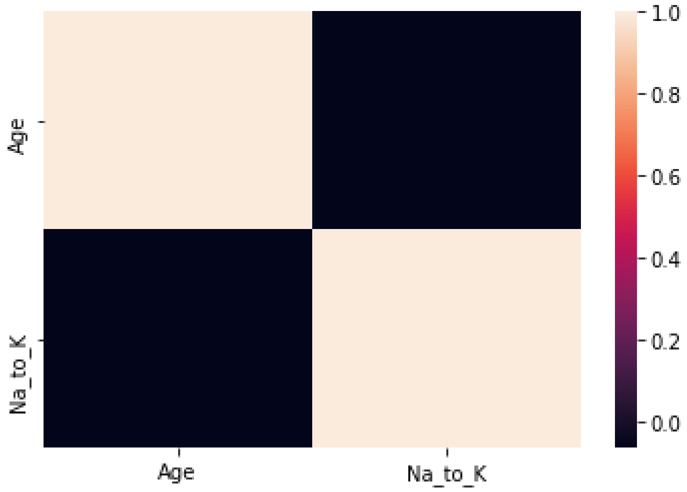
	Age	Na_to_K
0	23	25.355
1	47	13.093
2	47	10.114
3	28	7.798
4	61	18.043
...	...	...
195	56	11.567
196	16	12.006
197	52	9.894
198	23	14.020

In [13]:

```
sb.heatmap(df1.corr())
```

Out[13]:

```
<AxesSubplot:>
```



model building

In [14]:

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

In [15]:

```
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)
```

linear regression

In [16]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[16]:

```
LinearRegression()
```

In [17]:

```
print(lr.intercept_)
```

```
2.1316282072803006e-14
```

In [18]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[18]:

	Co_efficient
Age	1.0
Na_to_K	0.0

In [19]:

```
print(lr.score(x_test,y_test))
```

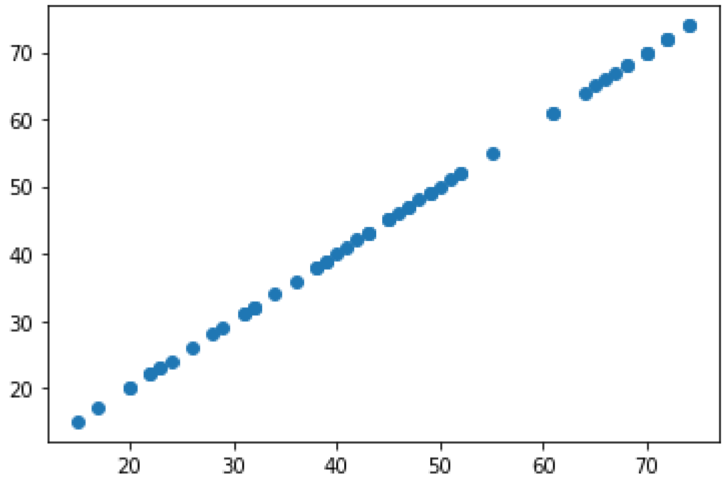
```
1.0
```

In [20]:

```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x1f55a4bad30>
```



lasso and ridge regression

In [21]:

```
lr.score(x_test,y_test)
```

Out[21]:

```
1.0
```

In [22]:

```
lr.score(x_train,y_train)
```

Out[22]:

```
1.0
```

In [23]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[24]:

```
0.999999927211552
```

In [25]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[25]:

```
0.9985991998289931
```

elasticnet

In [26]:

```
from sklearn.linear_model import ElasticNet
e = ElasticNet()
e.fit(x_train,y_train)
```

Out[26]:

```
ElasticNet()
```

In [27]:

```
print(e.coef_)
```

```
[ 0.99626426 -0.        ]
```

In [28]:

```
print(e.intercept_)
```

```
0.16373195410992025
```

In [29]:

```
predictions = e.predict(x_test)
predictions
```

Out[29]:

```
array([60.93585208, 69.90223046, 73.88728752, 41.01056679, 50.97320944,
       38.021774  , 23.07781004, 64.92090914, 22.08154577, 39.01803827,
       71.89475899, 34.03671694, 65.91717341, 43.00309532, 44.99562385,
       63.92464488, 67.90970193, 54.9582665 , 29.05539562, 69.90223046,
       51.9694737 , 32.04418842, 15.10769592, 73.88728752, 32.04418842,
       71.89475899, 47.98441665, 69.90223046, 39.01803827, 22.08154577,
       48.98068091, 38.021774  , 26.06660283, 49.97694517, 20.08901724,
       45.99188812, 60.93585208, 20.08901724, 48.98068091, 31.04792415,
       71.89475899, 42.00683106, 43.00309532, 23.07781004, 46.98815238,
       46.98815238, 67.90970193, 66.91343767, 40.01430253, 17.10022445,
       28.05913136, 32.04418842, 36.02924547, 31.04792415, 44.99562385,
       44.99562385, 51.9694737 , 24.0740743 , 48.98068091, 43.00309532])
```

In [30]:

```
print(e.score(x_test,y_test))
```

```
0.9999859144252782
```

In [31]:

```
from sklearn import metrics
```

mean absolute error

In [32]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.05188580949070006
```

mean squared error

In [33]:

```
print("Mean Squared Error:", metrics.mean_squared_error(y_test,predictions))
```

```
Mean Squared Error: 0.003979843923723614
```

root mean squared error

In [34]:

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
Root Mean Squared Error 0.06308600418257297
```

In [ ]:

```

```