Problem statement

Data collection

Importing libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\4_drug200 - 4_drug200.csv")
data
```

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

head

In [3]:
```
# to display first 8 dataset values
da=data.head(8)
da
```

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
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC

info

In [4]:
```
# to identify missing values
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
```

describe

In [5]:
```
# to display summary of the dataset
data.describe()
```

Out[5]:

	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

	Age	Na_to_K
max	74.000000	38.247000

columns

In [6]:
```
# to display headings of the dataset
data.columns
```

Out[6]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [7]:
```
a=data.dropna(axis=1)
a
```

Out[7]:

	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

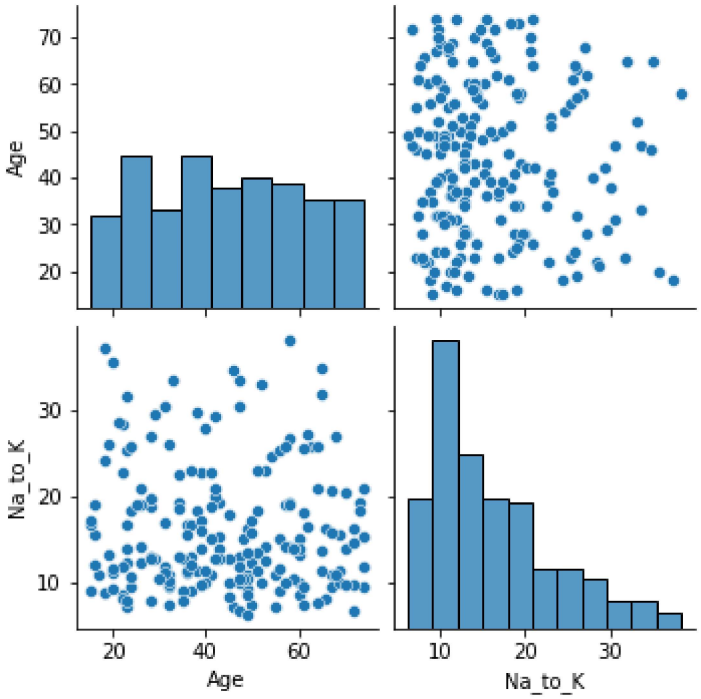
In [8]:
```
a.columns
```

Out[8]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

EDA and Visualization

In [9]:
```
sns.pairplot(a)
```

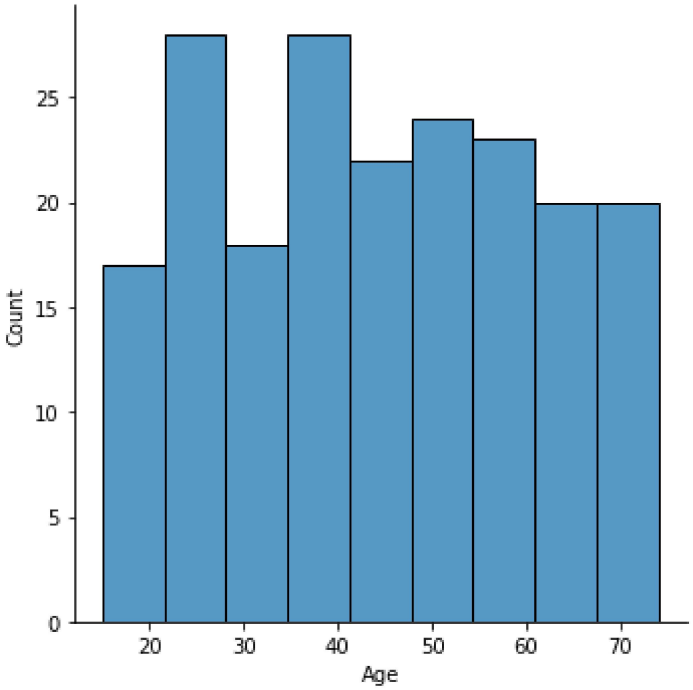
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x162e39e2be0>
```



distribution plot

In [10]:
```
sns.displot(a["Age"])
```

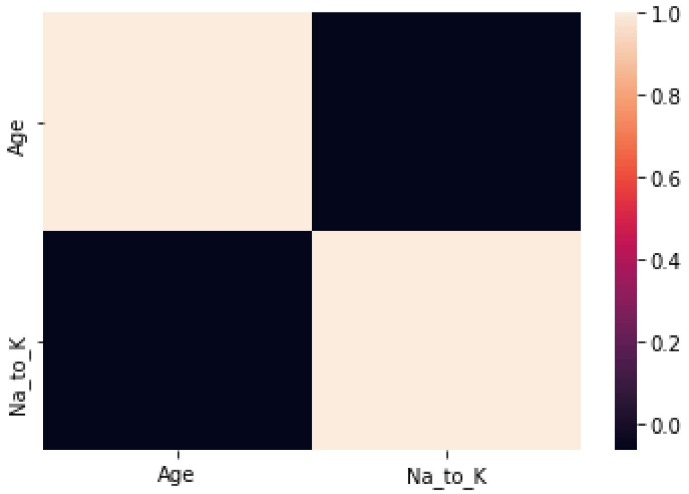
Out[10]: <seaborn.axisgrid.FacetGrid at 0x162e5376820>



correlation

In [11]:
```
dat=data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
sns.heatmap(dat.corr())
```

Out[11]: <AxesSubplot:>



To train the model-Model Building

In [12]:
```
x=a[['Age']]
y=a['Age']
```

In [13]:
```
# to split my dataset into training and test data
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 [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]: LinearRegression()

In [15]:
```
print(lr.intercept_)
```

```
0.0
```

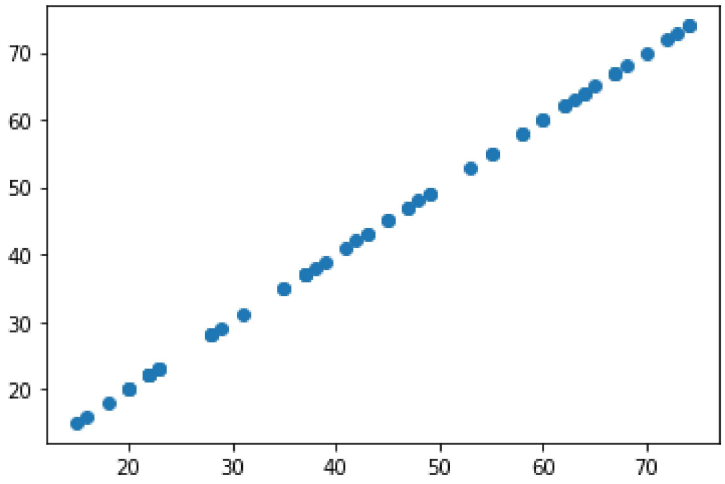
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Age	1.0

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x162e5c6f670>



In [18]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]: 1.0

Ridge regression

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]: 0.999999919557241

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]: 0.9999999196255493

Lasso regression

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[23]: 0.9984237671579392

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: 0.9984224275535903