Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# To import dataset
df=pd.read_csv('drug csv')
df
```

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

In [3]:

```
# To display top 10 rows
df.head(10)
```

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
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

Data Cleaning and Pre-Processing

In [4]:

```
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 [5]:
```
df.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
max	74.000000	38.247000

In [6]:
```
df.columns
```

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

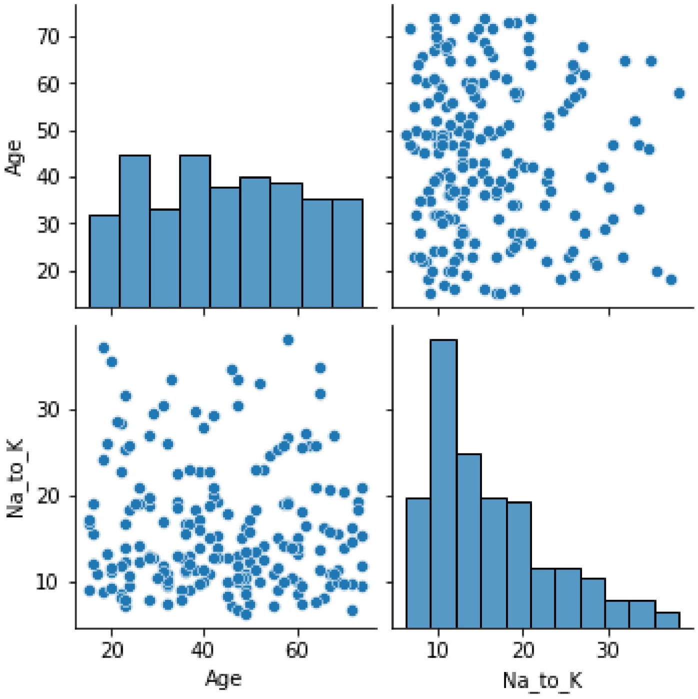
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

EDA and Visualization

In [8]:
```
sns.pairplot(a)
```

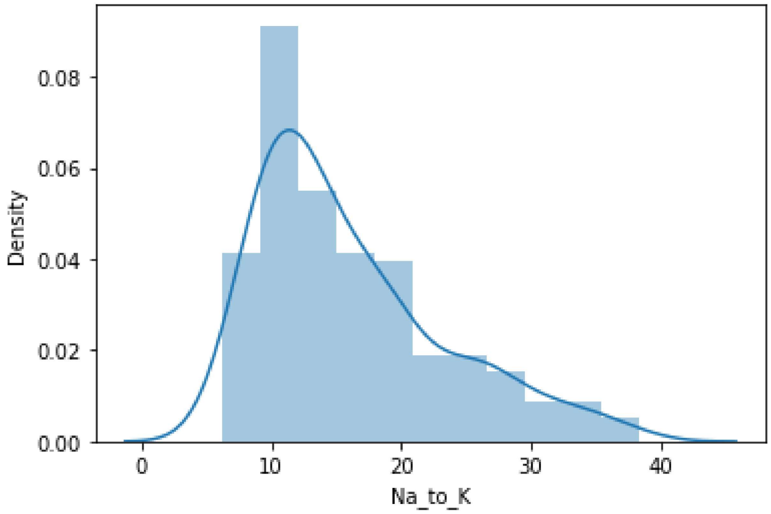
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x18efd21a910>
```



```
In [9]: sns.distplot(a['Na_to_K'])
```

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

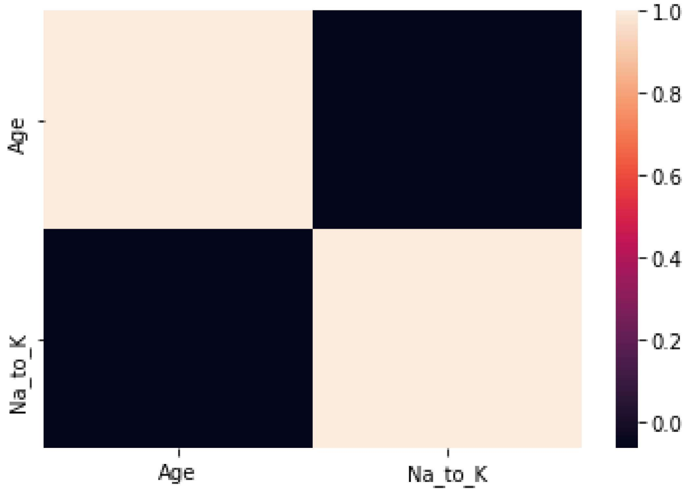
```
Out[9]: <AxesSubplot:xlabel='Na_to_K', ylabel='Density'>
```



```
In [10]: a1=a[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
```

```
In [11]: sns.heatmap(a1.corr())
```

```
Out[11]: <AxesSubplot:>
```



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [12]:

```
x=a1[['Age']]
y=a1['Na_to_K']
```

To split my dataset into training and test data

In [13]:

```
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_)
```

```
18.48275235837274
```

In [16]:

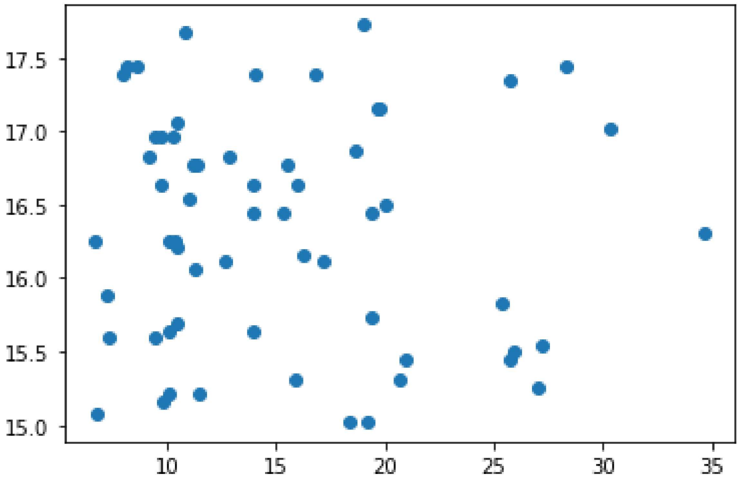
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Age	-0.047381

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

Out[17]: <matplotlib.collections.PathCollection at 0x18eff4cfc70>
```



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

         -0.050327243117328724
```

```
In [19]: from sklearn.linear_model import ElasticNet
         en = ElasticNet()
         en.fit(x_train,y_train)

Out[19]: ElasticNet()
```

```
In [20]: 
         print(en.coef_)

         [-0.04547795]
```

```
In [21]: print(en.intercept_)

         18.399423447617956
```

```
In [22]: 
         print(en.predict(x_test))

         [15.21596728 16.80769536 16.9441292  17.12604098 16.62578358 15.35240111
          16.76221742 17.12604098 17.62629838 16.4438718  15.62526878 15.67074673
          16.12552618 15.89813646 16.12552618 15.48883495 17.39890865 16.85317331
          15.71622467 16.9441292  17.30795276 16.76221742 17.35343071 17.39890865
          15.48883495 16.26196002 16.48934974 15.62526878 17.67177632 17.03508509
          16.4438718  16.80769536 16.62578358 16.30743796 16.98960714 15.30692317
          16.53482769 15.07953344 17.35343071 16.76221742 16.21648207 15.26144522
          17.35343071 15.07953344 15.53431289 17.39890865 16.62578358 16.4438718
          15.12501139 15.76170262 16.26196002 16.26196002 15.67074673 15.57979084
          15.85265851 15.26144522 15.35240111 16.9441292  16.17100413 16.08004824]
```

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

```
-0.04883932126284374
```

In [24]:
```
from sklearn import metrics
```

In [25]:
```

print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 5.772271662498251
```