

DATA COLLECTION

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

```
In [2]: # To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\drug.csv")
sd
```

```
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
sd.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]: sd.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]: # to display summary of statistics
sd.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

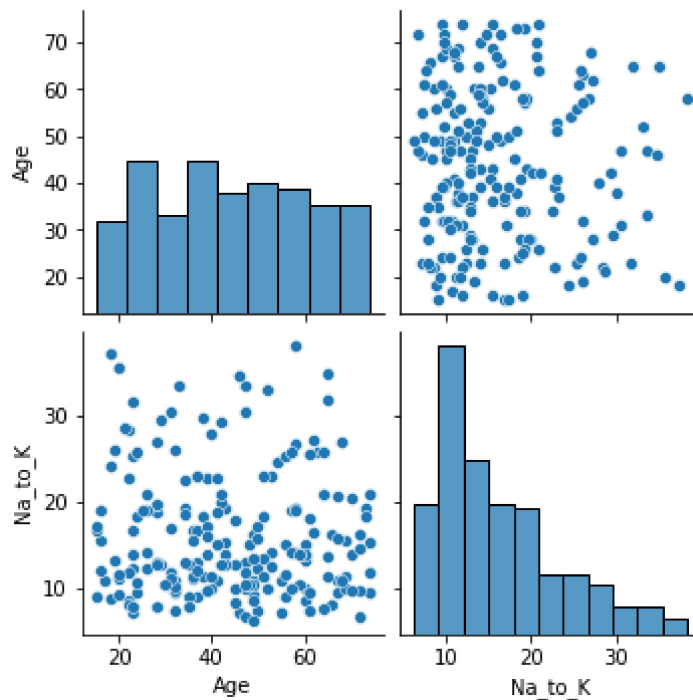
```
In [6]: #to display colums heading
sd.columns
```

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

EDA and visualization

```
In [7]: sns.pairplot(sd)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x1de54318d00>
```

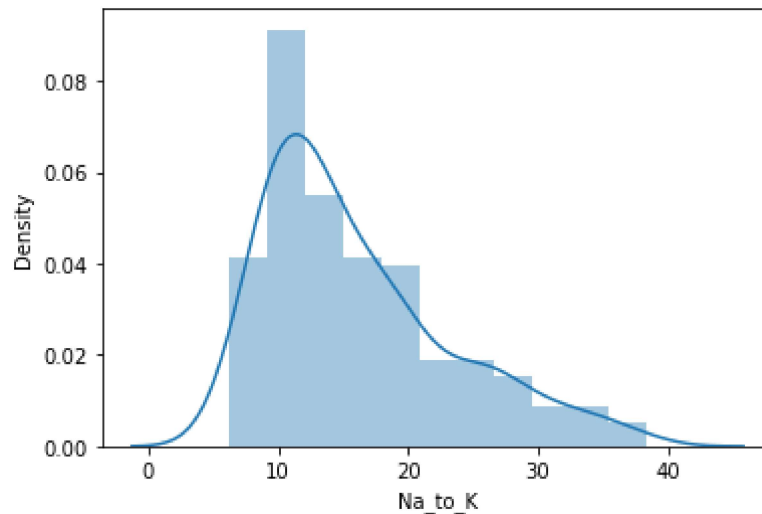


```
In [8]: sns.distplot(sd['Na_to_K'])
```

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 function for histograms).

```
warnings.warn(msg, FutureWarning)
```

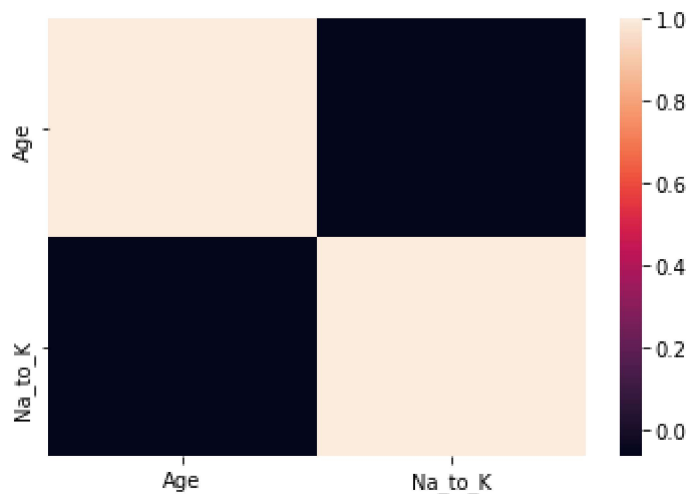
```
Out[8]: <AxesSubplot:xlabel='Na_to_K', ylabel='Density'>
```



```
In [9]: sd1=sd[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
```

```
In [10]: sns.heatmap(sd1.corr())
```

```
Out[10]: <AxesSubplot:>
```



TO TRAIN THE MODEL _MODEL BUILDING

we are going to train Linear Regression model; we need to split out the data into two variables x and y where x is independent on x (output) and y is dependent on x(output) address column as it is not required our model

```
In [11]: x= sd1[['Age']]  
y=sd1['Na_to_K']
```

```
In [12]: # To split my dataset into training data 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=0.4)
```

```
In [13]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[13]: LinearRegression()
```

```
In [14]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[14]: LinearRegression()
```

```
In [15]: print(lr.intercept_)  
  
16.643339716916135
```

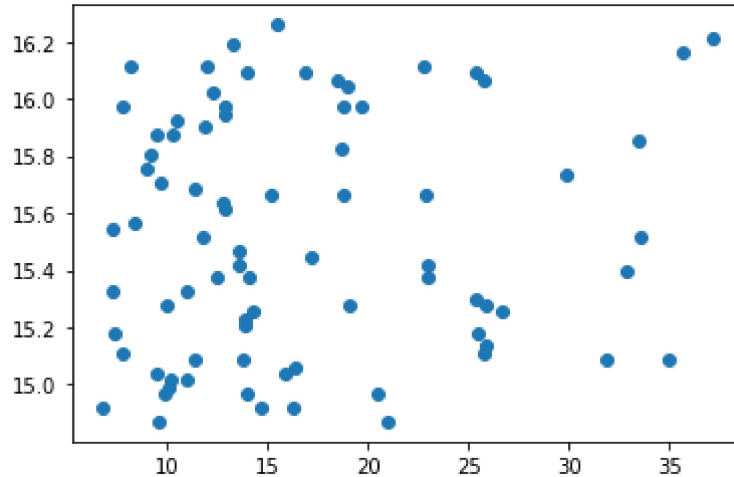
```
In [16]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[16]:
```

	Co-efficient
Age	-0.023963

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

```
Out[17]: <matplotlib.collections.PathCollection at 0x1de565eb220>
```



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

```
-0.018665901221473824
```

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

```
Out[19]: 0.0031192140242291
```

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

```
In [21]: dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

```
Out[21]: Ridge(alpha=10)
```

```
In [22]: dr.score(x_test,y_test)
```

```
Out[22]: -0.01866595996325371
```

```
In [23]: dr.score(x_train,y_train)
```

```
Out[23]: 0.0031192136707031137
```

```
In [24]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[24]: Lasso(alpha=10)
```

```
In [25]: la.score(x_test,y_test)
```

```
Out[25]: -0.021843905401421226
```

In [26]: `la.score(x_train,y_train)`

Out[26]: 0.0

In []: