00utqdtnw

July 31, 2023

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

[ ]:
```
df=pd.read_csv("/content/4_drug200.csv")
df
```

[ ]:
```
     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 x 6 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
   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
```

1 DATA CLEANING AND DATA PREPROCESSING

[ ]:
```
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
```

[ ]:
```
df.describe()
```

[ ]:
```
              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
```

[ ]:
```
df.columns
```

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

[ ]:
```
df1=df.dropna(axis=1)
df1
```

[ ]:
```
     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 x 6 columns]
```

```
[ ]: df1.columns
```

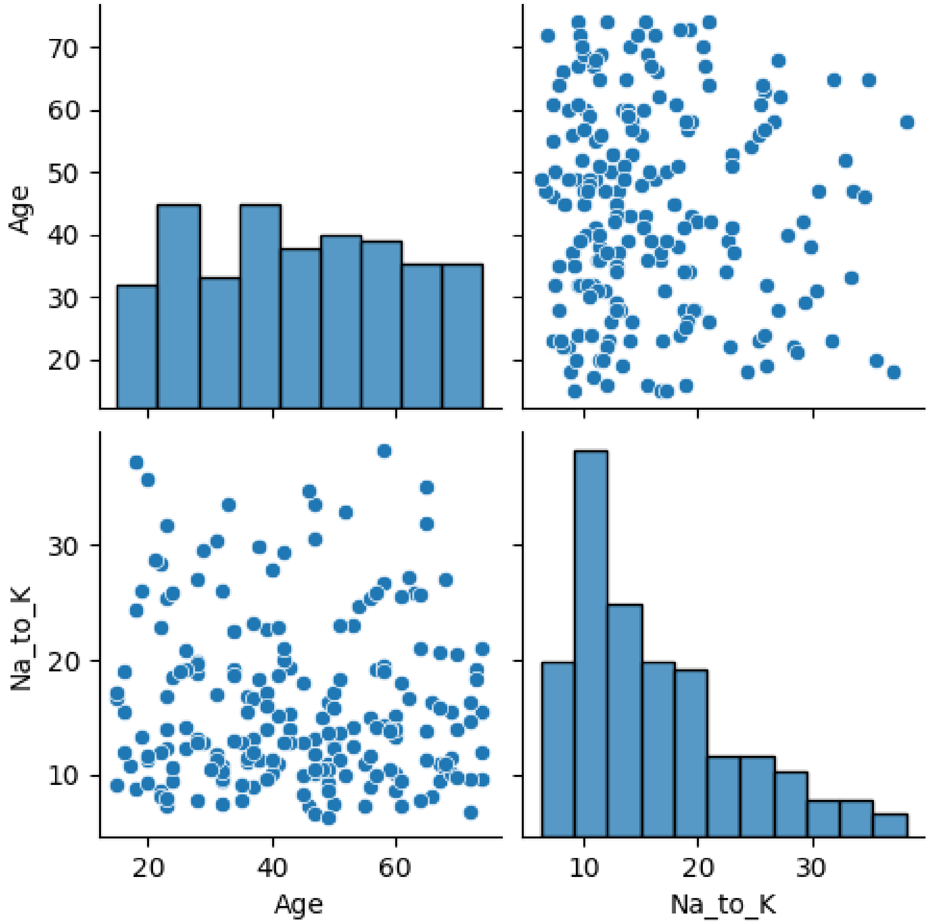
```
[ ]: Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

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

2 EDA AND VISUALIZATION

```
[ ]: sns.pairplot(df1)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x7e9127466770>
```

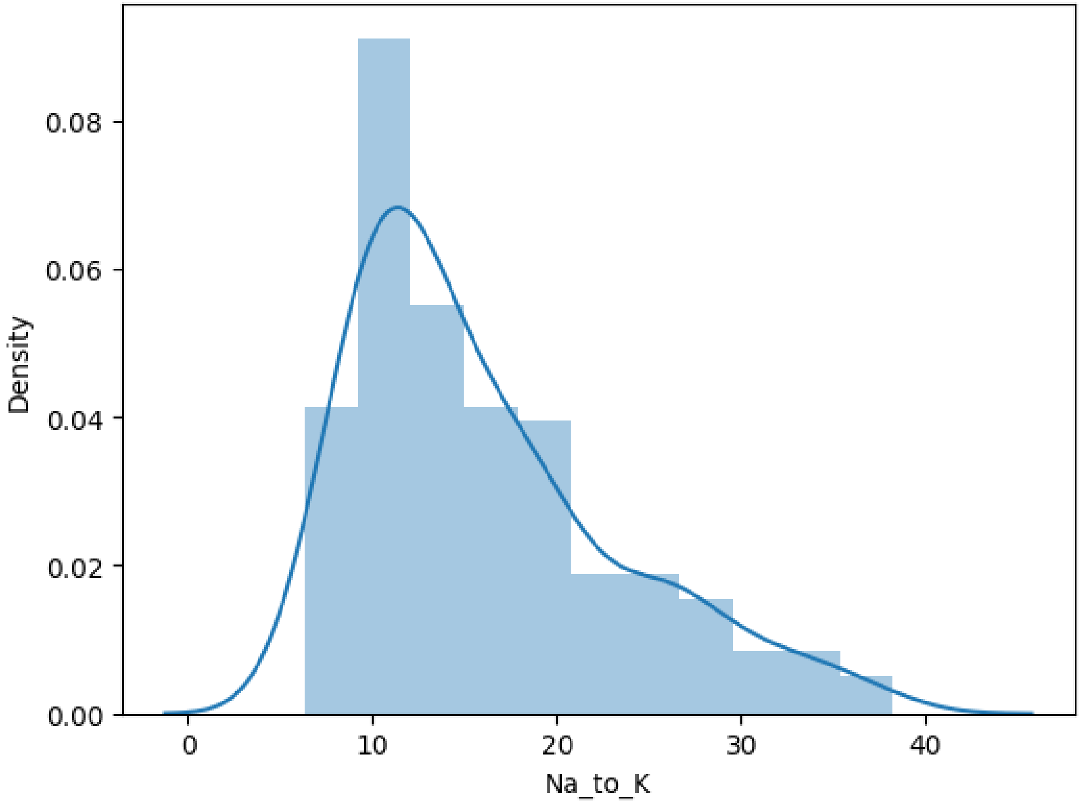


```
[ ]: sns.distplot(df1['Na_to_K'])
```

```
<ipython-input-11-4b6a442fe97b>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

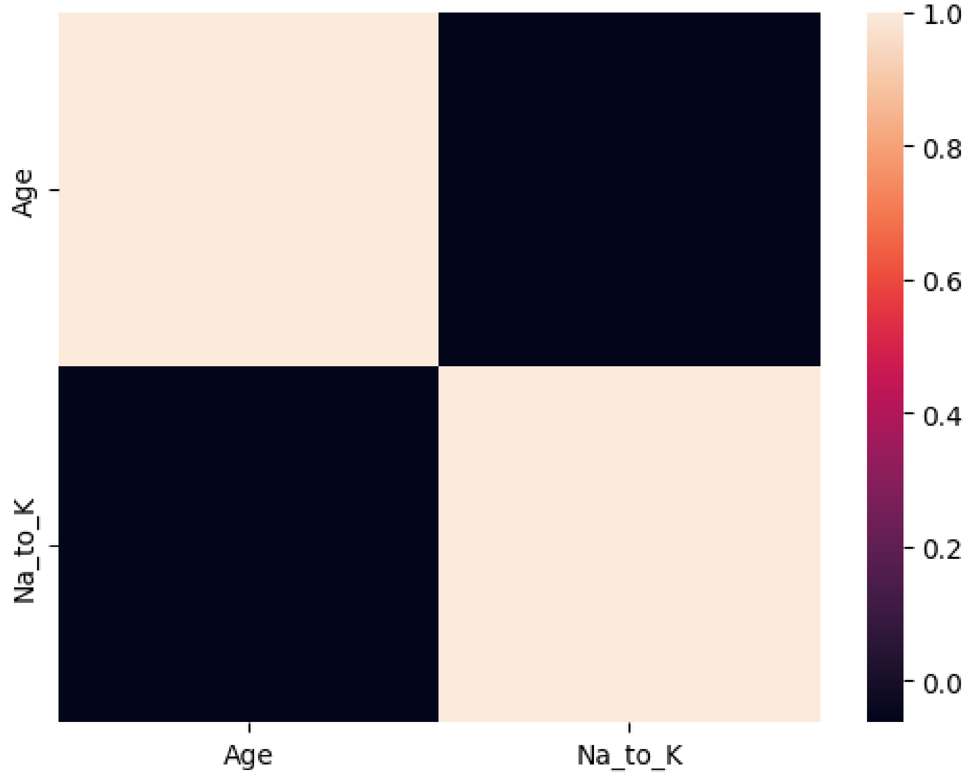
```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['Na_to_K'])
```

[ ]:
```
<Axes: xlabel='Na_to_K', ylabel='Density'>
```



[ ]:
```
sns.heatmap(df1.corr())
```

[ ]:
```
<Axes: >
```



3 TO TRAIN THE MODEL AND MODEL BULDING

[ ]:
```
x=df[['Age','Na_to_K']]
y=df['Na_to_K']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

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

[ ]:
```
LinearRegression()
```

[ ]:
```
lr.intercept_
```

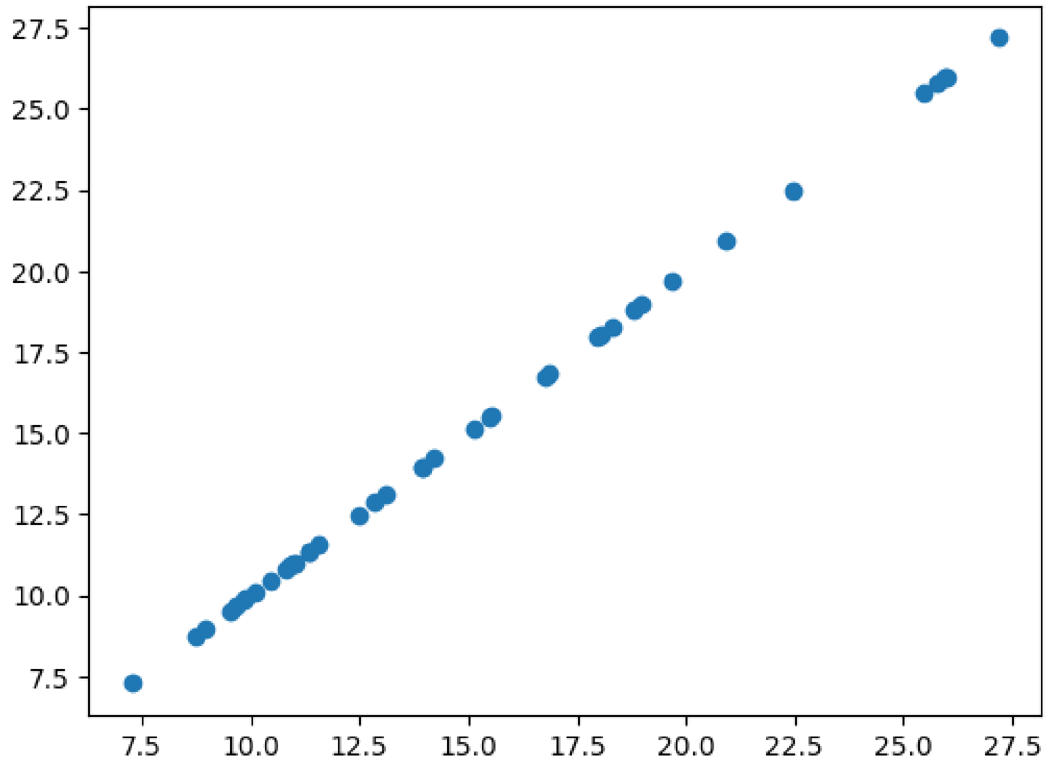
[ ]:
```
-3.552713678800501e-15
```

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

[ ]:
```
         Co-efficient
Age      1.629640e-18
Na_to_K  1.000000e+00
```

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

[ ]:
```
<matplotlib.collections.PathCollection at 0x7e912221cc70>
```



4 ACCURACY

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

[ ]:
```
1.0
```

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

[ ]:
```
1.0
```

[ ]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
```

```
rr.fit(x_train,y_train)
```

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

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

```
[ ]: 0.999998773006138
```

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

```
[ ]: 0.9999988058082202
```

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

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

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

```
[ ]: 0.968148097167346
```

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

```
[ ]: 0.9693674618474863
```

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

```
[ ]: ElasticNet()
```

[ ]:
```
print(en.coef_)
print(en.intercept_)
```

```
[-0.          0.98264968]
0.28287061102032496
```

[ ]:
```
prediction = en.predict(x_test)
prediction
```

```
[ ]: array([20.82909278, 18.94437069, 15.52966305, 15.50411416, 11.10577419,
           10.22138948, 25.62147527,  9.96098731,  9.09330765, 22.34925183,
           14.25221847, 19.61650307, 11.42906594, 13.97511126, 10.00520655,
            7.44147353, 17.92241502, 11.06941615, 12.9138496 , 11.64917946,
           14.0075387 , 10.92693195, 10.99178683, 13.18211297, 16.74520071,
           11.42611799, 10.54762917,  8.88105531, 25.80130016,  9.63179967,
           18.01281879, 12.56107837, 18.76552845, 16.84051772,  9.79197157,
        25.80621341, 18.26044651, 26.99423687, 15.17590917, 25.31587122])
```

[ ]:

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

[ ]:

```
0.9996869834073963
```

[ ]:

```
from sklearn import metrics
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error: ", np.sqrt(metrics.
 ↪mean_squared_error(y_test,prediction)))
```

```
Mean Absolute Error:  0.083686149174037445
Mean Squared Error:  0.009428228438214944
Root Mean Squared Error:  0.09709906507384582
```