pbo732pgr

July 28, 2023

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

[2]:
```
df=pd.read_csv("/content/4_drug200.csv")
df
```

[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 x 6 columns]

[3]:
```
df.head()
```

[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

1 DATA CLEANING AND DATA PREPROCESSING

[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
```

[5]:
```
df.describe()
```

[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

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

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

[7]:
```
df1=df.dropna(axis=1)
df1
```

[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 x 6 columns]

[8]:
```
df1.columns
```

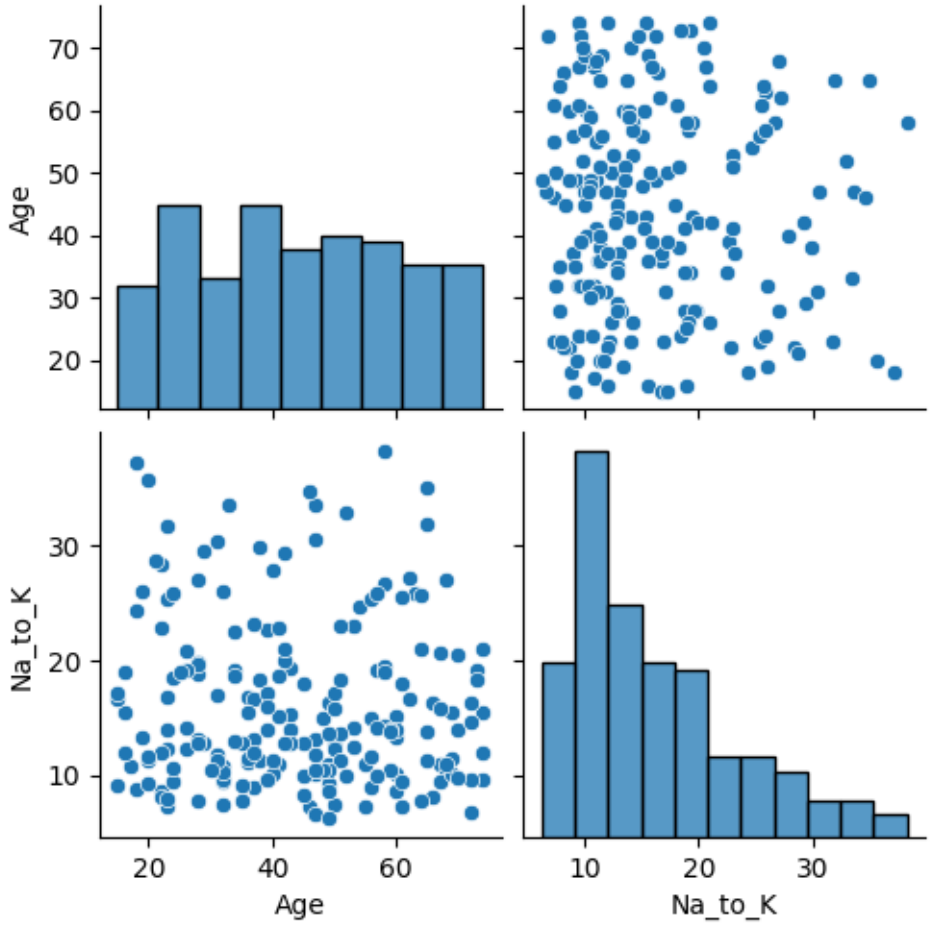
[8]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

[9]:
```
df1=df1[['Age','Na_to_K']]
```

2 EDA AND VISUALIZATION

[10]:
```
sns.pairplot(df1)
```

[10]:
```
<seaborn.axisgrid.PairGrid at 0x7fb37e492530>
```

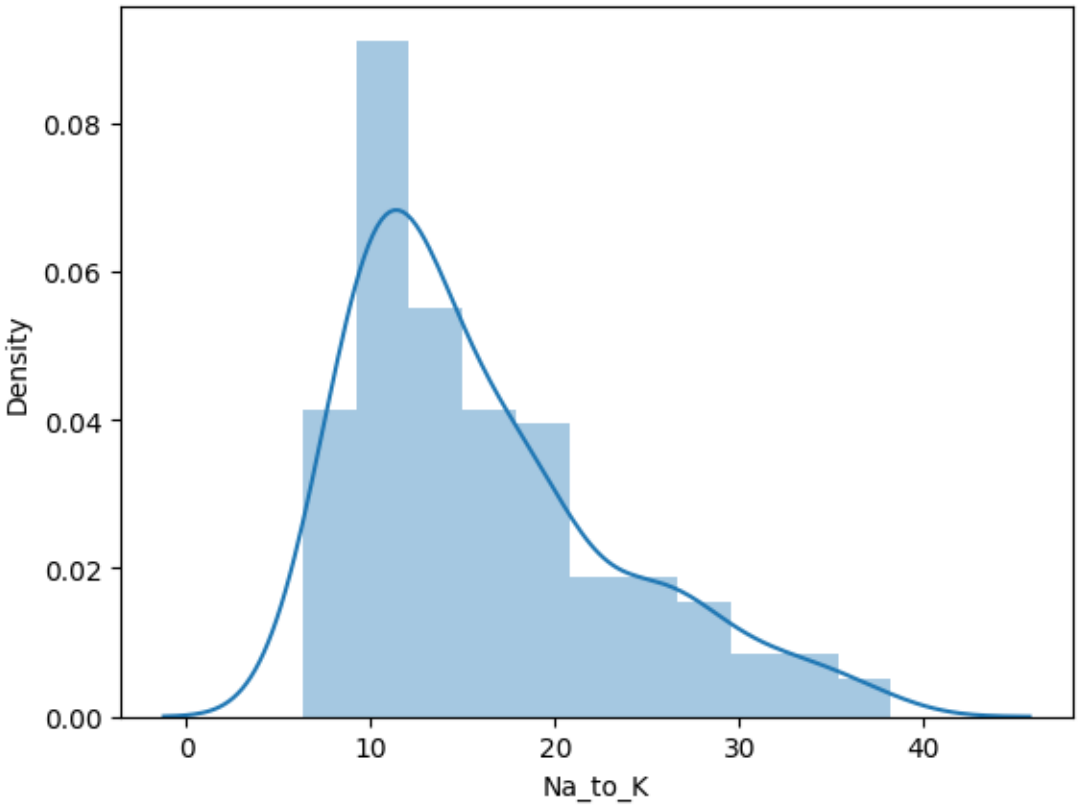


[11]:
```
sns.distplot(df1['Na_to_K'])
```

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

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

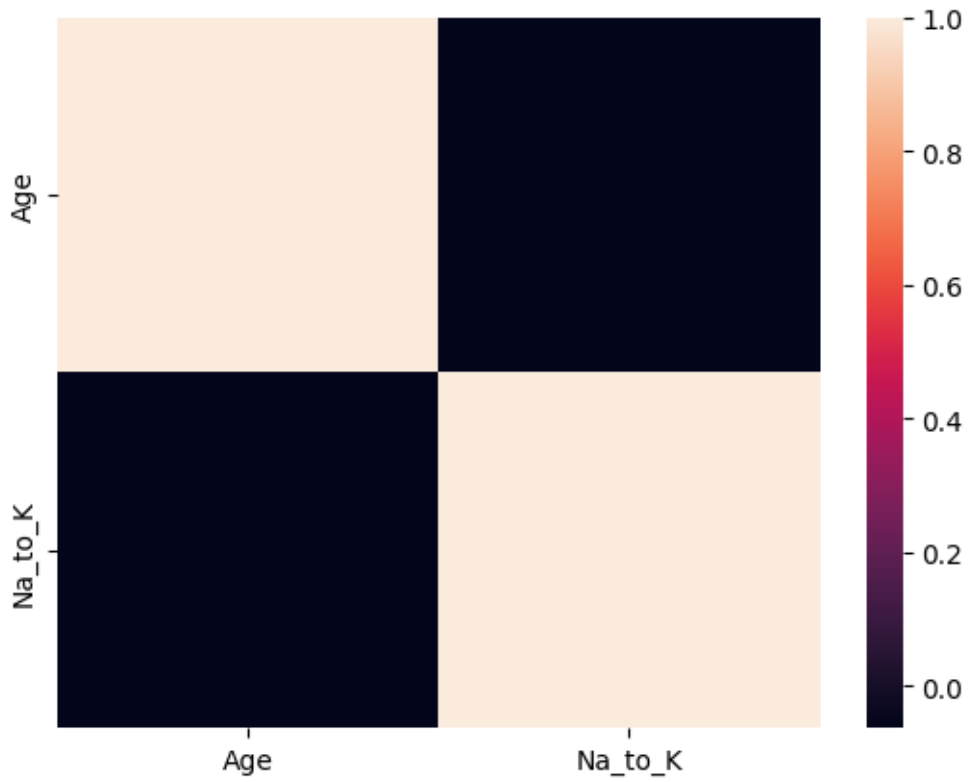
```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

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'])
```

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



[12]:
```
sns.heatmap(df1.corr())
```

[12]: <Axes: >



3 TO TRAIN THE MODEL AND MODEL BULDING

[13]:
```
x=df[['Age','Na_to_K']]
y=df['Na_to_K']
```

[14]:
```
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)
```

[15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[15]:
```
LinearRegression()
```

[16]:
```
lr.intercept_
```

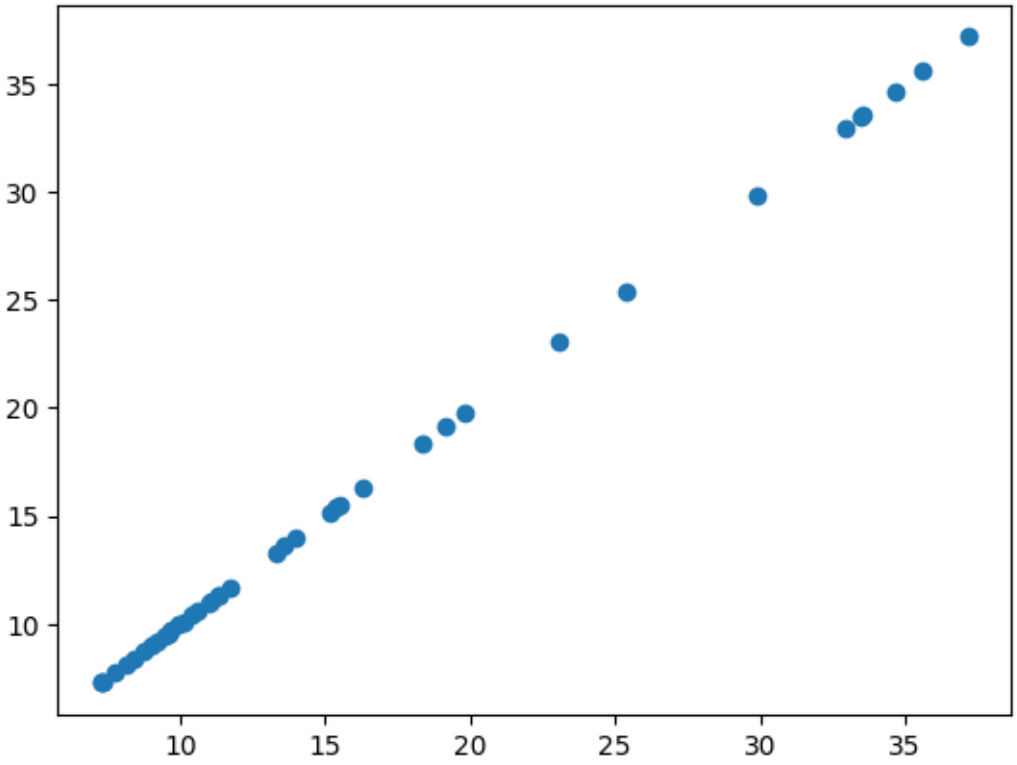
[16]:
```
-3.552713678800501e-15
```

[17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[17]:
```
          Co-efficient
Age      -8.099643e-19
Na_to_K   1.000000e+00
```

[18]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[18]:
```
<matplotlib.collections.PathCollection at 0x7fb37904eb00>
```



4 ACCURACY

[19]:
```
lr.score(x_test,y_test)
```

[19]:
```
1.0
```

[20]:
```
lr.score(x_train,y_train)
```

[20]:
```
1.0
```

[21]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
```

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

[21]: Ridge(alpha=10)

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

[22]: 0.9999979388401831

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

[23]: 0.9999979588976491

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

[24]: Lasso(alpha=10)

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

[25]: 0.9476178435931832

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

[26]: 0.9476204359023367