

ML LAB 8

Implement K Nearest Neighbors algorithm in a given business environment and comment on its efficiency and performance.

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from warnings import filterwarnings
filterwarnings('ignore')
```

```
In [2]: data = pd.read_csv('C:/Users/user/Downloads/archive (2)/drug200.csv')
```

```
In [3]: data.head()
```

```
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

```
In [4]: data.isnull().sum()
```

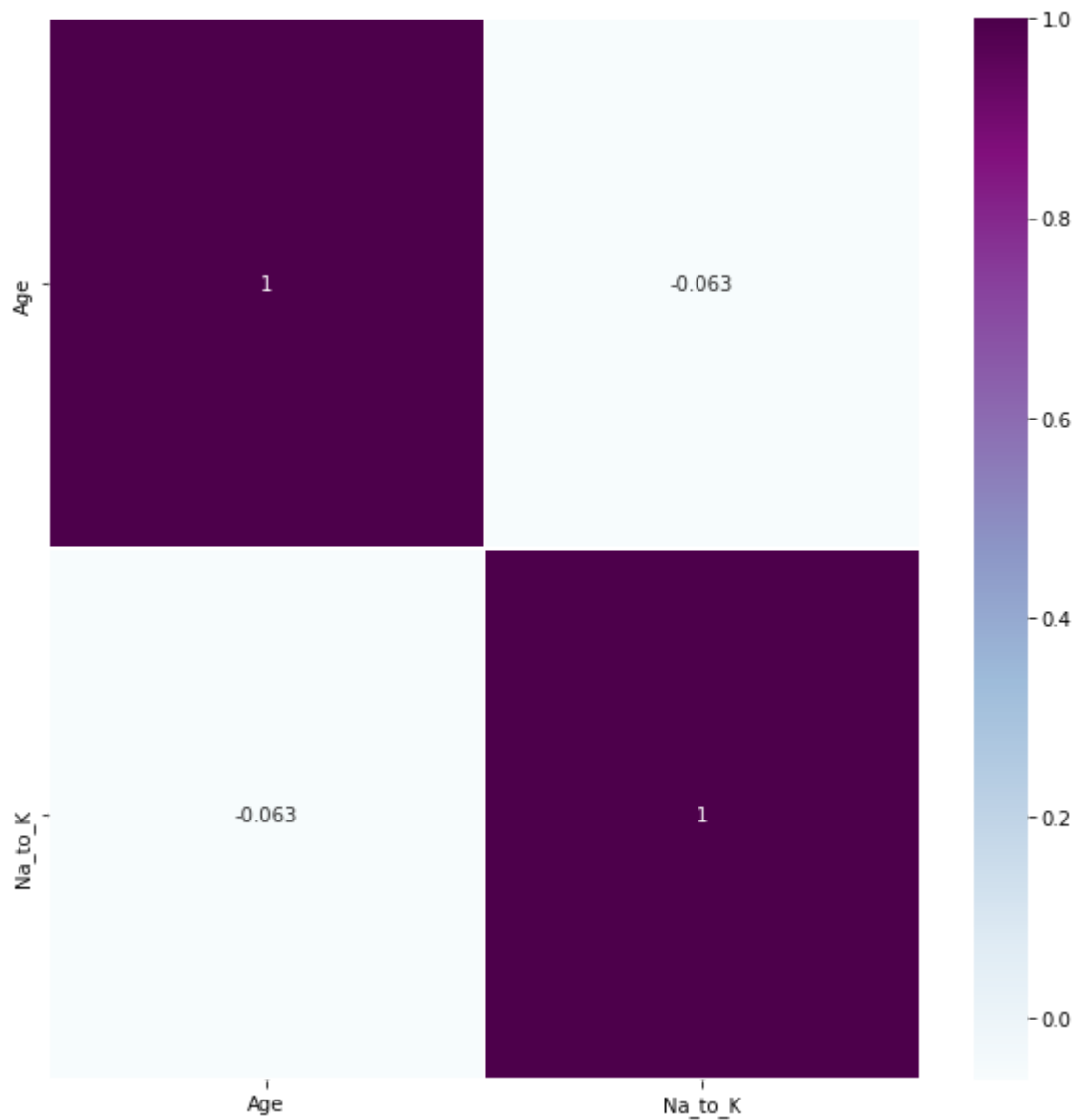
```
Out[4]: Age          0
Sex            0
BP            0
Cholesterol    0
Na_to_K       0
Drug          0
dtype: int64
```

```
In [5]: 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
```

there is no missing values in the data we have 6 columns and 200 rows

```
In [6]: fig, ax = plt.subplots(figsize = (10, 10))  
sns.heatmap(data.corr(), cmap = 'BuPu', cbar = True, linewidth = 0.5, annot = True,  
plt.show())
```

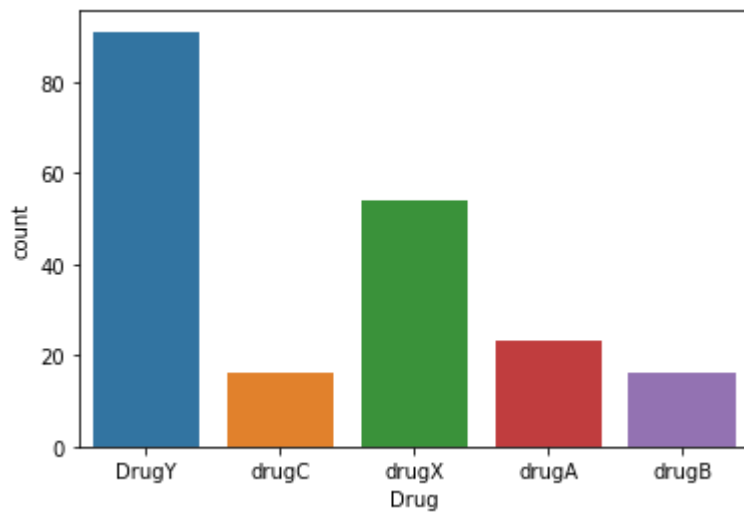


```
In [7]: data['Drug'].value_counts()
```

```
Out[7]: DrugY    91  
drugX    54  
drugA    23  
drugB    16  
drugC    16  
Name: Drug, dtype: int64
```

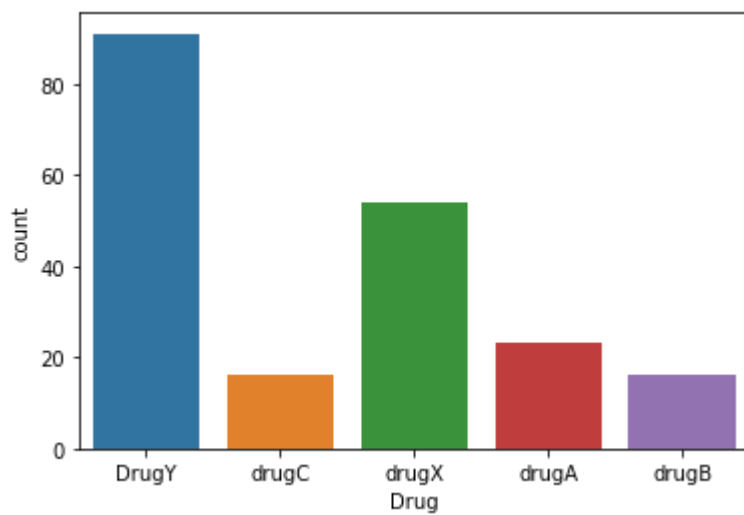
```
In [8]: sns.countplot(x = 'Drug', data= data)
```

```
Out[8]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```



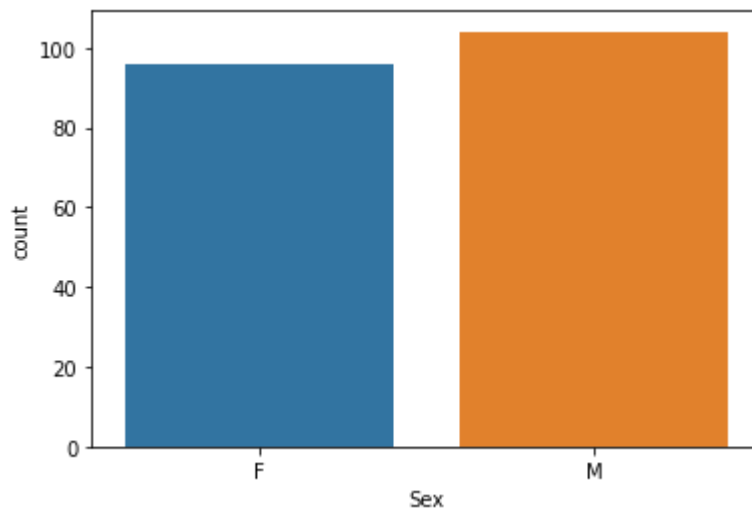
```
In [9]: sns.countplot(x = 'Drug', data= data)
```

```
Out[9]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```



```
In [10]: sns.countplot(x = 'Sex', data= data)
```

```
Out[10]: <AxesSubplot:xlabel='Sex', ylabel='count'>
```

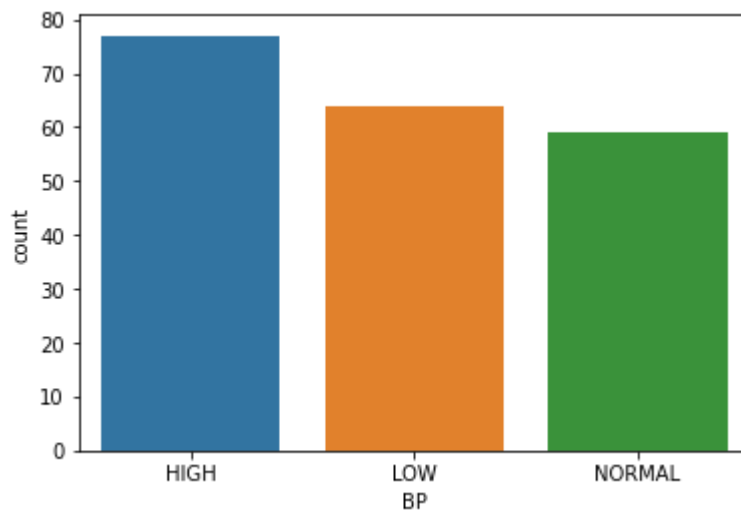


```
In [11]: data['BP'].value_counts()
```

```
Out[11]: HIGH      77  
        LOW       64  
        NORMAL    59  
        Name: BP, dtype: int64
```

```
In [12]: sns.countplot(x = 'BP', data= data)
```

```
Out[12]: <AxesSubplot:xlabel='BP', ylabel='count'>
```

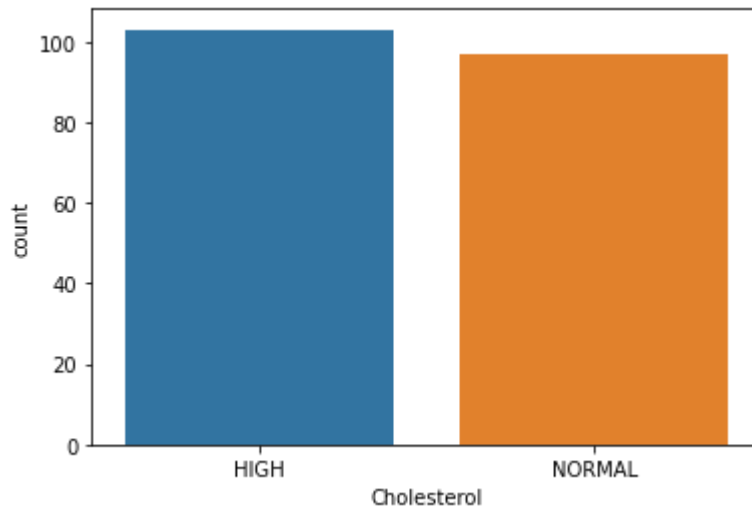


```
In [13]: data['Cholesterol'].value_counts()
```

```
Out[13]: HIGH      103  
         NORMAL    97  
         Name: Cholesterol, dtype: int64
```

```
In [14]: sns.countplot(x = 'Cholesterol', data= data)
```

```
Out[14]: <AxesSubplot:xlabel='Cholesterol', ylabel='count'>
```

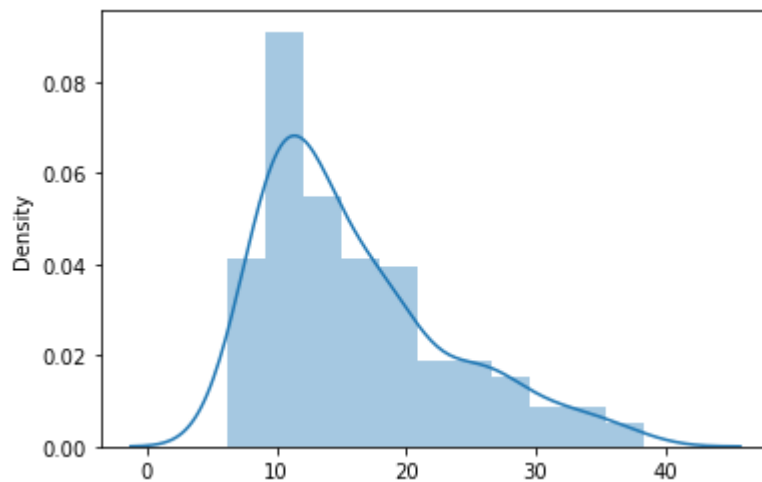


```
In [15]: data['Na_to_K'].describe()
```

```
Out[15]: count      200.000000  
         mean       16.084485  
         std        7.223956  
         min        6.269000  
         25%       10.445500  
         50%       13.936500  
         75%       19.380000  
         max       38.247000  
         Name: Na_to_K, dtype: float64
```

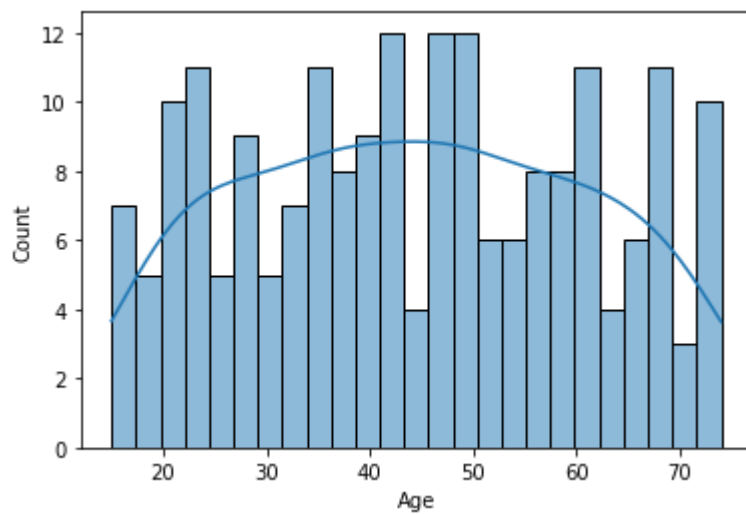
```
In [16]: sns.distplot(x = data['Na_to_K'])
```

```
Out[16]: <AxesSubplot:ylabel='Density'>
```



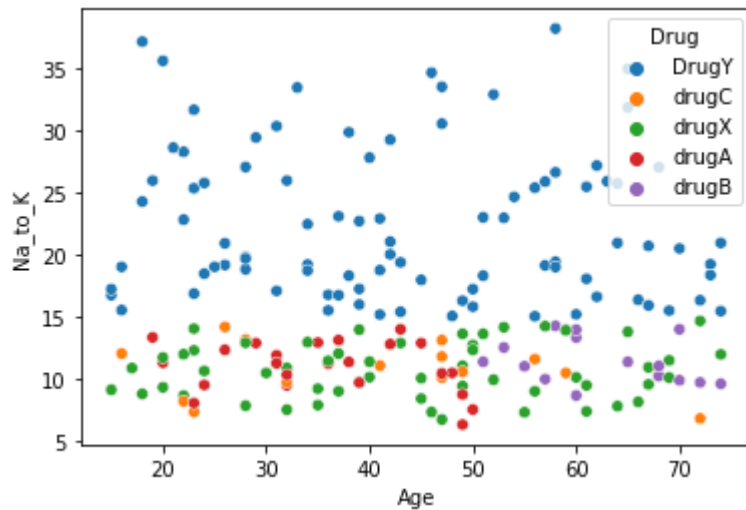
```
In [17]: sns.histplot(x = 'Age', kde=True, bins = 25, data = data)
```

```
Out[17]: <AxesSubplot:xlabel='Age', ylabel='Count'>
```



```
In [18]: sns.scatterplot(x = 'Age', y = 'Na_to_K', data = data, hue = 'Drug')
```

```
Out[18]: <AxesSubplot:xlabel='Age', ylabel='Na_to_K'>
```



In the last fig we find all the items have more than 15 Na_to_K have DrugY type

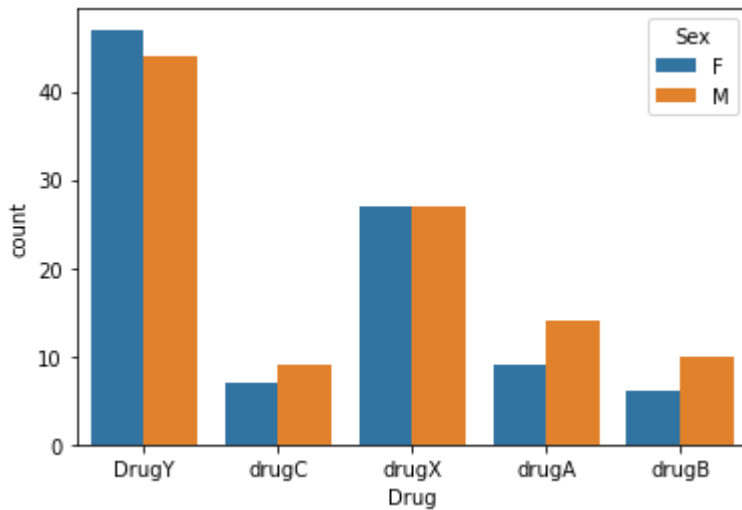
In the next We will find out the number of each Drug type per Sex

```
In [19]: data_sex_drug = data.groupby(['Drug', 'Sex']).size().reset_index(name = 'count')
print(data_sex_drug)
```

	Drug	Sex	count
0	DrugY	F	47
1	DrugY	M	44
2	drugA	F	9
3	drugA	M	14
4	drugB	F	6
5	drugB	M	10
6	drugC	F	7
7	drugC	M	9
8	drugX	F	27
9	drugX	M	27


```
In [20]: sns.countplot(x = 'Drug', data= data, hue = 'Sex')
```

```
Out[20]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```

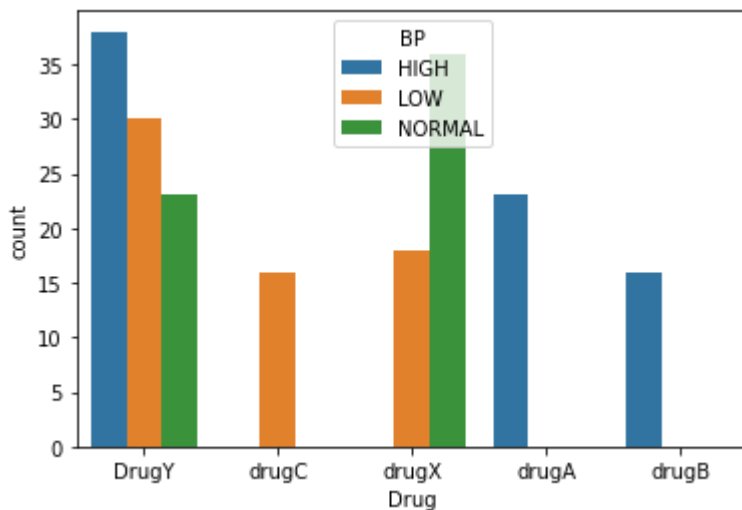


```
In [21]: data_BP_drug = data.groupby(['Drug', 'BP']).size().reset_index(name = 'count')
print(data_BP_drug)
```

	Drug	BP	count
0	DrugY	HIGH	38
1	DrugY	LOW	30
2	DrugY	NORMAL	23
3	drugA	HIGH	23
4	drugB	HIGH	16
5	drugC	LOW	16
6	drugX	LOW	18
7	drugX	NORMAL	36

```
In [22]: sns.countplot(x = 'Drug', data= data, hue = 'BP')
```

```
Out[22]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```

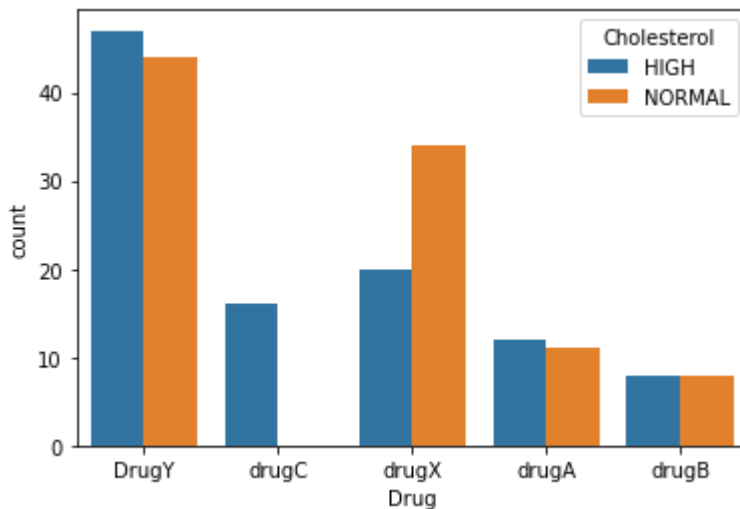


```
In [23]: data_Cholesterol_drug = data.groupby(['Drug', 'Cholesterol']).size().reset_index(name='count')
print(data_Cholesterol_drug)
```

	Drug	Cholesterol	count
0	DrugY	HIGH	47
1	DrugY	NORMAL	44
2	drugA	HIGH	12
3	drugA	NORMAL	11
4	drugB	HIGH	8
5	drugB	NORMAL	8
6	drugC	HIGH	16
7	drugX	HIGH	20
8	drugX	NORMAL	34

```
In [24]: sns.countplot(x = 'Drug', data= data, hue = 'Cholesterol')
```

```
Out[24]: <AxesSubplot:xlabel='Drug', ylabel='count'>
```



```
In [25]: data['Sex'] = data['Sex'].map({'M': 1, 'F': 0})
data['Cholesterol'] = data['Cholesterol'].map({'HIGH': 1, 'NORMAL': 0})
data['Drug'] = data['Drug'].map({'DrugY':1, 'drugC':2, 'drugX':3, 'drugA':4, 'drugB':5})
data.head()
```

```
Out[25]:
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	0	HIGH	1	25.355	1
1	47	1	LOW	1	13.093	2
2	47	1	LOW	1	10.114	2
3	28	0	NORMAL	1	7.798	3
4	61	0	LOW	1	18.043	1

```
In [26]: data.shape
```

```
Out[26]: (200, 6)
```

```
In [27]: data = pd.get_dummies(data)
data.head()
```

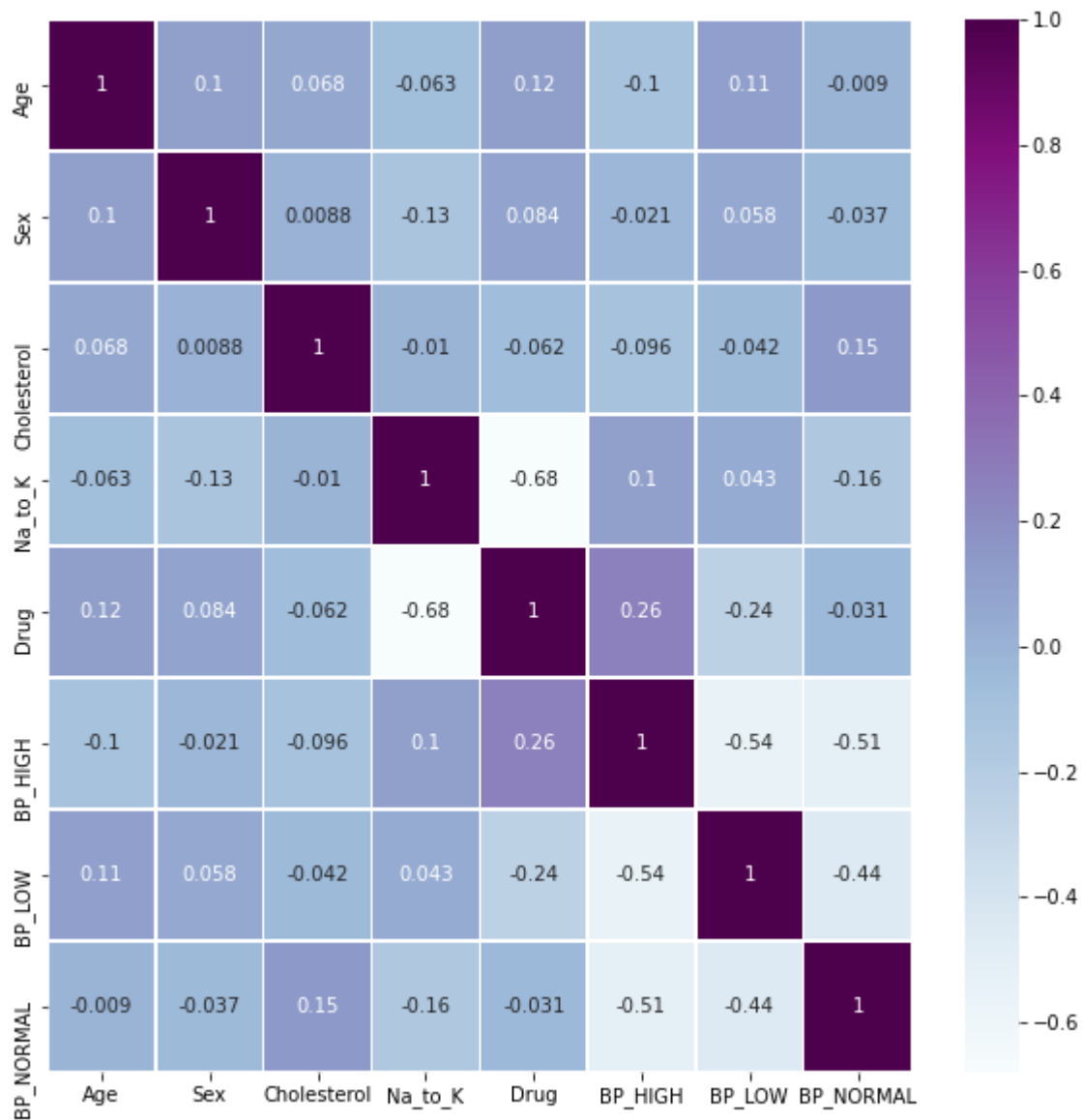
```
Out[27]:
```

	Age	Sex	Cholesterol	Na_to_K	Drug	BP_HIGH	BP_LOW	BP_NORMAL
0	23	0	1	25.355	1	1	0	0
1	47	1	1	13.093	2	0	1	0
2	47	1	1	10.114	2	0	1	0
3	28	0	1	7.798	3	0	0	1
4	61	0	1	18.043	1	0	1	0

```
In [28]: data.shape
```

```
Out[28]: (200, 8)
```

```
In [29]: fig, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(data.corr(), cmap = 'BuPu', cbar = True, linewidth = 0.5, annot = True)
plt.show()
```



```
In [30]: X = data.drop('Drug', axis = 1).values
y = data['Drug'].values.reshape((-1,1))
```

```
In [31]: from sklearn.model_selection import train_test_split
```

```
In [32]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random
print('x train shape {}'.format(X_train.shape))
print('x test shape {}'.format(X_test.shape))
print('y train shape {}'.format(y_train.shape))
print('y test shape {}'.format(y_test.shape))
```

```
x train shape (160, 7)
x test shape (40, 7)
y train shape (160, 1)
y test shape (40, 1)
```

```
In [33]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [34]: KNN_class = KNeighborsClassifier(n_neighbors = 3)
```

```
In [35]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_repo
```

```
In [37]: KNN_class.fit(X_train, y_train)
y_pred = KNN_class.predict(X_test)
print(KNN_class.score(X_train,y_train)*100)
KNN_score = accuracy_score(y_test, y_pred)
print(KNN_score*100)
```

```
83.75
70.0
```

```
In [38]: print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[16  0  0  0  1]
 [ 0  2  0  0  2]
 [ 0  3  6  2  2]
 [ 1  1  0  2  0]
 [ 0  0  0  0  2]]
```

	precision	recall	f1-score	support
1	0.94	0.94	0.94	17
2	0.33	0.50	0.40	4
3	1.00	0.46	0.63	13
4	0.50	0.50	0.50	4
5	0.29	1.00	0.44	2
accuracy			0.70	40
macro avg	0.61	0.68	0.58	40
weighted avg	0.82	0.70	0.72	40

Interpretation:

Of the entire test set, 82% of the drugs were predicted correctly.

```
In [ ]:
```