Machine Learning Project by Udit Arya

1. Introduction

Data Set Problems

In this type dataset of Machine learning model is needed in order **to predict the outcome of the drugs type** that might be suitable for the patient.

Objectives of Notebook

This notebook aims to:

- Dataset exploration using various types of data visualization.
- Build various ML models that can predict drug type.

 **The machine learning models used in this project are:**

1. Linear Logistic Regression
2. Linear Support Vector Machine (SVM)
3. K Neighbours
4. Naive Bayes (Categorical & Gaussian)
5. Decision Tree
6. Random Forest

The following is the **structure of the data set**.

Variable Name	Description	Sample Data
Age	Patient Age	23; 47; ...
Sex	Gender of patient (male or female)	F; M; ...
BP	Levels of blood pressure (high, normal, or low)	HIGH; NORMAL; LOW; ...
Cholesterol	Levels of cholesterol (high or normal)	1.4; 1.3; ...
Na_to_K	Sodium to potassium ratio in blood	25.355; 13.093; ...
Drug	Type of drug	DrugY; drugC; ...

---

2. Importing Libraries

Importing libraries that will be used in this notebook.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

3. Reading Data Set

After importing libraries, we will also **import the dataset** that will be used.

In [6]:
```
df_drug = pd.read_csv("drug200.csv")
```

Read the first 6 rows in the dataset.

In [5]:
```
df_drug.head()
```

Out[5]:

	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

Data type and checking null in dataset.

In [4]:
```
print(df_drug.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
None
```

From the results above, **there are no missing/null value** in this dataset

4. Initial Dataset Exploration

This section will explore raw dataset that has been imported.

4.1 Categorical Variables

In [5]:
```
df_drug.Drug.value_counts()
```

Out[5]:
```
DrugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

It can be seen that from results above, DrugY has more amount than other types of drugs

In [6]:
```
df_drug.Sex.value_counts()
```

Out[6]:
```
M    104
F     96
Name: Sex, dtype: int64
```

The distribution of patient gender is balanced.

In [7]:
```
df_drug.BP.value_counts()
```

Out[7]:
```
HIGH      77
LOW       64
NORMAL    59
Name: BP, dtype: int64
```

The distribution of blood pressure level is balanced.

In [8]:
```
df_drug.Cholesterol.value_counts()
```

Out[8]:
```
HIGH      103
NORMAL     97
Name: Cholesterol, dtype: int64
```

The distribution of cholesterol level is balanced.

4.2 Numerical Variables

This section will show mean, count, std, min, max and others using describe function. The skewness value for each numerical variables will also shown in this section.

In [9]:
```
df_drug.describe()
```

Out[9]:

	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 [10]:
```
skewAge = df_drug.Age.skew(axis = 0, skipna = True)
print('Age skewness: ', skewAge)
```

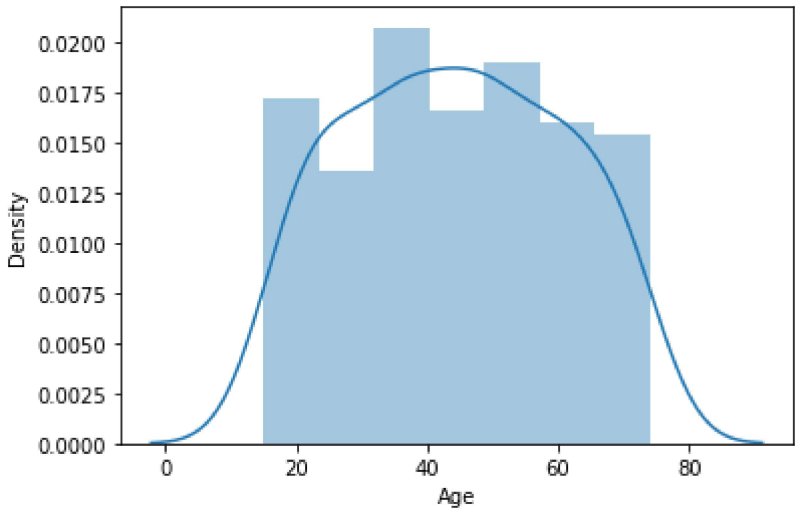
```
Age skewness:  0.03030835703000607
```

In [11]:
```
skewNatoK = df_drug.Na_to_K.skew(axis = 0, skipna = True)
print('Na to K skewness: ', skewNatoK)
```

```
Na to K skewness:  1.039341186028881
```

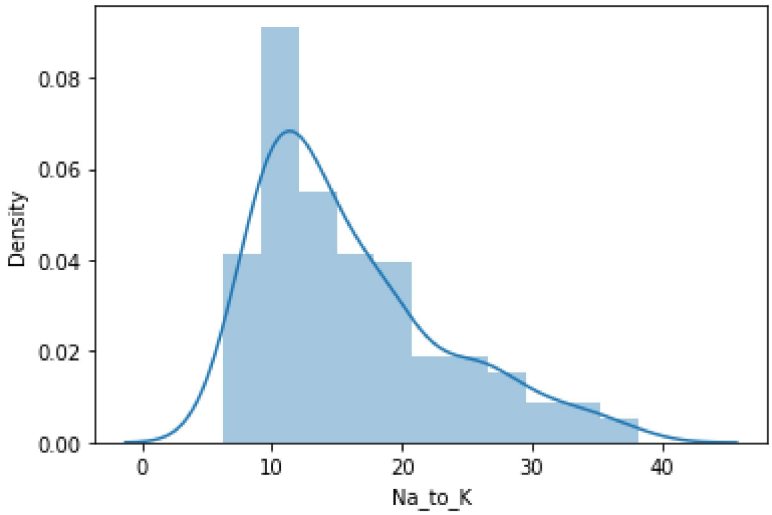
In [12]:
```
sns.distplot(df_drug['Age']);
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```



In [13]:
```
sns.distplot(df_drug['Na_to_K']);
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```



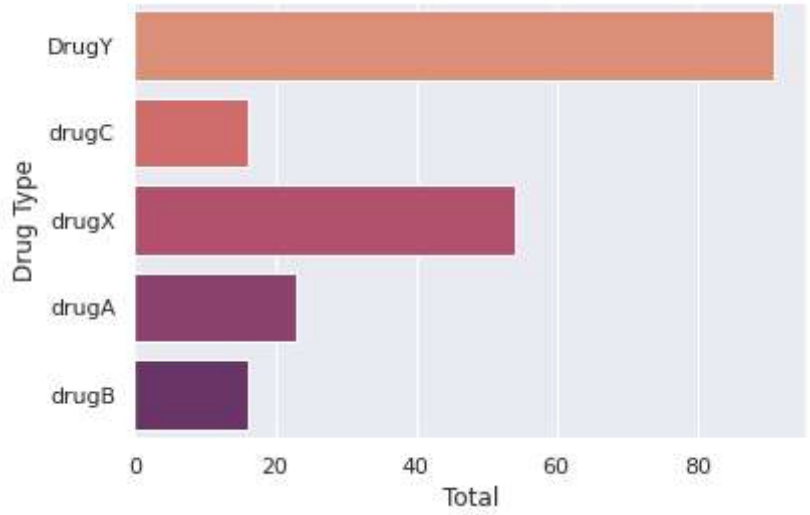
The distribution of **'Age'** column is **symetric**, since the skewness value between -0.5 and 0.5
The distribution of **'Na_to_K'** column is **moderately skewed**, since the skewness value is ***between 0.5 and 1***. It can also be seen from the histogram for 'Na_to_K' column

5. EDA

This section will explore variables in the dataset using different various plots/charts.

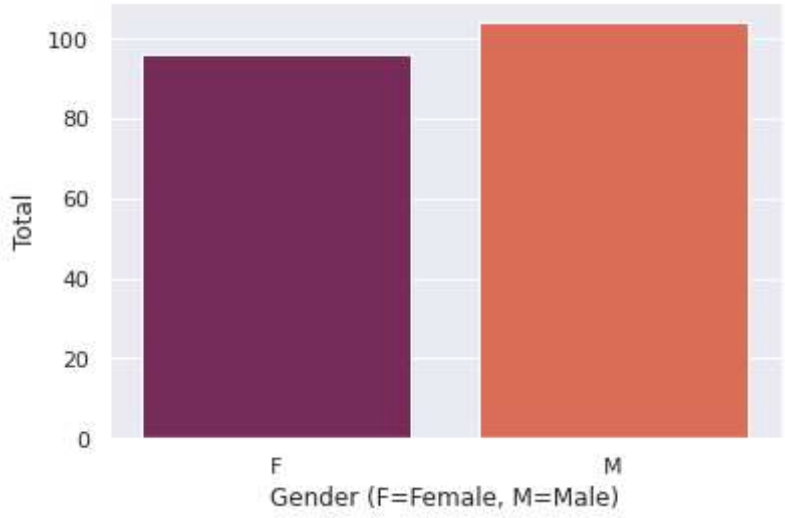
5.1 Drug Type Distribution

In [14]:
```
sns.set_theme(style="darkgrid")
sns.countplot(y="Drug", data=df_drug, palette="flare")
plt.ylabel('Drug Type')
plt.xlabel('Total')
plt.show()
```



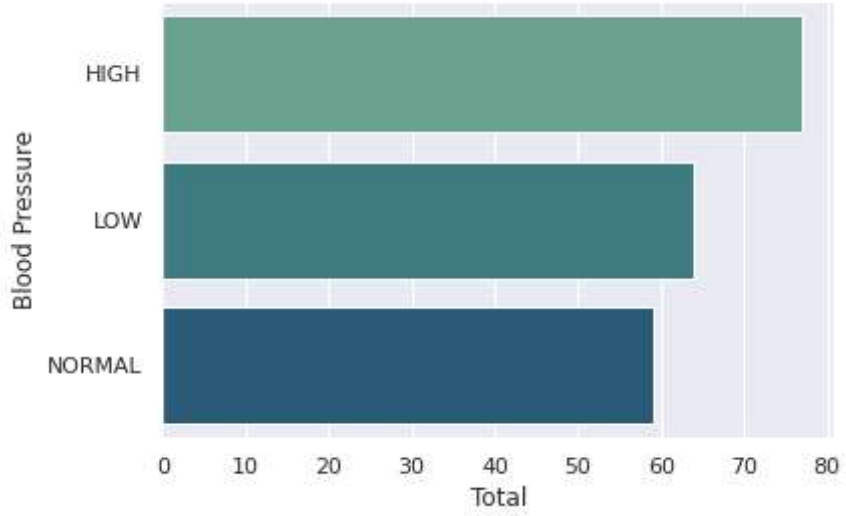
5.2 Gender Distribution

In [15]:
```
sns.set_theme(style="darkgrid")
sns.countplot(x="Sex", data=df_drug, palette="rocket")
plt.xlabel('Gender (F=Female, M=Male)')
plt.ylabel('Total')
plt.show()
```



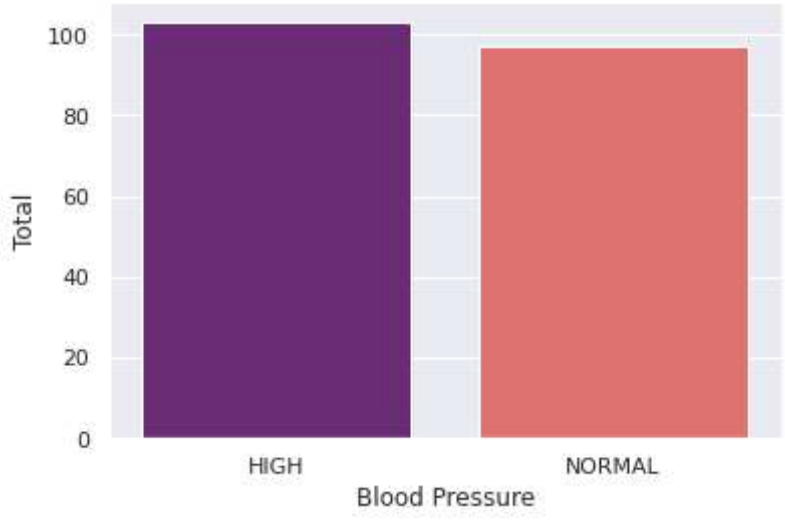
5.3 Blood Pressure Distribution

In [16]:
```
sns.set_theme(style="darkgrid")
sns.countplot(y="BP", data=df_drug, palette="crest")
plt.ylabel('Blood Pressure')
plt.xlabel('Total')
plt.show()
```



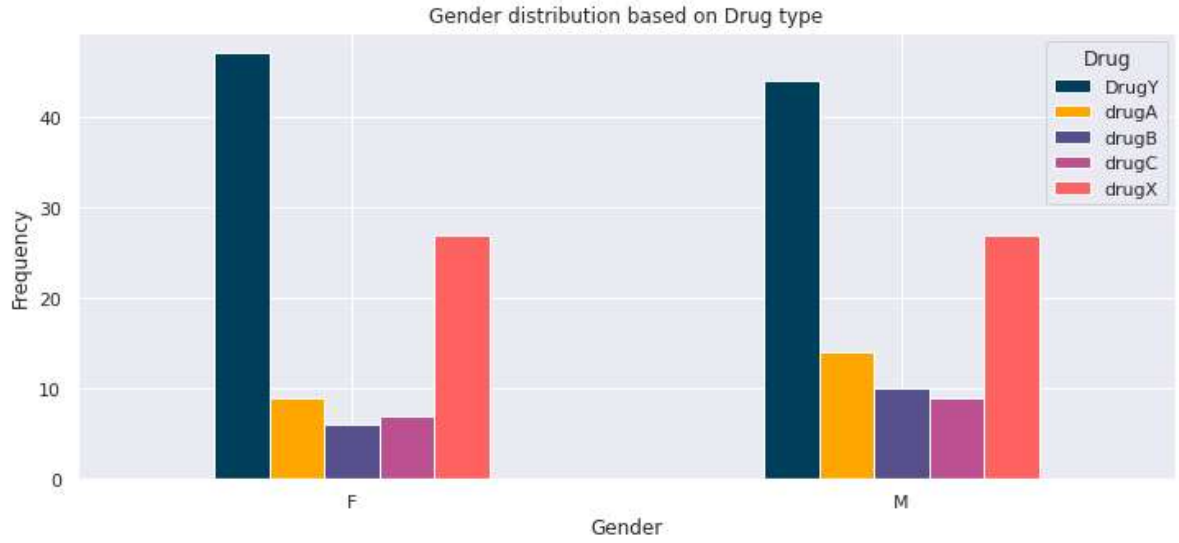
5.4 Cholesterol Distribution

In [17]:
```
sns.set_theme(style="darkgrid")
sns.countplot(x="Cholesterol", data=df_drug, palette="magma")
plt.xlabel('Blood Pressure')
plt.ylabel('Total')
plt.show()
```



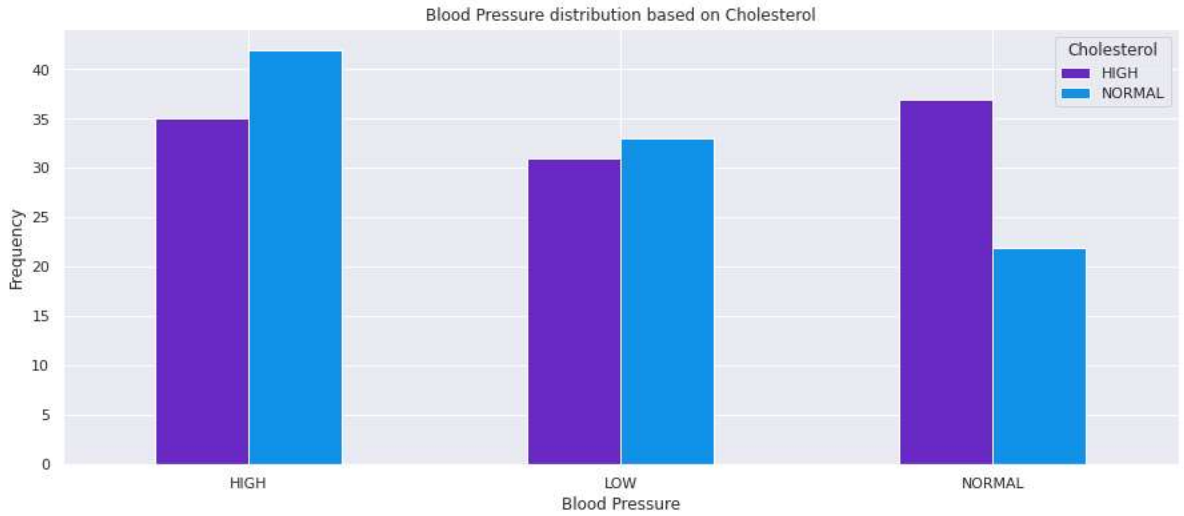
5.5 Gender Distribution based on Drug Type

In [18]:
```
pd.crosstab(df_drug.Sex,df_drug.Drug).plot(kind="bar",figsize=(12,5),color=['#
plt.title('Gender distribution based on Drug type')
plt.xlabel('Gender')
plt.xticks(rotation=0)
plt.ylabel('Frequency')
plt.show()
```



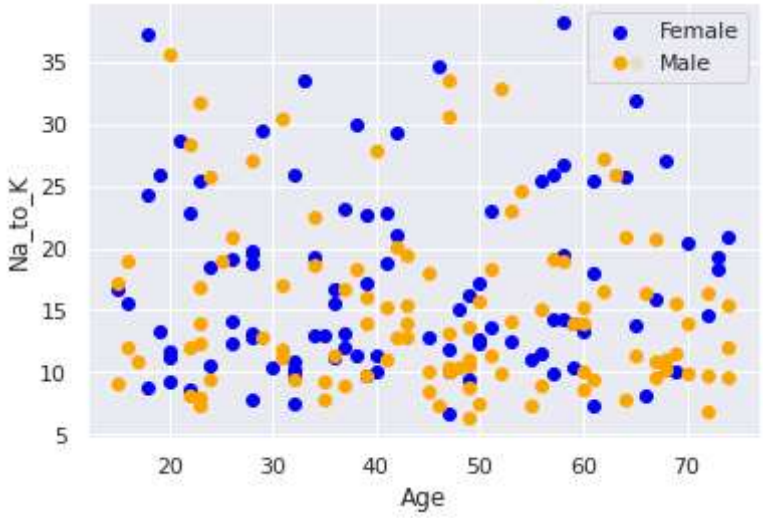
5.6 Blood Pressure Distribution based on Cholesetrol

In [19]:
```
pd.crosstab(df_drug.BP,df_drug.Cholesterol).plot(kind="bar",figsize=(15,6),col
plt.title('Blood Pressure distribution based on Cholesterol')
plt.xlabel('Blood Pressure')
plt.xticks(rotation=0)
plt.ylabel('Frequency')
plt.show()
```



5.7 Sodium to Potassium Distribution based on Gender and Age

In [20]:
```
plt.scatter(x=df_drug.Age[df_drug.Sex=='F'], y=df_drug.Na_to_K[(df_drug.Sex=='
plt.scatter(x=df_drug.Age[df_drug.Sex=='M'], y=df_drug.Na_to_K[(df_drug.Sex=='
plt.legend(["Female", "Male"])
plt.xlabel("Age")
plt.ylabel("Na_to_K")
plt.show()
```



6. Dataset Preparation

This section will prepare the dataset before building the machine learning models.

6.1 Data Binning

6.1.1 Age

The ag will be divided into **7 age categories**:

- Below 20 y.o.
- 20 - 29 y.o.
- 30 - 39 y.o.
- 40 - 49 y.o.
- 50 - 59 y.o.
- 60 - 69 y.o.
- Above 70.

In [7]:
```
bin_age = [0, 19, 29, 39, 49, 59, 69, 80]
category_age = ['<20s', '20s', '30s', '40s', '50s', '60s', '>60s']
df_drug['Age_binned'] = pd.cut(df_drug['Age'], bins=bin_age, labels=category_a
df_drug = df_drug.drop(['Age'], axis = 1)
```

6.1.2 Na_to_K

The chemical ratio will be divided into **4 categories**:

- Below 10.
- 10 - 20.
- 20 - 30.
- Above 30.

In [8]:
```
bin_NatoK = [0, 9, 19, 29, 50]
category_NatoK = ['<10', '10-20', '20-30', '>30']
df_drug['Na_to_K_binned'] = pd.cut(df_drug['Na_to_K'], bins=bin_NatoK, labels=
df_drug = df_drug.drop(['Na_to_K'], axis = 1)
```

6.2 Splitting the dataset

The dataset will be split into **70% training and 30% testing**.

In [10]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [24]:
```
X = df_drug.drop(["Drug"], axis=1)
y = df_drug["Drug"]

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, rar
```

6.3 Feature Engineering

The FE method that used is **one-hot encoding**, which is **transforming categorical variables into a form that could be provided to ML algorithms to do a better prediction**.

In [25]:
```
X_train = pd.get_dummies(X_train)
X_test = pd.get_dummies(X_test)
```

In [26]:
```
X_train.head()
```

Out[26]:

	Sex_F	Sex_M	BP_HIGH	BP_LOW	BP_NORMAL	Cholesterol_HIGH	Cholesterol_NORMAL
131	0	1	0	1	0	0	1
96	1	0	0	1	0	1	0
181	1	0	0	0	1	1	0
19	1	0	1	0	0	0	1
153	1	0	0	1	0	0	1

In [27]:
```
X_test.head()
```

Out[27]:

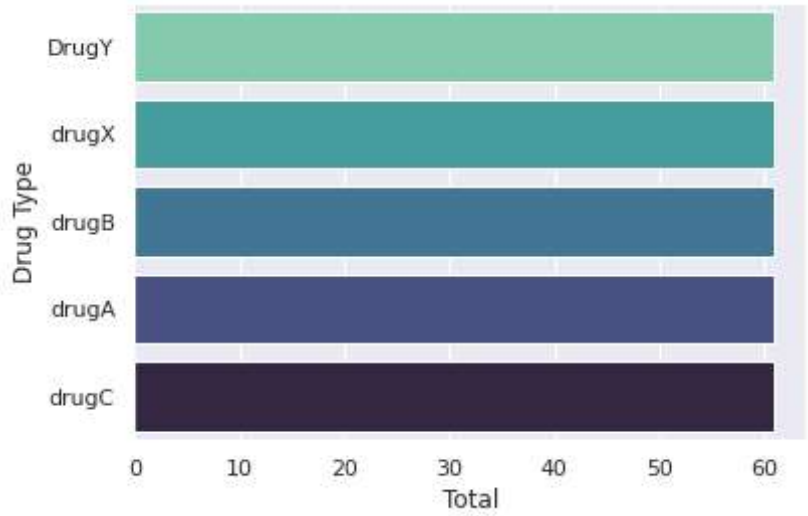
	Sex_F	Sex_M	BP_HIGH	BP_LOW	BP_NORMAL	Cholesterol_HIGH	Cholesterol_NORMAL
18	0	1	0	1	0	1	0
170	1	0	0	0	1	1	0
107	0	1	0	1	0	1	0
98	0	1	1	0	0	0	1
177	0	1	0	0	1	1	0

6.4 SMOTE Technique

Since the number of 'DrugY' is more than other types of drugs, **oversampling is carried out to avoid overfitting**.

In [28]:
```
from imblearn.over_sampling import SMOTE
X_train, y_train = SMOTE().fit_resample(X_train, y_train)
```

In [29]:
```
sns.set_theme(style="darkgrid")
sns.countplot(y=y_train, data=df_drug, palette="mako_r")
plt.ylabel('Drug Type')
plt.xlabel('Total')
plt.show()
```



As can be seen, the distrubtion of drug type are now balanced.

7. Models

7.1 Logistic Regression

In [30]:
```
from sklearn.linear_model import LogisticRegression
LRclassifier = LogisticRegression(solver='liblinear', max_iter=5000)
LRclassifier.fit(X_train, y_train)

y_pred = LRclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
LRAcc = accuracy_score(y_pred,y_test)
print('Logistic Regression accuracy is: {:.2f}%'.format(LRAcc*100))
```

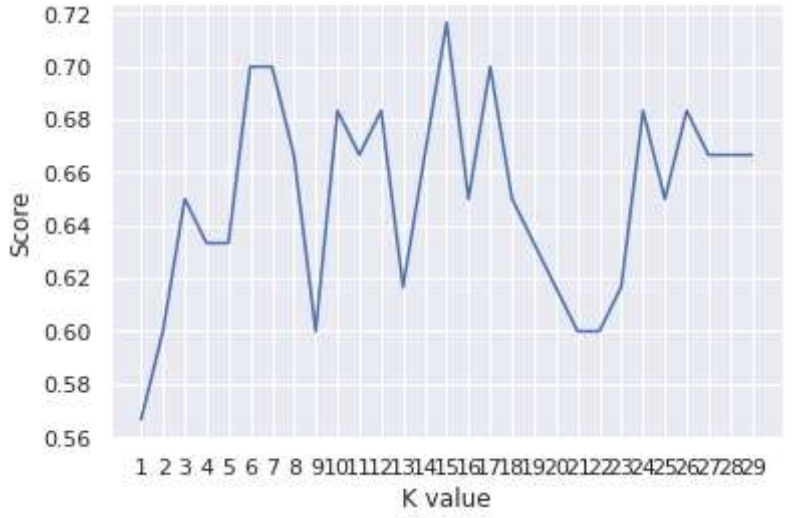
```
              precision    recall  f1-score   support

       DrugY       1.00      0.70      0.82        30
       drugA       0.71      1.00      0.83         5
       drugB       0.75      1.00      0.86         3
       drugC       0.67      1.00      0.80         4
       drugX       0.82      1.00      0.90        18

    accuracy                           0.85        60
   macro avg       0.79      0.94      0.84        60
weighted avg       0.89      0.85      0.85        60

[[21  2  1  2  4]
 [ 0  5  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0 18]]
Logistic Regression accuracy is: 85.00%
```

7.2 K Neighbours

In [31]:
```
from sklearn.neighbors import KNeighborsClassifier
KNclassifier = KNeighborsClassifier(n_neighbors=20)
KNclassifier.fit(X_train, y_train)

y_pred = KNclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
KNAcc = accuracy_score(y_pred,y_test)
print('K Neighbours accuracy is: {:.2f}%'.format(KNAcc*100))
```

```
              precision    recall  f1-score   support

       DrugY       0.65      0.67      0.66        30
       drugA       0.50      0.40      0.44         5
       drugB       0.33      0.33      0.33         3
       drugC       0.33      0.50      0.40         4
       drugX       0.75      0.67      0.71        18

    accuracy                           0.62        60
   macro avg       0.51      0.51      0.51        60
weighted avg       0.63      0.62      0.62        60

[[20  1  1  4  4]
 [ 2  2  1  0  0]
 [ 1  1  1  0  0]
 [ 2  0  0  2  0]
 [ 6  0  0  0 12]]
K Neighbours accuracy is: 61.67%
```

In [32]:
```
scoreListknn = []
for i in range(1,30):
    KNclassifier = KNeighborsClassifier(n_neighbors = i)
    KNclassifier.fit(X_train, y_train)
    scoreListknn.append(KNclassifier.score(X_test, y_test))

plt.plot(range(1,30), scoreListknn)
plt.xticks(np.arange(1,30,1))
plt.xlabel("K value")
plt.ylabel("Score")
plt.show()
KNAccMax = max(scoreListknn)
print("KNN Acc Max {:.2f}%".format(KNAccMax*100))
```




```
KNN Acc Max 71.67%
```

7.3 Support Vector Machine (SVM)

In [33]:
```
from sklearn.svm import SVC
SVCclassifier = SVC(kernel='linear', max_iter=251)
SVCclassifier.fit(X_train, y_train)

y_pred = SVCclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
SVCAcc = accuracy_score(y_pred,y_test)
print('SVC accuracy is: {:.2f}%'.format(SVCAcc*100))
```

```
              precision    recall  f1-score   support

       DrugY       0.82      0.77      0.79        30
       drugA       0.00      0.00      0.00         5
       drugB       0.75      1.00      0.86         3
       drugC       0.67      1.00      0.80         4
       drugX       0.82      1.00      0.90        18

    accuracy                           0.80        60
   macro avg       0.61      0.75      0.67        60
weighted avg       0.74      0.80      0.76        60

[[23  0  1  2  4]
 [ 5  0  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0 18]]
SVC accuracy is: 80.00%

/opt/conda/lib/python3.7/site-packages/sklearn/svm/_base.py:249: ConvergenceW
arning: Solver terminated early (max_iter=251).  Consider pre-processing your
data with StandardScaler or MinMaxScaler.
  % self.max_iter, ConvergenceWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:122
1: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

7.4 Naive Bayes

7.4.1 Categorical NB

In [34]:
```
from sklearn.naive_bayes import CategoricalNB
NBclassifier1 = CategoricalNB()
NBclassifier1.fit(X_train, y_train)

y_pred = NBclassifier1.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
NBAcc1 = accuracy_score(y_pred,y_test)
print('Naive Bayes accuracy is: {:.2f}%'.format(NBAcc1*100))
```

```
              precision    recall  f1-score   support

       DrugY       1.00      0.70      0.82        30
       drugA       0.71      1.00      0.83         5
       drugB       0.75      1.00      0.86         3
       drugC       0.50      0.50      0.50         4
       drugX       0.75      1.00      0.86        18

    accuracy                           0.82        60
   macro avg       0.74      0.84      0.77        60
weighted avg       0.86      0.82      0.81        60

[[21  2  1  2  4]
 [ 0  5  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  2  2]
 [ 0  0  0  0 18]]
Naive Bayes accuracy is: 81.67%
```

7.4.2 Gaussian NB

In [35]:
```
from sklearn.naive_bayes import GaussianNB
NBclassifier2 = GaussianNB()
NBclassifier2.fit(X_train, y_train)

y_pred = NBclassifier2.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
NBAcc2 = accuracy_score(y_pred,y_test)
print('Gaussian Naive Bayes accuracy is: {:.2f}%'.format(NBAcc2*100))
```

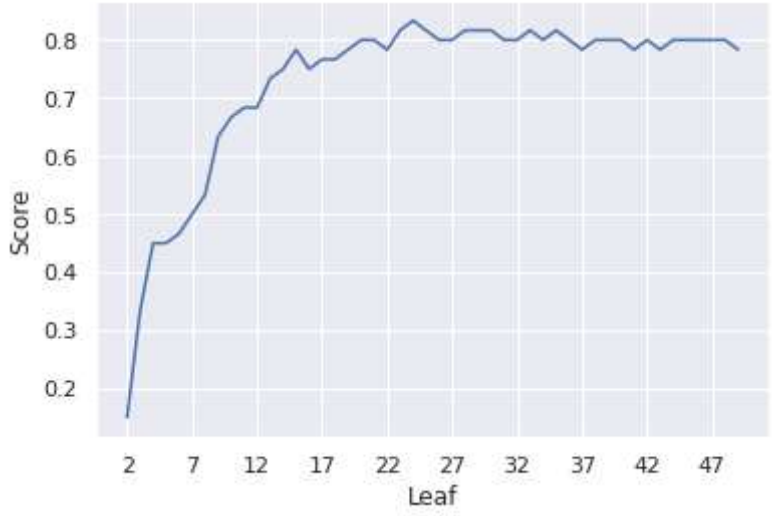
```
              precision    recall  f1-score   support

       DrugY       0.67      0.87      0.75        30
       drugA       0.71      1.00      0.83         5
       drugB       0.75      1.00      0.86         3
       drugC       0.67      0.50      0.57         4
       drugX       1.00      0.39      0.56        18

    accuracy                           0.72        60
   macro avg       0.76      0.75      0.72        60
weighted avg       0.77      0.72      0.70        60

[[26  2  1  1  0]
 [ 0  5  0  0  0]
 [ 0  0  3  0  0]
 [ 2  0  0  2  0]
 [11  0  0  0  7]]
Gaussian Naive Bayes accuracy is: 71.67%
```

7.5 Decision Tree

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
DTclassifier = DecisionTreeClassifier(max_leaf_nodes=20)
DTclassifier.fit(X_train, y_train)

y_pred = DTclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
DTAcc = accuracy_score(y_pred,y_test)
print('Decision Tree accuracy is: {:.2f}%'.format(DTAcc*100))
```

```
              precision    recall  f1-score   support

       DrugY       0.95      0.63      0.76        30
       drugA       0.50      0.80      0.62         5
       drugB       0.75      1.00      0.86         3
       drugC       0.67      1.00      0.80         4
       drugX       0.82      1.00      0.90        18

    accuracy                           0.80        60
   macro avg       0.74      0.89      0.79        60
weighted avg       0.84      0.80      0.80        60

[[19  4  1  2  4]
 [ 1  4  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0 18]]
Decision Tree accuracy is: 80.00%
```

In [37]:
```
scoreListDT = []
for i in range(2,50):
    DTclassifier = DecisionTreeClassifier(max_leaf_nodes=i)
    DTclassifier.fit(X_train, y_train)
    scoreListDT.append(DTclassifier.score(X_test, y_test))

plt.plot(range(2,50), scoreListDT)
plt.xticks(np.arange(2,50,5))
plt.xlabel("Leaf")
plt.ylabel("Score")
plt.show()
DTAccMax = max(scoreListDT)
print("DT Acc Max {:.2f}%".format(DTAccMax*100))
```



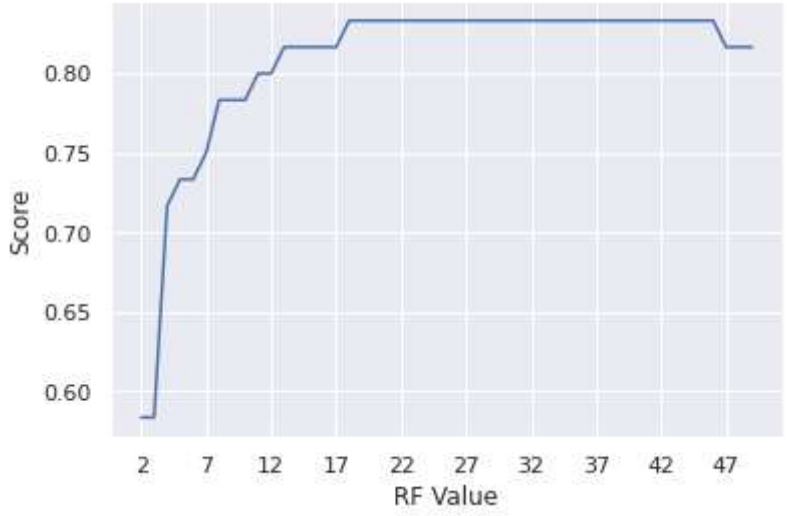

```
DT Acc Max 83.33%
```

7.6 Random Forest

In [38]:
```
from sklearn.ensemble import RandomForestClassifier

RFclassifier = RandomForestClassifier(max_leaf_nodes=30)
RFclassifier.fit(X_train, y_train)

y_pred = RFclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
RFAcc = accuracy_score(y_pred,y_test)
print('Random Forest accuracy is: {:.2f}%'.format(RFAcc*100))
```

```
              precision    recall  f1-score   support

       DrugY       1.00      0.67      0.80        30
       drugA       0.62      1.00      0.77         5
       drugB       0.75      1.00      0.86         3
       drugC       0.67      1.00      0.80         4
       drugX       0.82      1.00      0.90        18

    accuracy                           0.83        60
   macro avg       0.77      0.93      0.83        60
weighted avg       0.88      0.83      0.83        60

[[20  3  1  2  4]
 [ 0  5  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0 18]]
Random Forest accuracy is: 83.33%
```

In [39]:
```
scoreListRF = []
for i in range(2,50):
    RFclassifier = RandomForestClassifier(n_estimators = 1000, random_state =
    RFclassifier.fit(X_train, y_train)
    scoreListRF.append(RFclassifier.score(X_test, y_test))

plt.plot(range(2,50), scoreListRF)
plt.xticks(np.arange(2,50,5))
plt.xlabel("RF Value")
plt.ylabel("Score")
plt.show()
RFAccMax = max(scoreListRF)
print("RF Acc Max {:.2f}%".format(RFAccMax*100))
```




```
RF Acc Max 83.33%
```

8. Model Comparison

In [40]:
```
compare = pd.DataFrame({'Model': ['Logistic Regression', 'K Neighbors', 'K Nei
                        'Accuracy': [LRAcc*100, KNAcc*100, KNAccMax*100, SVCAc
compare.sort_values(by='Accuracy', ascending=False)
```

Out[40]:

	Model	Accuracy
0	Logistic Regression	85.000000
7	Decision Tree Max	83.333333
8	Random Forest	83.333333
9	Random Forest Max	83.333333
4	Categorical NB	81.666667
3	SVM	80.000000
6	Decision Tree	80.000000
2	K Neighbors Max	71.666667
5	Gaussian NB	71.666667
1	K Neighbors	61.666667

From the results, it can be seen that most of ML models can reach up to 80% accuracy in predicting classification of drug type.

In [ ]: