In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [3]:

```
df=pd.read_csv(r"C:\Users\Svijayalakshmi\Downloads\drug200.csv")
df
```

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
...	...	...	...	...	...	...
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 [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
```

In [6]:

```
df['Cholesterol'].value_counts()
```

Out[6]:

```
Cholesterol
HIGH      103
NORMAL     97
Name: count, dtype: int64
```

In [7]:

```
df['BP'].value_counts()
```

Out[7]:

```
BP
HIGH      77
LOW       64
NORMAL    59
Name: count, dtype: int64
```

In [9]:

```
convert={"Sex":{"yes":1,"no":0}}
df=df.replace(convert)
df
```

Out[9]:

	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 [10]:

```
convert={"Sex":{"F":1,"M":2}}
df=df.replace(convert)
df
```

Out[10]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	1	HIGH	HIGH	25.355	drugY
1	47	2	LOW	HIGH	13.093	drugC
2	47	2	LOW	HIGH	10.114	drugC
3	28	1	NORMAL	HIGH	7.798	drugX
4	61	1	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	1	LOW	HIGH	11.567	drugC
196	16	2	LOW	HIGH	12.006	drugC
197	52	2	NORMAL	HIGH	9.894	drugX
198	23	2	NORMAL	NORMAL	14.020	drugX
199	40	1	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [12]:

```
x=["Cholesterol","BP"]
y=["yes","no"]
all_inputs=df[x]
all_classes=df["Drug"]
```

In [13]:

```
(x_train,x_test,y_train,y_test)=train_test_split(all_inputs,all_classes,test_size=0.5)
```

In [14]:

```
clf=DecisionTreeClassifier(random_state=0)
```

In [15]:

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

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_18588\35389094.py in ?()
----> 1 clf.fit(x_train,y_train)

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\tree\_
classes.py in ?(self, X, y, sample_weight, check_input)
    885         self : DecisionTreeClassifier
    886             Fitted estimator.
    887         """
    888 
--> 889         super().fit(
    890             X,
    891             y,
    892             sample_weight=sample_weight,

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\tree\_
classes.py in ?(self, X, y, sample_weight, check_input)
    182             # We can't pass multi_output=True because that would a
llow y to be
    183             # csr.
    184             check_X_params = dict(dtype=DTYPE, accept_sparse="cs
c")
    185             check_y_params = dict(ensure_2d=False, dtype=None)
--> 186             X, y = self._validate_data(
    187                 X, y, validate_separately=(check_X_params, check_y
_params)
    188             )
    189             if issparse(X):

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.p
y in ?(self, X, y, reset, validate_separately, **check_params)
    575                 # :(
    576                 check_X_params, check_y_params = validate_separate
ly
    577                 if "estimator" not in check_X_params:
    578                     check_X_params = {**default_check_params, **ch
eck_X_params}
--> 579                 X = check_array(X, input_name="X", **check_X_param
s)
    580                 if "estimator" not in check_y_params:
    581                     check_y_params = {**default_check_params, **ch
eck_y_params}
    582                 y = check_array(y, input_name="y", **check_y_param
s)

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils
\validation.py in ?(array, accept_sparse, accept_large_sparse, dtype, orde
r, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure
_min_features, estimator, input_name)
    876                         )
    877                     array = xp.astype(array, dtype, copy=False)
    878                 else:
    879                     array = _asarray_with_order(array, order=orde
r, dtype=dtype, xp=xp)
--> 880             except ComplexWarning as complex_warning:
    881                 raise ValueError(
    882                     "Complex data not supported\n{}\n".format(arra
y)
```

```
    883                 ) from complex_warning

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils
\_array_api.py in ?(array, dtype, order, copy, xp)
    181     if xp is None:
    182         xp, _ = get_namespace(array)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:
    188         return xp.asarray(array, dtype=dtype, copy=copy)

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\ge
neric.py in ?(self, dtype)
   1996     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.
ndarray:
   1997         values = self._values
-> 1998         arr = np.asarray(values, dtype=dtype)
   1999         if (
   2000             astype_is_view(values.dtype, arr.dtype)
   2001             and using_copy_on_write()

ValueError: could not convert string to float: 'NORMAL'
```

In [ ]: