Decision Tree Classification

In this application, You will use DecisionTree classification algorithm to build a model from historical data of patients, and their response to different medications. Then you use the trained decision tree to predict the class of a unknown patient, or to find a proper drug for a new patient.

Import required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
%matplotlib inline
```

Load drugsinfo.csv data into drugs dataframe

In [ ]:

In [2]:
```
d=pd.read_csv("./drugsinfo.csv")
```

Check out the info(), head(), and describe() methods on drugs.

In [3]:
```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
Age            200 non-null int64
Sex            200 non-null object
BP             200 non-null object
Cholesterol    200 non-null object
Na_to_K        200 non-null float64
Drug           200 non-null object
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

In [4]:
```
d.head()
```

Out[4]:

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

Out[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

Exploratory Data Analysis

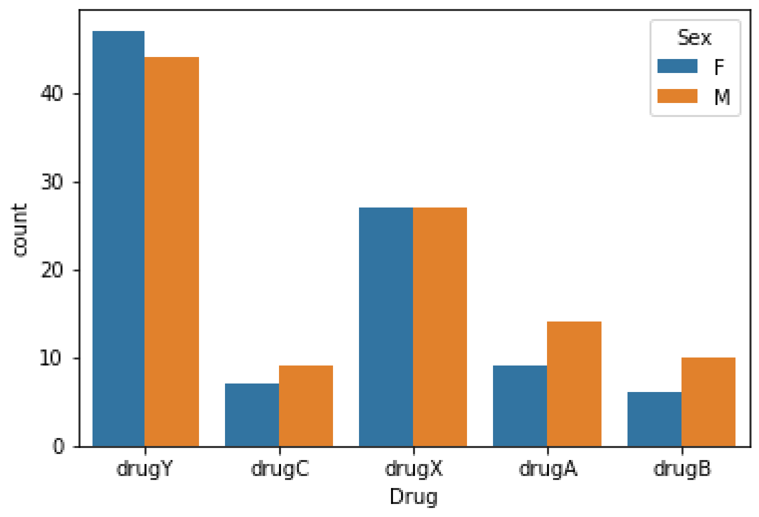
Create a countplot using seaborn showing the counts of each drug type with respect to Sex column as hue column

sns.countplot(x = 'col-name1', hue = 'col-name2', data = dataframeName)
where
col-name1=Drug
col_name2=Sex
dataframename=drugs

In [6]:
```
sns.countplot(x ='Drug', hue = "Sex", data = d)

# Show the plot
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b0d8cdbdd8>
```



Define X input features and y output feature

Drop column Drug and assign to X
Assign the column Drug to y

In [7]:
```
X = d.drop(['Drug'], axis=1)
```

In [8]:
```
y=d['Drug']
```

Apply LabelEncoder to transform the categorical strings with a value between 0 and n_classes-1

In [9]:
```
le_sex = LabelEncoder()
# create LabelEncoder object for BP
le_bp = LabelEncoder()

# create LabelEncoder object for Cholestrol
le_ch = LabelEncoder()

X['Sex'] = le_sex.fit_transform(X['Sex'])
# Transform the values of BP columns using fit_transform()

X['BP'] = le_bp.fit_transform(X['BP'])

# Transform the values of Cholestrol columns using fit_transform()


X['Cholesterol'] = le_ch.fit_transform(X['Cholesterol'])
```

Now print head of X. Observe the values of categorical columns have been changed to numerical values

In [10]:
```
X.head()
```

Out[10]:

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

Split X and y into training and testing sets

In [46]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=100, test_size=0.253)
```

Decision Tree Classifier model with criterion entropy

In [47]:
```
en = DecisionTreeClassifier(criterion ="entropy")
en.fit(X_train, y_train)
```

Out[47]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Predict the Test set results with criterion entropy

In [48]:
```
y_pred = en.predict(X_test)
```

Print the Confusion Matrix

In [49]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[ 3  0  0  0  0]
 [ 0  5  0  0  0]
 [ 0  0  4  0  0]
 [ 0  0  0 11  0]
 [ 0  0  0  0 28]]
```

Print the classification report

In [50]:
```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

      drugA       1.00      1.00      1.00         3
      drugB       1.00      1.00      1.00         5
      drugC       1.00      1.00      1.00         4
      drugX       1.00      1.00      1.00        11
      drugY       1.00      1.00      1.00        28

avg / total       1.00      1.00      1.00        51
```

In [17]:
```
from sklearn.metrics import accuracy_score
print('{0:0.4f}'.format(accuracy_score(y_test, y_pred)))
```

```
0.9091
```