

Assignment 8:

Using the Decision Tree algorithm, predict which drug among drug X, drug Y and drug C should be given to a patient. Find the accuracy of the decision tree in predicting the correct drug for the patient. **Dataset: drug.csv**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv('drug200.csv')

# 1. Display dataset information
# 2. List patients older than 50
# 3. Show top 5 patients with highest Na_to_K value
# 4. Count number of each drug prescribed

# 6. Maximum and Minimum age in the dataset
# 7. Maximum, Minimum, Average Na_to_K Level for patients who received drugX
# 8. Compare age distribution for different drugs
# 9. Plot Sodium-to-Potassium ratio vs Age
# 10. Train a Decision Tree on only Age + Na_to_K and find Accuracy

# 11. Train a Decision Tree and find/ display following:
# a). Evaluate accuracy and print accuracy and Classification Report
# b). Visualize Decision Tree
# c). display Confusion Matrix
```

Chapter 7 > Machine Learning Assignments

```
# 1. Display dataset information  
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
```

```
# 2. List patients older than 50  
df[df['Age'] > 50]
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
4	61	F	LOW	HIGH	18.043	DrugY
8	60	M	NORMAL	HIGH	15.171	DrugY
13	74	F	LOW	HIGH	20.942	DrugY

```
# 3. Show top 5 patients with highest Na_to_K value  
df.nlargest(5, 'Na_to_K')
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
96	58	0	1	0	38.247	DrugY
184	18	0	0	0	37.188	DrugY
98	20	1	0	1	35.639	DrugY
188	65	1	0	1	34.997	DrugY
194	46	0	0	0	34.686	DrugY

```
# 4. Count number of each drug prescribed  
df['Drug'].value_counts()
```

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

```
# 6. Maximum and Minimum age in the dataset
```

```
x=df[ 'Age' ].max()  
y=df[ 'Age' ].min()  
  
print("Maximum age=",x)  
print("Minimum age=",y)
```

```
Maximum age= 74  
Minimum age= 15
```

```
# 7. Maximum, Minimum, Average Na_to_K level for patients who received drugX
```

```
x=df[df[ 'Drug' ] == 'drugX'][ 'Na_to_K' ].max()  
  
y=df[df[ 'Drug' ] == 'drugX'][ 'Na_to_K' ].min()  
  
z=df[df[ 'Drug' ] == 'drugX'][ 'Na_to_K' ].mean()  
  
print(x)  
print(y)  
print(z)
```

```
14.642  
6.683  
10.650555555555558
```

```
# 8. Compare age distribution for different drugs
```

```
df.boxplot(column='Age', by='Drug')  
plt.show()
```

```
# 9. Plot Sodium-to-Potassium ratio vs Age
```

```
plt.scatter(df[ 'Age' ], df[ 'Na_to_K' ])  
plt.xlabel('Age')  
plt.ylabel('Na_to_K')  
plt.show()
```

```
# 10. Train a Decision Tree on only Age + Na_to_K

x = df[['Age', 'Na_to_K']]

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

model.fit(X_train, y_train)

print("Accuracy (Age + Na_to_K only):", accuracy_score(y_test, model.predict(X_test)))

Accuracy (Age + Na_to_K only): 0.6166666666666667
```

Now, Apply Decision Tree algorithm to predict which drug among drug X, drug Y and drug C should be given to a patient. Find the accuracy of the decision tree in predicting the correct drug for the patient. Print Classification Report Visualize Decision Tree and Confusion Matrix

```
# 11. Train a Decision Tree and find/ display following:
# a). Evaluate accuracy and print accuracy and Classification Report
# b). Visualize Decision Tree
# c). display Confusion Matrix

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

# Step 1: Load the dataset
df = pd.read_csv('drug200.csv')

# Step 2: Encode categorical features
sex = LabelEncoder() # Do not take Sex because column name
bp = LabelEncoder() # Do not take BP because column name
cholesterol = LabelEncoder() # Do not take Cholesterol because column name

df['Sex'] = sex.fit_transform(df['Sex']) # Male:1, Female:0
df['BP'] = bp.fit_transform(df['BP']) # High:0, Low:1, Normal:2
df['Cholesterol'] = cholesterol.fit_transform(df['Cholesterol']) # High:0, Normal:1
```

```
# Step 3: Feature selection
x = df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']]
y = df['Drug']

# Step 4: Split data into train-test sample, assume test size=20%
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Step 5: Train Decision Tree
model = DecisionTreeClassifier(criterion='entropy', max_depth=4)
model.fit(x_train, y_train)

# Step 6: Prediction
# predicts the output (target variable) for each sample in x_test using the trained model.
y_pred = model.predict(x_test)

# Step 7: Evaluation metrics
acc = accuracy_score(y_test, y_pred)

# Display Evaluation metric
print("Accuracy:", acc)

print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Step 8: Visualize Decision Tree
plt.figure(figsize=(16,10))

tree.plot_tree(model, feature_names=x.columns, class_names=model.classes_, filled=True)
plt.title("Decision Tree for Drug Prediction")
plt.show()

# Step 9: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu')

# cmap='YlGnBu' means Yl --> Yellow, Gn --> Green, Bu --> Blue

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```