```
import io, os, sys
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
```

```
path = r"C:\Users\Surojit Pan\Downloads\drug.csv"

# Try to read the given path (for local runs). If not found (Colab), ask to upload.
try:
    df = pd.read_csv(path)
except Exception:
    try:
        from google.colab import files
        print("Upload drug.csv from your computer (select the file).")
        uploaded = files.upload()
        fname = next(iter(uploaded))
        df = pd.read_csv(io.BytesIO(uploaded[fname]))
    except Exception:
        print("Could not auto-load file. Please ensure 'drug.csv' is available and re-run.")
        raise
```

```
Upload drug.csv from your computer (select the file).
Choose Files drug.csv
drug.csv(text/csv) - 6027 bytes, last modified: 11/8/2025 - 100% done
Saving drug.csv to drug (1).csv
```

```
target_col = None
for cand in ("Drug", "drug", "DRUG"):
    if cand in df.columns:
        target_col = cand
        break
if target_col is None:
    target_col = df.columns[-1]

X = df.drop(columns=[target_col])
y = df[target_col]
```

```
le_target = LabelEncoder()
y = le_target.fit_transform(y.astype(str))
```

```
X_enc = X.copy()
for c in X_enc.columns:
    if X_enc[c].dtype == 'object' or X_enc[c].dtype.name == 'category':
        X_enc[c] = LabelEncoder().fit_transform(X_enc[c].astype(str))
```

```
X_enc = X_enc.apply(pd.to_numeric, errors='coerce').fillna(0)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_enc, y, test_size=0.25, random_state=42, stratify=y)

# Train Decision Tree
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"Target column used: '{target_col}'")
print(f"Classes (label -> original): {dict(enumerate(le_target.inverse_transform(range(len(le_target.classes_)))))}")
print(f"Accuracy: {acc:.4f}\n")
print("Classification report:")
print(classification_report(y_test, y_pred, target_names=le_target.classes_))
print("Confusion matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Target column used: 'Drug'
Classes (label -> original): {0: 'drugA', 1: 'drugB', 2: 'drugC', 3: 'drugX', 4: 'drugY'}
Accuracy: 0.9800

Classification report:
              precision    recall  f1-score   support
```

```
           drugA       0.86      1.00      0.92         6
           drugB       1.00      0.75      0.86         4
           drugC       1.00      1.00      1.00         4
           drugX       1.00      1.00      1.00        13
           drugY       1.00      1.00      1.00        23

        accuracy                           0.98        50
       macro avg       0.97      0.95      0.96        50
    weighted avg       0.98      0.98      0.98        50

    Confusion matrix:
    [[ 6  0  0  0  0]
     [ 1  3  0  0  0]
     [ 0  0  4  0  0]
     [ 0  0  0 13  0]
     [ 0  0  0  0 23]]
```