```
import pandas as pd
import numpy as np
df=pd.read_csv('/content/drug200.csv')
df.head()
```

	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

```
df.columns
```

```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

```
x=df.drop('Drug', axis=1)
y=df['Drug']
```

```
print(x)
print(y)
```

```
[[23 25.355 False False False False]
 [47 13.093 True True False False]
 [47 10.114 True True False False]
 ...
 [52 9.894 True False True False]
 [23 14.02 True False True True]
 [40 11.349 False True False True]]
['DrugY' 'drugC' 'drugC' 'drugX' 'DrugY' 'drugX' 'DrugY' 'drugC' 'DrugY'
 'DrugY' 'drugC' 'DrugY' 'DrugY' 'DrugY' 'drugX' 'DrugY' 'drugX' 'drugA'
 'drugC' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY'
 'drugX' 'DrugY' 'DrugY' 'drugX' 'drugB' 'drugX' 'DrugY' 'drugX' 'drugX'
 'drugA' 'drugX' 'drugX' 'drugX' 'DrugY' 'drugB' 'DrugY' 'drugX' 'drugX'
 'drugX' 'drugA' 'drugC' 'DrugY' 'DrugY' 'DrugY' 'drugX' 'DrugY' 'DrugY'
 'drugB' 'drugC' 'drugB' 'DrugY' 'drugX' 'DrugY' 'DrugY' 'drugA' 'DrugY'
 'drugX' 'drugB' 'DrugY' 'drugA' 'drugX' 'DrugY' 'DrugY' 'drugB' 'DrugY'
 'drugX' 'DrugY' 'DrugY' 'DrugY' 'drugA' 'DrugY' 'drugA' 'drugX' 'drugB'
 'drugX' 'drugC' 'drugA' 'drugC' 'drugB' 'drugX' 'DrugY' 'DrugY' 'DrugY'
 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'drugX' 'DrugY' 'DrugY' 'DrugY'
 'DrugY' 'drugA' 'drugA' 'drugC' 'drugX' 'DrugY' 'drugX' 'drugX' 'DrugY'
 'drugB' 'DrugY' 'drugA' 'drugX' 'drugX' 'drugX' 'drugX' 'DrugY' 'drugX'
 'drugX' 'drugA' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'drugB' 'DrugY'
 'DrugY' 'drugX' 'DrugY' 'drugX' 'DrugY' 'DrugY' 'drugX' 'DrugY' 'DrugY'
 'drugX' 'drugB' 'drugA' 'drugB' 'drugX' 'drugA' 'DrugY' 'drugB' 'DrugY'
 'drugA' 'drugX' 'drugX' 'drugA' 'drugX' 'drugC' 'drugA' 'drugB' 'drugX'
 'drugX' 'DrugY' 'drugC' 'drugA' 'DrugY' 'drugC' 'drugX' 'drugX' 'drugB'
 'drugX' 'DrugY' 'DrugY' 'DrugY' 'DrugY' 'drugX' 'DrugY' 'drugA' 'drugX'
 'drugX' 'DrugY' 'DrugY' 'drugA' 'DrugY' 'drugA' 'DrugY' 'DrugY' 'DrugY'
 'DrugY' 'drugX' 'drugX' 'DrugY' 'DrugY' 'DrugY' 'drugB' 'drugA' 'DrugY'
 'DrugY' 'DrugY' 'drugA' 'DrugY' 'drugC' 'DrugY' 'drugC' 'drugC' 'drugX'
 'drugX' 'drugX']
```

```
x=np.array(x)
y=np.array(y).reshape(-1, 1)
```

```
df.isnull().sum()
```

	0
Age	0
Sex	0
BP	0
Cholesterol	0
Na_to_K	0
Drug	0

dtype: int64

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(
```

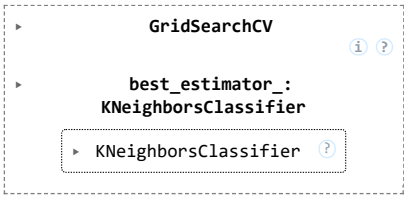
```
    x,y, test_size=0.3, random_state=42
)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'n_neighbors': np.arange(1, 10)}
```

```
grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(x_train, y_train)
```




```
from sklearn.preprocessing import LabelEncoder
x = pd.get_dummies(df.drop('Drug', axis=1), columns=['Sex', 'BP', 'Cholesterol'], drop_first=True)
y = df['Drug']
le = LabelEncoder()
y_encoded = le.fit_transform(y)

x = np.array(x)

print("Original y (first 5):", y.head().values)
print("Encoded y (first 5):", y_encoded[:5])
print("Classes encoded by LabelEncoder:", le.classes_)
```

```
Original y (first 5): ['DrugY' 'drugC' 'drugC' 'drugX' 'DrugY']
Encoded y (first 5): [0 3 3 4 0]
Classes encoded by LabelEncoder: ['DrugY' 'drugA' 'drugB' 'drugC' 'drugX']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x, y_encoded, test_size=0.3, random_state=42
)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
import numpy as np

knn = KNeighborsClassifier()
param_grid = {'n_neighbors': np.arange(1, 10)}

grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(x_train, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_
print(f"Best Parameters: {best_params}")
print(f"Best Score: {best_score}")
```

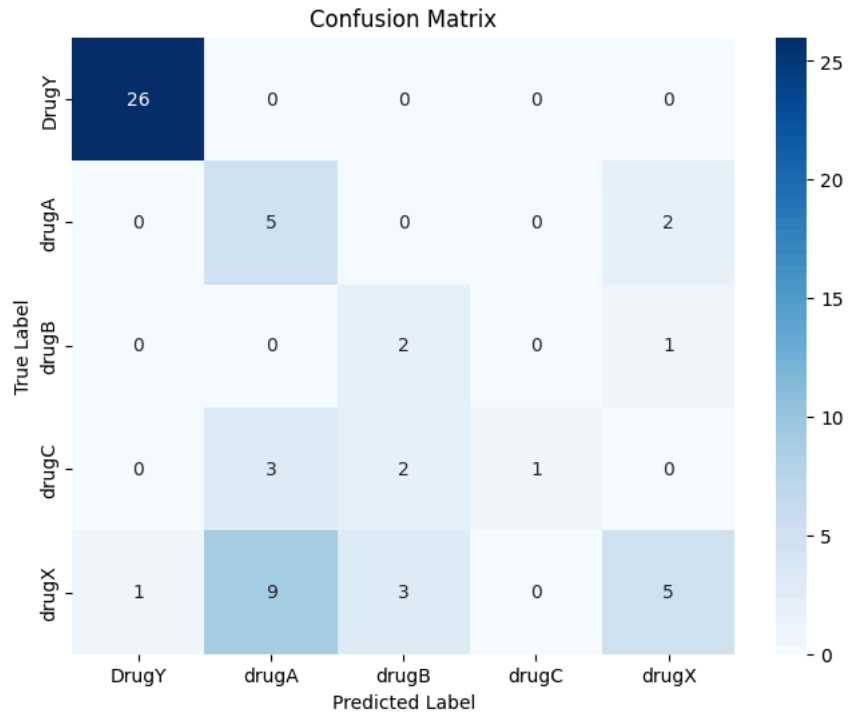
```
Best Parameters: {'n_neighbors': np.int64(3)}
Best Score: 0.65
```

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = grid_search.best_estimator_.predict(x_test)

print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

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

       DrugY       0.96      1.00      0.98        26
       drugA       0.29      0.71      0.42         7
       drugB       0.29      0.67      0.40         3
```

```
           drugC       1.00      0.17      0.29         6
           drugX       0.62      0.28      0.38        18

        accuracy                           0.65        60
       macro avg       0.63      0.57      0.49        60
    weighted avg       0.75      0.65      0.64        60


    Confusion Matrix:
    [[26  0  0  0  0]
     [ 0  5  0  0  2]
     [ 0  0  2  0  1]
     [ 0  3  2  1  0]
     [ 1  9  3  0  5]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```




```
from sklearn.metrics import classification_report

y_pred = grid_search.best_estimator_.predict(x_test)
report = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
overall_f1_score = report['weighted avg']['f1-score']

print(f"Overall F1-score: {overall_f1_score:.2f}")
```

```
Overall F1-score: 0.64
```