Practical 5

Aim: To implement a Machine Learning Classification model using a K Nearest Neighbors Classifier algorithm and enhance the model by K Fold and GridSearchCV cross-validation.

In [66]:

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Loading dataset

In [67]:

```
data = pd.read_csv(r"Practical5.csv")
X = data.iloc[:, [1, 2, 3, 4, 5, 6, 7]].values
y = data.iloc[:, -1].values
```

In [68]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

In [69]:

```
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)
```

In [70]:

```
knn = KNeighborsClassifier(n_neighbors=7)
```

In [71]:

```
knn.fit(X_train, y_train)
```

Out[71]:

```
KNeighborsClassifier(n_neighbors=7)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

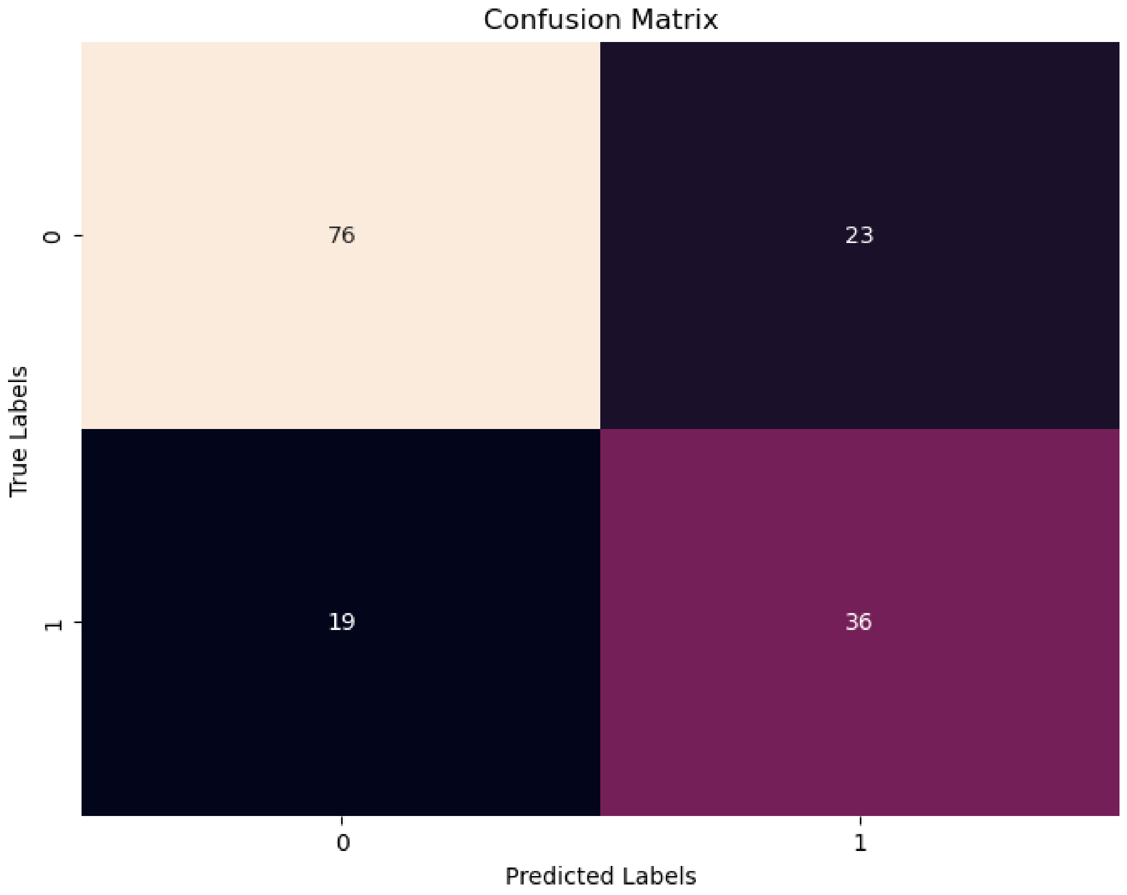
In [72]:
```
y_pred=knn.predict(X_test)
print(knn.predict(X_test))
```

```
[0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1
 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0
 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 1 0 1 0
 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0
 1 1 0 0 0 0]
```

In [73]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[76 23]
 [19 36]]
```

In [74]:
```
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



Type *Markdown* and LaTeX: α^2

In [75]:

```
X_train, X_test, y_train, y_test = train_test_split(
                X, y, test_size = 0.2, random_state=1)
```

In [76]:

```
from sklearn.model_selection import KFold


cv = KFold(n_splits=10)
# perform cross-validation procedure
for train_ix, test_ix in cv.split(X):
    # split data
    X_train, X_test = X[train_ix, :], X[test_ix, :]
    y_train, y_test = y[train_ix], y[test_ix]
    # fit and evaluate a model
    knn = KNeighborsClassifier(n_neighbors=7)
    knn.fit(X_train, y_train)
    y_pred=knn.predict(X_test)
    print(knn.predict(X_test))
    from sklearn.metrics import confusion_matrix
    cm= confusion_matrix(y_test,y_pred)
    print(cm)
    from sklearn.metrics import precision_recall_fscore_support
    precision, recall, f1_score,_ = precision_recall_fscore_support(y_test,
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1_score)
```

```
[0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 0
 0 0 0]
[[35 10]
 [16 16]]
Precision: [0.68627451 0.61538462]
Recall: [0.77777778 0.5       ]
F1 Score: [0.72916667 0.55172414]
[0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0
 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0
 0 1 1]
[[50  5]
 [ 9 13]]
Precision: [0.84745763 0.72222222]
Recall: [0.90909091 0.59090909]
F1 Score: [0.87719298 0.65      ]
[1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0
 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1
 1 0 0]
[[37  6]
 [17 17]]
Precision: [0.68518519 0.73913043]
Recall: [0.86046512 0.5       ]
F1 Score: [0.7628866  0.59649123]
[0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0
 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
 0 1 0]
[[36 11]
 [17 13]]
Precision: [0.67924528 0.54166667]
Recall: [0.76595745 0.43333333]
F1 Score: [0.72       0.48148148]
[0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0
 0 0 0]
[[42  8]
 [12 15]]
Precision: [0.77777778 0.65217391]
Recall: [0.84       0.55555556]
F1 Score: [0.80769231 0.6       ]
[0 1 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0
 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1
 0 0 0]
[[43  4]
 [12 18]]
Precision: [0.78181818 0.81818182]
Recall: [0.91489362 0.6       ]
F1 Score: [0.84313725 0.69230769]
[0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 1
 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0]
[[53 10]
 [ 8  6]]
Precision: [0.86885246 0.375     ]
Recall: [0.84126984 0.42857143]
F1 Score: [0.85483871 0.4       ]
[1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0
 0 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1
 0 1 0]
[[45  7]
 [ 6 19]]
```

```
Precision: [0.88235294 0.73076923]
Recall: [0.86538462 0.76      ]
F1 Score: [0.87378641 0.74509804]
[0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1
 0 1]
[[43  9]
 [10 14]]
Precision: [0.81132075 0.60869565]
Recall: [0.82692308 0.58333333]
F1 Score: [0.81904762 0.59574468]
[0 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1
 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0
 1 0]
[[38  8]
 [12 18]]
Precision: [0.76       0.69230769]
Recall: [0.82608696 0.6       ]
F1 Score: [0.79166667 0.64285714]
```

In [77]:

```
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, precision_recall_fscore_suppo

# Define the KFold cross-validation
cv = KFold(n_splits=9)

param_grid = {'n_neighbors': list(range(1, 51, 2))}



# Initialize the KNN classifier
knn = KNeighborsClassifier()

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=cv, sco

# Perform cross-validation procedure with GridSearchCV
for train_ix, test_ix in cv.split(X):
    # Split data
    X_train, X_test = X[train_ix, :], X[test_ix, :]
    y_train, y_test = y[train_ix], y[test_ix]

    # Fit model using GridSearchCV
    grid_search.fit(X_train, y_train)

    # Get the best KNN model found by GridSearchCV
    best_knn = grid_search.best_estimator_

    # Predict
    y_pred = best_knn.predict(X_test)

    # Evaluate
    cm = confusion_matrix(y_test, y_pred)
    precision, recall, f1_score, _ = precision_recall_fscore_support(y_test

    # Print results
    print("Confusion Matrix:")
    print(cm)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1_score)

    # You can also access the best hyperparameters found
    print("Best parameters found by GridSearchCV:", grid_search.best_params
Confusion Matrix:
[[45  7]
 [16 18]]
Precision: [0.73770492 0.72      ]
Recall: [0.86538462 0.52941176]
F1 Score: [0.79646018 0.61016949]
Best parameters found by GridSearchCV: {'n_neighbors': 13}
Confusion Matrix:
[[49 10]
 [14 13]]
Precision: [0.77777778 0.56521739]
Recall: [0.83050847 0.48148148]
F1 Score: [0.80327869 0.52      ]
Best parameters found by GridSearchCV: {'n_neighbors': 21}
Confusion Matrix:
[[39 10]
 [20 17]]
Precision: [0.66101695 0.62962963]
Recall: [0.79591837 0.45945946]
F1 Score: [0.72222222 0.53125   ]
Best parameters found by GridSearchCV: {'n_neighbors': 11}
Confusion Matrix:
[[38 12]
 [19 16]]
Precision: [0.66666667 0.57142857]
Recall: [0.76       0.45714286]
F1 Score: [0.71028037 0.50793651]
Best parameters found by GridSearchCV: {'n_neighbors': 13}
Confusion Matrix:
[[51  2]
 [15 17]]
Precision: [0.77272727 0.89473684]
Recall: [0.96226415 0.53125   ]
F1 Score: [0.85714286 0.66666667]
Best parameters found by GridSearchCV: {'n_neighbors': 25}
Confusion Matrix:
[[56  9]
 [12  8]]
Precision: [0.82352941 0.47058824]
Recall: [0.86153846 0.4       ]
F1 Score: [0.84210526 0.43243243]
Best parameters found by GridSearchCV: {'n_neighbors': 17}
Confusion Matrix:
[[56  7]
 [ 9 13]]
Precision: [0.86153846 0.65      ]
Recall: [0.88888889 0.59090909]
F1 Score: [0.875      0.61904762]
Best parameters found by GridSearchCV: {'n_neighbors': 13}
Confusion Matrix:
[[48  9]
 [13 15]]
Precision: [0.78688525 0.625     ]
Recall: [0.84210526 0.53571429]
F1 Score: [0.81355932 0.57692308]
Best parameters found by GridSearchCV: {'n_neighbors': 11}
Confusion Matrix:
[[44  8]
 [13 20]]
Precision: [0.77192982 0.71428571]
Recall: [0.84615385 0.60606061]
```

```
F1 Score: [0.80733945 0.6557377 ]
Best parameters found by GridSearchCV: {'n_neighbors': 15}
```