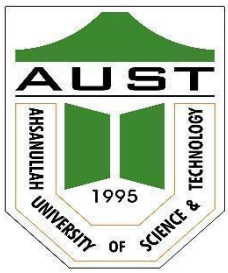


Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE4108**
Course Title : **Artificial Intelligence Lab**
Assignment No : **Term Assignment 3**

Date of Performance : **12.02.2023**
Date of Submission : **23.02.2023**

Submitted To : **Mr. Faisal Muhammad Shah & Mr. Raihan Tanvir**

Submitted By-
Group : **C_2**
Name : **Ashiqul Islam**
Id : **190104140**
Section : **C**

Topic: K Nearest Neighbor Classification and Decision Tree Classification

Source Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Reading the LiverPatient dataset
dataset = pd.read_csv('LiverPatient.csv')
dataset.head(10)

# Cleaning the dataset by dropping duplicate rows and rows containing null values
dataset.drop_duplicates(inplace = True)
dataset.dropna(inplace = True, how = 'any')

dataset.shape

# Splitting the features and target column from the dataset
X = dataset.iloc[:, 0:10]
Y = dataset.iloc[:, -1]

# Encoding the gender feature into numeric values and Scaling the numerical featuers
of the dataset
label_encoder = LabelEncoder()
X.gender = label_encoder.fit_transform(X.gender)

data_scaler = StandardScaler()
X = data_scaler.fit_transform(X)

X = pd.DataFrame(X)

# Creating objects of models and lists for calculating performance metrics
knn = KNeighborsClassifier()
dtree = DecisionTreeClassifier()
kf = KFold(n_splits = 7, shuffle = True, random_state = 10)
accuracy_knn = list()
precision_knn = list()
recall_knn = list()
f1_knn = list()

accuracy_dtree = list()
precision_dtree = list()
recall_dtree = list()
f1_dtree = list()
```

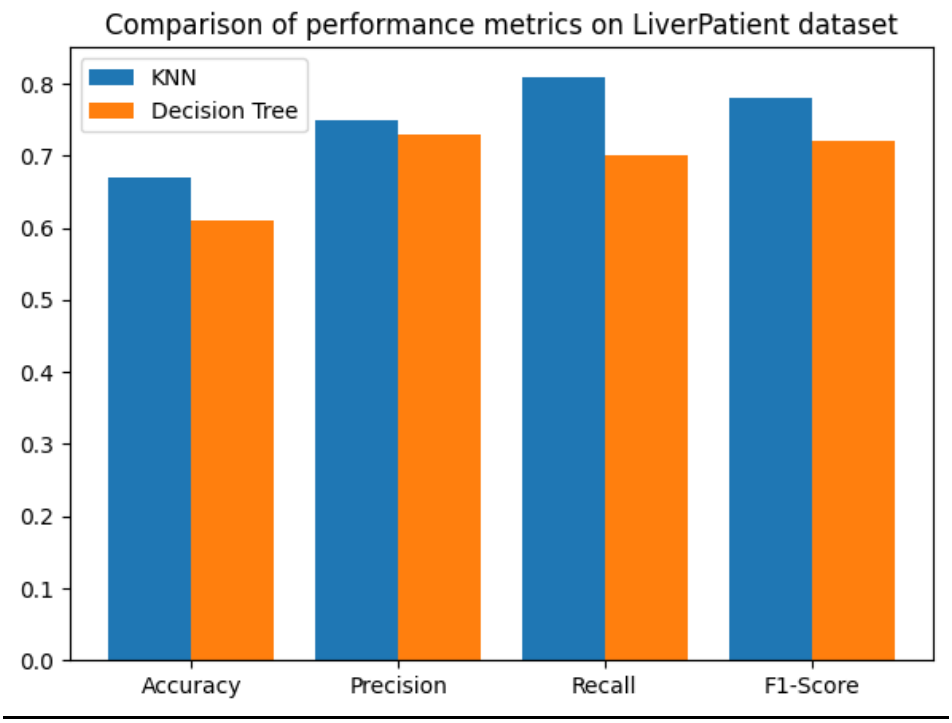
```
# Training models using K-Fold Cross Validation
for train_index, test_index in kf.split(X):
    x_train, x_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = Y.iloc[train_index], Y.iloc[test_index]

    knn.fit(x_train, y_train)
    dtree.fit(x_train, y_train)

    y_pred_knn = knn.predict(x_test)
    y_pred_dtree = dtree.predict(x_test)

    accuracy_knn.append(accuracy_score(y_test, y_pred_knn))
    accuracy_dtree.append(accuracy_score(y_test, y_pred_dtree))

    precision_knn.append(precision_score(y_test, y_pred_knn))
    precision_dtree.append(precision_score(y_test, y_pred_dtree))

    recall_knn.append(recall_score(y_test, y_pred_knn))
    recall_dtree.append(recall_score(y_test, y_pred_dtree))

    f1_knn.append(f1_score(y_test, y_pred_knn))
    f1_dtree.append(f1_score(y_test, y_pred_dtree))

knn_acc = round(np.average(accuracy_knn), 2)
knn_prec = round(np.average(precision_knn), 2)
knn_rec = round(np.average(recall_knn), 2)
knn_f1 = round(np.average(f1_knn), 2)

dtree_acc = round(np.average(accuracy_dtree), 2)
dtree_prec = round(np.average(precision_dtree), 2)
dtree_rec = round(np.average(recall_dtree), 2)
dtree_f1 = round(np.average(f1_dtree), 2)

print("Accuracy of KNN\t\t: ", knn_acc)
print("Precision of KNN\t: ", knn_prec)
print("Recall of KNN\t\t: ", knn_rec)
print("F1 Score of KNN\t\t: ", knn_f1)

print("Accuracy of Decision Tree\t: ", dtree_acc)
print("Precision of Decision Tree\t: ", dtree_prec)
print("Recall of Decision Tree\t\t: ", dtree_rec)
print("F1 Score of Decision Tree\t: ", dtree_f1)

# Bar chart for comparing performance of two models on this dataset

knn_metrics = [knn_acc, knn_prec, knn_rec, knn_f1]
dtree_metrics = [dtree_acc, dtree_prec, dtree_rec, dtree_f1]

metrics_names = ['Accuracy','Precision','Recall','F1-Score']
knn_metrics = [knn_acc, knn_prec, knn_rec, knn_f1]
dtree_metrics = [dtree_acc, dtree_prec, dtree_rec, dtree_f1]
```

```
x_axis = np.arange(len(metrics_names))

plt.figure(figsize = (7, 5))
plt.bar(x_axis - 0.2, knn_metrics, 0.4, label = 'KNN')
plt.bar(x_axis + 0.2, dtree_metrics, 0.4, label = 'Decision Tree')

plt.xticks(x_axis, metrics_names)
plt.title("Comparison of performance metrics on LiverPatient dataset")
plt.legend()
plt.show()
```

Input: LiverPatient.csv

Output:

```
Accuracy of KNN          :  0.67
Precision of KNN         :  0.75
Recall of KNN            :  0.81
F1 Score of KNN          :  0.78
Accuracy of Decision Tree       :  0.61
Precision of Decision Tree      :  0.73
Recall of Decision Tree         :  0.7
F1 Score of Decision Tree       :  0.72
```



Explanation:

The dataset *LiverPatient.csv* is used in this task for classification of patients into binary classes using the *K Nearest Neighbor Classification* algorithm and *Decision Tree Classification* algorithm. At first the dataset is read from the given csv file into a *pandas* DataFrame. Then for pre-processing, at first, unnecessary rows (rows with null values and duplicate rows) are removed. After that the dataset in split into two, feature columns and the target value column for fitting into the model. The non-numerical features are encoded using the *LabelEncoder* of *scikit-learn* and then all of the attributes are scaled using the *StandardScaler* of scikit-learn to ensure that each feature has a mean of 0 and a standard deviation of 1. This helps in ensuring that no feature is dominating the others.

After pre-processing the data, the *KNN* and *Decision* Tree Classification algorithms are trained on the dataset using the *KFold Cross-validation* technique to ensure that the models are trained and tested on different subsets of the data. The performance metrics like **accuracy, precision, recall and F1-score** are calculated for each model on each fold using the metrics module of *scikit-learn* and then averaged to give a final performance score for each model.

Finally, a multiple bar plot is created using the *pyplot* module of *matplotlib* to visually compare the performance of the two models based on the different performance metrics. From the graph it is evident that *KNN* is slightly performing better than the *Decision Tree Classifier* for this dataset. All of the performance metrics of *KNN* is valued higher than the performance metrics of the model using *Decision Tree Classifier*.