LAB # 03

HOME TASK

K-NEAREST NEIGHBOR (KNN) ALGORITHM

OBJECTIVE

Implementing K-Nearest Neighbor (KNN) algorithm to classify the data set.

Question: Using the K-Nearest Neighbors (KNN) algorithm, predict whether a student will pass or fail based on their study hours, attendance percentage, and participation in extra-curricular activities.

Solution:

```
#22F-BSE-138
#Muhammad Saud Hassan
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Sample dataset
data = {
    'Study Hours': [2, 4, 1, 5, 3, 2, 4, 3, 5, 1],
    'Attendance (%)': [60, 70, 50, 90, 80, 55, 75, 65, 85, 45],
    'Extra-Curricular': ['No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No'],
    'Result': ['Fail', 'Pass', 'Fail', 'Pass', 'Pass', 'Fail', 'Pass', 'Fail', 'Pass', 'Fail']
}

# Convert to DataFrame
df = pd.DataFrame(data)


#performing labelEncoding
le = preprocessing.LabelEncoder()
study_Hours_Encoded = le.fit_transform(df["Study Hours"])
attendance_Encoded = le.fit_transform(df["Attendance (%)"])
extra_curricular_Encoded = le.fit_transform(df["Extra-Curricular"])
result_Encoded = le.fit_transform(df["Result"])
```

```python
#Combining all features
features = list(zip(study_Hours_Encoded, attendance_Encoded, extra_curricular_Encoded, result_Encoded))

#Split dataset into training and testing sets
features_train, features_test,label_train,label_test=train_test_split(features, result_Encoded, test_size=0.3, random_state=42)

#Generate a model using K-Neighbors classifier
model = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

#Fit the dataset on classifier
model.fit(features_train, label_train)

#Perform prediction
predicted = model.predict(features_test)

#Print prediction
print("Prediction:", predicted)

#Confusion Matrix
conf_mat = confusion_matrix(label_test, predicted)
print("Confusion Matrix:")
print(conf_mat)

#Accuracy
accuracy = accuracy_score(label_test, predicted)
print("Accuracy:", accuracy)
```

```
Prediction: [1 1 0]
Confusion Matrix:
[[1 0]
 [0 2]]
Accuracy: 1.0
```