Assignment: K-Nearest Neighbors (KNN) Classification on Custom Fruit Dataset

Aim

To implement the K-Nearest Neighbors (KNN) algorithm on a custom fruit dataset for classification, evaluate its accuracy, and visualize the results.

Lab Setup

- Software: Python 3.x (e.g., Google Colab, Jupyter Notebook)
- Required Libraries:
    - pandas
    - scikit-learn
    - matplotlib
    - seaborn
- Installation commands (if needed):

  bash
  pip install pandas scikit-learn matplotlib seaborn
-

Input

- Custom dataset containing fruit attributes:
    - Weight: Numeric values representing weight of fruits.
    - Color Score: Numeric values representing a color property.
    - Label: Fruit category (Apple or Orange).

Expected Output

- Classification accuracy on the test set.
- Confusion matrix of predicted vs actual labels.
- Prediction result for a new input sample.
- Scatter plot showing fruit data points colored by label, and highlighted new sample.

Theory / Algorithm

K-Nearest Neighbors (KNN) is a supervised learning algorithm which classifies new data points based on the majority class of their k nearest neighbors in feature space. The key steps are:

1. Choose a number k

   k representing the number of neighbors to consider.
2. Compute distances between the new data point and all training samples.
3. Find the k

k closest neighbors.

4. Assign the most frequent label among these neighbors to the new point.

KNN is non-parametric and lazy learning, meaning it does not learn an explicit model but uses the stored training points for prediction.

Code

```
python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# Custom fruit dataset
data = {
    'Weight': [150, 170, 140, 130, 160, 180],
    'Color Score': [0.2, 0.1, 0.4, 0.9, 0.8, 0.7],
    'Label': ['Apple', 'Apple', 'Apple', 'Orange', 'Orange', 'Orange']
}
df = pd.DataFrame(data)
# Encode labels as numeric
df['Label'] = df['Label'].map({'Apple': 0, 'Orange': 1})
X = df[['Weight', 'Color Score']].values
y = df['Label'].values
# Split dataset into training and testing sets (67% train, 33% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
# Initialize and train KNN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# Predict test labels
y_pred = knn.predict(X_test)
# Print accuracy and confusion matrix
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
# Predict label for new sample
new_sample = [[155, 0.3]]
predicted_label = knn.predict(new_sample)[0]
label_map = {0: 'Apple', 1: 'Orange'}
print(f"Prediction for new input (Weight=155, Color Score=0.3): {label_map[predicted_label]}")
# Plot data points and new sample
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=df['Label'], palette='deep', s=100)
plt.scatter(new_sample[0][0], new_sample[0][1], color='red', s=200, marker='X', label='New Sample')
plt.xlabel('Weight')
plt.ylabel('Color Score')
plt.title('KNN Classification on Custom Fruit Dataset')
plt.legend()
plt.show()
```

Conclusion / Discussion

The KNN algorithm successfully classified fruits based on weight and color score with high accuracy on the test set. The confusion matrix shows the correctness of predictions. The new input sample's predicted label

demonstrates the algorithm's practical use. Visualization reveals the distribution of classes and highlights the new sample in feature space.