```
import numpy as np
import pandas as pd
from math import sqrt

# Load Dataset
data = pd.read_csv(r"C:\Users\Admin\Downloads\archive\Iris.csv")
print(data.head(5))

# Use all columns except the first (assuming first is an index or ID)
req_data = data.iloc[:, 1:]
print(req_data.head(5))

# Shuffle Data
shuffle_index = np.random.permutation(req_data.shape[0])
req_data = req_data.iloc[shuffle_index]
print(req_data.head(5))
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0            5.1           3.5            1.4           0.2  Iris-setosa
1            4.9           3.0            1.4           0.2  Iris-setosa
2            4.7           3.2            1.3           0.2  Iris-setosa
3            4.6           3.1            1.5           0.2  Iris-setosa
4            5.0           3.6            1.4           0.2  Iris-setosa
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm          Specie
145            6.7           3.0            5.2           2.3   Iris-virginic
105            7.6           3.0            6.6           2.1   Iris-virginic
129            7.2           3.0            5.8           1.6   Iris-virginic
75             6.6           3.0            4.4           1.4  Iris-versicolo
107            7.3           2.9            6.3           1.8   Iris-virginic
```

```
# Train-Test Split (70%-30%)
train_size = int(req_data.shape[0] * 0.7)
train_df = req_data.iloc[:train_size, :]
test_df = req_data.iloc[train_size:, :]

print('Train Shape:', train_df.shape)
print('Test Shape:', test_df.shape)

train = train_df.values
test = test_df.values
y_true = test[:, -1]  # last column is target
```

```
Train Shape: (105, 5)
Test Shape: (45, 5)
```

```
# Step 1: Euclidean Distance
def euclidean_distance(x_test, x_train):
    distance = 0
    for i in range(len(x_test) - 1):  # exclude label
        distance += (x_test[i] - x_train[i]) ** 2
    return sqrt(distance)
```

```
# Step 2: Get Neighbors
def get_neighbors(x_test, x_train, num_neighbors):
    distances = []
    for i in x_train:
        dist = euclidean_distance(x_test, i)
        distances.append((i, dist))

    # Sort by distance
    distances.sort(key=lambda x: x[1])

    # Select the closest neighbors
    neighbors = [distances[i][0] for i in range(num_neighbors)]
    return neighbors
```

```
# Step 3: Predict for one sample
def prediction(x_test, x_train, num_neighbors):
    neighbors = get_neighbors(x_test, x_train, num_neighbors)
    classes = [i[-1] for i in neighbors]  # get labels of neighbors
    predicted = max(classes, key=classes.count)  # majority vote
    return predicted
```

```
# Step 4: Accuracy Function
def accuracy(y_true, y_pred):
    num_correct = 0
    for i in range(len(y_true)):
        if y_true[i] == y_pred[i]:
            num_correct += 1
    return num_correct / len(y_true)
```

```
# Step 5: Predict for all test samples
y_pred = []
for i in test:
    y_pred.append(prediction(i, train, 5))  # using k = 5
```

```
# Step 6: Evaluate Accuracy
acc = accuracy(y_true, y_pred)
print("Predicted Labels:", y_pred[:5])
print("Accuracy:", acc)
```

```
Predicted Labels: ['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris
Accuracy: 0.9555555555555556
```