```
from math import sqrt


def euclidean_distance(row1, row2):
  distance = 0.0
  for i in range(len(row1)-1):
    distance += (row1[i] - row2[i])**2
  print(sqrt(distance))
  return sqrt(distance)


def get_neighbors(train, test_row, num_neighbors):
  distances = list()
  for train_row in train:
    dist = euclidean_distance(test_row, train_row)
    distances.append((train_row, dist))
  distances.sort(key=lambda tup: tup[1])
  neighbors = list()
  for i in range(num_neighbors):
    neighbors.append(distances[i][0])
  print(neighbors)
  return neighbors


def predict_classification(train, test_row, num_neighbors):
  neighbors = get_neighbors(train, test_row, num_neighbors)
  output_values = [row[-1] for row in neighbors]
  prediction = max(set(output_values), key=output_values.count)
  return prediction


dataset = [[1,2,3,2,1,3,0],
[2,1,3,3,1,2,0],
[1,1,2,3,2,2,0],
[2,2,3,3,2,1,0],
[6,5,7,5,6,7,1],
[5,6,6,6,5,7,1],
[5,6,7,5,7,6,1],
[7,6,7,6,5,6,1],
[7,6,5,5,6,7,1]]
```

```
prediction = predict_classification(dataset, dataset[-1], 3)
```

```
10.295630140987
10.392304845413264
10.723805294763608
10.04987562112089
2.449489742783178
2.6457513110645907
3.1622776601683795
2.6457513110645907
0.0
[[7, 6, 5, 5, 6, 7, 1], [6, 5, 7, 5, 6, 7, 1], [5, 6, 6, 6, 5, 7, 1]]
```

```
print('Expected %d, Got %d.' % (dataset[-1][-1], prediction))
```

```
Expected 1, Got 1.
```

✓ 0s completed at 2:12 PM ● ×