Experiment 10

AA Experiment

DATE:

Kreena Shah
60004210243
C'32

Aim : To understand & implement KD tree

Theory :
A KD-tree (K dimensional tree) is a space partitioning data structure that recursively subdivides a multidimensional space into regions associated with specific data points.
The primary objective of KD tree is to facilitate efficient multidimensional search operations, particularly nearest-neighbor searches

Structure :
(1) Node
(2) Splitting Dimension
(3) Splitting value
(4) Child Nodes

Querying in KD tree
(1) Nearest Neighbour Search
(2) Range Search
(3) Spatial Indexing

Code :

```
from sklearn.neighbors import KDTree
import numpy as np

data = np.array([[2, 3], [5, 4], [9, 6], [4, 7], [8, 1], [7, 2]])

kdtree = KDTree(data, leaf_size=30)

query_point = np.array([[9, 2]])
distances, indices = kdtree.query(query_point, k=2)

print("Query Point:", query_point)
print("Nearest Neighbors:")
for i, idx in enumerate(indices[0]):
```

```python
    print(f"Neighbor {i + 1}: {data[idx]}, Distance: {distances[0][i]}")
```

Output :

```
PS C:\Users\Admin\OneDrive\Desktop\DJ\SEM6_Pracs\AA> py .\kd.py
Query Point: [[9 2]]
Nearest Neighbors:
Neighbor 1: [8 1], Distance: 1.4142135623730951
Neighbor 2: [7 2], Distance: 2.0
```

Conclusion :

During my KD tree experiment, I faced challenges in understanding the recursive approach of construction & balancing the tree.

Debugging & Validating against small datasets helped me overcome these

Implementing efficient nearest neighbour search was a challenging task.

By studying the algorithm & optimizing travel strategies, such as pruning branches, I improved performance.

Optimizing range queries in high dimensional space was also challenging.

Techniques like using bounding boxes & tuning parameters helped streamline process

Through iterative refinement, I deeped my understanding of KD trees.