Introduction to Data Science

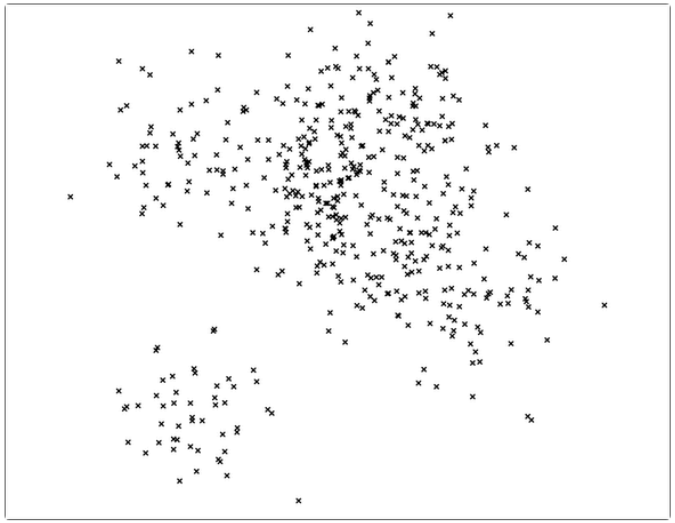
APPROXIMATE

K-NEAREST NEIGHBORS

BRIAN D'ALESSANDRO

APPROXIMATE NEAREST NEIGHBORS

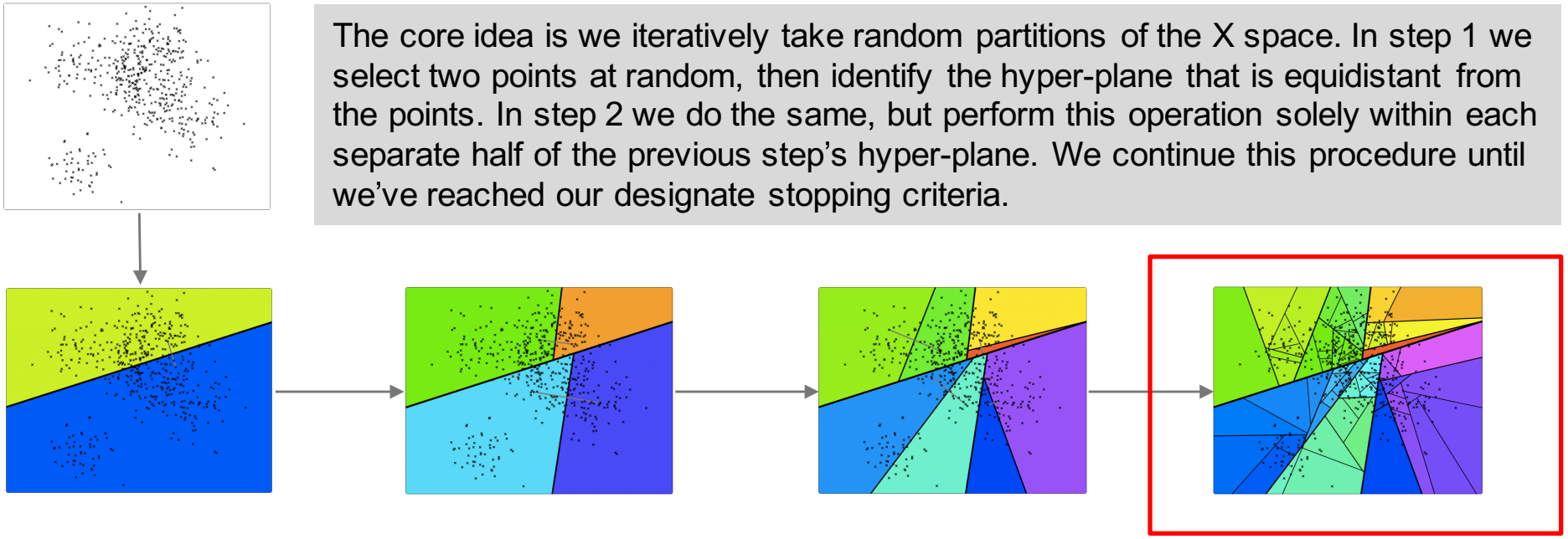
As mentioned before, searching for a nearest neighbor takes *O(n)* time, which can be prohibitive for large *n.* If we are willing to make some tradeoffs, we can design a nearest neighbor search in *O(log(n))* time.



Instead of searching through all points, we will partition the *X* space to create neighborhoods, and search only within those neighborhoods.

CREATING THE NEIGHBORHOODS

There are multiple approaches, so we'll use the *ANNOY* * package as an example

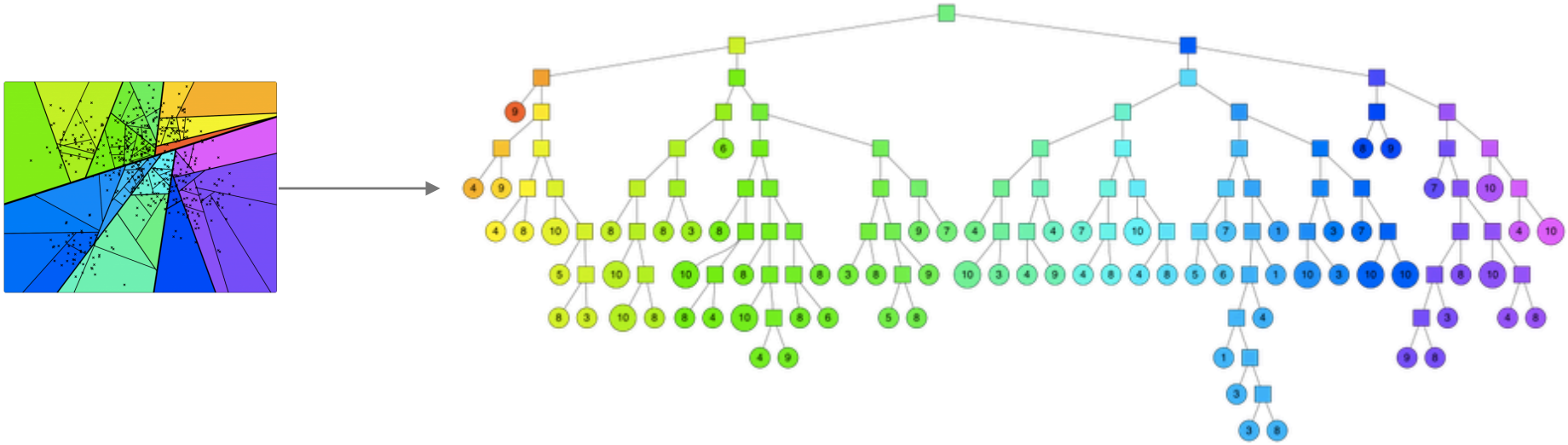


The core idea is we iteratively take random partitions of the X space. In step 1 we select two points at random, then identify the hyper-plane that is equidistant from the points. In step 2 we do the same, but perform this operation solely within each separate half of the previous step's hyper-plane. We continue this procedure until we've reached our designate stopping criteria.

Source:https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

SEARCHING FOR NEIGHBORS

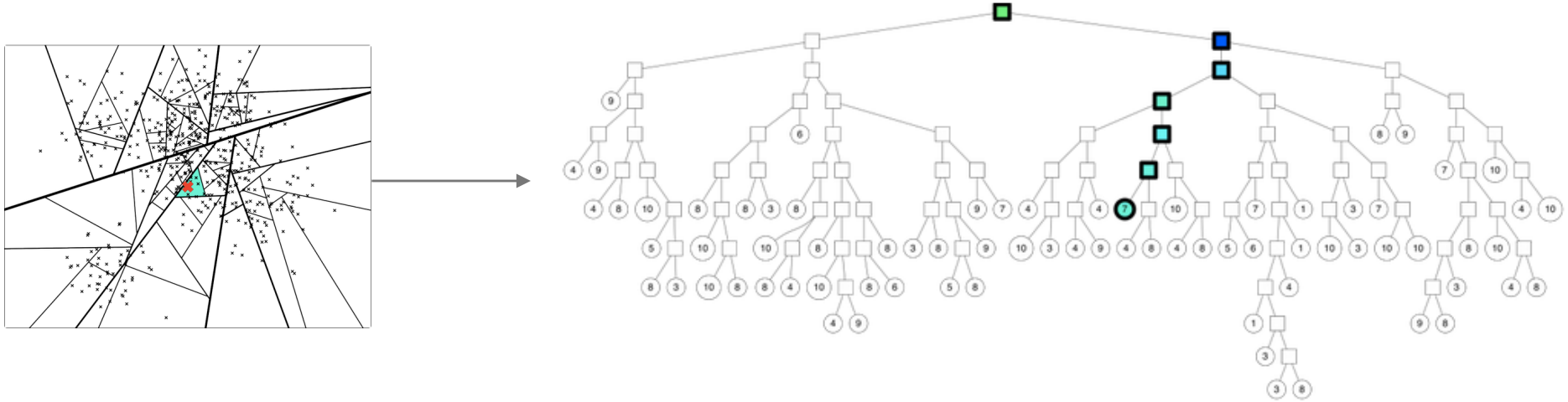
After K recursive splits, our partitioned data can be represented with a binary tree data structure. Finding a set of neighbors amounts to taking an input *x'*, traversing the tree to see which node it lands in, and taking the neighbors from that node.



Source:https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

SEARCHING FOR NEIGHBORS

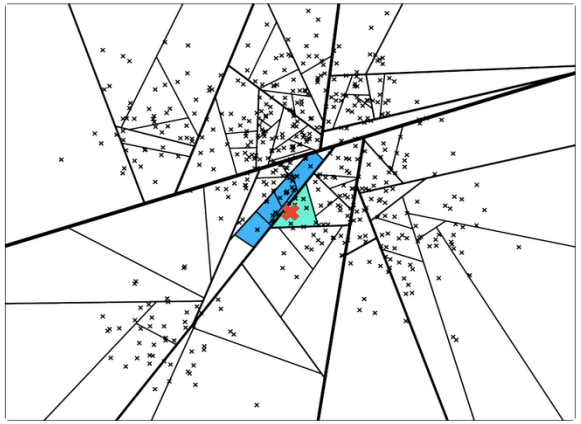
After K recursive splits, our partitioned data can be represented with a binary tree data structure. Finding a set of neighbors amounts to taking an input *x'*, traversing the tree to see which node it lands in, and taking the neighbors from that node.

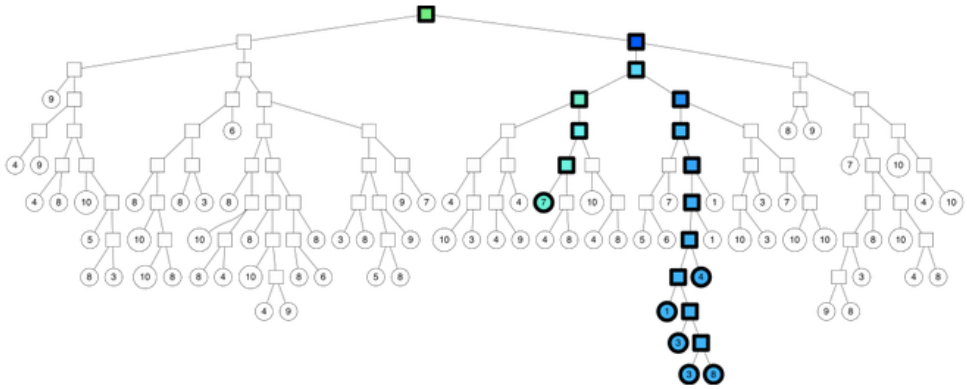


Source:https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

THE TRADEOFFS

Problem: You are limited to just the neighbors in the terminal node of the tree



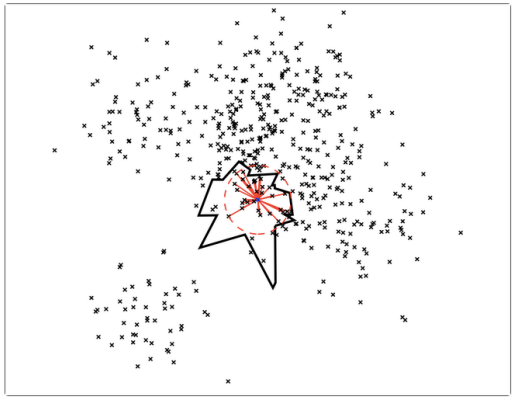


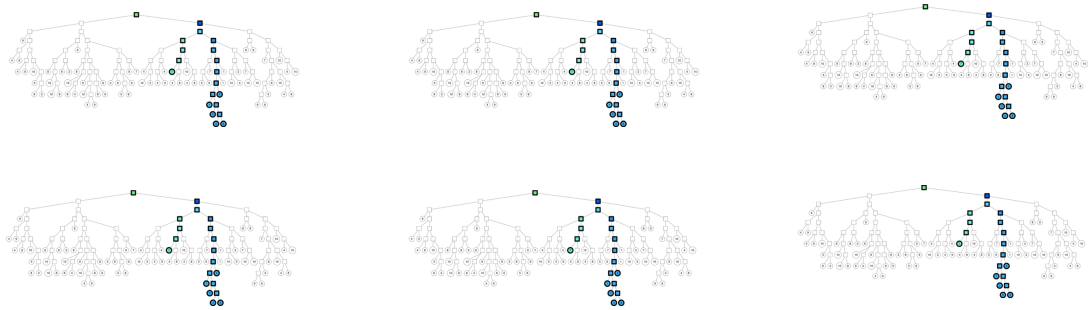
Solution: You don't have to use a single terminal node. Nearby splits (regions) can also be considered)

Source:https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

THE TRADEOFFS

Problem: You may miss true nearest neighbors that are on the boundaries of the partitions





Solution: You can build a forest using multiple trees. Run the full algorithm D times to get D separate neighborhoods for a point. When scoring, the candidate set becomes the union of all D neighborhoods.

Source:https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html