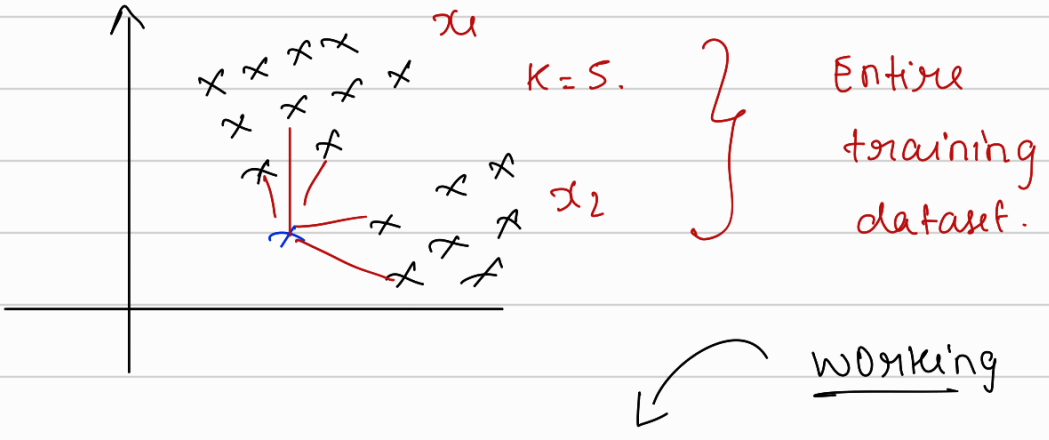
K- Nearest Neighbor

K-Nearest Neighbor is a simple, non-parametric machine learning algorithm used for classification and regression task. It works by finding the k closest training data points to a new unseen data points and making predictions based on their values

It is a supervised machine learning method. KNN does not require an explicit training phase where a model is learned. The algorithm simply stores the entire training dataset.

classification,

Let x_1, x_2 be the features and y be Binary output.



working

1. We have to initialize the 'K' values
   $K > 0, 1 \ldots \infty$ (its hyperparameter)

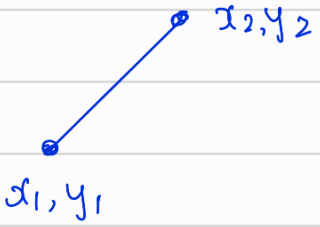
2. Find the K-Nearest neighbour for test data.

3. from those K=5, how many neighbors belong to 0 category and 1 category

To find the distance between test data and K-Nearest neighbor.

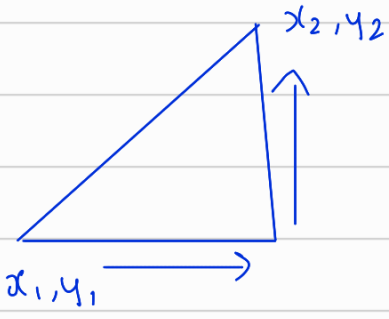
1. Euclidean distance.
2. Manhatton distance.

Euclidean distance :



$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Manhatton distance :



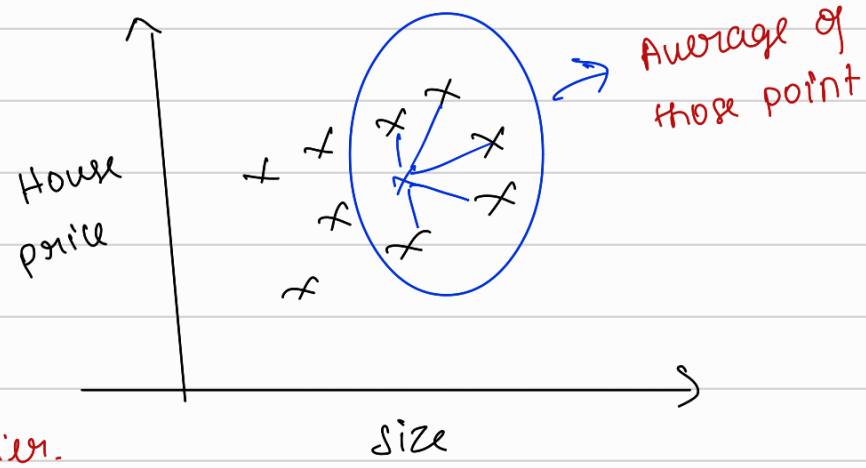
$$\text{distance} = |x_1 - x_2| + |y_1 - y_2|$$

Regression,

The algorithm predicts a continuous value based on the K-Nearest neighbor of a given input point.

The predicted value for the test point is the average of the K-nearest neighbors target values.

$$y' = \frac{1}{K} \sum_{i=1}^{K} y_i$$



find out median if there is a outlier.

Major issue is time complexity, finding the distance between test data and every point in the dataset and it takes O(n)

Now, we use

① KD tree
② Ball tree

} to optimize the KNN

These trees are created in form of binary tree.

Variants of KNN

① KD-tree, is a space partitioning data structure used to organize points in k-dimensional space, it is particularly useful for efficient nearest neighbors searches and making it a popular choice for implementing KNN

* Binary tree structure, Each node in the tree represent a point in the dataset and splits the space into two half spaces

* The dataset is recursively divided along one dimension at each level, alternating to built the tree.

* Efficient search, Allows for faster range searches and nearest neighbor queries compared to brute force approach.

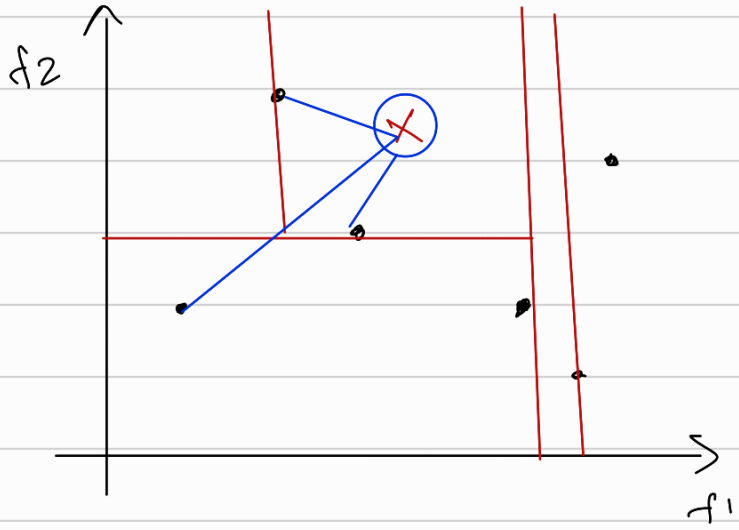
Working

* Choose a dimension to split the data, (x,y,z for 3D)
* Select a median point along the choosen dimension and ensures a balanced tree.

* Recursively partition the data into two subsets.
  (i) Left subtree, contains points less than the median on the choosen dimension.
  (ii) Right subtree, contains points greater than median.

Example, (2D)

1. Split by x-coordinate
2. Split by y-coordinate.
3. Split by x-coordinate again and so, on



→ form a Binary tree
(In short making groups)

Even backtracking is used to find the next nearest neighbor.

(easy to find the nearest point)

Ball tree,

Unlike KD tree, Ball trees partition the space into hyperspherical regions (balls) and making them more suitable for certain type of data distribution & distance metrics.

* Hierarchical partitioning, the space is divided into nested hyperspheres, each node representing a ball (region containing data points)

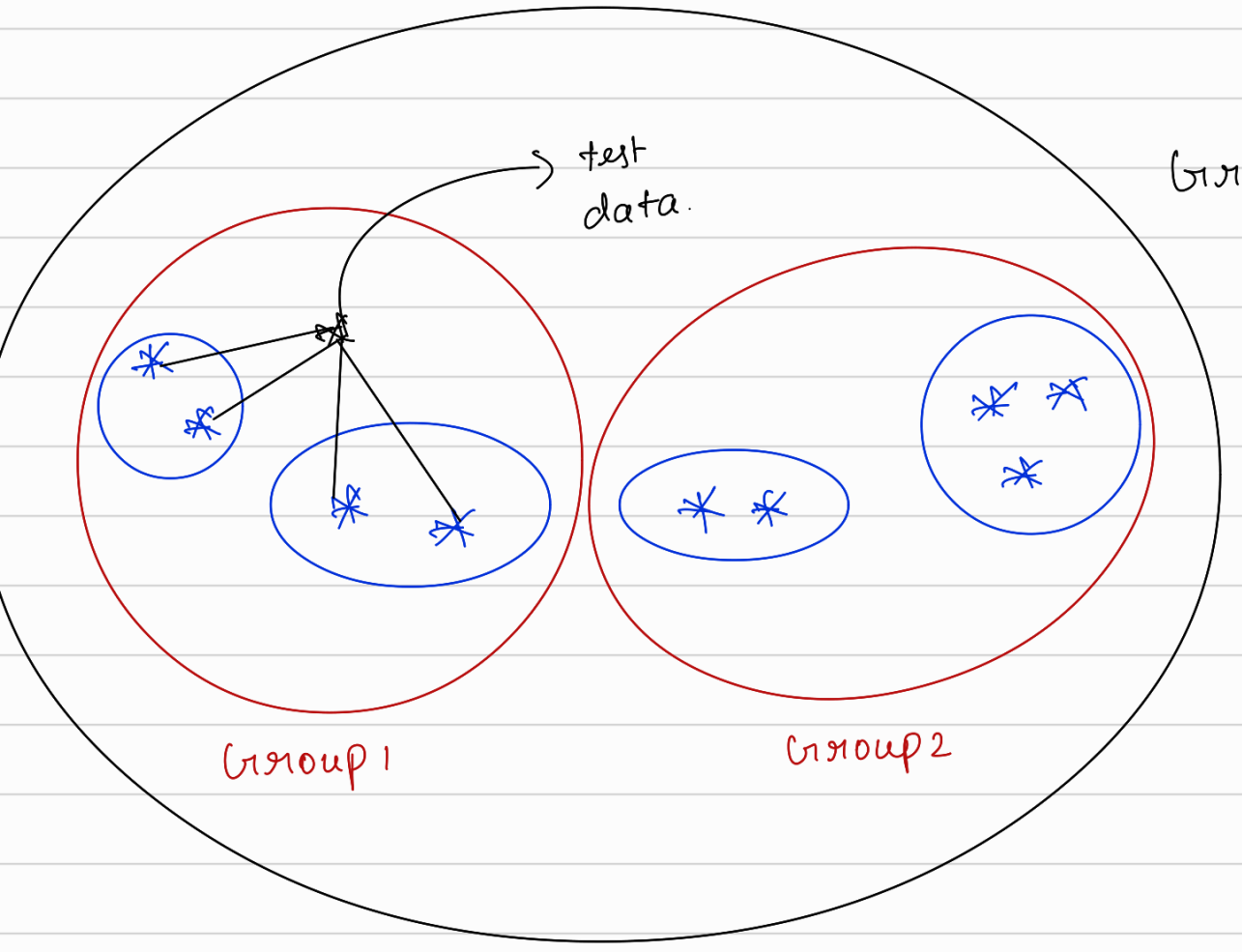
* Adaptability, works well with a variety of distance metrics.

* Efficient in Searching

Working,

* Select a subset of points to form a node.
* compute a centroid for the points in the current node.
* Calculate the radius from the centre to any point
* Split the points into two subsets based on proximity to two choosen pivot points.

* Repeat the process Recursively.


test
data.
Group.
Group 1
Group 2