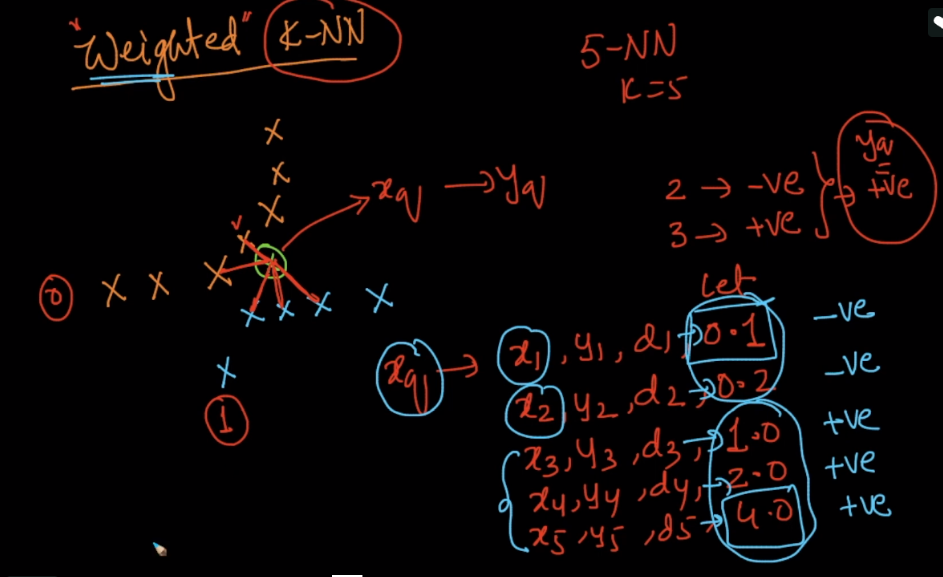
**Weighted k-NN:**

Intution behind k-NN is that, for k-NN we do majority voting to choose class, but isn’t it’s unfair to give the majority class point’s class label to query point instead of considering class of points which are most nearer to query point.

As given in below example our query point is placed such that there are 2 orange points and 3 blue points are near for 5-NN, And after majority vote we assign blue class to query point.

But as we can see orange points are more close to query point, even though they are neglected.



So for such we use weighted k-NN where we do following things

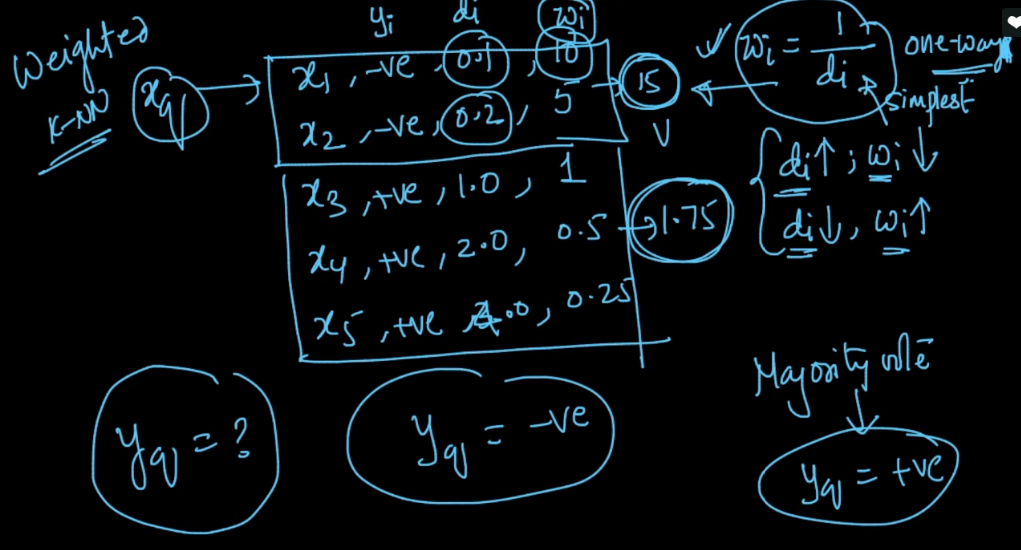
* assign wight to each point after picking k points, according their distances from query point.

There are several ways to find weight one simplest way is 1/distance.

So if distance decreases between points, the weight increases.

And if distance increases between points, the weight decreases.

* Sum weights of each class, as in below example we get sum of weight 15 for orange, and sum as 1.75 for blue points.
* And since weight of orange point is more (even thought the count of orange is less than blue), we assign orange class to query point.



Note:

* Weighting just one of the options we can use with k-NN. In pratie, we try multiple models with various options and pick the one that gives the best performance on the test data.  
    
  As for the example you gave, the +ve point with a large weight implies it is very close to the query points while the rest of the negative points are far away and hence have a smaller cumulative weight. So, it makes sense to use the closer point's class label as other points as far away from our query-point. But, at the end of the day, ML algos don't work for all cases and all query-points and hence will make mistakes. So, we use the CV and test-dataset to pick the best model amongst the bunch of possible ones.
* what if the outlier is closest to the query point and has the highest weight ,it would dominate the other points in that region,and hence will be classified wrong?

KNN is highly affected by outliers and scale. Hence it is recommended to handle outliers and scale before the KNN.

* The idea is to be able to be more robust against variations in distances of the k-nearest neighbors which may lead to wrong decisions. It leads to smoother decision surfaces.

The assumption is that neighbors closets to the sample should be given more relevance when deciding by voting to which class the sample belongs, since they are more similar.

This is specially relevant for samples close to the decision surfaces, which are sensible to effects like noise or sampling differences among classes

* 1) How do we use weighted k-NN for multiclass classification?  
  2) Do we use weighted k-NN for Regression also? If yes how do we do that

1) In case of Multi-class classification, let us assume we have 4 classes in our dataset A,B,C,D. If we are  working on 7-NN (say), then compute the reciprocal of distances of the points from the query point class wise (ie., 1/d1,1/d2,1/d3,1/d4) which ever class gives least weights,the query point is classified as that class point.  
2) Weighted KNN isn't applicable for Regression. It is only applicable for Classification.