

# Machine Learning

## K-NN

Machine Learning I (CC2008) - Assignment No. 1

Daniel Dias up202105076

Ivo Simões up202107703

Lucas Viana up202104660

Maio 2023

## Executive summary



Our main goal in this assignment was to expand our knowledge on a given algorithm (k-NN) and be able to modify it, such that it yields better results under different circumstances.

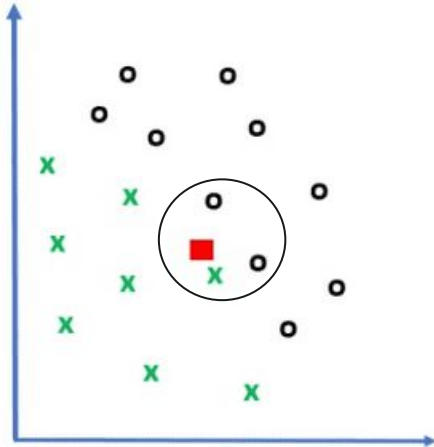
To tackle our proposal, we tried different approaches, ranging from applying weights to distances (both through point proximity and attribute importance) to normalizing classes.

In general, our modifications outperformed the original k-NN, even surpassing our expectations (with a few coming out on top). However, in certain cases, our modifications got beaten by the naive version.

## K-Nearest Neighbors (description)

This algorithm is a non-parametric machine learning algorithm used for regression and classification tasks.

It basically classifies a point based on the  $k$  nearest points to it. The label is attributed based on the most observed class between those  $k$  points. If a draw occurs, the result is decided randomly.



- Assuming  $K=3$ ;
- Two neighbors are **o** and one is **x**;
- The target is classified as **o**.

## K-Nearest Neighbors (advantages and disadvantages)

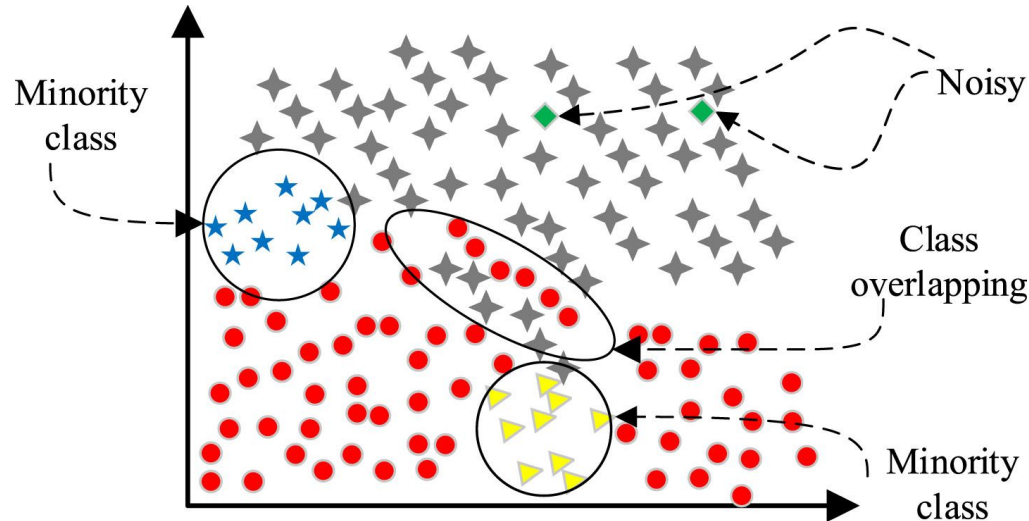


k-NN is simple to implement, works with a lot of distance metrics (euclidean, manhattan) and is extremely fast to train, since the data itself is the model that is used in reference for future predictions.


But, as with every other algorithm, it has its flaws. For this assignment, we will focus more on the k-NN's multiclass classification problem, while addressing other problems like class overlap, class imbalance and high dimensionality. We'll even test what happens if we combine the solutions we find for each problem.

## Multiclass classification and k-NN

If the closest neighbors belong to multiple different classes, k-NN is more susceptible to make inaccurate predictions. Class overlap, class imbalance and high dimensionality aggravate this problem since they boost the ambiguity of the guess, increase the algorithm's bias towards majorities and expand the volume of feature space, respectively.



## Proposal - Motivation



The core part of our proposal revolves around multiclass classification problems, but we found out that this type of data usually comes with overlapping, imbalance or high dimensionality.

We figured that, to solve a multiclass classification problem, we would need to find solutions for class overlap, class imbalance and high dimensionality too.

To do that, we came up with three different ways of solving the main problem, each focusing on one or more of each additional data characteristic.

## Proposal - Description

The first is a classic distance weighting technique where we simply calculate the inverse of the distance to value the closest training points over the more distant ones. This way, we can mitigate the class imbalance as well as class overlap.

The second is attribute importance. If we apply attribute-target correlation to the distance, we could maybe fix the high dimensionality problem. A way to do this would be apply the results found by a random forest regression directly to the distances of each attribute.

The last is normalization. Let's imagine a scenario where we use  $k=50$ , have 10 neighbours of the class A and 40 of the class B. By normalizing the results to the average number of elements per class we try to counter the bias that the algorithm could have towards classes with a greater number of neighbours (fig 1).

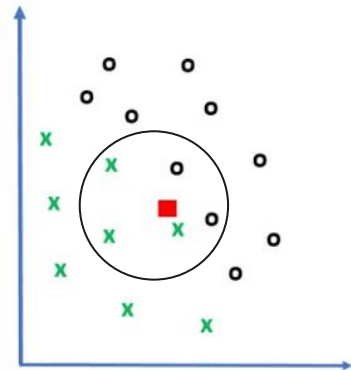
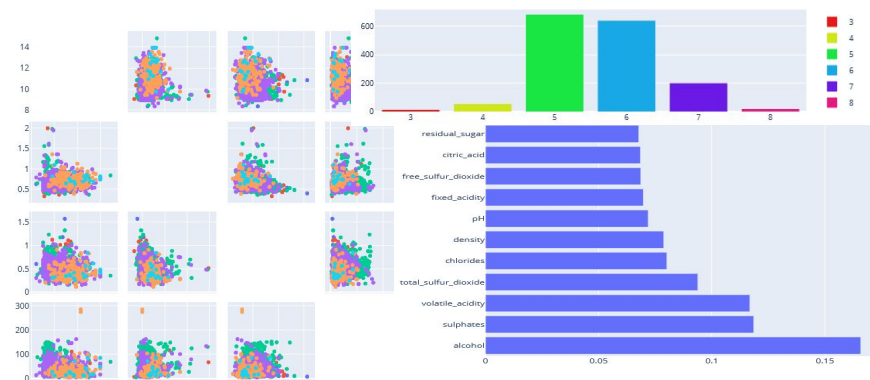


fig 1

## Experimental setup - Datasets and their characteristics

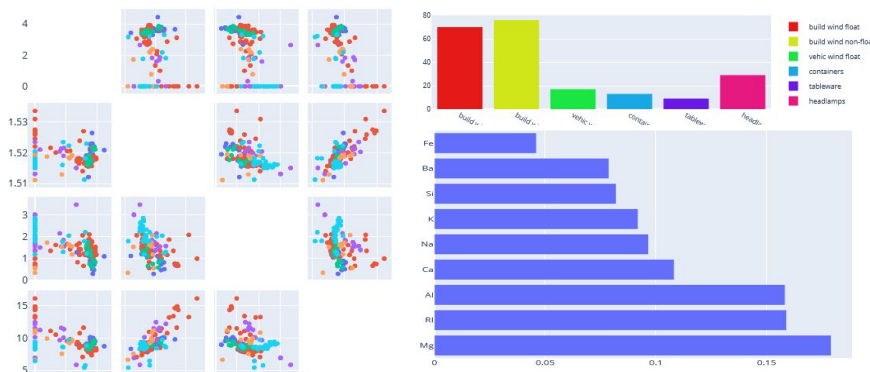
### Red Wine Quality (ID 40691):

- 12 attributes and 6 classes;
- Dense data;
- Class overlap (3-4 classes);
- Quite imbalanced;
- Well distributed importance.



### Glass (ID 41):

- 10 attributes and 6 classes;
- Rather dense data;
- Class overlap (2-3 classes);
- Noticeable imbalance;
- Slight difference in attribute importance.

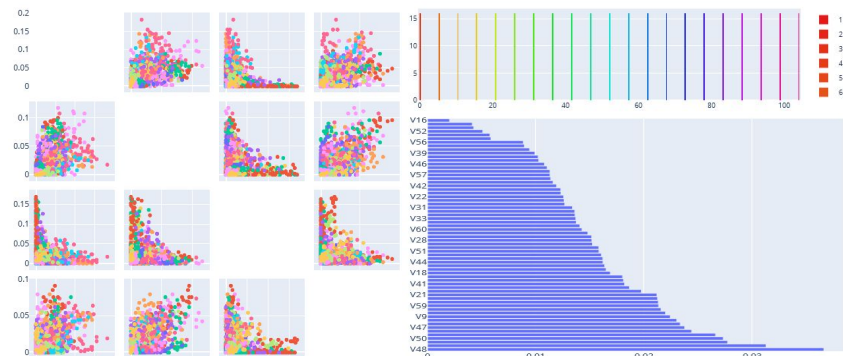




## Experimental setup - Datasets and their characteristics

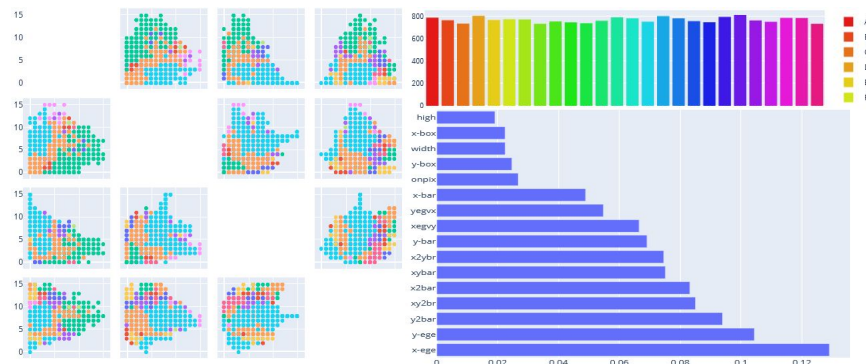
### 100 Plants Margin (ID 1491):

- 65 attributes and 100 classes;
- Dense data (with some outliers);
- Heavy class overlap (10+ classes);
- Perfectly balanced;
- Varied attribute importance.



### Letter (ID 6):

- 17 attributes and 26 classes;
- Moderate data density;
- Minimal overlap;
- Great feature balance;
- Significant attribute variance.



## Experimental setup



### Data characteristic of interest:

Although we picked datasets with varying class overlap, class imbalance and dimensionality, the main focus is the vast number of classes and high density.

### Hyperparameters of the algorithm:

The main hyperparameters of the k-NN algorithm include the number of neighbors (k) and the distance metric used for similarity calculation. Other considerations include data preprocessing, algorithm speed optimization, and handling of ties in class assignment.

Note: for data preprocessing we used the Standard Scaler and Label Encoder

## Experimental setup - performance estimation methodology

In order to study the performance of the modified algorithms we did a “simple” loop:

*#Tests for each dataset  
for dataset in data:*

*# Test knn models across varying ks  
for k in range 100:*

*#Naive knn*

*#knn Weighted (1/dist)*

*#knn Weighted (1/dist) Normalized*

*#knn Weighted (1/dist) Attributes*

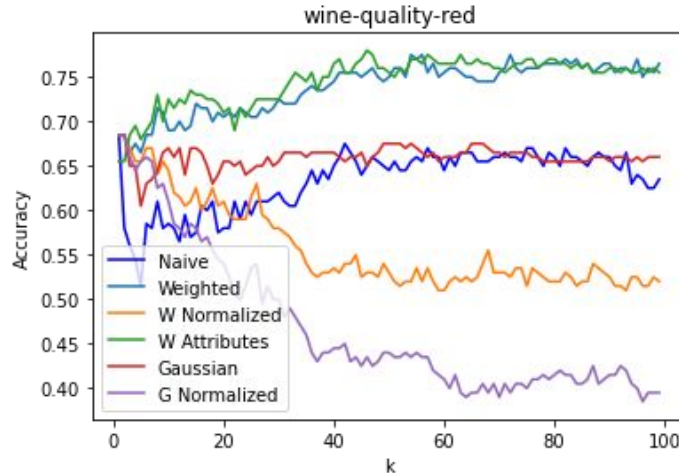
*#knn Weighted (gaussian)*

*#knn Weighted (gaussian) Normalized*

*#Plot the acc for each k*

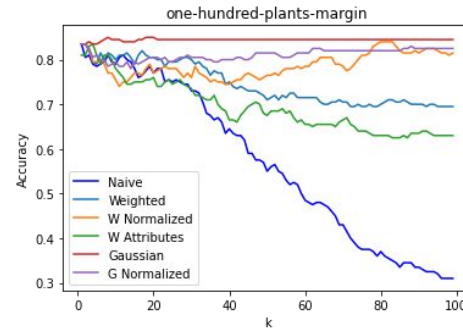
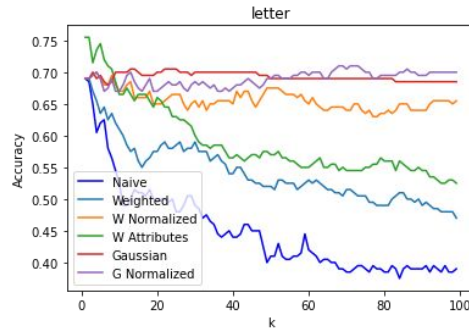
*#Print the avg accuracies*

For each one of the 4 datasets we had, we made the  $k$  vary from 1 to 100 and ran the algorithms for each value of  $k$ . The outcome were graphics alike the following one:



## Experimental results

As seen on the table/graphs all the “new algorithms” outperformed the naive knn. Overall, predicting the results by using the sum of the weighted/normalized/attributes distances, demonstrated to be better than the original “by vote” method.



	Naive	Weighted	W Normalized	W Attributes	Gaussian	G Normalized
wine-quality-red	4	2	5	1	3	6
glass	6	2	5	3	1	4
one-hundred-plants-margin	6	4	3	5	1	2
letter	6	5	3	4	1	2
Avg Position	5.500	3.250	4	3.250	1.500	3.500

## Conclusions and future work



This assignment made us realize that by only modifying what seems to be a small parameter (e.g. giving weights to the distances), the algorithm yielded better results on a certain type of datasets.

That makes us believe that by adapting any algorithm to the dataset in study can make a considerable difference on the outcome. The skills we learnt from this assignment will definitely be important and used on future works.