Umeå University
Department of Computing Science

Fundamentals of artificial intelligence
5DV121

K nearest neighbours

Thomas Ranvier *Valentin Lecompte*

supervised by
Thomas Johannson
Carl-Anton Anserud
Anders Broberg
Adam Dahlgren Lindström
Ola Ringdahl

September 27, 2018

Chapter 1

Context

The K nearest neighbours is one of the simplest machine learning algorithms, it is a supervised learning algorithm since we use known datas to create a training set.

It works in a very simple way, example with a number of neighbours k:

- First, a training set is built from known datas.
- When we want to classify a test data the algorithm will compare that data to all the datas in the training set. For each of them it will compute the distance between them.
- Then the algorithm looks at the k closest neighbours and return the category that comes the most in those selected neighbours.

Chapter 2

Experimentation

2.1 Determine important features

The first thing we wanted to do was to determine which features are the more important for each dataset. To do so we decided to use a 'forest of tree' algorithm. This algorithm is available to use in the scikit library. To know the importance of each features on the datas we just have to give the dataset to the forest of trees, train it, and the importance of each feature is available in the $feature\_importances\_$ array.

Bellow are the bar plots of the importance of every features for both datasets.

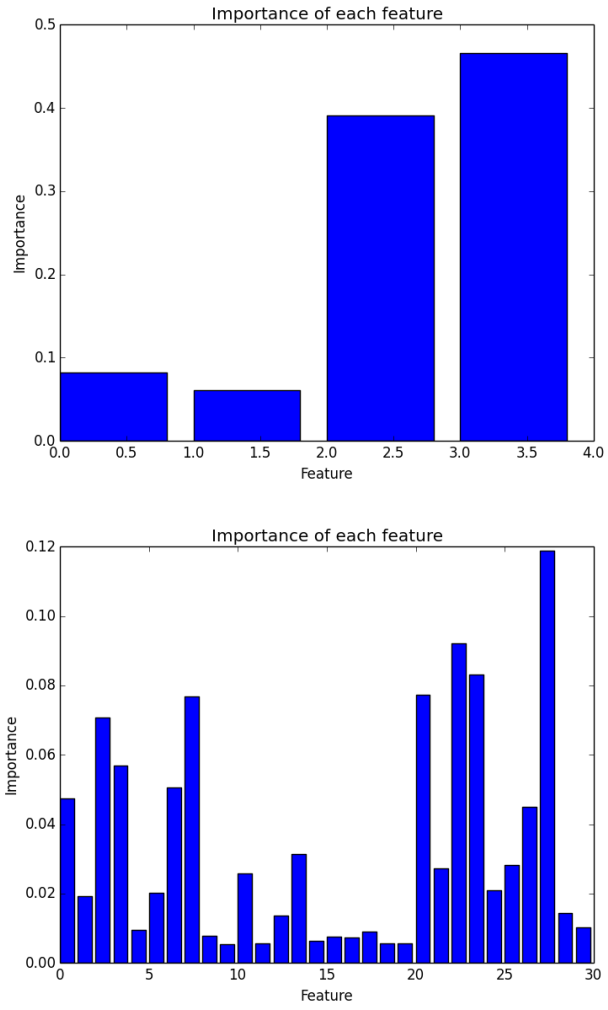


Figure 1: Importance of every feature in both cases

By analyzing those graphs we can quickly determine that the more important features for both cases are:

1. Features 2 and 3, corresponding to petal length (cm) and petal width (cm).

2. Features 0, 2, 3, 6, 7, 20, 22, 23, 26, 27. Those correspond to mean/worst radius, mean/worst perimeter, mean/worst area, mean/worst concavity and mean/worst concave points.

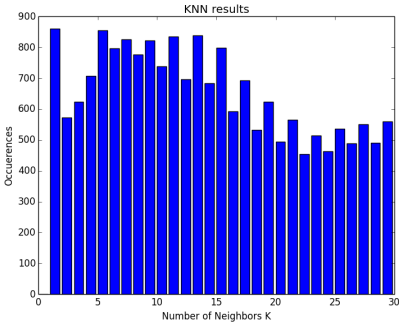
2.2 Determine the best k values

2.2.1 Initial approach

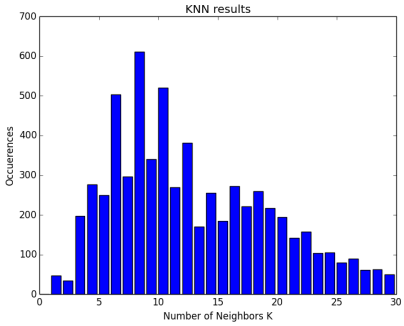
Once we had determine which features to use we had to determine the number of neighbours k to use.

To do so we made a script that trains 2000 times the knn algorithm for one dataset and one test size. After each iteration we stock the k for which the test result was the higher. At the end of the program we display the k that gave the most of time the higher result.

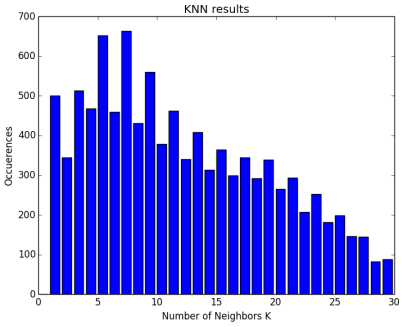
We runned that script for the two datasets, once with a test size at 0.33 and a second time at 0.66, and those are the results we have get:



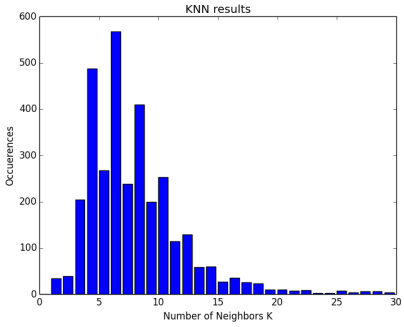
(a) iris, 0.33



(c) breastcancer, 0.33



(b) iris, 0.66



(d) breastcancer, 0.66

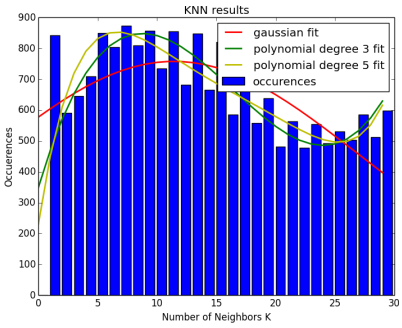
- Dataset iris, test size at 0.33, the best k is 1.
- Dataset iris, test size at 0.66, the best k is 7.
- Dataset breastcancer, test size at 0.33, the best k is 8.
- Dataset breastcancer, test size at 0.66, the best k is 6.

Those were satisfying results for the breastcancer dataset but not really for the iris one, we noticed that the k we got was the one for which the results were the highest but it was not always in the 'area' where the results were the highest.

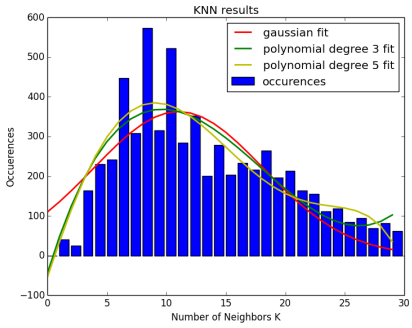
2.2.2 Final approach

To get a better approximation we decided to use functions to fit our datas: We first used a Gaussian function, the results were great for the breastcancer dataset but not for the iris one, so we decided to test with polynomial functions. We used a polynomial function of degree 3 and another one of degree 5, those two give better results for iris dataset.

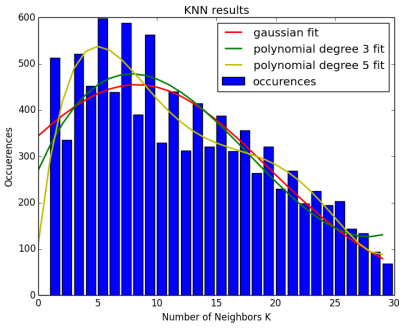
To select the best k we displayed each time those three functions, choose the function that seemed to fit the best our datas and defined k by the highest point on that curve:



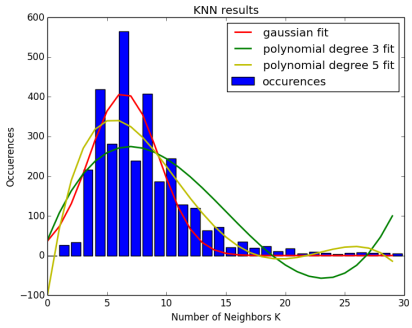
(a) iris, 0.33



(c) breastcancer, 0.33



(b) iris, 0.66



(d) breastcancer, 0.66

- Dataset iris, test size at 0.33, the highest point on best fitting function (poly 5) is 7.
- Dataset iris, test size at 0.66, the highest point on best fitting function (poly 5) is 5.
- Dataset breastcancer, test size at 0.33, the highest point on best fitting function (poly 5) is 8.
- Dataset breastcancer, test size at 0.66, the highest point on best fitting function (gaussian) is 6.

2.3 Comparing the training and test scores

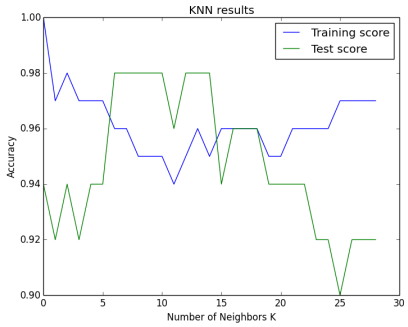
When we compare the training score to the test score we can detect three main cases:

– training score $\gg$ test score, it might be a sign of over fitting. The training score might be very high because there are a lot of datas in the training set and it is very adapted to it, but it won't be adapted for new datas coming from the test set.

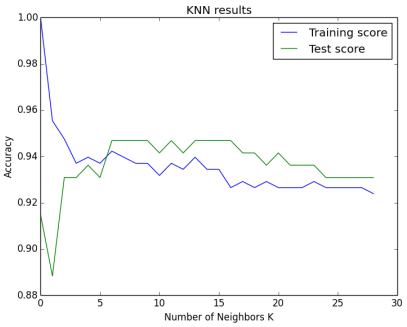
– training score $\approx$ test score, it might be a sign of good adaptability from the algorithm. It probably means that the number of neighbours k and the size of the training set are both well selected. Those are the settings that should give the best results on new test datas.

– training score and test score give both poor results, it is a sure sign of under fitting. It can mean that the k value is too high and that there are not enough training datas.

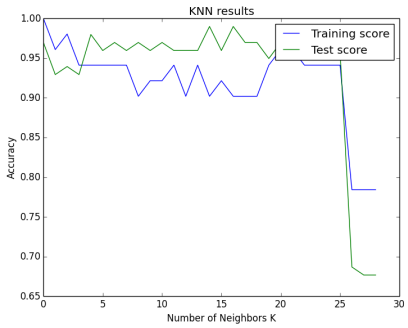
That means that if we have selected good values for our different k we should see that the test and the training score should be the closest to each other in the area of our k.



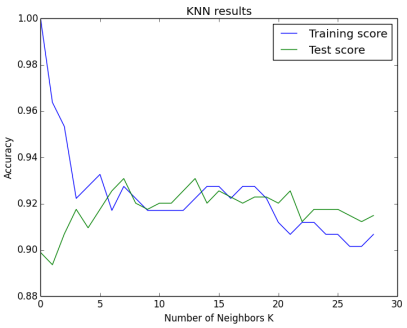
(a) iris, 0.33



(c) breastcancer, 0.33



(b) iris, 0.66



(d) breastcancer, 0.66

By analyzing the graphs on the figure above we can see indeed that around our selected values of k for the different cases the training and test score are quite close.

2.3.1 Under fitting

We can visualize a case in which the results are good but suddenly they considerably drop down due to high under fitting:

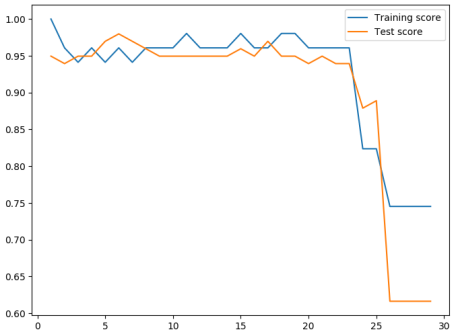
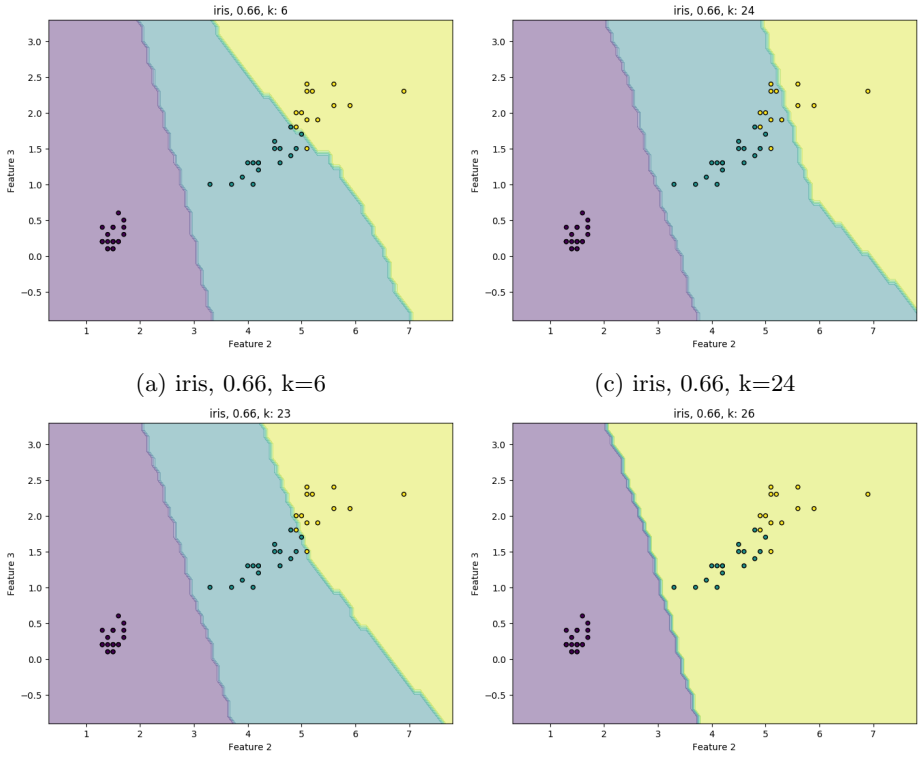


Figure 5: iris, 0.66

It is with the iris datapack and a set size at 0.66. With those settings the training set contains very few points for each of the three categories, consequently when the k value goes up the results drop down.

To illustrate that point we decided to display the boundaries created by the knn algorithm for the different k values and this is our result:



(a) iris, 0.66, k=6

(c) iris, 0.66, k=24

(b) iris, 0.66, k=23

(d) iris, 0.66, k=26

Figure 6: iris, 0.66, under fitting

We can see that under 23 the three boundaries are well placed, which matches our good results. At $k = 24$ the scores drop, we can see that the blue area goes too far on the yellow one, this is a big issue since about half the yellow points will now be classified as blue. At $k = 26$ the scores drop lower again, it is due to the fact that the blue area instantly disappeared. After that all the blue points will be classified as yellow, which is not good. We can see that the graph in figure 5 correspond perfectly to those changes of boundaries.

This is the perfect example of how under fitting works, it is all due to the fact that there is very few training datas for a too high value of k.

2.3.2 Over fitting

We also have a good example of over fitting, it is the iris dataset with a test size at 0.33:

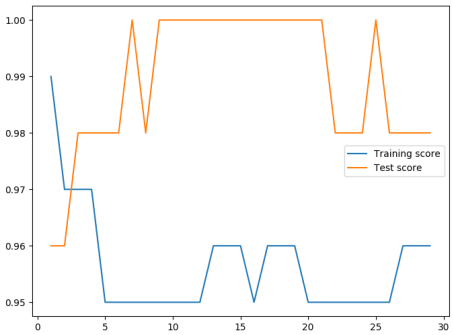
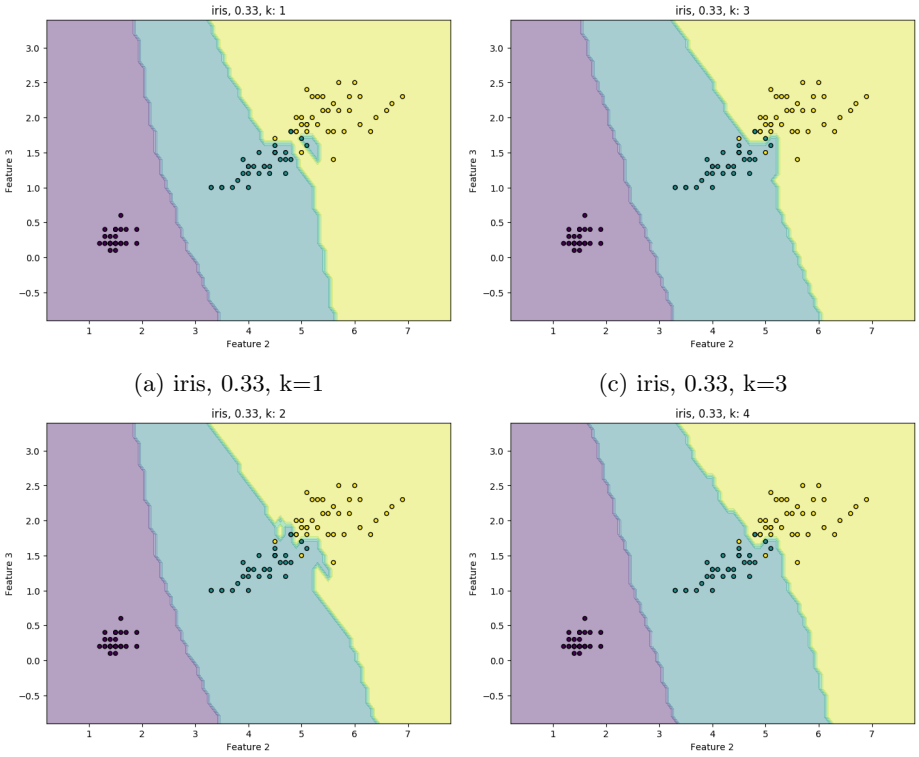


Figure 7: iris, 0.33

We can see in the graph above that for $k <= 3$ there are clear signs of an over fitting, the training score is higher than the test score. To prove that it is over fitting we once again displayed the boundaries:



(a) iris, 0.33, k=1

(c) iris, 0.33, k=3

(b) iris, 0.33, k=2

(d) iris, 0.33, k=4

Figure 8: iris, 0.33, over fitting

We can see on the figure 8a that there is an 'island', it is the blue zone in the yellow area. That is a clear sign of over fitting, it means that the points in that zone will be classified as blue when they are probably yellow. It can be explained by the fact that we only look for one nearest neighbours, when k will go up that 'island' disappears, as we can see on the other figures.

It correspond to what we can observe in figure 7, since the training score is above the test one for $k \leq 3$ and then the test score is above.

Chapter 3

Conclusion

We can conclude that if we expect satisfying results of classification using a knn algorithm it is important to chose a k value adapted to the size of the training set.

How to run our program

To make our tests we used 3 Python scripts that you can use as following:

- python knn.py X –percent Y
- python test_features.py X
- python test_k.py X –percent Y –iterations N

All scripts need the dataset X in the parameter:

$$X = \begin{cases} 1\text{: The Iris dataset} \\ 2\text{: The Breastcancer dataset} \end{cases}$$

For the *knn.py* and *test_k.py* scripts, the optional –percent parameter can be set. It determines the percentage of the dataset that must be used as training. By default it is 0.33.

For the *test_k.py* script, the optional –iterations parameter can be set. It determines the number of iteration needed to test the k value. By default it is 500.