Analysis of Performance of KNN on Waveforms Dataset

Chelsy Mena, Serhii Vakulenko — Students: (MLDM M1), University UJM

December 2024

Abstract

This paper presents the core of basic machine learning techniques to analyze the waveform data set and train a suitable k nearest neighbors classifier. Instead of using standard library solutions, our own implementations of the algorithms are proposed. The results demonstrate the stability of the classification accuracy when reducing the sample size, which indicates the effectiveness of the proposed approaches.

1 Introduction

K Nearest Neighbors is a supervised classification algorithm in which we strive to classify a point in an m-dimensional space by looking at the label of the k points closest to it by a predefined measure of distance. At its core, it is one of the simplest algorithms to implement, because there is only this distance to consider and store, but this operation is computationally expensive when done every time a new sample has to be classified.

To overcome this, several modifications to the process have been proposed. Some times the distances are used as weights so that the closest neighbors are more important in the classification, other times there is a precalculation and storage of the training samples and relevant distances to cut the necessary checks to do when a new point is queried. In this work, we will explore some of these strategies.

2 Core

The data set chosen for this study is the `waveform.data`, which consists of: 5000 data samples, 3 classes, and 21 attributes, all of which include noise [1]. The primary goal is to design, implement and evaluate a classification model, and compare its performance with the optimal Bayes classification rate of 86% accuracy.

2.1 Experimental Setup

Tools and Environment:

- **Programming Language:** Python
- **Libraries:** NumPy, pandas, seaborn, Matplotlib
- **Environment:** Jupyter Notebook

2.2 Exploratory Data Analysis

To understand the data structure and select the optimal classification methods, a comprehensive exploratory analysis of the `waveform.data` dataset was conducted. This allowed us to identify the characteristics of the data and guided further processing.

Basic statistics:

- 5000 observations, 21 features, 3 balanced classes
- Mean values and standard deviations of features in the 1 to 3 range

Data Distribution:

- Q-Q plots confirmed the normal distribution of features
- Random pairing 2D scatter plots revealed a single cluster structure with largely overlapping classes

Outlier analysis (IQR):

- Negligible number of outliers ($< 1\%$)
- Even distribution by features
- Does not require special processing [3]

The results indicate a high-quality and balanced dataset, which allowed us to focus on optimizing classification methods to solve the problem of class overlap.

2.3 K Nearest Neighbor Model Training

We started by splitting our data set into 4000 training samples and 1000 test samples completely at random.

The main metric for the discussions in this paper is the accuracy, defined as follows:

$$\text{accuracy} = \frac{\text{Correct predictions}}{\text{Total Predictions}}$$

We conducted a series of experiments, starting with a cross-validation to tune the hyper-parameter k (number of neighbors), in which we varied the value of k from 1 to 100 and the number of folds as well.

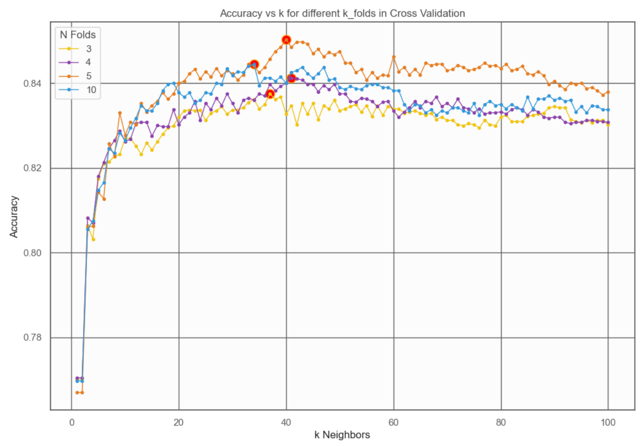

Figure 1: Accuracy vs k for different k-folds in cross-validation

2.3.1 Cross-Validation Optimization

According to the study **Kohavi (1995)** [2], stratified 10-fold cross-validation is considered the best method to assess accuracy. However, our results showed otherwise:

- **3 folds:** 83.75% (variance: 0.000284)
- **4 folds:** 84.13% (variance: 0.000256)
- **5 folds:** 85.03% (variance: 0.000158)
- **10 folds:** 84.45% (variance: 0.000362)

The 5-fold cross-validation provided the best balance between accuracy and stability, demonstrating the lowest variance among all configurations. The behavior of these experiments is shown in figure1.

2.3.2 Choosing the Optimal k

In the cross-validation procedure discussed above, the highest accuracy of 85.03% was achieved with $k = 40$. This value is smaller than the often used first heuristic of $\sqrt{n\_samples}$ which in this case is 63, perfect, since it is often recommended to choose the smallest k that provides a high accuracy (to reduce computational complexity).

- $k = 40$: maximum accuracy with acceptable complexity
- 5 folds: best stability of results with lower computational cost

2.4 Data Reduction Techniques

Two data reduction techniques were tested on the training data set. First, the Bayesian region cleaning, in which our algorithm attempts to remove as many points

k	**Data Set**	**Accuracy**	**Time**
1	Full	79.6	105.1
1	Bayesian Region	80.6	79.4
1	Condensed	74	14.85
40	Full	85.2	6.16
40	Bayesian Region	85	5.24
40	Condensed	82.7	0.85

Table 1: Accuracy and Execution Time for Different training Data Sets

as possible in areas in which two classes overlap; and secondly, we take this treated dataset and try to condense it as much as possible keeping only the points strictly necessary for a new prediction.

We found that in these sequential steps our data, which initially consisted of 4000 samples, dropped to 3163 after the first step, and after the second plummeted to only 602 observations. We tested the performance of our KNN classifier in the three mentioned datasets and found an 86.2% reduction in execution time with a loss of less than 3% in accuracy for a 1-NN, and similar results when using our previously tuned k = 40. The detailed results can be seen in table 1.

In this case, the drop of accuracy might be a deterrent to use the condensed data set, since the execution time is not too large in the first place. However, in larger applications with perhaps an impatient end user, it might be worthwhile. The Bayesian region cleaning specifically gives enough advantage without the loss of accuracy that it should be employed.

2.5 Speed-Up Technique

A KD-Tree approach was implemented with the intention of speeding up nearest-neighbor searches. But this was not the case for our data set, we noticed a 7% increase in the execution time, as shown in figure 2, even when we did not observe any loss of accuracy. It is very likely that this was caused by the fact that we did not have enough samples to fill the 21 expected levels of the tree. We could only reach 10 levels, and by then some leaves only had one point left.

As is known, as the number m of dimensions increases, the performance of trees of m dimensions deteriorates [4]. The obtained results then, are expected.

2.6 Data Imbalance

Data Imbalance Generation: Data Imbalance was generated by oversampling some of the classes to ensure

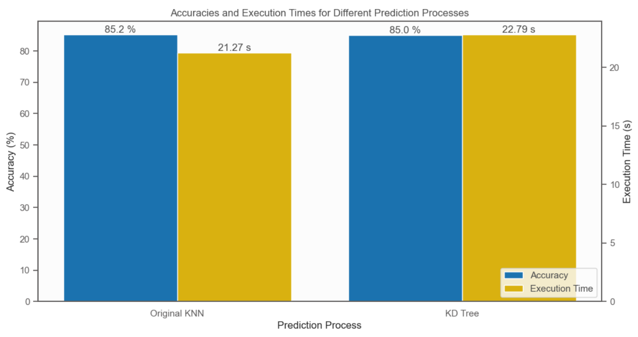

Figure 2: Accuracies and Execution Times for Different Prediction Processes

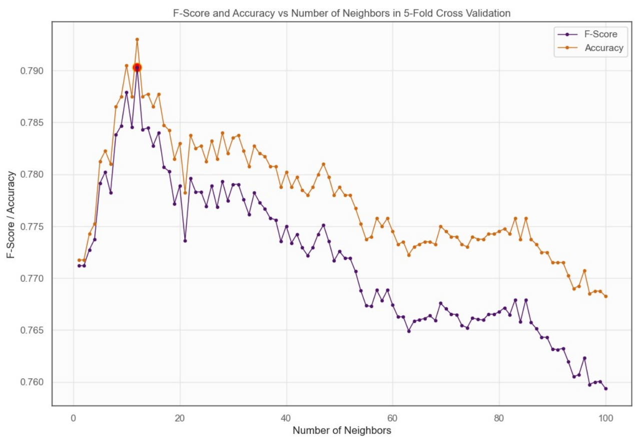

Figure 3: F-Score and Accuracy vs Number of Neighbors in 5-Fold Cross Validation

a chosen ratio of 20% for the `0.0` class, 30% for the `1.0` class and the remaining 50% for the `2.0` class.

Metric: The f-score is a measure of how a classifier behaves with respect to a class. There are two types of mistakes, a positive identification when the sample is not part of the said class, and a negative identification when it is. This is a per-class metric so to have an idea of the global behavior of the classifier we considered a version of the multi-class f-score in which the f-score for each class is calculated and then weighted by the proportion it constitutes in the training data set.

Model Training: With this new data set and performance metric, we performed a 5-fold cross-validation for which we found that the best result was an **f-score of 0.79 with k=12.**3

The quick peak and subsequent decline suggest that including more neighbors quickly starts to blur class boundaries. When we take few neighbors they are more likely to be of the same class as the point we are classifying, but if one class is represented much more in the data than the others, as the number of neighbors increases, the probability of misclassification for the less represented classes increases.

3 Analysis of Results

During the experiments, it was determined that the best performance of the k-NN model is achieved when using 40 nearest neighbors and 5-fold cross-validation, which provided the optimal balance of accuracy of 85.03% and stability of the results. The implementation of the data cleaning method based on the Bayesian region turned out to be an effective solution that allowed to significantly speed up the algorithm (by 86.2%) with minimal loss of accuracy. An attempt to improve the performance using the KD-tree structure did not bring the expected results due to the peculiarities of the dataset - the high dimensionality and insufficient number of samples led to a slowdown in the algorithm by 7%. The developed model demonstrated high performance, reaching an accuracy of 85.03%, which almost corresponds to the theoretical maximum of Bayesian classification (86%), and this is especially impressive considering the complexity of the task due to significant overlap of classes in the data.

4 Conclusion

The data set is quite robust as designed. Although it is stated that there is some noise in it, it can be treated as normal, and the little noise added only seems to have caused a marginal amount of outliers. Additionally, the similar scale of the values in the features ensure it is a fairly contained region of the hyper-space and as such, it is a good data set for a K Nearest Neighbors Classifier.

Perhaps this is why the various experiments carried out did not improve accuracy in any significant way, and why even in the simplest version of the algorithm its accuracy was already quite close to the performance found by the authors in [1] with a Bayesian classifier. However, it was possible to greatly reduce the time used to predict by reducing the data set until only the points that were strictly necessary remained, without sacrificing performance.

Further treatment of this dataset may include experiments in dimensionality reduction, to then apply other techniques that were hindered by the modest number of samples available.

References

[1] Leo Breiman. *Classification and regression trees.* Routledge, 2017.

[2] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Morgan Kaufman Publishing*, 1995.

[3] PJ Rousseeuw PJ and AM Leroy. Robust regression and outlier detection, 1987.

[4] Robert F Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6: 579–589, 1991.