Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-023-09882-x

*k*NN Classification: a review

Panos K. Syriopoulos[1] · Nektarios G. Kalampalikis[1] · Sotiris B. Kotsiantis[1] · Michael N. Vrahatis[1]

Accepted: 21 June 2023

Abstract
The *k*-nearest neighbors (*k*/NN) algorithm is a simple yet powerful non-parametric classifier that is robust to noisy data and easy to implement. However, with the growing literature on *k*/NN methods, it is increasingly challenging for new researchers and practitioners to navigate the field. This review paper aims to provide a comprehensive overview of the latest developments in the *k*/NN algorithm, including its strengths and weaknesses, applications, benchmarks, and available software with corresponding publications and citation analysis. The review also discusses the potential of *k*/NN in various data science tasks, such as anomaly detection, dimensionality reduction and missing value imputation. By offering an in-depth analysis of *k*/NN, this paper serves as a valuable resource for researchers and practitioners to make informed decisions and identify the best *k*/NN implementation for a given application.

Keywords *k*-nearest neighbor classifier · Lazy learning · Instance-based learning · Software · Benchmarks

Mathematics Subject Classification (2010) 62H30 · 68T05 · 68Q32

1 Introduction

The *k-nearest neighbors algorithm* (*k*NN) is a non-parametric, supervised learning classifier, which performs classification regarding the grouping of a given data point. Although *k*NN can be used for either classification or regression tasks, it is typically applied as a classification algorithm based on the assumption that similar points can be found near one another. In

✉ Panos K. Syriopoulos
p.syriopoulos@math.upatras.gr

Nektarios G. Kalampalikis
nkalamp@math.upatras.gr

Sotiris B. Kotsiantis
sotos@math.upatras.gr

Michael N. Vrahatis
vrahatis@math.upatras.gr

1 Computational Intelligence Laboratory (CILab), Department of Mathematics, University of Patras, GR-26110 Patras, Greece

general, in this approach the most commonly used distance is the Euclidean distance. The starting concept about the kNN approach is due to Fix and Hodges in 1951 [1], while Cover and Hart expands this approach in 1967 [2]. kNN is well-known and widely used algorithm. It is simple and accurate, particularly in the case of small datasets. In general, it is applied for tackling problems including among others, pattern recognition, data mining, recommendation systems, intrusion detection and financial predictions.

The kNN algorithm is related to applicable geometry as well as to notions of convex and discrete geometry. Specifically, from mathematical point of view the kNN approach is based on the problem to determine the radius of an n-dimensional sphere in the Euclidean space $\mathbb{E}^n$ centered at a given point $c \in \mathbb{E}^n$ that encloses k points of a given set of points in $\mathbb{E}^n$. This problem first appeared in 1860 by Sylvester in [3] who proposed the problem of drawing the smallest circle enclosing a given finite set of points in the plane. Also in 1901 Jung gave an answer to the question of best possible estimate of the size of an n-dimensional sphere in $\mathbb{E}^n$ of a given diameter, (*i.e.* the maximal distance of any two points of the set). Specifically, first, Jung established the following result in his dissertation in 1901:

Theorem 1 (**Jung's covering theorem (1901)** [4, 5]) *Let δ be the diameter of a bounded subset $\mathcal{M}$ of $\mathbb{E}^n$, (containing more than a single point). Then, (1) there exists a unique spherical surface S_r^{n-1} of smallest radius r enclosing $\mathcal{M}$, and (2) $r \leqslant [n/(2n+2)]^{1/2}\delta$.*

Also, an elegant proof of Theorem 1 was given by Blumenthal and Wahlin in 1941 [6]. Furthermore, for a more recent proof applying Helly's Theorem see [7].

Next, by defining the barycentric radius of a simplex to be the radius of the smallest ball centered at the barycenter and containing the simplex as well as the barycentric radius $\beta = \beta(\mathcal{M})$ of a subset $\mathcal{M}$ of $\mathbb{E}^n$ to be the supremum of the barycentric radii of simplices with vertices in $\mathcal{M}$, then by applying the Helly's theorem we have proved in [8] that a bounded set $\mathcal{M}$ can be covered by a ball of radius the barycentric radius $\beta(\mathcal{M})$, which in many cases gives a better result than Jung's covering theorem.

To this end, suppose that $\sigma^m = \{v^0, v^1, \ldots, v^m\}$ is an m-simplex in $\mathbb{E}^n$, $m \leqslant n$, (*i.e.* the convex hull of $m+1$ affinely independent points $v^i, i = 0, 1, \ldots, m$ in $\mathbb{E}^n$) and let $K = (m+1)^{-1}\sum_{i=0}^m v^i$ be its barycenter. Consider now the *barycentric radius* β of σ^m as $\beta = \max_i \|K - v^i\|$. Based on the above we give the following covering theorem for simplices.

Theorem 2 (**Covering theorem for simplices (1986)** [8, 9]) *Any m-simplex $\sigma^m = \{v^0, v^1, \ldots, v^m\}$ in $\mathbb{E}^n$, $m \leqslant n$, is enclosable by the spherical surface S_β^{m-1} with*

$$\beta = \frac{1}{m+1} \max_{0\leqslant i\leqslant m} \left\{ m \sum_{\substack{j=0\\ j\neq i}}^{m} ||v^i - v^j||^2 - \sum_{p=0,\, p\neq i}^{m-1} \sum_{\substack{q=p+1\\ j\neq i}}^{m} ||v^i - v^j||^2 \right\}^{1/2}.$$

The computation of the barycentric radius does not require any additional computational cost and it can be easily achieved during the computation of the longest edge δ that it is required for the Jung's estimate.

Suppose now that $\mathcal{M}$ is a bounded subset of $\mathbb{E}^n$, with diameter δ containing more than $n+1$ points, then the *barycentric radius* β of $\mathcal{M}$ is the supremum of barycentric radii between every set of $n + 1$ points of $\mathcal{M}$; if $\mathcal{M}$ contains ℓ points, $\ell \leqslant n + 1$, then the *barycentric radius* β of $\mathcal{M}$ is given by

$$\beta = \frac{1}{\ell + 1} \max_{0 \leqslant i \leqslant \ell} \left\{ \ell \sum_{\substack{j=0 \\ j \neq i}}^{\ell} ||v^i - v^j||^2 - \sum_{p=0, p \neq i}^{\ell-1} \sum_{\substack{q=p+1 \\ j \neq i}}^{\ell} ||v^i - v^j||^2 \right\}^{1/2} .$$

Now, since $\mathcal{M}$ is bounded and $\beta \leqslant n(n + 1)^{-1}\delta$, the set of barycentric radii of $\mathcal{M}$ is a bounded set and β is a positive (finite) number. Taking into consideration the above we give the following theorem.

Theorem 3 (**Improved Jung's covering theorem (1988)** [8]) *Let δ and β be the diameter and the barycentric radius, respectively, of a bounded subset $\mathcal{M}$ of $\mathbb{E}^n$ (containing more than a single point). Then (1) there exists a unique spherical surface S_r^{n-1} of smallest radius r enclosing $\mathcal{M}$ and (2) $r \leqslant \min\{\beta, \ [n/(2n + 2)]^{1/2}\delta\}$.*

From the above it is evident that issues related to kNN algorithms have been in the spotlight for many years. Next, we give additional introductory topics concerning kNN.

As already mentioned, kNN classification algorithm is a well-known and widely used non-parametric supervised learning approach. It belongs to the family of instance-based learning algorithms, a concept explained by Aha *et al.* [10]. That is, the training instances are stored in memory without explicitly learning a model. The training instances (referred to as "knowledge", training set, or simply dataset) will only be processed in the prediction phase. For each new data instance, a query to the database returns the k closest training instances based on some distance function. In the simplest case, the new object is classified to the majority class of its neighbors. The same approach is employed for non-parametric regression [11], the value of the query can be determined by the average of its neighboring training points. *Distance* always refers to distances between the d-dimensional feature vectors. The k-neighborhood of a vector x is denoted $N_k(x)$ and contains the k closest training points to x. If $C = \{1, 2, \ldots, m\}$ is the set of class labels, and $C(x')$ denotes the class label of point x', then the majority rule classifies query x by the expression:

$$\hat{C}(x) = \max_{i \in C} \left\{ \sum_{x' \in N_k(x)} \mathbf{1}\big(C(x') = i\big) \right\},$$

where $\mathbf{1}(\cdot)$ is the indicator function. Thus, the estimated probability of class i conditional on the position of the query in feature space, x, can be calculated as:

$$\hat{p}\big(C(x) = i \mid x\big) = \frac{1}{k} \sum_{x' \in N_k(x)} \mathbf{1}\big(C(x') = i\big).$$

Analogous formulas can be given for regression tasks. For a binary classification task (where $|C| = 2$), the decision can be given in terms of a threshold function

$$\hat{C}(x) = i \iff \mathbf{1}\big(\hat{p}(i \mid x) \geqslant 0.5\big).$$

Notice that the inequality is not strict, indicating that ties can be settled arbitrarily. The simple kNN majority rule exposes weaknesses that are relevant in all kNN revisions. Firstly, different values of the hyperparameter k result in different classifications. A natural trade-off between large and small values of k can be observed. Suppose that the training set consists of n classified points, $x_1, x_2, \ldots, x_n$, and denote the number of observations of each class as $C_i = |\{x : C(x) = i\}|$. Naturally, we would expect unequal class representations, that is, for some class i, $C_i > C_j$ for all $j \neq i$. If we choose the value of the hyperparameter k to be equal to be large enough, then the kNN majority rule will always yield $\hat{C}(x) = i$. On the other hand, if we choose $k = 1$, classification will be susceptible to errors due to outliers. Secondly, we can measure distances with a variety of functions, and each would produce a possibly different k-neighborhood. The choice of distance function is not obvious. Last but not least, the k-neighborhood has to be determined during classification at run-time. Additionally, modern day applications usually entail vast amounts of data. How do we address run-time complexity and storage complexity? What is more, the majority rule is just one of the available kNN classification rules. Is there merit in choosing a different classification rule? The answers to these questions form the four pillars of a good kNN application. That is, if the practitioner makes an informed **choice** of (a) the hyperparameter k, (b) the distance function, (c) the classification rule and (d) the query algorithm, then there is high probability that their implementation can reach their accuracy and complexity targets. These issues are further explored in subsequent sections. However, one should note that kNN is regarded as one of the top 10 algorithms in data mining, Wu et al. [12]. Many advocate its use even in competitive domains such as image classification, Boiman et al. [13], where it's currently found in the state-of-the-art, Bandaragoda et al. [14].

Despite the aforementioned challenges in classification, kNN plays a central role in data science in general. In fact, it is the weaknesses of kNN that inspired innovative ways to tackle data related issues. For example, in the area of noise reduction, a general idea is scoring points based on their similarity to their neighbors, *e.g.* Bandaragod *et al.* [15], Pang et al. [16]. Anomaly detection methods based on kNN also show competitive performance. Theoretical evidence show that kNN based anomaly detection methods show a 'gravity defiant behavior'. Specifically, Ting et al. [17] use computational geometry to find a closed form expressions for the lower and upper bounds of the *area under the receiver operating characteristic curve* for the 1NN anomaly detector. The proportion of anomalies (among other things) determines that optimal performance bounds occur at finite data volumes. A similar behavior is observed for other kNN-based anomaly detection methods. Another major area of application is in missing data imputators, a review is given by Berreta and Santaniello [18]. Triguero et al. [19] advocate the use of kNN methods as means of creating smart data out of big data. Additionally, the intuition that 'close' or 'similar' points have similar characteristics makes for an obvious use in recommender systems (*e.g.* Adeniyi *et al.* [20]). Other useful aspects of kNN can be found in subsequent sections.

This review article aims to provide an overview of the current research on k-nearest neighbor, help identify potential new applications, assess the advantages and limitations of the algorithm, teach readers how they can use the algorithm to solve specific problems, and list the available benchmarks, software, publications, along with citation analyses. Related review works include: [21] which is a very nice but short discussion of the advantages and disadvantages of kNN, [22] which provides a brief and insightful examination of distance metrics used in kNN, [23] an imbalanced-focused kNN study through weighting strategies, and [24] provides a comparison of different kNN variants and concludes that ARSkNN [25] outperforms the others in the examined large datasets. Another interesting study is [26] where it was concluded that Hassanat kNN was the best model for predicting diseases, due to its highest average accuracy of 83.62%, precision and recall values. However, we have exhaustively searched the literature and to our knowledge there is no other work that gathers the wide variety of kNN related topics in one paper, and this is what we attempt in this paper. The kNN is an incredibly versatile machine learning algorithm. It is a non-parametric method that achieves strong performance across a wide range of tasks in many domains. An effective k-nearest neighbor review article can be beneficial for both research and industry applications. Getting the most out of the kNN algorithm requires understanding the underlying mathematics, parameter selection, and feature preprocessing. With this knowledge users can tailor their kNN model to the specific task at hand.

The paper is structured as follows. Section 2 is a review of the basic theory behind kNN in statistical discrimination. Section 3 is solely concerned with classification: we delineate the sources of error, discuss methods for the determination of k, give an introduction to the concept of metric learning, and describe some key variations of the kNN classification rule. Section 4 discusses data reduction techniques, and Section 5 is a short survey on query algorithms. Sections 6 and 7 present available software and benchmarks. Section 8 is a synopsis with concluding remarks. Throughout the paper, the reader will find tables with publications and citation counts, as well as citations and links to available software. All presented tables include bibliometric data that were indexed by Google Scholar.

2 kNN theory

This section provides an account of advances in the literature of kNN theory. Our aim is to substantiate the four pillars of kNN methods as argued in the introduction. These are:

(1) the choice of hyperparameter k,
(2) the choice of classification rule,
(3) the choice of distance function and
(4) the choice of query algorithm.

We decided to include only what we believe are absolutely essential publications, and provide detailed high-level overviews on some of the results. In doing so, our purpose is to assist new researchers in developing a solid understanding of the theoretical foundation behind kNN, and provide a bridge between theory and application for the practitioner. The results presented here will help in the interpretation of several kNN variations cited in subsequent sections.

kNN was introduced by Fix and Hodges [27], originally published in 1951, and proposed as a method for non-parametric statistical discrimination[1]. First and foremost, the authors make a formal distinction between parametric and non-parametric methods. Consider the two population discrimination problem. Suppose a random variable Z, whose value is observed to be z, is distributed over a d-dimensional space according to either distribution F, or distribution G. The problem of statistical discrimination is to decide on the distribution of Z based on its observation. More specifically, we would like to find a function whose value at z determines at which population it belongs to. According to Fix and Hodges, the practitioner may find themselves in three possible scenarios where:

(i) the density functions of F and G (denoted f and g) are known,

[1] The word ‘discrimination’ refers to methods of assigning an observation to its probability distribution, deciding over a finite set of possible distributions of origin. The subtle difference between ‘discrimination’ and ‘classification’ is that the later assigns one out of a chosen number of labels.

(ii) the densities of F and G are known except for a subset of parameter values,
(iii) the existence of density functions f and g can only be assumed.

In the first case, an optimal discrimination rule (in terms of expected accuracy) has been proven by Welsh in 1936 [28], and is given by a threshold function determined by the ratio of the two densities at point z. Notice that case (ii) reverts back to case (i) after estimation of the missing parameters (based on some sample of observations). In case (iii), *i.e.* in the absence of any knowledge, an informed decision to be made from the sample of observations only. The kNN rule is particularly designed for case (iii), non-parametric[2] as it makes no assumption of the underlying distributions other than that they exist, and is the first of its kind. Obviously, any approach in cases (ii) or (iii) can never surpass the performance of the optimal decision rule in the setting of perfect information. The question of interest is: how does the kNN rule compare to the optimal rule if knowledge about the densities were to be known?

Fix and Hodge's result can be summarised as follows. Consider samples $X = \{X_1, X_2, \ldots, X_n\}$ from F and $Y = \{Y_1, Y_2, \ldots, Y_m\}$ from G. Assume a metric with which distances can be measured. The critical assumption about F and G is that they are absolutely continuous. This will ensure that the density functions f and g are continuous almost everywhere with respect to the Lebesgue measure. Given observation z, consider a neighborhood around z, $N(z)$ with a given diameter, and denote by r_X and r_Y the ratio of points of X and Y in $N(z)$ respectively. In the asymptotic case where the number of observations n and m grow to infinity (with similar speeds), and the diameter of the neighborhood around z is decreasing slowly, the ratio r_X/r_Y coincides with the optimal rule (derived from perfect information) with probability one.

There are several observations to be made. First, the rule described above concerns the ratio r_X/r_Y, which is different from the majority rule generally employed in kNN applications. Thus, we shall refer to this rule as the kNN discrimination rule. Secondly, the classification decisions of the kNN discrimination rule coincide with those of the optimal rule for (almost) every point in space. That is, asymptotically, the two rules have the same error rates and agree on their assignments with probability one. Additionally, the relative representation of points from F and points from G need not be balanced. The only requirement is that the ratio m/n, together with its inverse, is bounded away from zero in the limit. However, questions can be raised regarding the applicability of this rule. For a good approximation of the optimal rule, a large number of nearest neighbors is required, but with a count still relatively minuscule compared to the total number of observations. Nevertheless, the result provides a theoretically sound method for non-parametric point density approximation. The ratio r_X/r_Y corresponds to the ratio of the estimated densities at point z. The formula for the estimated density is given later in this subsection. Note that the same result can be extended to discrimination of several populations.

The first theoretical result about the majority rule was published by Cover and Hart in 1967 [2]. Upper and lower bounds were given on the performance of: (a) the kNN majority rule in the two class problem, (b) the 1NN majority rule in M class problems. The intermediate result is that, in any separable metric space, and sample of n i.i.d random variables $x_1, x_2, \ldots, x_n$, the k'th nearest neighbor of a point x, denoted by x^k, converges to x with probability one as the sample size, n, increases to infinity. Furthermore, the aforementioned result is true for any probability measure. The bounds found by Cover and Hart are as tight as possible and are given in terms of the (optimal) Bayes risk which we define below.

[2] Due to the non-parametric nature of kNN methods, the parameter k, and any other parameter found in variants of kNN, are correctly referred to as *hyperparameters*.

Assume M probability density functions $f_1, f_2, \ldots, f_M$ corresponding to M classes with prior class probabilities $h_1, h_2, \ldots, h_M$ summing to one. The probability of class i, conditional on observation x (referred to simply as conditional class probability) is denoted $h_i(x)$. From Bayes theorem it follows that:

$$h_i(x) = \frac{h_i f_i(x)}{\sum_{j=1}^{M} h_j f_j(x)}. \tag{1}$$

The Bayes risk is given in terms of the conditional class probabilities and a loss function $L(i, j)$, which reflects the cost of allocating an observation of class i to class j. It follows that the risk associated with placing observation x to category j is:

$$r_j(x) = \sum_{i=1}^{M} h_i(x) L(i, j). \tag{2}$$

The optimal decision rule selects the index j for which the risk is minimal, denote it by $r^*(x)$. We can define risk for the kNN rule in the same manner. Following this paradigm, the bounds on the expected risk of the 1NN rule, denoted by R, are given in terms of the expected risk of the optimal strategy, $R^* = E[r^*(x)]$. For a 0-1 loss function, risk coincides with error rate and we have the following inequalities:

$$\begin{aligned} &R^* < R < R^*(1 - R^*), && M = 2,\\ &R^* < R < R^*\Big(2 - \frac{M}{M-1} R^*\Big), && M > 2. \end{aligned}$$

In both cases (for all values of M), the 1NN rule is bounded above by twice the Bayes risk. In that sense, in the asymptotic case, half of the available information is contained in the nearest neighbor, *i.e.* any other decision rule can at best reduce the 1NN error by a half. In the paper it is also shown that the bounds for 1NN and kNN coincide for $M = 2$.

Contrary to the intuitive belief that a large k value should prove more robust, Cover and Hart demonstrate by example that the 1NN rule has a strictly lower probability of error than any other rule with $k \neq 1$ for problems where the densities have disjoint support. This raises questions regarding the optimal value of k for different kinds of problem. Furthermore, the combinatorial nature of the given argument considers all possible class assignments of samples of fixed size. Thus, undoubtedly, we shall expect that the optimal k value for a particular sample depends (apart from the true distributions) on the relative representation of each class, however, in classifying unseen observations, the k value that yields minimal expected risk should also depend on the prior probabilities. The later issue in the literature of classification is known as the class imbalance problem. Finally, notice that for $M = 2$ the upper and lower bounds coincide in situations of perfect certainty ($R^* = 0$) and perfect uncertainty ($R^* = 1/2$).

Following these results many kNN variants followed shortly after. An idea that led to another theoretical milestone was the one proposed by Hellman in 1970 [29]. Designed to be of practical interest, this kNN rule rejects points whose k-nearest neighbors fail to form a large enough majority. The idea is that by sacrificing the ability to classify every single point, the performance of the kNN rule with rejection option can improve in terms of error rate. More specifically, the (k, l) rule rejects points if the majority class fails to amass at least $l > k/2$ observations among the k-nearest neighbors. It was proven later by Loizou and Maybank [30] that the asymptotic behavior of (k, l) rules can approximate the Bayesian error rate with rejection option, a conjecture at the time. There could be more kNN rules with error rates consistent with the Bayesian rule, or at least within acceptable bounds of error. A lot of the modern kNN variations come without analogous proofs, nevertheless, there has been a lot of success in improving the error rates.

So far we were concerned with the asymptotic behaviors of kNN. However, on the application side, the topic of interest is performance on finite samples. A crucial paper by Fukanaga and Hostetler in 1981 [31], calculates the locally optimal value of k for the kNN density estimate. The approximate expression uncovers a relationship between the number of dimensions, the distance metric used, and the local curvature of the density function with the locally optimal value of k. Although cumbersome, we provide an outline of this result because it induces strong intuitions regarding what constitutes a good choice of k and a good choice of distance metric.

The outline goes as follows. Consider a random variable X in d-dimensional space having distribution F, and a random sample of n independent, identically distributed (i.i.d.) points from F. Furthermore, assume a quadratic distance metric s.t. $d^2(X, Y) = (Y-X)^\top A(Y-X)$ for some symmetric and positive-definite matrix A. We can form the following random variables:

distance to the k'th NN :	d_k,
k-sphere around X :	$S(X) = \{Y : \ d(X, Y) \leqslant d_k\}$,
volume of the k-sphere :	$v(X) = \int_{S(X)} dY$,
kNN density estimate from Fix and Hodges :	$\hat{f}_n(X) = \dfrac{k-1}{Nv(X).}$

The sphere around a random point X, denoted $S(X)$, has radius defined by the distance to the k'th nearest neighbor. Consequently, the volume of $S(X)$, denoted $v(X)$, also depends on the distribution F and the number of sample points n (and also dimension d). This dependence is exploited to find an expression for the expected square error between the kNN density estimate and the true density, which is given by:

$$\begin{aligned} J_n(X) &= E\{[\hat{f}_n(X) - f(X)]^2\} \\ &= E\Big\{\frac{1}{v^2(X)}\Big\}\Big(\frac{k-1}{n}\Big)^2 - 2E\Big\{\frac{1}{v(X)}\Big\}\Big(\frac{k-1}{n}\Big) f(X) + f^2(X). \end{aligned} \tag{3}$$

The idea is to optimize the expression above in terms of k. In order to do this, estimates for the expectations of $1/v(X)$ and $1/v^2(X)$ are required. The authors decide to approach this issue by expressing the volume in terms of the coverage of the sphere $S(X)$, given by:

$$u(X) = \int_{S(X)} f(Y)\, dY.$$

The advantage identified by the authors is that $u(X)$, seen as a random variable, follows a Beta distribution with parameters $(k, n-k+1)$. Assuming continuity of the partial derivatives of third order for the density f, and using the symmetry of the sphere $S(X)$, a Taylor expansion of f yields the following relationship:

$$u(X) \approx f(X)v(X) + c(X)v^{1+2/d}(X), \tag{4}$$

where $c(X)$ contains dependence on (i) the dimension, (ii) the matrix A used in the distance calculations, and (iii) the curvature of the density f. The expression above can be exploited for an approximate solution for $v(X)$ in terms of $f(X)$, $c(X)$, and $u(X)$[3]. What is left is taking expectations and substituting in (3) for $J(X)$. After minimizing $J(X)$ for k, the optimal k value is given by:

$$k_0(n) = \left(\frac{d(d+2)^2 \pi^2 f^{2+2/d}(X)}{\Gamma^{4/d}\left(\frac{d+2}{2}\right) \mathrm{Tr}^2 \left\{ \left[\frac{A}{|A|^{1/d}} \right]^{-1} \left[\frac{\partial^2 f(X)}{\partial X^2} \right] \right\}} \right)^{\frac{d}{d+4}}, \tag{5}$$

where Γ is the Gamma function, $Tr(A)$ the trace, and $|A|$ the determinant of operator A.

Equation (5) delineates the relationship between the locally optimal k, the dimensionality of the space d, the distance metric (induced by matrix A), and the curvature of the probability density function. We can make the following observations regarding the optimal $k_0(n)$ at point X for a sample of n observations:[-0.25cm]

(a) the optimal k depends on the dimensionality in a non-trivial manner,
(b) the optimal k is inversely related to a 'measure' of curvature (given by the trace in the denominator),
(c) the optimal k depends on how the distance metric 'measures' curvature.

We should expect that if the density function changes abruptly in the local region, a smaller sphere around the point X will yield a better estimate. Choosing a smaller k value will produce a smaller sphere $S(X)$, and therefore, possibly, a more precise estimate. To unpack the effect of the distance metric, the authors optimize the overall square error (integral of (3)):

$$I_n = \int J_n(X)\, dX.$$

For the class of Gaussian density functions, it was found that the optimal A matrix is given by the inverse of the covariance matrix $A^* = \Sigma^{-1}$. That is, distances along directions of low variance should contribute more in the distance calculation.

To summarize, we have provided overviews of crucial theoretical results concerning kNN. We clarified the difference between parametric and non-parametric estimations. We have seen that the kNN discrimination rule coincides with the (optimal) Bayesian rule in the asymptotic case, and touched upon the subject of imbalanced learning. We have seen that the error rate of the kNN majority rule is bounded above by the twice the Bayes error, and established that different problems have different optimal values of k. We provided resources for theory behind the kNN rule with rejection option. Finally, we provided intuition regarding the relationships between error rates, distance measure, k value, and dimensionality with mathematical evidence.

3 Classification

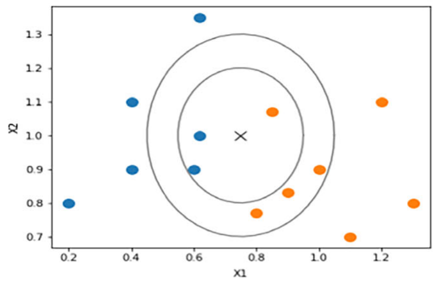
This section focuses on kNN classification. We discuss the sources of error as well as survey some of the kNN variations. We explore the core issues related to the choice of hyperparameter k, the choice of distance metric, and the choice of decision rule in association with possible characteristics of the observed distributions in the data.

[3] The key observation here is that the volume containing the k-nearest neighbors of X shrinks in expectation as the number n of observations increases. Therefore, we can solve for the volume $v(X)$ in equation (4), and use the approximation $u(X) = v(X) f(X)$ to replace the $v^{2/d}$ term.

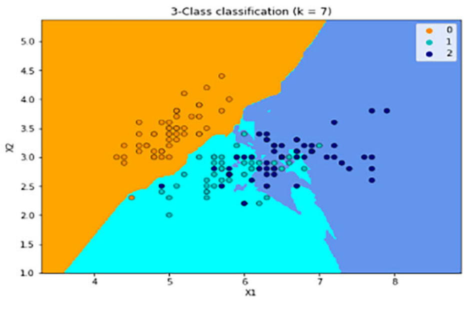
3.1 Sources of error

As in many classification methods the sources of error include overlapping class regions, class imbalances, outliers and noise. Apart from data preprocessing methods, the means by which we can tackle these issues are: the choice of k, the choice of distance metric, and the choice of classification rule. For example, the following figures depict the decision boundaries locally and globally for the kNN majority rule under the Euclidean metric. Notice that in Fig. 1(a) the classification of the query point (depicted as x) is different for $k = 3$ and for $k = 6$.

As a general principle, the severity of each source of error should be evaluated in association to the observed distributions of the data. We can identify class overlap as the fundamental source of error. This can be argued directly from the fact that the kNN discrimination rule (and the majority rule within an error bound) approximate the Bayesian rule as the sample size tends to infinity. To be precise, consider two normal distributions with different means and unit variance, $N(\mu_1, 1)$ and $N(\mu_2, 1)$. Suppose the priors are equal and the density functions f and g overlap. Assuming $\mu_1 < \mu_2$, we can find a point $\mu_1 < x^* < \mu_2$ such that $f(x^*) = g(x^*) = p^*$. At x^*, the optimal Bayes rule given by the ratio $f(x^*)/g(x^*) = 1$ is at a state of complete uncertainty, *i.e.* the classification of x^* can be settled arbitrarily. At the same time, points in the interval $I_{(-)} = (x^* - \epsilon, x^*)$ satisfy $f(x)/g(x) > 1$ and are always classified as coming from f. However, there is always a probability that the true density of origin was g, which results in an inevitable source of error in the Bayesian framework. The same argument can be made for points in $I_{(+)} = (x^*, x^* + \epsilon)$. In this case, the effect on the error rate depends on p^*. In small samples the effect of overlaps can be even more severe. Tang and He [32] demonstrate by example that the deviation of the kNN majority rule from the Bayes rule can be significant. They ascribe this issue to differences in density: the distribution with higher density at the query point is more likely to win the majority vote. They attempt to tackle this challenge with the proposed *Extended-NN* (ENN) rule. The idea deployed here is to look at classified datapoints which consider the unseen example to be in their k-neighborhood, and iteratively calculate a coherence statistic. The coherence of a class is based on the average number of correct inclusions in the k-neighborhoods of the corresponding datapoints. The assignment that maximizes the sum of the class coherence statistics wins. It can be argued that this rule matches the Bayes decision more closely near the intersection of the observed densities. A recent approach on the matter is that of Yuan et al. [33].



(a) k-nearest Neighbors for k=3 and k=6.



(b) Decision boundary plot.

Fig. 1 In plot (a) the unseen observation is classified as blue for $k = 3$ while it is classified as orange for $k = 6$. Plot (b) exhibits the decision boundary formed in a 3-class example

The problem of class imbalance has been widely studied in the context of machine learning (see for example He and Garcia [34], Fernández *et al.* [35]). Data imbalance generally refers to cases where one or more classes are underrepresented in the dataset. Reasons for imbalances in the data can be intrinsic (some classes have naturally smaller priors compared to others), or extrinsic (these usually relate to data acquisition or maintenance costs). We can further differentiate between relative imbalance, *e.g.* cases where some classes are underrepresented but still there is a reasonable volume of data in each class, and imbalance due to rarity, where some classes are rare to find even in the region of space where they reside. In occurrences of relative imbalance, it would be natural to try to incorporate the prior probabilities in the decision rule. Cases of imbalance due to rarity are different. Whether or not there are reasons to believe that the dataset provides a good approximation of the priors, the complication here is that the k-neighborhood of a query will rarely include a minority instance. A common approach is to look for the k-nearest neighbors within each class and bias the decision appropriately. For example, Zhang *et al.* in [36] propose the *K-Rare Nearest Neighbors* (KRNN) where k is variable so as to include at least some examples of the under-represented classes. The decision is made based on the confidence levels of globally and locally defined confidence intervals. A recent survey by Sun and Chen [23] outlines several kNN classification rules specifically designed for imbalanced datasets. In another recent survey by Zhang [37], several generic classification rules are defined and tested on artificially imbalanced datasets. The following table, Table 1, consists of kNN algorithms specifically designed for imbalanced datasets sorted by year together with their citations:

Noise can be divided into noise in the attributes and noise in the class labels. The later implies that some points in the knowledge-base have erroneous class labels. Their effect can be mitigated by simply choosing a larger k value. The same applies in the presence of outliers, larger k values are generally more resistant. When it comes to noise in the attributes there is evidence that certain distance metrics are more resilient than others (Abu Alfeilat *et al.* [53]).

Table 1 kNN for handling imbalanced datasets, where "Ref." indicates the Reference, "TNC" stands for total number of citations, while "CpY" indicates the citations per year

Ref.	Author(s)	Year	TNC	CpY
[38]	Zeraatkar and Afsari	2021	13	6.5
[39]	Wang *et al.*	2020	13	4.3
[40]	Patel and Thakur	2017	28	4.7
[41]	Liu *et al.*	2017	58	9.7
[42]	Li and Zhang	2017	72	12.0
[43]	Nikpour *et al.*	2017	13	2.2
[44]	Ando	2016	26	3.7
[45]	Yu *et al.*	2015	107	13.4
[46]	Zhu *et al.*	2015	41	5.1
[47]	Hajizadeh *et al.*	2014	14	1.6
[48]	Dubey and Pudi	2013	67	6.7
[49]	Zhang and Li	2013	33	3.3
[36]	Zhang *et al.*	2013	87	8.7
[50]	Kriminger *et al.*	2012	37	3.4
[51]	Liu and Chawla	2011	195	16.3
[52]	Song *et al.*	2007	320	20.0

An effective noise reduction technique is to simply disregard what would be misclassified points in the training set (Fig. 2).

The final remark, and perhaps the biggest challenge in kNN classification, is that the sources of error can manifest in different ways at different regions of space. For example, a class may be dense in a region of space but rare in another. The magnitude of noise may correlate with certain attribute values. The density of a class may vary rapidly in some region (suggesting a small k value), but slowly in another. We believe that a systematic way of identifying sources of error locally would quickly translate to significant accuracy improvements.

3.2 Choice of k

In Section 2 the locally optimal k value was shown to depend on (a) the dimensionality of the problem, (b) the distance metric used, (c) the convexity of the probablity density. Efficient methods for selecting the value of the hyperparameter k is an ongoing endeavor. In terms of accuracy we believe that the most clear-cut approaches performs the best. A notable example, is the assignment of a k value to each observation in the dataset, through ten-fold cross-validation. This idea was proposed by García-Pedraja et al. [54] and experiments show improved accuracy over regular kNN. The selection of k values considers both the local best k and the global best k so as to avoid large deviations in the values assigned to neighboring points. During classification, every new instance, inherits its nearest neighbor's k value. There also are approaches based on direct (convex) optimization, that may improve accuracy. For instance, Zhang *et al.* [55] propose a reconstruction method. The knowledge base is divided in a training set and a test set. Let each row of the matrices X, Y correspond to a datapoint in the training set and test set respectively. Reconstruction methods attempt to find a weight matrix W that minimizes the loss function $L = \|WX - Y\|$ where $\| \cdot \|$ is the Frobenius norm. Achieving a sparse W matrix is key, so, L_1 and L_2 regularization terms are added to the loss. Zhang includes a locality preserving regularization term and positivity constraints to ensure that points are reconstructed from their neighbors. For each data point in the test set, the number of non-zero elements of the corresponding row of W determine a suitable k value. The k value of future queries is inherited from neighbors. This work is extended in Zhang *et al.* [56], a decision tree (named k^*-tree) is constructed for faster queries for a loss in accuracy.

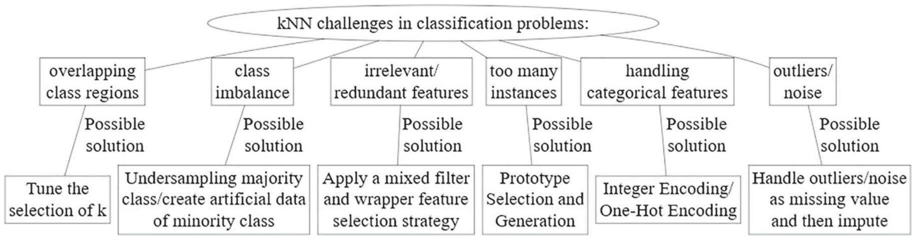

kNN challenges in classification problems:
overlapping class regions
class imbalance
irrelevant/ redundant features
too many instances
handling categorical features
outliers/ noise
Possible solution
Tune the selection of k
Undersampling majority class/create artificial data of minority class
Apply a mixed filter and wrapper feature selection strategy
Prototype Selection and Generation
Integer Encoding/ One-Hot Encoding
Handle outliers/noise as missing value and then impute

Fig. 2 k-nearest Neighbor challenges in classification problems

3.3 Choice of distance metric

Generally speaking, data instances with d features are considered as points within an d-dimensional feature space. Since the prediction is determined by comparing the distances to the query, the absolute position of instances is less significant than their relative positions. Ideally, the distance metric should maximize the distance between instances of different classes, while minimizing the distance between instances belonging to the same class.

Abu Alfeilat *et al.* [53] experimented upon a large number of different distance functions on real world datasets and concluded that classification performance is drastically affected by the choice of distance function. In particular, they came to the conclusion that there is no optimal distance function that suits all datasets, and metrics belonging to the same families showed analogous classification results. This could be attributed to the fact that some application domains favor certain metrics over the others. For example, a similar expariment for the medical domain conducted by Hu *et al.* [57] found that the Chi-square distance function performs the best. Moreover, it was found that a few distance metrics were more resilient to noise compared to others. Noteably, one of the best distance functions in [53] was non-convex.

Apart from employing cross validation, one could use distance metric learning to find an appropriate distance metric. For the purpose of maximizing classification accuracy these methods involve applying a linear or nonlinear transformation to the feature space. A convex optimization approach was for the first time proposed by Xing *et al.* [58]. They considered the Mahalanobis distance which is defined by $d_A(x, y) = (x - y)^\top A(x - y)$, where $A = W^\top W$ is a positive semi-definite matrix $A \succeq 0$, Here W corresponds to the space transformation. The intuition behind their idea was to find the optimal matrix A that maximizes distances between points in different classes and minimizes distances between points belonging to the same class. Let $y_{ij} = 1$ if the i'th and j'th points in the knowledge base are part of the same class, that is $y_i = y_j$ and zero otherwise. The objective function can be written as:

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} (1 - y_{ij}) \sqrt{d_A(x_i, x_j)}, \\ \text{subject to:} \quad & \sum_{i,j} y_{ij} d_A(x_i, x_j) \leqslant 1, \\ & A \succeq 0. \end{aligned} \tag{6}$$

The first condition forces points belonging to the same class to stay close, while the objective maximizes distance between points of different classes. Similarly, in the approach of Shalev-Shwartz *et al.* [59] a Mahalanobis metric is learned online. A scalar threshold b forces points belonging to the same class to be at most $b-1$ distance apart. Conversely, points in different classes are at least $b + 1$ distance apart. Both approaches boost the performance of kNN and also have the advantage of being convex optimization problems. However due to the nature of kNN methods a weakness of linear space transformations shows up. Different patterns in the relevant positions of points may be observed in different locations of the feature space. *Neighborhood Component Analysis* (NCA) is a novel idea that tries to circumvent this problem. Proposed by Goldberger et al. [60], in NCA points are assigned random neighborhoods. The probability that point j ends up in the assigned neighborhood point i in the transformed space is inversely related to their distance. Weinberger *et al.* [61] based on the idea that only the distances of neighboring points are relevant, formulated a convex optimization problem which shows similarity to that of Eq. (6), but instead of summing over all pairs of points, it sums over neighbors. The resulting transformation is known as the *Large Margin Nearest Neighbor* (LMNN).

Taking into account the aforementioned points, it is evident that the choice of metric is crucial to kNN applications. Linear transformations do improve the accuracy of kNN, however, one disadvantage is that they are global. Attempts to localize the optimal distance metric are certainly relevant. By extension, all literature on space transformations can contribute to improve the accuracy of kNN classification.

3.4 Variations of kNN

In this section we showcase a number of different kNN rules that address the sources of error in different ways. Certain categories of kNN rules are often used as building blocks of more elaborate classifiers. Our rough categorization is: local hyperplane methods, fuzzy methods, weighting methods, and reconstruction methods. Many of the latest kNN rules involve combining these approaches meaningfully.

The kNN variant of k-local hyperplanes was introduced by Pascal and Yoshua [62]. The general intuition is that each class lays on a non-linear manifold in feature space. The rule assigns unlabeled data points to the closest manifold. Let $\mathcal{C} = \{1, 2, \ldots, m\}$ be the set of class labels. The k-nearest neighbors from each class are determined for a total of km nearest neighbors. If $\mathcal{N}_{C_i} = \{x_1, x_2, \ldots, x_k\}$ are the k-nearest neighbors belonging to class i, then the local approximation of the corresponding manifold consists of a linear combination of the points in $\mathcal{N}_{C_i}$. For each i in $\mathcal{C}$ the coefficients of the linear combinations are optimised in such a way, so as to minimize the distance to the unclassified observation. Another notable kind of kNN algorithms are fuzzy kNN algorithms. Further study material can be found in Derrac *et al.* [63]. Weighting schemes to weight the labels and/or features, some examples are Gou *et al.* [64], Dudani [65], Liu and Chawla [51]. An example of reconstruction methods is in Zhang *et al.* [66]. They employee the locality preserving projection developed by He and Niyogi [67] to reconstruct the test sample from the training sample, thus producing a weighting of nearest neighbors and a good selection of k.

Recent years have seen many researchers combine the known approaches and embed their own innovation into them. Yu *et al.* [45] integrated the method of k-local hyperplanes to a fuzzy relative transform decision rule for the purpose of dealing with class imbalances. Susan and Kumar [68] applied a linear transformation (*e.g.*, LMNN or NCA) to the feature space. Then, the k-nearest neighbors were partitioned into two clusters: the points that are closest to the farthest neighbor, and the points that are closest to the closest neighbor. The second cluster is used for a final decision. These are but a subset of examples with fruitful combinations of ideas.

There are many more recent algorithms, *e.g.* the algorithm presented in [69] is an efficient and improved version of the k-Nearest Neighbor algorithm called Ensemble Centroid Displacement-Based k-NN. The algorithm leverages the homogeneity of nearest neighbors of test instances to increase accuracy. In [70] a non-parametric method for nearest neighbor classification using global dissimilarities in variance was proposed and tested on twelve real datasets. The results showed that the proposed algorithm achieved higher accuracy compared to the Local Mean k-Nearest Neighbor algorithm.

4 Feature selection and data reduction

It is known that the performance of query algorithms is severely degraded in the presence high-dimensional data. Feature selection (FS) methods reduce the dimensionality of the data but can also improve performance by disregarding irrelevant or redundant dimensions of the data. A comparative study by Rogati and Yang [71] showed that the accuracy of kNN can benefit greatly from FS. A broad categorization of FS methods comprises of filter-based methods, wrapper-based methods, embedded methods, and hybrid methods. The purpose is to identify small subsets of features that produce the same or even greater accuracy. For more information we refer the reader to Li *et al.* [72]. There are advantages in using a kNN classifier as a base for many of these approaches. Consider a general wrapper-based strategy for FS that includes (a) a feature subset selector, (b) a feature subset evaluator. There are many strategies for selecting feature subsets, for kNN, Tahir *et al.* [73] propose a Tabu search strategy. The feature subset evaluator is a classifier, which in most cases needs to be re-trained and tested for every feature subset. Fundamentally, kNN has minimal to no-training stage. Additionally, many distance functions can be calculated recursively. For example the Euclidean distance satisfies:

$$d_E(x, x')^2 = \sum_{i=1}^{d-1} (x_i - x_i')^2 + (x_d - x_d')^2.$$

Thus, in practise, a distance matrix can be calculated once and updated recursively to reflect the distances in the current feature subset (Wang *et al.* [74]).

A major challenge in the FS literature is the algorithmic instability caused when the number of features far exceeds the number of observations. Ensemble kNN wrapper-based methods are believed to be able to tackle instability issues (Li *et al.* [75], Park and Kim [76]).

An alternatinve FS approach for kNN, is proposed by Xiao and Chaovalitwongse [77]. They showed that if the kNN rule is based on the distance to the centroids of each class, then the FS problem can be cast as a convex optimisation problem. The idea is to learn a Mahalanobis matrix and perform l_1 regularization to promote sparsity for the Mahalanobis matrix. The sparse matrix can effectively nullify certain dimensions of the data.

Complementary to FS, data reduction methods reduce the space requirements by reducing the number of training instances. The efficiency of the data reduction methods are often judged by the trade-off between storage requirements and classifier accuracy, and run-time complexity. A complete taxonomy, comparison, and extensive experimentation of these methods is given by Garcia *et al.* [78]. The empirical finding is that approximate query methods on the complete dataset compete in run-time performance with basic kNN search on the reduced dataset. A notable algorithm is that of Arnaiz-González *et al.* [79]. In contrast to most other methods, the proposed algorithm has linear complexity. Even though the data reduction rates and the resulting classification accuracy were not among the top performers, it is a solution for extremely large datasets. Alternatively, there are solutions in distributed frameworks, *i.e.* a MapReduce solution by Triguero *et al.* [80]. Recent publications, among others, include prototype selection for imbalanced datasets: Sisodia and Sisodia [81], prototype selection with local feature weighting: Zhang *et al.* [82], prototype selection for kNN regression: Song *et al.* [83]. The authors of [82] presented an improved k-Nearest Neighbor algorithm (IKNN_PSLFW) combining prototype selection and local feature weighting as an approach to address the drawbacks of the traditional kNN rule, such as poor efficiency and class overlapping. Experiments were conducted to examine the success of the IKNN_PSLFW, and the results displayed superior performance in comparison to kNN and other machine learning algorithms, in addition to high efficiency.

5 Nearest neighbor matching algorithms

From a historical perspective, the problem of identifying nearest neighbors was posed independently by many researchers. For example, Minsky and Papert posed the 'best match' problem [84], a special case of kNN where the dataset is composed of binary words of fixed length[4], and Knuth named the 'post office search' problem, again a special case where the dataset consists of points in the plane [86]. The introduction of kNN methods for density estimation and classification drew significant attention to these problems with the aim of generalizing to high-dimensional spaces, general distance functions, and finding efficient query algorithms. A query algorithm is composed of two parts: an algorithm that maps the data into mathematical objects that can be searched effectively (preprocessing), and, a searching algorithm. Thus, performance can be assessed in terms of three components:[-0.25cm]

(a) the time complexity of the preprocessing step (computation cost),
(b) the space complexity of the mathematical objects produced by preprocessing and
(c) the time complexity of the searching method (query complexity).

When $d = 1$ the optimal strategy involves sorting the points for a $\mathcal{O}(n \log n)$ preprocessing time, and performing the query with binary search, resulting in $\mathcal{O}(\log n)$ query time for the 1NN case. As shown by Shamos [87] in 1975, the same procedure can be followed for $d = 2$ with $\mathcal{O}(n^2)$ space, only this time binary search is performed on slices of the Voronoi diagram of the data. For a dataset of n points, the Voronoi diagram consists of n distinct regions, each containing exactly one point. For metrics satisfying the triangle inequality, the Voronoi diagram partition $\mathbb{R}^d$ into convex regions, sharing only their common boundaries of equidistant points. The regions are defined such that if a query falls within a Voronoi region, its closest neighbor is the associated data point. As shown by Chew and Dyrsdale III [88] in 1985, violating the triangle inequality results in non-convex regions, which cannot be searched with the same efficiency. The first tree structure approach for $d > 2$ was the quad tree proposed by Finkel and Bentley in 1974 [89], however, the prevailing data structure is the kd-tree (Friedman *et al.* [90]) due to logarithmic (expected) query complexity and linear space complexity. Several authors have modified kd-trees for better performance. Beygelzimer *et al.* [91] proposed cover-trees, Silpa-Anan and Hartley [92] proposed optimised kd-trees. Nister and Stewenius [93] proposed the vocabulary tree, which uses hierarchical k-means. Ball-trees partition the manifold the points are on. Ball-trees is an example of 'multidimensional' tree-structures. The decision at each node does not depend solely on a single dimension of the dataset. The method is shown to perform much better in higher dimensions that other binary tree-structures. In a comparison by Muja and Lowe [94] it was shown that the multiple randomized trees are the most effective for high dimensional data. However, there is evidence that these partitioning methods scale poorly with the number of dimensions (see Indyk [95]).

[4] For this particular case, an optimal algorithm has been proven by Ronald [85] in 1974.

For many data intensive applications, there is no known exact nearest-neighbor search algorithm with acceptable efficiency. In practise, most applications settle for approximate search. A simple and popular hashing method is the locality sensitive hashing (LSH), Indyk and Motwani [96]. The idea is to encode data points into hashes. Hashes are designed so that the probability of two points sharing a hash is much higher for points that are closer together. Given some distance function d, Indyk and Motwani define a family of hash functions $\mathcal{H}$ to be $(r1, r2, p1, p2)$-sensitive if for any two points p, q in the dataset the following conditions are satisfied:[-0.2cm]

1. if $p \in B(q, r_1)$, then $\Pr_{h\in\mathcal{H}}[h(p) = h(q)] \geqslant p_1$,[-0.1cm]
2. if $p \notin B(q, r_2)$, then $\Pr_{h\in\mathcal{H}}[h(p) = h(q)] \leqslant p_2$,

where $B(p, r)$ denotes the hypersphere centered at point p with radius r. A hash family is meaningful if $p_1 > p_2$ and $r_1 < r_2$. The gap between p_1 and p_2 can be amplified by concatenating several hash functions. Practically, a collection of hashes split the data points into several partitions. To process a query, brute force search is applied to elements with the same hash values. LSH is grounded in the theory of random projections. In the simplest case, points are projected on random lines passing through the origin. These lines are then discretized into small line segments each with a corresponding id. Thus, a table of hash codes is created, with pointers to the points corresponding to each hash. Hashing methods result in fast and effective queries. Query times achieved are sublinear. In general, the quality of the hash functions determine the quality of the method (for details see Muja and Lowe [97]). However, until recently, space partitions produced by random projections had been widely studied in the average case. In the last decade a lot of work has been dedicated in studying worst case scenarios. If a dataset consists of points sparsely distributed around the origin, the probability of collision of far away points is small. On the contrary, points in dense clusters are more likely to collide with far away points. For these reasons, recent methods take into account the distribution of the data locally, for details see He *et al.* [98], Xu *et al.* [99], Iwamura *et al.* [100], Andoni and Razenshteyn [101]. The literature on hashing methods is growing fast and we direct the reader to a recent survey by Wang *et al.* [102].

In other lines of work, Jegou *et al.* [103] proposed a product quantization approach in which the feature space is decomposed into low dimensional subspaces and the data points are represented by compact codes. Babenko and Lempitsky [104] proposed the inverted multi-index, obtained by replacing standard quantization with product quantization. A more in-depth analysis is given in review papers by Vasuki and Vanathi [105], and a more recent by Wu and Yu [106].

Another family of methods is graph-based. These are fundamentally different from tree methods. Instead of reccursively cutting the dataset, these methods build general graphs in which vertices associate to data points (or subsets of data points). The query is an effective exploration of the graph(s). Empirical results place graph methods in the current state of the art. We direct the reader to recent surveys by Wang *et al.* [107] and Shimomura *et al.* [108].

Concluding, a lot of work has been done for decentralized framework solutions. Chatzimilioudis *et al.* [109] developed Spitfire, a high performance distributed algorithm. Gieseke *et al.* [110] presented a GPU based algorithm for *kd*-trees. Kim *et al.* [111] propose parallel *k*NN using MapReduce. Maillo *et al.* [112] also provided a solution for exact *k*-nearest neighbor classification based on Spark.

6 Available software

In this section, we provide a selection of available Nearest Neighbor software implementations in no particular order. These software packages and tools are open-source and available for free use, originating from industry, academia, and individuals. The purpose of this section is to provide readers with a list of available software for their use and reference. We conclude this section with a table containing software packages, their publication reference, and citation count (Table 2).

6.1 Annoy

Approximate Nearest Neighbors Oh Yeah (Annoy)[5] is a C++ library with Python bindings that was initially developed in 2013 during Spotify's Hack Week by Erik Bernhardsson. This library is currently used for music recommendations by Spotify. Annoy distinguishes itself from other nearest neighbor tools by allowing the use of static files as indexes, which enables parallel and distributed processing by sharing memory and indexes across different processes. Annoy supports various distance measures, including Euclidean, Manhattan, Cosine, Hamming, and Dot (Inner) Product distance [113]. It is memory-efficient, and can build indexes on disk for big datasets that will not fit into memory. The developers claim that Annoy performs better on datasets with less than a hundred dimensions, but it is still effective on datasets with up to a thousand dimensions.

While, the construction of indexes is separate from the lookup procedure, it is not possible to add more items to the tree after its creation. Annoy builds a tree structure using a LSH algorithm that is based on random projections. The algorithm randomly selects a hyperplane that divides the space into two sub-spaces at each intermediate node of the tree. This process is repeated n_trees times, resulting in a forest of trees. Annoy has two main parameters: n_trees and search_k. The first parameter specifies the number of trees in the index, and it affects the build time and the index size. Larger values of this parameter will return more accurate results, but larger indexes. The developers recommend setting n_trees to the largest possible value based on the available computer memory. The second parameter, search_k, determines the number of nodes to inspect during searching, and affects the search performance. Setting higher values improve the accuracy of the results but require more run time. By default, Annoy sets search_k to n*n_trees, where n is the number of approximate nearest neighbors. The developers advise setting search_k to the largest possible value that meets the time constraints of the queries. These two parameters are mostly independent.

Annoy is available on C++ and wraps for Python 2 and 3 and other programming languages such as R, Java, Scala, Ruby, Go, Lua, Node, Scala and Rust.

6.2 FAISS

Facebook AI Similarity Search (FAISS)[6] [114] is a library developed by Facebook AI Research for efficient similarity search and clustering. It includes both exact and approximate k-nearest neighbor searching, and offers a range of features to improve speed and memory usage. For example, FAISS can batch process multiple vectors, store indexes on

5 Github Annoy repository.

6 Github FAISS repository.

disk instead of RAM to enable processing of large datasets, and trade precision for speed or memory usage.

FAISS supports L_2 (Euclidean) and Dot Product vector comparison. However, a number of the available indexes support the L_1,L_∞, L_p, Canberra, BrayCurtis and Jensen-Shannon metrics [113]. Furthermore, Cosine similarity and Mahalanobis distance can be used by first preprocessing the data. Currently, FAISS offers 10 different indexes that are tailored for high-dimensional and large datasets. However, it is not recommended for low-dimensional data. FAISS also includes auto parameter tuning for specifically run time parameters.

FAISS is written in C++ with complete wrappers for Python. Lastly, FAISS provides multiprocessing on CPU and GPU level.

6.3 SPTAG

Space Partition Tree and Graph (SPTAG) [115] is a library developed by Microsoft Research and Microsoft Bing for large scale approximate k-nearest neighbor searching. SPTAG supports only L_2 (Euclidean) distance and Cosine distance, and the supported indexes are K-D trees [116] (SPTAG-KDT) and balanced k-means trees [117] (SPTAG-BKT). K-D trees have a lower index building cost, but k-means trees have higher search accuracy in very high-dimensional datasets. Both are combined with relative neighborhood graphs. SPTAG offers more than 10 parameters for parameter setting, regardless of the index used. Tuning of these parameters can be automated through Microsoft's Neural Network Intelligence (NNI) [7] tool.

SPTAG supports GPU and distributed processing (*e.g.*, performing searches over multiple machines). SPTAG is written in C++, but is also available in Python 3, Java, and C#.

6.4 NGT

Neighborhood Graph and Tree for Indexing High-dimensional Data (NGT)[8] is a library for approximate nearest neighbor searching in high dimensional vector spaces against a large volume of data. It was developed by Yahoo! JAPAN and is used by the search engine Vald(NGT)[9]. It employs one of the following three searching methods: NGT, a graph and tree-based method; QG, a quantized graph-based method; and QBG, a quantized blob graph-based method.

The NGT method can employ both tree-based and graph-based indexes. Graph-based indexes are divided into two categories: Approximate k-Nearest Neighbor Graphs (*i.e.*, ANNG [118], ANNGT[119, 120], PANNG [121, 122]) and Optimized Nearest Neighbors Graph (ONNG [123]). NGT supports several distances and similarities, including Manhattan, Euclidean, Angular, Hamming, Jaccard, Poincare, Lorentz distances, and the Cosine Similarity [113]. On the other side, QG supports only the Euclidean distance and the Cosine Similarity, while QBG supports only the Euclidean distance.

According to the developers, QG performs better than NGT, while QBG can handle datasets with billions of objects. However, NGT supports distributed processing, shared memory and batch processing making it more appropriate for server use. All three methods are available in Linux and macOS operating systems. NGT is compatible with Python, Ruby,

7 NNI tool

8 Github NGT repository.

9 Vald search engine

Rust, Javascript, NodeJS, Go, C, and C++, while QB and QBG support only C and C++. Searching can be done in Python for all three methods.

6.5 Scikit-learn

Scikit-learn[10] [124] is a widely-used machine learning library in Python, which offers several exact k-nearest neighbor methods for search, classification, regression, and component analysis, among others. Scikit-learn implements these functionalities using three nearest neighbor algorithms: Brute Force searching, kd-tree, and Ball-tree data structures [125]. While the user can choose the algorithm, Scikit-learn can also automatically select an appropriate algorithm based on various problem conditions, such as the number of samples, dimensionality of the dataset, intrinsic dimensionality and sparsity of the dataset, number of neighbors k, number of query points, and leaf size of the trees. Specifically, the Brute Force algorithm is picked if any of the following conditions apply: the input data is sparse, the 'precomputed' metric is used, the number of features exceeds 15, the value of k is greater or equal than half of the number of samples, and if the effective_metric_ is not part of the VALID_METRICS for either 'kd_tree' or 'ball_tree'. Otherwise, the first algorithm out of 'kd_tree' and 'ball_tree' with an effective_metric_ in its VALID_METRICS list is selected. However, this heuristic is based on the following assumptions: the number of query points being at least the same order as the number of training points, the leaf_size being close to its default value of 30, and the intrinsic dimensionality of the data being generally too high for tree-based methods if the number of features is greater than 15.

Scikit-learn supports a variety of distance metrics, which can be found in its DistanceMetric class. These include metrics intended for real, integer, and Boolean valued vector spaces, as well as support for user-defined distances. Furthermore, Scikit-learn provides support for CPU-based multiprocessing for the classification, regression, and transformation tasks.

6.6 SciPy

SciPy[11] [126] is a Python library that focuses on scientific computing, including modules for optimization, linear algebra, differential equations, and integration. One of its packages, called spatial, contains algorithms for spatial structures such as kd-trees. The method 'query' of the provided KDTree class returns either the approximate or exact k-nearest neighbors. SciPy provides only the Minkowski p-norms [113], but the user can select which one to use through the parameter p of the method. Moreover, CPU-based parallel processing is available via the 'workers' parameter. Although written in Fortran, C, and C++, SciPy is available in Python.

6.7 WEKA

Waikato Environment for Knowledge Analysis (Weka)[12] [127] is a user-friendly data analysis and machine learning tool that comes with a graphical user interface (GUI), making it easily accessible to users without programming knowledge. It includes algorithms and data structures for exact nearest neighbor search, such as Ball-tree, CoverTree [91], kd-tree and

[10] Scikit-learn website

[11] Scipy website

[12] Weka website

Table 2 *k*NN Software, where "Ref." indicates the Reference, "TNC" stands for total number of citation, while "CpY" indicates the citations per year

Ref.	Author(s)	Year	TNC	CpY	URL
[130]	Jayaram Subramanya *et al.*	2019	62	15.5	link
[114]	Johnson *et al.*	2019	2153	538.3	link
[131]	Chen *et al.*	2018	30	6.0	link
[132]	Curtin *et al.*	2018	77	15.4	link
[133]	Malkov and Yashunin	2018	825	165.0	link
[134]	Muja and Lowe	2014	1582	175.8	link
[135]	Boytsov and Naidan	2013	118	11.8	link
[124]	Pedregosa *et al.*	2011	72278	6023.2	link
[127]	Hall *et al.*	2009	24445	1746.1	link
[129]	Arya *et al.*	1998	3759	150.4	link

LinearNNSearch (*i.e.*, Brute Force search). Weka also has k-nearest neighbor classification functionalities, which can be found in its Instance Based knowledge (IBk) class, the IBkLG package which extends the original IBk class, and the NNge package. WEKA was developed at the University of Waikato, New Zealand, and is fully implemented in Java.

6.8 ANN

ANN[13] [128] is a C++ library that features data structures and algorithms for both exact and approximate k-nearest neighbor searching. For approximate searching, the user must specify an approximation factor $\epsilon \geq 0$. By default, ANN uses the Euclidean distance, but it can be modified to support any of the other Minkowski distance metrics, including the L_1 (Manhattan) and the L_∞ (Max) metrics. However, ANN does not support other types of similarity or distance measures, such as the Cosine similarity, and modifying ANN's structure to support these measures is challenging. For high-dimensional data, brute-force searching is preferred. ANN supports kd-Trees and Box-decomposition trees (bd-trees) [129] data structures. Both of the trees can be modified and tailored to a specific problem. ANN provides two methods to search both trees: standard search and priority search. The first, visits the tree nodes based on the hierarchy of the search tree structure. The latter, visits tree nodes in increasing order of distance from the query point, but it has a higher overhead since it uses a heap. Standard search is preferred when the error bound ϵ is small while priority search seems superior when the error bound has larger value or early termination is used (an upper limit to the number of points that will be searched before termination).

ANN also includes two programs: ann_test, and ann2fig. The first, facilitates the generation of data and query sets and provides performance statistics about the conducted experiments. Ann2fig has the ability to illustrate the data structure in a simple graphics format through a dumped data structure file.

[13] ANN website

7 Available benchmarks

This section presents several freely available and open-source benchmarking tools that have been developed to address the need for standardized, fair and trustworthy comparisons of algorithm implementations. These benchmarks facilitate easy and reliable comparison of different algorithm implementations and enable researches to identify the most promising solution to specific problems. Moreover, these benchmarks can be used to improve upon automatic parameter tuning. It is important to note that benchmarks compare the code implementations of algorithms rather than the algorithms themselves [136]. However, by comparing multiple implementations, one may infer about the nature of algorithms.

7.1 ANN-benchmarks

ANN-Benchmarks[14] [137] is a nearest neighbor specific benchmarking tool which has been widely used to benchmark approximate nearest neighbor algorithm implementations such as Annoy, FAISS, SPTAG, and NGT. This tool allows for automatic testing of a range of parameter settings and its results can be visualized as Python's matplotlib plots, LaTeX plots, or interactive plots on a website. The effectiveness of the implementations is measured using quality and performance measures. Quality measures include a distance-based definition of the recall measure and $(1 + \epsilon)$-approximate recall which use the distance of the k-th true nearest neighbor as threshold distance. Performance measures are split into two categories: preprocessing step performance and query implementation performance. The former is measured by the index size in kB of the data structure after completion and its build time in seconds. The latter is measured by the number of distance computations needed, and the time to execute and retrieve query results. Performance and quality measurements require a programmatic interface for the construction of data structures and query execution. While a Python interface is recommended, a text-based protocol whose overhead is not entirely negligible is also available. ANN-Benchmarks is installed via a Docker build file, and by default, it uses only one thread on a single CPU to avoid unfair advantages of multi-threaded implementations. However, the tool provides a batch mode that makes the entirety of the system resources available to the experiment. is worth noting that ANN-Benchmarks is aimed at in-memory nearest neighbor algorithms.

An evaluation of 15 algorithm implementations on 7 datasets for many different parameter configurations was conducted. The implemented algorithms belonged to one of the three groups: graph-based, tree-based, or hashing-based algorithms. The results suggest that graph-based algorithms have the highest number of queries per second, with HNSW [133, 135] and ONNG being the fastest, with ONNG being more robust if there is no global structure in the dataset. However, graph-based algorithms require a high preprocessing time to build their data structures, which is not ideal for regularly changing datasets. In this case, a small and quick-to build index data structures like FAISS' inverted file index is preferred. For applications that prioritize high recall values, then generally speaking, a larger index will yield better results than a smaller index and is more robust to the choice of query parameters. Lastly, exact nearest neighbor searching algorithms seem to be a viable option for lower-dimensional datasets.

[14] ANN-Benchmarks Github repository.

A billion-scale dataset version of ANN-Benchmarks, Big-ANN-Benchmarks[15] [138], was introduced as a challenge in NeurIPS 2021. Big-ANN-Benchmarks consisted of three different tracks. Track T1 used FAISS as baseline and evaluated implementations on an Azure Virtual Machine with limited DRAM, which usually is a bottleneck in serving billion-scale indices. The winning implementation of this track was: KST-ANN-T1. In track T2, competing implementations had access to SSDs in addition to the limited DRAM, and the baseline used was DiskANN[16] [130]. The winning implementation of this track was Block-based Approximate Nearest Neighbor (BBAnn)[17]. In track T3, any hardware configurations were allowed, and FAISS was used as the baseline. The two co-winners of this track were: Intel's OptaNNE GraphNN[18] and Nvidia's CUANNS MultiGPU[19].

7.2 NNS benchmark

Nearest Neighbor Search (NNS) Benchmark[20] [139] is another nearest neighbor benchmark that specializes in the Euclidean distance. In their original work, the authors evaluated the performance of 15 state-of-the-art algorithms from various scientific domains and their newly proposed algorithm. They used 18 real-world datasets, which included image, audio, video, and textual data, as well as 2 synthetic datasets. The authors employed the speedup of Algorithm A as a performance evaluation metric, which is defined as t_{BF}/t_A, where t_x is the average search time of Algorithm x, and t_{BF} is the query time of a brute-force (BF) linear scan algorithm. They used the standard recall measure to assess the quality of the implementations. Both measures were averaged over all queries in the query workload.

Most of the algorithms are implemented in C++, and all their implementations were modified to disable hardware-specific optimizations and other implementation tricks for fairer comparisons. The experimental evaluation consisted of two rounds. In the first round, the authors compared implementations of algorithms that belonged to the same category. These categories were LSH-based methods, encoding-based methods, tree-based space partition methods, and neighborhood-based methods. Then, representatives of each category were chosen based on their performance. SRS [140] was picked as the representative of the LSH-based methods. For the encoding-based category, OPQ [141] was chosen. Annoy and FLANN [134] were the representatives of the tree-based space partition methods. Lastly, KGraph[21] and HNSW represented the neighborhood-based methods. Their newly proposed algorithm Diversified Proximity Graph (DPG) [139], was also part of the second round.

In the second round of evaluation, these representatives were compared, and the following conclusions were made. Supposing sufficient computing resources for the off-line index construction and main memory to store that index, DPG and HNSW seem to be the best choices for approximate nearest neighbor searching on high dimensional data. Annoy is also recommended for its better trade-off between search performance and index size, as well as its construction time. This means that search performance is not drastically reduced by lowering the number of trees. In the case of large-scale datasets with moderate computing resources,

[15] big-ANN-Benchmarks Github repository

[16] Github DiskANN repository

[17] BBBAnn Github repository

[18] GraphNN Python Implementation

[19] CUANNS Python file

[20] NNS benchmark Github repository

[21] KGraph Github Repository

OPQ and SRS are recommended due to their small index sizes and construction time. Moreover, SRS has a theoretical guarantee of searching quality and handles data updates easily. Lastly, the authors did a further analysis of the features of each representative algorithm.

7.3 Other machine learning benchmarks

Other machine learning benchmarks that do not specifically cater to nearest neighbor techniques include among others: Mlpack [132], OpenML benchmarking suites [142], and the ShinyLearner benchmarking tool [143]. While these tools offer similar functionalities such as automation of some parts of experiments, each has unique features.

Mlpack in conjunction with Valgrind [144] can display memory usage over time for a particular implementation. The OpenML benchmarking suites, such as OpenML-CC18 [142] and its predecessor OpenML100 [145], are sets of OpenML tasks designed to evaluate algorithms under a specific set of conditions. Numerous suites, published along with experiment results and configurations, are available on the OpenML platform[22] [146]. Lastly, ShinyLearner[23] is a classification only benchmarking tool that does not require programming but rather a set of commands. These sets of commands can also be generated through a Web-based tool[24].

8 Synopsis and concluding remarks

We have conducted a review of kNN methods. The kNN approach is typically applied as a classification algorithm based on the assumption that similar points can be found near one another.

We have contributed to the large literature of kNN methods by reviewing a diverse range of material related to kNN, from theoretical work in non-parametric discrimination, decision rules and kNN related methods, matching algorithms, as well as a comprehensive discussion of benchmarks and available software together with citation analysis.

The kNN classifier is a transparent method with great intuitive appeal. It is part of the family of instance based algorithms, and it makes no assumptions about the underlying data distribution, hence non-parametric. This trait is crucial because real world data rarely obey typical theoretical assumptions. The work of Cover and Hart [2] peaked interest in the kNN classifiers and ignited a lot of research in nearest-neighbor search algorithms. We defined the four pillars of a good kNN applications to be:

(1) the choice of hyperparameter k,
(2) the choice of classification rule,
(3) the choice of distance function and
(4) the choice of query algorithm.

This opinion is based on both practical and theoretical considerations. The theory of non-parametric estimation and discrimination unveils a relationship between the locally optimal choice of k and the convexity of the probability density function. What is more, deviation from the (optimal) Bayes rule can be improved with a proper choice of distance metric. These

[22] OpenML website

[23] ShinyLearner Github repository.

[24] ShinyLearner web tool.

Table 3 *k*NN citations, where "Ref." indicates the Reference, "TNC" stands for total number of citations, while "CpY" indicates the citations per year

Ref.	Author(s)	Year	TNC	CpY
[70]	Deng *et al.*	2023	0	0
[69]	Wang *et al.*	2023	1	1
[26]	Uddin *et al.*	2022	51	51
[82]	Zhang *et al.*	2022	9	9
[24]	Agarwal and Poornalatha	2021	7	3.5
[147]	Cunningham and Delany	2021	1035	517.5
[14]	Dwibedi *et al.*	2021	204	102.0
[23]	Sun and Chen	2021	9	4.5
[68]	Susan and Kumar	2021	8	4.0
[33]	Yuan *et al.*	2021	9	4.5
[148]	Shaban *et al.*	2020	161	53.7
[21]	Taunk *et al.*	2019	228	57
[66]	Zhang *et al.*	2018	151	30.2
[112]	Maillo *et al.*	2017	319	53.2
[56]	Zhang *et al.*	2017	864	144.0
[55]	Zhang *et al.*	2017	511	85.2
[36]	Zhang *et al.*	2017	87	14.5
[20]	Adeniyi *et al.*	2016	455	65.0
[44]	Ando	2016	26	3.7
[18]	Beretta and Santaniello	2016	463	66.1
[149]	Deng *et al.*	2016	506	72.3
[57]	Hu *et al.*	2016	452	64.6
[150]	Miao *et al.*	2016	26	3.7
[101]	Andoni and Razenshteyn	2015	311	38.9
[151]	Begum *et al.*	2015	47	5.9
[54]	GarcÃa-Pedrajas *et al.*	2015	109	13.6
[32]	Tang and He	2015	117	14.6
[74]	Wang *et al.*	2015	156	19.5
[77]	Xiao and Chaovalitwongse	2015	33	4.1
[45]	Yu *et al.*	2015	107	13.4
[104]	Babenko and Lempitsky	2014	421	46.8
[63]	Derrac *et al.*	2014	120	13.3
[134]	Muja and Lowe	2014	1582	175.8
[152]	Van Hulse and Khoshgoftaar	2014	99	11.0
[48]	Dubey and Pudi	2013	67	6.7
[153]	Eirola *et al.*	2013	55	5.5
[154]	Guo *et al.*	2013	1301	130.1
[155]	Imandoust *et al.*	2013	560	56.0
[100]	Iwamura *et al.*	2013	55	5.5
[156]	Jin *et al.*	2013	233	23.3
[22]	Kataria and Singh	2013	197	19.7
[78]	Garcia *et al.*	2012	749	68.1
[64]	Gou *et al.*	2012	260	23.6

Table 3 continued

Ref.	Author(s)	Year	TNC	CpY
[75]	Li *et al.*	2011	149	12.4
[51]	Liu and Chawla	2011	195	16.3
[157]	Triguero *et al.*	2011	145	12.1
[99]	Xu *et al.*	2011	189	15.8
[98]	He *et al.*	2010	171	13.2
[103]	Jegou *et al.*	2010	3705	285.0
[25]	Ting *et al.*	2010	72	5.54
[13]	Boiman *et al.*	2009	1538	109.9
[158]	Fayed and Atiya	2009	170	12.1
[159]	Kamath and Mahato	2009	13	0.9
[94]	Muja and Lowe	2009	3838	274.1
[61]	Weinberger and Saul	2009	4469	319.2
[160]	Wong *et al.*	2009	987	70.5
[92]	Silpa-Anan and Hartley	2008	921	61.4
[73]	Tahir *et al.*	2007	260	16.3
[91]	Beygelzimer *et al.*	2006	1061	62.4
[93]	Nister and Stewenius	2006	4922	289.5
[161]	Wang *et al.*	2006	131	8.2
[162]	Sfetsos and Siriopoulos	2004	83	4.4
[95]	Indyk	2004	246	12.9
[67]	He and Niyogi	2003	5307	265.4
[129]	Arya *et al.*	1998	3759	150.5
[96]	Indyk and Motwani	1998	5619	224.8
[163]	Wettschereck and Dietterich	1993	81	2.7
[164]	Yianilos	1993	1639	54.6
[11]	Altman	1992	6004	193.7
[27]	Fix and Hodges	1989	3481	102.4
[10]	Aha, Kibler, *et al.*	7213	1991	232.7
[30]	Loizou and Maybank	1987	38	1.1
[90]	Friedman *et al.*	1977	3964	86.2
[65]	Dudani	1976	1779	37.9
[31]	Fukunaga and Hostetler	1973	240	4.8
[29]	Hellman	1973	275	5.5

relationships can be inferred from Fukanaga and Hostetler's closed-form approximation of the locally optimal k value [31]. Their result applies to the kNN density estimate, but, it forms a basis for understanding many of the kNN variations. Many variants of kNN classifiers have not received the same theoretical attention, however, the classification accuracy has been raised considerably in practice. For those that hold Bayesian estimation to a golden standard, the kNN density estimator approximates the optimal Bayesian rule asymptotically, which also implies that a sufficiently large dataset is required to achieve reliable predictions. However, variations of kNN have shown good resilience in the presence of overlapping classes, which is the fundamental source of error in Bayesian thinking, and in class imbalances. Moreover, modern kNN methods also involve a fair bit of learning. A locally optimal k value can be found with cross validation or estimated with convex optimization techniques. We note that distance metric learning methods continue to advance. All these reasons lead us to believe that kNN methods may prove competitive with the state of the art in classification.

The main consideration in kNN applications is storage and run-time complexity. Exact nearest neighbors algorithms such as tree-based methods have efficient query times but scale poorly with dimensionality. Approximate nearest neighbor methods significantly reduce query run-times in high-dimensional datasets. Hashing, quantization, graph techniques, and indexing methods all contribute to the diverse literature of query algorithms that enable the use of kNN in large data domains. GPU based algorithms and adaptations for data streams also exist. Furthermore, there is a big literature in data reduction techniques that can be applied to the kNN classifier efficiently. Wrapper methods for feature selection synergize with kNN, especially when the distance metric can be calculated recursively. Methods of prototype selection and generation can reduce the number of training instances without hindering classification accuracy greatly.

An important final remark is that kNN based methods find utility in virtually every aspect of data science. There are competitive noise reduction, outlier detection, missing value imputation and sampling methods, often paired with theoretical results. We close with Table 3 which contains publications related to kNN classification together with citation counts.

Acknowledgements The authors express their gratitude to the anonymous reviewers for their valuable feedback and insightful comments on the earlier version of the manuscript. Their constructive criticism greatly contributed to the improvement of the paper, and we are grateful for their time and effort in reviewing our work.

Declarations

The authors declare that there is no competing interest of any kind, directly or indirectly related to this work. A data availability statement is not applicable.

References

1. Fix, E., Hodges, J.L.: Discriminatory analysis, nonparametric discrimination, consistency properties. Project 21-49-004. Report No.4 USAF School of Aviation Medicine Randolph Field, Texas, USA, 1–21 (1951)
2. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory IT- **13**(1), 21–27 (1967)
3. Sylvester, J.J.: On Poncelet's approximate valuation of surd forms. Philos. Mag. **20**, 203–222 (1860)
4. Jung, H.W.E.: Ueber die kleinste Kugel, die eine ráumliche Figur einschliesst. J. Reine Angew. Math. **123**, 241–257 (1901)
5. Jung, H.W.E.: Ueber den kleinsten Kreis, der eine ebene Figur einschliesst. J. Reine Angew. Math. **137**, 310–313 (1909)
6. Blumenthal, L.M., Wahlin, G.E.: On the spherical surface of smallest radius enclosing a bounded subset of n-dimensional Euclidean space. Bull. Amer. Math. Soc. **47**, 771–777 (1941)
7. Guggenheimer, H.W.: Applicable Geometry. R. E. Krieger Publishing Co, Huntigton, New York (1977)
8. Vrahatis, M.N.: A variant of Jung's theorem. Bull. Greek Math. Soc. **29**, 1–6 (1988)
9. Vrahatis, M.N.: An error estimation for the method of bisection in Rn. Bull. Greek Math. Soc. **27**, 161–174 (1986)
10. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach, Learn (1991)
11. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
12. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., et al.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)

13. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
14. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9588–9597 (2021)
15. Bandaragoda, T.R., Ting, K.M., Albrecht, D., Liu, F.T., Wells, J.R.: Efficient anomaly detection by isolation using nearest neighbour ensemble. In: 2014 IEEE International Conference on Data Mining Workshop, pp. 698–705 (2014). IEEE
16. Pang, G., Ting, K.M., Albrecht, D.: Lesinn: Detecting anomalies by identifying least similar nearest neighbours. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 623–630 (2015). IEEE
17. Ting, K.M., Washio, T., Wells, J.R., Aryal, S.: Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors. Mach. Learn. **106**(1), 55–91 (2017)
18. Beretta, L., Santaniello, A.: Nearest neighbor imputation algorithms: a critical evaluation. BMC medical informatics and decision making **16**(3), 197–208 (2016)
19. Triguero, I., García-Gil, D., Maillo, J., Luengo, J., García, S., Herrera, F.: Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. WIREs Data Mining and Knowledge Discovery **9**(2) (2019)
20. Adeniyi, D.A., Wei, Z., Yongquan, Y.: Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method. Appl. Comput. Inform. **12**(1), 90–108 (2016)
21. Taunk, K., De, S., Verma, S., Swetapadma, A.: A brief review of nearest neighbor algorithm for learning and classification. In: 2019 International Conference on Intelligent Computing and Control Systems (ICCS), pp. 1255–1260 (2019). IEEE
22. Kataria, A., Singh, M.: A review of data classification using k-nearest neighbour algorithm. Int. J. of Emerg. Technol. Adv. Eng. **3**(6), 354–360 (2013)
23. Sun, B., Chen, H.: A survey of nearest neighbor algorithms for solving the class imbalanced problem. Wirel. Commun. Mob. Comput. **2021** (2021)
24. Agarwal, Y., Poornalatha, G.: Analysis of the nearest neighbor classifiers: a review. Advances in Artificial Intelligence and Data Engineering: Select Proceedings of AIDE **2019**, 559–570 (2021)
25. Ting, K.M., Zhou, G.-T., Liu, F.T., Tan, J.S.C.: Mass estimation and its applications. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '10, pp. 989–998. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1835804.1835929
26. Uddin, S., Haque, I., Lu, H., Moni, M.A., Gide, E.: Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction. Sci. Reports **12**(1), 1–11 (2022)
27. Fix, E., Hodges, J.L.: Discriminatory analysis. nonparametric discrimination: Consistency properties. Int. Stat. Rev./Rev. Int. de Stat. **57**(3), 238–247 (1989)
28. Welch, B.L.: Note on discriminant functions. Biometrika **31**(1/2), 218–220 (1939)
29. Hellman, M.E.: The nearest neighbor classification rule with a reject option. IEEE Trans. Syst. Sci. Cybern. **6**(3), 179–185 (1970)
30. Loizou, G., Maybank, S.J.: The nearest neighbor and the bayes error rates. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI- **9**(2), 254–262 (1987)
31. Fukunaga, K., Hostetler, L.: Optimization of k nearest neighbor density estimates. IEEE Trans. Inf. Theory **19**(3), 320–326 (1973)
32. Tang, B., He, H.: Enn: Extended nearest neighbor method for pattern recognition [research frontier]. IEEE Comput. Intell. Mag. **10**(3), 52–60 (2015)
33. Yuan, B.-W., Luo, X.-G., Zhang, Z.-L., Yu, Y., Huo, H.-W., Johannes, T., Zou, X.-D.: A novel density-based adaptive k nearest neighbor method for dealing with overlapping problem in imbalanced datasets. Neural Comput. Appl. **33**(9), 4457–4481 (2021)
34. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl, Data Eng (2009)
35. Fernández, A., del Río, S., Chawla, N.V., Herrera, F.: An insight into imbalanced big data classification: outcomes and challenges. Complex & Intell. Syst. **3**(2), 105–120 (2017)
36. Zhang, X., Li, Y., Kotagiri, R., Wu, L., Tari, Z., Cheriet, M.: Krnn: k rare-class nearest neighbour classification. Pattern Recognit. **62**, 33–44 (2017)
37. Zhang, S.: Challenges in knn classification. IEEE Trans. Knowl. Data Eng. **34**(10), 4663–4675 (2022). https://doi.org/10.1109/TKDE.2021.3049250
38. Zeraatkar, S., Afsari, F.: Interval-valued fuzzy and intuitionistic fuzzy-knn for imbalanced data classification. Pattern Recogn. Appl. **184**, 115510 (2021)

39. Wang, Z., Li, Y., Li, D., Zhu, Z., Du, W.: Entropy and gravitation based dynamic radius nearest neighbor classification for imbalanced problem. Knowl.-Based Syst. **193**, 105474 (2020)
40. Patel, H., Thakur, G.S.: Classification of imbalanced data using a modified fuzzy-neighbor weighted approach. Int. J. Intell. Eng. Syst. **10**(1), 56–64 (2017)
41. Liu, S., Zhang, J., Xiang, Y., Zhou, W.: Fuzzy-based information decomposition for incomplete and imbalanced data learning. IEEE Trans. Fuzzy Syst. **25**(6), 1476–1490 (2017)
42. Li, Y., Zhang, X.: Improving k nearest neighbor with exemplar generalization for imbalanced classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 321–332 (2011). Springer
43. Nikpour, B., Shabani, M., Nezamabadi-pour, H.: Proposing new method to improve gravitational fixed nearest neighbor algorithm for imbalanced data classification. In: 2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), pp. 6–11 (2017). IEEE
44. Ando, S.: Classifying imbalanced data in distance-based feature space. Knowl. Inf. Syst. **46** (2016)
45. Yu, Z., Chen, H., Liu, J., You, J., Leung, H., Han, G.: Hybrid k-nearest neighbor classifier. IEEE Trans. Cybern. **46**(6), 1263–1275 (2015)
46. Zhu, Y., Wang, Z., Gao, D.: Gravitational fixed radius nearest neighbor for imbalanced problem. Knowl.-Based Syst. **90**, 224–238 (2015)
47. Hajizadeh, Z., Taheri, M., Jahromi, M.Z.: Nearest neighbor classification with locally weighted distance for imbalanced data. Int. J. Comput. Commun. Eng. **3**(2), 81 (2014)
48. Dubey, H., Pudi, V.: Class based weighted k-nearest neighbor over imbalance dataset. In: Advances in Knowledge Discovery and Data Mining, pp. 305–316. Springer, Berlin, Heidelberg (2013)
49. Zhang, X., Li, Y.: A positive-biased nearest neighbour algorithm for imbalanced classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 293–304 (2013). Springer
50. Kriminger, E., Prόncipe, J.C., Lakshminarayan, C.: Nearest neighbor distributions for imbalanced classification. In: The 2012 International Joint Conference on Neural Networks (IJCNN) pp. 1–5 (2012). IEEE
51. Liu, W., Chawla, S.: Class confidence weighted knn algorithms for imbalanced data sets. In: Advances in Knowledge Discovery and Data Mining, pp. 345–356. Springer, Berlin, Heidelberg (2011)
52. Song, Y., Huang, J., Zhou, D., Zha, H., Giles, C.L.: Iknn: Informative k-nearest neighbor pattern classification. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 248–264 2007. Springer
53. Abu Alfeilat, H., Hassanat, A., Lasassmeh, O., Tarawneh, A., Alhasanat, M., Eyal-Salman, H., Prasath, S.: Effects of distance measure choice on K-nearest neighbor classifier performance: A review. Big Data 7 (2019)
54. García-Pedrajas, N., Romero del Castillo, J.A., Cerruela-García, G.: A proposal for local k values for k -nearest neighbor rule. IEEE Trans. Neural Netw. Learn. Syst. **28**(2), 470–475 (2017)
55. Zhang, S., Li, X., Zong, M., Zhu, X., Cheng, D.: Learning k for KNN classification. ACM Trans. Intell. Syst. Technol. (TIST) **8**(3), 1–19 (2017)
56. Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R.: Efficient knn classification with different numbers of nearest neighbors. IEEE Trans. Neural Netw. Learn. Syst. **29**(5), 1774–1785 (2017)
57. Hu, L.-Y., Huang, M.-W., Ke, S.-W., Tsai, C.-F.: The distance function effect on k-nearest neighbor classification for medical datasets. SpringerPlus 5 (2016)
58. Xing, E., Jordan, M., Russell, S.J., Ng, A.: Distance metric learning with application to clustering with side-information. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge, MA (2002)
59. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proceedings of the Twenty-First International Conference on Machine Learning. ICML '04, p. 94. Association for Computing Machinery, New York, NY, USA (2004)
60. Goldberger, J., Hinton, G.E., Roweis, S., Salakhutdinov, R.R.: Neighbourhood components analysis. In: Saul, L., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17. MIT Press, Cambridge, MA (2004)
61. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. (2005)
62. Vincent, P., Bengio, Y.: K-local hyperplane and convex distance nearest neighbor algorithms. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14. MIT Press, Cambridge, MA (2001)
63. Derrac, J., García, S., Herrera, F.: Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. Information Sciences **260**, 98–119 (2014)
64. Gou, J., Du, L., Zhang, Y., Xiong, T.: A new distance-weighted k-nearest neighbor classifier. J. Inf, Comput. Sci. **9**, 1429–1436 (2012)

65. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst., Man, Cybern. SMC-**6**(4), 325–327 (1976)
66. Zhang, S., Cheng, D., Deng, Z., Zong, M., Deng, X.: A novel knn algorithm with data-driven k parameter computation. Pattern Recognition Letters **109**, 44–54 (2018). Special Issue on Pattern Discovery from Multi-Source Data (PDMSD)
67. He, X., Niyogi, P.: Locality preserving projections. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge, MA (2003)
68. Susan S., K.A.: Dst-ml-eknn: Data space transformation with metric learning and elite k-nearest neighbor cluster formation for classification of imbalanced datasets 1133 (2021)
69. Wang, A.X., Chukova, S.S., Nguyen, B.P.: Ensemble k-nearest neighbors based on centroid displacement. Inf. Sci. **629**, 313–323 (2023)
70. Deng, S., Wang, L., Guan, S., Li, M., Wang, L.: Non-parametric nearest neighbor classification based on global variance difference. Int. J. Comput. Intell. Syst. **16**(1), 26 (2023)
71. Rogati, M., Yang, Y.: High-performing feature selection for text classification. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management, pp. 659–661 (2002)
72. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H.: Feature selection: A data perspective. ACM Compu. Surv. (CSUR) **50**(6), 1–45 (2017)
73. Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous feature selection and feature weighting using hybrid tabu search/K-nearest neighbor classifier. Pattern Recogn. Lett. **28**(4), 438–446 (2007)
74. Wang, A., An, N., Chen, G., Li, L., Alterovitz, G.: Accelerating wrapper-based feature selection with K-nearest-neighbor. Knowl.-Based Syst. 83, 81–91 (2015)
75. Li, S., Harner, E.J., Adjeroh, D.A.: Random knn feature selection-a fast and stable alternative to random forests. BMC bioinformatics **12**(1), 1–11 (2011)
76. Park, C.H., Kim, S.B.: Sequential random k-nearest neighbor feature selection for high-dimensional data. Expert Syst. Appl. **42**(5), 2336–2342 (2015)
77. Xiao, C., Chaovalitwongse, W.A.: Optimization models for feature selection of decomposed nearest neighbor. IEEE Trans. Syst., Man, Cybern.: Syst. **46**(2), 177–184 (2016)
78. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
79. Arnaiz-González, Á., Díez-Pastor, J.-F., Rodríguez, J.J., García-Osorio, C.: Instance selection of linear complexity for big data. Knowl.-Based Syst. **107**, 83–95 (2016)
80. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: Mrpr: a mapreduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
81. Sisodia, D., Sisodia, D.S.: Quad division prototype selection-based knearest neighbor classifier for click fraud detection from highly skewed user click dataset. Eng. Sci.Technol., Int. J. **28**, 101011 (2022)
82. Zhang, X., Xiao, H., Gao, R., Zhang, H., Wang, Y.: K-nearest neighbors rule combining prototype selection and local feature weighting for classification. Knowl.-Based Syst. **243**, 108451 (2022)
83. Song, Y., Liang, J., Lu, J., Zhao, X.: An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing **251**, 26–34 (2017)
84. Minsky, M., Papert, S.: An introduction to computational geometry. Cambridge tiass., HIT **479**, 480 (1969)
85. Rivest, R.L.: On the optimality of elia's algorithm for performing bestmatch searches. In: IFIP Congress, pp. 678–681 (1974)
86. Knuth, D.E., et al.: The Art of Computer Programming, vol. 3. Addison- Wesley Reading, MA (1973)
87. Shamos, M.I.: Geometric complexity. In: Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, pp. 224–233 (1975)
88. Chew, L.P., Dyrsdale III, R.L.: Voronoi diagrams based on convex distance functions. In: Proceedings of the First Annual Symposium on Computational Geometry, pp. 235–244 (1985)
89. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. Acta Informatica **4**(1), 1–9 (1974)
90. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. (TOMS) **3**(3), 209–226 (1977)
91. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104 (2006)
92. Silpa-Anan, C., Hartley, R.: Optimised kd-trees for fast image descriptor matching. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008). IEEE
93. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), **2**, pp. 2161–2168 (2006). Ieee
94. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP **2**(331–340), 2 (2009)

95. Indyk, P.: Nearest neighbors in high-dimensional spaces (2004)
96. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. Conf. Proc. Ann. ACM Symp. Theory Comput. 604–613 (2000)
97. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014)
98. He, J., Liu, W., Chang, S.-F.: Scalable similarity search with optimized kernel hashing. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1129–1138 (2010)
99. Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: Complementary hashing for approximate nearest neighbor search. In: 2011 International Conference on Computer Vision, pp. 1631–1638 (2011)
100. Iwamura, M., Sato, T., Kise, K.: What is the most efficient way to select nearest neighbor candidates for fast approximate nearest neighbor search? In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3535–3542 (2013)
101. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing. STOC '15, pp. 793–801. Association for Computing Machinery, New York, NY, USA
102. Wang, J., Zhang, T., song, j., Sebe, N., Shen, H.T.: A survey on learning to hash. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 769–790 (2018)
103. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2010)
104. Babenko, A., Lempitsky, V.: The inverted multi-index. IEEE Trans. Pattern Anal. Mach. Intell. **37**(6), 1247–1260 (2014)
105. Vasuki, A., Vanathi, P.: A review of vector quantization techniques. IEEE Potentials **25**(4), 39–47 (2006)
106. Wu, Z.-b., Yu, J.-q.: Vector quantization: a review. Front. Inf. Technol. & Electron. Eng. **20**(4), 507–524 (2019)
107. Wang, M., Xu, X., Yue, Q., Wang, Y.: A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. arXiv preprint arXiv:2101.12631 (2021)
108. Shimomura, L.C., Oyamada, R.S., Vieira, M.R., Kaster, D.S.: A survey on graph-based methods for similarity searches in metric spaces. Inf. Syst. **95**, 101507 (2021)
109. Chatzimilioudis, G., Costa, C., Zeinalipour-Yazti, D., Lee, W.-C., Pitoura, E.: Distributed in-memory processing of all k nearest neighbor queries. IEEE Trans. Knowl. Data Eng. **28**(4), 925–938 (2015)
110. Patwary, M.M.A., Satish, N.R., Sundaram, N., Liu, J., Sadowski, P., Racah, E., Byna, S., Tull, C., Bhimji, W., Dubey, P., et al.: Panda: Extreme scale parallel k-nearest neighbor on distributed architectures. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 494–503 (2016). IEEE
111. Kim, W., Kim, Y., Shim, K.: Parallel computation of k-nearest neighbor joins using mapreduce. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 696–705 (2016). IEEE
112. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: KNN-IS: An iterative spark-based design of the k-nearest neighbors classifier for big data. Knowl.-Based Syst.**117**, 3–15 (2017)
113. Deza, M.M., Deza, E.: Encyclopedia of distances. In: Encyclopedia of Distances, pp. 1–583. Springer, Heidelberg (2009)
114. Johnson, J., Douze, M., Jígou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)
115. Chen, Q., Wang, H., Li, M., Ren, G., Li, S., Zhu, J., Li, J., Liu, C., Zhang, L., Wang, J.: SPTAG: A Library for Fast Approximate Nearest Neighbor Search. (2018). https://github.com/Microsoft/SPTAG
116. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
117. Lamrous, S., Taileb, M.: Divisive hierarchical k-means. In: 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), pp. 18–18 (2006). IEEE
118. Iwasaki, M.: Proximity search in metric spaces using approximate k nearest neighbor graph. IPSJ Trans. Database **3**(1), 18–28 (2010)
119. Iwasaki, M.: Proximity search using approximate k nearest neighbor graph with a tree structured index. IPSJ J. **52**(2), 817–828 (2011)
120. Iwasaki, M.: Applying a graph-structured index to product image search. J. Inst. Image Electr. Eng. of Japan **42**(5), 633–641 (2013).https://doi.org/10.11371/iieej.42.633
121. Iwasaki, M.: Pruned bi-directed k-nearest neighbor graph for proximity search. In: SISAP (2016)
122. Sugawara, K., Kobayashi, H., Iwasaki, M.: On approximately searching for similar word embeddings. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2265–2275 (2016)

123. Iwasaki, M., Miyazaki, D.: Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. (2018). arXiv preprint arXiv:1810.07355
124. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
125. Omohundro, S.M.: Five Balltree Construction Algorithms. International Computer Science Institute Berkeley, Berkeley, CA (1989)
126. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**(3), 261–272 (2020)
127. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
128. Arya, S., Mount, D.: Ann: library for approximate nearest neighbor searching. In: Proceedings of IEEE CGC Workshop on Computational Geometry, Providence, RI (1998)
129. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM) **45**(6), 891–923 (1998)
130. Jayaram Subramanya, S., Devvrit, F., Simhadri, H.V., Krishnawamy, R., Kadekodi, R.: Diskann: Fast accurate billion-point nearest neighbor search on a single node. Adv. Neural Inf. Process. Syst. **32** (2019)
131. Chen, Q.,Wang, H., Li, M., Ren, G., Li, S., Zhu, J., Li, J., Liu, C., Zhang, L., Wang, J.: SPTAG: A library for fast approximate nearest neighbor search. GitHub. (2018) https://github.com/Microsoft/SPTAG
132. Curtin, R.R., Edel, M., Lozhnikov, M., Mentekidis, Y., Ghaisas, S., Zhang, S.: mlpack 3: a fast, flexible machine learning library. Journal of Open Source Software **3**(26), 726 (2018)
133. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2018)
134. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014)
135. Boytsov, L., Naidan, B.: Engineering efficient and effective non-metric space library. In: International Conference on Similarity Search and Applications, pp. 280–293 (2013). Springer
136. Kriegel, H.-P., Schubert, E., Zimek, A.: The (black) art of runtime evaluation: Are we comparing algorithms or implementations? Knowledge and Information Systems **52**(2), 341–378 (2017)
137. Aumüler, M., Bernhardsson, E., Faithfull, A.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In: International Conference on Similarity Search and Applications, pp. 34–49 (2017). Springer
138. Simhadri, H.V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., et al.: Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. (2022) arXiv preprint arXiv:2205.03763
139. Li, W., Zhang, Y., Sun, Y., Wang, W., Zhang, W., Lin, X.: Approximate Nearest Neighbor Search on High Dimensional Data–Experiments, Analyses, and Improvement (v1.0). (2016). arXiv arXiv:1610.02455
140. Sun, Y., Wang, W., Qin, J., Zhang, Y., Lin, X.: Srs: solving capproximate nearest neighbor queries in high dimensional euclidean space with a tiny index. Proc, VLDB Endowment (2014)
141. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization. IEEE Tans. Pattern Anal. Mach. Intell. **36**(4), 744–755 (2013)
142. Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R.G., van Rijn, J.N., Vanschoren, J.: Openml benchmarking suites. (2017). arXiv preprint arXiv:1708.03731
143. Piccolo, S.R., Lee, T.J., Suh, E., Hill, K.: Shinylearner: A containerized benchmarking tool for machine-learning classification of tabular data. GigaScience **9**(4), 026 (2020)
144. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. ACM Sigplan notices **42**(6), 89–100 (2007)
145. Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R.G., van Rijn, J.N., Vanschoren, J.: Openml benchmarking suites and the openml100. stat 1050, 11 (2017)
146. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explor. Newsl. **15**(2), 49–60 (2014)
147. Cunningham, P., Delany, S.J.: k-nearest neighbour classifiers-a tutorial. ACM Computing Surveys (CSUR) **54**(6), 1–25 (2021)
148. Shaban, W.M., Rabie, A.H., Saleh, A.I., Abo-Elsoud, M.A.: A new covid- 19 patients detection strategy (cpds) based on hybrid feature selection and enhanced knn classifier. Knowl.-Based Syst. **205**, 106270 (2020)
149. Deng, Z., Zhu, X., Cheng, D., Zong, M., Zhang, S.: Efficient knn classification algorithm for big data. Neurocomputing **195**, 143–148 (2016)

150. Miao, X., Gao, Y., Chen, G., Zheng, B., Cui, H.: Processing incomplete k nearest neighbor search. IEEE Trans. Fuzzy Syst. **24**(6), 1349–1363 (2016)
151. Begum, S., Chakraborty, D., Sarkar, R.: Data classification using feature selection and knn machine learning approach. In: 2015 International Conference on Computational Intelligence and Communication Networks (CICN), pp. 811–814 (2015)
152. Van Hulse, J., Khoshgoftaar, T.M.: Incomplete-case nearest neighbor imputation in software measurement data. Inf. Sci. **259**, 596–610 (2014)
153. Eirola, E., Doquire, G., Verleysen, M., Lendasse, A.: Distance estimation in numerical data sets with missing values. Inf. Sci. **240**, 115–128 (2013)
154. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: Knn model-based approach in classification. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pp. 986–996 (2003). Springer
155. Imandoust, S.B., Bolandraftar, M., et al.: Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. Int. J. Eng. Res. Appl. **3**(5), 605–610 (2013)
156. Jin, Z., Li, C., Lin, Y., Cai, D.: Density sensitive hashing. IEEE Trans. Cybern. **44**(8), 1362–1371 (2013)
157. Triguero, I., García, S., Herrera, F.: Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recog. **44**(4), 901–916 (2011)
158. Fayed, H.A., Atiya, A.F.: A novel template reduction approach for the knearest neighbor method. IEEE Trans. Neural Networks **20**(5), 890–896 (2009)
159. Kamath, S.D., Mahato, K.K.: Principal component analysis (pca)-based k-nearest neighbor (k-nn) analysis of colonic mucosal tissue fluorescence spectra. Photomed. Laser Surg. **27**(4), 659–668 (2009)
160. Wong, W.K., Cheung, D.W.-l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139–152 (2009)
161. Wang, J., Neskovic, P., Cooper, L.N.: Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence. Pattern Recog. **39**(3), 417–423 (2006)
162. Sfetsos, A., Siriopoulos, C.: Time series forecasting with a hybrid clustering scheme and pattern recognition. IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Hum. **34**(3), 399–405 (2004)
163. Wettschereck, D., Dietterich, T.: Locally adaptive nearest neighbor algorithms. In: Cowan, J., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems, vol. 6. Morgan-Kaufmann, Burlington, MA (1993)
164. Yianilos, P.N.: Data structures and algorithms for nearest neighbor. In: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, vol. 66, p. 311 (1993). SIAM