PeerJ Computer Science

Random k conditional nearest neighbor for high-dimensional data

Jiaxuan Lu and Hyukjun Gweon

University of Western Ontario, London, ON, Canada

ABSTRACT

The k nearest neighbor (kNN) approach is a simple and effective algorithm for classification and a number of variants have been proposed based on the kNN algorithm. One of the limitations of kNN is that the method may be less effective when data contains many noisy features due to their non-informative influence in calculating distance. Additionally, information derived from nearest neighbors may be less meaningful in high-dimensional data. To address the limitation of nearest-neighbor based approaches in high-dimensional data, we propose to extend the k conditional nearest neighbor (kCNN) method which is an effective variant of kNN. The proposed approach aggregates multiple kCNN classifiers, each constructed from a randomly sampled feature subset. We also develop a score metric to weigh individual classifiers based on the level of separation of the feature subsets. We investigate the properties of the proposed method using simulation. Moreover, the experiments on gene expression datasets show that the proposed method is promising in terms of predictive classification performance.

Subjects Data Mining and Machine Learning, Data Science

Keywords K nearest neighbor, High-dimensional data, Nonparametric classification

Submitted 23 July 2024
Accepted 18 October 2024
Published 24 January 2025

Corresponding author
Hyukjun Gweon, hgweon@uwo.ca

Academic editor
Álvar Arnaiz-González

Additional Information and Declarations can be found on page 19

DOI **10.7717/peerj-cs.2497**

INTRODUCTION

Supervised learning is one of the fundamental parts of machine learning that studies the relationship between input feature variables and a target output variable. It is employed across various fields to enhance decision-making efficacy through algorithms that learn informative patterns from data. Among the numerous supervised learning methods, the k nearest neighbor (kNN) (*Fix & Hodges, 1989*) method is one of the most widely used (*Singh, Thakur & Sharma, 2016*; *Ray, 2019*; *Sarker, 2021*). This method classifies a new instance using local information around the target query. Despite its simplicity, the kNN method has been successfully used in many applications including economic forecasting (*Imandoust & Bolandraftar, 2013*), disease diagnosis (*Sarkar & Leong, 2000*), and forestry (*Chirici et al., 2016*). In addition to the original version of kNN, there are variation approaches that modify aspects such as the distance metric, the definition of the neighborhood, or the classification rule. For example, the weight-adjusted kNN (*Han, Karypis & Kumar, 2001*) assigns weights to attributes according to the distance and determines the extent to which an attribute influences the classification. Discriminant adaptive nearest neighbor (*Tibshirani & Hastie, 1996*) proposed a local linear discriminant analysis for computing neighborhoods and could be employed in any neighborhood-based classifiers. Adaptive kNN (*Sun & Huang, 2010*) computes an optimal value of k for each test data using its nearest neighbor. Probabilistic nearest neighbor (*Holmes & Adams, 2002*) proposed a probabilistic framework that accommodates uncertainty in k as well as in the strength of the interaction between neighbors.

Most nearest-neighbor-based approaches utilize all feature variables as input for classification. However, real-world data may include a considerable number of features, many of which are often irrelevant to the outcome variable. The existence of non-informative features can be problematic for classification since the noisy features will dilute and mislead correct information, resulting in a significant increase in classification error (*Clarke et al., 2008*). This issue often occurs in high-dimensional data such as gene data (*Johnstone & Titterington, 2009*). Additionally, the curse of dimensionality may render distance-based information, which is crucial for the use of nearest-neighbor approaches, less effective in high-dimensional data (*Verleysen & François, 2005*; *Nettleton, Orriols-Puig & Fornells, 2010*; *Beyer et al., 1999*).

The ensemble method, which aggregates results from multiple classifiers to make the final decision, has gained much interest over the past three decades (*Piao et al., 2014*; *Moon et al., 2007*). To effectively classify high-dimensional data using kNN-based methods, many researchers have applied the ensemble technique with kNN. Voting kNN (*Grabowski, 2002*) utilizes multiple kNN classifiers with different values of k for voting the final result. *Zhou & Yu (2005)* proposed BagInRand (Bagging in Randomness), a bagging variant that constructs component kNN classifiers by altering both the training set and distance metric. *Li, Harner & Adjeroh (2014)* developed an ensemble of kNN classifiers with randomly sampled features. *Gul et al. (2018)* validates and ranks the component classifiers by bootstrapping from the training set. *Park & Kim (2015)* proposed a feature selection method based on nearest-neighbor ensemble classifiers. These techniques aim to refine kNN for high-dimensional data by utilizing and aggregating multiple classifiers.

In this article, we propose a novel nearest-neighbor-based method for high-dimensional data. Our method is based on the k conditional nearest neighbor algorithm (kCNN) (*Gweon, Schonlau & Steiner, 2019*), which efficiently provides class probability as well as classification results. We extend kCNN for high-dimensional data by applying it to a number of random subsets of features. Furthermore, our method further weighs the random subsets based on their importance. The performance and effectiveness of the proposed method are evaluated through simulation and experimental studies on a set of gene data.

The rest of this article is organized as follows. We introduce the proposed methods and techniques in the 'Method' section. We conduct simulation studies for an extensive analysis of the proposed method in different parameter settings in the 'Simulation' section. In 'Experimental study on gene expression data', we apply our method to a set of gene expression data and compare its predictive performance with other competitors. Finally, in the 'Conclusion' section, we conclude the article.

METHOD

In this section, we describe the proposed method and discuss the model parameters.

Random k conditional nearest neighbor

The random k conditional nearest neighbor (RkCNN) method combines kCNN classifiers that are used in different random feature subsets. Among the q features, a feature subset of size m ($1 \leq m \leq q$) is created using simple random sampling without replacement from the data feature space $\mathcal{F}$. Each component kCNN model is then trained on one of these subsets.

Let Y be the output variable with L classes and $\mathbf{y}$ be an observed value of Y. Let $\mathbf{X}$ be the $n \times q$ feature matrix consisting of n input vectors each of length q. Consider a random feature subset $\mathcal{F}_j \subseteq \mathcal{F}$ where the elements of $\mathcal{F}_j$ are selected features. The matrix corresponding to the random subset $\mathcal{F}_j$ is denoted by

$$\mathbf{X}'_j = \begin{pmatrix} \mathbf{x}'^T_1 \\ \vdots \\ \mathbf{x}'^T_n \end{pmatrix},$$

where $\mathbf{x}'_i$ is the feature vector of the ith observation with m features selected for $\mathcal{F}_j$.

When predicting the class of a new instance, the estimated probabilities are calculated by applying kCNN to the feature subset. In particular, for a given query vector $\mathbf{x}$, kCNN looks up the k-th nearest neighbor from each class and assigns the corresponding class probabilities

$$\hat{P}_j(Y = c|\mathbf{x}) = \frac{\left\| \mathbf{x}', \mathbf{x}'_{k|c} \right\|_2^{-1}}{\sum_{l=1}^{L} \left\| \mathbf{x}', \mathbf{x}'_{k|l} \right\|_2^{-1}} \tag{1}$$

where $\|\cdot\|_2$ is the L2-norm, and $\mathbf{x}'_{k|c}$ represents the k-th nearest neighbor of $\mathbf{x}'$ from class c.

Applying Eq. (1) to all h random feature subsets results in h sets of distinct class probability estimates. The proposed method aggregates these class probability estimations using weighted averaging, where the weights depend on the level of informativeness of the feature subsets for classification. If none of the features in a random subset plays an important role in classification, class probability estimates obtained from this subset space will add noisy information, potentially deteriorating the classification performance. Consequently, the proposed method assigns weights to estimated class probabilities from individual classifiers based on the informative level of their respective subset spaces. Specifically, we quantify the informativeness level for each feature subset using the separation score (for a detailed description of the separation score, please refer to 'Model Parameters') and compute the weights accordingly.

Given a set of separation scores $S_1, \ldots, S_h$, we sort the separation score in descending order and denote them using order statistics, such as $S_{(1)}, S_{(2)} \ldots S_{(h)}$. For effective weighting, the RkCNN algorithm disregards results from noisy feature subset spaces with relatively low separation scores. The weights are then computed using the relative magnitudes of the remaining scores. Specifically, we only consider the class probability results from the r subsets that correspond to the top r separation scores, denoted as $S_{(1)}, \ldots, S_{(r)}$. Based on these r remaining scores, the weights for each classifier are computed as

$$w_{(j)} = \frac{S_{(j)}}{\sum_{l=1}^{r} S_{(l)}}.$$

RkCNN estimates the aggregated probability

$$\hat{P}(Y = c|\mathbf{x}) = \sum_{j=1}^{r} \left[\hat{P}_{(j)}(Y = c|\mathbf{x}) \cdot w_{(j)}\right]$$

where $\hat{P}_{(j)}$ is the estimated probability obtained in the feature subset corresponding to $S_{(j)}$. For the classification task, RkCNN chooses the class with the greatest probability. That is,

$$\hat{Y} = \arg\max_c \hat{P}(Y = c|\mathbf{x}).$$

Algorithm 1 summarises the procedure of RkCNN. We also illustrate the procedure of RkCNN in Fig. 1. The tuning parameters for RkCNN are discussed in 'The number of nearest neighbors: k'.

Algorithm 1 RkCNN

```
1:  for $j = 1$ to $h$ do
2:      Sample $m$ features from the feature space $\mathcal{F}$, denoted as $\mathcal{F}_j$, where $\mathcal{F}_j \subseteq \mathcal{F}$
3:      Calculate the corresponding Separation Score $S_j = \frac{\mathrm{BV}}{\mathrm{WV}}$
4:  end for
5:  Sort $\{S_j\}_{j=1}^{h}$ in descending order, denoting the sorted values as $S_{(j)}$, i.e., $S_{(1)} \geq S_{(2)} \geq \cdots \geq S_{(h)}$
6:  Sort $\mathcal{F}_j$ by $S_{(j)}$, denoted as $\mathcal{F}_{(j)}$
7:  for $j = 1$ to $r$ do
8:      Use the matrix of $\mathcal{F}_{(j)}$, $\mathbf{X}'_{(j)}$, and $\mathbf{y}$ to construct a kCNN model, denote as $kCNN_{(j)}$
9:      Calculate weights $w_{(j)} = \frac{S_{(j)}}{\sum_{l=1}^{r} S_{(l)}}$
10:     Calculate $\hat{P}_{(j)}(Y = c|\mathbf{x})$, the predicted probability of class $c$ by $kCNN_{(j)}$
11: end for
12: Integrated probability for class $c$ is, $\hat{P}(Y = c|\mathbf{x}) = \sum_{j=1}^{r}\left[\hat{P}_{(j)}(Y = c|\mathbf{x}) \cdot w_{(j)}\right]$
13: Classify $\mathbf{x}$ to class $c$ which maximise $\hat{P}(Y = c|\mathbf{x})$
```

Separation score

The proposed RkCNN method assigns different weights to individual feature subsets. Utilizing noisy feature subsets that contain irrelevant features does not enhance model performance, it is advisable to assign small weights to such subsets. If a subset consists of features irrelevant to the classification task, no pattern is expected between the set of selected features and the class label. Conversely, kCNN will perform well in a subset in which the classes are well-separated.

To quantify the level of discriminative information available in any given feature subset, we use the between-group and within-group variances. Specifically, the variance between groups is defined as the variance among the center points of each class.

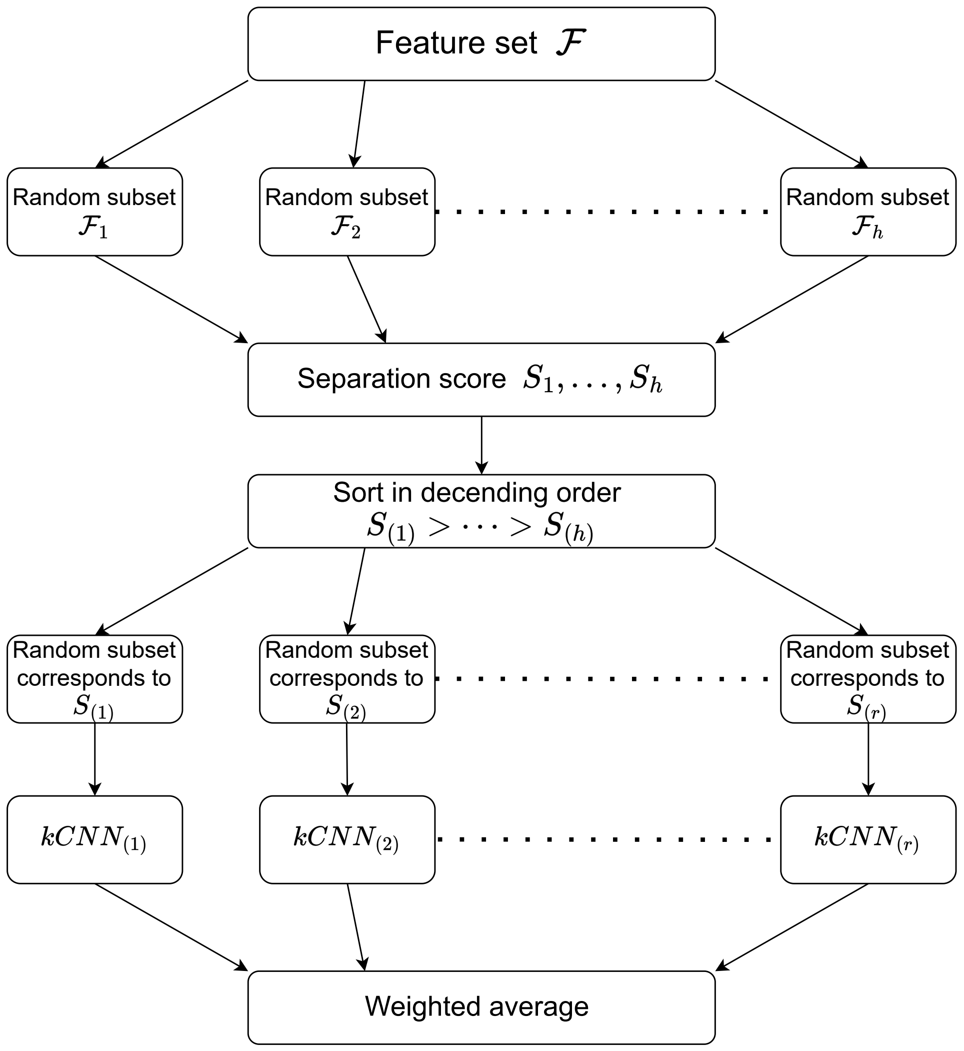
Feature set $\mathcal{F}$
Random subset $\mathcal{F}_1$
Random subset $\mathcal{F}_2$
Random subset $\mathcal{F}_h$
Separation score $S_1, \ldots, S_h$
Sort in decending order $S_{(1)} > \cdots > S_{(h)}$
Random subset corresponds to $S_{(1)}$
Random subset corresponds to $S_{(2)}$
Random subset corresponds to $S_{(r)}$
$kCNN_{(1)}$
$kCNN_{(2)}$
$kCNN_{(r)}$
Weighted average

Figure 1 **Illustration of the RkCNN algorithm.**
Full-size DOI: 10.7717/peerjcs.2497/fig-1

Let $\bar{\mathbf{x}}_\mathbf{c}$ be the mean vector for the class c and $\bar{\mathbf{x}}$ be the overall mean vector. For a feature subset $\mathcal{F}_j$, the between-group variance (BV) is the variance of L center points

$$\mathrm{BV} = \sum_{c=1}^{L} \frac{\left\|\bar{\mathbf{x}}_c' - \bar{\mathbf{x}}'\right\|_2}{L-1}.$$

The variance within groups is the mean of the variances in each class. Let $\mathbf{x}_{i_c}$ be the ith observations in class c, where $i_c = 1, \ldots, N_c$. The variance for class c is

$$Var_c = \sum_{i_c=1}^{N_c} \frac{\left\|\mathbf{x}_{i_c}' - \bar{\mathbf{x}}_c'\right\|_2}{N_c - 1}$$

where N_c is the number of data points in class c.

The variance within groups is the average of L variances

$$\mathrm{WV} = \sum_{c=1}^{L} \frac{Var_c}{L} = \sum_{c=1}^{L}\sum_{i_c=1}^{N_c} \frac{\left\|\mathbf{x}'_{i_c} - \bar{\mathbf{x}}'_c\right\|_2}{(N_c-1)L}.$$

Then, the separation score for the subset is defined as

$$S = \frac{\mathrm{BV}}{\mathrm{WV}} = \left[\sum_{i=c}^{L} \frac{\left\|\bar{\mathbf{x}}_c - \bar{\mathbf{x}}'\right\|_2}{L-1}\right] \Big/ \left[\sum_{c=1}^{L}\sum_{i_c=1}^{N_c} \frac{\left\|\mathbf{x}'_{i_c} - \bar{\mathbf{x}}'_c\right\|_2}{(N_c-1)L}\right].$$

A small S value indicates a relatively low BV compared to WV, suggesting that the class means are close to the overall mean with larger within-class variances. In such cases, the class means are relatively close to the overall mean and each class has a relatively larger variance. Hence, the class probability estimated by kCNN will be barely meaningful. Conversely, a large S value suggests that the feature subset's BV is relatively large in comparison to its WV. This indicates that the class means are distant from the overall mean, and the data is well-separable within this feature subspace. Such a feature subset is highly advantageous for utilizing kCNN, as it allows the classifier to effectively distinguish between classes.

Note that the computation of separation scores is model-free, as the separation score does not require the implementation of the kCNN. This implies that the computation can be quick once the following two matrices have been calculated.

$$\mathbf{B} = \begin{pmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_L \end{pmatrix} = \begin{pmatrix} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}) \circ (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}) & \cdots & (\bar{\mathbf{x}}_L - \bar{\mathbf{x}}) \circ (\bar{\mathbf{x}}_L - \bar{\mathbf{x}}) \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_N \end{pmatrix} = \begin{pmatrix} \dfrac{(\mathbf{x}_1 - \bar{\mathbf{x}}_1) \circ (\mathbf{x}_1 - \bar{\mathbf{x}}_1)}{N_1 - 1} & \cdots & \dfrac{(\mathbf{x}_N - \bar{\mathbf{x}}_L) \circ (\mathbf{x}_N - \bar{\mathbf{x}}_L)}{N_L - 1} \end{pmatrix}$$

where $\circ$ is the element-wise product, $\mathbf{b}_c$ and $\mathbf{w}_i$ represent the c-th and ith column vectors of $\mathbf{B}$ and $\mathbf{W}$, respectively.

For a feature subset $\mathcal{F}_j$, let $\mathbf{z}_j = (z_{1j}, z_{2j} \ldots, z_{qj})$ be the indicator vector, where z_{ij} takes the value 1 if ith feature is in $\mathcal{F}_j$ and 0 otherwise. Then BV and WV can be obtained by

$$\mathrm{BV} = \frac{\sum_{c=1}^{L} \sqrt{\mathbf{b}_c^T \mathbf{z}_j}}{L-1} \text{ and } \mathrm{WV} = \frac{\sum_{i=1}^{N} \sqrt{\mathbf{w}_i^T \mathbf{z}_j}}{L}. \tag{2}$$

The result Eq. (2) implies that once matrices B and W are obtained, the computation of BV and WV (and thus the separation score) for individual feature subsets can be quick because no additional computations for individual vectors in B and W are needed.

While RkCNN generates h random subsets, the method only uses the subsets with the top r $(\leq h)$ separation scores. Assuming that the number of informative features follows a hypergeometric distribution, in the presence of many non-informative features, most subsets may contain only a few informative features. This implies that the distribution of the separation score will likely be positively skewed. Using the top r subsets helps to block the contribution of non-informative subsets.

Selecting the top r subsets is analogous to filter methods in feature selection. While filter methods select top-ranked individual features, RkCNN ranks random subsets based on their separation scores.

It is worth noting that the proposed separation score is related to Fisher's linear discriminant analysis (LDA) (*Fisher, 1936*) which measures the level of separation using the variances between and within the classes. Fisher's LDA projects the data points onto a one-dimensional space such that the projections have large between-group variance and small within-group variance. Although both Fisher's LDA and our separation score use the between-/within-class variances, Fisher's LDA is designed to identify the vector that maximizes the ratio of between-group variance to within-group variance. In contrast, we use the separation score to quantify the level of separability in random feature subsets.

Model parameters

In this section, we discuss the model parameters of RkCNN.

The number of nearest neighbors: k

As with other kNN-based approaches, the performance of RkCNN is dependent on the choice of k. Additionally, the parameter k serves as an adjuster for the bias–variance trade-off (*Hastie, Tibshirani & Friedman, 2009*). A small value of k yields a model with a large variance and low bias. The prediction highly relies on local patterns from a small number of nearest observations, making it more flexible to the training data and reducing bias. However, this flexibility also renders the model sensitive to noise or minor changes, leading to higher variance. In contrast, a large value of k results in a model with a large bias and low variance. It averages over a greater number of neighbors for predictions. Although this oversimplification may overlook complicated patterns, it enhances stability across datasets, reducing variance and weakening the impact of minor changes in the training data.

The proposed RkCNN method takes the average of many kCNN models from different feature subsets. In the context of the bias–variance trade-off, this averaging technique helps reduce the variance without affecting the bias (*Gupta et al., 2022*). This implies that RKCNN is likely most effective when individual kCNN models have large variance but small bias. Consequently, we expect that RkCNN will perform well at small values of k.

The number of features in any feature subset: m

The parameter m represents the number of features selected for each random subset within the RkCNN framework. Increasing m enhances the likelihood of a feature being sampled for a random subset. However, excessively large values of m may lead to substantial overlap among the features in different random subsets, resulting in similar subset structures and correlated performance across these models. Conversely, the main concern with small values of m is that the resulting model may overlook interactions between the features that are potentially useful for classification. In the extreme case where $m = 1$, selecting the top r subsets is equivalent to selecting the top r features based on their separation scores.

Consequently, these separation scores reflect only the marginal contributions of individual features, without accounting for any interactions with other features. Thus, an appropriate value of m should balance the exploitation of feature interactions while maintaining a low level of overlap between the random subsets.

The number of contributing subsets: r

The parameter r represents the number of kCNN models that contribute to the RkCNN prediction. Increasing r is intended to enhance model robustness and model efficiency by incorporating a diverse set of models, each trained on a random subset of features. This diversity helps to reduce the risk of overfitting and improves the generalization of the model.

As RkCNN bases its predictions on the average of r model outcomes, the influence of any individual classifier diminishes as r increases, leading to a more stable prediction. This process allows the ensemble to capture more diverse patterns and finer details in the data, typically improving the overall accuracy. However, incorporating additional kCNN models increases the runtime and computational load. Therefore, while a higher r is desirable for improving model performance, it must be balanced against the computational costs.

Optimally, RkCNN performs best at higher r values, provided that the computational overhead remains within an acceptable range. This balance ensures that RkCNN remains both effective and efficient, making it suitable for various applications where robustness and quick prediction times are critical.

The total number of sampled random subsets: h

The parameter h $(\geq r)$ represents the total number of feature subsets generated by the RkCNN. Based on the separation scores, the RkCNN method only uses the top r out of h subsets for prediction. The remaining $(h-r)$ subsets are considered noisy subsets and thus excluded, as they do not significantly contribute to classification outputs. This selective approach allows RkCNN to focus computational resources on the most informative subsets, enhancing the efficiency and effectiveness of the model.

It is important to note that the subset filtering procedure is conducted prior to the construction of kCNN classifiers. That is, increasing h does not affect the number of kCNN classifiers ultimately constructed, thereby having a minimal impact on the computational time of the RkCNN algorithm by variations in h.

In summary, we expect that RkCNN performs well at a small value of k, a relatively small to moderate value of m, and a large value of r. Lastly, h can be any integer greater than or equal to r, and we examine $h = r, 2r \ldots, 10r$ in the 'Simulation and Experimental Study on Gene Expression Data' section.

SIMULATION

In this section, we explore the performance of the RkCNN model through a series of simulations. The simulation experiments are designed with various tunable parameters that allow us to systematically investigate the model's behavior in environments with different levels of noisy features.

Each simulation setting is crafted to mimic realistic scenarios where the presence of noise can significantly impact classification accuracy. Through these experiments, we aim to provide a comprehensive evaluation of the RkCNN model, focusing on its effectiveness in handling diverse and challenging data landscapes.

Data generation

Following the methodology outlined in Gul et al. (2018), the informative features of the two classes are generated with correlated and uncorrelated structures respectively. Suppose there are d $(\leq q)$ informative features in the dataset. For class A, the feature variables are generated from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_A, \omega\Psi)$, where the mean vector $\boldsymbol{\mu}_A$ is uniform with each element set to 2. Here, ω is a scaling factor that controls the overall variance of the features, modulating the extent of spread within class A. The variance–covariance matrix Ψ, a $d \times d$ matrix, is defined as

$$\Psi = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,d} \\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{d,1} & \sigma_{d,2} & \cdots & \sigma_{d,d} \end{pmatrix}$$

where $\sigma_{i,j} = (1/2)^{|i-j|}$. Conversely, the feature variables for class B are generated independently from $\mathcal{N}(\boldsymbol{\mu}_B, \mathcal{I}_d)$, where the mean vector $\boldsymbol{\mu}_B$ has value 1 on all elements and $\mathcal{I}_d$ is the $d \times d$ identity matrix, leading to uncorrelated features.

In each class, features are generated from identical distributions, ensuring that informativeness and separability are uniformly controlled. Then, non-informative features are generated from an independent standard normal distribution to complete the dataset.

Additionally, to explore scenarios involving multiple classes, we extend the data generation scheme to include a third class, C. Instances for class C are generated from $\mathcal{N}(\boldsymbol{\mu}_C, \omega_C\mathcal{I}_d)$, where the mean vector $\boldsymbol{\mu}_C$ has value 3 on all elements. ω_C is the variance parameter specific to class C, controlling the dispersion of feature values in Class C and influencing the degree of overlap with other classes, thereby affecting the classification challenge.

In the remainder of this section, we conduct a comprehensive examination of the performance of the proposed method across various simulation settings. This involves varying the data size, the number of non-informative features, the number of classes, and the value of ω.

Simulation results and analysis

Separation score vs number of informative features

We evaluate the proposed separation score as an informativeness measure of feature subsets. The relationship between the separation score and the number of informative features is illustrated in Fig. 2. For each subplot within the figure, the simulation data contained 1,000 instances per class, comprising 20 informative features and 500 non-informative features as noisy factors. We analyzed the distribution of 800 separation scores.

The result indicated that the separation score for a feature subset increased as the subset contained more informative features. Notably, the most significant increase in score

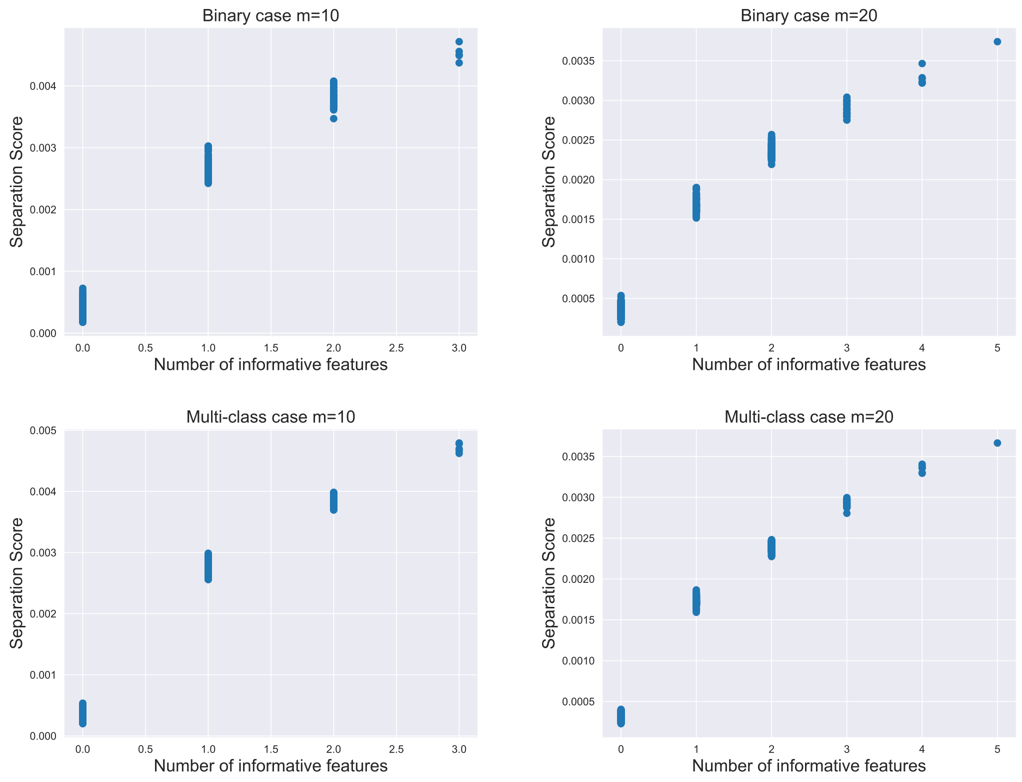

Figure 2 The distribution of separation scores under different simulation settings. The value of m represents the number of features in random subsets.

Full-size DOI: 10.7717/peerjcs.2497/fig-2

occurred when moving from zero to one informative feature, *i.e.*, transitioning from a subset of purely noisy features to one containing an informative element. This pattern held true in both binary and multi-class scenarios, emphasizing the separation score as a potential proxy for the unknown count of informative features in practical applications.

Effect of k and m

Figure 3 illustrates the number and proportion of informative features within the $r = 200$ contributing subsets across varying values of m. The left panel showed a positive correlation between m and the number of informative features. As more features were included in a random subset, the likelihood of sampling informative features increased. Conversely, the right panel revealed a negative correlation between m and the proportion of the informative features. This indicated that larger subsets tended to be more dominated by noisy features, diluting the concentration of informative elements.

Figure 4 shows RkCNN's classification accuracy across different values of m and k in the binary and multi-class problems. We varied ω and ω_C to adjust the extent of spread in the classes A and C, affecting their degree of overlap. When the classes were well-separated (*i.e.*, $\omega = 1$ for the binary and $\omega = \omega_C = 1$ for the multi-class problems), neither k nor m significantly affected RkCNN's accuracy. However, as the class overlap increased with higher ω and ω_C, the performance of RkCNN became sensitive to the choice of k and

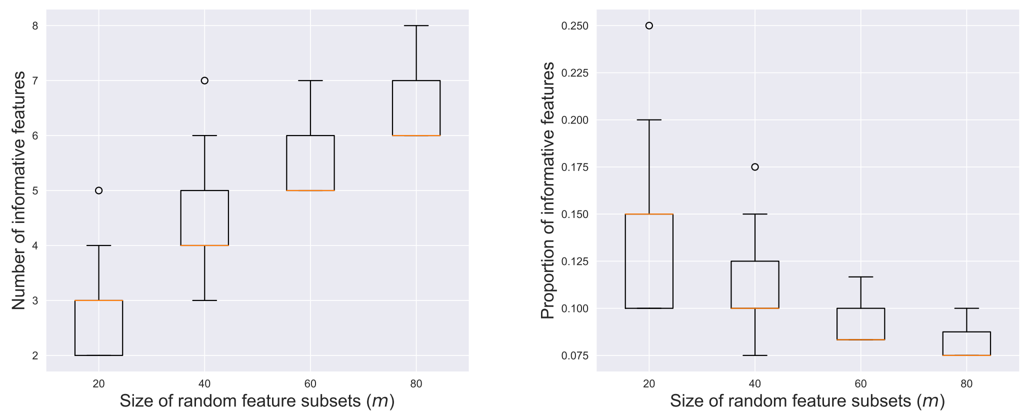

Figure 3 **Number (left) and proportion (right) of informative features among the contributing subsets.**

Full-size DOI: 10.7717/peerjcs.2497/fig-3

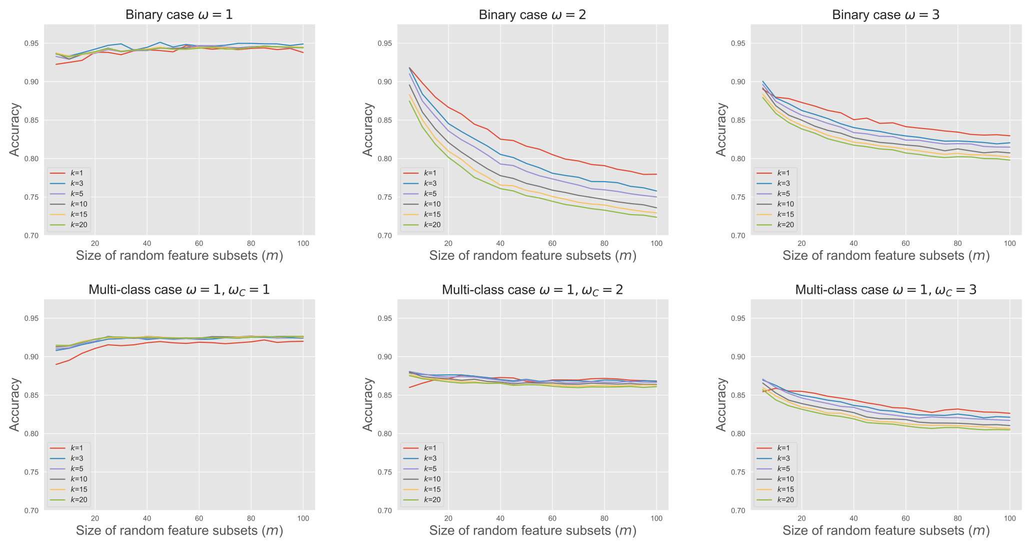

Figure 4 **RkCNN classification accuracy for different k and m.**

Full-size DOI: 10.7717/peerjcs.2497/fig-4

m. The decreased accuracy at higher m values was attributed to the raised proportion of non-informative features, as depicted in the right panel of Fig. 3, which compromised the precision of class probability estimations. Notably, smaller values of k generally performed well regardless of m in the majority of cases. Our result regarding the relationship between k and the level of class overlap is also consistent with that of *García, Mollineda & Sánchez (2008)* who use the kNN classifier. Based on these findings, smaller values of k and m were recommended to optimize the classification performance of RkCNN, particularly under conditions of significant class overlap.

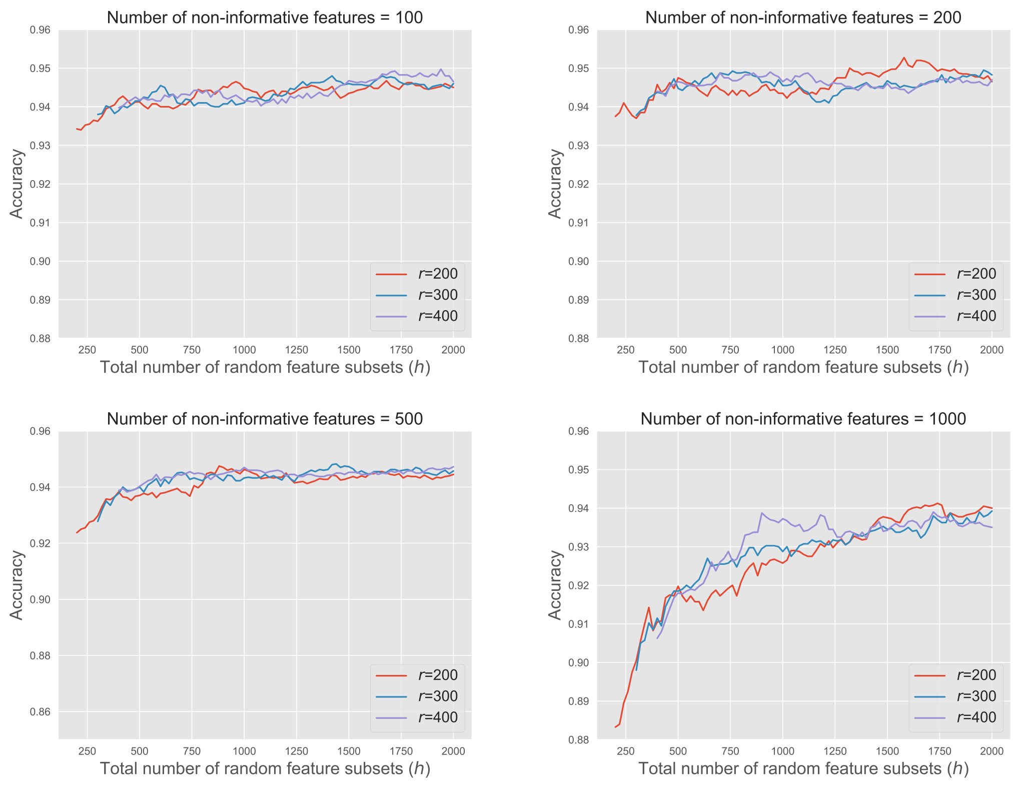

Figure 5 RkCNN with *r* and *h* at different noisy levels.
Full-size DOI: 10.7717/peerjcs.2497/fig-5

Effect of r and h at different noisy levels

Figure 5 shows the accuracy of RkCNN across different values of r and h under various noisy conditions, with the number of non-informative features ranging from 100 to 1,000. We fix $k = 1$ and $m = \lceil \sqrt{q} \rceil$ for this analysis.

The results demonstrated that increasing h, the total number of feature subsets, improved model accuracy, and the marginal gains in accuracy decreased as h became larger. This pattern suggested that although larger values of h led to better classification results, the benefits of adding more subsets grow smaller, eventually leading to a stabilization of performance at higher values of h.

When the number of non-informative features was relatively small, such as 100 or 200, the performance of RkCNN showed relative insensitivity to changes in h, maintaining stable accuracy across a broad range of h values. Conversely, in circumstances with 500 and especially 1,000 non-informative features, classification accuracy improved significantly as h increased. This was due to the larger pool of candidate subsets, which provided a higher probability of selecting subsets enriched with informative features. This trend highlighted that beyond a certain point, further increases in h did not substantially boost performance, as the accuracy plateaued.

The variation in r (200, 300, 400) exhibited similar patterns across all scenarios, suggesting that while a very large r is not necessary for achieving optimal performance,

Table 1 **Misclassification rate for the methods on datasets with 20 informative features and different numbers of non-informative features.**

Num of feature	kNN	kCNN	RkNN	ESkNN	RkCNN
20	0.070	0.047	0.103	0.208	0.083
20+50	0.088	0.070	0.055	0.190	0.042
20+100	0.180	0.190	0.123	0.165	0.057
20+200	0.135	0.125	0.178	0.188	0.047
20+500	0.285	0.280	0.350	0.293	0.073
20+1000	0.283	0.298	0.420	0.333	0.113

maintaining a sufficiently large r ensures efficient use of computational resources without redundant calculations. Overall, the analysis underscores the importance of carefully adjusting h and r according to the noise level in the data to optimize the accuracy of the RkCNN model.

Comparison

We compare RkCNN with several nearest-neighbor-based methods, including kNN, kCNN, and random k nearest neighbor (RkNN) and the ensemble of subset of k nearest neighbor (ESkNN). The misclassification rates under different numbers of non-informative features are presented in Table 1 and the corresponding average runtimes of ensemble methods are presented in Fig. 6. The tuning parameters were fixed at $k = 3$ and $r = 200$ through the comparison. For ESkNN, all other tuning parameters were set according to the recommendations in *Gul et al. (2018)*. Given the diversity in feature numbers across datasets, we set the value of m for three ensemble methods associated with the feature size, specifically $m = \lceil \sqrt{q} \rceil$.

From the results in Table 1, kNN and kCNN performed competitively when only informative features were presented as shown in the first row of the table. Under these conditions, kCNN exhibited the lowest error rate among the individual methods, and RkCNN showcased the lowest error rate among the ensemble methods. This was expected since the effectiveness of feature utilization is maximized in the absence of noise. When non-informative features were included, RkCNN outperformed all other methods in terms of classification error rate. The error rate increased as more non-informative features were introduced into the dataset, yet RkCNN remained robust against this increase in noise. Among the three ensemble methods compared, RkCNN consistently outperformed both kCNN and ESkNN in all scenarios.

Figure 6 presents the runtimes of RkNN, ESkNN and RkCNN at $r = 200$. For all ensemble methods, runtimes increased with the number of features. Among the methods, RkNN exhibited the fastest runtimes compared to ESkNN and RkCNN. The runtime of RkCNN was affected by the choice of the parameter h. When $h = r$, RkCNN's runtime was comparable to that of RkNN. However, when $h = 3r$, RkCNN's runtime increased due to the additional computations for separation scores.

The performance variations across different numbers of non-informative features illustrated RkCNN's robustness, leading us to further investigate the impact of the spread

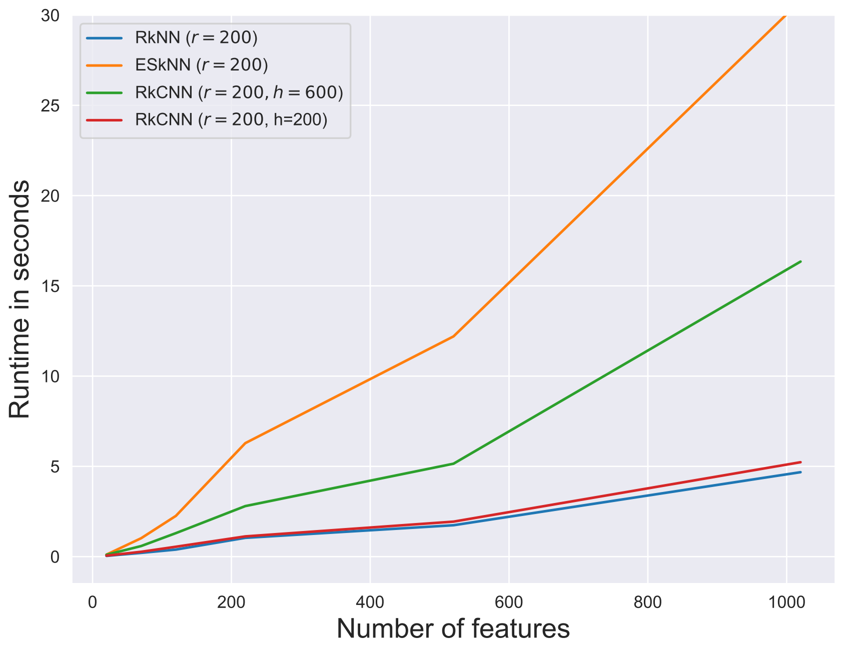

Figure 6 **Average runtime for different ensemble methods.**
Full-size DOI: 10.7717/peerjcs.2497/fig-6

parameter ω on classification accuracy. The misclassification rates for different values of ω are presented in Table 2, offering insights into how each method copes with varying levels of entropy or uncertainty in class distributions.

As ω increased, particularly to 3 or 5, the ensemble methods exhibited their highest misclassification rates, indicating significant uncertainty in class distributions. At high levels of class overlap, nearest-neighbor-based approaches tended to perform well at $k = 1$, which is consistent with the results of *García, Mollineda & Sánchez (2008)*. However, an interesting trend emerged where further increases in ω led to better classification results, suggesting a threshold beyond which the clarity of class separability enhances the models' performance. RkCNN consistently outperformed other methods in all simulated cases, demonstrating superior robustness to both increased numbers of non-informative features and varied ω values.

This pattern was particularly notable in environments with higher noise levels. When 500 and 1,000 non-informative features were present, RkCNN's advantage became more pronounced, confirming its effectiveness in managing complex scenarios with significant class overlap and high noise levels. Although the error rates generally increased with more non-informative features, RkCNN's performance remained comparatively stable. This stability underscored RkCNN's potential as a reliable tool for applications where data conditions were challenging.

Throughout this section, the comparative analyses and simulation studies have consistently highlighted RkCNN's performance under various testing conditions. From managing datasets with large amounts of non-informative features to coping with increased

Table 2 Misclassification rate for the methods on datasets with 20 informative features, varying numbers of non-informative features, and different values of ω.

	200 non-informative features									
ω k	**kNN**		**kCNN**		**RkNN**		**ESkNN**		**RkCNN**	
	1	**3**	**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**
1	0.188	0.135	0.188	0.125	0.175	0.178	0.188	0.188	0.068	0.055
3	0.267	0.272	0.267	0.255	0.188	0.233	0.235	0.215	0.113	0.157
5	0.295	0.303	0.295	0.310	0.115	0.188	0.205	0.198	0.095	0.162
10	0.305	0.340	0.305	0.353	0.080	0.100	0.090	0.178	0.035	0.125
15	0.320	0.348	0.320	0.370	0.042	0.065	0.073	0.140	0.020	0.093
	500 non-informative features									
1	0.323	0.285	0.323	0.280	0.338	0.338	0.262	0.248	0.085	0.057
3	0.323	0.380	0.323	0.350	0.368	0.365	0.305	0.298	0.127	0.150
5	0.335	0.370	0.335	0.370	0.280	0.308	0.225	0.223	0.140	0.140
10	0.373	0.385	0.373	0.397	0.193	0.210	0.140	0.111	0.057	0.110
15	0.360	0.400	0.360	0.413	0.123	0.155	0.090	0.113	0.042	0.080
	1,000 non-informative features									
1	0.405	0.283	0.405	0.298	0.377	0.428	0.305	0.248	0.162	0.127
3	0.382	0.360	0.382	0.308	0.402	0.430	0.360	0.300	0.195	0.215
5	0.387	0.382	0.387	0.370	0.363	0.392	0.275	0.250	0.165	0.190
10	0.363	0.395	0.363	0.443	0.285	0.333	0.170	0.167	0.125	0.135
15	0.373	0.392	0.373	0.433	0.235	0.280	0.095	0.118	0.070	0.105

entropy in class distributions, RkCNN proved to be effective in high-dimensional data that contain noisy features.

EXPERIMENTAL STUDY ON GENE EXPRESSION DATA

To evaluate and compare the performance of the proposed model, we conducted experiments using real-world gene expression datasets specifically designed for cancer microarray data classification analysis, as referenced in *Mramor et al. (2007)*. These datasets are accessible for download at https://file.biolab.si/biolab/supp/bi-cancer/main-2.html. Table 3 summarises some statistics of each dataset including the number of observations, the number of features, the number of classes, and the class distributions. Some datasets, such as Tumor2 and Tumor3, originate from the same study but define their class labels differently, resulting in variable class counts. The datasets are predominantly high-dimensional, with feature counts ranging from 2308 to 22283, and most include no more than 100 observations, with some exhibiting significant class imbalance.

A comparative analysis is made between the proposed RkCNN method and various well-known classifiers, including kNN, kCNN, and RkNN. We also include the random forest (RF) (*Breiman, 2001*) and support vector machine (SVM) (*Cortes & Vapnik, 1995*) in the model comparison. For all nearest-neighbor-based methods, we tuned the model parameters using a grid search with $k = 1, 2$ and $m = 20, 50, 80$ for RkNN and RkCNN. We fix $r = 300$ for RkNN and RkCNN, and $h = 900$ for RkCNN. RF and SVM were implemented using Python's scikit-learn package. The classification accuracies were

Table 3 **Data summary of gene expression datasets used in the experiments.**

	Instances	Features	Classes	Class distributions
MLL	72	12,533	3	24/20/28
SRBCT	83	2,308	4	29/11/18/25
DLBCL	77	7,070	2	58/19
Prostate	102	12,533	2	50/52
LUNG	203	12,600	5	139/17/6/21/20
LEU	72	5,147	2	42/25
AML	54	12,625	2	28/26
Childhood2	110	8,280	2	50/60
Childhood4	60	8,280	4	13/21/16/10
Breast Cancer	24	12,625	2	14/10
Breast Colon	52	22,283	2	31/21
Bladder	40	5,724	3	10/19/11
CML	28	12,625	2	12/16
Tumor2	23	9,945	2	11/12
Tumor3	23	9,945	3	11/3/9
gastric2	30	4,522	2	8/22
gastric3	30	4,522	3	8/5/17
LL2	19	15,434	2	9/10
LL3	29	15,434	3	9/10/10
lungCancer	34	10,541	3	17/8/9
medulloblastoma	23	1,465	2	10/13
prostate cancer	20	12,627	2	10/10
braintumor	40	7,129	5	10/10/10/4/6
glioblastoma	50	12,625	4	14/7/14/15

evaluated using leave-one-out cross-validation (LOOCV). Considering imbalanced class distributions, we also used the Matthews correlation coefficient (*Matthews, 1975* MCC). The metric incorporates the full confusion matrix, providing a robust metric for reflecting classifier performance across both majority and minority classes. MCC values range from −1 to 1, where a higher value indicates better performance. An MCC of 1 denotes perfect classification, 0 indicates random prediction, and −1 reflects complete disagreement between predicted and actual values.

Tables 4 and 5 demonstrate that RkCNN outperformed all nearest-neighbor-based methods in 20 and 18 out of 24 datasets in terms of accuracy and MCC, respectively. In addition, RkCNN showed comparable performance to RF and, on average, achieved the highest LOOCV accuracy and MCC, as well as the best rankings on both metrics. These results validate that RkCNN not only competes effectively with traditional machine learning algorithms but also excels in addressing the complexities and variabilities inherent in gene expression data.

Overall, the proposed RkCNN method demonstrates leading or high rankings across the microarray datasets. When other methods were in the lead, RkCNN tended to be the second-best method. We also conducted pairwise Wilcoxon test (*Wilcoxon, 1945*) to assess

Table 4 **LOOCV accuracy for each method across the gene expression datasets.** 'Average accuracy' represents the average accuracy of each method. 'Average rank' represents the average of the method's ranking across datasets.

Dataset	kNN	kCNN	RkNN	RkCNN	RF	SVM
MLL	0.819	0.819	0.861	0.889	0.931	0.903
SRBCT	0.855	0.855	0.904	0.952	1	0.964
DLBCL	0.831	0.870	0.896	0.909	0.883	0.818
Prostate	0.833	0.853	0.882	0.892	0.902	0.873
LUNG	0.887	0.911	0.892	0.901	0.926	0.906
LEU	0.699	0.849	0.781	0.863	0.973	0.877
AML	0.593	0.556	0.611	0.556	0.648	0.537
Childhood2	0.673	0.755	0.755	0.782	0.818	0.745
Childhood4	0.367	0.450	0.450	0.467	0.400	0.450
Breast Cancer	0.833	0.750	0.792	0.833	0.833	0.875
Breast Colon	0.923	0.923	0.904	0.923	0.942	0.808
Bladder	0.725	0.725	0.850	0.925	0.875	0.825
CML	0.429	0.500	0.500	0.500	0.464	0.464
Tumor2	0.870	0.870	0.913	0.913	0.913	0.609
Tumor3	0.609	0.783	0.652	0.739	0.783	0.565
gastric2	0.933	1	1	1	0.933	0.833
gastric3	0.767	0.733	0.833	0.833	0.767	0.700
LL2	1	1	1	1	0.947	0.947
LL3	0.966	0.966	0.931	0.966	0.897	0.897
lungCancer	0.618	0.618	0.676	0.735	0.735	0.588
medulloblastoma	0.565	0.565	0.565	0.609	0.304	0.478
Prostate cancer	0.600	0.700	0.700	0.700	0.700	0.600
braintumor	0.725	0.725	0.700	0.700	0.725	0.675
glioblastoma	0.580	0.580	0.720	0.740	0.760	0.640
Average accuracy	0.738	0.766	0.782	0.805	0.794	0.732
Average rank	4.583	3.646	3.271	2.375	2.583	4.542

whether the differences in rankings between the methods were statistically significant. Each test was performed as a one-sided test with a 5% significance level. According to the Wilcoxon test results, RkCNN ranked significantly better than the other methods in both accuracy and MCC, except RF, with no significant difference in ranking between RkCNN and RF(p-value = 0.549 in accuracy and 0.292 in MCC).

Findings in this section underscore the efficacy of RkCNN in practical applications, demonstrating its classification performance in scenarios characterized by high-dimensional data. The adaptability and superior performance of RkCNN suggest its suitability for complex classification tasks where accurate and reliable predictions are crucial.

CONCLUSION

In this article, we presented RkCNN, an innovative ensemble classification method that aggregates kCNN classifiers generated from random feature subsets. We also proposed

Table 5 LOOCV MCC for each method across the gene expression datasets. 'Average accuracy' represents the average accuracy of each method. 'Average rank' represents the average of the method's ranking across datasets.

Dataset	kNN	kCNN	RkNN	RkCNN	RF	SVM
MLL	0.744	0.744	0.771	0.853	0.896	0.854
SRBCT	0.801	0.801	0.868	0.918	0.984	0.950
DLBCL	0.502	0.664	0.787	0.707	0.585	0.460
Prostate	0.667	0.706	0.745	0.786	0.786	0.747
LUNG	0.763	0.818	0.785	0.786	0.839	0.808
LEU	0.287	0.664	0.486	0.615	0.910	0.734
AML	0.182	0.113	0.267	0.151	0.262	0.068
Childhood2	0.358	0.507	0.485	0.560	0.578	0.491
Childhood4	0.100	0.223	0.223	0.256	0.203	0.226
Breast	0.657	0.543	0.580	0.676	0.657	0.742
Breast	0.841	0.841	0.841	0.841	0.880	0.629
Bladder	0.595	0.595	0.689	0.806	0.884	0.741
CML	−0.108	−0.043	0.132	−0.021	−0.344	−0.300
Tumor2	0.742	0.768	0.840	0.840	0.763	0.233
Tumor3	0.350	0.641	0.414	0.641	0.391	0.228
gastric2	0.853	1.000	1.000	1.000	0.829	0.553
gastric3	0.616	0.616	0.707	0.645	0.666	0.473
LL2	1.000	1.000	1.000	1.000	0.900	0.900
LL3	0.950	0.950	0.950	0.950	0.950	0.856
lungCancer	0.369	0.369	0.475	0.512	0.742	0.344
medulloblastoma	0.115	0.115	0.115	0.115	−0.250	−0.271
prostate	0.218	0.436	0.314	0.436	0.503	0.204
braintumor	0.655	0.655	0.655	0.655	0.626	0.595
glioblastoma	0.437	0.437	0.619	0.620	0.591	0.511
Average MCC	0.529	0.590	0.615	0.639	0.618	0.491
Average rank	4.563	3.604	3.063	2.438	2.813	4.521

the separation score metric to weigh class probability estimates derived from individual kCNN models. Comprehensive simulations were conducted to illustrate the efficacy of separation scores in enhancing model performance. Additionally, we undertook an in-depth exploration of the impact of each parameter on RkCNN's performance, further deepening our comprehension of its functional mechanisms.

The simulation and experimental studies demonstrated that RkCNN outperforms other nearest-neighbor-based ensemble approaches, showcasing its effective handling of complex, high-dimensional data environments. Although the parameter tuning process is experimental within this article, other approaches such as adaptive tuning or theoretical analysis deserve further investigation.

While RkCNN filters random subsets of features based on their separation scores, it does not perform feature selection. However, the concept of ranking random subsets could potentially be adapted to filter individual features, thereby introducing a feature selection approach. This potential extension will be addressed in future work.

Beyond its application to gene expression data, the versatility of the proposed method suggests it could be effectively deployed in the analysis of other high-dimensional datasets, which are often characterized by a large proportion of non-informative features. This applicability potential makes RkCNN a promising tool in a wide range of data-intensive fields.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding
This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC, grant numbers 04698 (PI: Hyukjun Gweon)). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures
The following grant information was disclosed by the authors:
The Natural Sciences and Engineering Research Council of Canada (NSERC): 04698.

Competing Interests
The authors declare there are no competing interests.

Author Contributions
- Jiaxuan Lu performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Hyukjun Gweon conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability
The following information was supplied regarding data availability:

The datasets used are available at: https://file.biolab.si/biolab/supp/bi-cancer/main-2.html.

The python code files are available in the Supplemental Files and Zenodo: aljers. (2024). aljers/RkCNN: RkCNN Release Version 1.0 (Main). Zenodo. https://doi.org/10.5281/zenodo.13855998.

Supplemental Information
Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.2497#supplemental-information.

REFERENCES

Beyer K, Goldstein J, Ramakrishnan R, Shaft U. 1999. When is "Nearest Neighbor" meaningful? In: Beeri C, Buneman P, eds. *Proceedings of International Conference on Database Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 217–235 DOI 10.1007/3-540-49257-7_15.

Breiman L. 2001. Random forests. *Machine Learning* **45**:5–32 DOI 10.1023/A:1010933404324.

Chirici G, Mura M, McInerney D, Py N, Tomppo EO, Waser LT, Travaglini D, McRoberts RE. 2016. A meta-analysis and review of the literature on the k-Nearest Neighbors technique for forestry applications that use remotely sensed data. *Remote Sensing of Environment* **176**:282–294 DOI 10.1016/j.rse.2016.02.001.

Clarke R, Ressom HW, Wang A, Xuan J, Liu MC, Gehan EA, Wang Y. 2008. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer* **8**(**1**):37–49 DOI 10.1038/nrc2294.

Cortes C, Vapnik V. 1995. Support-vector networks. *Machine Learning* **20**:273–297 DOI 10.1007/BF00994018.

Fisher RA. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7**(**2**):179–188 DOI 10.1111/j.1469-1809.1936.tb02137.x.

Fix E, Hodges JL. 1989. Discriminatory analysis. Nonparametric discrimination: consistency properties. *International Statistical Review* **57**(**3**):238–247 DOI 10.2307/1403797.

García V, Mollineda RA, Sánchez JS. 2008. On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications* **11**:269–280 DOI 10.1007/s10044-007-0087-5.

Grabowski S. 2002. Voting over multiple k-NN classifiers. In: *Proceedings of the international conference on modern problems of radio engineering, telecommunications and computer science*. 223–225 DOI 10.1109/TCSET.2002.1015937.

Gul A, Perperoglou A, Khan Z, Mahmoud O, Miftahuddin M, Adler W, Lausen B. 2018. Ensemble of a subset of kNN classifiers. *Advances in Data Analysis and Classification* **12**:827–840 DOI 10.1007/s11634-015-0227-5.

Gupta N, Smith J, Adlam B, Mariet ZE. 2022. Ensembles over classifiers: a bias-variance perspective. *Transactions on Machine Learning Research* **2022** DOI 10.48550/arXiv.2206.10566.

Gweon H, Schonlau M, Steiner SH. 2019. The k conditional nearest neighbor algorithm for classification and class probability estimation. *PeerJ Computer Science* **5**:e194 DOI 10.7717/peerj-cs.194.

Han E-HS, Karypis G, Kumar V. 2001. Text categorization using weight adjusted k-nearest neighbor classification. In: Cheung D, Williams GJ, Li Q, eds. *Advances in knowledge discovery and data mining*. Berlin, Heidelberg: Springer, 53–65 DOI 10.1007/3-540-45357-1_9.

Hastie T, Tibshirani R, Friedman JH. 2009. *The elements of statistical learning: data mining, inference, and prediction*. 2nd edition. New York: Springer DOI 10.1007/978-0-387-84858-7.

Holmes CC, Adams NM. 2002. A probabilistic nearest neighbour method for statistical pattern recognition. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **64**(**2**):295–306 DOI 10.1111/1467-9868.00338.

Imandoust S, Bolandraftar M. 2013. Application of K-nearest neighbor (KNN) approach for predicting economic events theoretical background. *International Journal of Engineering Research and Applications* **3**:605–610.

Johnstone IM, Titterington DM. 2009. Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society a: Mathematical, Physical and Engineering Sciences* **367(1906)**:4237–4253 DOI 10.1098/rsta.2009.0159.

Li S, Harner EJ, Adjeroh DA. 2014. Random KNN. In: *Proceedings of the 2014 IEEE international conference on data mining workshop*. Piscataway: IEEE, 629–636 DOI 10.1109/ICDMW.2014.112.

Matthews B. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica Et Biophysica Acta (BBA)—Protein Structure* **405(2)**:442–451 DOI 10.1016/0005-2795(75)90109-9.

Moon H, Ahn H, Kodell RL, Baek S, Lin C-J, Chen JJ. 2007. Ensemble methods for classification of patients for personalized medicine with high-dimensional data. *Artificial Intelligence in Medicine* **41(3)**:197–207 DOI 10.1016/j.artmed.2007.07.003.

Mramor M, Leban G, Demšar J, Zupan B. 2007. Visualization-based cancer microarray data classification analysis. *Bioinformatics* **23(16)**:2147–2154 DOI 10.1093/bioinformatics/btm312.

Nettleton DF, Orriols-Puig A, Fornells A. 2010. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review* **33(4)**:275–306 DOI 10.1007/s10462-010-9156-z.

Park CH, Kim SB. 2015. Sequential random k-nearest neighbor feature selection for high-dimensional data. *Expert Systems with Applications* **42(5)**:2336–2342 DOI 10.1016/j.eswa.2014.10.044.

Piao Y, Park HW, Jin CH, Ryu KH. 2014. Ensemble method for classification of high-dimensional data. In: *2014 international conference on big data and smart computing*. 245–249 DOI 10.1109/BIGCOMP.2014.6741445.

Ray S. 2019. A quick review of machine learning algorithms. In: *Proceedings of the 2019 international conference on machine learning, big data, cloud and parallel computing*. 35–39 DOI 10.1109/COMITCon.2019.8862451.

Sarkar M, Leong T. 2000. Application of K-nearest neighbors algorithm on breast cancer diagnosis problem. In: *Proceedings of the AMIA symposium*. 759–763.

Sarker IH. 2021. Machine learning: algorithms, real-world applications and research directions. *SN Computer Science* **2**:160 DOI 10.1007/s42979-021-00592-x.

Singh A, Thakur N, Sharma A. 2016. A review of supervised machine learning algorithms. In: *Proceedings of the 2016 3rd international conference on computing for sustainable global development*. 1310–1315.

Sun S, Huang R. 2010. An adaptive k-nearest neighbor algorithm. In: *Proceedings of the 2010 seventh international conference on fuzzy systems and knowledge discovery, vol. 1*. 91–94 DOI 10.1109/FSKD.2010.5569740.

Tibshirani R, Hastie T. 1996. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18(06)**:607–616 DOI 10.1109/34.506411.

Verleysen M, François D. 2005. The curse of dimensionality in data mining and time series prediction. In: Cabestany J, Prieto A, Sandoval F, eds. *Computational intelligence and bioinspired systems. IWANN 2005. Lecture notes in computer science, vol 3512*. Berlin, Heidelberg: Springer, 758–770 DOI 10.1007/11494669_93.

Wilcoxon F. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin* **1**(**6**):80–83 DOI 10.2307/3001968.

Zhou Z-H, Yu Y. 2005. Adapt bagging to nearest neighbor classifiers. *Journal of Computer Science and Technology* **20**(**1**):48–54 DOI 10.1007/s11390-005-0005-5.