SUBiNN: a stacked uni- and bivariate kNN sparse ensemble

Elsten, T.; Rooij, M. de

Advances in Data Analysis and Classification (2022) 16:847–874
https://doi.org/10.1007/s11634-021-00462-7

REGULAR ARTICLE

SUBiNN: a stacked uni- and bivariate *k*NN sparse ensemble

Tiffany Elsten[1] · **Mark de Rooij**[1]

Received: 9 July 2020 / Revised: 1 September 2021 / Accepted: 3 September 2021 /
Published online: 6 October 2021

Abstract
Nearest Neighbor classification is an intuitive distance-based classification method. It has, however, two drawbacks: (1) it is sensitive to the number of features, and (2) it does not give information about the importance of single features or pairs of features. In stacking, a set of base-learners is combined in one overall ensemble classifier by means of a meta-learner. In this manuscript we combine univariate and bivariate nearest neighbor classifiers that are by itself easily interpretable. Furthermore, we combine these classifiers by a Lasso method that results in a sparse ensemble of nonlinear main and pairwise interaction effects. We christened the new method SUBiNN: Stacked Uni- and Bivariate Nearest Neighbors. SUBiNN overcomes the two drawbacks of simple nearest neighbor methods. In extensive simulations and using benchmark data sets, we evaluate the predictive performance of SUBiNN and compare it to other nearest neighbor ensemble methods as well as Random Forests and Support Vector Machines. Results indicate that SUBiNN often outperforms other nearest neighbor methods, that SUBiNN is well capable of identifying noise features, but that Random Forests is often, but not always, the best classifier.

Keywords Classification · Distance · Ensemble methods · Lasso regression · Nearest neighbors

Mathematics Subject Classification 62H30

1 Introduction

K Nearest Neighbors (*k*NN) is an intuitive, distance-based method for classification. The outcome class of observation *i* is predicted by the most frequently occurring class among its *k* nearest neighbors in the feature space. A small value for *k* causes

✉ Mark de Rooij
rooijm@fsw.leidenuniv.nl

[1] Methodology and Statistics Department, Institute of Psychology, Leiden University, PO Box 9555, 2300 RB Leiden, The Netherlands

predictions to be precise, but holds the risk of being unstable (Hassanat et al. 2014). With large k the predictions are more stable but probably biased. kNN's nonparametric nature and the lack of assumptions make it suitable and effective in many settings (Hastie et al. 2001). Although kNN can be used for both binary as well as multiclass classification, in the current manuscript we focus on binary classification.

kNN is, however, not without its drawbacks. We like to point out two drawbacks. First, kNN's prediction accuracy is negatively affected by the number of features, resulting in a *curse of dimensionality* (Hastie et al. 2001). Second, the kNN classifier does not allow for the identification or interpretation of the effect of the individual features or pairs of features.

Past research focused on improving the predictive accuracy by creating an ensemble of classifiers, composed of random subsets of features or some optimal selection of features (Bay 1999; Domeniconi and Yan 2004; Gul et al. 2016; Li et al. 2011; Zhou and Yu 2005). For an ensemble method to work well, the separate models need to be both diverse and accurate (Bay 1999). kNN, however, is a stable method when it comes to small changes in the input data, especially for large k.

In Bagging, bootstrap samples from the data are taken and a classifier is fitted on each bootstrap sample. These classifiers are then used to create predictions by either using their average, a majority vote, or some other decision rule. Breiman (1996) identified kNN's stability as the limiting factor for the performance with bagging. Bay (1999) reported better performance for bagging when using a random subset of the features without resampling. Domeniconi and Yan (2004) noted that using a subset of features carries the risk of possibly including non-informative features or discarding informative ones, and for that reason make use of feature relevance. Gul et al. (2016) combined the use of a random set of features while also using resampled versions of the data.

Boosting is an iterative algorithm that gives more weight to data points that are difficult to classify. This is done by either directly weighting the input data, or by weighted resampling (Dietterich 2000). Weighting the data points does not improve the situation for kNN, however, because the classification of a data point depends on its neighbors and never on its own weight (Bay 1999). García-Pedrajas and Ortiz-Boyer (2009) propose two methods in which not the points are weighted, but the input space is modified in such a way that difficult data points are more easy to classify. Neo and Ventura (2012) adapted the boosting by weighting algorithm by adapting the influence of the k nearest neighbors instead of the weight of the point itself, which they do by warping the distance function.

In stacking, the out-of-sample predictions of multiple base-learners are the input for a meta-learner to produce the final predictions. The stacking method originated with Wolpert (1992), who suggested that using predictions of classifiers holds more information than the original input itself. Breiman (1996) proposed a regression meta-learner, and noted that using the cross-validated base-learner predictions is necessary to avoid over-fitting. The literature lacks studies of stacking solely kNN models, but using a combination of support vector machines, kNN, and random forest is common (Mirończuk and Protasiewicz 2019; Wang et al. 2019; Yadrintsev and Sochenkov 2019).

In this paper, we propose SUBiNN, a Stacked Uni- and Bivariate Nearest Neighbors classifier. SUBiNN is an ensemble of *k*NN classifiers that deals with the reduction of dimensionality by using only uni- and bivariate *k*NN classifiers, and allows for the interpretation of relevance of those features and their pairwise interactions by using the Lasso as a meta-learner. Training the base-learners on different subsets of features causes diversity between classifiers, which facilitates the improving effect that stacking can have. The meta-learner then gives an importance to the different base-learners in the form of coefficients, which can be used to filter out uninformative base-learners.

This paper is organized as follows. In the next section, we outline SUBiNN and the choices we made for implementation. We also show a small empirical application. In Sect. 3, we provide a pilot study on the choice of k within SUBiNN. In Sect. 4, we describe four simulation experiments to test the classification performance of SUBiNN and compare it with other nearest neighbor based classifiers as well as Random Forests and the Support Vector Machine. In Sect. 5, we compare SUBiNN with the same classification methods on a large set of benchmark data sets. We provide a discussion in terms of classification performance as well as feature selection. We conclude with some discussion.

2 SUBiNN

Stacking involves base-learners that generate cross-validated predictions and a meta-learner that takes these predictions and combines them into a final predictive model. Wolpert (1992) noted that in an ideal situation the base-learners are mutually orthogonal. Opitz and Maclin (1999) noted that the base-learners have to be both different and perform well for the ensemble to have an improving effect. In this paper, we will use *k*NN as base-learners and train them on a single feature or a pair of features, so that each base-learner uses different information. The nonnegative lasso regression will be used as a meta-learner.

We denote by $\mathbf{y}$ the n-vector with binary outcomes, $y_i \in \{0, 1\}$ for $i = 1, \ldots, n$. Furthermore, let $\mathbf{X}$ be the $n \times P$ matrix of features or predictor variables with elements x_{ip} for $p = 1, \ldots, P$.

2.1 Base-learners: *k*NN

Because distances depend on the scaling of the original variables, we first standardize the features to have zero mean and variance 1. On each combination of features X_p and X_q, with $p = 1, \ldots, P$ and $q = p, \ldots, P$, we apply *k*NN with 10 fold cross-validation. If $p = q$, we fit a univariate classifier leading to a main effect. If $p \neq q$ we fit a bivariate classifier, which indicates a pairwise interaction effect. As distance measure we use the Euclidean distance. The choice of k is important for the performance of nearest neighbor classifiers and is a manifestation of the bias variance trade-off. Small k will lead to unbiased but variable predictions, while large k will lead to biased but stable predictions. The optimal value for k depends on many factors such as the sample

size, the covariance structure of the predictors, and the class proportions (Enas and Choi 1986).

Using 10-fold cross-validation with a nearest neighbor classifier, we obtain for every combination of p and q a probabilistic prediction for every observation in the data, that is, the proportion of neighbors that are in class 1. These cross-validated predictions are collected in the $n \times R$ matrix $\mathbf{Z}$, where $R = P + P(P-1)/2$. Note that the columns of the matrix $\mathbf{Z}$ all have the same range, that is, the probabilities lie in the range 0 to 1.

2.2 Meta-learner: Lasso regression

The meta-learner takes $\mathbf{y}$ and $\mathbf{Z}$ as input. Because R might be very large, we need a meta-learner that regularizes. Furthermore, we like to have a meta-learner that results in a sparse solution. Therefore we use Lasso regression (Tibshirani 1996, 2011).

As the kNN predictions already have the same range we do not use any further standardization of these variables. Moreover, because the cross-validated predictions are already in the range of the outcome variable, we do not need an intercept in the lasso regression model. Therefore, in the Lasso regression we minimize the following loss function

$$\mathcal{L}(\beta_1, \ldots, \beta_R) = \sum_{i}^{n} \left(y_i - \sum_{r}^{R} z_{ir} \beta_r \right)^2 + \lambda \left(\sum_{r}^{R} |\beta_r| \right),$$

where we impose a nonnegativity constraint on the regression weights, that is we require $\beta_r \geq 0$. Breiman (1996) suggested this non-negativity constraint on the coefficients of the regression meta-learner. Further support for this constraint is found in Leblanc and Tibshirani (1996) and Van Loon et al. (2020). For the choice of an optimal λ value we use 10-fold cross-validation.

Linear regression, as opposed to logistic regression, might seem a weird choice when the outcome variable is dichotomous. We would like to point out that the classification boundary of a linear regression and a logistic regression are often quite similar. Assumptions for linear regression, such as independent normally distributed residuals with mean zero and constant variance, when applied to a dichotomous outcome are not tenable. The assumptions, however, mainly influence the standard errors, the resulting statistical tests, and the p-values of a linear regression, not the estimates themselves, nor the predictions. In SUBiNN, we are not interested in standard errors or test statistics for the regression coefficients. Instead, we focus on predictive accuracy (Shmueli 2010). The final predictions may go out of the range, in which case we simply set them to zero or one. The choice for a linear model instead of a logistic model is also motivated by interpretational issues: The interpretation in terms of a weighted combination on the probability scale is much simpler than on the log-odds scale.

Because we use linear regression as a meta-learner, the final outcome of SUBiNN becomes a weighted combination of the predictions of the kNN base-learners. If an estimated regression coefficient equals zero (i.e., $\hat{\beta}_r = 0$) the corresponding pair of features (p and q) have no effect on the outcome. If $p = q$, and the regression

coefficient for the corresponding base-learner is zero there is no main effect for this feature; if $p \neq q$ and the regression coefficient for the corresponding base-learner equals zero there is no interaction effect for this pair of features. The magnitude of the lasso regression coefficients provides variable importance measures.

2.3 Final model and predictions for new data

After we fitted the lasso regression models, we know which main effects and which pairwise interactions are important for predictive purposes (those with $\beta_r > 0$) and which are irrelevant (those with $\beta_r = 0$). Predictions of the SUBiNN model for a new observation with features $\mathbf{x}_+$ look into the selected subspaces. Suppose, that only the pairs of variables 1 and 2, and 3 and 7 are selected by the lasso with estimated regression weights 0.8 and 0.2, respectively. Then we compute the proportion of nearest neighbors with $y_i = 1$ in the bivariate spaces of predictors 1 and 2, and 3 and 7 for the complete training data; say they are 0.6 and 0.4, respectively. The estimated probability for the new observation to be in class $Y = 1$ is therefore $0.8 \times 0.6 + 0.2 \times 0.4 = 0.56$ and the observation would be best classified in class $Y = 1$ (if we use the threshold of 0.5).

2.4 Empirical application

To show the merits of SUBiNN, we apply it here to the classification of patients with a panic disorder on the basis of personality characteristics. For 200 subjects we have information on the big five personality scales neuroticism, extraversion, openness, agreeableness, and conscientiousness. Furthermore, we have an assessment whether this person has a panic disorder or not. The data consist of a subsample from the study reported in Spinhoven et al. (2009).

The first step in the analysis is to obtain predicted probabilities of *k*NN classifiers for each of the personality characteristics and for each pair of the personality characteristics. We use $k = 14 \approx \sqrt{200}$ for the number of neighbors. The 10-fold cross-validated predictions (probabilities) of *k*NN are collected in the matrix $\mathbf{Z}$, with 200 rows and 15 (i.e., $P + P(P-1)/2$ with $P = 5$) columns. In the second step, this matrix is used as the predictor matrix in a linear lasso regression with non-negativity constraints. We select an optimal value of the penalty parameter of the lasso through 10-fold cross-validation and with this optimal value obtain the estimated regression weights. This whole procedure is repeated 100 times and we took the median of the 100 regression weights as our final regression weights.

For this data set, four base-learners were selected. The univariate base-learners for the features openness (regression weight 0.33) and agreeableness (regression weight 0.18), and the bivariate base-learners for (1) openness and conscientiousness (regression weight 0.13) and (2) agreeableness and conscientiousness (regression weight 0.30). *k*NN classification plots are shown in Fig. 1 where for the univariate base-learners we also use two-dimensional graphs in order to get a clearer picture. Each of the four plots indicates nonlinear decision boundaries.

To obtain a prediction about a new person, we only need values of the three predictors openness, agreeableness, and conscientiousness. Such a final prediction is

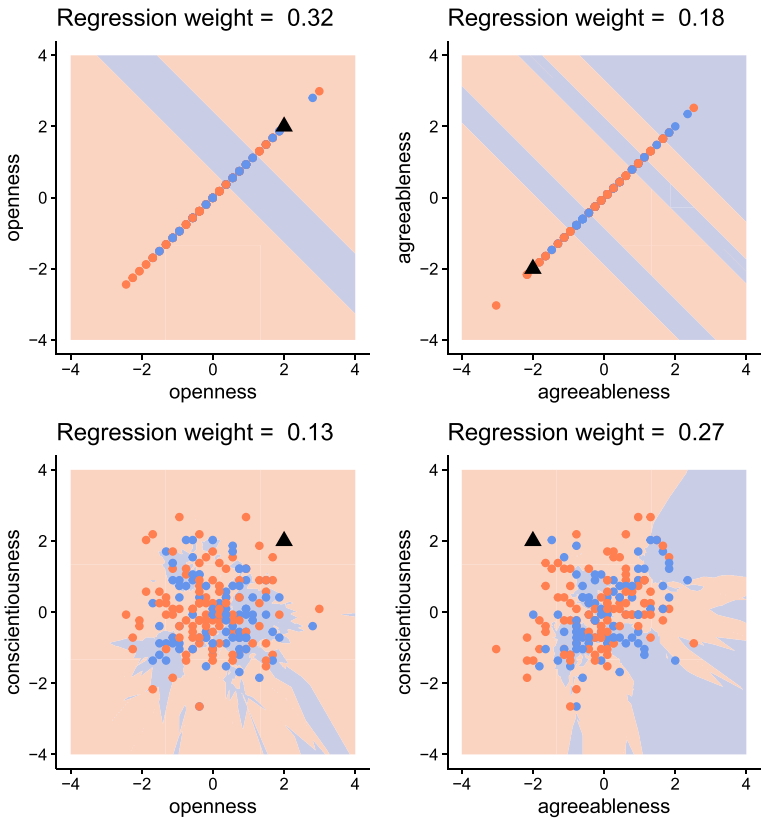

Fig. 1 Nearest Neighbor classification plots for the four selected base-learners for the panic disorder data. Red points (in black and white: light grey) indicate subjects with panic disorder, while blue points (in black and white: dark grey) indicate subjects without panic disorder. The dots indicate observations, while the background colour indicates predictions. The black triangle indicates a new observation for which we like to make a prediction. The upper two plots give the univariate base-learners, the lower two plots give the bivariate base-learners

an additive combination of the predictions of the four base-learners. Suppose, we have a new person with standardized scores of 2, −2, and 2 on these three features, respectively. This person is also indicated in Fig. 1 by the black triangle. Based on the four base-leaners, we obtain estimated probabilities for having panic disorder. These four probabilities are 0.56 (openness), 0.73 (agreeableness), 0.57 (openness − conscientiousness), and 0.71 (agreeableness − conscientiousness). To obtain the final probability of having panic disorder we take a weighted average of these four probabilities, that is

$$0.33 \times 0.56 + 0.18 \times 0.73 + 0.13 \times 0.57 + 0.30 \times 0.71 = 0.60,$$

where the weights are the estimated regression coefficients of the Lasso. We would classify this person into the panic disorder class.

3 Pilot study: the choice of *k*

The effect of k on predictions with kNN is large. Gul et al. (2016) fit their kNN model using an optimal value of k, which is derived by means of cross-validation. In SUNiNN this would add another layer of cross-validation in SUBiNN. A small pilot study was therefore performed to study the effect of k on the predictive performance, base-learners selection, and model execution time.

3.1 Setup

For this experiment we made use of two benchmark datasets. The Dystrophy dataset from the 'ipred' package (Peters and Torsten 2019) and the Diabetes dataset from the 'mlbench' package (Leisch and Dimitriadou 2010). The Dystrophy dataset consists of 194 observations on 5 features (we used age and the four serum markers), for the Diabetes dataset this is 768 observations on 8 features.

For a k of 5, 10, $\sqrt{n}$ and cross-validated optimal value, the model's performance was measured with 10-fold cross-validation over 100 replications. We recorded which base-learners were selected, the model's execution time, and the prediction accuracy. For $k = \sqrt{n}$, n refers to the number of observations in the training set within that fold. Within each fold, $\sqrt{n}$ becomes 13 for Dystrophy, and 26 for Diabetes. For $k = opt$, the function `tune` from the 'e1071' package (Meyer et al. 2019) was used. The range of possible k values was set from 1 to 20.

3.2 Results and conclusion

The results in Table 1 show that different values of k causes small differences in accuracy and sometimes large difference in the number of base-learners selected. While one could expect that a cross-validated value of k would cause better performance, this does not hold for these datasets. For the Diabetes data set, the difference between $k = \sqrt{n}$ and $k = opt$ is virtually 0, while *opt* results on average in more selected base-learners. For the Dystrophy data set, the best k is 10, thereby also selecting on average the smallest number of base-learners. The computational time for tuning k through cross-validation is substantial for both datasets.

From the results of $k = 5$, 10, and $\sqrt{n}$, the effect of k on the selected number of base-learners becomes visible. With a smaller k, the model is less stable and base-learner predictions are more variable between runs. This increases the chance that a base-learner that was fit on a non-informative feature or pair of features is selected.

The likely cause for the *non-optimal* performance of the optimal k is that this optimal value is unstable. It is calculated within another layer of cross-validation, leaving less training samples for fitting. The conclusion can be drawn that the performance gain by calculating the optimal k is not worth the added complexity. For that reason, in the remainder of this manuscript we will use $k = \sqrt{n}$. It is less arbitrary than choosing 5 or 10 because it depends on the sample size of the data, and has the property that it creates more stability in predictions and results in a lower number of base-learners selected.

Table 1 The average prediction accuracy, number of selected (non-zero coefficient) base-learners and execution time in seconds for SUBiNN

k	Accuracy	b-learners	Time
(a) *Diabetes*			
5	0.754	11.521	6.985
10	0.759	9.197	7.384
$\sqrt{n}$	0.761	6.195	9.172
opt	0.761	7.975	630.025
(b) *Dystrophy*			
5	0.892	6.914	3.248
10	0.895	5.386	3.392
$\sqrt{n}$	0.885	5.552	3.572
opt	0.889	6.331	159.266

4 Simulation studies

In this Section, we will report about four simulation studies. The first two are replications from those in Gul et al. (2016) where we added SUBiNN. Simulation experiments three and four are adaptations of the first two.

In these simulation studies, we study the classification and feature selection performance of SUBiNN, compare the performance of SUBiNN with other nearest neighbor based classifiers, that is, kNN, bagged kNN (BkNN), random kNN (RkNN), Multiple Feature Selection (MFS), and the ensemble method proposed by Gul et al. (2016), ESkNN. In our comparison, we also take two other, well known, classifiers into account: Random Forests (RF) and Support Vector Machine (SVM). Both often have high classification performance but are also considered black box techniques. We choose for these two methods as they require minimal tuning in comparison to, for example, gradient boosted trees or extreme gradient boosted trees. Bentéjac et al. (2021), for example, conclude that with the default settings of these boosted tree methods the performance is inferior to Random Forests. We like to know whether the nearest neighbor classifiers, and especially our SUBiNN approach, is competitive with these. These models are compared in terms of their classification performance. For SUBiNN, we also analyze its feature selection capabilities.

Two-class data are generated to include both non-informative features and informative features with a varying covariance and correlation structure. The inclusion of non-informative features in the first and third experiment is a means to test a model's robustness with respect to noise input features. Adapting the covariance structure of informative features for one of the two classes in the second simulation experiment, allows for the investigation of the effect of an increased difference in variances between classes. Lastly, the inclusion of a varying correlation structure between features in the fourth experiment is meant to show the effect of the difference in correlation structure between the two classes while keeping the variance the same.

4.1 Data generation for main simulation experiments

For the first two experiments the data generation follows the set up described in Gul et al. (2016). The last two experiments change the covariance structure of one of the classes.

For all experiments we generate 20 informative features from a multivariate normal distribution. For class 1 these features have a mean of 2 and a specified covariance matrix, $N(\mathbf{2}, \mathbf{\Sigma})$, where the definition of $\mathbf{\Sigma}$ varies for the four simulation experiments (see below). Class 2 has a mean of 1 and constant variance 1, $N(\mathbf{1}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix. The data generation model therefore follows that of a quadratic discriminant analysis, where the covariance matrices differ per class, and lead to non-linear classification boundaries. Non-informative features are drawn from a standard normal distribution, irrespective of the outcome class. The sample size is 1000 for all simulation experiments.

In the first part (experiment 1 and 2), the covariance matrix as specified by Gul et al. (2016) is used, $\mathbf{\Sigma} = w\mathbf{\Psi}$, where $\mathbf{\Psi}$ has elements $\psi_{p,q} = (1/2)^{|p-q|}$ for $p, q = 1, \ldots, 20$.

This covariance matrix follows an autoregressive structure, where the covariance between features declines with the distance between features. In the first experiment we use $w = 1$, and vary the number of non-informative features (0, 50, 100, 200 and 500). In the second experiment, the number of uninformative features is fixed at 50, and w is taken as 3, 5, 10, 15, and 20. All variances and covariances for the features determining class 1 are multiplied by w.

The data generation mechanism in experiment 1 without the non-informative features represents a worst case scenario for our SUBiNN model, in the sense that there is relatively good separation between the 2 classes in 20 dimensional space but that there is more and more overlap between the classes in lower dimensional subspaces. Because SUBiNN works with 1 and 2 dimensional subspaces this will have a detrimental effect. Other feature selection nearest neighbor classifiers (such as R*k*NN and MFS) will also work less good compared to *k*NN and B*k*NN that use all features. In Fig. 2 we show two bivariate scatterplots for data generated in this experiment. In the left hand side plot we show 'adjacent' features, where the correlation between the 2 features is relatively high in class 1, but low in class 2. On the right hand side we show a scatterplot of features 1 and 20, the features that have the lowest correlation in class 1, similar to the correlation between the features in class 2.

We expect that adding non-informative features does not have a large influence on SUBiNN, because separation between the classes is difficult in the two-dimensional spaces based on non-informative features. Therefore we expect the meta-learner to exclude those base-learners. In contrast, the non-informative features will have a negative effect on the classification performance of *k*NN and B*k*NN.

In experiment 2, by changing the w we obtain different variances and covariances in class 1, both are multiplied with this constant. That also means that the covariances that are zero (for example between feature 1 and features 13 till 20) remain zero for all w. Furthermore, by increasing the variances, the observations of class 2 get more

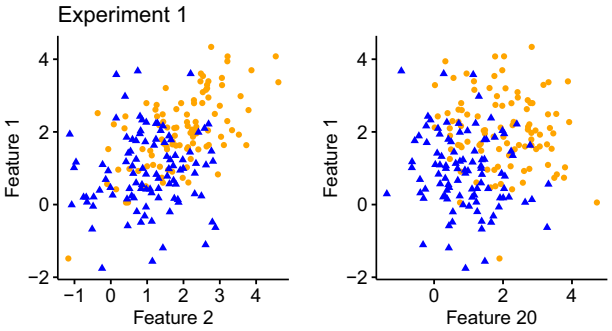

Fig. 2 Scatterplots of informative features 1 versus 2 (left hand side) and 1 versus 20 (right hand side) for the data generation model from experiment 1 (where $w = 1$), showing that in the low dimensional spaces there is quite some overlap between the observations in class 1 (indicated by •) and class 2 (indicated by ▲). The covariance between features 1 and 2 for class 1 is much higher than the covariance between features 1 and 20

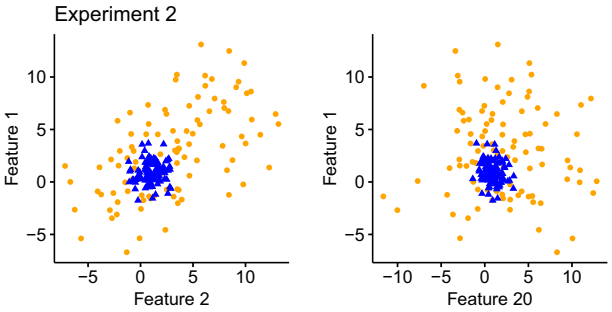

Fig. 3 Scatterplots of informative features 1 versus 2 (left hand side) and 1 versus 20 (right hand side) for the data generation model from experiment 2 where $w = 20$, showing that in the two-dimensional sub spaces the observations in class 1 (indicated by •) and class 2 (indicated by ▲) can be quite well distinguished. The variance of the features in class 1 is the determining factor. The covariance between features 1 and 2 for class 1 is much higher than the covariance between features 1 and 20

and more in the middle of the observations of class 1. See Fig. 3 for scatterplots of features 1 and 2 and features 1 and 20 for this experiment. In this case, the increased variances become meaningful and by increasing the variances of class 1 separation becomes possible. So, with increasing w values we expect SUBiNN to have increased classification performance. We do not expect the meta-learner to favor any pair of features from the informative ones; as can be seen in Fig. 3 separation between the classes is equally good in the two-dimensional spaces of features 1 and 2 as in the two-dimensional spaces of features 1 and 20.

With $\mathbf{\Sigma} = w\mathbf{\Psi}$, increasing w increases both the variance of the features and the covariances between features, causing the correlation to be roughly constant across w's. In contrast with what Gul et al. (2016) state in their article, generating data in this manner data with a varying w does not result in a differing correlation structure between variables. Neither within the first class nor when pooling the two classes together.

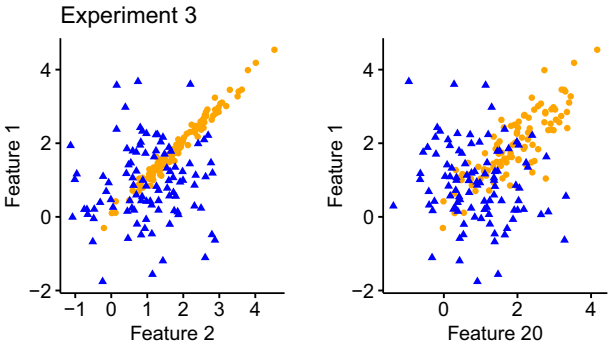

Fig. 4 Scatterplots of informative features 1 versus 2 (left hand side) and 1 versus 20 (right hand side) for the data generation model from experiment 3, showing that the observations of class 1 (indicated by •) and 2 (indicated by ▲) can be distinguished reasonably well. The correlation between the features is the determining factor. This correlation is stronger for adjacent features (left hand side plot) than for features far apart (right hand side plot)

Therefore, we designed a second set of experiments (experiments 3 and 4) where the correlation structure between the informative features is manipulated while keeping the variances of the informative features constant. For experiments 3 and 4, the covariance matrix of class 1 is adapted, and $\mathbf{\Sigma} = \mathbf{\Phi}$, with elements $\phi_{p,q} = 0.99^{w|p-q|}$.

The third experiment is again about introducing a different number of non-informative features (0, 50, 100, 200, 500), leaving $w = 1$. In the fourth and last experiment, the correlation structure of class one is adapted, by changing the value of $w = 3, 5, 10, 15, 20$. At $w = 1$, all features are extremely collinear, at $w = 20$ the largest correlation is between any two adjacent features 0.82 and virtually 0 between the features with their indices furthest apart.

In experiment 3, we again have the same situation as in experiment 1 where the separation between the two classes is good in 20 dimensional space. In the two-dimensional subspaces the separation of the classes is created by the correlation between adjacent features (see Fig. 4) . That is, features 1 and 2 have a correlation of 0.99 in class 1, whereas this correlation is 0 in class 2. Most information for distinguishing the two classes is therefore found in adjacent features. We therefore expect the meta-learner to select base-learners based on adjacent features. Like before, we do not expect non-informative features to have much influence on the classification performance of SUBiNN.

In experiment 4, in the data generating mechanism we vary values of w. Since, with increasing w the correlations become smaller, we expect that classification performance of SUBiNN decreases with increasing w. With increasing values of w it becomes more difficult to distinguish the two classes in the two-dimensional subspaces (compare Fig. 4 with Fig. 5)

For each experiment we use 100 replications. Data sets with sample size 1000 were generated and partitioned into 10 sets. The eight models are subsequently fitted on 9 of these, including all tuning and possibly internal cross-validation. Out of sample predictions were made on the left out part. That means that in total we have 100 (repetitions) times 10 sets of model estimates and corresponding prediction errors.

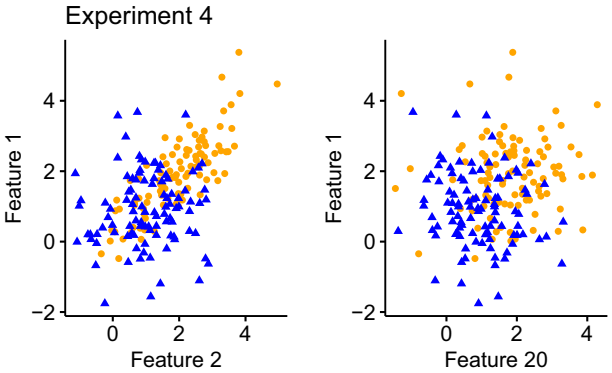

Fig. 5 Scatterplots of informative features 1 versus 2 (left hand side) and 1 versus 20 (right hand side) for the data generation model from experiment 4 wit $w = 20$, showing that with increasing value of w it becomes more difficult to distinguish the two classes (compare to Fig. 4 which shows the case for $w = 1$). Class 1 is indicate by • and class 2 by ▲

As the measure of prediction error we used the misclassification rate. The cross-validated prediction errors of all replications are averaged to obtain the final models' misclassification rate. Furthermore, for each of the 1000 results per condition we look at the number of times a particular feature or pair of features is selected by SUBiNN's meta-learner.

4.2 Software implementations

R version 3.6.1 was used (R Core Team 2019). Random samples from the multivariate normal distributions were drawn using the function `mvrnorm` from the package 'MASS' (Venables and Ripley 2002). Any added non-informative features are drawn from a standard normal distribution, using base R's `rnorm` function.

For the implementation in R of the seven models, we follow from Gul et al. (2016). For simple *k*NN, we used the function `knn` from the package 'class' (Venables and Ripley 2002). B*k*NN is implemented by fitting 1001 of these *k*NN models where the input data is sampled with replacement. The final prediction is a majority vote of the 1001 outcomes. For these two models we used $k = \sqrt{n}$.

For R*k*NN and MFS we used the `rknn` function from the 'rknn' package (Li 2015). The parameters of importance are again k, r the number of models fitted (taken again to be 1001), and *mtry*, the number of random features drawn at each fitting. Following Gul et al. (2016), we used a subset size of 1/3rd of the number of features, with a minimum of 2. For MFS we fit 1001 *k*NN model using a random draw of the features with replacement (again 1/3rd of the total number of features, with at least 2) and take the majority vote of the 1001 results.

For RF we have used the function `best.randomForest` from the package 'e1071' (Meyer et al. 2019), which also uses the function `randomForest` from the package 'randomForest' (Liaw and Wiener 2002). The automatic `best.random Forest` function does parameter selection without range specification, using 10-

fold cross-validation which is implemented with the argument *tunecontrol* where `sampling = 'cross'` and `cross = 10`.

SVM is implemented using the `ksvm` function from the package 'kernlab' (Karatzoglou et al. 2004) where the kernel is said to be `rbfdot` and the `kpar` is set to `automatic` to allow for automatic optimal parameter selection.

For ES*k*NN we made use of the package 'ES*k*NN' (Gul et al. 2015). In the first stage, 1001 models are fitted, with the number of random features taken to be 1/3 of the total number of features and the percentage of models to be selected is set to 40%. This function then returns the best 40% of models which can be used for prediction with the `predict.esknnClass` function.

SUBiNN is implemented analogous to the implementation specified in the previous chapter. All base-learners are *k*NN models fitted using the package 'class' (Venables and Ripley 2002), fit on all single features and pairwise combinations of features. The predictions produced by these base-learners are used as input for the Lasso meta-learner, using `cv.glmnet`. The `predict` function and the `glmnet` object are used to generate predictions for the test samples.

R code for all our analysis can be found on the github website of the first author (https://github.com/TElsten/subinn).

4.3 Results

4.3.1 Experiment 1: adding noise

For this experiment, we have chosen to evaluate the result of SUBiNN by comparing its accuracy and execution time to the other models. Additionally, the stability of base-learner selection is taken into consideration. The results with respect to classification performance are shown in Table 2.

When looking at the performance of SUBiNN, two things become apparent. First, the prediction accuracy as compared to the other methods is bad. Second, adding

Table 2 Average misclassification rate (mean) and standard deviations (Std) of the 8 methods for data with 20 informative and a varying number of added noise features

Features		*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN
20	Mean	0.039	0.039	0.043	0.043	0.044	0.039	0.055	0.108
	Std	0.006	0.006	0.007	0.007	0.006	0.006	0.008	0.009
20 + 50	Mean	0.050	0.049	0.048	0.047	0.048	0.051	0.058	0.108
	Std	0.008	0.008	0.007	0.007	0.007	0.008	0.009	0.009
20 + 100	Mean	0.056	0.055	0.050	0.050	0.049	0.055	0.064	0.107
	Std	0.008	0.008	0.008	0.008	0.006	0.007	0.009	0.009
20 + 200	Mean	0.067	0.064	0.054	0.054	0.051	0.058	0.074	0.106
	Std	0.008	0.008	0.007	0.008	0.006	0.008	0.009	0.008
20 + 500	Mean	0.097	0.091	0.064	0.063	0.054	0.067	0.100	0.107
	Std	0.013	0.012	0.010	0.010	0.007	0.009	0.010	0.007

non-informative features has no effect on the prediction accuracy of SUBiNN. With regard to the first outcome, as alluded in Sect. 4.1 the data generating mechanism is not beneficial for SUBiNN. The reason is that in each of the two-dimensional subspaces the classes are not well separable, while separation becomes easier when more informative features are used simultaneously. The fewer dimensions are used in such a case, the larger the overlap between the two clouds of class observations. None of the base-learners will thus produce very good predictions.

We see that other nearest neighbor classifiers that work on subsets of features (RkNN, MFS, and ESkNN) also perform less good when compared to classifiers that use all features (kNN and BkNN). The reasoning is the same, separation is easier in higher dimensional spaces in this case. The performance of RF and SVM without non-informative features is comparable to kNN and BkNN.

With regard to the second outcome, SUBiNN is able to filter out the non-informative base-learners in the meta-step and is unaffected by the introduction of more noise. This appears to be a main strength of SUBiNN. Whereas SUBiNNs classification performance remains the same irrespective of the number of non-informative features, the performance of all other classification methods deteriorates when non-informative features are included.

On average, the meta-learner used 22 base-learners, but never contained a non-informative feature. Over the replications, the number of selected base-learners fluctuated only slightly, but this fluctuation was unrelated to the number of non-informative features. Figure 6 shows the number of times each informative base-learner was used for $w = 1$ and 500 non-informative features. Remember that we used 1000 replications, i.e., the maximum frequency in the cells is 1000. The figure does not include the half-informative (pair of informative and non-informative feature) or non-informative base-learners because they were never selected by the meta-learner.

As shown by the diagonal of Fig. 6, none of the single-feature base-learners was ever selected by the meta-learner. This is not surprising, because these one-dimensional features carry less information that distinguishes the two classes. The reasoning is the same as before, the lower the dimensionality the more overlap between classes. There is no clear pattern in which base-learners are selected. Adjacent features (with correlation 0.5 within class 1, but zero in class 2) seem to play a role as well as features with a small correlation (i.e., pairs 4 or more apart that have a correlation smaller than $0.5^4 = 0.06$ within class 1). From Fig. 6, it is also clear that the feature selection was unstable between runs. The average of 22 base-learners selected are spread across the 190 possible pairs with no very clear preference. Base-learner selection does not seem to be very stable in this situation.

A major downside of the SUBiNN model is its execution time (see Table 3). The addition of 500 non-informative features resulted in $\binom{520}{2} + 520 = 135{,}460$ potential base-learners which is a massive number. On the other hand, the final predictions can be made rapidly using only 22 base-learners.

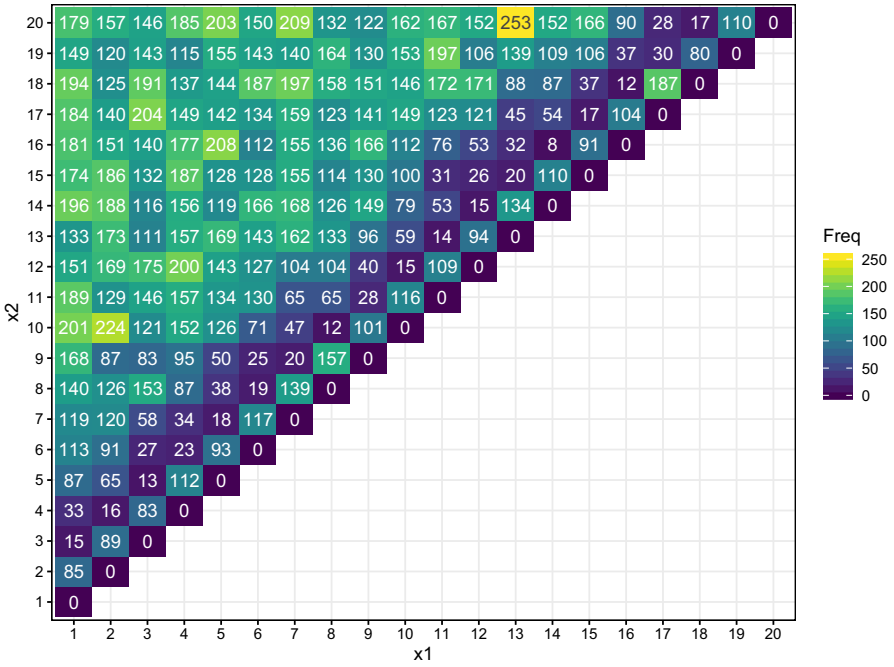

Fig. 6 Frequency of base-learner selection with low covariance data, $w = 1$, 500 uninformative features

Table 3 Average execution time in seconds of classifying data with varying number of added non-informative features

Features	*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN
20	0.0	35.6	20.8	21.6	57.3	3.7	137.2	40.2
20 + 50	0.1	103.8	36.9	37.6	154.6	8.0	210.5	409.7
20 + 100	0.2	166.1	63.2	63.0	254.8	12.5	307.6	1164.4
20 + 200	0.3	304.0	108.6	108.2	469.7	22.5	530.2	3947.4
20 + 500	2.5	2245.2	228.4	227.7	1106.4	58.2	1073.7	21367.2

4.3.2 Experiment 2: varying covariance structure

Table 4 shows the classification results with increased variance and covariances (increasing values of w) and 50 uninformative features.

The SUBiNN model performs better when the data for class 1 is generated with higher values of w. This was as expected, increasing the variances and covariances for class 1 while keeping the variance for class 2 constant, simply means increasing the differences between the two classes, making them more easily distinguishable.

SUBiNN already performs better than the other nearest neighbor classifiers when $w = 3$ and the differences become larger with larger values of w. We see that for the *k*NN and B*k*NN classifiers classification accuracy deteriorates with increasing w. For all nearest neighbor methods that take a subset of the features, performance increases

Table 4 Average misclassification rate (Mean) and standard deviations (Std) of the 8 classification methods for data with 50 uninformative features and increasing variance/covariance of features

w		*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN
3	Mean	0.226	0.227	0.200	0.200	0.049	0.091	0.181	0.137
	Std	0.014	0.014	0.013	0.012	0.007	0.009	0.011	0.008
5	Mean	0.291	0.292	0.228	0.228	0.025	0.091	0.211	0.096
	Std	0.013	0.013	0.010	0.011	0.005	0.009	0.009	0.008
10	Mean	0.332	0.335	0.198	0.198	0.005	0.079	0.166	0.035
	Std	0.012	0.012	0.008	0.008	0.002	0.008	0.009	0.005
15	Mean	0.341	0.343	0.158	0.158	0.001	0.072	0.118	0.016
	Std	0.010	0.010	0.008	0.008	0.001	0.007	0.008	0.003
20	Mean	0.340	0.343	0.128	0.128	0.001	0.067	0.085	0.007
	Std	0.010	0.010	0.008	0.008	0.001	0.007	0.007	0.003

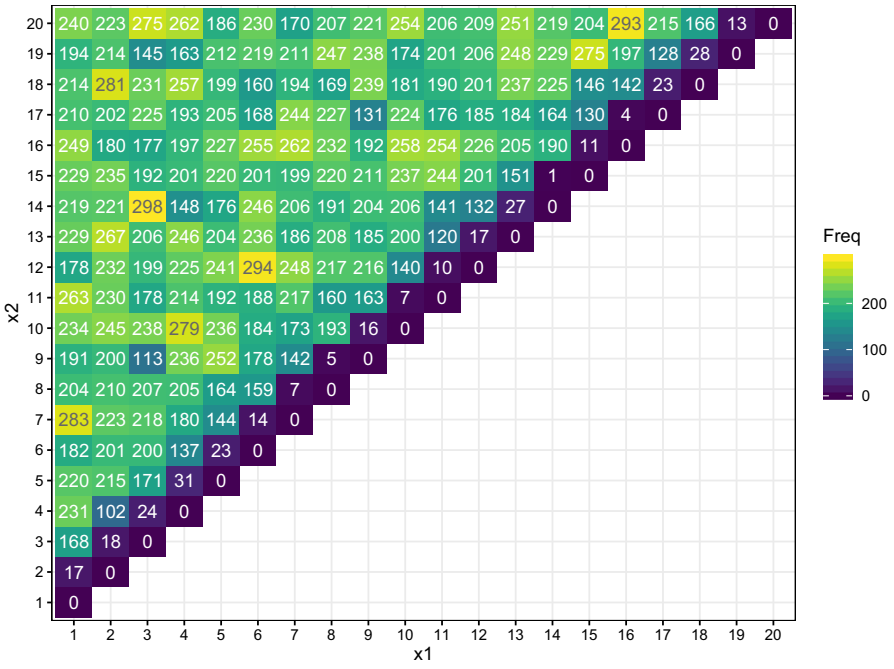

Fig. 7 Frequency of base-learner selection with high covariance data, $w = 20$, 50 uninformative features

with increasing w. SUBiNN outperforms the SVM when w becomes large (≥ 10). Random Forest (RF) is the best classifier for this situation.

In Fig. 7 we show the number of times each informative base-learner was selected for the situation where $w = 20$. The model has a preference for selecting feature pairs with low covariance in class 1 (right hand side of Fig. 3 instead of left hand side Fig. 3). The feature pairs with the highest covariance, for instance the pairs 1-2, 2-3, etc., are

Table 5 Average misclassification rate (Mean) and standard deviation (Std) of the 8 classification methods for data with 20 informative features with high correlation ($w = 1$), and an increasing number of uninformative features

Features		*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN
20	Mean	0.463	0.465	0.055	0.055	0.001	0.003	0.063	0.027
	Std	0.007	0.007	0.009	0.009	0.001	0.002	0.009	0.006
20 + 50	Mean	0.435	0.439	0.181	0.181	0.005	0.144	0.176	0.027
	Std	0.031	0.031	0.031	0.031	0.003	0.011	0.018	0.006
20 + 100	Mean	0.373	0.376	0.190	0.190	0.016	0.161	0.191	0.027
	Std	0.039	0.041	0.028	0.028	0.007	0.012	0.018	0.006
20 + 200	Mean	0.324	0.323	0.187	0.186	0.062	0.174	0.202	0.027
	Std	0.046	0.048	0.032	0.032	0.020	0.014	0.020	0.007
20 + 500	Mean	0.285	0.281	0.182	0.182	0.120	0.186	0.218	0.027
	Std	0.036	0.037	0.021	0.022	0.011	0.011	0.017	0.006

not often selected. Again, SUBiNN did not select single-feature base-learners, because such a single feature can not capture the quadratic nature of the decision boundaries. The meta-learner makes use of more base-learners as w increases, ranging from on average 22 at $w = 1$ to 36 at $w = 20$. Again, no uninformative base-learners were ever included by the meta-learner. Figure 7 shows that the most favoured base-learners are selected up to 300 out of 1000 times.

4.3.3 Experiment 3: correlation and noise

In this experiment we generated data with highly collinear features in class 1 and increasing numbers of non-informative features. The results are shown in Table 5.

Like in experiment 1, SUBiNN is insensitive to the non-informative features. The classification performance is equally good, no matter the number of non-informative features. Furthermore, the classification performance of SUBiNN is better than that of all other nearest neighbor classifiers. Comparing SUBiNN with SVM, we see that in the case of no uninformative features SVM performs better, but with uninformative features SUBiNN outperforms the SVM. A similar patter can be observed when comparing SUBiNN to RF, although in that case the number of non-informative features should be large (≥ 200) before SUBiNN outperforms RF.

Figure 8 shows that the meta-learner preferred base-learners trained on the most highly-correlated pairs of features in class 1 (cf., Fig. 4). SUBiNN used on average 21.5 base-learners. The non-informative base-learners were never included in the final model. Like before, no base-learner fit on a single feature was ever included in the final model.

4.3.4 Experiment 4: varying correlation structure

Table 6 shows the results with different levels of feature correlation, which decreases by increasing w. The number of uninformative features is kept constant at 50. SUBiNN

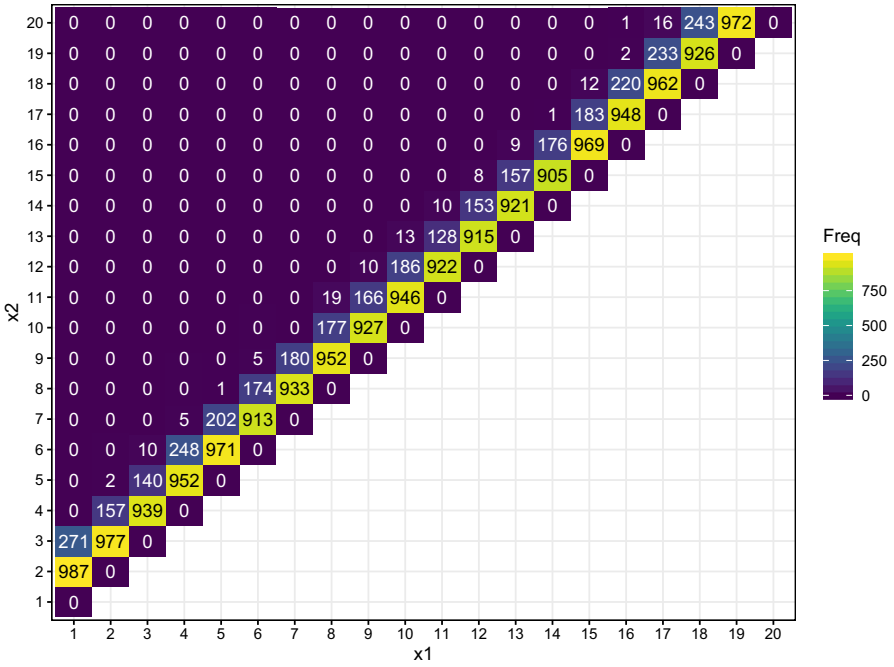

Fig. 8 Frequency of base-learner selection with highly correlated data, $w = 1$, 500 uninformative features

Table 6 Average misclassification rate (Mean) and standard deviation (Std) of the 8 models for data with 20 informative features of decreasing correlation as w increases, and 50 uninformative features

w		kNN	BkNN	RkNN	MFS	RF	SVM	ESkNN	SUBiNN
3	Mean	0.372	0.374	0.156	0.155	0.049	0.140	0.166	0.049
	Std	0.044	0.047	0.023	0.023	0.012	0.011	0.017	0.007
5	Mean	0.310	0.310	0.132	0.132	0.081	0.135	0.154	0.064
	Std	0.037	0.039	0.015	0.015	0.006	0.010	0.013	0.007
10	Mean	0.207	0.203	0.106	0.107	0.087	0.120	0.130	0.082
	Std	0.033	0.033	0.010	0.009	0.007	0.010	0.013	0.007
15	Mean	0.143	0.139	0.094	0.094	0.081	0.107	0.109	0.091
	Std	0.023	0.022	0.008	0.008	0.007	0.008	0.011	0.007
20	Mean	0.117	0.114	0.086	0.087	0.076	0.100	0.100	0.097
	Std	0.019	0.019	0.009	0.009	0.007	0.009	0.011	0.008

outperforms all other nearest neighbor classifiers in most situations. The performances becomes roughly equal when $w = 20$. SUBiNN outperforms the SVM for all w, excepts $w = 20$. The performance of SUBiNN is comparable to that of RF, although at larger values of w, RF seems to do a little better.

Similar to experiment 3, SUBiNN has a preference for base-learners consisting of pairs of features with the highest correlation, see Fig. 9 which shows the number

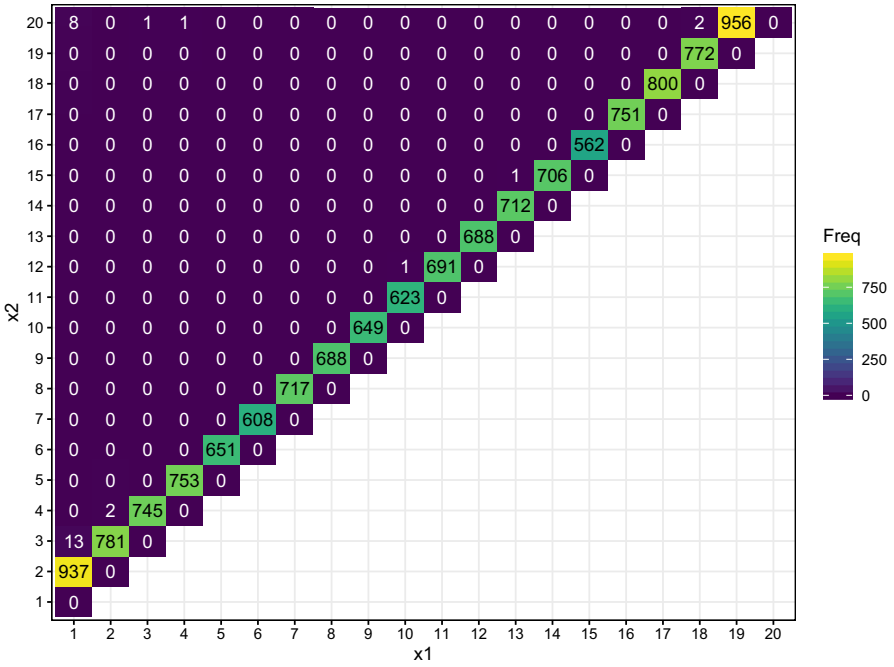

Fig. 9 Frequency of base-learner selection with low-correlation data, $w = 20$, 50 uninformative features

of features selected at $w = 20$. Comparing the number of base-learners selected by SUBiNN, we see that as the correlation decreases the number of base-learners also decreases, ranging from on average 21 when $w = 1$ to on average 14 when $w = 20$.

4.4 Conclusion

The four simulation experiments with a varying number of added uninformative features and a varying covariance and correlation structure have shown that in the right conditions, SUBiNN can perform well. While in experiment 1 SUBiNN performs badly, in experiments 2, 3 and 4 SUBiNN outperformed the other nearest neighbor classifiers. Furthermore, SUBiNN filters out successfully all uninformative features, which is a great advantage of SUBiNN. The latter result is probably due to the choice of a relatively high value for k. Compared to SVM, SUBiNN performed better in experiments 3 and 4 and also in experiment 2 when $w \geq 10$. Compared to RF, SUBiNN often performs worse, except for experiment 3 with large numbers of non-informative features.

5 Benchmark data study

We used 22 benchmark data sets to compare the same set of models as in the simulation studies. The eight models will be compared by their classification accuracy and feature selection capabilities.

5.1 Data

Table 7 specifies the datasets' sample size, number of features, and the resulting number of SUBiNN base-learners. For all datasets, cases with missing values were omitted. Furthermore, in accordance with the procedure from Gul et al. (2016), the maximum number of observations used was 1000, if this was a subset of the full dataset the 1000 rows were sampled at random.

Both categorical and nominal features were transformed into numerical features. If the nominal feature had more than 2 levels, they were transformed into indicator variables, one for each level of the feature. The Cylinder Bands dataset forms an exception, for this dataset the nominal features with more than 2 levels were omitted, following Gul et al. (2016).

Table 7 Specification of benchmark datasets

Name	Sample size	Features	Num/Cat/Nom	B-learners
Haberman (haber)	306	3	3/0/0	6
Mammography (mammo)	830	4	1/1/2	66
Transfusion (transf)	748	4	4/0/0	10
Phoneme (phone)	1000	5	5/0/0	15
Liver disorders[a] (bupa)	345	5	5/0/0	15
Appendicitis (appen)	106	7	7/0/0	28
Dystrophy (dystr)	194	7	6/0/1	28
Pima Indians diabetes (diabe)	768	8	8/0/0	36
Biopsy (biops)	683	9	9/0/0	45
SAHeart (heart)	462	9	8/1/0	45
Indian liver (Indian)	579	10	9/0/1	55
Solar flare[b] (solar)	323	10	7/0/3	231
Credit approval[c] (credit)	653	15	7/0/8	820
House votes (house)	232	16	0/16/0	136
Hepatitis (hepat)	80	19	6/12/1	190
Two norm (twono)	1000	20	20/0/0	210
Cylinder bands[d] (bands)	365	24	18/0/5	300
German credit[e] (german)	1000	24	24/0/0	300
Breast cancer (wpbc)	194	33	33/0/0	561
Sonar (sonar)	208	60	60/0/0	1830
Glaucoma (glauc)	153	66	66/0/0	2211
Musk (musk)	476	166	166/0/0	13861

The dataset name links to the source, the abbreviation in brackets will be used in subsequent tables.
[a]Used outcome is dichotomized $x_6 > 5$, x_7 is omitted. Mcdermott and Forsyth (2016) wrote an article on the frequent improper use of this dataset.
[b]Used outcome is dichotomized $x_{11} > 0$. Indicator variables were created for the 3 nominal features.
[c]The cost matrix was not used.
[d]Nominal features with more than 2 levels were omitted.
[e]The numerical version was used

5.2 Models

The software implementation of the eight models is analogous to the implementation in the simulation studies. All eight models are fit on the 22 benchmark datasets by means of 10-fold cross-validation, for 100 replications.

5.3 Results

Table 8 shows the cross-validated classification error of the models on the 22 benchmark datasets. The last 'B-learners' column denotes the average number of base-learners that were used by SUBiNN's meta-learner. In the following we first discuss the classification accuracy and thereafter provide more detailed results about the selection of base-learners.

When we compare SUBiNN against the other nearest neighbor classifiers, we see that for 8 data sets it performs the best, for 9 data sets it performed as intermediate, and for 5 data sets it performed worst. We ordered the rows in Table 8 in those three groups, such that in the upper part the data sets appear where SUBiNN performs better than the other nearest neighbor methods, in the middle part its performance is intermediate, and in the lower part SUBiNN performs worse than the other nearest neighbor classifiers. The three parts are indicated by the dashed lines.

For the 8 data sets that SUBiNN performed best among the nearest neighbor classifiers, RF performs a little better for 7 of the 8 data sets, although the differences are in 5 cases within one standard deviation. For 1 data set, the Glaucoma dataset, SUBiNN outperforms all other methods, including RF.

For the 5 data sets that SUBiNN performed worst among the nearest neighbor classifiers, in one case RF performed slightly worse. One of the five datasets is the solar data, for which all classification methods basically perform equally good. Only with the Twonorm and Cylinder Bands datasets did SUBiNN perform substantially worse than the other models. The Twonorm dataset is a simulated data set with characteristics similar to the datasets used in the first simulation experiment, where there is relatively good separation in the high dimensional space, but quite some overlap of the classes in every two-dimensional subspace. SUBiNN also makes considerably more errors than the other models for the Cylinder Bands dataset. This dataset has a large number of nominal features which appears problematic for SUBiNN because there is not much information in the two-dimensional subspaces of this problem.

The only dataset for which SUBiNN obtains a considerably better performance is the Glaucoma dataset. It does so while using on average only 5.6 base-learners, a great reduction in dimensionality from the complete dataset of 66 features. *k*NN and ES*k*NN have the highest missclassification rate for this dataset. An explanation for this can be found in Peters et al. (2003), who used this dataset in their study to the applicability of indirect variables in classifying glaucoma. The dataset contains three of these indirect variables, which are derived from clinical expert judgement and prior analysis. Along with those 3 *indirect* variables are the 63 measurement (*direct*) variables. For SUBiNN, these features resulted in 2211 potential base-learners. In almost every run, base-learners (64), (64, 66), and (65, 66) which correspond to the

Table 8 Misclassification rate (with standard deviations in smaller font) of the 8 classification methods on different benchmark datasets

	*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN	B-learners
Mammo	0.212	0.209	0.196	0.196	0.193	0.204	0.198	0.195	5.934
	0.005	0.005	0.003	0.003	0.003	0.005	0.005	0.003	
Phone	0.202	0.204	0.202	0.202	0.143	0.185	0.211	0.177	5.908
	0.005	0.004	0.004	0.005	0.005	0.005	0.007	0.004	
Dystr	0.149	0.147	0.141	0.141	0.109	0.110	0.129	0.115	5.568
	0.009	0.008	0.009	0.009	0.008	0.008	0.013	0.007	
Diabe	0.253	0.249	0.261	0.261	0.235	0.240	0.247	0.241	5.957
	0.006	0.006	0.005	0.005	0.005	0.006	0.006	0.006	
Credi	0.205	0.200	0.177	0.177	0.122	0.148	0.136	0.136	9.011
	0.006	0.005	0.005	0.004	0.004	0.005	0.006	0.000	
Hepat	0.149	0.137	0.148	0.149	0.112	0.139	0.417	0.131	7.090
	0.018	0.018	0.011	0.010	0.015	0.017	0.093	0.019	
Germa	0.272	0.274	0.296	0.296	0.235	0.242	0.271	0.264	7.052
	0.004	0.004	0.002	0.002	0.005	0.005	0.008	0.005	
Glauc	0.188	0.178	0.177	0.177	0.095	0.150	0.182	0.068	5.587
	0.014	0.012	0.011	0.011	0.007	0.008	0.019	0.007	
Haber	0.255	0.252	0.269	0.269	0.279	0.268	0.259	0.258	3.574
	0.007	0.008	0.008	0.008	0.009	0.008	0.011	0.009	
Trans	0.208	0.206	0.231	0.231	0.245	0.210	0.234	0.232	4.902
	0.005	0.005	0.004	0.004	0.006	0.004	0.006	0.005	
Bupa	0.210	0.212	0.232	0.232	0.207	0.196	0.226	0.221	5.495
	0.006	0.005	0.005	0.005	0.009	0.006	0.011	0.009	
Biops	0.030	0.031	0.029	0.029	0.027	0.040	0.034	0.031	14.508

Table 8 continued

	*k*NN	B*k*NN	R*k*NN	MFS	RF	SVM	ES*k*NN	SUBiNN	B-learners
	0.002	0.001	0.000	0.000	0.002	0.002	0.003	0.002	
Heart	0.288	0.288	0.305	0.306	0.313	0.287	0.302	0.295	6.333
	0.009	0.007	0.007	0.007	0.010	0.008	0.013	0.009	
India	0.306	0.306	0.281	0.281	0.294	0.291	0.290	0.291	8.300
	0.008	0.008	0.003	0.003	0.010	0.005	0.010	0.007	
House	0.223	0.219	0.187	0.187	0.221	0.203	0.225	0.201	4.885
	0.016	0.015	0.004	0.004	0.013	0.009	0.034	0.014	
Sonar	0.276	0.279	0.259	0.258	0.161	0.174	0.227	0.235	12.873
	0.012	0.013	0.011	0.012	0.013	0.013	0.019	0.016	
Musk	0.221	0.219	0.188	0.187	0.101	0.106	0.177	0.207	18.288
	0.008	0.007	0.008	0.007	0.007	0.007	0.013	0.011	
Appen	0.126	0.126	0.137	0.136	0.132	0.134	0.137	0.139	4.417
	0.011	0.008	0.006	0.006	0.007	0.009	0.016	0.013	
Solar	0.173	0.172	0.174	0.174	0.176	0.171	0.174	0.175	8.225
	0.002	0.002	0.001	0.001	0.004	0.003	0.003	0.002	
Twono	0.026	0.026	0.027	0.027	0.033	0.025	0.044	0.111	23.024
	0.002	0.002	0.002	0.002	0.003	0.002	0.005	0.007	
Bands	0.320	0.316	0.319	0.319	0.234	0.290	0.339	0.366	11.093
	0.012	0.009	0.008	0.008	0.013	0.009	0.018	0.011	
Wpbc	0.241	0.245	0.239	0.239	0.208	0.232	0.233	0.246	9.157
	0.008	0.008	0.004	0.004	0.010	0.008	0.014	0.012	

The B-learners column refers to the average number of base-learners that were used by SUBiNN

indirect measurements, were used in the final model. The meta-learner barely made use of any of the direct measurement variables but chose combinations of the three indirect variables. Note that for this data set the univariate base-learner for feature 64 was selected in every run. So, whereas in the simulation study univariate base-learners were never selected, application of SUBiNN to this data set shows that indeed the univariate base-leaners can have a contribution to the final model.

Considering, more generally, the choice of base-learners it seems that when SUBiNN performs relatively good, it uses a small number of base-learners, as can be witnessed in the last column in Table 8.

6 Conclusion and discussion

In this paper we proposed SUBiNN, a stacked uni- and bivariate nearest neighbor classifier. Researchers are often interested in the effects of predictor variables/features on an outcome variable or in the interaction effect of features on an outcome variable. Usually, such main effects and interaction effects are included in a logistic regression model. The logistic regression model, however, assumes that these effects are linear effects which is a rather strong assumption. Nearest neighbor analysis imposes no functional form, but does not have the capability to distinguish main and interaction effects. Moreover, nearest neighbor analysis is sensitive to the dimensionality of the feature space. In SUBiNN we built an ensemble method by combining nearest neighbor analyses in one and two-dimensional subspaces, thereby overcoming the curse of dimensionality and at the same time enabling identification of main and pairwise interaction effects without imposing a functional form on these effects.

In this respect, SUBiNN works similarly to generalized additive models that also focus on sums of one and possible two-dimensional smoothers (Hastie and Tibshirani 1990; Wood 2017). In SUBiNN, we finally have a sum of one and two-dimensional nearest neighbor classifiers, where the number of neighbors can be used to control the smoothness in a way.

In nearest neighbor classifiers the number of neighbors, k, is important. Small values k lead to unbiased models with high variance, while high values of k lead to biased but stable models. In a small experimental study we investigated the choice of k for SUBiNN and found no big differences in classification performance. Therefore, we choose to use $k = \sqrt{n}$ in the simulation studies and benchmark study. When applying SUBiNN to an empirical dataset a researcher might choose to use k as a tuning parameter, fitting the whole procedure with different values of k and selecting the one with best classification performance. Another strategy is to define an even larger set of base-learners where for every combination of features p and q, base-learners are trained with varying value of k. The meta-learner then makes a selection out of this enlarged set of base-learners.

In the simulation studies, we compare SUBiNN to other nearest neighbor classifiers and two methods that usually classify very accurately, Random Forests and Support Vector Machines. We generated data following a quadratic discriminant analysis model where we varied the covariance matrices and the number of uninformative features. One strong result for SUBiNN is its robustness with respect to noise features. SUBiNN

never selected a base-learner involving an uninformative feature. In comparison to other NN classifiers, SUBiNN performed relatively good if there exist pairs of features that separate the observations of the two classes quite well, such as in experiment 2 and 3. When the separating information is mainly available in higher dimensional subspaces the performance of SUBiNN is not that good (such as in experiment 1).

The comparison in terms of classification performance of SUBiNN with RF and SVM point out that overall RF is the best classifier. There have been further improvements on ensembles of classification trees that we did not study, such as (extreme) gradient boosted trees. They might even be a bit better in terms of classification performance, but require more tuning in order to outperform RF. Such tuning also carries the risk of overfitting. RF and other classification tree ensembles, however, are often considered black-box techniques (similar to SVM). Recently, there have been advances in developing interpretational tools for RF and other black box techniques (Sies and Van Mechelen 2020; Ribeiro et al. 2016). Others have adapted ensemble methods with trees to increase interpretability (Meinshausen 2010) and searched for optimal tree ensembles (Khan et al. 2020, 2021). We note that both in our simulation experiments as well as in the benchmark data set, there was at least one dataset or condition for which SUBiNN outperformed RF.

A clear disadvantage of SUBiNN is its considerable execution time (see Table 3) when the number of (uninformative) features was increased. As a result of the sparsity created by Lasso, this long execution time only applies to the training stage and not to the final task of predicting the outcome. High performance computing can be used to parallelize computations, that is, every base learner can be trained on a different computer/core. Methods to approximate the *k*NN distance mapping could be implemented, such as Locality Sensitive Hashing (Andoni and Indyk 2008).

We focussed on binary classification. Many classification problems have more than two classes and *k*NN is well suitable for such situations. To generalize SUBiNN for multiclass classification, the only thing that is needed is to have a good meta-learner. Two options are regularized linear discriminant analysis (Clemmensen et al. 2011; Trendafilov and Jolliffe 2007) or polytomous logistic regression with lasso-type penalty (Friedman et al. 2010). Further research is needed.

The choice for only using univariate and bivariate base-learners stems from the wish to identify the main- and interaction effects of the input. Base-learners could be fit on subsets of three, four, or more features, thereby introducing three-way, four-way, and higher-order interactions of features. However, in practice three-way interaction term are often difficult to interpret, let alone interactions of a higher order. Moreover, the two-dimensional subspaces with classification boundaries can be relatively easy visualized, which becomes more difficult for three or higher-dimensional subspaces.

In all our analysis we used the Euclidean distance to find the neighbors. The Euclidean distance is a simple choice, but might not be an optimal choice in all situations. Other distance measures have been developed, see Cox and Cox (2000) for many types of proximity measures for different kind of features and Gower (1971) for a general distance measure for features with different characteristics. Another choice we made is to standardize the features to have zero mean and unit variance. Although distance measures need some kind of standardized features, other choices could result

in different conclusions. Different scaling methods have been studied mainly in the context of cluster analysis (Schaffer and Green 1996; Steinley 2004).

We choose to use Lasso linear regression as the meta-learner. Lasso tends to arbitrarily select a single base-learner among a group of highly correlated base-learners. Zou and Hastie (2005) note that an Elastic Net regularization has the potential of alleviating this problem by either including or excluding groups of highly correlated variables.

In sum, we proposed and evaluated a new stacking ensemble learner based on univariate and bivariate kNN classifiers. When the meta-learner selects a small number of base-learners the results can be understood in terms of nonlinear main effects and two-variable interaction effects. SUBiNN often performs better than other NN-based classification methods and under certain conditions even outperformed Random Forests.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable feedback and suggestions.

References

Andoni A, Indyk P (2008) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun ACM 51(1):117

Bay SD (1999) Nearest neighbor classification from multiple feature subsets. Intell Data Anal 3(3):191–209

Bentéjac C, Csörgő A, Martínez-Muñoz G (2021) A comparative analysis of gradient boosting algorithms. Artif Intell Rev 54(3):1937–1967

Breiman L (1996) Stacked regressions. Mach Learn 24(1):49–64

Clemmensen L, Hastie T, Witten D, Ersbøll B (2011) Sparse discriminant analysis. Technometrics 53(4):406–413

Cox T, Cox M (2000) Multidimensional scaling, 2nd edn. CRC monographs on statistics and applied probability. CRC Press, Chapman & Hall, Boca Raton

Dietterich T (2000) An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Mach Learn 40(2):139–157

Domeniconi C, Yan B (2004) Nearest neighbor ensemble. In: Proceedings of the 17th international conference on pattern recognition, 2004. ICPR 2004

Enas GG, Choi SC (1986) Choice of the smoothing parameter and efficiency of k-nearest neighbor classification. Comput Math Appl 12(2):235–244

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22

García-Pedrajas N, Ortiz-Boyer D (2009) Boosting k-nearest neighbor classifier by means of input space projection. Expert Syst Appl 36(7):10570–10582

Gower JC (1971) A general coefficient of similarity and some of its properties. Biometrics 27(4):857–871

Gul A, Perperoglou A, Khan Z, Mahmoud O, Adler W, Miftahuddin M, Lausen B (2015) ESKNN: ensemble of subset of K-nearest neighbours classifiers for classification and class membership probability estimation. R package version 1

Gul A, Perperoglou A, Khan Z, Mahmoud O, Miftahuddin M, Adler W, Lausen B (2016) Ensemble of a subset of KNN classifiers. Adv Data Anal Classif 12(4):827–840
Hassanat AB, Abbadi MA, Altarawneh GA (2014) Solving the problem of the k parameter in the KNN classifier using an ensemble learning approach. Int J Comput Sci Inf Secur 12(8):33–39
Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning. Springer, Berlin
Hastie TJ, Tibshirani RJ (1990) Generalized additive models, vol 43. CRC monographs on statistics and applied probability. CRC Press, Chapman & Hall, Boca Raton
Karatzoglou A, Smola A, Hornik K, Zeileis A (2004) kernlab—an S4 package for kernel methods in R. J Stat Softw 11(9):1–20
Khan Z, Gul A, Perperoglou A, Miftahuddin M, Mahmoud O, Adler W, Lausen B (2020) Ensemble of optimal trees, random forest and random projection ensemble classification. Adv Data Anal Classif 14(1):97–116
Khan Z, Gul N, Faiz N, Gul A, Adler W, Lausen B (2021) Optimal trees selection for classification via out-of-bag assessment and sub-bagging. IEEE Access 9:28591–28607
Leblanc M, Tibshirani R (1996) Combining estiamates in regression and classification. J Am Stat Assoc 91(436):1641
Leisch F, Dimitriadou E (2010) *mlbench: Machine learning benchmark problems*. R package version 2.1-1
Li S (2015) *rkNN: Random KNN classification and regression*. R package version 1.2-1
Li S, Harner EJ, Adjeroh DA (2011) Random KNN feature selection—a fast and stable alternative to random forests. BMC Bioinform 12(1):450
Liaw A, Wiener M (2002) Classification and regression by randomforest. R News 2(3):18–22
Mcdermott J, Forsyth RS (2016) Diagnosing a disorder in a classification benchmark. Pattern Recognit Lett 73:41–43
Meinshausen N (2010) Node harvest. Ann Appl Stat 4(4):2049–2072
Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2019) e1071: Misc functions of the department of statistics, probability theory groups (Formerly: E1071), TU Wien. R Package version 1.7.-3
Mirończuk MM, Protasiewicz J (2019) Recognising innovative companies by using a diversified stacked generalisation method for website classification. Appl Intell 50(1):42–60
Neo TKC, Ventura D (2012) A direct boosting algorithm for the k-nearest neighbor classifier via local warping of the distance metric. Pattern Recognit Lett 33(1):92–102
Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11:169–198
Peters A, Lausen B, Michelson G, Gefeller O (2003) Diagnosis of glaucoma by indirect classifiers. Methods Inf Med 42(01):99–103
Peters A, Torsten H (2019) ipred: Improved predictors. R package version 0.9-9
R Core Team (2019) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
Ribeiro MT, Singh S, Guestrin C (2016) "Why should I trust you?" explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp 1135–1144
Schaffer C, Green P (1996) An empirical comparison of variable standardization methods in cluster analysis. Multivar Behav Res 31(2):149–167
Shmueli G (2010) To explain or to predict. Stat Sci 25:289–310
Sies A, Van Mechelen I (2020) C443: a methodology to see a forest for the trees. J Classif 37:730–753
Spinhoven P, De Rooij M, Heiser W, Smit JH, Penninx BW (2009) The role of personality in comorbidity among anxiety and depressive disorders in primary care and specialty care: a cross-sectional analysis. Gen Hosp Psychiatry 31(5):470–477
Steinley D (2004) Standardizing variables in k-means clustering. In: *Classification, clustering, and data mining applications*, pp 53–60. Springer, Berlin Heidelberg
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Stat Methodol) 58(1):267–288
Tibshirani R (2011) Regression shrinkage and selection via the lasso: a retrospective. J R Stat Soc Ser B (Stat Methodol) 73(3):273–282
Trendafilov NT, Jolliffe IT (2007) Dalass: variable selection in discriminant analysis via the lasso. Comput Stat Data Anal 51(8):3718–3736
Van Loon W, Fokkema M, Szabo B, De Rooij M (2020) Stacked penalized logistic regression for selecting views in multi-view learning. Inf Fus 61:113–123

Venables WN, Ripley BD (2002) Modern applied statistics with S, 4th edn. Springer, New York (ISBN 0-387-95457-0)
Wang Q, Zhao D, Wang Y, Hou X (2019) Ensemble learning algorithm based on multi-parameters for sleepstaging. Med Biol Eng Comput 57(8):1693–1707
Wolpert DH (1992) Stacked generalization. Neural Netw 5(2):241–259
Wood SN (2017) Generalized additive models: an introduction with R. Chapman and Hall/CRC Press, Boca Raton
Yadrintsev VV, Sochenkov IV (2019) The hybrid method for accurate patent classification. Lobachevskii J Math 40(11):1873–1880
Zhou Z-H, Yu Y (2005) Adapt bagging to nearest neighbor classifiers. J Comput Sci Technol 20(1):48–54
Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B (Stat Methodol) 67(2):301–320