Received July 11, 2020, accepted July 15, 2020, date of publication July 17, 2020, date of current version July 29, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3010099*

# A $k$-Nearest Neighbours Based Ensemble via Optimal Model Selection for Regression

**AMJAD ALI[1], MUHAMMAD HAMRAZ[1], POOM KUMAM[2,5], (Member, IEEE), DOST MUHAMMAD KHAN[1], (Member, IEEE), UMAIR KHALIL[1], MUHAMMAD SULAIMAN[3], AND ZARDAD KHAN[1,4]**

[1]Department of Statistics, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan
[2]KMUTT Fixed Point Research Laboratory, Department of Mathematics, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), Bangkok 10140, Thailand
[3]Department of Mathematics, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan
[4]Department of Mathematical Sciences, University of Essex, Colchester CO4 3SQ, U.K.
[5]Department of Medical Research, China Medical University Hospital, China Medical University, Taichung City 40402, Taiwan

Corresponding authors: Zardad Khan (zardadkhan@awkum.edu.pk) and Poom Kumam (poom.kum@kmutt.ac.th)

This work was supported by the Center of Excellence in Theoretical and Computational Science (TaCS-CoE), King Mongkut's University of Technology Thonburi (KMUTT).

**ABSTRACT** Ensemble methods based on $k$-NN models minimise the effect of outliers in a training dataset by searching groups of the $k$ closest data points to estimate the response of an unseen observation. However, traditional $k$-NN based ensemble methods use the arithmetic mean of the training points' responses for estimation which has several weaknesses. Traditional $k$-NN based models are also adversely affected by the presence of non-informative features in the data. This paper suggests a novel ensemble procedure consisting of a class of base $k$-NN models each constructed on a bootstrap sample drawn from the training dataset with a random subset of features. In the $k$ nearest neighbours determined by each $k$-NN model, stepwise regression is fitted to predict the test point. The final estimate of the target observation is then obtained by averaging the estimates from all the models in the ensemble. The proposed method is compared with some other state-of-the-art procedures on 16 benchmark datasets in terms of coefficient of determination ($R^2$), Pearson's product-moment correlation coefficient ($r$), mean square predicted error ($MSPE$), root mean squared error ($RMSE$) and mean absolute error ($MAE$) as performance metrics. Furthermore, boxplots of the results are also constructed. The suggested ensemble procedure has outperformed the other procedures on almost all the datasets. The efficacy of the method has also been verified by assessing the proposed method in comparison with the other methods by adding non-informative features to the datasets considered. The results reveal that the proposed method is more robust to the issue of non-informative features in the data as compared to the rest of the methods.

**INDEX TERMS** $k$-NN, random $k$-NN, regression, stepwise model selection, ensemble learning, non-informative features.

## I. INTRODUCTION

Supervised learning is a machine learning task dealing with functions that map an input to an output based on samples in pairs of input and outputs. $k$-nearest neighbours ($k$-NN) algorithm is considered as one of the top ten supervised learning methods used for classification and regression [1]–[3]. It uses a set of $k$-nearest observations to decide on the response value of a test case thus trying to minimize the effect of outliers in a training dataset. This algorithm is fast, simple and easy to understand, robust to outliers in training data and effective if training data is large [4], [5]. Despite being simple, $k$-NN gives viable results and in some cases even beats other complex learning methods. However, $k$-NN suffers from various data related issues, for example, non-informative features and noise in the data.

Combination of individual $k$-NN models in conjunction with randomization technique(s) further improves prediction accuracy. Randomization techniques usually involves taking random samples from the training data and/or the given feature set for building the base $k$-NN models. This increases diversity in the base models reducing their chances

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Xiao.

to repeat the same error [6]–[9]. Several ensemble techniques have been proposed based on the above notion. Examples of the algorithms are bootstrap aggregated $k$-NN [10], random $k$-NN [11], ensemble of subset of $k$-NN [12], etc. Majority of the $k$-NN ensembles use simple or weighted mean of the target variable values in the neighbourhood of the test point determined by each of the base models. Final estimate is computed by pooling all base $k$-NN models results. Such an estimate is highly sensitive to non-informative features and scale of the data [8].

This paper presents the idea of an ensemble of $k$-NN models, the optimal $k$-NN ensemble (O$k$-NN-E) based on fitting stepwise regression for observations in the test data. The stepwise model selection minimizes the effect of the non-informative features in the data on the response variable. The $k$-NN models are constructed on bootstrap samples taken from the training data along with taking a random subset of features. Sequential backward and/or forward model selection based on the $k$-nearest neighbours identified by each $k$-NN model and the selected feature subset is used for predicting the test sample. The final estimate of the test point is obtained by averaging the results of all the optimal $k$-NN models in the ensemble. A total of 16 bench mark datasets have been used to assess the performance of the proposed O$k$-NN-E. For further assessment, non-informative features generated random from uniform distribution over the interval $[0, 1]$ are added to the datasets used in the paper. The number of non-informative features added to each of the dataset is the same as the original number of features in the data. Coefficient of determination ($R^2$), Pearson's product moment correlation coefficient ($r$), mean square predicted error ($MSPE$), root mean squared error ($RMSE$) and mean absolute error ($MAE$) have been used as performance metrics. Boxplots of the results have also been constructed. The rest of the paper is organized as follows. A summary of the related work is given in Section II. The proposed method, its mathematical description and its algorithm are given in Section III-B. Experiments are given in Section IV. This section consits of a brief description of the datasets used, the addition of non-informative features, experimental set-up and a detailed description of the results obtained. The paper ends with a conclusion given in Section V.

## II. RELATED WORK

Many authors have developed methods for improving the performance of the standard $k$-NN method. As the standard $k$-NN method assigns equal weights to observations in the neighbourhood of a test observation, Bailey [13] proposed weighted $k$ nearest neighbour (W$k$-NN) to overcome this problem by assigning weights to the neighbours according to the distance calculated. This method, however, uses all the training observations in the training data thus making the method global. To reduce size of the data and boost up query time of the $k$-NN method, Gowda and Krishna [14], Angiulli [15] and Alpaydin [16] suggested the idea of condensed nearest neighbour (C-NN) by removing observations showing similarity and not adding extra information. However, C-NN depends on the order of data and is likely to ignore points on the boundary. Gates [17] proposed a similar method, the reduced nearest neighbour (R-NN), by removing patterns which do not affect the results in the training data. This method reduces training data size and removes templates. However, this method, like C-NN, is computationally complex. Guo *et al.* [18] proposed model based $k$-NN method that estimates test data using a model based on the training data. This method also reduces training data size and is shown to improve prediction accuracy. This method fails to consider marginal data outside the region. Clustered $k$ nearest neighbor [19] selects the nearest neighbours from the clusters to overcome defects of uneven distributions of training samples and is consequently robust in nature. Selection of the threshold used for distances between observations within clusters in this method is difficult. Criteria for finding the optimum $k$ value for each category is also unknown. Modified $k$ nearest neighbour (M$k$-NN) [20], uses weights and validity of data point to classify nearest neighbour. $k$-d tree nearest neighbour ($k$d-NN) [21], divides the training data exactly into half plane and it is used for the organisation of multi-dimensional points. It is simple, fast and produces perfectly balanced tree. However, it is computationally complex, requires intensive search and blindly slice points into half which may miss data structure.

In addition to the above work, there are several ensemble methods using $k$-NN or its modification as the base learner. One of the fundamental ensemble methods is bagging (bootstrap aggregation) [22] which construct a new model to approximate the exact bootstrap expectation of the model [23]–[25]. This method has become the fundamental step in various state-of-the-art ensemble learning methods. In bagging, a large number $B$ of bootstrap samples are randomly drawn from the learning data and the base learning model is applied on each of the sample. Response for a new observation is predicted by averaging (for regression problems) or majority voting (for classification problems) based on predictions made by all the $B$ base models [22]. The method proposed in this paper also uses bagging as one of the steps for improved prediction performance. Steele [10] extended the idea of exact bagging to bootstrap sub-sampling schemes (with and without replacement). Various ensemble methods exist that combine bagging with feature subspace for building base $k$-NN models [11], [12], [53]. This is done by randomly selecting a subset of features for each bootstrap sample to build the base model [11], [12]. Random feature subset selection is also implemented in the proposed method as one of the steps of the method. The proposed method uses the subspace idea in a novel way in that a further search is made in the subset of features selected initially for best fitting observations in nearest neighbourhood of a test point in each base $k$-NN model. This inculcates additional randomness in the models and thus improves the overall performance of the ensemble method. Other studies have considered optimizing the number of nearest neighbours, $k$, in the base $k$-NN models for ensemble learning [26], [31]. Garcia-Pedrajas and Ortiz-Boyer [9] proposed boosting $k$-NN by using two methodologies. The first method involves selecting a subset of the features and the second transform the inputs by using non-linear projection of the features. Several other studies consider boosting methods for $k$-NN model to gain improved performance [27], [28].

Furthermore, there has been several methods proposed in literature based on the idea of locally linear models. The ensemble of locally linear models predict each data point in the test dataset based on a model build on its nearest neighbours in the training dataset. This idea was initially presented for classification problems by Bottou and Vapnik [46] and later used for regression problems choosing models based on leave-one-out cross validation [47]. A locally linear regression fits a linear model for each observation in the data in the form of a linear combination of its nearest neighbours [47]. The current paper has used locally linear model fitting for each $k$-NN model in conjunction with bootstrapping and random feature selection for each bootstrap to produce diversity and accuracy in the base models. Locally linear model fitting is done for each observation in the test data using both forward and backward model selection methods based on selected subset of features.

Kainulainen *et al.* [29] proposed a $k$-NN ensemble using feature subsets selected by forward model selection method on random samples of the data for building each base $k$-NN model. They also suggested using values of $k$, the number of nearest neighbours, optimized for each $k$-NN base model. The current paper has considered feature selection for each base $k$-NN model based on a random subset of features and a bootstrap sample taken from the training data. Model selection is done by using the nearest observations identified by each base $k$-NN model unlike the method in [29]. The proposed method is also inspired from the method of Kang and Kang [30] where ensemble of multiple linear models is used. The proposed method improves prediction performance in three ways.

1) Each base model is based on random bootstrap sample of the training data with random subset of features making the base models diverse.
2) Model selection in each base model is done based on the observations in the nearest neighbourhood composed of $k$ samples.
3) The effect of non-informative features is eliminated by forward/backward stepwise regression. The fitted models are also allowed to predict the test observations thus avoiding the adverse effects of simple mean estimation.

## III. METHOD

Suppose a training data $\mathcal{L} = (X, Y)$, where $X$ is $p$ features matrix with $n$ rows (observations) and $Y$ is the response variable. Let $x'$ be a $p$-dimensional query/test observation, and it is required to predict the target value i.e. $\hat{y}$ for $x'$. Let $B$ be the number of bootstrap samples drawn from training dataset $\mathcal{L} = (X, Y)$. The samples are drawn such that a random subset of features of size $d < p$ is considered with each sample. The $k$ nearest nieghbours for $x'$ are identified in each of the bootstrap samples using some distance metric, say, Euclidean distance. Each bootstrap sample thus reduces to a data matrix $M_{k\times(d+1)}$. Let $H_j(.)$ be a linear regression model consisting of $j = 0, 1, 2, \ldots, d$ variables and $C_j(.)$ be the corresponding value of the objective function used for model assessment. The proposed method fits an initial (empty) regression model $H_0(.)$. $H_0(.)$ is then updated as $H_1(.)$ by adding one from the $d$ variables that optimizes the objective function value $C_1(.)$. Similarly $H_1(.)$ is updated as $H_2(.)$ by adding another from the remaining $d-1$ variables optimizing the objective function value $C_2(.)$. Stop updating $H_j(.)$ to $H_{j+1}(.)$ when $C_{j+1}(.) - C_j(.) < \tau$, where $\tau$ a threshold set for the gain in predictive capability of model $H_{j+1}(.)$ as compared to model $H_j(.)$. Call the final model from the data matrix $M_{k\times(d+1)}$ as $\hat{H}_j(.)$. Let $\hat{y}_t$, $(t = 1, 2, \ldots, B)$, be the value predicted for $x'$ by the stepwise regression model. In this way, estimates $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_B$ of the same test point $(x')$ are obtained, which are $B$ predicted values. To get the overall estimated of the test point, the arithmetic mean of all these predicted values is calculated.

In this method one can use; forward, backward and both stepwise model selection procedures. In case of forward selection, there is no issue with the value of $k$, that is $k$ may be less, equal or greater than the number of features. In case of backward selection, the value of $k$ should be taken equal to or greater than the number of features, in that backward selection starts from the full model. In cases where the number of features is more than the number of observations, a random sample (with replacement) of the required size need to be selected to use backward model selection.

### A. MATHEMATICAL MODEL

Let $b$ is a ''campaigner'' value for the true model parameter vector $\beta$ and the amount $y_i - (x_i)^T b$, known as the residual term for the $i$th sample, which measures the perpendicular distance between the hyperplane $y = x^T b$ and the point $(x_i, y_i)$. Thus, it measures the level of fit between the true data points and the model values. The ''sum of squared residuals'' is a quantity which measures the whole model fit, i.e.,

$$S_t(b) = \sum_{i=1}^{k}(y_i - x_i^T b)^2 = (y - Xb)^T(y - Xb),$$

where, $t = 1, 2, \ldots, B$ and $i = 1, 2, \ldots, k$, $T$ is used for matrix transpose and $k$ is the number of samples in the neighbourhood of the test point. $S_t(b)$ is a quadratic function of $b$ and hence it possesses a single minimum value at $b = \hat{\beta}$, which can be obtained through the explicit expression:

$$\hat{\beta}_t = argmin_{b\epsilon\Re} S_t(b) = (X^T X)^{-1} X^T y.$$

By this way, $B$ stepwise models are obtained for a test point $x'$. Each of the models is built on a subset of features taken randomly from the given feature space. Estimate the response of $x'$ by all these functions to obtain $B$ values i.e.

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_B$ and pool them as

$$\hat{y} = \frac{1}{B}\sum_{t=1}^{B}\hat{y}_t,$$

to get the final overall estimate of the response.

### B. ALGORITHM

The proposed O-$k$-NN-E takes the following steps.

1) Take $B$ bootstrap samples from training data along with randomly selecting a subset of $d < p$ features.
2) Apply $k$-NN on each sample to identify the $k$ nearest neighbours for a test observation $x'$.
3) Form a data matrix $M_{k\times(d+1)}$ consisting of the $k$ observations identified by $k$-NN and $d$ features from each bootstrap sample.
4) Fit stepwise linear regression models on the $B$ matrices and use them to predict $x'$.
5) Final prediction for $x'$ is the average of all the $B$ predictions by the regression models.

Pseudo code of the proposed method O$k$-NN-E is given in Algorithm 1 along with an illustrating flow chart in Figure 1.

**Algorithm 1** Pseudo Code for O$k$-NN-E

```
/* Construct B $k$-NN models each on a bootstrap sample and $d < p$ features */
$p \leftarrow$ number of features in the dataset;
$d \leftarrow$ number of features for each $k$-NN;
$B \leftarrow$ number of $k$-NN learners;
$k \leftarrow$ number of nearest nieghbours in each $k$-NN;

for $t = 1 \rightarrow B$ do
    ★ Built base $k$-NN;
    ★ Identify $k$ nearest samples for test point $x'$ in base $k$-NN model using Euclidean distance;
    ★ Fit stepwise regression model $H_j(.)$ using $k$ points in $k$-NN model to estimate $x'$;
    ★ Predict $x'$ through $H_j(.)$ model;
    ★ Save all the results obtained by $H_j(.)$;
end for

Pool all $B$ estimates to get final result i.e., $\hat{y} = \sum_{t=1}^{B}\hat{y}_t$
```

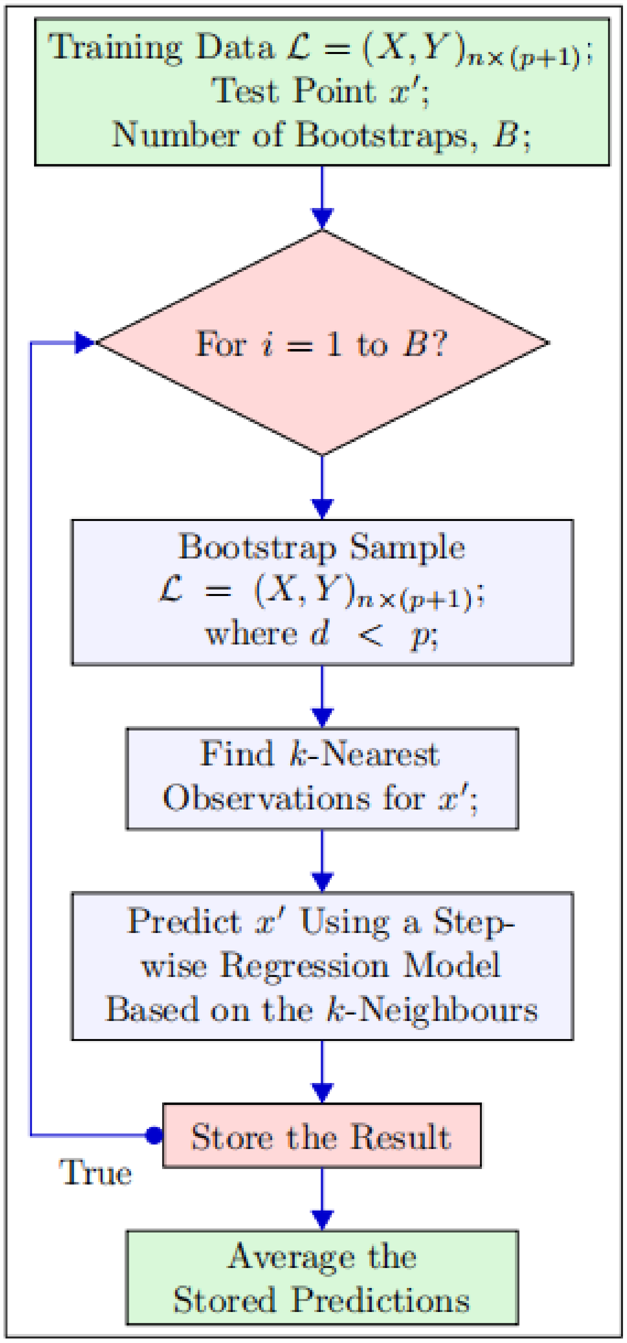
Training Data $\mathcal{L} = (X, Y)_{n\times(p+1)}$;
Test Point $x'$;
Number of Bootstraps, $B$;
For $i = 1$ to $B$?
Bootstrap Sample
$\mathcal{L} = (X, Y)_{n\times(p+1)}$;
where $d < p$;
Find $k$-Nearest Observations for $x'$;
Predict $x'$ Using a Stepwise Regression Model Based on the $k$-Neighbours
Store the Result
True
Average the Stored Predictions

**FIGURE 1.** Flow chart of the proposed "O$k$-NN-E" method.

## IV. EXPERIMENTS AND RESULTS

### A. BENCHMARK DATASETS

#### 1) BENCHMARK DATASETS WITH ORIGINAL FEATURES

To compare the proposed method with the other state-of-the-art methods, a total of 16 datasets are used. These datasets are taken from various open sources. A brief summary of the datasets is given in Table 1. The table shows the number of observations, number of variables and the sources against each dataset. Datasets are arranged in ascending order with respect to the number of features.

#### 2) BENCHMARK DATASETS WITH CONTRIVED FEATURES

To further assess the method, random non-informative features are added to the datasets. Number of non-informative features added to each dataset is the same as the original number of features in the dataset. All the features are generated from uniform distribution over the interval [0, 1].

### B. EXPERIMENTAL SETUP

Experiments carried out on the given datasets are designed as follows. Each dataset is divided randomly into two

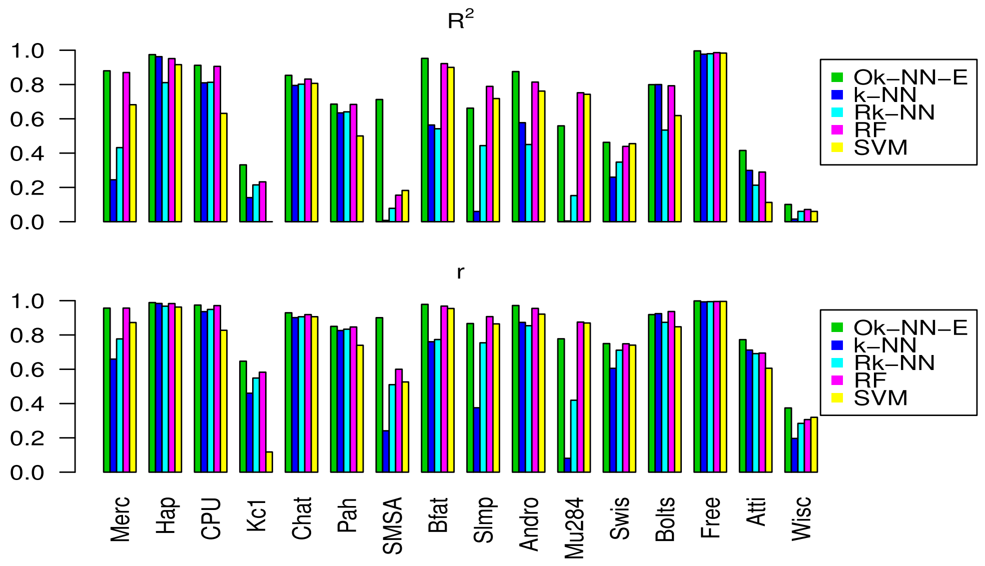

**FIGURE 2.** Bar plots for $R^2$ and $r$ for datasets with original features.

**TABLE 1.** Benchmark datsets. Table shows the number of observations $n$, number of features $p$ and source against each dataset.

| Dataset | n | p | Sources |
|---|---|---|---|
| Free | 39 | 4 | [32], [33] |
| Swiss | 47 | 5 | [33], [34] |
| Atti | 30 | 6 | [35] |
| CPU | 209 | 7 | [36] |
| Bolts | 40 | 7 | https://www.openml.org/d/193 |
| Kc1 | 145 | 8 | http://openml.org/ |
| slump | 103 | 9 | [36], [37] , |
| Mu284 | 284 | 9 | http://openml.org/ |
| Merc | 53 | 9 | https://www.openml.org/d/1090 |
| Hap | 158 | 10 | https://www.openml.org/d/40916 |
| Chat | 235 | 12 | https://www.openml.org/d/695 |
| SMSA | 59 | 14 | https://www.openml.org/d/1091 |
| Bfat | 252 | 14 | https://www.openml.org/d/560, [37], [38] |
| Wisc | 194 | 32 | https://www.openml.org/d/191 |
| Andro | 49 | 35 | [39] |
| Pah | 80 | 112 | https://www.openml.org/d/424 |

non-overlapping parts; a training part with 70% observations and a testing part with the remaining 30% of observations. For the proposed method, the values of hyper-parameters are fixed for simplicity purpose as given below. A total of five hundred $k$-NN models are constructed each on a bootstrap sample from the training part along with randomly selecting $d = \sqrt{(p)}$ number of features. The value of $k$ is fixed for all the given datasets at $k = 0.1 \times n$, forward stepwise model selection is executed using $AIC$ as the model selection criterion in each $k$-NN model using the $k$ nearest observations and the random set of $d$ features. Final results are the average of the results obtained from all the 500 runs. The hyper-parameters might effect the performance of the proposed method and could be fine-tuned using cross-validation, for example. A discussion on the effect of hyper-parameters on the prediction performance of the proposed method is given in Section IV-D.

Random forest is used as implemented in the R package `randomForest` [42]. For random $k$-NN, the R package `rknn` [45] is used, whereas for SVM, R package `kernlab` [41] is used. For $k$-NN, R package `knn` [44] is used. For random forest, `nodesize`, `ntree` and `mtry` are fine-tuned using `tune.randomForest` function available within the R-Package `e1071` [40]. For selecting the node size value a grid search is made in the values (1, 5, 10, 15, 20, 25, 30), for tuning `ntree` we tried values (500, 1000, 1500, 2000) and for tuning `mtry`, a search in values $(\sqrt{p}, p/5, p/4, p/3, p/2)$ is made. All the possible values of `mtry` are checked in cases where $p < 12$. $k$-NN is fine-tuned by using the R function `tune.knn` within the R library "e1071" for different values of the hyper-parameter $k$ i.e. $k = 1, \ldots, 10$. Similarly, random $k$-NN is fine-tuned by searching values of $k = 1, \ldots, 10$ and the number of features in $\sqrt{p}, p/5, p/4, p/3, p/2$. The rest of the settings are kept intact as given in the R$k$-NN [45] R package. All the above hyper-parameters values are fine-tuned using 10-fold cross validation on the training data by searching the best values from the given search spaces. The same is implemented in the R packages used, i.e. `tune.randomForest` and `tune.knn`, for tuning the hyper-parameters. SVM is used with linear kernel as implemented in the R package `kernlab` [41] with default setting.

### C. RESULTS

The results given in Table 2 reveal that the proposed O$k$-NN-E has outperformed all the other methods on almost all the datasets. For quick insights, the results are also given in the form of bar plots (see Figures 2, 3). O$k$-NN-E is giving higher $R^2$ values than the other methods on 13 datasets, random forest is giving the highest $R^2$ values on 2 datasets. $k$-NN gave better results than the other on 1 dataset in terms of $R^2$.

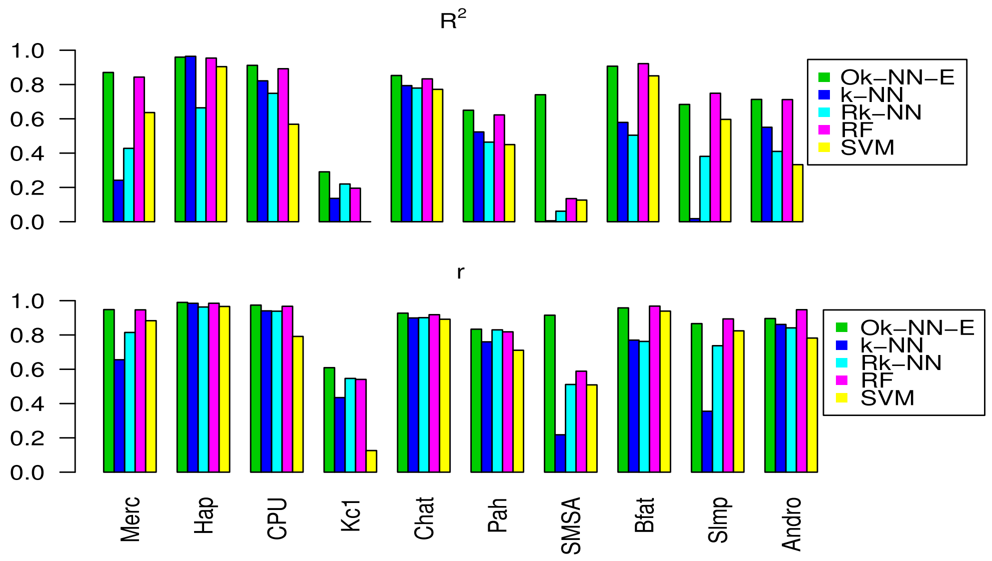

**FIGURE 3.** Bar plots for $R^2$ and $r$ for datasets with adding non-informative features.

**TABLE 2.** Proportion of explained variations $R^2$, Pearson's product moment correlation coefficient $r$, predicted mean square error *PMSE*, root mean square error *RMSE* and mean absolute error *MAE* for the methods against each dataset with original features.

| Matrics | Methods | Datasets | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Merc | Hap | CPU | Kc1 | Chat | Pah | SMSA | Bfat | Slmp | Andro | Mu284 | Swiss | Bolts | Free | Atti | Wiscn |
| | O$k$-NN-E | **0.880** | **0.974** | **0.912** | **0.332** | **0.854** | **0.686** | **0.713** | **0.953** | 0.662 | **0.875** | 0.559 | **0.463** | 0.799 | **0.996** | **0.415** | **0.101** |
| | $k$-NN | 0.244 | 0.963 | 0.810 | 0.140 | 0.795 | 0.635 | 0.008 | 0.564 | 0.060 | 0.577 | 0.004 | 0.260 | **0.800** | 0.977 | 0.299 | 0.015 |
| $R^2$ | R$k-NN$ | 0.432 | 0.811 | 0.813 | 0.215 | 0.802 | 0.641 | 0.078 | 0.542 | 0.444 | 0.450 | 0.152 | 0.348 | 0.534 | 0.980 | 0.213 | 0.060 |
| | RF | 0.870 | 0.951 | 0.906 | 0.232 | 0.832 | 0.684 | 0.155 | 0.922 | **0.789** | 0.814 | **0.752** | 0.439 | 0.793 | 0.986 | 0.290 | 0.071 |
| | SVM | 0.682 | 0.916 | 0.632 | 0.105 | 0.807 | 0.500 | 0.182 | 0.900 | 0.718 | 0.762 | 0.743 | 0.455 | 0.619 | 0.983 | 0.113 | 0.060 |
| | O$k$-NN-E | **0.957** | **0.989** | **0.974** | **0.647** | **0.929** | **0.850** | **0.901** | **0.978** | 0.867 | **0.972** | 0.778 | **0.750** | 0.919 | **0.999** | **0.773** | **0.374** |
| | $k$-NN | 0.659 | 0.984 | 0.936 | 0.460 | 0.901 | 0.826 | 0.241 | 0.760 | 0.375 | 0.873 | 0.081 | 0.606 | 0.924 | 0.993 | 0.712 | 0.197 |
| $r$ | R$k$-NN | 0.777 | 0.968 | 0.949 | 0.549 | 0.906 | 0.834 | 0.510 | 0.773 | 0.754 | 0.854 | 0.419 | 0.711 | 0.874 | 0.994 | 0.691 | 0.284 |
| | RF | **0.957** | 0.983 | 0.971 | 0.583 | 0.919 | 0.846 | 0.600 | 0.968 | **0.907** | 0.955 | **0.875** | 0.749 | **0.937** | 0.995 | 0.694 | 0.307 |
| | SVM | 0.873 | 0.963 | 0.827 | 0.118 | 0.906 | 0.740 | 0.526 | 0.954 | 0.865 | 0.922 | 0.870 | 0.741 | 0.848 | 0.996 | 0.606 | 0.320 |
| | O$k$-NN-E | **0.020** | **0.128** | **18081.150** | **28.219** | **1587.976** | **0.002** | **607.442** | **16.599** | 78.106 | **0.109** | 812.982 | **160.439** | **210.613** | **0.001** | **82.162** | **1855.585** |
| | $k$-NN | 0.113 | 0.191 | 42970.100 | 43.902 | 2252.036 | 0.003 | 2813.130 | 153.552 | 216.501 | 0.408 | 2053.541 | 223.508 | 214.822 | 0.005 | 106.088 | 2257.439 |
| $PMSE$ | R$k$-NN | 0.094 | 0.943 | 45712.290 | 41.520 | 2184.488 | 0.003 | 2573.213 | 162.020 | 129.741 | 0.526 | 1517.056 | 202.810 | 497.530 | 0.005 | 120.174 | 1954.245 |
| | RF | 0.025 | 0.245 | 23285.400 | 39.875 | 1836.718 | **0.002** | 2195.482 | 27.909 | **47.956** | 0.184 | **441.871** | 170.520 | 219.682 | 0.003 | 106.647 | 1952.488 |
| | SVM | 0.057 | 0.425 | 86687.04 | 60.419 | 2163.604 | 0.004 | 2577.096 | 35.671 | 64.423 | 0.232 | 455.106 | 173.776 | 406.732 | 0.004 | 132.205 | 2025.512 |
| | O$k$-NN-E | **0.141** | **0.358** | **134.466** | **5.312** | **39.849** | **0.045** | **24.646** | **4.074** | 8.838 | **0.330** | 28.513 | **12.666** | **14.513** | **0.032** | **9.064** | **43.077** |
| | $k$-NN | 0.336 | 0.437 | 207.292 | 6.626 | 47.456 | 0.055 | 53.039 | 12.392 | 14.714 | 0.639 | 45.316 | 14.950 | 14.657 | 0.071 | 10.300 | 47.513 |
| $RMSE$ | R$k$-NN | 0.307 | 0.971 | 213.804 | 6.444 | 46.739 | 0.055 | 50.727 | 12.729 | 11.390 | 0.725 | 38.949 | 14.241 | 22.305 | 0.071 | 10.962 | 44.207 |
| | RF | 0.158 | 0.495 | 152.596 | 6.315 | 42.857 | **0.045** | 46.856 | 5.283 | **6.925** | 0.429 | **21.021** | 13.058 | 14.822 | 0.055 | 10.327 | 44.187 |
| | SVM | 0.239 | 0.652 | 294.427 | 7.773 | 46.515 | 0.063 | 50.765 | 5.973 | 8.026 | 0.482 | 21.333 | 13.182 | 20.168 | 0.063 | 11.498 | 45.006 |
| | O$k$-NN-E | **0.114** | **0.552** | 140.259 | 1.939 | **68.613** | 0.020 | **14.536** | **6.556** | 7.780 | **0.133** | 64.608 | **15.273** | 12.808 | **0.036** | **8.670** | **48.521** |
| | $k$-NN | 0.329 | 1.004 | 167.807 | 2.005 | 77.740 | 0.019 | 26.106 | 22.640 | 11.681 | 0.305 | 111.718 | 17.330 | **11.098** | 0.081 | 9.271 | 51.963 |
| $MAE$ | R$k$-NN | 0.303 | 2.381 | 172.946 | 1.929 | 76.042 | 0.020 | 23.113 | 23.818 | 14.624 | 0.341 | 96.617 | 17.071 | 22.202 | 0.073 | 9.505 | 50.065 |
| | RF | 0.129 | 1.603 | **120.076** | **1.894** | 70.436 | **0.018** | 20.109 | 8.192 | 10.346 | 0.190 | 41.979 | 15.901 | 13.401 | 0.066 | 9.111 | 49.538 |
| | SVM | 0.203 | 0.969 | 197.652 | 2.063 | 71.187 | 0.021 | 19.548 | 7.747 | **7.069** | 0.224 | **36.493** | 15.356 | 17.992 | 0.066 | 10.205 | 48.893 |

Random $k$-NN and SVM did not outperform the other methods on any of the datasets considered. In terms of Pearson's product moment correlation coefficients ($r$), the proposed methods is better than the others on 12 datasets and similar to random forest on 1 dataset. Random forest is better than the others on 3 datasets. In terms of predicted mean square

**TABLE 3.** **Proportion of explained variations $R^2$, Pearson's product moment correlation coefficient $r$, predicted mean square error *PMSE*, root mean square error *RMSE* and mean absolute error *MAE* for the methods against each dataset with added non-informative features.**

| Matrics | Methods | Merc | Hap | CPU | Kc1 | Chat | Pah | SMSA | Bfat | Slmp | Andro |
|---|---|---|---|---|---|---|---|---|---|---|---|
|  | O$k$-NN-E | **0.870** | 0.960 | **0.912** | **0.256** | **0.853** | **0.650** | **0.723** | 0.907 | 0.684 | **0.713** |
|  | $k$-NN | 0.242 | **0.965** | 0.822 | 0.106 | 0.794 | 0.523 | 0.105 | 0.579 | 0.018 | 0.551 |
| $R^2$ | R$k$-NN | 0.428 | 0.664 | 0.749 | 0.203 | 0.779 | 0.464 | 0.210 | 0.504 | 0.381 | 0.410 |
|  | RF | 0.843 | 0.954 | 0.892 | 0.165 | 0.833 | 0.623 | 0.135 | **0.922** | **0.749** | 0.712 |
|  | SVM | 0.637 | 0.904 | 0.568 | 0.100 | 0.772 | 0.450 | 0.126 | 0.851 | 0.597 | 0.333 |
|  | O$k$-NN-E | **0.948** | **0.990** | **0.974** | **0.609** | **0.927** | **0.834** | **0.915** | 0.958 | 0.866 | 0.896 |
|  | $k$-NN | 0.655 | 0.985 | 0.940 | 0.435 | 0.899 | 0.760 | 0.210 | 0.770 | 0.355 | 0.861 |
| $r$ | R$k$-NN | 0.815 | 0.963 | 0.939 | 0.547 | 0.901 | 0.830 | 0.511 | 0.762 | 0.737 | 0.841 |
|  | RF | 0.946 | 0.985 | 0.967 | 0.541 | 0.918 | 0.818 | 0.589 | **0.968** | **0.894** | **0.947** |
|  | SVM | 0.883 | 0.966 | 0.791 | 0.126 | 0.892 | 0.711 | 0.509 | 0.939 | 0.824 | 0.782 |
|  | O$k$-NN-E | **0.011** | 0.105 | **9482.000** | **15.773** | **829.362** | **0.001** | **204.856** | 17.178 | 37.672 | **0.134** |
|  | $k$-NN | 0.060 | **0.094** | 20227.370 | 21.901 | 1173.130 | 0.002 | 1576.130 | 77.752 | 115.621 | 0.210 |
| $PMSE$ | R$k$-NN | 0.051 | 0.873 | 29423.170 | 20.507 | 1265.580 | 0.002 | 1459.790 | 92.184 | 74.024 | 0.280 |
|  | RF | 0.015 | 0.121 | 12853.410 | 20.505 | 950.274 | **0.001** | 1259.970 | **14.775** | **29.377** | 0.140 |
|  | SVM | 0.034 | 0.253 | 49836.960 | 29.919 | 1314.340 | 0.002 | 1483.200 | 28.056 | 48.599 | 0.316 |
|  | O$k$-NN-E | **0.105** | 0.324 | **97.376** | **3.972** | **28.799** | **0.032** | **14.313** | 4.145 | 6.138 | **0.366** |
|  | $k$-NN | 0.245 | **0.307** | 142.223 | 4.680 | 34.251 | 0.045 | 39.700 | 8.818 | 10.753 | 0.458 |
| $RMSE$ | R$k$-NN | 0.226 | 0.934 | 171.532 | 4.528 | 35.575 | 0.045 | 38.207 | 9.601 | 8.604 | 0.529 |
|  | RF | 0.122 | 0.348 | 113.373 | 4.528 | 30.827 | **0.032** | 35.496 | **3.844** | **5.420** | 0.374 |
|  | SVM | 0.184 | 0.503 | 223.242 | 5.470 | 36.254 | 0.045 | 38.512 | 5.297 | 6.971 | 0.562 |
|  | O$k$-NN-E | **0.064** | **0.613** | 88.074 | 1.131 | **36.313** | **0.009** | **5.563** | 7.841 | 6.674 | 0.141 |
|  | $k$-NN | 0.172 | 0.966 | 93.049 | 1.046 | 40.723 | 0.012 | 12.186 | 11.498 | **5.943** | 0.156 |
| $MAE$ | R$k$-NN | 0.166 | 1.537 | 132.075 | **1.022** | 42.924 | 0.012 | 10.598 | 12.584 | 8.259 | 0.182 |
|  | RF | 0.075 | 0.928 | **75.793** | 1.118 | 37.294 | 0.010 | 10.405 | **3.882** | 6.264 | **0.123** |
|  | SVM | 0.122 | 0.683 | 153.577 | 1.055 | 43.993 | 0.011 | 9.731 | 6.029 | 6.478 | 0.193 |

error ($PMSE$), O$k$-NN-E has outperformed the other methods on 13 datasets and similar to random forest on 1 dataset. Random forest gives the best results on 2 datasets. Random $k$-NN, $k$-NN and SVM could not perform well on any of the datasets compared to the rest of the methods. Similarly, O$k$-NN-E outperformed all the other methods based on root mean square error ($RMSE$) and mean absolute error ($MAE$).

For the case of added non-informative features, the results are given in Table 3 for randomly selected 10 datasets. It is evident from the results that the proposed O$k$-NN-E is robust to the issue of non-informative features in the data. The proposed O$k$-NN-E is giving promising results on 5 datasets in terms of each of the performance measures considered. Random forest has outperformed the other methods on 2 datasets based on each of the performance measures. Random forest and O$k$-NN-E gave similar results on 1 dataset based on $RMSE$.

For further assessment, boxplots of $R^2$, $r$, predicted mean square error $RMSE$, root mean square error $RMSE$ and mean absolute error $MAE$ values obtained from all the 500 runs of the experiments are also constructed. The boxplots are given

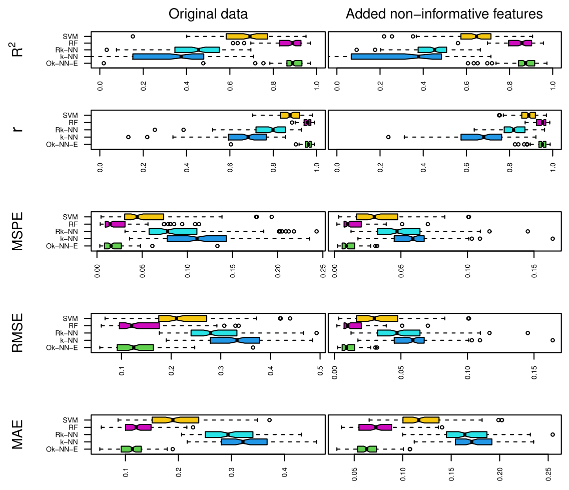

**FIGURE 4.** **Boxplots for Merc.**

in Figures 4-13. The left panel in the figures are boxplots for datasets with their original features whereas the right panel are the ones for datasets with added non-informative features.

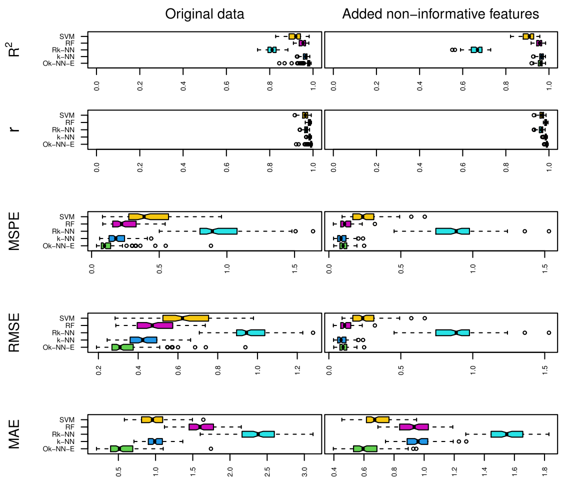

**FIGURE 5.** Boxplots for Hap.

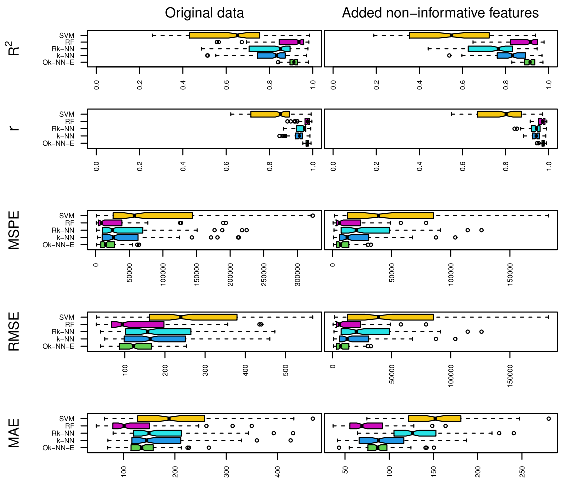

**FIGURE 6.** Boxplots for CPU.

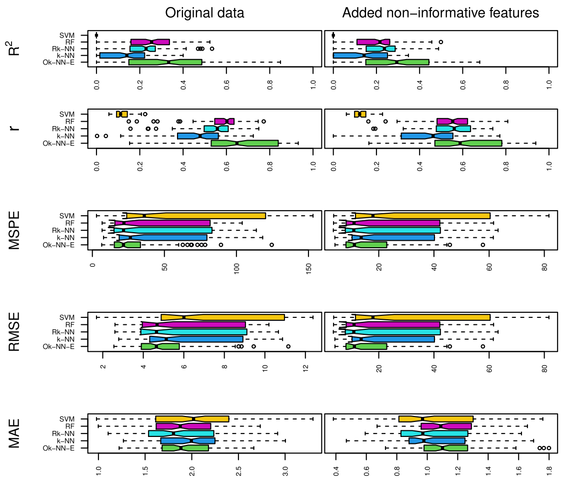

**FIGURE 7.** Boxplots for Kc1.

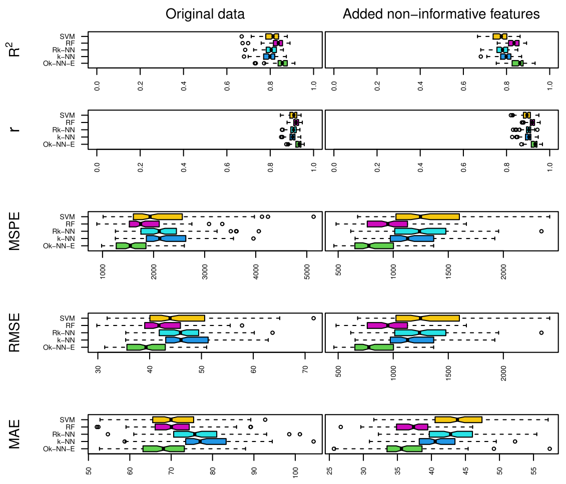

**FIGURE 8.** Boxplots for Chat.

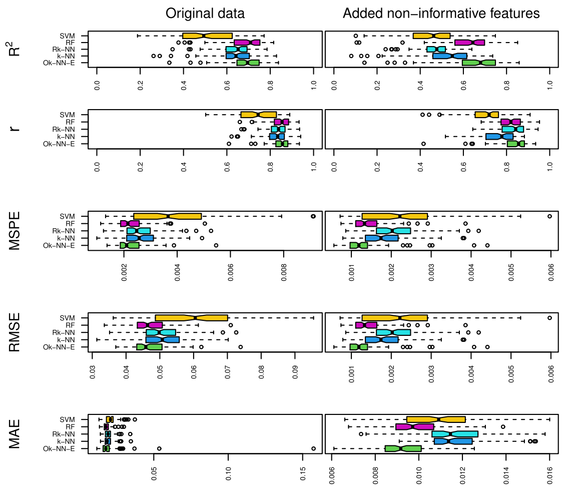

**FIGURE 9.** Boxplots for Pah.

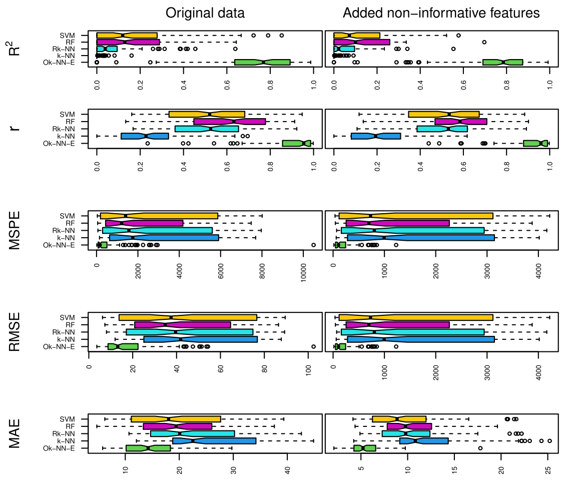

**FIGURE 10.** Boxplots for SMSA.

The figures reveal that the proposed O*k*-NN-E is less affected by non-informative features in the data as compared to the other methods considered.

### *D. HYPER-PARAMETERS ASSESSMENT*

This section presents the effect of hyper-parameters involved in O*k*-NN-E on its performance. There are 3 hyper-parameters that could effect the performance of the

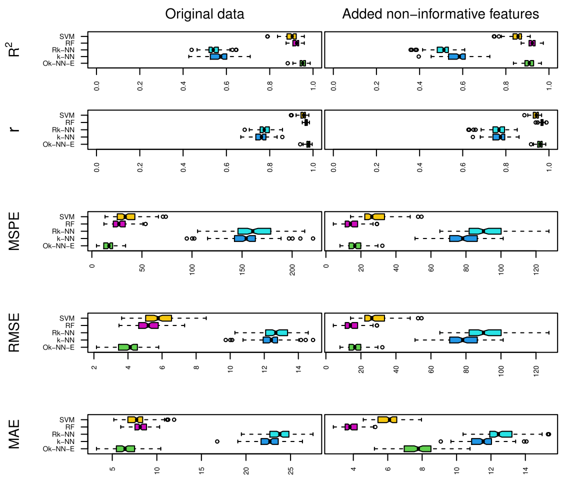

**FIGURE 11.** Boxplots for Bfat.

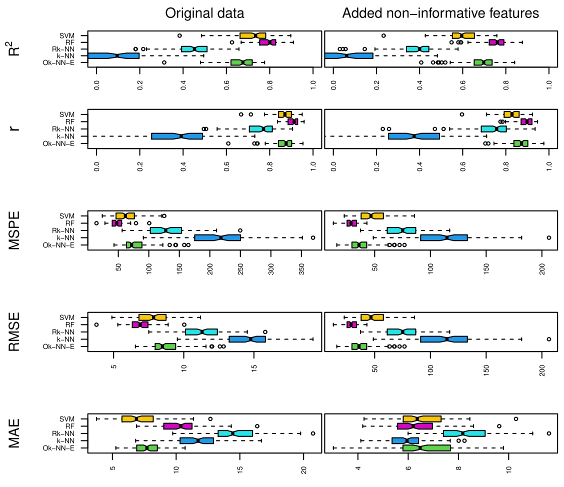

**FIGURE 12.** Boxplots for Slmp.

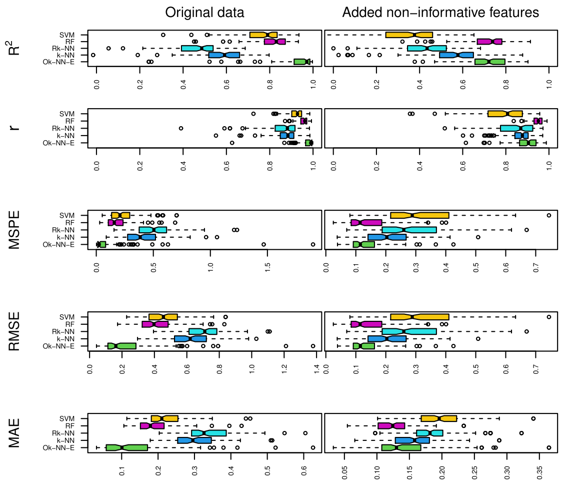

**FIGURE 13.** Boxplots for Andro.

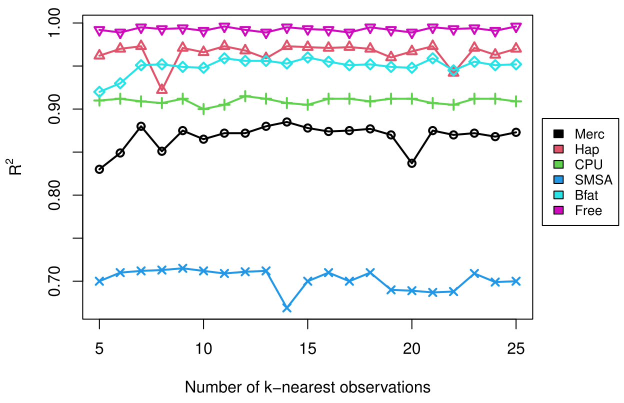

**FIGURE 14.** Plot of $R^2$ for different number of nearest neighbours ($k$).

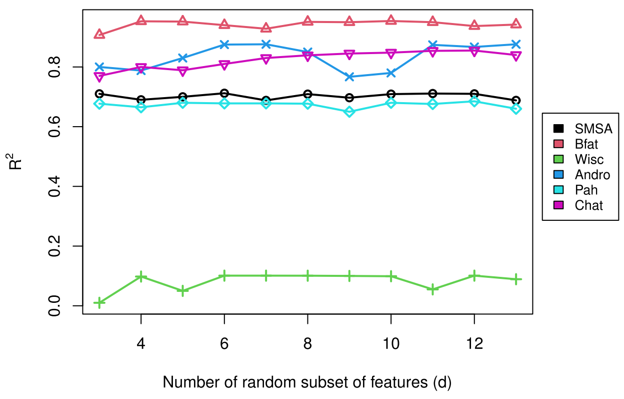

**FIGURE 15.** Plot of $R^2$ for different feature subset size ($d$).

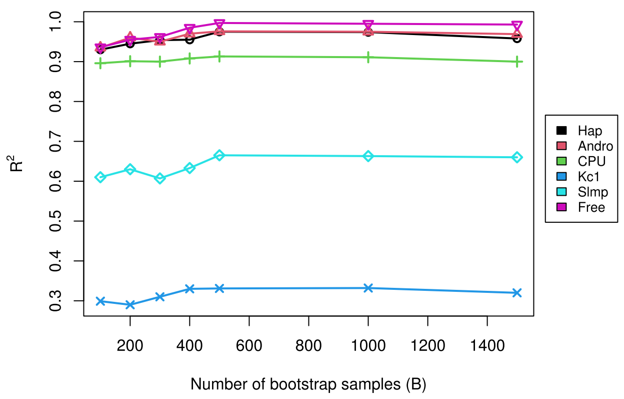

**FIGURE 16.** Plot of $R^2$ for different number of bootstrap samples ($B$).

proposed method. These are, the number of nearest neighbours ($k$), the number of features ($d$) selected randomly for each base $k$-NN model and the number of bootstrap samples ($B$) taken from the training data. For assessing each parameter, 6 datasets are randomly selected from the given benchmark problems. Effect of these parameters on O$k$-NN-E is shown in Figures 14-16. It is evident from Figure 14 that the number of nearest neighbours does effect the performance of the method and should therefore be fine-tuned. Similarly, the number of features taken randomly for bootstrapping also effect the performance of the method and needs to be fine-tuned. For the case of bootstrap samples ($B$), the method becomes stable after increasing this number from $B = 500$. Thus it suggests that $B = 500$ is a decent number as increasing this involves additional computational cost. In the current paper, hyper-parameter values are fixed for simplicity purposes as mentioned in Section IV-B. However, one could use cross-validation to select appropriate values for these parameters.

## V. CONCLUSION

This paper presented the idea of model selection based on a random feature subset in the $k$-nearest neighbours of each test data observation. The process of model selection is stretched to a large pool of $k$ nearest neighbours models by bootstrapping method to construct an ensemble model. The proposed method is compared with $k$-NN, random $k$-NN, random forest and support vector machine on 16 datasets using $R^2$, Pearson's product moment correlation coefficient $r$, predicted mean square error $RMSE$, root mean square error $RMSE$ and mean absolute error $MAE$ as performance metrics. The results given in the paper show that the proposed method has outperformed the rest of the methods on almost all the datasets considering the performance metrics used.

$k$-NN and random $k$-NN have consistently performed poor as compared to the proposed O$k$-NN-E method due to their sensitivity to non-informative features in the data. Non-informative features in the data adversely affect $k$-NN and $k$-NN based ensembles in that non-informative features still play their role in the estimation of the response based on the nearest neighbours. The proposed method fixes this problem by selecting the best features in a stepwise fashion to estimate the response variable value via linear regression model. This phenomenon has also been demonstrated by adding extra non-informative features to the existing datasets. By the addition of non-informative features, the proposed method has shown more robust as compared to the rest of the methods. As $k$-NN and R$k$-NN are based on the standard approach of averaging the responses of the $k$-nearest observations of a test observations, these methods have shown poor performance with added non-informative features in the data. This revealed that model selection in the nearest neighbourhood of test observations fixes performance issues of $k$-NN models in general and in the case of non-informative features in the data in particular.

Moreover, the proposed method constructs each $k$-NN model on a subset of features that ensures diversity in the ensemble in addition to the randomization brought by bootstrapping. Accuracy in the base $k$-NN models in ensured by proposed stepwise model selection based on the $k$-nearest observations and the random features set. Thus, the analyses given in the paper also show that by increased randomization and accuracy, a more powerful ensemble model could be constructed.

Model selection in each $k$-NN model in the ensemble could be time consuming given high dimensional problems and the proposed method may incur high computational cost as compared to ordinary $k$-NN methods. One possible way is to use parallel computing as implanted in the R package `parallel` [43] by parallelizing Step 3 of the algorithm i.e. fitting $B$ stepwise models. The proposed method in the paper does not take into account scale of the variables in the datasets and use them in their original format. Although the results of the method are promising as compared to the rest of the methods, it could further be improved by considering the appropriate distance metric in nearest neighbours identification for each data accordingly. Another possibility for further improving the proposed method is to use it in conjunction with feature selection methods as given in [48]–[52]. This could be done by selecting features from the entire set of features using the feature selection methods and then taking random set of features from the selected features for each base $k$-NN model in the ensemble.

This work can also be extended to the situations where the response variable is categorical i.e. classification problems. To achieve this, one could use the logit link function and allowing for stepwise model selection based on the nearest neighbours in each $k$-NN base model in the ensemble.

The proposed method is implemented in an R package `OkNNE`.

## REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-134, no. 1, pp. 21–27, Jan. 1967.

[2] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[4] N. Bhatia and Vandana, "Survey of nearest neighbor techniques," 2010, *arXiv:1007.0085*. [Online]. Available: http://arxiv.org/abs/1007.0085

[5] S. G. Kulkarni and M. V. Babu, "Introspection of various K-nearest neighbor techniques," *UACEE Int. J. Adv. Comput. Sci. Appl.*, vol. 3, no. 2, pp. 103–106, 2013.

[6] S. Bay, "Nearest neighbor classification from multiple feature subsets," *Intell. Data Anal.*, vol. 3, no. 3, pp. 191–209, Sep. 1999.

[7] S. I. Kaneko and S. Igarashi, "Combining multiple k-neighbor classifiers using feature combinations," *J. IECI*, vol. 2, no. 3, pp. 319–323, 2000.

[8] C. Domeniconi and B. Yan, "Nearest neighbor ensemble," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 1, Aug. 2004, pp. 228–231.

[9] N. García-Pedrajas and D. Ortiz-Boyer, "Boosting k-nearest neighbor classifier by means of input space projection," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10570–10582, Sep. 2009.

[10] B. M. Steele, "Exact bootstrap k-nearest neighbor learners," *Mach. Learn.*, vol. 74, no. 3, pp. 235–255, Mar. 2009.

[11] S. Li, E. J. Harner, and D. A. Adjeroh, "Random KNN," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Dec. 2014, pp. 629–636.

[12] A. Gul, A. Perperoglou, Z. Khan, O. Mahmoud, M. Miftahuddin, W. Adler, and B. Lausen, "Ensemble of a subset of kNN classifiers," *Adv. Data Anal. Classification*, vol. 12, no. 4, pp. 827–840, Dec. 2018.

[13] T. Bailey, "A note on distance-weighted k-nearest neighbor rules," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, no. 4, pp. 311–313, Apr. 1978.

[14] K. Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-25, no. 4, pp. 488–490, Jul. 1979.

[15] F. Angiulli, "Fast condensed nearest neighbor rule," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA: ACM, 2005, pp. 25–32.

[16] E. Alpaydin, "Voting over multiple condensed nearest neighbors," in *Lazy Learning*. Dordrecht, The Netherlands: Springer, 1997, pp. 115–132.

[17] G. Gates, "The reduced nearest neighbor rule (corresp.)," *IEEE Trans. Inf. Theory*, vol. 18, no. 3, pp. 431–433, May 1972.

[18] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *Proc. OTM Confederated Int. Conf. 'Move Meaningful Internet Syst.'* Berlin, Germany: Springer, Nov. 2003, pp. 986–996.

[19] Y. Zhou, Y. Li, and S. Xia, "An improved KNN text classification algorithm based on clustering," *J. Comput.*, vol. 4, no. 3, pp. 230–237, Mar. 2009.

[20] H. Parvin, H. Alizadeh, and B. Minaei-Bidgoli, "MKNN: Modified k-nearest neighbor," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 1. Hong Kong: Newswood Limited, Oct. 2008, pp. 1–4.

[21] R. F. Sproull, "Refinements to nearest-neighbor searching ink-dimensional trees," *Algorithmica*, vol. 6, nos. 1–6, pp. 579–589, Jun. 1991.

[22] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[23] B. Caprile, S. Merler, C. Furlanello, and G. Jurman, "Exact bagging with k-nearest neighbour classifiers," in *Proc. Int. Workshop Multiple Classifier Syst.* Berlin, Germany: Springer, Jun. 2004, pp. 72–81.

[24] Z.-H. Zhou and Y. Yu, "Adapt bagging to nearest neighbor classifiers," *J. Comput. Sci. Technol.*, vol. 20, no. 1, pp. 48–54, Jan. 2005.

[25] Z.-H. Zhou and Y. Yu, "Ensembling local learners through multimodal perturbation," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 35, no. 4, pp. 725–735, Aug. 2005.

[26] S. Grabowski, "Voting over multiple k-NN classifiers," in *Proc. Modern Problems Radio Eng., Telecommun. Comput. Sci.*, Feb. 2002, pp. 223–225.

[27] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, Bari, Italy, 1996, pp. 148–156, ".

[28] J. O'Sullivan, J. Langford, R. Caruana, and A. Blum, "FeatureBoost: A meta-learning algorithm that improves model robustness," in *Proc. ICML*, Jun. 2000, pp. 703–710.

[29] L. Kainulainen, Y. Miche, E. Eirola, Q. Yu, B. Frenay, E. Severin, and A. Lendasse, "Ensembles of local linear models for bankruptcy analysis and prediction," *Case Stud. Bus., Ind. Government Statist.*, vol. 4, no. 2, pp. 116–133, 2014.

[30] S. Kang and P. Kang, "Locally linear ensemble for regression," *Inf. Sci.*, vol. 432, pp. 199–209, Mar. 2018.

[31] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.

[32] A. E. Freeny, *A Portable Linear Regression Package with Test Programs*. Murray Hill, NY, USA: Bell Laboratories memorandum, 1977.

[33] R. A. Becker, J. M. Chambers, and A. R. Wilks, *The New S Language*. Belmont, CA, USA: Wadsworth, 1988.

[34] F. Mosteller and J. W. Tukey, *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA, USA: Addison-Wesley, 1977.

[35] S. Chatterjee and B. Price, *Regression Analysis by Example*, 2nd ed. New York, NY, USA: Wiley, 1977, sec. 3.7, p. 68.

[36] K. Bache and M. Lichman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California Irvine, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[37] I.-C. Yeh, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," *Cement Concrete Compos.*, vol. 29, no. 6, pp. 474–480, Jul. 2007.

[38] C. Bailey, *Smart Exercise: Burning Fat, Getting Fit*, Boston, MA, USA: Houghton-Mifflin Co., 1994, pp. 179–186.

[39] E. V. Hatzikos, G. Tsoumakas, G. Tzanis, N. Bassiliades, and I. Vlahavas, "An empirical study on sea water quality prediction," *Knowl.-Based Syst.*, vol. 21, no. 6, pp. 471–478, Aug. 2008.

[40] D. Meyer, E. Dimitriadou, K. Hornic, A. Weingessel, and F. Leisch, *Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, document e1071 R package version 1.6-8, 2017. [Online]. Available: https://CRAN.R-project.org/package=e1071

[41] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, "Kernlab-AnS4Package for kernel methods inR," *J. Stat. Softw.*, vol. 11, no. 9, pp. 1–20, 2004. Online]. Available: http://www.jstatsoft.org/v11/i09/

[42] A. Liaw and M. Wiener, "Classification and regression by random-forest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

[43] *R: A Language and Environment for Statistical Computing*, R Found. Stat. Comput., R C Team, Vienna, Austria, 2017. [Online]. Available: https://www.R-project.org/

[44] A. Beygelzimer, S. Kakadet, J. Langford, S. Arya, and D. S. M. Li. (2013). *FNN: Fast Nearest Neighbor Search Algorithms and Applications, R Package Version 1.1*. [Online]. Available: https://CRAN.R-project.org/package=FNN

[45] S. Li. (2015). *rknn: Random KNN Classification and Regression, R Package Version 1.2-1*. [Online]. Available: https://CRAN.R-project.org/package=rknn

[46] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Comput.*, vol. 4, no. 6, pp. 888–900, Nov. 1992.

[47] G. Bontempi, H. Bersini, and M. Birattari, "The local paradigm for modeling and control: From neuro-fuzzy to lazy learning," *Fuzzy Sets Syst.*, vol. 121, no. 1, pp. 59–72, Jul. 2001.

[48] Z. Yang, Y. Liang, H. Zhang, H. Chai, B. Zhang, and C. Peng, "Robust sparse logistic regression with the $L_q$ $(0 < q < 1)$ regularization for feature selection using gene expression data," *IEEE Access*, vol. 6, pp. 68586–68595, 2018.

[49] J.-N. Sun, H.-Y. Yang, J. Yao, H. Ding, S.-G. Han, C.-Y. Wu, and H. Tang, "Prediction of cyclin protein using two-step feature selection technique," *IEEE Access*, vol. 8, pp. 109535–109542, 2020.

[50] Q. Hu, X.-S. Si, A.-S. Qin, Y.-R. Lv, and Q.-H. Zhang, "Machinery fault diagnosis scheme using redefined dimensionless indicators and mRMR feature selection," *IEEE Access*, vol. 8, pp. 40313–40326, 2020.

[51] Z. Khan, M. Naeem, U. Khalil, D. M. Khan, S. Aldahmani, and M. Hamraz, "Feature selection for binary classification within functional genomics experiments via interquartile range and clustering," *IEEE Access*, vol. 7, pp. 78159–78169, 2019.

[52] B. Chatterjee, T. Bhattacharyya, K. K. Ghosh, P. K. Singh, Z. W. Geem, and R. Sarkar, "Late acceptance hill climbing based social ski driver algorithm for feature selection," *IEEE Access*, vol. 8, pp. 75393–75408, 2020.

[53] J. Gu, L. Jiao, F. Liu, S. Yang, R. Wang, P. Chen, Y. Cui, J. Xie, and Y. Zhang, "Random subspace based ensemble sparse representation," *Pattern Recognit.*, vol. 74, pp. 544–555, Feb. 2018.