Neural Processing Letters (2021) 53:459–515
https://doi.org/10.1007/s11063-020-10380-y

On Regularization Based Twin Support Vector Regression with Huber Loss

Umesh Gupta[1] · **Deepak Gupta**[1]

Accepted: 18 October 2020 / Published online: 3 January 2021

Abstract
Twin support vector regression (TSVR) is generally employed with ε-insensitive loss function which is not well capable to handle the noises and outliers. According to the definition, Huber loss function performs as quadratic for small errors and linear for others and shows better performance in comparison to Gaussian loss hence it restrains easily for a different type of noises and outliers. Recently, TSVR with Huber loss (HN-TSVR) has been suggested to handle the noise and outliers. Like TSVR, it is also having the singularity problem which degrades the performance of the model. In this paper, regularized version of HN-TSVR is proposed as regularization based twin support vector regression (RHN-TSVR) to avoid the singularity problem of HN-TSVR by applying the structured risk minimization principle that leads to our model convex and well-posed. This proposed RHN-TSVR model is well capable to handle the noise as well as outliers and avoids the singularity issue. To show the validity and applicability of proposed RHN-TSVR, various experiments perform on several artificial generated datasets having uniform, Gaussian and Laplacian noise as well as on benchmark different real-world datasets and compare with support vector regression, TSVR, ε-asymmetric Huber SVR, ε-support vector quantile regression and HN-TSVR. Here, all benchmark real-world datasets are embedded with a different significant level of noise 0%, 5% and 10% on different reported algorithms with the proposed approach. The proposed algorithm RHN-TSVR is showing better prediction ability on artificial datasets as well as real-world datasets with a different significant level of noise compared to other reported models.

Keywords Support vector regression · Twin support vector regression · Gaussian noise · Huber loss · Laplacian noise

✉ Deepak Gupta
deepakjnu85@gmail.com; deepak@nitap.ac.in

Umesh Gupta
er.umeshgupta@gmail.com

[1] National institute of Technology Arunachal Pradesh, Yupia, PapumPare, Arunachal Pradesh 791112, India

1 Introduction

Over the last decade, support vector machine (SVM) (13) has played a leading role in the field of classification and regression problems which utilize the gist of statistical learning theory. The state of the art methods like SVM, attracts many areas such as medical imaging [40, 69], financial time series [7, 9], cybercrime [18], remote sensing applications [10, 44], sediment [29] and many more. The main aim of SVM is to obtain the optimal solution through solving the quadratic programming problems (QPPs) (Cristianini and Shawe-Taylor [15]) in the consideration of the structural risk minimization principle. SVMs are well applicable to handle small sized data samples. It is having better generalization performance in binary classification but high learning cost i.e. O (m^3) and more sensitive towards the noise of the training data sample. To reduce these drawbacks of SVMs, a twin version of SVMs for binary classification has been formulated named TWSVM by Jayadeva et al. [36]. This TWSVM is highly motivated from PSVM [23] in which it generates two non- parallel hyperplanes that are nearer to either positive or negative class, respectively and unit distance separable to each other. An experiment result related to TWSVM has been shown in the literature that the computational cost is four times superior to standard SVM because it solves a pair of small sizes QPPs rather than one complex QPP. The popularity of SVM based models increases due to its low learning cost, e.g. Kumar and Gopal [39], Gupta and Gupta [26], Gupta et al. [28], Bai et al. [3] and Tang et al. [60, 61]. In the regression problem, SVR has been used for finding the optimal regressor that consists of two sets of constraints [19]. These constraints are trying to separate the data sample in ε-insensitive field [57]. Peng [50] have formulated a twin version of SVR termed as TSVR model, influenced by the TWSVM for the regression problem. TSVR is generating two functions, which measure both ε-insensitive bound of the regressor. In the gist of TWSVM, TSVR also solves a pair of small size QPPs, unlike SVR as one large QPP, which is responsible for attaining high computational speed to TSVR. In the computational field, the main drawback is that to fit the data having different noise and outliers as well as to deal with overfitting phenomenon.

To achieve the robustness, several variants of SVR that deals with different types of noises and outliers, are discussed in the literature as Hwang et al. [35], Cui and Yan [16], Chen et al. [8], Tang et al. [59] and Chen et al. [7, 9], Yang and Xu [79] etc. Other popular models are to assign different weight values to the samples to reduce the effect of outliers as follows: Xu and Wang [75], Xu et al. [74], Tanveer et al. [62] and Mao et al. [43] etc. Some researchers include fuzzy-based membership values to training points through assigning some importance in terms of penalty to the data samples [12, 22, 6]. For avoiding the overfitting problem, some valuable work is added in the field of regression [37, 82] but some gap is still present. As we already know the importance of noise in the computation, so the right choice of loss functions which are associated with particular noise in the training data sample will give the proper direction towards optimal generalization. There are several loss functions which generally consider in different regression estimation problem such as ε- insensitive loss [13], Gaussian loss [31, 71, 73], Laplacian loss [46, 80], Pinball loss [25, 54, 78], Quadratic loss [30, 70], Huber loss [5, 8, 45, 52], Asymmetric loss [32, 76], 1- norm [41], Soft insensitive loss [11, 84], Non-convex loss [83], and Hinge loss [56].

Niu et al. [47] have proposed a new approach termed as TSVR with Huber loss (HN-TSVR) that tested on real-world datasets and Gaussian noise data. In this approach, HN-TSVR performs well in comparison to TSVR with Gaussian loss function (GN-TSVR) and TSVR. For further study, the reader can follow: [33, 34, 42, 47, 72]. To strongly handle the asymmetric noise distribution and outliers in the observed real world data, Balasundaram and Meena

[5] have proposed SVR with asymmetric Huber loss where solutions are attained by using functional iterative approach. In the similar pattern, Balasundaram and Prasad [4] have also proposed TSVR based variant named as robust TSVR with Huber loss which is solved by function iterative approach and Newton iterative with Armijo step size to less sensitivity of noise and outliers. To improve the sparseness and prediction ability of the regression model, Anand et al. [1] have improved the ε-support vector quantile regression model (ε-SVQR) by adding the regularization term for quantile estimation. The ε-SVQR model mainly focuses on the sparseness of the regression model. Gu et al. [24] also have investigated a new fast clustering-based approach for TSVR to lessen the effect of outliers and noise in the observed data examples by following the prior structural information and successive over relaxation algorithm. Due to influence of noise and outliers in observed real world data, SVR and TSVR regression based models attract in the literature [49, 51, 77, 81]; Wang et al. [66–68].

Our proposed approach, RHN-TSVR, is highly influenced by HN-TSVR approach. Here, we further improve this HN-TSVR model through the addition of the regularization term to follow the structural risk minimization principle which shows more effective and efficient generalization performance and lead to a well-posed model. The advantage of our proposed algorithm is that it contains a strong convex optimization function and works well for different types of noises and outliers. The efficacy and usability of RHN-TSVR are discussed based on various numerical experiments using twenty-four artificial generated and forty-two real-world datasets. It shows better prediction performance in the case of both types of datasets. Our proposed approach is also validated through the standard statistical test which also signifies that our approach is performed better for different types of noisy data. The attractive characteristics of proposed RHN-TSVR follow as:

1. As we know that the TSVR and HN-TSVR suffer from the possible singularity problem, so to resolve this problem, we reformulate the primal problem by adding a regularization term in the objective functions.
2. Our proposed approach RHN-TSVR is following the structured risk minimization principle that leads to our model convex and well-posed.
3. By using the Huber loss function, the proposed approach RHN-TSVR effectively deals with noisy data.
4. Our proposed approach RHN-TSVR is tested and validated on both various real world datasets having different significant level of noise 0%,5% and 10% as well as on artificial datasets.

In this paper, all the sections are organized in this manner. In Sect. 2, related work such as SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR formulations have been stated for the non-linear case. An improved and regularized version of HN-TSVR (RHN-TSVR) is proposed diligently under Sects. 3. In Sect. 4, numerical experiments on benchmark different real world and several artificial generated datasets are conducted properly while Sect. 5 dwells on conclusion and future scope.

2 Related Work

In this related work section, the mathematical formulation of standard SVR, TSVR,ε-AHSVR, ε-SVQR and HN-TSVR is derived for the non-linear case. Assume $B \in (B_1, \ldots, B_n)$ is the matrix of size $m \times n$ in which g^{th} vector is x_g^t and $y \in R^m$. For non-linear kernel function, $K(B, B^t)$ is defined for gh^{th} entry as $\{(K(B, B^t)\}_{gh} = k(x_g, x_h)$ and $K(x,$

$B^t) = (k(x, x_1), \ldots, k(x, x_m))$ be a row vector. The non-linear regression function on a training set can be obtained through mapping of data to higher-dimensional space $\phi(.)$.

$\{(x_g, y_g)\}_{g=1}^m$	Training samples	e	Vector of one's
$x_g \in R^n$	Input sample	$(a)_+ = \max\{a, 0\}$	Plus function definition
$y_g \in R$	Desired output	$\phi(.)$	Higher-dimensional feature space

2.1 Support Vector Regression (SVR)

According to Vapnik [64], standard SVR uses the ε- insensitive loss for finding the unknown value of w and b through the solution of following QPP (Cristianini and Shawe-Taylor [15]) in such a way:

$$\min_{w,b,\lambda_1,\lambda_2} \frac{1}{2}||w||^2 + C(e^t\lambda_1 + e^t\lambda_2)$$

subject to:

$$\begin{aligned} &y_g - (\phi(x_g)w + be) \leq \varepsilon e + \lambda_{1g},\ \lambda_{1g} \geq 0, \\ &(\phi(x_g)w + be) - y_g \leq \varepsilon e + \lambda_{2g}, \text{ for } g = 1, \ldots, m, \end{aligned} \tag{1}$$

where slack variables are $\lambda_1 \in (\lambda_{11}, \ldots, \lambda_{1m})^t$ and $\lambda_2 \in (\lambda_{21}, \ldots, \lambda_{2m})^t$; input parameters are $C > 0$, $\varepsilon > 0$.

After introducing the Lagrange's multipliers η_1, η_2 and apply sufficient conditions, the Wolfe dual of the Eq. (1) is given as:

$$\min_{\eta_1,\eta_2 \in R^m} \frac{1}{2} \sum_{g,h=1}^{m} (\eta_{1g} - \eta_{2g})k(x_g, x_h)(\eta_{1h} - \eta_{2h}) + \varepsilon \sum_{g=1}^{m} (\eta_{1g} + \eta_{2g}) - \sum_{g=1}^{m} y_g(\eta_{1g} - \eta_{2g})$$

subject to:

$$\begin{aligned} &\sum_{g=1}^{m} e^t(\eta_{1g} - \eta_{2g}) = 0, \\ &\text{and} \quad 0 \leq \eta_1, \eta_2 \leq Ce. \end{aligned} \tag{2}$$

The non-linear decision regression function $f(.)$ will be calculated by solving the dual problem as represented in Eq. (2) (Cristianini and Shawe-Taylor [15]) for any new input $x^* \in R^n$ as

$$f(x^*) = \sum_{z=1}^{m} (\eta_{1z} - \eta_{2z})k(x^*, x_z) + b. \tag{3}$$

For more details, see [15, 19].

2.2 Twin Support Vector Regression (TSVR)

The twin model of SVR named TSVR is proposed [50] in which it finds a pair of non-parallel kernel generating functions related to both ε-insensitive down-bound $f_1(x) = K(x^t, B^t)w_1 + b_1$ and up-bound function $f_2(x) = K(x^t, B^t)w_2 + b_2$, where $w_1, w_2 \in R^m$,$b_1, b_2 \in R$. The formulation of TSVR in primal form is given as:

$$\min_{w_1, b_1, \lambda_1 \in R^{2m+1}} \frac{1}{2}||y - \varepsilon_1 e - (K(B,\ B^t)w_1 + b_1 e)||^2 + C_1 e^t \lambda_1$$

subject to:

$$y - (K(B, B^t)w_1 + b_1 e) \geq \varepsilon_1 e - \lambda_1,\ \lambda_1 \geq 0 \tag{4}$$

and

$$\min_{w_2, b_2, \lambda_2 \in R^{2m+1}} \frac{1}{2}||y + \varepsilon_2 e - (K(B,\ B^t)w_2 + b_2 e)||^2 + C_2 e^t \lambda_2$$

subject to:

$$(K(B, B^t)w_2 + b_2 e) - y \geq \varepsilon_2 e - \lambda_2,\ \lambda_2 \geq 0 \tag{5}$$

where the slack variables are $\lambda_1,\ \lambda_2$; input parameters $C_1,\ C_2 > 0,\ \ \varepsilon_1,\ \varepsilon_2 > 0$ are chosen a priori.

Similarly, solve unconstrained optimization problems as mentioned in Eqs. (4) and (5) with Lagrange's multiplier $\eta_1,\ \eta_2$ and apply Karush–Kuhn–Tucker (KKT) sufficient conditions, we get

$$\max_{w_1, b_1 \in R^{m+1}} \frac{1}{2}\eta_1^t Z(Z^t Z)^{-1} Z^t \eta_1 + (y - \varepsilon_1 e)^t Z(Z^t Z)^{-1} Z^t \eta_1 - (y - \varepsilon_1 e)^t \eta_1$$
$$\text{subject to}: \ 0 \leq \eta_1 \leq C_1 e, \tag{6}$$

and

$$\max_{w_2, b_2 \in R^{m+1}} \frac{1}{2}\eta_2^t Z(Z^t Z)^{-1} Z^t \eta_2 + (y + \varepsilon_2 e)^t Z(Z^t Z)^{-1} Z^t \eta_2 + (y + \varepsilon_2 e)^t \eta_2$$
$$\text{subject to}: \ 0 \leq \eta_2 \leq C_2 e, \tag{7}$$

where $Z = [\ K(B,\ B^t)\ e\ ]$ is the augmented matrix.

The values $w_1,\ w_2,\ b_1$ and b_2 are obtained from Eqs. (6) and (7) as follows:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (Z^t Z + \delta I)^{-1} Z^t (u_1 - \eta_1) \text{ and } \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Z^t Z + \delta I)^{-1} Z^t (u_2 + \eta_2), \tag{8}$$

where $u_1 = y - e\varepsilon_1$ and $u_2 = y + e\varepsilon_2$.

We add δI a term in the matrix $(Z^t Z)^{-1}$ with small value where $\delta > 0$. Its final prediction value for a new test sample can be obtained to take the average of functions $f_1(x)$ and $f_2(x)$.

2.3 ε-Insensitive Asymmetric Huber Support Vector Regression (ε-AHSVR)

The ε-insensitive asymmetric Huber based SVR is proposed by Balasundaram and Meena [5] in which it finds $w \in R^m$ and $b \in R$ through the solution of QPPs using Huber loss in such a way:

$$\min_{(w,b) \in R^{m+1}} \frac{1}{2}(w^t w + b)^2 + \frac{1}{2}C[||(y - ((K(B, B^t)w + be) - \varepsilon_\alpha e)_+||^2$$
$$- ||(y - ((K(B, B^t)w + be) - (\varepsilon_\alpha + \zeta_\alpha)e)_+||^2 + ||(((K(B, B^t)w + be) - y - \varepsilon_\beta e)_+||^2$$
$$- ||(((K(B, B^t)w + be) - y - (\varepsilon_\beta + \zeta_\beta)e)_+||^2] \tag{9}$$

where $\varepsilon_\alpha,\ \varepsilon_\beta > 0; \zeta_\alpha,\ \ \zeta_\beta > 0; C > 0$ are inputs.

One can rewrite the (9) problem by considering $\varepsilon_\alpha = \varepsilon_\beta = \varepsilon$, $Z = [\, K(B,\, B^t)\; e\,]$ and $\psi = \begin{bmatrix} w \\ b \end{bmatrix}$ in this manner as follows:

$$\min_{\psi \in R^{m+1}} \frac{1}{2}\psi^t\psi + \frac{1}{2}C[||(y - Z\psi - \varepsilon e)_+||^2 - ||(y - Z\psi - (\varepsilon + \zeta_\alpha)e))_+||^2 + ||(Z\psi - y - \varepsilon e)_+||^2 - ||(Z\psi - y - (\varepsilon + \zeta_\beta)e)_+||^2] \tag{10}$$

To solve the (10) by following the function iterative approach as $\psi^{g+1} = \left(\frac{I}{C} + Z^g Z\right)^{-1} Z \left[y + \frac{|y - Z\psi^g - \varepsilon e| - |Z\psi^g - y - \varepsilon e|}{2} + (Z\psi^g - y - (\varepsilon + \zeta_\alpha)e)_+ - (y - Z\psi^g - (\varepsilon + \zeta_\beta)e)_+\right]$ for $g = 0, 1, \ldots$

2.4 ε-Insensitive Support Vector Quantile Regression (ε-SVQR)

A novel ε-SVQR model uses the ε-insensitive pinball loss function for quantile estimation [1] to find the unknown value of w and b through the solution of QPPs in such a way:

$$\min_{w, b, \lambda_1, \lambda_2} \frac{1}{2}||w||^2 + C\sum_{g=1}^{m}(\theta\lambda_{1g} + (1-\theta)\lambda_{2g})$$

subject to:

$$y_g - (\phi(x_g)w + be) \le \varepsilon(1-\theta) + \lambda_{1g},\ \lambda_{1g} \ge 0$$

and

$$(\phi(x_g)w + be) - y_g \le \theta\varepsilon + \lambda_{2g},\ \lambda_{1g} \ge 0,\ \lambda_{2g} \ge 0 \text{ for } g = 1, \ldots, m \tag{11}$$

where slack variables are $\lambda_{1g} := \frac{\lambda_{1g}}{\theta}$ and $\lambda_{2g} := \frac{\lambda_{2g}}{1-\theta}$; $\theta > 0$ is the quantile; input parameters are $C > 0$, $\varepsilon > 0$.

After introducing the Lagrange's multipliers η_1, η_2 and apply sufficient conditions, the Wolfe dual of the Eq. (11) is given as:

$$\min_{\eta_1, \eta_2 \in R^m} \frac{1}{2}\sum_{g,h=1}^{m}(\eta_{1g} - \eta_{2g})k(x_g, x_h)(\eta_{1h} - \eta_{2h}) - \sum_{g=1}^{m}(\eta_{1g} - \eta_{2g})y_g + \sum_{g=1}^{m}((1-\theta)\varepsilon\eta_{1g} + \theta\varepsilon\eta_{2g})$$

subject to:

$$\sum_{g=1}^{m}(\eta_{1g} - \eta_{2g}) = 0,$$
$$\text{and } 0 \le \eta_{1_g} \le C\theta,\ g = 1, 2, \ldots, m,$$
$$0 \le \eta_{2g} \le C(1-\theta),\ g = 1, 2, \ldots, m, \tag{12}$$

where the kernel function is $k(x_g, x_h) = \phi(x_g)^t\phi(x_h)$.

The non-linear decision regression function $f(.)$ will be calculated by solving the dual problem as represented in Eq. (12) for any new input $x^* \in R^n$ as

$$f(x^*) = \sum_{g=1}^{m}(\eta_{1g} - \eta_{2g})k(x^*, x_g) + b. \tag{13}$$

2.5 Twin Support Vector Regression with Huber Loss (HN-TSVR)

In order to improve the prediction ability, a hybrid approach is proposed by combination of Huber loss function and Twin model of SVR, termed as TSVR with Huber loss (HN-TSVR) in this sub-section.

Huber loss function [47]

$$c(\lambda_g) = \begin{cases} \frac{\lambda_g^2}{2}, & if\ \lambda_g \le \varepsilon \\ \varepsilon|\lambda_g| - \frac{\varepsilon^2}{2}, & otherwise \end{cases} where\ \varepsilon = \varepsilon_1^* \qquad c(\zeta_g) = \begin{cases} \frac{\zeta_g^2}{2}, & if\ \zeta_g \le \varepsilon \\ \varepsilon|\zeta_g| - \frac{\varepsilon^2}{2}, & otherwise \end{cases} where\ \varepsilon = \varepsilon_2^*$$

where ε_1^*, ε_2^* are input parameters.

HN-TSVR regression involves the following optimization problems as

$$\min_{w_1,b_1,\lambda} \frac{1}{2}||y - \varepsilon_1 e - (K(B, B^t)w_1 + b_1 e)||^2 + C_1 e^t\Big(\sum_{g\in D_1}\frac{1}{2}\lambda_g^2 + \varepsilon \sum_{g\in D_1'}(\lambda_g - \frac{1}{2}\varepsilon)\Big)$$

$$\text{subject to: } y - (K(B, B^t)w_1 + b_1 e) \ge \varepsilon_1 e - \lambda,\ \lambda \ge 0, \tag{14}$$

and

$$\min_{w_2,b_2,\zeta} \frac{1}{2}||y + \varepsilon_2 e - (K(B, B^t)w_2 + b_2 e)||^2 + C_2 e^t\Big(\sum_{g\in D_2}\frac{1}{2}\zeta_g^2 + \varepsilon \sum_{g\in D_2'}(\zeta_g - \frac{1}{2}\varepsilon)\Big)$$

$$\text{subject to: } (K(B, B^t)w_2 + b_2 e) - y \ge \varepsilon_2 e - \zeta,\ \zeta \ge 0, \tag{15}$$

where $D_1 = \{g|\ 0 \le \lambda_g < \varepsilon\}$ and $D_1' = \{g|\ \lambda_g \ge \varepsilon\}$; $D_2 = \{g|\ 0 \le \zeta_g < \varepsilon\}$ and $D_2' = \{g|\ \zeta_g \ge \varepsilon\}$. The slack variables are λ, ζ.

So, derive the dual formulation with sufficient conditions are shown as

$$\min_{\eta_1} \frac{1}{2}\eta_1^t Z(Z^t Z)^{-1}Z^t\eta_1 - (y - \varepsilon_1 e)^t Z(Z^t Z)^{-1}Z^t\eta_1 + (y - \varepsilon_1 e)^t\eta_1 + \frac{1}{2C_1}\eta_1^t\eta_1$$

subject to:

$$0 \le \eta_1 \le C_1\varepsilon_1^* e, \tag{16}$$

and

$$\min_{\eta_2} \frac{1}{2}\eta_2^t Z(Z^t Z)^{-1}Z^t\eta_2 + (y + \varepsilon_2 e)^t Z(Z^t Z)^{-1}Z^t\eta_2 - (y + \varepsilon_2 e)^t\eta_2 + \frac{1}{2C_2}\eta_2^t\eta_2$$

subject to:

$$0 \le \eta_2 \le C_2\varepsilon_2^* e, \tag{17}$$

where, $Z = [\,K(B,\ B^t)\ e\,]$ is the augmented matrix.

The values of w_1, w_2, b_1, b_2 can be solved as

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (Z^t Z)^{-1}Z^t(u_1 - \eta_1) \text{ and } \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Z^t Z)^{-1}Z^t(u_2 + \eta_2). \tag{18}$$

where $u_1 = y - e\varepsilon_1$, $u_2 = y + e\varepsilon_2$.

Further, the final prediction value can be obtained same as TSVR. For more details see Niu et al. [47].

3 Proposed Regularization Based Twin Support Vector Regression with Huber Loss (RHN-TSVR)

Twin model of SVR (TSVR) deals with ε-insensitive loss but fail to address the Gaussian noise data. To handle this problem, Niu et al. [47] have proposed an approach named HN-TSVR which deals with Huber loss function but it has failed to follow the structured minimization principle. To avoid the singularity problem of HN-TSVR, we are adding one regularization term $\frac{C_3}{2}(||w_1||^2+b_1^2)$ and $\frac{C_4}{2}(||w_2||^2+b_2^2)$ in the primal problem of (19) and (20) respectively that leads to a stable and well-posed model, named as regularization based twin support vector regression with Huber loss which follows the gist of statistical learning theory. The Mathematical formulation of RHN-TSVR is written as:

1. RHN-TSVR requires two kernel generating functions as $f_1(x) = K(x^t,\ B^t)w_1 + b_1$, and $f_2(x) = K(x^t,\ B^t)w_2 + b_2$.
2. The proposed approach involves the following optimization problems as

$$\begin{aligned}\min_{w_1,b_1,\lambda}\ & \frac{1}{2}||y-\varepsilon_1 e-(K(B,B^t)w_1+b_1e)||^2 \\ &+\ C_1e^t\left(\sum_{g\in D_1}\frac{1}{2}\lambda_i^2+\varepsilon\sum_{g\in D_1'}(\lambda_i-\frac{1}{2}\varepsilon)\right)+\frac{C_3}{2}(||w_1||^2+b_1^2 \\ \text{subject to}:\ & y-(K(B,B^t)w_1+b_1e)\geq\varepsilon_1e-\lambda,\ \lambda\geq 0, \end{aligned} \tag{19}$$

and

$$\begin{aligned}\min_{w_2,b_2,\zeta}\ & \frac{1}{2}||y+\varepsilon_2 e-(K(B,B^t)w_2+b_2e)||^2 \\ &+\ C_2e^t\left(\sum_{g\in D_2}\frac{1}{2}\zeta_i^2+\varepsilon\sum_{g\in D_2'}(\zeta_i-\frac{1}{2}\varepsilon)\right)+\frac{C_4}{2}(||w_2||^2+b_2^2) \\ \text{subject to}:\ & (K(B,B^t)w_2+b_2e)-y\geq\varepsilon_2e-\zeta,\ \zeta\geq 0, \end{aligned} \tag{20}$$

where, $D_1 = \{g|\ 0 \leq \lambda_g < \varepsilon\}$ and $D_1^{'} = \{g|\ \lambda_g \geq \varepsilon\}$; $D_2 = \{g|\ 0 \leq \zeta_g < \varepsilon\}$ and $D_2^{'} = \{g|\ \zeta_g \geq \varepsilon\}$; slack variables are λ, ζ; input parameters are $C_1,\ C_2 > 0; \varepsilon_1,\ \varepsilon_2 > 0$.

3. By introducing the Lagrangian multipliers η_1, η_2, α_1, α_2 and apply sufficient KKT conditions to the Eq. (19) and (20)

$$\begin{aligned}L_1(w_1,b_1,\eta_1,\alpha_1) = & \frac{1}{2}||(y-\varepsilon_1e-(K(B,B^t)w_1+b_1e))||^2 \\ & +C_1e^t\left(\sum_{g\in D_1}\frac{1}{2}\lambda_g^2+\varepsilon\sum_{g\in D_1'}(\lambda_g-\frac{1}{2}\varepsilon)\right) \\ & +\frac{C_3}{2}(||w_1||^2+b_1^2)-\eta_1^t(y-\varepsilon_1e-(K(B,B^t)w_1+b_1e)+\lambda)-\alpha_1^t\lambda \end{aligned} \tag{21}$$

$$L_2(w_2, b_2, \eta_2, \alpha_2) = \frac{1}{2}||(y + \varepsilon_2 e - (K(B, B^t)w_2 + b_2 e))||^2 + C_2 e^t \left(\sum_{g \in D_2} \frac{1}{2}\zeta_g^2 + \varepsilon \sum_{g \in D_2^{'}} (\zeta_g - \frac{1}{2}\varepsilon) \right) + \frac{C_4}{2}(||w_2||^2 + b_2^2) - \eta_2^t(y + \varepsilon_2 e - (K(B, B^t)w_2 + b_2 e) + \zeta) - \alpha_2^t \zeta \tag{22}$$

4a. Then, we find the gradient of (21) according to KKT conditions with respect to w_1, b_1 and λ:

$$\frac{\partial L_1}{\partial w_1} = -K(B,\ B^t)^t(y - K(B,\ B^t)w_1 - b_1 e - \varepsilon_1 e) + K(B,\ B^t)^t \eta_1 + C_3 w_1^t = 0,$$

$$\frac{\partial L_1}{\partial b_1} = -e^t(y - \varepsilon_1 e - K(B,\ B^t)w_1 - b_1 e) + e^t \eta_1 + C_3 b_1 = 0,$$

$$\frac{\partial L_1}{\partial \lambda} = C_1 v_g - \eta_{1g} - \alpha_{1g} = 0.$$

where $v_g = \frac{\partial(c(\lambda_g))}{\partial(\lambda_g)} = \begin{cases} \lambda_g \ if\ g \in D_1 \\ \varepsilon \quad if\ g \in D_1^{'} \end{cases}, for\ g \in D_1.$

Here, we have $\lambda_g < \varepsilon$, thus $v_g \leq \varepsilon$. Also, $\alpha_{1g} \geq 0$, then we can get $0 \leq \eta_{1g} \leq C_1 v_g$. Therefore we can conclude that $0 \leq \eta_{1g} \leq C_1 \varepsilon$.

4b. Similar to (21), find the gradient of Eq. (22) with respect to w_2, b_2 and ζ:

$$\frac{\partial L_2}{\partial w_2} = -K(B,\ B^t)^t(y - K(B,\ B^t)w_2 - b_2 e + \varepsilon_2 e) + K(B,\ B^t)^t \eta_2 + C_4 w_2^t = 0,$$

$$\frac{\partial L_2}{\partial b_2} = -e^t(y + \varepsilon_2 e - K(B,\ B^t)w_2 - b_2 e) + e^t \eta_2 + C_4 b_2 = 0,$$

$$\frac{\partial L_2}{\partial \zeta} = C_2 v_g^{'} - \eta_{2g} - \alpha_{2g} = 0.$$

where, $v_g^{'} = \frac{\partial(c(\zeta_g))}{\partial(\zeta_g)} = \begin{cases} \zeta_g \ if\ i \in D_2 \\ \varepsilon \quad if\ i \in D_2^{'} \end{cases}, for\ i \in D_2.$

Here, we have $\zeta_g < \varepsilon$, thus $v_g^{'} \leq \varepsilon$. Also,$\alpha_{2g} \geq 0$, then we can get $0 \leq \eta_{2g} \leq C_2 v_g^{'}$. Therefore we can conclude that $0 \leq \eta_{2g} \leq C_2 \varepsilon$.

5a. By following the same approach [47], we get the dual formulation of (19) as

$$\min_{\eta_1} \ \frac{1}{2}\eta_1^t Z(Z^t Z + C_3 I)^{-1} Z^t \eta_1 - (y - \varepsilon_1 e)^t Z(Z^t Z + C_3 I)^{-1} Z^t \eta_1 + (y - \varepsilon_1 e)^t \eta_1 + \frac{1}{2C_1}\eta_1^t \eta_1$$
$$\text{subject to}: \ 0 \leq \eta_1 \leq C_1 \varepsilon_1^* e, \tag{23}$$

where, $Z = [\, K(B,\ B^t)\ e\,]$ is the augmented matrix.

5b. Similar to above, we get the dual formulation of (20) as

$$\min_{\eta_2} \ \frac{1}{2}\eta_2^t Z(Z^t Z + C_4 I)^{-1} Z^t \eta_2 - (y + \varepsilon_2 e)^t Z(Z^t Z + C_4 I)^{-1} Z^t \eta_2 + (y + \varepsilon_2 e)^t \eta_2 + \frac{1}{2C_2}\eta_2^t \eta_2$$
$$\text{subject to}: \ 0 \leq \eta_2 \leq C_2 \varepsilon_2^* e, \tag{24}$$

6 The values of w_1, w_2, b_1, b_2 can be obtained as

$$\psi_1 = \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (Z^t Z + C_3 I)^{-1} Z^t (u_1 - \eta_1),$$

and

$$\psi_2 = \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Z^t Z + C_4 I)^{-1} Z^t (u_2 + \eta_2) \tag{25}$$

where $u_1 = y - \varepsilon_1 e$, $u_2 = y + \varepsilon_2 e$.

7 The end regressor value can be determined for a new test sample to take the average of functions $f_1(x)$ and $f_2(x)$.

$$f(x) = \frac{f_1(x) + f_2(x)}{2}. \tag{26}$$

One can follow the Algorithm 3.1 of proposed approach RHN-TSVR as:

Algorithm 3.1 *Non-linear RHN-TSVR*

Input: Initialize the input parameters $C_1, C_2, C_3, C_4 > 0$; $\varepsilon_1, \varepsilon_2 > 0$; slack variables are λ, ζ; $D_1 = \{g \mid 0 \le \lambda_g < \varepsilon\}$, $D_1^{'} = \{g \mid \lambda_g \ge \varepsilon\}$, $D_2 = \{g \mid 0 \le \zeta_g \ge \varepsilon\}$, $D_2^{'} = \{g \mid \zeta_g \ge \varepsilon\}$ and the input kernel matrix $K(B, B^t)$.

Output: The final regressor function is given by equation (26) where the values of (w_1, b_1) and (w_2, b_2) are computed from equation (25).

Process:

Training

Step1: Construct the primal problems (19) and (20) of proposed RHN-TSVR by adding $\frac{C_3}{2}(\| w_1 \|^2 + b_1^2)$ and $\frac{C_4}{2}(\| w_2 \|^2 + b_2^2)$ in the formulation of HN-TSVR (14) and (15) respectively;

Step2: Apply the Lagrangian multiplier $\eta_1, \eta_2, \alpha_1, \alpha_2$ and apply sufficient KKT conditions to (19) and (20) respectively to obtain as (21) and (22) correspondingly;

Step3: Calculate the gradient with respect to (w_1, b_1, λ) and (w_2, b_2, ζ) then equate them to 0;

Step4: Find the dual of the (19) and (20) as given in (23) and (24) respectively;

Step5: Solve $\psi_1 = \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (Z^t Z + C_3 I)^{-1} Z^t (y - \varepsilon_1 e - \eta_1)$ and $\psi_2 = \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Z^t Z + C_4 I)^{-1} Z^t (y + \varepsilon_2 e + \eta_2)$

where $Z = [K(B, B^t) \quad e]$ is the augmented matrix;

Step6: By solving step 5, one can find the value of (ψ_1, ψ_2).

Testing

Step1: For any test example, determine the predicted value by using the end regressor from (26).

Remark One can observe that for TWSVM, the authors in [55] gave an improvement by adding a regularization term in the objective function aiming at minimizing the structural risk by maximizing the margin. This method is called TBSVM, where the bias term is also penalized. But penalizing the bias term will not affect the result significantly and will change the optimization problem slightly. From a geometric point of view, it is sufficient to penalize the norm of w in order to maximize the margin [45].

Table 1 All parameters with their range and concerned algorithms

Parameters	Range	Model
$C, C_1 = C_2, C_3 = C_4$	$\{10^{-5}, \ldots, 10^{5}\}$	SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR, RHN-TSVR
μ	$\{2^{-5}, \ldots, 2^{5}\}$	SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR, RHN-TSVR
ε	$\{0.1\}$	SVR
$\varepsilon_1,\ \varepsilon_2$	$\{0.001,\ 0.01,\ 0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$	TSVR
$(\varepsilon_\alpha = \varepsilon_\beta)$	$\{0.001,\ 0.01,\ 0.1\}$	ε-AHSVR
ε	$\{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$	ε-SVQR
$(\varepsilon_1 = \varepsilon_2), (\varepsilon_1^* = \varepsilon_2^*)$	$\{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$	HN-TSVR, RHN-TSVR
$\zeta_\alpha, \zeta_\beta$	$\{0.1,\ 1.0,\ 1.345\}$	ε-AHSVR
θ	$\{0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7, 0.8\}$	ε-SVQR

4 Numerical Experiments and Results

In this section, we have demonstrated a number of experiments to validate the efficacy of our new proposed approach RHN-TSVR in comparison to existing approaches such as SVR, TSVR,ε-AHSVR, ε-SVQR and HN-TSVR on various artificial datasets having different types of noises and real-world datasets at significant noise level 0%, 5% and 10%.

4.1 Experimental Setup

To perform this experiment, some hardware and software are required to execute the numerical experiment such as one desktop PC, one CPU with 4 Gigabyte RAM and a high-speed 64-bit processor as i5@intel 3.20 GHz, operating system @ Microsoft windows 10 and MATLAB software. An optimization toolbox MOSEK (accessible from https://www.mosek.com) is also needed to solve the QPP in case of SVR, TSVR, ε-SVQR and HN-TSVR and RHN-TSVR models. In this paper, Gaussian kernel function is considered for nonlinear consideration as $K(x_{z_1}, x_{z_2}) = \exp\left(-\mu||x_{z_1} - x_{z_2}||^2\right)$, for $z_1, z_2 = 1, \ldots, m$, where kernel parameter $\mu > 0$. Here, all the parameters with their set of ranges corresponding to concerned algorithms are defined in Table 1. The 10-fold cross-validation is applied for all interested concerned algorithms on benchmark standard real world as well as artificial generated datasets.

Here, the root mean square error (RMSE) is considered to compute the prediction error for all interested algorithms on test data. Its formula is shown as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{z=1}^{N}\left(y_z - \hat{y}_z\right)^2}$$

where y_z considered as observed data, $\hat{y}_z$ considered as predicted data and N considered total test data.

4.2 Artificial Data Set

4.2.1 Uniform and Gaussian noise

We generate artificial datasets to test the effectiveness of our proposed RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR whose function definitions are mentioned in Table 2. In Table 2, two types of noise are considered: I) Uniform noise $\Theta \in U\ (a,\ b)$ with interval $(a,\ b)$; II) Gaussian noise $\Theta \in N\ (\mu,\ \sigma^2)$ with mean μ and variance σ^2. In Table 2, Functions from 1 to 14 are using uniform variability of noise with symmetric distribution and Functions from 15 to 18 are having the heteroscedastic noise structure in which noise is computed on the basis of the value of input sample.

The performance analysis of RHN-TSVR model with standard SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR using Gaussian kernel on artificially generated datasets is tabulated in Table 3. Table 4 contains the average ranks of all concerned algorithms over artificially generated datasets for the Gaussian kernel. The proposed algorithm RHN-TSVR has the lowest rank among SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR in Table 4 that signifies our approach is far better than others. The performance of RHN-TSVR is better in 10 out of 18 artificial functions for both types of noise either uniform or Gaussian noise from Table 3. It also shows prominent impact in both uniform variabilities of noise as well as heteroscedastic noise structure.

Figures 1 and 2 are plotted corresponding to artificial Function 13 and Function 14 from Table 3 in order to show the uniform variability of noise with symmetric distribution respectively. Figures 3 and 4 are plotted corresponding to artificial Functions 15 and Function 16 from Table 3 results using Gaussian kernel in order to show the predictions with heteroscedastic noise structure respectively. One can find from Fig. 1, 2, 3 and 4 that our proposed approach RHN-TSVR is having closed relationship with the original one in comparison to other reported approaches. Hence, one can say that RHN-TSVR is very well capable to deal with uniform noise as well as heteroscedastic noise structure if present in the dataset.

4.2.2 Laplacian Noise

Further, we generate artificial datasets having another type noise i.e. Laplacian noise to validate our RHN-TSVR approach with SVR, TSVR,ε-AHSVR, ε-SVQR, and HN-TSVR whose function definitions are mentioned in Table 5. The Laplacian noise is considered as $\Psi \in L\ (\mu,\ b)$ with interval $(0,\ \ 1)$.

The prediction performance of RHN-TSVR to conventional SVR, TSVR, ε-AHSVR, ε-SVQR, and HN-TSVR using Gaussian kernel on artificially generated datasets have been tabulated in Table 6. Here, proposed RHN-TSVR shows better prediction capability in 4 cases among 6. Further, the average ranks of all models are computed in Table 7. The proposed algorithm RHN-TSVR has the lowest rank among all that signifies our approach outperforms to others. We have plotted Fig. 5 corresponding to Function 19 to understand the close relationship between prediction and original values. In this case, our proposed RHN-TSVR is having similar performance as HN-TSVR.

Table 2 Different artificially generated functions using Uniform noise and Gaussian noise with their definition and domain of definition

Function name	Function definition	Domain of definition	Noise Type
Function1	$f(x) = \left(\frac{4}{\lvert x_1\rvert+2}\right) + \cos(2x_1) + \sin(3x_1) + \Theta$	$x_1 \in [-10,\ 10]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function2			Type B:$\Theta \in N(0,\ 0.1^2)$
Function3	$f(x_1, x_2, x_3, x_4, x_5) = 0.79+1.27x_1x_2+1.56x_1x_4$	$x_i \in [0, 1]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function4	$+3.42x_2x_5+2.06x_3x_4x_5+\Theta$	$i \in \{1,2,3,4,5\}$	Type B:$\Theta \in N(0,\ 0.1^2)$
Function5	$f(x_1, x_2) = 42.659(0.1 + x_1(0.05 + x_1^4 - 10x_1^2x_2^2 + 5x_2^4)) + \Theta$	$x_1, x_2 \in [-0.5, 0.5]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function6			Type B:$\Theta \in N(0,\ 0.1^2)$
Function7	$f(x_1, x_2, x_3, x_4, x_5) = 10\sin\pi x_1x_2 + 20(x_3-0.5)^2 + 10x_4 + 5x_5 + \Theta$	$x_i \in [0, 1]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function8		$i = 1, 2, 3, 4, 5$	Type B:$\Theta \in N(0,\ 0.1^2)$
Function9	$f(x_1) = \left(\frac{4}{\lvert x_1\rvert+2}\right) + \cos(2x_1) + \sin(3x_1) + \Theta$	$x_1 \in [-10,\ 10]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function10			Type B:$\Theta \in N(0,\ 0.2^2)$
Function11	$f(x_1, x_2) = 1.3356(\exp(3(x_2 - 0.5))\sin(4\pi(x_2 - 0.9))$	$x_i \in [0, 1]$	Type A:$\Theta \in U(-0.2,\ 0.2)$
Function12	$+1.5(1 - x_1)) + 1.5(1 - x_1) + \exp(2x_1 - 1)$	$i \in \{1,2\}$	Type B:$\Theta \in N(0,\ 0.2^2)$
	$\sin(3\pi(x_1 - 0.6)(x_1 - 0.6)) + \Theta$		
Function13	$f(x_1) = \left(\frac{1}{0.3\sqrt{2\pi}}\right)\exp\left(-((x_1 - 2)(x_1 - 2))/\left(2\left(0.3^2\right)\right)\right)$	$x_1 \in [0, 10]$	Type A:$\Theta \in U(-0.5,\ 0.5)$
Function14	$+\left(\frac{1}{1.2\sqrt{2\pi}}\right)\exp\left(-((x_1 - 7)(x_1 - 7))/\left(2\left(1.2^2\right)\right)\right) + \Theta$		Type B:$\Theta \in N(0,\ 0.1^2)$
Function15	$f_1(x) = \frac{\sin(x)}{x}$ such that $y_i = f_1(x_i) + \left(0.5 - \frac{\lvert x_i\rvert}{8\pi}\right)\Theta_i$	$x_i \in U(-4\pi,\ 4\pi)$,	Type A: $\Theta \in U(-1,\ 1)$
Function16		$i = 1, 2, \ldots, 200$	Type B: $\Theta \in N(0,\ 0.5^2)$
Function17	$f(x) = x + 2\exp(-16x^2)$	$x_i = 0.01(i - 1) - 1$,	Type A: $\Theta \in U(-1,\ 1)$
Function18	such that $y_i = f(x_i) + (x_i+0.5)\Theta_i$	$i = 1, 2, \ldots, 200$	Type B: $\Theta \in N(0,\ 0.5^2)$

Table 3 Performance comparison of RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR using Gaussian kernel on synthetic data sets with Uniform and Gaussian noise

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Function1 (150X2,500X2)	**0.060254** (10^1,0.1,2^0) 0.24846	0.07662 (10^0,10^{-3},2^0) 0.07161	0.102363 (10^3,10^{-3}, 1.345, 1.345, 2^1) 0.00237	0.08825 (10^1,0.1,0.3,2^1) 0.11429	0.07716 (10^1, 0.1, 0.1, 2^0) 0.13004	0.12935 (10^{-5},10^{-5}, 0.9, 0.1, 2^0) 0.0068301
Function2 (150X2,500X2)	0.084763 (10^5,0.1,2^{-1}) 0.27253	0.02778 (10^0,10^{-1},2^0) 0.09476	0.025112 (10^5,10^{-3}, 1, 1.345, 2^1) 0.00214	0.04269 (10^3,0.1,0.3,2^{-1}) 0.13124	0.02775 (10^1, 0.9, 0.3, 2^0) 0.10073	**0.02199** (10^1,10^{-5}, 0.9, 0.9, 2^1) 0.00916
Function3 (150X6,500X6)	0.01362 (10^5,0.1,2^{-3}) 0.23213	0.01921 (10^3,10^{-3},2^{-2}) 0.17381	0.01156 (10^5,10^{-1}, 1, 1.345, 2^{-2}) 0.00603	0.01356 (10^5,0.3,0.5,2^{-3}) 0.06577	0.02379 (10^5, 0.5, 0.3, 2^0) 0.19114	**0.00988** (10^5,10^{-5}, 0.1, 0.3, 2^{-3}) 0.0118904
Function4 (150X6,500X6)	0.061408 (10^4,0.1,2^{-5}) 0.23359	0.01346 (10^4,10^{-1},2^0) 0.10612	0.00632 (10^5,10^{-3}, 0.1, 1.345, 2^3) 0.00569	0.02464 (10^5,0.1,0.5,2^{-3}) 0.06941	0.01333 (10^5, 0.9, 0.5, 2^0) 0.19064	**0.00602** (10^1,10^{-5}, 0.9, 0.5, 2^3) 0.0154294
Function5 (150X3,500X3)	0.051805 (10^4,0.1,2^5) 0.23124	0.03731 (10^3,10^{-1},2^5) 0.10908	**0.031527** (10^3,10^{-1}, 1, 1.345, 2^5) 0.01027	0.04023 (10^5,0.3,0.5,2^5) 0.07013	0.03731 (10^3, 0.9, 0.5, 2^5) 0.16043	0.03919 (10^5,10^{-5}, 0.9, 0.5, 2^5) 0.0139697

Table 3 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Function6 (150X3,500X3)	0.068543 (10^4,0.1,2^2) 0.24664	0.00725 (10^2,10^{-1},2^5) 0.10067	0.004275 (10^5,10^{-3}, 0.1, 1.345, 2^5) 0.00794	0.03068 (10^5,0.1,0.2,2^3) 0.07966	0.00727 (10^3, 0.3, 0.9, 2^5) 0.12189	**0.00406** (10^0,10^{-5}, 0.1, 0.5, 2^5) 0.015654
Function7 (150X6,500X6)	0.017707 (10^4,0.1,2^{-4}) 0.21608	0.008 (10^3,10^{-3},2^1) 0.10053	0.010915 (10^5,10^{-1}, 1, 1.345, 2^{-4}) 0.01112	0.02645 (10^5,0.1,0.6,2^{-5}) 0.07513	0.00799 (10^5, 0.1, 0.1, 2^1) 0.12071	**0.00795** (10^5,10^{-5}, 0.1, 0.7, 2^0) 0.0161369
Function8 (150X6,500X6)	0.062695 (10^3,0.1,2^1) 0.23092	0.00418 (10^1,10^{-1},2^3) 0.10265	0.002216 (10^5,10^{-3}, 0.1, 1.345, 2^2) 0.00669	0.02145 (10^5,0.1,0.1,2^{-1}) 0.07909	0.00418 (10^5, 0.5, 0.3, 2^3) 0.10959	**0.00169** (10^1,10^{-5}, 0.1, 0.1, 2^0) 0.0169987
Function9 (150X2,500X2)	0.088101 (10^2,0.1,2^0) 0.27421	**0.08105** (10^0,10^{-1},2^0) 0.07858	0.083806 (10^2,10^{-3}, 1.345, 1.345, 2^1) 0.00213	0.09462 (10^3,0.1,0.5,2^{-1}) 0.12864	0.08129 (10^1, 0.9, 0.1, 2^0) 0.07415	0.08129 (10^1,10^{-3}, 0.9, 0.1, 2^0) 0.0151367
Function10 (150X2,500X2)	0.078086 (10^2,0.1,2^0) 0.48633	0.0158 (10^2,10^{-3},2^1) 0.07912	0.011333 (10^5,10^{-3}, 0.1, 1.345, 2^1) 0.0013	0.04075 (10^5,0.1,0.2,2^{-1}) 0.14998	0.01579 (10^5, 0.1, 0.9, 2^1) 0.06906	**0.0112** (10^{-5},10^{-5}, 0.1, 0.1, 2^1) 0.0126496
Function11 (150X3,500X3)	0.027556 (10^5,0.1,2^2) 0.25697	**0.01022** (10^5,10^{-1},2^4) 0.10644	0.016189 (10^5,10^{-1}, 1, 1.345, 2^3) 0.01082	0.04191 (10^5,0.3,0.1,2^5) 0.07923	0.0138 (10^5, 0.1, 0.7, 2^3) 0.15925	0.01293 (10^5,10^{-5}, 0.9, 0.3, 2^3) 0.017398
Function12 (150X3,500X3)	0.061853 (10^4,0.1,2^3) 0.28748	0.00696 (10^0,10^{-1},2^5) 0.08644	**0.004961** (10^5,10^{-3}, 0.1, 1.345, 2^4) 0.00726	0.03326 (10^3,0.1,0.1,2^3) 0.07364	0.00689 (10^3, 0.9, 0.1, 2^5) 0.09879	0.00531 (10^0,10^{-5}, 0.9, 0.3, 2^5) 0.0173283

Table 3 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Function13 (200X2,450X2)	0.136885 (10^0,0.1,2^1) 0.26544	0.11783 (10^1,10^{-3},2^{-1}) 0.12948	**0.08155** (10^1,10^{-1}, 1, 1.345, 2^1) 0.00388	0.10239 (10^1,0.9,0.2,2^1) 0.12211	0.11656 (10^5, 0.1, 0.5, 2^{-1}) 0.10624	0.08749 (10^5,10^1, 0.3, 0.5, 2^1) 0.0225593
Function14 (200X2,500X2)	0.186935 (10^0,0.1,2^4) 0.25514	0.12023 (10^5,10^{-3},2^0) 0.10377	0.184837 (10^1,10^{-1}, 1, 1.345, 2^5) 0.00275	0.15779 (10^1,0.5,0.1,2^5) 0.10492	0.12034 (10^3, 0.1, 0.7, 2^0) 0.09102	**0.10566** (10^5,10^1, 0.1, 0.9, 2^3) 0.0235153
Function15 (200X2,500X2)	0.034488 (10^2,0.1,2^{-5}) 0.46716	0.01879 (10^3,10^{-1},2^{-5}) 0.16623	0.019379 (10^1,10^{-1}, 1, 1.345, 2^{-4}) 0.00225	0.02682 (10^3,0.3,0.5,2^{-5}) 0.22456	**0.01878** (10^5, 0.9, 0.7, 2^{-5}) 0.30141	**0.01878** (10^5,10^{-3}, 0.9, 0.7, 2^{-5}) 0.024471
Function16 (200X2,500X2)	0.036077 (10^2,0.1,2^{-3}) 0.47924	0.02116 (10^{-5},10^{-3},2^{-3}) 0.15487	0.0222 (10^3,10^{-2}, 1, 1.345, 2^{-3}) 0.0046	0.02166 (10^3,0.1,0.5,2^{-5}) 0.22893	0.0219 (10^0, 0.9, 0.1, 2^{-3}) 0.236	**0.01955** (10^5,10^1, 0.1, 0.5, 2^{-3}) 0.028278
Function17 (200X2,500X2)	0.147868 (10^0,0.1,2^3) 0.46149	0.15262 (10^{-5},10^{-1},2^1) 0.143	0.100892 (10^1,10^{-1}, 1.345, 1.345, 2^4) 0.00233	0.18809 (10^1,0.7,0.8,2^5) 0.24851	0.15262 (10^{-5}, 0.1, 0.1, 2^1) 0.14581	**0.07032** (10^1,10^{-1}, 0.9, 0.5, 2^3) 0.0183895
Function18 (200X2,500X2)	0.052586 (10^1,0.1,2^3) 0.44357	0.05026 (10^0,10^{-3},2^3) 0.14172	0.044265 (10^2,10^{-2}, 1, 1.345, 2^3) 0.0035	**0.03809** (10^1,0.1,0.5,2^3) 0.21918	0.05027 (10^1, 0.9, 0.1, 2^3) 0.2795	0.05027 (10^1,10^{-3}, 0.9, 0.1, 2^3) 0.0184345

The best result is shown as boldface

Table 4 Average ranks of proposed RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR on RMSE using Gaussian kernel on synthetic data sets with Uniform and Gaussian noise

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Function1	1	2	5	4	3	6
Function2	6	4	2	5	3	1
Function3	4	5	2	3	6	1
Function4	6	4	2	5	3	1
Function5	6	2.5	1	5	2.5	4
Function6	6	3	2	5	4	1
Function7	5	3	4	6	2	1
Function8	6	3.5	2	5	3.5	1
Function9	5	1	4	6	2.5	2.5
Function10	6	4	2	5	3	1
Function11	5	1	4	6	3	2
Function12	6	4	1	5	3	2
Function13	6	5	1	3	4	2
Function14	6	2	5	4	3	1
Function15	6	3	4	5	1.5	1.5
Function16	6	2	5	3	4	1
Function17	3	4.5	2	6	4.5	1
Function18	6	3	2	1	4.5	4.5
Average rank	5.2777778	3.1388889	2.7777778	4.5555556	3.3333333	**1.9166667**

Table 5 Different artificially generated functions having Laplacian noise with their definition and domain of definition

Function name	Function definition	Domain of definition
Function19	$f(x) = \left(\frac{4}{\lvert x_1\rvert+2}\right) + \cos(2x_1) + \sin(3x_1) + \Psi$	$x_1 \in [-10,\ 10]$
Function20	$f(x) = ((1 + \sin(2x_1 + 3x_2))/(3.5 + \sin(x_1 - x_2))) + \Psi$	$x_i \in [-2, 2]$ $i \in \{1,2\}$
Function21	$f(x_1, x_2) = \exp(x_1 \sin(\pi x_2)) + \Psi$	$x_1, x_2 \in [-1, 1]$
Function22	$f(x) = 0.02[(12 + 3x - 3.5x^2 + 7.2x^3)(1 + \cos 4\pi x)(1 + 0.8\sin 3\pi x)] + \Psi$	$x \in [-0.25, 0.2\ 5]$
Function23	$f(x_1, x_2, x_3, x_4, x_5) = 10\sin\pi x_1 x_2 + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \Psi$	$x_i \in [0, 1]$ $i = 1, 2, 3, 4, 5$
Function24	$f(x) = 0.2\sin(2\pi x) + 0.2x^2 + 0.3$ such that $y_i = f(x_i) + (0.1x_i^2 + 0.05)\Psi_i$	$x_i = 0.01(i - 1) - 1, i = 1, 2, \ldots, 200$

4.3 Real World Datasets

Here, 42 standard benchmark real-world datasets at different significant noise levels such as 0%, 5% and 10% are used to demonstrate the performance of RHN-TSVR. The details of datasets follow as:

Table 6 Performance comparison of RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR using Gaussian kernel on synthetic data sets with Laplacian noise

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Function19 (150X2,500X2)	0.416258 (10^2,0.1,2^{-1}) 0.55812	0.404259 (10^0,0.9,2^{-1}) 0.03316	0.40645 (10^3,10^{-1}, 1, 1, 2^{-1}) 0.00513	0.47416 (10^3,0.7,0.5,2^{-1}) 0.02272	**0.404141** (10^1, 0.9, 0.1, 2^{-1}) 0.02615	**0.404141** (10^1,10^{-3}, 0.9, 0.1, 2^{-1}) 0.0083096
Function20 (150X3,500X3)	0.146758 (10^0,0.1,2^0) 0.47995	0.217388 (10^0,0.9,2^{-1}) 0.02454	0.252318 (10^{-2},10^{-1}, 0.1, 1, 2^4) 0.01452	0.21538 (10^1,0.5,0.5,2^1) 0.02619	0.217806 (10^1, 0.9, 0.1, 2^{-1}) 0.02312	**0.138944** (10^1,10^2, 0.5, 0.1, 2^3) 0.0125658
Function21 (150X3,500X3)	0.193024 (10^5,0.1,2^{-5}) 0.46148	0.215517 (10^0,0.9,2^{-3}) 0.02405	0.213251 (10^{-1},10^{-1}, 1, 1.345, 2^0) 0.01195	0.24943 (10^{-1},0.1,0.8,2^5) 0.03729	0.215388 (10^1, 0.9, 0.1, 2^{-3}) 0.02207	**0.188105** (10^1,10^1, 0.9, 0.1, 2^0) 0.0189023
Function22 (150X2,500X2)	0.107321 (10^0,0.1,2^4) 0.43419	0.116748 (10^0,0.9,2^2) 0.03422	**0.081983** (10^0,10^{-1}, 0.1, 0.1, 2^5) 0.00426	0.11815 (10^5,0.3,0.4,2^{-1}) 0.0151	0.118841 (10^1, 0.1, 0.1, 2^3) 0.02572	0.104688 (10^1,10^1, 0.1, 0.1, 2^5) 0.0133986
Function23 (150X6,500X6)	0.190689 (10^2,0.1,2^1) 0.38126	0.240025 (10^0,0.1,2^1) 0.02595	0.24585 (10^0,10^{-1}, 1, 1, 2^2) 0.01102	0.28425 (10^1,0.3,0.3,2^3) 0.0199	0.244521 (10^0, 0.1, 0.7, 2^1) 0.02311	**0.18934** (10^1,10^{-1}, 0.9, 0.1, 2^1) 0.0136633
Function24 (200X2,500X2)	0.026582 (10^2,0.1,2^1) 0.89983	0.021008 (10^0,0.9,2^1) 0.06646	0.027998 (10^4,10^{-1}, 0.1, 0.1, 2^2) 0.00481	**0.01086** (10^3,0.1,0.4,2^1) 0.02411	0.019294 (10^1, 0.1, 0.1, 2^1) 0.08291	0.012581 (10^0,10^{-4}, 0.9, 0.3, 2^1) 0.0169828

The best result is shown as boldface

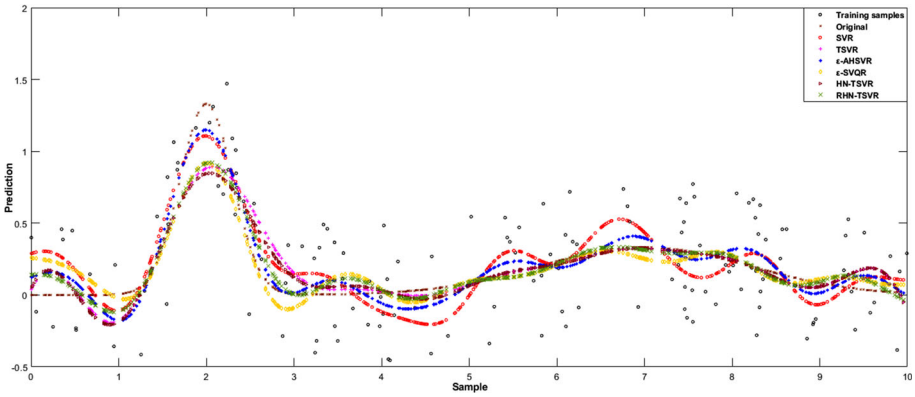

Fig. 1 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Function 13 artificial generated dataset. Gaussian kernel was used

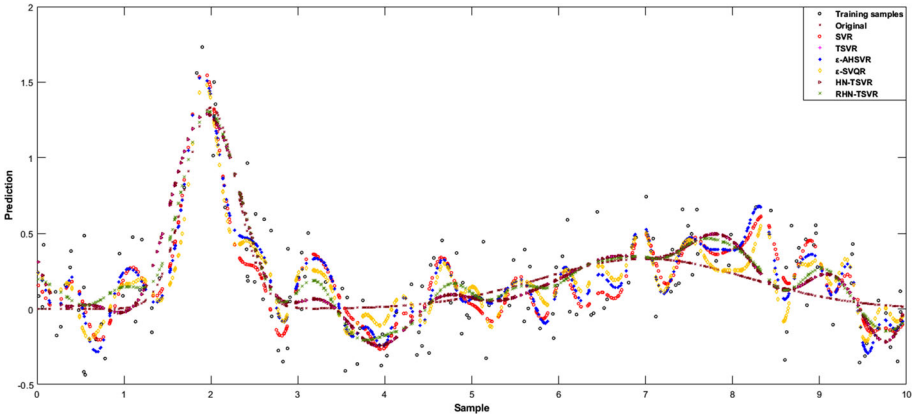

Fig. 2 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Function 14 artificial generated dataset. Gaussian kernel was used

Real world datasets	Repositories
Forestfires, Machine_CPU, Auto-original, Winequality, Gas_furnace, Quake	(UCI datasets repositories, [63])
SantafeA	[53]
The inverse dynamics of a Flexible robot arm	[21]
The financial time series datasets: S&P500, INFY, ONGC_NS, XOM, ATX, BSESN, DJI, GDAXI, MXX, N225	[20, 27]
Space Ga	[58]

Real world datasets	Repositories
KEEL time-series datasets: NNGC1_dataset_E1_V1_001, NNGC1_dataset_F1_V1_008, NNGC1_dataset_F1_V1_009, NNGC1_dataset_F1_V1_010, NNGC1_dataset_F1_V1_006, NN5_Complete_109, NN5_Complete_104, NN5_Complete_106, NN5_Complete_103, NN5_Complete_101, NN5_Complete_105, NN5_Complete_111, D1dat_1_2000	[38]
Wankara, Laser, Dee, Friedman, Mortgage	[38]
Roszman1, Gauss1, Chwirut2	[48]
Vineyard	[65]
COVID-19_spain	[14]

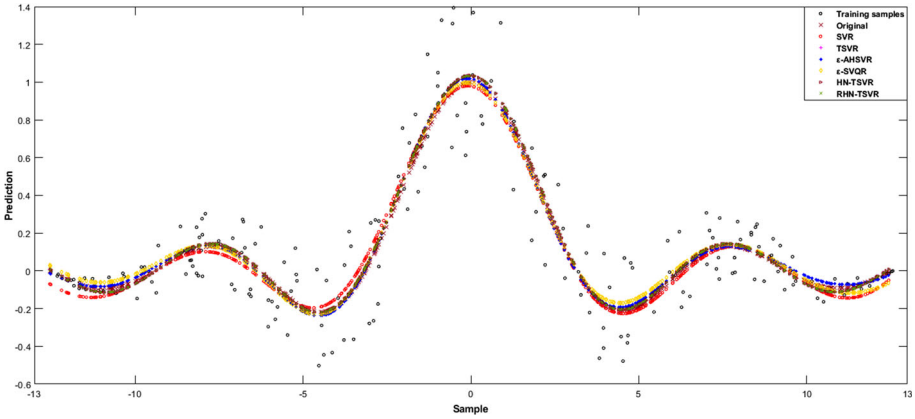

Fig. 3 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Function 15 artificial generated dataset. Gaussian kernel was used

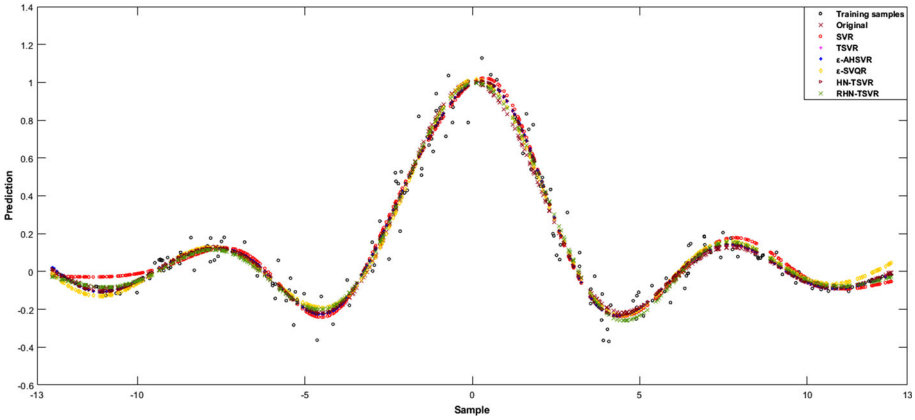

Fig. 4 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Function 16 artificial generated dataset. Gaussian kernel was used

Table 7 Average ranks of proposed RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR on RMSE using Gaussian kernel on synthetic data sets with Laplacian noise

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Function19	5	3	4	6	1.5	1.5
Function20	2	4	6	3	5	1
Function21	2	5	3	6	4	1
Function22	3	4	1	5	6	2
Function23	2	3	5	6	4	1
Function24	5	4	6	1	3	2
Average rank	3.1666667	3.8333333	4.1666667	4.5	3.9166667	**1.4166667**

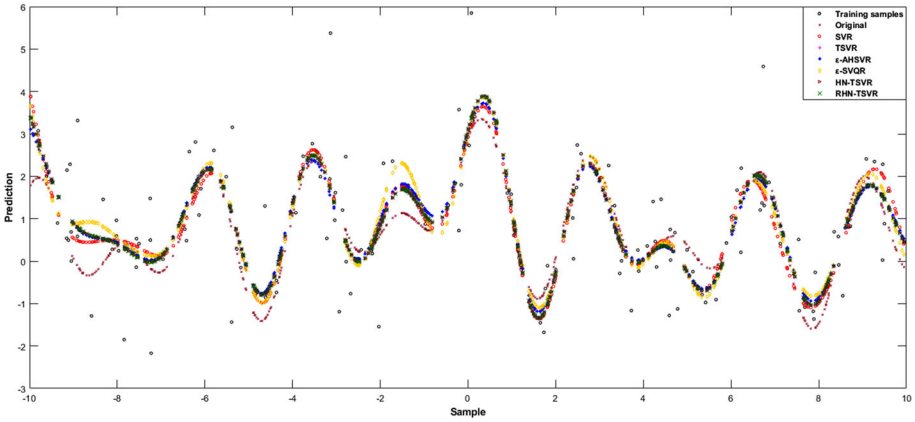

Fig. 5 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Function 19 artificial generated dataset. Gaussian kernel was used

4.3.1 At Significant Noise Level 0%

The numerical experiment results for SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and proposed approach RHN-TSVR on 42 standard real-world benchmark datasets at significant noise level 0% are obtained and tabulated in Table 8. It consists of the prediction accuracy with optimal parameters value and the learning time for all reported approaches. The performance of our proposed RHN-TSVR over benchmark real-world datasets is better in 22 cases compared to SVR, TSVR, ε-AHSVR, ε-SVQR, and HN-TSVR. Further, the average ranks are determined to perform the statistical test in Table 9 based on RMSE values for the Gaussian kernel. We can see that the average rank of our proposed RHN-TSVR is least in compared to other reported approaches. One can reach to conclusion that RHN-TSVR performs better on real world datasets at significant noise level 0%. Figures 6 and 7 are plotted corresponding to Machine CPU and Gas furnace datasets respectively in order to show the prediction performance graphically. Both graphs are clearly showing a better prediction capability of RHN-TSVR.

Now, a statistical analysis is implemented using a Friedman test (Demsar [17]) by using the average ranks of algorithms as mentioned in Table 9. Here, we decide the null hypothesis

Table 8 Performance comparison of RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR using Gaussian kernel with noise 0% on real world datasets

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
Forestfires (150X13,367X13)	0.07059 $(10^{-3}, 0.1, 2^5)$ 0.55583	0.07058 $(10^{-4}, 0.1, 2^5)$ 0.063014	0.070592 $(10^{-5}, 10^{-3}, 0.1, 0.1, 2^1)$ 0.09428	**0.070476** $(10^1, 0.1, 0.1, 2^{-5})$ 0.02012	0.07058 $(10^{-5}, 0.5, 0.5, 2^5)$ 0.25727	0.07059 $(10^5, 10^5, 0.3, 0.3, 2^{-5})$ 0.0125241
Machine CPU (100X8,109X8)	0.03506 $(10^4, 0.1, 2^{-2})$ 0.24525	0.02102 $(10^3, 0.9, 2^{-2})$ 0.041461	0.050225 $(10^5, 10^{-1}, 0.1, 0.1, 2^{-3})$ 0.332	0.036146 $(10^1, 0.1, 0.7, 2^{-5})$ 0.00977	0.03035 $(10^3, 0.9, 0.5, 2^{-1})$ 0.20161	**0.01579** $(10^3, 10^{-5}, 0.9, 0.3, 2^{-3})$ 0.0107687
Auto-original (100X8,292X8)	0.33338 $(10^0, 0.1, 2^1)$ 0.2708	0.29912 $(10^{-2}, 0.1, 2^0)$ 0.033108	**0.171313** $(10^1, 10^{-1}, 0.1, 1, 2^{-5})$ 0.06359	0.260394 $(10^1, 0.1, 0.7, 2^{-1})$ 0.01014	0.29899 $(10^{-1}, 0.1, 0.1, 2^0)$ 0.04945	0.29899 $(10^{-1}, 10^{-3}, 0.1, 0.1, 2^0)$ 0.0126264
Winequality (500X12,4398X12)	0.13377 $(10^2, 0.1, 2^{-3})$ 6.31838	0.13437 $(10^{-5}, 0.7, 2^{-1})$ 0.719632	**0.130454** $(10^5, 10^{-1}, 0.1, 1, 2^{-3})$ 0.13756	0.132434 $(10^3, 0.1, 0.5, 2^{-5})$ 0.47543	0.135 $(10^0, 0.1, 0.1, 2^{-1})$ 0.61889	0.13486 $(10^0, 10^{-5}, 0.1, 0.1, 2^{-3})$ 0.0832587
SantafeA (500X6,495X6)	0.05654 $(10^0, 0.1, 2^2)$ 5.59668	0.03811 $(10^{-1}, 0.9, 2^3)$ 0.758993	0.081809 $(10^3, 10^{-2}, 0.1, 1, 2^1)$ 0.1015	0.039983 $(10^3, 0.1, 0.5, 2^{-1})$ 0.86175	0.03922 $(10^0, 0.9, 0.1, 2^3)$ 0.64863	**0.0375** $(10^5, 10^{-5}, 0.1, 0.7, 2^3)$ 0.113057

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Gas_furnace (150X7,143X7)	0.07154 (10^5, 0.1,2^{-5}) 0.47456	0.04394 (10^2,0.1,2^{-4}) 0.068103	0.084165 (10^1,10^{-1}, 0.1, 1, 2^{-3}) 0.05107	0.05659 (10^3,0.1, 0.2, 2^{-3}) 0.0444	0.04158 (10^1, 0.1, 0.1, 2^{-3}) 0.08725	**0.0383** (10^0,10^{-5}, 0.1, 0.3, 2^{-5}) 0.0174328
Quake (500X4,1678X4)	0.17349 (10^{-1}, 0.1,2^{-2}) 9.48833	0.17227 (10^0,0.9,2^{-5}) 0.758743	**0.17214** (10^{-1},10^{-1}, 0.1, 0.1, 2^{-3}) 0.14973	0.172642 (10^{-3},0.1, 0.7, 2^{-3}) 0.36931	0.17219 (10^0, 0.9, 0.3, 2^{-5}) 0.82559	0.17224 (10^0,10^1, 0.9, 0.3, 2^{-5}) 0.0794874
Flex_robotarm (500X10,519X10)	0.04337 (10^3, 0.1,2^{-5}) 9.5114	0.03595 (10^1,0.1,2^{-1}) 0.795891	0.056807 (10^5,10^{-1}, 0.1, 0.1, 2^{-5}) 0.0878	0.025886 (10^5,0.1, 0.4, 2^{-5}) 0.24667	0.03595 (10^3, 0.1, 0.9, 2^{-1}) 1.20655	**0.02046** (10^3,10^{-5}, 0.9, 0.7, 2^{-1}) 0.10096
S&P500 (200X6,550X6)	0.19605 (10^5, 0.1,2^{-1}) 1.41245	0.13739 (10^{-1},0.9,2^1) 0.113956	0.255194 (10^{-1},10^{-1}, 0.1, 1, 2^{-1}) 0.02659	**0.095518** (10^3,0.1, 0.8, 2^{-3}) 0.04174	0.13983 (10^1, 0.9, 0.1, 2^1) 0.33202	0.13355 (10^0,10^{-5}, 0.1, 0.3, 2^{-1}) 0.0243076
Space-Ga (500X7,2607X7)	0.30306 (10^{-1}, 0.1,2^{-1}) 9.72392	0.29628 (10^{-3},0.1,2^5) 0.697419	0.30902 (10^1,10^{-1}, 0.1, 1, 2^5) 0.09576	**0.285357** (10^{-1},0.1, 0.6, 2^{-1}) 0.39695	0.29628 (10^{-5}, 0.1, 0.1, 2^5) 0.92673	0.29306 (10^0,10^{-1}, 0.9, 0.1, 2^5) 0.107667
Gauss1 (75X2,175X2)	0.738274 (10^5,0.1,2^{-5}) 0.12142	0.479003 (10^3, 0.9, 2^3) 0.01401	0.738025 (10^3,10^{-1}, 0.1, 0.1, 2^3) 0.00066	0.645752 (10^5,0.1, 0.8, 2^{-5}) 0.00861	**0.478936** (10^5, 0.9, 0.5, 2^3) 0.02833	0.619614 (10^1,10^1, 0.9, 0.9, 2^3) 0.0061953

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
Chwirut2 (17X2,37X2)	0.120144 $(10^0,0.1,2^5)$ 0.00941	**0.078331** $(10^{-5}, 0.5, 2^5)$ 0.01558	0.131455 $(10^1,10^{-1}, 0.1, 1, 2^5)$ 0.00014	0.10763 $(10^1,0.1, 0.3, 2^5)$ 0.00408	0.078332 $(10^{-5}, 0.3, 0.7, 2^5)$ 0.02219	0.101378 $(10^3,10^{-5}, 0.9, 0.7, 2^3)$ 0.0053057
Roszman1 (8X2,17X2)	0.833004 $(10^2,0.1,2^5)$ 0.00441	0.147794 $(10^{-5}, 1, 2^0)$ 0.01062	0.704285 $(10^2,10^{-1}, 0.1, 1, 2^5)$ 0.00009	0.774556 $(10^1,0.1, 0.3, 2^5)$ 0.00621	**0.147779** $(10^{-5}, 1, 0.7, 2^0)$ 0.02386	0.955187 $(10^3,10^{-5}, 1, 0.7, 2^0)$ 0.0039767
INFY (226X6,525X6)	0.061211 $(10^3,0.1,2^{-5})$ 0.98812	0.040511 $(10^{-1}, 0.9, 2^{-3})$ 0.0482	0.047693 $(10^5,10^{-1}, 0.1, 1, 2^{-5})$ 0.02781	0.041628 $(10^3,0.1, 0.5, 2^{-5})$ 0.05389	0.040633 $(10^0, 0.9, 0.1, 2^{-3})$ 0.06636	**0.040282** $(10^{-1},10^{-5}, 0.9, 0.1,$ $2^{-5})$ 0.0205775
ONGC_NS (221X6,514X6)	0.029434 $(10^1,0.1,2^{-5})$ 0.91666	0.028493 $(10^0, 0.1, 2^{-3})$ 0.05983	0.044317 $(10^3,10^{-1}, 0.1, 1, 2^{-5})$ 0.02583	0.02502 $(10^3,0.1, 0.5, 2^{-5})$ 0.05644	0.029634 $(10^1, 0.9, 0.1, 2^{-3})$ 0.06879	**0.023333** $(10^1,10^{-5}, 0.1, 0.1,$ $2^{-5})$ 0.0232562
XOM (226X6,525X6)	0.03609 $(10^1,0.1,2^{-3})$ 0.97152	0.033274 $(10^{-1}, 0.1, 2^{-3})$ 0.04656	0.05045 $(10^5,10^{-1}, 0.1, 1, 2^{-3})$ 0.034	0.033264 $(10^1,0.1, 0.5, 2^{-3})$ 0.05051	0.033344 $(10^{-1}, 0.1, 0.7, 2^{-3})$ 0.05523	**0.032806** $(10^{-3},10^{-5}, 0.1, 0.1,$ $2^{-5})$ 0.0252035
ATX (222X6,515X6)	0.413258 $(10^{-1},0.1,2^0)$ 0.93081	0.155567 $(10^{-1}, 0.1, 2^{-1})$ 0.04733	0.475983 $(10^{-1},10^{-1}, 0.1, 1, 2^1)$ 0.03145	0.129202 $(10^3,0.1, 0.2, 2^{-3})$ 0.05126	0.161982 $(10^0, 0.1, 0.3, 2^{-1})$ 0.09139	**0.067275** $(10^{-1},10^{-5}, 0.9, 0.7,$ $2^{-3})$ 0.0260819
BSESN (220X6,513X6)	0.197459 $(10^0,0.1,2^{-1})$ 0.92109	0.080313 $(10^{-1}, 0.1, 2^{-1})$ 0.05851	0.117697 $(10^1,10^{-1}, 0.1, 1, 2^{-3})$ 0.0298	**0.034469** $(10^3,0.1, 0.7, 2^{-5})$ 0.04303	0.08503 $(10^0, 0.1, 0.3, 2^{-1})$ 0.0663	0.04276 $(10^{-1},10^{-5}, 0.1, 0.1,$ $2^{-3})$ 0.0254383

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
DJI (226X6,525X6)	0.437188 (10^{-1},0.1,2^1) 0.98217	0.252219 (10^{-1}, 0.1, 2^{-1}) 0.05257	0.445899 (10^{-1},10^{-1}, 0.1, 0.1, 2^1) 0.02312	0.134225 (10^1,0.1, 0.8, 2^{-5}) 0.04319	0.304002 (10^{-1}, 0.1, 0.1, 2^0) 0.06687	**0.106785** (10^0,10^{-5}, 0.1, 0.1, 2^{-3}) 0.0278918
GDAXI (227X6,528X6)	0.033826 (10^5,0.1,2^{-5}) 1.00256	0.032456 (10^0, 0.9, 2^{-3}) 0.06494	0.059894 (10^3,10-1, 0.1, 1, 2^{-5}) 0.03418	0.050758 (10^1,0.1, 0.6, 2^{-1}) 0.05458	0.036447 (10^1, 0.9, 0.3, 2^{-3}) 0.07007	**0.028567** (10^{-1},10^{-5}, 0.9, 0.1, 2^{-5}) 0.0259832
MXX (225X6,525X6)	0.096186 (10^5,0.1,2^{-5}) 0.98059	0.207771 (10^{-3}, 0.9, 2^{-1}) 0.0454	0.15942 (10^5,10^{-1}, 0.1, 1, 2^{-3}) 0.02529	0.170982 (10^3,0.1, 0.5, 2^{-3}) 0.05175	0.208147 (10^{-1}, 0.1, 0.1, 2^{-1}) 0.07338	**0.070677** (10^{-1},10^{-5}, 0.1, 0.1, 2^{-3}) 0.0260515
N225 (220X6,512X6)	0.100205 (10^5,0.1,2^{-3}) 0.94593	0.0391 (10^{-1}, 0.9, 2^{-3}) 0.04503	0.215338 (10^{-1},10^{-1}, 0.1, 1, 2^1) 0.01587	**0.032827** (10^3,0.1, 0.7, 2^{-5}) 0.04882	0.040257 (10^0, 0.1, 0.1, 2^{-3}) 0.07263	0.032986 (10^{-1},10^{-5}, 0.1, 0.1, 2^{-5}) 0.0236554
Wankara (97X10,224X10)	0.056582 (10^3,0.1,2^0) 0.18213	0.025151 (10^{-1}, 0.9, 2^{-3}) 0.02217	0.059292 (10^3,10^{-1}, 0.1, 0.1, 2^{-5}) 0.00379	0.031492 (10^3,0.1, 0.2, 2^{-3}) 0.01671	0.025243 (10^1, 0.9, 0.7, 2^{-3}) 0.03012	**0.024392** (10^1,10^{-5}, 0.1, 0.1, 2^{-3}) 0.014395
Laser (298X5,695X5)	0.057145 (10^3,0.1,2^{-3}) 1.72804	**0.036527** (10^{-1}, 0.9, 2^3) 0.08	0.076187 (10^5,10^{-2}, 0.1, 0.1, 2^{-3}) 0.04045	0.039881 (10^1,0.1, 0.4, 2^1) 0.18004	0.037124 (10^3, 0.9, 0.1, 2^3) 0.14206	0.037946 (10^3,10^{-5}, 0.1, 0.9, 2^3) 0.0432735
Dee (110X7,255X7)	0.104984 (10^0,0.1,2^{-3}) 0.22914	0.142128 (10^0, 0.1, 2^{-5}) 0.02127	0.129019 (10^1,10^{-1}, 1, 0.1, 2^{-3}) 0.00309	**0.101998** (10^1,0.1, 0.7, 2^{-5}) 0.02589	0.145534 (10^1, 0.1, 0.1, 2^{-5}) 0.03195	0.102515 (10^1,10^{-1}, 0.9, 0.1, 2^{-3}) 0.0154981

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Friedman (360X6,840X6)	0.058623 (10^3,0.1,2^{-1}) 2.57757	0.044491 (10^{-5}, 0.1, 2^0) 0.12724	0.067515 (10^3,10^{-1}, 0.1, 1, 2^{-1}) 0.08704	0.045422 (10^5,0.1, 0.5, 2^{-5}) 0.24806	0.044433 (10^0, 0.1, 0.5, 2^0) 0.18492	**0.043238** (10^{-5},10^{-5}, 0.1, 0.3, 2^{-1}) 0.0423292
Mortgage (315X16,734X16)	0.047516 (10^3,0.1,2^{-1}) 1.97172	**0.007537** (10^3, 0.9, 2^0) 0.11881	0.061293 (10^1,10^{-1}, 0.1, 0.1, 2-5) 0.05237	0.024022 (10^1,0.1, 0.8, 2^{-3}) 0.2937	0.007581 (10^1, 0.9, 0.1, 2^0) 0.21581	0.00775 (10^{-1},10^{-5}, 0.9, 0.3, 2^1) 0.0464666
NNGC1_dataset_E1_V1_001 (111X5,259X5)	0.182661 (10^1,0.1,2^1) 0.2343	0.180173 (10^{-1}, 0.9, 2^1) 0.01953	**0.172927** (10^1,10^{-1}, 0.1, 1, 2^3) 0.00626	0.181204 (10^1,0.1, 0.6, 2^1) 0.01454	0.180518 (10^0, 0.1, 0.1, 2^1) 0.02345	0.17386 (10^0,10^0, 0.9, 0.1, 2^3) 0.0142513
NNGC1_dataset_F1_V1_008 (269X5,626X5)	0.123395 (10^1,0.1,2^5) 1.41931	0.073928 (10^3, 0.1, 2^5) 0.13796	0.114369 (10^1,10^{-1}, 0.1, 1, 2^5) 0.02353	0.104162 (10^1,0.1, 0.5, 2^5) 0.07411	**0.073928** (10^5, 0.1, 0.5, 2^5) 0.10784	**0.073928** (10^5,10^{-3}, 0.1, 0.5, 2^5) 0.0428835
NNGC1_dataset_F1_V1_009 (269X5,626X5)	0.092724 (10^5,0.1,2^1) 1.44621	0.06457 (10^{-3}, 0.9, 2^5) 0.07942	0.092574 (10^5,10^{-1}, 0.1, 1, 2^1) 0.02105	0.076728 (10^1,0.1, 0.7, 2^3) 0.0875	**0.064281** (10^{-1}, 0.9, 0.1, 2^5) 0.08413	**0.064281** (10^{-1},10^{-3}, 0.9, 0.1, 2^5) 0.0320651
NNGC1_dataset_F1_V1_010 (269X5,626X5)	0.086069 (10^5,0.1,2^{-3}) 1.41684	**0.044163** (10^{-3}, 0.9, 2^5) 0.10058	0.084937 (10^5,10^{-1}, 0.1, 1, 2^1) 0.01928	0.06221 (10^1,0.1, 0.7, 2^3) 0.07695	0.04426 (10^0, 0.1, 0.1, 2^5) 0.0761	0.04426 (10^0,10^{-3}, 0.1, 0.1, 2^5) 0.0402401
NNGC1_dataset_F1_V1_006 (521X5,1214X5)	0.060847 (10^5,0.1,2^{-1}) 5.75965	0.037313 (10^{-1}, 0.9, 2^5) 0.31331	0.065632 (10^5,10^{-1}, 0.1, 1, 2^1) 0.08754	0.045866 (10^1,0.1, 0.6, 2^3) 0.3148	**0.037242** (10^0, 0.9, 0.1, 2^5) 0.28168	**0.037242** (10^0,10^{-3}, 0.9, 0.1, 2^5) 0.10657

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
NN5_Complete_109 (237X5,550X5)	0.109583 (10^0,0.1,2^3) 1.09254	0.107237 (10^{-1}, 0.9, 2^3) 0.06317	0.110924 (10^1,10^{-1}, 0.1, 0.1, 2^3) 0.01518	0.102385 (10^1,0.1, 0.6, 2^3) 0.05181	0.106897 (10^0, 0.1, 0.1, 2^3) 0.07392	**0.095636** (10^0,10^0, 0.3, 0.1, 2^5) 0.026553
NN5_Complete_104 (237X5,550X5)	0.174167 (10^{-1},0.1,2^3) 1.07709	0.144628 (10^{-3}, 0.1, 2^3) 0.05759	0.153973 (10^1,10^{-1}, 0.1, 1, 2^5) 0.01743	0.154479 (10^1,0.1, 0.8, 2^3) 0.05446	0.143688 (10^0, 0.9, 0.1, 2^3) 0.06773	**0.142988** (10^0,10^0, 0.1, 0.1, 2^5) 0.030209
NN5_Complete_106 (237X5,550X5)	0.189308 (10^0,0.1,2^3) 1.10129	0.174154 (10^{-1}, 0.1, 2^3) 0.06459	0.165296 (10^1,10^{-1}, 0.1, 1, 2^5) 0.01407	0.187904 (10^1,0.1, 0.6, 2^3) 0.05242	0.171795 (10^{-1}, 0.1, 0.1, 2^3) 0.06597	**0.15994** (10^0,10^0, 0.9, 0.1, 2^5) 0.0335387
NN5_Complete_103 (237X5,550X5)	0.128696 (10^0,0.1,2^{-1}) 1.08169	0.130192 (10^0, 0.1, 2^{-1}) 0.06974	0.131486 (10^1,10^{-1}, 0.1, 0.1, 2^1) 0.02025	**0.128318** (10^1,0.1, 0.4, 2^{-3}) 0.06604	0.129899 (10^{-5}, 0.1, 0.3, 2^{-1}) 0.06392	0.129068 (10^0,10^{-1}, 0.1, 0.3, 2^0) 0.027654
NN5_Complete_101 (237X5,550X5)	0.149576 (10^0,0.1,2^1) 1.08068	0.154734 (10^0, 0.9, 2^0) 0.06989	0.150469 (10^1,10^{-1}, 0.1, 0.1, 2^3) 0.01455	0.14708 (10^1,0.1, 0.6, 2^1) 0.05376	0.155191 (10^1, 0.1, 0.1, 2^0) 0.06886	**0.144747** (10^0,10^0, 0.9, 0.1, 2^3) 0.0264708
NN5_Complete_105 (237X5,550X5)	0.127845 (10^0,0.1,2^1) 1.08763	0.13341 (10^{-1}, 0.9, 2^1) 0.04819	0.130747 (10^1,10^{-1}, 1, 0.1, 2^3) 0.0148	0.129324 (10^1,0.1, 0.4, 2^1) 0.0635	0.134297 (10^0, 0.1, 0.1, 2^1) 0.09553	**0.126683** (10^0,10^0, 0.9, 0.1, 2^5) 0.031413
NN5_Complete_111 (237X5,550X5)	0.089431 (10^0,0.1,2^3) 1.09501	0.092115 (10^{-5}, 0.9, 2^3) 0.07277	0.087204 (10^1,10^{-1}, 0.1, 0.1,$^{25)}$ 0.0199	0.094835 (10^1,0.1, 0.2, 2^3) 0.0468	0.092115 (10^{-5}, 0.1, 0.1, 2^3) 0.06021	**0.08508** (10^{-1},100, 0.1, 0.1, 2^5) 0.0297235

Table 8 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
D1dat_1_2000 (599X5,1396X5)	**0.091946** $(10^3,0.1,2^{-1})$ 7.51017	0.135055 $(10^{-1}, 0.9, 2^3)$ 0.51861	0.10349 $(10^3,10^{-1}, 0.1, 1, 2^{-3})$ 0.10285	0.115531 $(10^3,0.1, 0.4, 2^1)$ 0.39425	0.145437 $(10^0, 0.1, 0.9, 2^3)$ 0.32529	0.121417 $(10^{-1},10^{-5}, 0.1, 0.1,$ $2^1)$ 0.117241
Vineyard (16X4,36X4)	**0.141193** $(10^0,0.1,2^0)$ 0.00798	0.298839 $(10^0, 0.9, 2^{-1})$ 0.01456	0.234064 $(10^1,10^{-1}, 0.1, 0.1, 2^1)$ 0.00015	0.188534 $(10^1,0.1, 0.1, 2^1)$ 0.00692	0.241009 $(10^1, 0.9, 0.3, 2^{-3})$ 0.01197	0.240423 $(10^0,101, 0.9, 0.5, 2^1)$ 0.0103753
COVID-19_spain (251X5,585X5)	0.167119 $(10^3,0.1,2^1)$ 1.21284	**0.154007** $(10^{-3}, 0.5, 2^5)$ 0.07972	0.167119 $(10^{-5},10^{-3}, 0.1, 0.1,$ $2^{-5})$ 0.01419	0.156555 $(10^5,0.1, 0.7, 2^5)$ 0.04876	0.154173 $(10^{-1}, 0.1, 0.9, 2^5)$ 0.04778	0.154061 $(10^{-5},10\text{-}5, 0.9, 0.1,$ $2^5)$ 0.024674

The best result is shown as boldface

Table 9 Average ranks of SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on RMSE values using Gaussian kernel with noise 0% for real world dataset

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Forestfires	5.5	2.5	5.5	1	2.5	4
Machine CPU	4	2	6	5	3	1
Auto-original	6	5	1	2	3.5	3.5
Winequality	3	4	1	2	6	5
SantafeA	5	2	6	4	3	1
Gas_furnace	5	3	6	4	2	1
Quake	6	4	1	5	2	3
Flex_robotarm	5	3.5	6	2	3.5	1
S&P500	5	3	6	1	4	2
Space-Ga	5	3.5	6	1	3.5	2
Gauss1	6	2	5	4	1	3
Chwirut2	5	1	6	4	2	3
Roszman1	5	2	3	4	1	6
INFY	6	2	5	4	3	1
ONGC_NS	4	3	6	2	5	1
XOM	5	3	6	2	4	1
ATX	5	3	6	2	4	1
BSESN	6	3	5	1	4	2
DJI	5	3	6	2	4	1
GDAXI	3	2	6	5	4	1
MXX	2	5	3	4	6	1
N225	5	3	6	1	4	2
Wankara	5	2	6	4	3	1

Table 9 continued

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Laser	5	1	6	4	2	3
Dee	3	5	4	1	6	2
Friedman	5	3	6	4	2	1
Mortgage	5	1	6	4	2	3
NNGC1_dataset_E1_V1_001	6	3	1	5	4	2
NNGC1_dataset_F1_V1_008	6	3	5	4	1.5	1.5
NNGC1_dataset_F1_V1_009	6	3	5	4	1.5	1.5
NNGC1_dataset_F1_V1_010	6	1	5	4	2.5	2.5
NNGC1_dataset_F1_V1_006	5	3	6	4	1.5	1.5
NN5_Complete_109	5	4	6	2	3	1
NN5_Complete_104	6	3	4	5	2	1
NN5_Complete_106	6	4	2	5	3	1
NN5_Complete_103	2	5	6	1	4	3
NN5_Complete_101	3	5	4	2	6	1
NN5_Complete_105	2	5	4	3	6	1
NN5_Complete_111	3	4	2	6	5	1
D1dat_1_2000	1	5	2	3	6	4
Vineyard	1	6	3	2	5	4
COVID-19_spain	5.5	1	5.5	4	3	2
Average rank	4.5952381	3.1309524	4.6666667	3.1666667	3.4285714	**2.0119048**

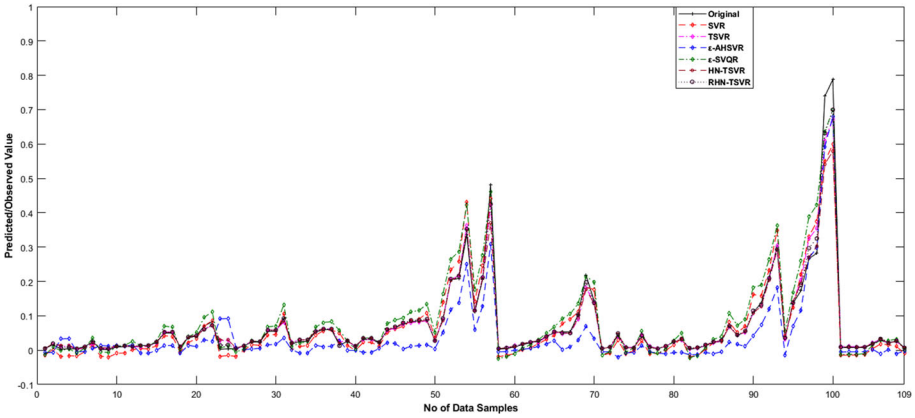

Fig. 6 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Machine CPU dataset with 0% noise. Gaussian kernel was used

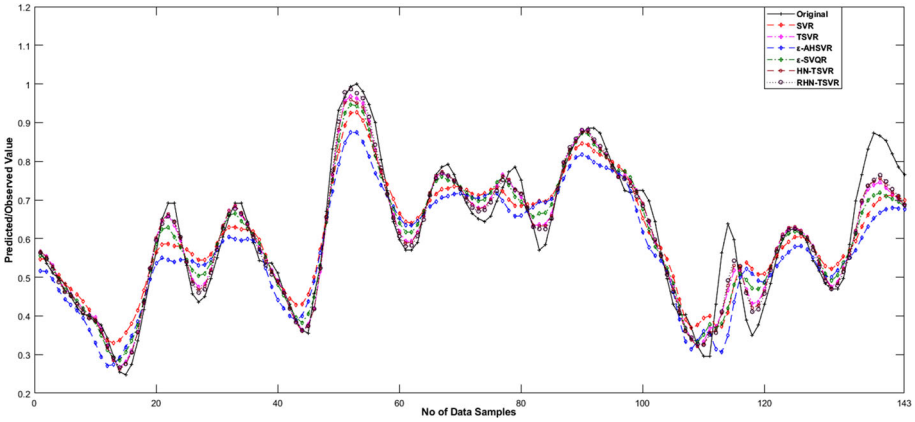

Fig. 7 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Gas furnace dataset with 0% noise. Gaussian kernel was used

H_0: All the reported algorithms such as SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR are equivalent.

In this context, now compute both χ_F^2 and F_F based on Table 9 as shown below:

$$\chi_F^2 = \frac{12 \times 42}{6 \times 7}\Big[\big(4.595238^2 + 3.130952^2 + 4.666667^2 + 3.166667^2 + 3.428571^2 + 2.011905^2\big) - \left(\frac{6 \times 7^2}{4}\right)\Big] = 60.3299,$$

and

$$F_F = \frac{(42-1) \times 60.3299}{(42 \times (6-1)) - 60.3299} = 16.5265,$$

Here, the critical value, corresponding to (5, 205) the degree of freedom at the probability level $\Phi = 0.05$, is smaller than F_F ($16.5265 > 2.2581$). So, the null hypothesis H_0 is

rejected and further performed pairwise Nemenyi test. Now, calculate the critical difference at a significant level ($p = 0.10$) as

Critical difference $= q_\alpha \sqrt{\frac{k\times(k+1)}{6\times N}} = 2.589\sqrt{\frac{6\times(6+1)}{6\times 42}} = 1.057$,

where, k and N are the no of reported algorithms and datasets respectively and $q_\alpha = 2.589$ (Demsar [17]).

One can conclude some points from this Nemenyi test that are discussed below:

1. The average rank of proposed algorithm RHN-TSVR is compared with SVR and TSVR i.e. (4.595238−2.011905= 2.58333) and (3.130952−2.011905= 1.119048) respectively. The differences are greater than critical difference 1.057 which shows, RHN-TSVR is superior to SVR and TSVR.
2. Now calculate the average rank differences of RHN-TSVR to ε-AHSVR and ε-SVQR i.e. (4.666667−2.011905= 2.654762) and (3.166667−2.011905= 1.154762) respectively which are also greater than 1.057. Hence, it justifies the effectiveness of our proposed RHN-TSVR.
3. Check the dissimilarity of average ranks between RHN-TSVR and HN-TSVR, i.e. (3.428571−2.011905= 1.416667). It is also higher than the critical difference (1.416667 > 1.057) that declares, our proposed approach RHN-TSVR outperforms to HN-TSVR.

4.3.2 At Significant Noise Level 5%

In this section, we have verified the applicability of RHN-TSVR model on real-world datasets at significant noise level 5%. All the results are computed and placed in Table 10 corresponding to 42 benchmark datasets. In Table 10, one can easily find the RMSE values with the computational time of all reported approaches. One can notice that RHN-TSVR model has shown better performance for 18 cases. The average ranks of each model are determined in Table 11 based on RMSE values to perform the statistical test. Here, RHN-TSVR has shown the lowest average rank in Table 11. Overall, we can analyze that RHN-TSVR may be one of the better choice for noisy datasets. We have plotted the prediction value graph for Machine CPU and Gas furnace datasets having significant noise level 5%, as shown in Figs. 8 and 9 respectively. The interpretation of these graphs is crystal clear and easily understandable that RHN-TSVR shows a closer relationship with the desired output.

Further, the Friedman statistical test is performed to validate the efficacy of proposed approach RHN-TSVR on noisy data at significant level 5% with SVR, TSVR, ε-AHSVR, ε-SVQR and HN-TSVR by using the results of Table 11.

$$\chi_F^2 = \frac{12\times 42}{6\times 7}\Big[\big(4.33333^2 + 3.404762^2 + 4.547619^2 + 3.452381^2 + 3.380952^2 + 1.880952^2\big) - \left(\frac{6\times 7^2}{4}\right)\Big] = 52.2653,$$

And

$$F_F = \frac{(42-1)\times 52.2653}{(42\times(6-1)) - 52.2653} = 13.9336$$

Here, the critical value, corresponding to (5, 205) the degree of freedom is smaller than F_F (13.9336 > 2.2581). So the null hypothesis H_0 is rejected thus perform pairwise test. Now, calculate the critical difference at a significant level ($p = 0.10$) to perform the Nemenyi

Table 10 Performance comparison of SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR using Gaussian kernel with noise 5% on real world datasets

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Forestfires (150X13,367X13)	0.070592 (10^{-5}, 0.1,2^5) 0.229976	0.0842079 (10^{-4}, 0.9, 2^5) 0.09435	0.070588 (10^{-5},10^{-3}, 0.1, 0.1, 2^5) 0.0112	**0.069557** (10^{-1},0.9, 0.1, 2^5) 0.08134	0.08422 (10^3, 0.9, 0.1, 2^5) 0.22418	0.07057 (10^{-5},10^5, 0.9, 0.1, 2^5) 0.012158
Machine CPU (100X8,109X8)	0.052104 (10^5, 0.1,2^{-5}) 0.11723	0.0485319 (10^{-1}, 0.1, 2^{-2}) 0.04732	0.07941 (10^5,10^{-3}, 0.1, 0.1, 2^{-3}) 0.0042	0.072152 (10^5,0.1, 0.5, 2^{-5}) 0.04398	0.05934 (10^0,0.1, 0.1, 2^{-1}) 0.05237	**0.04173** (10^0,10^{-5}, 0.1, 0.1, 2^{-3}) 0.0118531
Auto-original (100X8,292X8)	**0.155361** (10^3, 0.1,2^{-5}) 0.100113	0.16358 (10^{-2}, 0.9, 2^{-1}) 0.04578	0.156916 (10^3,10^{-1}, 0.1, 1, 2^{-1}) 0.01078	0.157831 (10^1,0.1, 0.5, 2^{-3}) 0.04071	0.1639 (10^{-1}, 0.1, 0.5, 2^{-1}) 0.13337	0.15585 (10^0,10^{-5}, 0.1, 0.1, 2^{-3}) 0.0137573
Winequality (500X12,4398X12)	0.158162 (10^0, 0.1,2^{-1}) 2.7085	0.181833 (10^{-1}, 0.9, 2^{-2}) 0.92887	0.160284 (10^3,10^{-1}, 0.1, 0.1, 2^{-3}) 0.1069	0.225503 (10^3,0.1, 0.5, 2^{-5}) 1.41081	0.17557 (10^0, 0.9, 0.3, 2^{-3}) 0.991	**0.1538** (10^0,10^{-1}, 0.9, 0.1, 2^0) 0.0646298
SantafeA (500X6,495X6)	0.10608 (10^0, 0.1,2^3) 3.07441	0.0665399 (10^{-1}, 0.9, 2^2) 0.84628	**0.061044** (10^5,10^{-2}, 0.1, 1, 2^1) 0.13854	0.082625 (10^1,0.1, 0.4, 2^3) 1.50956	0.06684 (10^{-1}, 0.9, 0.1, 2^3) 0.97786	0.06568 (10^0,10^{-5}, 0.9, 0.1, 2^1) 0.0511625

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Gas_furnace (150X7,143X7)	0.094555 (10^0, 0.1,2^{-3}) 0.228517	0.0603093 (10^{-1}, 0.7, 2^{-4}) 0.09493	0.061808 (10^5,10^{-3}, 0.1, 1, 2^{-5}) 0.01632	0.080781 (10^1,0.1, 0.1, 2^{-5}) 0.09148	0.05582 (10^0, 0.1, 0.1, 2^{-3}) 0.11075	**0.04891** (10^0,10^{-5}, 0.1, 0.1, 2^{-5}) 0.0181955
Quake (500X4,1678X4)	0.186957 (10^{-1}, 0.1,2^{-1}) 2.76314	0.185072 (10^0, 0.9, 2^{-5}) 0.82747	0.185139 (10^{-1},10^{-1}, 0.1, 0.1, 2^{-3}) 0.15029	0.19368 (10^{-1},0.5, 0.1, 2^{-5}) 1.63681	0.1851 (10^0, 0.9, 0.3, 2^{-5}) 0.91814	**0.18506** (10^0,10^1, 0.1, 0.3, 2^{-5}) 0.0648712
Flex_robotarm (500X10,519X10)	0.044356 (10^3, 0.1,2^{-5}) 2.91903	0.0455536 (10^{-1}, 0.9, 2^{-1}) 0.88575	0.056341 (10^5,10^{-1}, 0.1, 0.1, 2^{-5}) 0.11533	0.029939 (10^3,0.1, 0.1, 2^{-5}) 1.56187	0.04608 (10^0, 0.9, 0.1, 2^{-1}) 0.92991	**0.02582** (10^0,10^{-5}, 0.9, 0.1, 2^{-1}) 0.0688442
S&P500 (200X6,550X6)	0.039455 (10^1, 0.1,2^{-5}) 0.413842	**0.0264072** (10^{-1}, 0.9, 2^{-5}) 0.15642	0.045908 (10^5,10^{-1}, 0.1, 1, 2^{-5}) 0.02588	0.027269 (10^3,0.1, 0.4, 2^{-5}) 0.191	0.02683 (10^0, 0.1, 0.1, 2^{-5}) 0.17724	0.02663 (10^{-1},10^{-5}, 0.1, 0.1, 2^{-5}) 0.0197694
Space-Ga (500X7,2607X7)	0.29754 (10^0, 0.1,2^{-5}) 2.78144	0.327565 (10^0, 0.1, 2^5) 1.01007	0.292501 (10^1,10^{-1}, 1, 0.1, 2^{-5}) 0.12059	**0.273409** (10^{-1},0.1, 0.6, 2^{-1}) 1.32311	0.33016 (10^0, 0.1, 0.1, 2^5) 0.97394	0.28284 (10^0,10^{-1}, 0.9, 0.1, 2^5) 0.0719253
Gauss1 (75X2,175X2)	0.399263 (10^1, 0.1,2^{-5}) 0.062618	0.391528 (10^3, 0.9, 2^0) 0.01613	0.414253 (10^3,10^{-1}, 0.1, 1.345, 2^{-5}) 0.0006	0.409457 (10^1,0.5, 0.7, 2^1) 0.01021	0.391532 (10^5, 0.9, 0.1, 2^0) 0.02981	**0.389198** (10^3,10^{-1}, 0.1, 0.3, 2^1) 0.0072498

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Chwirut2 (17X2,37X2)	0.097839 (10^3, 0.1,2^3) 0.005741	0.116295 (10^{-1}, 0.9, 2^{-3}) 0.01351	0.093916 (10^1,10^{-3}, 0.1, 1, 2^5) 0.0002	**0.087731** (10^5,0.3, 0.3, 2^3) 0.00406	0.114574 (10^0, 0.9, 0.3, 2^{-3}) 0.02547	0.108905 (10^1,10^{-5}, 0.1, 0.1, 2^3) 0.004341
Roszman1 (8X2,17X2)	0.079154 (10^2, 0.0001,2^3) 0.002892	0.049779 (10^{-1}, 1, 2^0) 0.01399	0.212352 (10^2,10^{-3}, 0.1, 1, 2^5) 0.00008	0.292596 (10^5,0.3, 0.3, 2^3) 0.00387	**0.047258** (10^0, 1, 0.3, 2^0) 0.02349	0.048173 (10^1,10^{-5}, 1, 0.1, 2^0) 0.004168
INFY (226X6,525X6)	0.048422 (10^3, 0.1,2^{-5}) 0.512362	0.044445 (10^0, 0.3, 2^{-5}) 0.06776	0.050969 (10^5,10^{-1}, 0.1, 1, 2^{-5}) 0.02874	0.046417 (10^3,0.1, 0.6, 2^{-3}) 0.05323	**0.043197** (10^0, 0.9, 0.1, 2^{-5}) 0.05714	**0.043197** (10^0,10^{-3}, 0.9, 0.1, 2^{-5}) 0.0147357
ONGC_NS (221X6,514X6)	0.040992 (10^3, 0.1,2^{-5}) 0.545728	0.036287 (10^0, 0.1, 2^{-5}) 0.05809	0.045566 (10^5,10^{-1}, 0.1, 0.1, 2^{-5}) 0.02695	0.035167 (10^3,0.1, 0.4, 2^{-5}) 0.05503	0.036136 (10^1, 0.9, 0.1, 2^{-5}) 0.06659	**0.035021** (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0200796
XOM (226X6,525X6)	0.044282 (10^1, 0.1,2^{-3}) 0.52792	0.040461 (10^{-3}, 0.9, 2^{-3}) 0.05028	0.046998 (10^3,10^{-1}, 0.1, 0.1, 2^{-3}) 0.01829	**0.040378** (10^1,0.1, 0.5, 2^{-3}) 0.05652	0.040412 (10^{-1}, 0.1, 0.1, 2^{-3}) 0.10638	0.040412 (10^{-1},10^{-3}, 0.1, 0.1, 2^{-3}) 0.0254784
ATX (222X6,515X6)	0.027291 (10^1, 0.1,2^{-3}) 0.512387	0.025448 (10^0, 0.9, 2^{-5}) 0.05701	0.037052 (10^3,10^{-1}, 0.1, 1, 2^{-3}) 0.03231	**0.02157** (10^3,0.1, 0.4, 2^{-3}) 0.04981	0.023878 (10^0, 0.5, 0.1, 2^{-5}) 0.06775	0.024036 (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.020381
BSESN (220X6,513X6)	0.027462 (10^1, 0.1,2^{-3}) 0.529594	0.02329 (10^{-1}, 0.9, 2^{-5}) 0.05242	0.030826 (10^5,10^{-1}, 0.1, 1, 2^{-5}) 0.01959	0.025185 (10^1,0.1, 0.6, 2^{-5}) 0.04508	0.023207 (10^0, 0.9, 0.1, 2^{-5}) 0.06566	**0.022333** (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0285901

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
DJI (226X6,525X6)	0.024042 (10^1, 0.1,2^{-5}) 0.541827	0.02023 (10^{-1}, 0.9, 2^{-5}) 0.04965	0.021978 (10^3,10^{-1}, 0.1, 1, 2^{-3}) 0.01969	0.040544 (10^1,0.1, 0.7, 2^{-3}) 0.0463	**0.019661** (10^0, 0.1, 0.1, 2^{-5}) 0.06078	**0.019661** (10^0,10^{-3}, 0.1, 0.1, 2^{-5}) 0.0217884
GDAXI (227X6,528X6)	0.029271 (10^1, 0.1,2^{-5}) 0.615071	0.028296 (10^{-1}, 0.1, 2^{-5}) 0.05101	0.032066 (10^3,10^{-1}, 1, 1, 2^{-5}) 0.03326	**0.026** (10^3,0.1, 0.5, 2^{-5}) 0.05606	0.028228 (10^0, 0.9, 0.1, 2^{-5}) 0.07403	0.027102 (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0287863
MXX (225X6,525X6)	0.042338 (10^1, 0.1,2^{-3}) 0.559407	0.03836 (10^{-1}, 0.9, 2^{-3}) 0.04882	0.044005 (10^3,10^{-1}, 0.1, 1, 2^{-3}) 0.01992	0.040381 (10^1,0.1, 0.6, 2^{-3}) 0.0505	0.038028 (10^0, 0.1, 0.1, 2^{-3}) 0.12634	**0.036046** (10^0,10^{-5}, 0.1, 0.1, 2^{-5}) 0.0187388
N225 (220X6,512X6)	0.03575 (10^1, 0.1,2^{-5}) 0.497952	0.033073 (10^{-1}, 0.9, 2^{-5}) 0.0423	0.034927 (10^3,10^{-1}, 1, 1, 2^{-3}) 0.02333	0.034832 (10^1,0.1, 0.5, 2^{-3}) 0.0498	**0.032898** (10^0, 0.1, 0.1, 2^{-5}) 0.04164	0.033887 (10^0,10^{-5}, 0.1, 0.1, 2^{-5}) 0.0189589
Wankara (97X10,224X10)	0.064073 (10^5, 0.1,2^{-5}) 0.095646	**0.029472** (10^{-1}, 0.9, 2^{-3}) 0.01693	0.039629 (10^3,10^{-2}, 0.1, 0.1, 2^{-5}) 0.00491	0.034031 (10^1,0.1, 0.1, 2^{-1}) 0.01904	0.030836 (10^{-1}, 0.1, 0.1, 2^{-3}) 0.02922	0.030752 (10^0,10^{-5}, 0.1, 0.1, 2^{-5}) 0.0168666
Laser (298X5,695X5)	0.081302 (10^0, 0.1,2^1) 1.08662	0.04679 (10^{-1}, 0.1, 2^3) 0.10288	0.051436 (10^3,10^{-3}, 1, 1, 2^1) 0.04178	0.058167 (10^5,0.1, 0.4, 2^{-5}) 0.29224	0.042864 (10^0, 0.1, 0.1, 2^1) 0.08458	**0.040318** (10^0,10^{-5}, 0.1, 0.1, 2^1) 0.0288311
Dee (110X7,255X7)	0.107666 (10^0, 0.1,2^0) 0.12765	0.108599 (10^0, 0.9, 2^{-5}) 0.01828	**0.100024** (10^1,10^{-1}, 1, 0.1, 2^{-3}) 0.0039	0.107159 (10^1,0.1, 0.3, 2^{-1}) 0.01765	0.109692 (10^1, 0.9, 0.1, 2^{-5}) 0.02564	0.104123 (10^1,10^{-1}, 0.9, 0.1, 2^{-5}) 0.0134618

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
Friedman (360X6,840X6)	0.061985 $(10^1, 0.1, 2^{-1})$ 1.33317	0.050917 $(10^{-1}, 0.9, 2^{-1})$ 0.12606	0.070737 $(10^3, 10^{-1}, 1, 1, 2^{-1})$ 0.05842	0.049461 $(10^5, 0.1, 0.3, 2^{-5})$ 0.42496	0.051177 $(10^0, 0.1, 0.1, 2^{-1})$ 0.1547	**0.049195** $(10^{-5}, 10^{-5}, 0.1, 0.1, 2^{-1})$ 0.0283486
Mortgage (315X16,734X16)	0.056068 $(10^0, 0.1, 2^{-5})$ 1.05694	0.018067 $(10^{-1}, 0.1, 2^{-1})$ 0.10178	0.037756 $(10^3, 10^{-3}, 0.1, 1, 2^{-5})$ 0.05046	0.041226 $(10^1, 0.1, 0.6, 2^1)$ 0.46162	0.018797 $(10^0, 0.1, 0.1, 2^{-1})$ 0.10658	**0.01766** $(10^0, 10\text{ -}5, 0.1, 0.1, 2^{-3})$ 0.0292358
NNGC1_dataset_E1_V1_001 (111X5,259X5)	**0.174273** $(10^0, 0.1, 2^1)$ 0.119395	0.180527 $(10^{-1}, 0.1, 2^1)$ 0.01801	0.174901 $(10^1, 10^{-1}, 0.1, 1, 2^3)$ 0.00866	0.181674 $(10^1, 0.1, 0.4, 2^1)$ 0.01282	0.180662 $(10^0, 0.9, 0.1, 2^1)$ 0.0269	0.174368 $(10^0, 10^0, 0.9, 0.1, 2^3)$ 0.017439
NNGC1_dataset_F1_V1_008 (269X5,626X5)	0.077357 $(10^1, 0.1, 2^3)$ 0.737512	0.057912 $(10^{-1}, 0.9, 2^5)$ 0.1161	0.069382 $(10^1, 10^{-2}, 0.1, 1.345, 2^5)$ 0.02783	0.061483 $(10^1, 0.1, 0.4, 2^5)$ 0.07307	**0.057788** $(10^{-1}, 0.9, 0.1, 2^5)$ 0.07614	**0.057788** $(10^{-1}, 10^{-3}, 0.9, 0.1, 2^5)$ 0.0272107
NNGC1_dataset_F1_V1_009 (269X5,626X5)	0.077045 $(10^3, 0.1, 2^{-1})$ 0.735128	0.050753 $(10^{-1}, 0.9, 2^3)$ 0.10377	0.080904 $(10^1, 10^{-1}, 0.1, 1, 2^5)$ 0.02837	0.055231 $(10^1, 0.1, 0.5, 2^3)$ 0.06838	0.051426 $(10^0, 0.9, 0.1, 2^3)$ 0.09655	**0.048986** $(10^0, 10^{-1}, 0.9, 0.1, 2^5)$ 0.022648
NNGC1_dataset_F1_V1_010 (269X5,626X5)	0.073579 $(10^3, 0.1, 2^1)$ 0.745154	**0.044397** $(10^{-1}, 0.9, 2^3)$ 0.12389	0.068758 $(10^3, 10^{-2}, 0.1, 1, 2^3)$ 0.03004	0.057324 $(10^3, 0.1, 0.3, 2^1)$ 0.08042	0.045481 $(10^0, 0.9, 0.1, 2^3)$ 0.07167	0.05176 $(10^{-1}, 10^{-5}, 0.1, 0.1, 2^3)$ 0.0318337
NNGC1_dataset_F1_V1_006 (521X5,1214X5)	0.063795 $(10^3, 0.1, 2^1)$ 3.01858	**0.041367** $(10^{-1}, 0.9, 2^5)$ 0.37961	0.06495 $(10^3, 10^{-1}, 0.1, 0.1, 2^3)$ 0.08197	0.049231 $(10^1, 0.1, 0.3, 2^3)$ 0.28967	0.042234 $(10^{-1}, 0.9, 0.1, 2^5)$ 0.24892	0.047416 $(10^0, 10^{-5}, 0.9, 0.1, 2^3)$ 0.0936319

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
NN5_Complete_109 (237X5,550X5)	0.160178 (10^0, 0.1,2^1) 0.559677	0.169086 (10^{-1}, 0.1, 2^1) 0.07075	**0.153803** (10^1,10^{-1}, 0.1, 0.1, 2^3) 0.0163	0.167964 (10^1,0.1, 0.5, 2^1) 0.06157	0.170163 (10^0, 0.1, 0.1, 2^1) 0.07691	0.159525 (10^0,10^0, 0.1, 0.1, 2^3) 0.0250756
NN5_Complete_104 (237X5,550X5)	0.25227 (10^0, 0.1,2^1) 0.557894	0.246728 (10^{-1}, 0.9, 2^0) 0.08172	**0.223425** (10^1,10^{-1}, 0.1, 1, 2^5) 0.01747	0.243549 (10^1,0.1, 0.6, 2^1) 0.06105	0.245921 (10^0, 0.9, 0.1, 2^0) 0.08696	0.226614 (10^0,10^0, 0.9, 0.1, 2^5) 0.0190927
NN5_Complete_106 (237X5,550X5)	0.212644 (10^1, 0.1,2^1) 0.563886	0.227166 (10^{-3}, 0.3, 2^1) 0.05488	0.223541 (10^1,10^{-1}, 0.1, 0.1, 2^3) 0.01608	**0.203542** (10^1,0.1, 0.5, 2^1) 0.05971	0.226991 (10^{-1}, 0.1, 0.3, 2^1) 0.06767	0.207904 (10^0,10^0, 0.9, 0.1, 2^5) 0.0196831
NN5_Complete_103 (237X5,550X5)	**0.128071** (10^0, 0.1,2^{-1}) 0.55103	0.129677 (10^0, 0.9, 2^{-3}) 0.06733	0.130405 (10^1,10^{-1}, 1, 0.1, 2^{-1}) 0.02415	0.129381 (10^1,0.3, 0.4, 2^{-3}) 0.0482	0.129794 (10^1, 0.9, 0.1, 2^{-3}) 0.09831	0.129352 (10^1,10^{-1}, 0.1, 0.1, 2^{-1}) 0.0197878
NN5_Complete_101 (237X5,550X5)	0.150229 (10^0, 0.1,2^1) 0.566503	0.154038 (10^{-1}, 0.9, 2^1) 0.06516	0.154147 (10^1,10^{-1}, 0.1, 0.1, 2^1) 0.02724	**0.146225** (10^1,0.1, 0.5, 2^1) 0.05404	0.153808 (10^{-1}, 0.9, 0.1, 2^1) 0.07726	0.147679 (10^0,10^0, 0.9, 0.1, 2^3) 0.026214
NN5_Complete_105 (237X5,550X5)	0.206735 (10^0, 0.1,2^1) 0.59797	0.20295 (10^{-1}, 0.9, 2^1) 0.07722	0.212735 (10^1,10^{-1}, 1, 0.1, 2^3) 0.01905	0.215317 (10^1,0.1, 0.3, 2^1) 0.0591	0.204263 (10^0, 0.9, 0.1, 2^1) 0.07342	**0.202491** (10^0,10^0, 0.1, 0.1, 2^3) 0.0223573
NN5_Complete_111 (237X5,550X5)	**0.18836** (10^0, 0.1,2^1) 0.555449	0.211531 (10^{-5}, 0.7, 2^1) 0.06467	0.218224 (10^1,10^{-1}, 1, 0.1, 2^3) 0.01603	0.194376 (10^1,0.1, 0.4, 2^1) 0.06702	0.211115 (10^{-1}, 0.1, 0.1, 2^1) 0.06854	0.202472 (10^0,10^0, 0.1, 0.1, 2^3) 0.0195859

Table 10 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
D1dat_1_2000 (599X5,1396X5)	0.056375 $(10^3, 0.1, 2^{-5})$ 3.88997	0.039592 $(10^{-1}, 0.1, 2^1)$ 0.50661	0.056756 $(10^5, 10^{-1}, 0.1, 1, 2^{-3})$ 0.11601	0.04234 $(10^1, 0.1, 0.3, 2^1)$ 0.37059	0.039961 $(10^0, 0.1, 0.1, 2^1)$ 0.46588	**0.038144** $(10^0, 10^{-5}, 0.9, 0.1, 2^0)$ 0.0892223
Vineyard (16X4,36X4)	0.237345 $(10^0, 0.1, 2^{-1})$ 0.004569	0.236484 $(10^1, 0.9, 2^{-5})$ 0.01397	0.246205 $(10^1, 10^{-1}, 0.1, 0.1,$ $2^{-1})$ 0.00014	**0.213259** $(10^3, 0.5, 0.2, 2^{-1})$ 0.00665	0.236413 $(10^1, 0.9, 0.7, 2^{-5})$ 0.01649	0.240828 $(10^1, 10^1, 0.7, 0.1, 2^0)$ 0.0119425
COVID-19_spain (251X5,585X5)	0.118454 $(10^1, 0.1, 2^{-5})$ 0.650153	0.097704 $(10^{-1}, 0.1, 2^1)$ 0.08398	0.08227 $(10^3, 10^{-3}, 0.1, 1.345,$ $2^{-3})$ 0.01543	**0.081305** $(10^1, 0.3, 0.3, 2^{-3})$ 0.06166	0.097513 $(10^0, 0.1, 0.1, 2^1)$ 0.0508	0.084161 $(10^{-1}, 10^{-5}, 0.9, 0.1,$ $2^0)$ 0.0200923

The best result is shown as boldface

Table 11 Average ranks of SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR, and RHN-TSVR on RMSE values using Gaussian kernel with noise 5% for real world dataset

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Forestfires	4	5	3	1	6	2
Machine CPU	3	2	6	5	4	1
Auto-original	1	5	3	4	6	2
Winequality	2	5	3	6	4	1
SantafeA	6	3	1	5	4	2
Gas_furnace	6	3	4	5	2	1
Quake	5	2	4	6	3	1
Flex_robotarm	3	4	6	2	5	1
S&P500	5	1	6	4	3	2
Space-Ga	4	5	3	1	6	2
Gauss1	4	2	6	5	3	1
Chwirut2	3	6	2	1	5	4
Roszman1	4	3	5	6	1	2
INFY	5	3	6	4	1.5	1.5
ONGC_NS	5	4	6	2	3	1
XOM	5	4	6	1	2.5	2.5
ATX	5	4	6	1	2	3
BSESN	5	3	6	4	2	1
DJI	5	3	4	6	1.5	1.5
GDAXI	5	4	6	1	3	2
MXX	5	3	6	4	2	1
N225	6	2	5	4	1	3
Wankara	6	1	5	4	3	2

Table 11 continued

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Laser	6	3	4	5	2	1
Dee	4	5	1	3	6	2
Friedman	5	3	6	2	4	1
Mortgage	6	2	4	5	3	1
NNGC1_dataset_E1_V1_001	1	4	3	6	5	2
NNGC1_dataset_F1_V1_008	6	3	5	4	1.5	1.5
NNGC1_dataset_F1_V1_009	5	2	6	4	3	1
NNGC1_dataset_F1_V1_010	6	1	5	4	2	3
NNGC1_dataset_F1_V1_006	5	1	6	4	2	3
NN5_Complete_109	3	5	1	4	6	2
NN5_Complete_104	6	5	1	3	4	2
NN5_Complete_106	3	6	4	1	5	2
NN5_Complete_103	1	4	6	3	5	2
NN5_Complete_101	3	5	6	1	4	2
NN5_Complete_105	4	2	5	6	3	1
NN5_Complete_111	1	5	6	2	4	3
D1dat_1_2000	5	2	6	4	3	1
Vineyard	4	3	6	1	2	5
COVID-19_spain	6	5	2	1	4	3
Average rank	4.333333	3.404762	4.547619	3.4523809	3.380952	**1.880952**

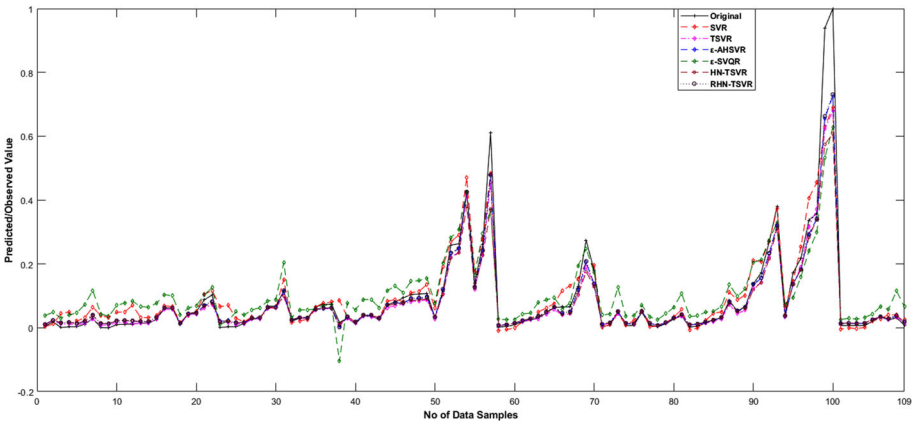

Fig. 8 Prediction over the testing dtaset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Machine CPU dataset with 5% noise. Gaussian kernel was used

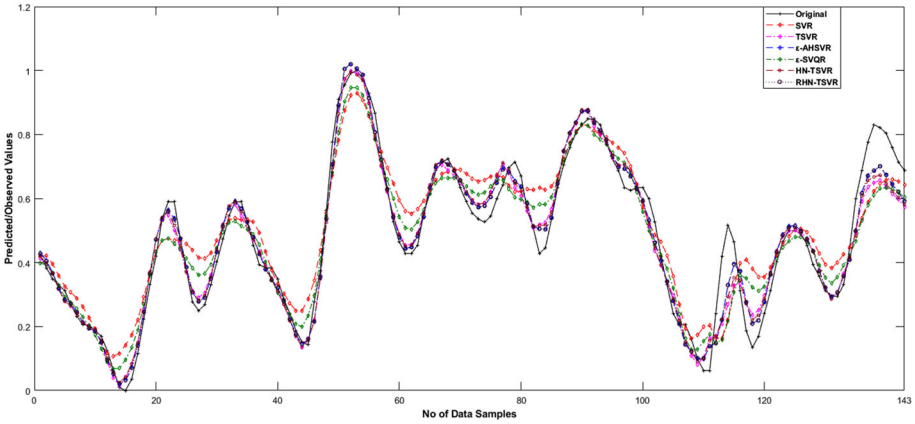

Fig. 9 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Gas furnace dataset with 5% noise. Gaussian kernel was used

test. The critical difference is the same as to the previous case i.e. 1.057. One can analyze few points as below:

1. The differences of the average ranks of RHN-TSVR compared to SVR, TSVR, ε-AHSVR and ε-SVQR are always greater than 1.057. Hence, it shows that RHN-TSVR is superior.
2. Now, check the dissimilarity of average ranks between RHN-TSVR and HN-TSVR, i.e. $(3.380952 - 1.889052 = 1.5)$ which is greater than the critical difference $(1.5 > 1.057)$. It declares that our proposed approach RHN-TSVR is better than HN-TSVR.

4.3.3 At Significant Noise Level 10%

Similar to significant noise level 5%, we have added more noise by increasing the significant noise level up to 10% on real-world datasets and tested the performance of proposed RHN-TSVR into the more noisy environment. All the significant results of SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and proposed approach RHN-TSVR are placed in Table 12 at noise

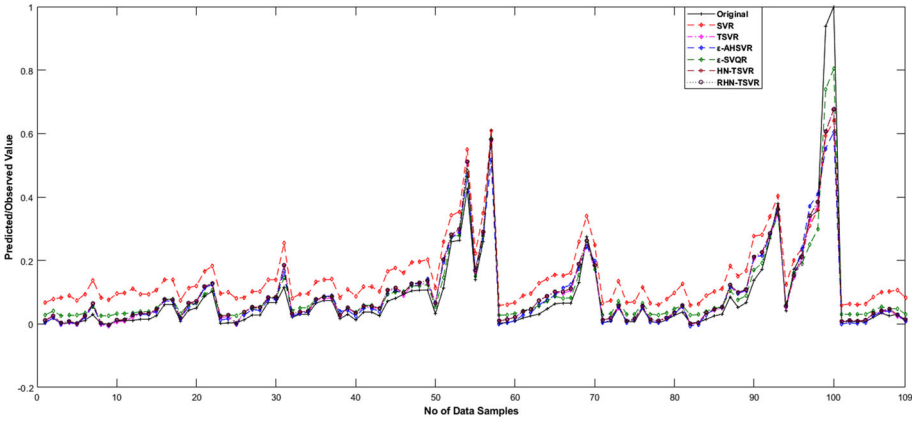

Fig. 10 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Machine CPU dataset with 10% noise. Gaussian kernel was used

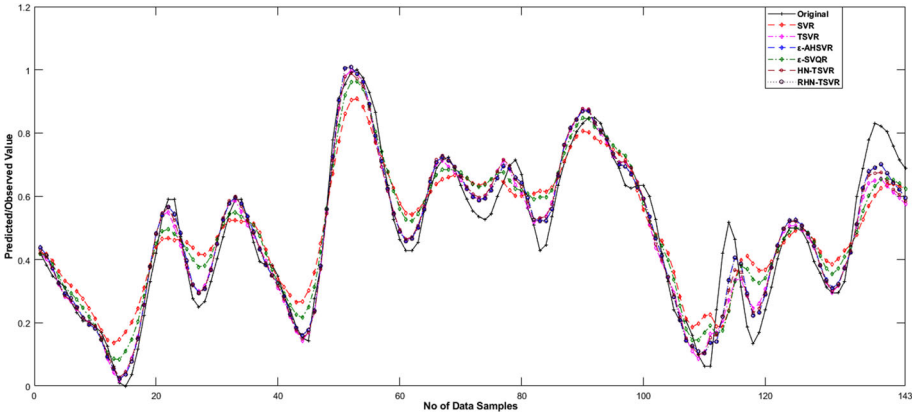

Fig. 11 Prediction over the testing dataset by SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR and RHN-TSVR on the Gas furnace dataset with 10% noise. Gaussian kernel was used

level 10%. It is clearly visible from Table 12 that the RHN-TSVR performs better in 18 cases overall. The average rank of all reported approaches with RHN-TSVR is computed in Table 13 where RHN-TSVR is having the lowest average rank. Similar to previous cases, the prediction graphs of Machine CPU and Gas furnace datasets at significant noise level 10% have been plotted in Figs. 10 and 11 and the same conclusion may be suggested.

Same as to previous cases, compute the values of χ_F^2 and F_F using Table 13 as

$$\begin{aligned}\chi_F^2 &= \frac{12\times 42}{6\times 7}\Big[\big(5.047619^2+3.2619048^2+3.2142857^2+3.833333^2\\ &\quad +3.547619^2+2.0952381^2\big)-\left(\frac{6\times 7^2}{4}\right)\Big]\\ &\cong 55.442\\ F_F &= \frac{(42-1)\times 55.442}{(42\times(6-1))-55.442}=14.707,\end{aligned}$$

Table 12 Performance comparison of RHN-TSVR with SVR, TSVR, ε-AHSVR, ε-SVQR, and HN-TSVR using Gaussian kernel with noise 10% on real world datasets

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
Forestfires (150X13,367X13)	0.070556 $(10^{-5}, 0.1, 2^{-5})$ 0.21571	0.0871489 $(10^{-4}, 0.9, 2^{5})$ 0.12293	0.070377 $(10^{-4}, 10^{-3}, 0.1,$ $1.345, 2^{5})$ 0.05959	**0.068864** $(10^{-1}, 0.9, 0.3, 2^{-5})$ 0.09252	0.08716 $(10^{5}, 0.9, 0.5, 2^{5})$ 0.18846	0.07046 $(10^{1}, 10^{5}, 0.1, 0.5, 2^{5})$ 0.010531
Machine CPU (100X8,109X8)	0.091585 $(10^{1}, 0.1, 2^{-5})$ 0.10812	**0.0505539** $(10^{-1}, 0.1, 2^{-5})$ 0.16241	0.058089 $(10^{2}, 10^{-3}, 1, 1.345,$ $2^{-5})$ 0.02408	0.035819 $(10^{3}, 0.001, 0.2, 2^{-5})$ 0.01103	0.05156 $(10^{-1}, 0.1, 0.1, 2^{-5})$ 0.04804	0.05156 $(10^{-1}, 10^{-3}, 0.1, 0.1,$ $2^{-5})$ 0.008127
Auto-original (100X8,292X8)	0.180002 $(10^{0}, 0.1, 2^{-1})$ 0.09423	0.16907 $(10^{-1}, 0.9, 2^{-1})$ 0.05376	0.172661 $(10^{5}, 10^{-2}, 1, 1.345,$ $2^{-2})$ 0.02505	0.17522 $(10^{5}, 0.3, 0.2, 2^{-1})$ 0.05034	0.16966 $(10^{0}, 0.9, 0.1, 2^{-1})$ 0.04592	**0.15891** $(10^{0}, 10^{-5}, 0.9, 0.1,$ $2^{-3})$ 0.0085984
Winequality (500X12,4398X12)	0.194176 $(10^{3}, 0.1, 2^{-5})$ 2.74866	**0.168979** $(10^{-1}, 0.9, 2^{-2})$ 0.88092	0.196418 $(10^{4}, 10^{-2}, 0.1, 1.345,$ $2^{-2})$ 0.60085	0.218789 $(10^{3}, 0.1, 0.5, 2^{-5})$ 1.39773	0.17353 $(10^{0}, 0.9, 0.1, 2^{-1})$ 1.26227	0.17405 $(10^{0}, 10^{-5}, 0.1, 0.1,$ $2^{-3})$ 0.0681787
SantafeA (500X6,495X6)	0.092708 $(10^{5}, 0.1, 2^{-5})$ 2.94318	0.0656795 $(10^{-1}, 0.9, 2^{2})$ 0.98025	0.069208 $(10^{4}, 10^{-3}, 0.1, 1.345,$ $2^{2})$ 0.45914	**0.040906** $(10^{5}, 0.1, 0.4, 2^{-3})$ 1.68406	0.07466 $(10^{0}, 0.9, 0.1, 2^{3})$ 1.01333	0.06537 $(10^{0}, 10^{-5}, 0.9, 0.1, 2^{1})$ 0.0525275

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Gas_furnace (150X7,143X7)	0.099002 (10^0, 0.1,2^{-5}) 0.23578	0.0622738 (10^{-1}, 0.9, 2^{-4}) 0.10164	0.052053 (10^5,10^{-3}, 0.1, 1.345, 2^{-5}) 0.05392	0.081241 (10^1,0.1, 0.1, 2^{-5}) 0.08915	0.05777 (10^0, 0.9, 0.1, 2^{-3}) 0.0824	**0.0516** (10^{-1},10^{-5}, 0.9, 0.1, 2^{-5}) 0.0185242
Quake (500X4,1678X4)	0.185914 (10^{-1}, 0.1,2^{-5}) 2.84951	**0.18575** (10^0, 0.1, 2^{-5}) 0.96815	0.186022 (10^{-3},10^{-3}, 1, 1.345, 2^{-3}) 0.35004	0.185954 (10^{-3},0.1, 0.3, 2^{-3}) 1.49636	0.18578 (10^0, 0.9, 0.3, 2^{-5}) 0.89834	0.18619 (10^1,10^5, 0.3, 0.3, 2^{-3}) 0.0608908
Flex_robotarm (500X10,519X10)	0.049069 (10^1, 0.1,2^{-3}) 2.78802	0.0446311 (10^{-1}, 0.9, 2^{-3}) 1.18666	**0.026773** (10^5,10^{-3}, 0.1, 1.345, 2^{-2}) 0.37359	0.029822 (10^3,0.1, 0.1, 2^{-5}) 1.46689	0.04443 (10^0, 0.1, 0.1, 2^{-3}) 1.23052	0.02832 (10^0,10^{-5}, 0.9, 0.1, 2^{-1}) 0.0653991
S&P500 (200X6,550X6)	0.031468 (10^1, 0.1,2^{-5}) 0.4072	0.0280403 (10^{-1}, 0.9, 2^{-4}) 0.30475	0.026838 (10^4,10^{-3}, 0.1, 1.345, 2^{-5}) 0.22688	0.054827 (10^3,0.1, 0.6, 2^{-3}) 0.18523	0.02819 (10^0, 0.9, 0.1, 2^{-5}) 0.11794	**0.02665** (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0243972
Space-Ga (500X7,2607X7)	0.386259 (10^3, 0.1,2^1) 2.71523	0.322389 (10^{-5}, 0.1, 2^5) 0.68511	0.297924 (10^2,10^{-3}, 0.1, 1.345, 2^5) 0.38471	**0.271389** (10^{-1},0.1, 0.6, 2^{-1}) 1.26518	0.32239 (10^{-5}, 0.1, 0.3, 2^5) 0.8876	0.28372 (10^1,10^{-1}, 0.5, 0.1, 2^5) 0.0599814
Gauss1 (75X2,175X2)	0.391899 (10^0,0.1,2^{-3}) 0.06196	0.385943 (10^3, 0.9, 2^0) 0.01724	**0.381646** (10^1,10^{-3}, 0.1, 1.345, 2^{-4}) 0.0329	0.420118 (10^1,0.3, 0.8, 2^{-1}) 0.00907	0.385946 (10^5, 0.9, 0.3, 2^0) 0.02471	0.384733 (10^5,10^{-1}, 0.1, 0.5, 2^1) 0.0053786

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Chwirut2 (17X2,37X2)	0.095709 (10^0,0.1,2^5) 0.00532	0.111636 (10^0, 0.9, 2^{-3}) 0.01408	0.094656 (10^4,10^{-3}, 1, 1.345, 2^5) 0.00143	0.096011 (10^1,0.1, 0.4, 2^5) 0.00409	0.111998 (10^1, 0.9, 0.1, 2^{-3}) 0.02217	**0.093851** (10^0,10^{-5}, 0.1, 0.1, 2^3) 0.0063419
Roszman1 (8X2,17X2)	0.435543 (10^2,0.0001,2^5) 0.00308	0.048457 (10^0, 1, 2^0) 0.01349	0.156611 (10^2,10^{-3}, 1, 1.345, 2^5) 0.00038	0.298359 (10^1,0.1, 0.4, 2^5) 0.00394	0.047929 (10^1, 1, 0.1, 2^0) 0.01341	**0.045437** (10^0,10^{-5}, 1, 0.1, 2^0) 0.0066292
INFY (226X6,525X6)	0.048232 (10^3,0.1,2^{-5}) 0.54122	0.045631 (10^{-3}, 0.7, 2^{-5}) 0.05414	0.047149 (10^5,10^{-3}, 1, 1.345, 2^{-5}) 0.14077	**0.042672** (10^1,0.1, 0.4, 2^{-3}) 0.04483	0.044563 (10^0, 0.9, 0.1, 2^{-5}) 0.07999	0.046205 (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0163618
ONGC_NS (221X6,514X6)	0.040414 (10^1,0.1,2^{-5}) 0.51419	0.037272 (10^{-1}, 0.9, 2^{-5}) 0.04756	**0.035483** (10^4,10^{-3}, 0.1, 1.345, 2^{-5}) 0.10074	0.03802 (10^5,0.1, 0.2, 2^{-3}) 0.05531	0.037329 (10^0, 0.1, 0.1, 2^{-5}) 0.05609	0.0358 (10^{-1},10^{-5}, 0.1, 0.1, 2^{-5}) 0.0151793
XOM (226X6,525X6)	0.043564 (10^1,0.1,2^{-5}) 0.54421	0.040508 (10^{-3}, 0.1, 2^{-3}) 0.05942	0.040229 (10^4,10^{-3}, 1, 1.345, 2^{-4}) 0.17387	0.04221 (10^1,0.1, 0.4, 2^{-3}) 0.04784	0.040477 (10^{-1}, 0.9, 0.1, 2^{-3}) 0.04748	**0.040179** (10^{-1},10^{-5}, 0.1, 0.1, 2^{-5}) 0.0196942
ATX (222X6,515X6)	0.031246 (10^1,0.1,2^{-5}) 0.50765	**0.030522** (10^{-1}, 0.1, 2^{-1}) 0.05096	0.035835 (10^5,10^{-3}, 0.1, 1.345, 2^{-1}) 0.09628	0.039418 (10^5,0.1, 0.6, 2^{-5}) 0.05423	0.031933 (10^0, 0.9, 0.1, 2^{-1}) 0.0676	0.031411 (10^0,10^{-5}, 0.9, 0.1, 2^{-1}) 0.0201047
BSESN (220X6,513X6)	0.024455 (10^1,0.1,2^{-5}) 0.50639	0.025717 (10^{-1}, 0.3, 2^{-3}) 0.04473	0.023904 (10^4,10^{-2}, 0.1, 1.345, 2^{-5}) 0.09047	0.023822 (10^3,0.1, 0.6, 2^{-5}) 0.04013	0.024003 (10^0, 0.9, 0.1, 2^{-5}) 0.11163	**0.023631** (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0189221

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
DJI (226X6,525X6)	0.060085 (10^0,0.1,2^{-1}) 0.56475	0.038362 (10^{-1}, 0.9, 2^{-3}) 0.04941	0.059886 (10^1,10^{-2}, 0.1, 1.345, 2^{-1}) 0.11015	0.050961 (10^1,0.1, 0.7, 2^{-1}) 0.04794	**0.038214** (10^0, 0.9, 0.1, 2^{-3}) 0.06362	0.054533 (10^1,10^0, 0.1, 0.1, 2^{-1}) 0.0188101
GDAXI (227X6,528X6)	0.035349 (10^1,0.1,2^{-5}) 0.55039	0.02908 (10^{-1}, 0.1, 2^{-5}) 0.04355	0.02882 (10^5,10^{-3}, 1, 1.345, 2^{-5}) 0.11852	0.029441 (10^3,0.1, 0.4, 2^{-5}) 0.05947	0.029419 (10^0, 0.9, 0.1, 2^{-5}) 0.06226	**0.028775** (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0240547
MXX (225X6,525X6)	0.040878 (10^1,0.1,2^{-3}) 0.52684	0.0394 (10^{-3}, 0.3, 2^{-3}) 0.04676	0.040972 (10^5,10^{-3}, 1, 1.345, 2^{-3}) 0.10279	**0.034736** (10^3,0.1, 0.4, 2^{-5}) 0.05477	0.040958 (10^0, 0.9, 0.3, 2^{-3}) 0.06418	0.041988 (10^0,10^{-5}, 0.9, 0.3, 2^{-3}) 0.0191296
N225 (220X6,512X6)	0.03902 (10^1,0.1,2^{-5}) 0.51209	0.035597 (10^{-1}, 0.1, 2^{-5}) 0.04948	0.035549 (10^3,10^{-3}, 1, 1.345, 2^{-4}) 0.09	0.038293 (10^1,0.1, 0.5, 2^{-5}) 0.03976	**0.03552** (10^{-1}, 0.9, 0.1, 2^{-5}) 0.05325	**0.03552** (10^{-1},10^{-3}, 0.9, 0.1, 2^{-5}) 0.0193947
Wankara (97X10,224X10)	0.063142 (10^1,0.1,2^{-5}) 0.09453	**0.030462** (10^{-1}, 0.1, 2^{-3}) 0.01633	0.035809 (10^4,10^{-3}, 0.1, 1.345, 2^{-5}) 0.00244	0.044248 (10^1,0.1, 0.1, 2^{-3}) 0.02061	0.03271 (10^0, 0.1, 0.1, 2^{-3}) 0.02543	0.0349 (10^0,10^{-5}, 0.1, 0.1, 2^{-5}) 0.0199694
Laser (298X5,695X5)	0.085416 (10^1,0.1,2^0) 1.02388	0.047765 (10^{-1}, 0.1, 2^3) 0.10974	0.044438 (10^5,10^{-3}, 0.1, 1.345, 2^1) 0.02504	0.070032 (10^1,0.1, 0.4, 2^{-1}) 0.30005	0.049892 (10^0, 0.1, 0.1, 2^3) 0.09753	**0.043683** (10^{-1},10^{-5}, 0.1, 0.1, 2^1) 0.0249245
Dee (110X7,255X7)	0.112566 (10^0,0.1,2^0) 0.13122	0.105096 (10^0, 0.9, 2^{-5}) 0.01948	**0.102947** (10^2,10^{-1}, 1, 1.345, 2^{-5}) 0.00378	0.107498 (10^1,0.1, 0.7, 2^{-5}) 0.02758	0.105666 (10^1, 0.1, 0.1, 2^{-5}) 0.02575	0.104154 (10^1,10^{-1}, 0.9, 0.1, 2^{-5}) 0.0162945

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
Friedman (360X6,840X6)	0.069441 (10^0,0.1,2^{-1}) 1.32811	0.051352 (10^{-1}, 0.1, 2^{-1}) 0.10366	0.053727 (10^3,10^{-3}, 1, 1.345, 2^{-1}) 0.04133	**0.048789** (10^5,0.1, 0.3, 2^{-5}) 0.40358	0.051929 (10^0, 0.1, 0.1, 2^{-1}) 0.14075	0.051929 (10^0,10^{-3}, 0.1, 0.1, 2^{-1}) 0.0346828
Mortgage (315X16,734X16)	0.059066 (10^0,0.1,2^{-5}) 1.09222	**0.015498** (10^{-1}, 0.3, 2^{-3}) 0.11248	0.017121 (10^5,10^{-3}, 0.1, 1.345, 2^{-5}) 0.02693	0.046823 (10^1,0.1, 0.1, 2^{-5}) 1.69654	0.016332 (10^0, 0.9, 0.1, 2^{-3}) 0.10476	0.016587 (10^0,10^{-5}, 0.9, 0.1, 2^{-5}) 0.0350727
NNGC1_dataset_E1_V1_001 (111X5,259X5)	0.188128 (10^1,0.1,2^1) 0.12438	0.180734 (10^{-5}, 0.1, 2^0) 0.01945	0.17804 (10^1,10^{-1}, 0.1, 1.345, 2^2) 0.00346	0.179554 (10^1,0.3, 0.5, 2^1) 0.01199	0.185076 (10^0, 0.1, 0.3, 2^1) 0.03176	**0.17484** (10^0,10^0, 0.9, 0.1, 2^3) 0.0158275
NNGC1_dataset_F1_V1_008 (269X5,626X5)	0.076954 (10^0,0.1,2^5) 0.74556	0.060209 (10^{-1}, 0.9, 2^5) 0.12007	0.060038 (10^3,10^{-3}, 1, 1.345, 2^5) 0.02009	0.065316 (10^1,0.1, 0.1, 2^5) 0.07343	**0.060003** (10^{-1}, 0.9, 0.1, 2^5) 0.09681	**0.060003** (10^{-1},10^{-3}, 0.9, 0.1, 2^5) 0.0231774
NNGC1_dataset_F1_V1_009 (269X5,626X5)	0.080366 (10^3,0.1,2^{-1}) 0.74747	0.055165 (10^{-1}, 0.7, 2^3) 0.10069	0.053169 (10^1,10^{-3}, 1, 1.345, 2^5) 0.0182	0.061428 (10^3,0.1, 0.2, 2^1) 0.08028	0.055746 (10^0, 0.9, 0.1, 2^3) 0.09388	**0.052221** (10^0,10^{-1}, 0.1, 0.1, 2^5) 0.0302937
NNGC1_dataset_F1_V1_010 (269X5,626X5)	0.069884 (10^3,0.1,2^0) 0.74931	0.044151 (10^{-1}, 0.9, 2^3) 0.09345	0.043029 (10^1,10^{-3}, 0.1, 1.345, 2^5) 0.01833	0.052085 (10^1,0.1, 0.2, 2^3) 0.08409	0.045211 (10^0, 0.9, 0.1, 2^3) 0.10088	**0.042634** (10^0,10^{-1}, 0.9, 0.1, 2^5) 0.0342732
NNGC1_dataset_F1_V1_006 (521X5,1214X5)	0.06958 (10^1,0.1,2^3) 2.97391	0.047711 (10^{-1}, 0.9, 2^5) 0.50363	0.046947 (10^1,10^{-3}, 1, 1.345, 2^5) 0.07101	0.061655 (10^1,0.1, 0.3, 2^5) 0.3305	0.048554 (10^0, 0.9, 0.1, 2^5) 0.38209	**0.046072** (10^0,10^{-1}, 0.9, 0.1, 2^5) 0.0727278

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE ($C_1 = C_2$, ε, μ) Time	ε-AHSVR RMSE (C, $\varepsilon_\alpha = \varepsilon_\beta$, ζ_α, ζ_β, μ) Time	ε-SVQR RMSE (C, ε, θ, μ) Time	HN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time	RHN-TSVR RMSE ($C_1 = C_2$, $\varepsilon_1 = \varepsilon_2$, $\varepsilon_1^* = \varepsilon_2^*$, μ) Time
NN5_Complete_109 (237X5,550X5)	0.166291 (10^0,0.1,2^1) 0.55777	0.171426 (10^{-1}, 0.1, 2^1) 0.09335	0.16871 (10^1,10^{-3}, 0.1, 1.345, 2^3) 0.01475	**0.155751** (10^1,0.1, 0.3, 2^1) 0.0501	0.172463 (10^0, 0.1, 0.1, 2^1) 0.08337	0.159981 (10^0,10^0, 0.1, 0.1, 2^3) 0.0246379
NN5_Complete_104 (237X5,550X5)	0.248315 (10^0,0.1,2^1) 0.57739	0.255563 (10^{-1}, 0.9, 2^0) 0.07181	0.233829 (10^{-1},10^{-3}, 0.1, 1.345, 2^4) 0.01461	0.271226 (10^1,0.1, 0.7, 2^1) 0.05814	0.254416 (10^0, 0.9, 0.1, 2^0) 0.09904	**0.211625** (10^1,10^1, 0.9, 0.1, 2^5) 0.0218523
NN5_Complete_106 (237X5,550X5)	0.214481 (10^1,0.1,2^1) 0.56271	0.225332 (10^{-5}, 0.7, 2^1) 0.0672	0.224239 (10^1,10^{-3}, 0.1, 1.345, 2^4) 0.01511	**0.206398** (10^1,0.1, 0.5, 2^1) 0.05335	0.225333 (10^{-5}, 0.1, 0.5, 2^1) 0.06475	0.20665 (10^0,10^0, 0.9, 0.1, 2^5) 0.0196063
NN5_Complete_103 (237X5,550X5)	0.128811 (10^0,0.1,2^{-1}) 0.57129	0.129722 (10^0, 0.9, 2^{-3}) 0.07779	0.12998 (10^1,10^{-3}, 1, 1.345, 2^{-1}) 0.01483	**0.127552** (10^1,0.3, 0.4, 2^{-1}) 0.05419	0.129951 (10^1, 0.1, 0.1, 2^{-3}) 0.07177	0.129655 (10^1,10^{-1}, 0.1, 0.1, 2^{-1}) 0.0198243
NN5_Complete_101 (237X5,550X5)	**0.150035** (10^0,0.1,2^1) 0.55307	0.157233 (10^0, 0.1, 2^0) 0.07508	0.155235 (10^1,10^{-3}, 1, 1.345, 2^2) 0.01382	0.15831 (10^1,0.3, 0.4, 2^1) 0.06478	0.157715 (10^1, 0.1, 0.1, 2^0) 0.07666	0.151755 (10^0,10^0, 0.5, 0.1, 2^3) 0.0237155
NN5_Complete_105 (237X5,550X5)	0.207145 (10^0,0.1,2^1) 0.5438	0.199703 (10^{-1}, 0.9, 2^1) 0.06793	0.200107 (10^1,10^{-3}, 1, 1.345, 2^5) 0.0142	0.208579 (10^1,0.1, 0.3, 2^1) 0.0545	**0.196538** (10^{-1}, 0.9, 0.1, 2^5) 0.07154	0.19833 (10^0,10^0, 0.9, 0.1, 2^5) 0.0243086
NN5_Complete_111 (237X5,550X5)	0.217826 (10^0,0.1,2^3) 0.55873	0.214238 (10^{-1}, 0.9, 2^0) 0.05185	0.204766 (10^{-1},10^{-3}, 0.1, 1.345, 2^5) 0.01406	**0.204239** (10^1,0.1, 0.4, 2^1) 0.06088	0.21311 (10^0, 0.9, 0.1, 2^0) 0.0681	0.206733 (10^0,10^0, 0.1, 0.1, 2^3) 0.0197837

Table 12 continued

Dataset (Train size, Test size)	SVR RMSE (C, ε, μ) Time	TSVR RMSE $(C_1 = C_2,$ $\varepsilon, \mu)$ Time	ε-AHSVR RMSE $(C, \varepsilon_\alpha = \varepsilon_\beta, \zeta_\alpha,$ $\zeta_\beta, \mu)$ Time	ε-SVQR RMSE $(C, \varepsilon,$ $\theta, \mu)$ Time	HN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time	RHN-TSVR RMSE $(C_1 = C_2, \varepsilon_1 = \varepsilon_2,$ $\varepsilon_1^* = \varepsilon_2^*, \mu)$ Time
D1dat_1_2000 (599X5,1396X5)	0.056778 $(10^3, 0.1, 2^{-5})$ 4.0872	0.042117 $(10^{-1}, 0.9, 2^0)$ 0.71631	0.043586 $(10^4, 10^{-3}, 1, 1.345, 2^{-1})$ 0.08671	0.043239 $(10^3, 0.1, 0.2, 2^{-1})$ 0.47343	0.043216 $(10^0, 0.9, 0.1, 2^0)$ 0.4761	**0.041703** $(10^0, 10^{-5}, 0.9, 0.1, 2^{-1})$ 0.0961339
Vineyard (16X4,36X4)	0.256848 $(10^0, 0.1, 2^{-1})$ 0.00456	0.245166 $(10^0, 0.9, 2^{-5})$ 0.01548	0.239557 $(10^2, 10^{-1}, 0.1, 1.345, 2^{-1})$ 0.00014	0.242875 $(10^1, 0.1, 0.1, 2^{-1})$ 0.00743	0.239164 $(10^1, 0.1, 0.1, 2^{-5})$ 0.02454	**0.233692** $(10^1, 10^1, 0.1, 0.1, 2^0)$ 0.0122392
COVID-19_spain (251X5,585X5)	0.11715 $(10^1, 0.1, 2^{-5})$ 0.63258	0.104614 $(10^{-1}, 0.1, 2^1)$ 0.07824	0.113934 $(10^4, 10^{-3}, 0.1, 1.345, 2^1)$ 0.01442	**0.091393** $(10^3, 0.3, 0.5, 2^{-1})$ 0.04905	0.104057 $(10^0, 0.1, 0.1, 2^1)$ 0.04803	0.098165 $(10^{-1}, 10^{-5}, 0.9, 0.1, 2^0)$ 0.0281752

The best result is shown as boldface

Table 13 Average ranks of SVR, TSVR, ε-AHSVR, ε-SVQR, HN-TSVR, and RHN-TSVR on RMSE values using Gaussian kernel with noise 10% for real world dataset

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Forestfires	4	5	2	1	6	3
Machine_8	5	1	4	6	2.5	2.5
Auto-original	6	2	4	5	3	1
Winequality	4	1	5	6	2	3
SantafeA	6	3	4	1	5	2
Gas_furnace	6	4	2	5	3	1
Quake	3	1	5	4	2	6
Flex_robotarm	6	5	1	3	4	2
S&P500	5	3	2	6	4	1
Space-Ga	6	4	3	1	5	2
Gauss1	5	3	1	6	4	2
Chwirut2	3	5	2	4	6	1
Roszman1	6	3	4	5	2	1
INFY	6	3	5	1	2	4
ONGC_NS	6	3	1	5	4	2
XOM	6	4	2	5	3	1
ATX	2	1	5	6	4	3
BSESN	5	6	3	2	4	1
DJI	6	2	5	3	1	4
GDAXI	6	3	2	5	4	1
MXX	3	2	5	1	4	6
N225	6	4	3	5	1.5	1.5
Wankara	6	1	4	5	2	3

Table 13 continued

Datasets	SVR	TSVR	ε-AHSVR	ε-SVQR	HN-TSVR	RHN-TSVR
Laser	6	3	2	5	4	1
Dee	6	3	1	5	4	2
Friedman	6	2	5	1	3.5	3.5
Mortgage	6	1	4	5	2	3
NNGC1_dataset_E1_V1_001	6	4	2	3	5	1
NNGC1_dataset_F1_V1_008	6	4	3	5	1.5	1.5
NNGC1_dataset_F1_V1_009	6	3	2	5	4	1
NNGC1_dataset_F1_V1_010	6	3	2	5	4	1
NNGC1_dataset_F1_V1_006	6	3	2	5	4	1
NN5_Complete_109	3	5	4	1	6	2
NN5_Complete_104	3	5	2	6	4	1
NN5_Complete_106	3	5	4	1	6	2
NN5_Complete_103	2	4	6	1	5	3
NN5_Complete_101	1	4	3	6	5	2
NN5_Complete_105	5	3	4	6	1	2
NN5_Complete_111	6	5	2	1	4	3
D1dat_1_2000	6	2	5	4	3	1
Vineyard	6	5	3	4	2	1
COVID-19_spain	6	4	5	1	3	2
Average rank	5.047619	3.2619048	3.2142857	3.8333333	3.547619	**2.0952381**

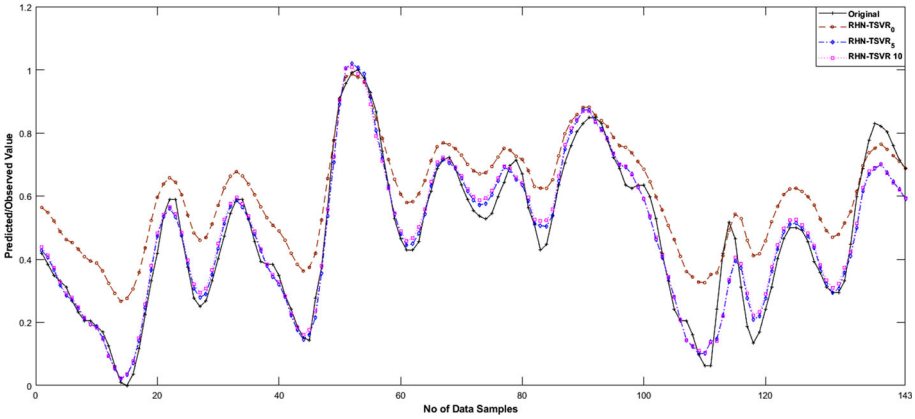

Fig. 12 Prediction/observed value over the testing dataset by RHN-TSVR on the Gas furnace dataset with 0%, 5% and 10% noise. Gaussian kernel was used

Here, F_F is also greater than the critical value on degree of freedom (5, 205) i.e. ($14.707 > 2.2581$). So, null hypothesis H_0 is also rejected in this case which means all the models may have some significant differences. Now, we perform Nemenyi test on these algorithms and few points may be concluded based on test. Similar to previous two cases, RHN-TSVR is still having lowest average rank at significant noise level 10% and the average rank differences with others are always greater than the critical difference i.e. 1.057. So, one can notice that overall the performance of RHN-TSVR is outperformed to others models.

4.3.4 Effect of Increasing Noise Percentage

In this section, we have shown the effect of increasing significant noise level percentage like 0%, 5%, and 10% on real-world dataset. As we have already discussed the performance of proposed RHN-TSVR on real-world datasets at different significant noise level in the previous sections. We have plotted the performance accuracy for Gas furnace dataset in Fig. 12 at different noise levels 0%, 5% and 10%.

One can see that the test data samples are plotted in the form of the black line and the prediction results obtained on RHN-TSVR for different noise levels 0%, 5% and 10% are shown in the form of brown, blue, and pink dotted lines respectively. From Fig. 12, one can analyze the impact of increasing noise percentage for proposed RHN-TSVR model. Here, noisy results are having a closer relationship with the desired output which justifies the applicability and usability of the proposed RHN-TSVR model for noisy environment.

5 Conclusion and Future Scope

In this paper, regularized version of TSVR with Huber loss propose to avoid the singularity problem of HN-TSVR by implementing the structural risk minimization principle as regularization based twin support vector regression with Huber loss (RHN-TSVR) and further, justify the noise insensitivity of the proposed model by different variation of the significant noise level such as 0%, 5%, and 10%. As we know that TSVR accommodates ε-insensitive loss function which is not capable to deal with different types of noise and outliers. Classical

Huber loss function performs as quadratic for small error and linear for others. It is the combination of the Laplacian loss and Gaussian loss function which gives better generalization performance for data having Gaussian noise and outliers. The performance of the proposed approach is tested and validated on different benchmark real-world datasets at different significant levels and on artificial datasets having a different type of noises for the non-linear kernel. The proposed RHN-TSVR shows better prediction accuracy in most of the cases and takes less or comparable computational time in comparison to existing approaches such as SVR, TSVR, ε-AHSVR, ε-SVQR, and HN-TSVR. A comparative study is analyzed based on numerical experiments which justify the importance of RHN-TSVR model in comparison to reported approaches especially to deal with the data, having noise and outliers. One can apply this model for the financial time series forecasting which can be one of the applications. In future, the computational cost may be reduced by suggesting the iterative approach.

Compliance with Ethical Standards

Conflict of interest Authors have no conflict of interest.

References

1. Anand P, Rastogi R, Chandra S (2019) A new asymmetric ϵ-insensitive pinball loss function based support vector quantile regression model. Appl Soft Comput 94:1–14
2. Anand P, Rastogi R, Chandra S (2019) Support vector regression via a combined reward cum penalty loss function. arXiv: 1904.12331v2 [cs.LG] version: 2, pp 1–13
3. Bai L, Shao Y-H, Wang Z, Li C-N (2019) Clustering by twin support vector machine and least square twin support vector classifier with uniform output coding. Knowl-Based Syst 163:227–240
4. Balasundaram S, Prasad SC (2019) Robust twin support vector regression based on Huber loss function. Neural Comput Appl 32:1–25
5. Balasundaram S, Meena Y (2019) Robust support vector regression in primal with asymmetric Huber loss. Neural Process Lett 49(3):1399–1431
6. Chen S-G, Xiao-Jun W (2018) A new fuzzy twin support vector machine for pattern classification. Int J Mach Learn Cybernet 9(9):1553–1564
7. Chen, S, Liu X, Li B (2018) A cost-sensitive loss function for machine learning. In: International conference on database systems for advanced applications vol 10829. LNCS. Springer, Cham, pp 255–268
8. Chen C, Yan C, Zhao N, Guo B, Liu G (2017) A robust algorithm of support vector regression with a trimmed Huber loss function in the primal. Soft Comput 21(18):5235–5243
9. Chen Z, Matousek R, Wanke P (2018) Chinese bank efficiency during the global financial crisis: a combined approach using satisficing DEA and support vector machines. N Am J Econ Finance 43:71–86
10. Chen C, Li Y, Yan C, Dai H, Liu G (2015) A robust algorithm of multiquadric method based on an improved Huber loss function for interpolating remote-sensing-derived elevation data sets. Remote Sens 7(3):3347–3371
11. Chu W, Sathiya Keerthi S, Ong CJ (2004) Bayesian support vector regression using a unified loss function. IEEE Trans Neural Netw 15(1):29–44
12. Chuang C-C (2007) Fuzzy weighted support vector regression with a fuzzy partition. IEEE Trans Syst Man Cybern Part B (Cybern) 37(3):630–640
13. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
14. COVID19S (2020)[online]. https:/dataverse.harvard.edu/dataset.xhtml/
15. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
16. Cui W, Yan Xu (2009) Adaptive weighted least square support vector machine regression integrated with outlier detection and its application in QSAR. Chemometr Intell Lab Syst 98(2):130–135
17. Demšar, J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30

18. Deylami H-M, PrasadSingh Y (2012) Cybercrime detection techniques based on support vector machines. Artif Intell Res 2(1):1
19. Drucker H, Burges CJC, Kaufman L, Smola AJ, Vapnik V (1997) Support vector regression machines. In: Advances in neural information processing systems, vol 9. pp 155–161
20. Financial Dataset (2020) [online]. http://finance.yahoo.com
21. Flexible Robot Arm (2020) [online]. http://homes.esat.kuleuven.be/~smc/daisydata.html
22. Forghani Y, Sigari Tabrizi R, Sadoghi Yazdi H, Mohammad-R. Akbarzadeh-T (2011) Fuzzy support vector regression. In: 2011 1st international eConference on computer and knowledge engineering (ICCKE), IEEE (2011), pp 28–33
23. Fung GM, Mangasarian OL (2005) Multicategory proximal support vector machine classifiers. Mach Learn 59(1-2):77–97
24. Gu B, Fang J, Pan F, Bai Z (2020) Fast clustering-based weighted twin support vector regression.". Soft Comput 24:1–17
25. Gupta U, Gupta D (2019) An improved regularization based Lagrangian asymmetric ν-twin support vector regression using pinball loss function. Appl Intell 49(10):3606–3627
26. Gupta U, Gupta, D (2018) Lagrangian twin-bounded support vector machine based on L2-norm. In: Recent developments in machine learning and data analytics, vol 740. AISC. Springer, Singapore, pp 431–444
27. Gupta D, Pratama M, Ma Z, Li J, Prasad M (2019) Financial time series forecasting using twin support vector regression. PLoS ONE 14(3):0211402
28. Gupta U, Gupta D, Prasad M (2018) Kernel target alignment based fuzzy least square twin bounded support vector machine. In: 2018 IEEE symposium series on computational intelligence (SSCI). IEEE
29. Hazarika BB, Gupta D, Berlin M (2020) Modeling suspended sediment load in a river using extreme learning machine and twin support vector regression with wavelet conjunction. Environ Earth Sci 79:234
30. Hong DH, Hwang C (2005) Interval regression analysis using quadratic loss support vector machine. IEEE Trans Fuzzy Syst 13(2):229–237
31. Huang M-L (2015) Intersection traffic flow forecasting based on ν-GSVR with a new hybrid evolutionary algorithm. Neurocomputing 147:343–349
32. Huang X, Shi L, Suykens JAK (2014) Asymmetric least squares support vector machine classifiers.". Comput Stat Data Anal 70:395–405
33. Huber PJ (1964) Robust estimation of a location parameter. Ann Math Stat 35(1):73–101
34. Huber PJ (1996) Robust statistical procedures, vol 68. SIAM, Philadelphia
35. Hwang C, Hong DH, Seok KH (2006) Support vector interval regression machine for crisp input and output data. Fuzzy Sets Syst 157(8):1114–1125
36. Jayadeva, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
37. Kaneko H, Funatsu K (2014) Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants. Chemometr Intell Lab Syst 137:57–66
38. KEEL (2020) [online]. https://sci2s.ugr.es/keel/html/
39. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
40. Liu LL, Zhao Y, Kong L, Liu M, Dong L, Ma F, Pang Z (2018) Robust real-time heart rate prediction for multiple subjects from facial video using compressive tracking and support vector machine. J Med Imaging 5(2):024503
41. Liu X, Zhu T, Zhai L, Liu J (2017) Mass classification of benign and malignant with a new twin support vector machine joint $l_{2,1}$ - norm. Int J Mach Learn Cybern 10:1–17
42. Mangasarian OL, Musicant DR (2000) Robust linear and support vector regression. IEEE Trans Pattern Anal Mach Intell 22(9):950–955
43. Mao X, Wang Y, Liu X, Guo Y (2017) An adaptive weighted least square support vector regression for hysteresis in piezoelectric actuators. Sens Actuators A 263:423–429
44. Maulik U, Chakraborty D (2017) Remote sensing image classification: a survey of support-vector-machine-based advanced techniques. IEEE Geosci Remote Sens Mag 5(1):33–52
45. Mehrkanoon S, Huang X, Suykens JAK (2014) Non-parallel support vector classifiers with different loss functions. Neurocomputing 143:294–301
46. Melacci S, Belkin M (2011) Laplacian support vector machines trained in the primal. J Mach Learn Res 12(Mar):1149–1184
47. Niu J, Chen J, Yitian X (2017) Twin support vector regression with Huber loss. J Intell Fuzzy Syst 32(6):4247–4258
48. NLREG repositories (2020) [online]. http://www.nlreg.com/

49. Ouyang X, Zhao N, Gao C, Wang L (2019) An efficient twin projection support vector machine for regression. Eng Lett 27(1):103–107
50. Peng X (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23(3):365–372
51. Peng X, Chen D (2019) An $$ l_1 $$-norm loss based twin support vector regression and its geometric extension. Int J Mach Learn Cybernet 10(9):2573–2588
52. Puthiyottil A, Balasundaram S, Meena Y (2020) "L1-norm support vector regression in primal based on huber loss function. In: Proceedings of ICETIT 2019, vol 605. LNEE. Springer, Cham, pp 195–205
53. SantaFeA dataset (2020) [online]. http://lib.stat.cmu.edu/datasets
54. Shen X, Niu L, Qi Z, Tian Y (2017) Support vector machine classifier with truncated pinball loss. Pattern Recognit 68:199–210
55. Shao YH, Zhang C, Wang X, Deng N (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6):962–968
56. Singla M, Ghosh D, Shukla KK, Pedrycz W (2020) "Robust twin support vector regression based on rescaled hinge loss. Pattern Recognit 105:107395
57. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
58. SpaceGa dataset (2020).[online]. http://lib.stat.cmu.edu/datasets
59. Tang L, Tian Y, Yang C, Pardalos PM (2018) Ramp-loss nonparallel support vector regression: robust, sparse and scalable approximation. Knowl-Based Syst 147:55–67
60. Tang L, Tian Y, Pardalos PM (2019) A novel perspective on multiclass classification: regular simplex support vector machine. Inf Sci 480:324–338
61. Tang L, Tian Y, Li W, Pardalos PM (2020) Structural improved regular simplex support vector machine for multiclass classification. Appl Soft Comput 91:106235
62. Tanveer M, Shubham K, Aldhaifallah M, Ho SS (2016) An efficient regularized K-nearest neighbor based weighted twin support vector regression. Knowl-Based Syst 94:70–87
63. UCI data repository (2020) [online]. https://archive.ics.uci.edu/ml/
64. Vapnik V (2000) The nature of statistical learning theory. Springer, Berlin
65. Vineyard dataset (2020) [online]. https://data.gov.au/dataset/
66. Wang L, Gao C, Zhao N, Chen X (2020) Wavelet transform-based weighted $$\nu $$ ν-twin support vector regression. Int J Mach Learn Cybernet 11(1):95–110
67. Wang L, Gao C, Zhao N, Chen X (2019) A projection wavelet weighted twin support vector regression and its primal solution. Appl Intell 49(8):3061–3081
68. Wang K, Pei H, Ding X, Zhong P (2019a) Robust proximal support vector regression based on maximum correntropy criterion. Sci Progr 2019:1–11
69. Wang C, Li Z, Dey N, Li Z, Ashour AS, Fong SJ, Simon Sherratt R, Wu L, Shi F (2018) Histogram of oriented gradient based plantar pressure image feature extraction and classification employing fuzzy support vector machine. J Med Imaging Health Inf 8(4):842–854
70. Wang K, Zhong P (2014) Robust support vector regression with flexible loss function. Int J Signal Process Image Process Pattern Recognit 7(4):211–220
71. Wu Q (2010) A hybrid-forecasting model based on Gaussian support vector machine and chaotic particle swarm optimization. Expert Syst Appl 37(3):2388–2394
72. Wu Q, Yan H (2009) Product sales forecasting model based on robust ν-support vector machine. Comput Integrated Manuf Syst 15(6):1081–1087
73. Wu Q, Law R, Xin X (2012) A sparse Gaussian process regression model for tourism demand forecasting in Hong Kong. Expert Syst Appl 39(5):4769–4774
74. Xu Q, Zhang J, Jiang C, Huang X, He Y (2015) Weighted quantile regression via support vector machine. Expert Syst Appl 42(13):5441–5451
75. Xu Y, Wang L (2014) K-nearest neighbor-based weighted twin support vector regression. Appl Intell 41(1):299–309
76. Xu Y, Li X, Pan X, Yang Z (2017) Asymmetric ν-twin support vector regression. Neural Comput Appl 30:1–16
77. Yang L, Ding G, Yuan C, Zhang M (2020) Robust regression framework with asymmetrically analogous to correntropy-induced loss. Knowl-Based Syst 191:105211
78. Yang L, Dong H (2018) Support vector machine with truncated pinball loss and its application in pattern recognition. Chemometr Intell Lab Syst 177:89–99
79. Yang, Z, Xu Y (2018) A safe sample screening rule for Laplacian twin parametric-margin support vector machine. Pattern Recognit 84:1–12
80. Yang L, Ren Z, Wang Y, Dong H (2017) A robust regression framework with laplace kernel-induced loss. Neural Comput 29(11):3014–3039
81. Ye Y, Gao J, Shao Y, Li C, Jin Y, Hua X (2020) Robust support vector regression with generic quadratic nonconvex ε-insensitive loss. Appl Math Model 82:235–251

82. Zhang J, Zheng C-H, Xia Y, Wang B, Chen P (2017) Optimization enhanced genetic algorithm-support vector regression for the prediction of compound retention indices in gas chromatography. Neurocomputing 240:183–190
83. Zhao Y, Sun J (2008) Robust support vector regression in the primal. Neural Netw 21(10):1548–1555
84. Zhu J, Hoi SCH, Rung-Tsong Lyu M (2008) Robust regularized kernel regression. IEEE Trans Syst Man Cybern Part B (Cybern) 38(6):1639–1644