symmetry

MDPI

Article

Computing Expectiles Using *k*-Nearest Neighbours Approach

Muhammad Farooq [1], Sehrish Sarfraz [2], Christophe Chesneau [3], Mahmood Ul Hassan [4,*], Muhammad Ali Raza [5], Rehan Ahmad Khan Sherwani [6] and Farrukh Jamal [7]

1 Department of Statistics, GC University Lahore, Lahore 54000, Pakistan; muhammad.farooq@gcu.edu.pk
2 Department of Statistics, University of Gujrat, Gujrat 50700, Pakistan; sehrishsarfraz123@gmail.com
3 Department of Mathematics, Université de Caen, LMNO, Campus II, Science 3, 14032 Caen, France; christophe.chesneau@unicaen.fr
4 Department of Statistics, Stockholm University, SE-106 91 Stockholm, Sweden
5 Department of Statistics, GC University Faisalabad, Faisalabad 38000, Pakistan; ali.raza@gcuf.edu.pk
6 College of Statistical and Actuarial Sciences, University of the Punjab, Lahore 54000, Pakistan; rehan.stat@pu.edu.pk
7 Department of Statistics, The Islamia University of Bahawalpur, Bahawalpur 61300, Pakistan; farrukh.jamal@iub.edu.pk
* Correspondence: mahmood.ul-hassan@stat.su.se

Abstract: Expectiles have gained considerable attention in recent years due to wide applications in many areas. In this study, the *k*-nearest neighbours approach, together with the asymmetric least squares loss function, called ex-*kNN*, is proposed for computing expectiles. Firstly, the effect of various distance measures on ex-*kNN* in terms of test error and computational time is evaluated. It is found that Canberra, Lorentzian, and Soergel distance measures lead to minimum test error, whereas Euclidean, Canberra, and Average of (L_1, L_∞) lead to a low computational cost. Secondly, the performance of ex-*kNN* is compared with existing packages er-boost and ex-svm for computing expectiles that are based on nine real life examples. Depending on the nature of data, the ex-*kNN* showed two to 10 times better performance than er-boost and comparable performance with ex-svm regarding test error. Computationally, the ex-*kNN* is found two to five times faster than ex-svm and much faster than er-boost, particularly, in the case of high dimensional data.

Keywords: asymmetric least squares loss function; *k*-nearest neighbours approach; expectiles; machine learning; high dimensional data

Citation: Farooq, M.; Sarfraz, S.; Chesneau, C.; Ul Hassan, M.; Raza, M.A.; Sherwani, R.A.K.; Jamal, F. Computing Expectiles Using *k*-Nearest Neighbours Approach. *Symmetry* **2021**, *13*, 645. https://doi.org/10.3390/sym13040645

Academic Editor: José Carlos R. Alcantud

Received: 27 February 2021
Accepted: 8 April 2021
Published: 11 April 2021

1. Introduction

Given independent data $D_n := ((x_1, y_1), \ldots, (x_n, y_n))$ that were drawn from unknown probability distribution P on $X \times Y$, where $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$, the symmetric loss functions, such as least absolute deviation loss or least squares loss lead to study the center of the conditional distribution $\mathrm{P}(Y|X = x)$ by estimating the conditional median $\mathrm{med}(Y|X = x)$ or the conditional mean $\mathbb{E}(Y|X = x)$, respectively. To investigate $\mathrm{P}(\cdot|x)$ beyond the center, one well-known approach is computing quantiles that were proposed by Koenker and Bassett Jr. [1]. If $\mathrm{P}(\cdot|x)$ has strictly positive Lebesgue density, the conditional τ-quantile q_τ, $\tau \in (0, 1)$ of Y given $x \in X$ is the solution of

$$\mathrm{P}(Y \leq q_\tau | X = x) = \tau\,.$$

Another approach is computing expectiles proposed by Newey and Powell [2] and it has gained considerable attention recently. Assume that $Q := \mathrm{P}(Y|x)$ such that $|Q|_1 := \int_Y y\, dQ(y) < \infty$, then the conditional τ-expectile μ_τ for each $\tau \in (0, 1)$ is the *unique* solution of

$$\tau \int_{\mu_\tau}^{\infty} (y - \mu_\tau)\, dQ(y) = (1 - \tau) \int_{-\infty}^{\mu_\tau} (\mu_\tau - y)\, dQ(y)\,.$$

It is well-known that quantiles or expectiles can be computed algorithmically, see, for example, Efron [3] and Abdous and Remillard [4]. To be more precise, for all $t \in \mathbb{R}$ and a fixed $\tau \in (0,1)$, one needs to solve the optimization problem

$$\eta_\tau := \arg\min_{t\in\mathbb{R}} \mathbb{E}_{y\sim Q} L_\tau(y,t)\,, \tag{1}$$

where L_τ is the asymmetric loss function, which, for $p \geq 1$, is defined by

$$L_\tau(y,t) = \begin{cases} (1-\tau)|y-t|^p\,, & \text{if} \quad y < t\,, \\ \tau|y-t|^p\,, & \text{if} \quad y \geqslant t\,. \end{cases} \tag{2}$$

For $p = 1$, we reach the asymmetric least absolute deviation (ALAD) loss function from (2) and, consequently, conditional quantiles $q_\tau = \eta_\tau$ from (1). Analogously, for $p = 2$, L_τ from (2) is the asymmetric least squares (ALS) loss and, thus, (1) produces conditional expectiles $\mu_\tau = \eta_\tau$.

In general, for fixed τ, expectiles do not coincide with quantiles. The decision of computing either expectiles or quantiles depends on the applications at hand. For example, if one is interested to compute the threshold below which τ-fraction of observations lies, then τ-quantile is the right choice. On the other hand, to compute the thresholds, such that the ratio of gain (average deviations above threshold) and loss (average deviations below threshold) is equal to k where $k := \frac{1-\tau}{\tau}$, then τ-expectile is the right choice. Broadly speaking, quantiles are the tail probabilities, while expectiles are considered tail expectations. This distinction of expectiles from quantiles made expectiles applicable in the fields of, for example, demography [5], education [6], and extensively in finance (see, e.g., Wang et al. [7] and Kim and Lee [8]). In fact, expectiles are the only risk measures that satisfy the well-known properties of coherence and elicitability (see, e.g., Bellini et al. [9]), and they are are proved to be better alternative to quantile-based value-at-risk (VaR). Furthermore, one can immediately realized the well-known performance measure in portfolio management known as *gain-loss ratio* or Ω-*ratio* by τ-expectile for any $\tau \in (0,1)$, see Keating and Shadwick [10] for more details.

Different semiparametric and non-parametric approaches have been proposed in the literature for estimating expectiles. For example, Schnabel and Eilers [11] proposed an algorithm using P-splines, which is found to be difficult to implement on the problems involving multiple predictors. Another algorithm that is based on gradient boosting is proposed by [12] and Yang and Zou [13]. It is observed by Farooq and Steinwart [14] through experiments that boosting-based algorithm becomes computationally expensive when the dimensions of input space increases. Recently, Farooq and Steinwart [14] developed an SVM-like solver while considering sequential minimal optimization. Although the solver is found to be efficient when compared with an R-package er-boost developed by Yang and Zou [13], but SVM-based solver is found time-sensitive to the training set size.

It is important to note that the aforementioned algorithms are required to select an appropriate nonlinear function. To estimate conditional expectiles directly, Yao and Tong [15] used the kernelized iteratively reweighted approach, where kernel is used to assign higher weights to the points that are closer to the query point. However, choosing the right kernel for a problem at hand is tricky. Moreover, this method leads to the curse of dimensionality issue. Another simple, yet very popular, approach for estimating conditional quantities is the k-nearest neighbours that has competed with the advanced complex machine learning approaches when it comes to dealing with complex problems. Since the introduction of kNN in 1967, it has been applied in many applications regarding classification, regression, and missing value imputation, such as pattern recognition, economic forecasting, data compression, outlier detection, and genetics, etc., see [16]. One key advantage of the kNN approach is that it does not require a smoothness assumption of functions, which is, in general, necessary in advanced techniques of classification and regression. With these advantages, we use kNN together with the ALS loss function to compute expectiles and named the algorithm ex-kNN. We then compare the ex-kNN with and R-package, called

er-boost proposed by Yang and Zou [13], and show, numerically, that ex-kNN not only better performs in terms of accuracy and computational cost, but is also less sensitive to the dimensions of input space as compared to er-boost. In addition, we show that the performance of ex-kNN is comparable with an SVM-like solver ex-svm proposed by [14]. Moreover, ex-kNN is found to be less sensitive to the training set size as compared to ex-svm.

This paper is organized, as follows: Section 2 gives a brief overview of the kNN approach in the context of computing expectiles. This also includes a concise introduction of different distance measures (Section 2.1) and the procedure of choosing the best value for k (Section 2.2)—two aspects of kNN that greatly influence its performance. Section 3 covers the experiments while considering nine real life datasets in order to evaluate the effect of various distance measures on the performance of ex-kNN as well as comparing the results with existing packages for computing expectiles, like er-boost and ex-svm. Section 4 provides the concluding remarks.

2. K-Nearest Neighbours Expectile Regression

The kNN approach to computing expectiles is explained in this section. Because the performance of kNN depends on the selection of suitable distance measures and the best value of k-neighbours, a detailed discussion on these two aspects is given in Sections 2.1 and 2.2, respectively. To this end, let $\mathbf{x}_q \in X$ be a query point and $R := \{\mathbf{x}_1, \ldots, \mathbf{x}_r\} \in X$ be a set of reference points. Subsequently, the kNN approach searches the k nearest neighbours for query point $\mathbf{x}_q$ in the reference set R based on a specified distance measure, and then uses the set $\{y_1, \ldots, y_k\}$ that corresponds to the k nearest neighbours for classification or regression.

Once the set $\{y_1, \ldots, y_k\}$ corresponding to k-nearest neighbours is achieved, expectiles can be computed by solving the empirical version of (1), which is, for $p = 2$, we solve the optimization problem

$$\hat{\mu}_\tau := \arg\min_{t \in \mathbb{R}} \sum_{i=1}^{k} L_\tau(y_i, t_i)\,. \tag{3}$$

Because the loss function used in (3) is quadratic in nature, the problem (3) can be solved by using iterative reweighted least squares (IRLS) algorithm. To be more precise, we assume an initial estimate of the expectile e_τ and generate a sequence of weights τ (when $y_i < e_\tau$) and $1 - \tau$ (when $y_i \geq e_\tau$) for all $i = 1, 2, \ldots, k$. Subsequently, we update the estimate of e_τ by

$$e_\tau := \frac{(1-\tau)\left(\sum_{i=1}^{k} y_i | y < e_\tau\right) + \tau\left(\sum_{i=1}^{k} y_i | y \geq e_\tau\right)}{\sum_{i=1}^{k}(1-\tau)|y < e_\tau + \sum_{i=1}^{k} \tau | y < e_\tau}\,. \tag{4}$$

and, hence, the corresponding weights, repeatedly, until convergence is achieved. Note that one may initialize e_τ by the average value of $\{y_1, \ldots, y_k\}$. Following [11], Procedure 1 presents the pseudo code for computing expectiles using IRLS.

Procedure 1 Computing Expectiles

Input: response variable y, $\tau \in (0, 1)$, stopping criteria $T \leftarrow T_0$
Initialize: $e_\tau^* \leftarrow \frac{1}{k}\sum_{i=1}^{k} y_i$, $g \leftarrow g_0$
while $(g < T)$ **do**
  $w \leftarrow \{(1-\tau)|y_i < e_\tau^*\} \cup \{\tau | y_i \geq e_\tau^*\}$ for all $i = 1 = 1, 2, \ldots, k$
  $e_\tau \leftarrow \frac{1}{\sum_{i=1}^{k} w_i} \sum_{i=1}^{k} w_i\, y_i$
  $g \leftarrow |e_\tau - e_\tau^*|$
  $e_\tau^* \leftarrow e_\tau$
end while
return e_τ^*

Here, g is the margin of error achieved in an iteration and T is the stopping criterion. Fast convergence can be achieved by making good choice of T and initialization of e_τ^*.

Moreover, in the case when $\tau = 0$ or $\tau = 1$, the τ-expectile can be considered to be the minimum or maximum value of the set $\{y_1, \ldots, y_k\}$, respectively.

2.1. Distance Measures

As mentioned above, one of the aspects that influences the performance of kNN algorithm is the distance measure that is used to identify the closest neighbours in the training data. The Euclidean is a commonly used distance measure while implementing kNN. However, to our knowledge, there is no study indicating that it is the best suitable choice with kNN in all cases. Therefore, several studies have been dedicated to exploring the suitable distance measure with kNN for the given problem at hand. For instance, Mulak and Talhar [17] evaluated the performance of kNN with four distance measures on the KDD data set and found that the Manhattan distance measure is the best in terms of classification accuracy. Lopes and Ribeiro [18] investigated the impact of five distance measures, such as Euclidean, Manhattan, Canberra, Chebychev, and Minkowsky, for various small datasets and found that Euclidean and Manhattan perform better for most of the datasets. Extending the investigation, Kittipong et al. [19] investigated the performance of kNN with eleven distance measures and determined that Manhattan, Minkowsky, and Chebychev lead to better performance. Analogously, Todeschini et al. [20] considered eighteen distance measures and sorted out the best five distance measures leading to better performance.

A more detailed investigation in this regard has been done by [21], where they have considered 54 different distance measures from seven families of distances and shown that no single distance metric can be considered to be good enough for all types of datasets. In other words, the choice of the distance measures with kNN depends on many factors, such as nature of input variables, number of dimensions, and size of datasets. This raises the need of also considering different distance measures in our study for computing expectiles using kNN and to identify the one that leads to the best performance in our case for a specified dataset. For this purpose, we consider a set of distance measures that have been found to be the best in the aforementioned studies for various datasets. This set includes the Euclidean distance, Manhattan distance, Chebychev distance, Canberra distance, Soergel distance, Lorentzian distance, Cosine distance, Contracted JT distance, Clark distance Squared Chi-Squared distance, Average (L_1, L_∞) distance, Divergence distance, Hassanat distance, and Whittaker's Index association. A brief description of these distance measures is given in the following. We refer to [21] for more details on these and other lists of distance measures. To this end, we assume that $\mathbf{x}_i = \{x_{i1}, \ldots, x_{ip}\}$ and $\mathbf{x}_j = \{x_{j1}, \ldots, x_{jp}\}$ are the p-dimensional ith and jth data points in X^p.

- *Euclidean Distance* (ED) is also called L_2 norm or Ruler distance and defined by

$$d_E(\mathbf{x}_i, \mathbf{x}_j) := \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2}\,. \tag{5}$$

- *Manhattan Distance* (MD) is also known as L_1 distance, and defined as the sum of absolute difference of elements of $\mathbf{x}_i$ and $\mathbf{x}_j$ for $i = j = 1, 2, \ldots, p$, which is,

$$d_{MD}(\mathbf{x}_i, \mathbf{x}_j) := \sum_{l=1}^{p} |x_{il} - x_{jl}|\,. \tag{6}$$

- *Chebychev Distance* (CbD) is the maximum value distance and specified by

$$d_{CbD}(\mathbf{x}_i, \mathbf{x}_j) := \max_l |x_{il} - x_{jl}|. \tag{7}$$

- *Canberra Distance* (CD) is a weighted version of Manhattan distance measure and defined by

$$d_{CD}(\mathbf{x}_i, \mathbf{x}_j) := \sum_{l=1}^{p} \frac{|x_{il} - x_{jl}|}{|x_{il}| + |x_{jl}|}. \tag{8}$$

Note that (8) is sensitive to small changes when both $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to zero.

- Soergel Distance (SoD) is widely used for calculating the evolutionary distance and obeying all four properties of a valid distance measure. It is listed by

$$d_{SoD}(\mathbf{x}_i, \mathbf{x}_j) := \frac{\sum_{l=1}^{p} |x_{il} - x_{jl}|}{\sum_{l=1}^{p} (x_{il} + x_{jl})}. \tag{9}$$

- *Lorentzain Distance* (LD) is defined as a natural log of absolute distance between vector $\mathbf{x}_i$ and $\mathbf{x}_j$, that is

$$d_{LD}(\mathbf{x}_i, \mathbf{x}_j) := \sum_{l=1}^{p} \ln(1 + |x_{il} - x_{jl}|), \tag{10}$$

where one is added to avoid the log of zero and to ensure non-negative property of a distance metric.

- *Cosine Distance* is derived from a cosine similarity that measures the angle between two vectors. It is specified by

$$d_{CosD}(\mathbf{x}_i, \mathbf{x}_j) := 1 - \frac{\sum_{l=1}^{p} x_{il}\, y_{jl}}{\sqrt{\sum_{l=1}^{p} x_{il}^2 \sum_{l=1}^{p} x_{jl}^2}}. \tag{11}$$

- *Jaccard Distance* (JacD) measures dissimilarity between two vectors. It is defined by

$$d_{JacD}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{l=1}^{p} (x_{il} - xjl)^2}{\sum_{l=1}^{p} x_{il}^2 + \sum_{l=1}^{p} x_{jl}^2 - \sum_{l=1}^{p} x_{il} x_{jl}}. \tag{12}$$

- *Clark Distance* is also called the coefficient of divergence. It is the square root of half of divergence distance. It is defined by

$$d_{DivD}(\mathbf{x}_i, \mathbf{x}_j) := \sqrt{\sum_{l=1}^{p} \frac{(x_{il} - x_{jl})^2}{(x_{il} + x_{jl})^2}}. \tag{13}$$

- *Squared Chi-Squared Distance* belongs to the family of L_2 and it is defined by

$$d_{SCSD}(\mathbf{x}_i, \mathbf{x}_j) := \sum_{l=1}^{p} \frac{(x_{il} - x_{jl})^2}{|x_{il} + x_{jl}|}. \tag{14}$$

- *Average* (L_1, L_∞) *Distance* is the average of Manhattan distance and Chebyshev distance. It is defined by

$$d_{AvgD}(\mathbf{x}_i, \mathbf{x}_j) := \frac{\sum_{l=1}^{p} |x_{il} - x_{jl}| + \max|x_{il} - x_{jl}|}{2}. \tag{15}$$

- *Divergence Distance* is defined by

$$d_{ClaD}(\mathbf{x}_i, \mathbf{x}_j) := 2 \sum_{l=1}^{p} \frac{(x_{il} - x_{jl})^2}{(x_{il} + x_{jl})^2}. \tag{16}$$

- *Hassanat Distance* (HasD) is defined by

$$d_{Has}(\mathbf{x}_i, \mathbf{x}_j) := \sum_{l=1}^{p} D(x_{il}, x_{jl}), \tag{17}$$

where for $l = 1, 2, \ldots, p$

$$D(x_{il}, x_{jl}) := \begin{cases} 1 - \dfrac{1 - \min(x_{il}, x_{jl})}{\max(x_{il}, x_{jl})}, & \text{if} \quad \min(x_{il}, x_{jl}) \geq 0, \\ 1 - \dfrac{1 - \min(x_{il}, x_{jl}) + |\min(x_{il}, x_{jl})|}{\max(x_{il}, x_{jl}) + |\min(x_{il}, x_{jl})|}, & \text{if} \quad \min(x_{il}, x_{jl}) < 0. \end{cases}$$

It is important to note that (17) is invariant to different scale, noise, and outlier, and it is always bounded by $[0, 1[$. Contrary to other distances measures, $HasD(.,.)$ shows high similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$ when it approaches zero where as it shows high dissimilarity when it approaches one. Moreover, it can only achieve limiting value one if $\max(x_{il}, x_{jl}) \to +\infty$ or $\min(x_{il}, x_{jl}) \to -\infty$.

- *Whittaker's Index of Association Distance*

$$d_{WIAD}(\mathbf{x}_i, \mathbf{x}_j) := \frac{1}{2} \sum_{l=1}^{p} \left| \frac{x_{il}}{\sum_{l=1}^{p} x_{il}} - \frac{x_{jl}}{\sum_{l=1}^{p} x_{jl}} \right|. \tag{18}$$

2.2. Selecting Best k

The selection of an appropriate value for k plays a key role in determining the performance of kNN. A lot of work has been done so far in the context of classification, regression, and missing data imputation in order to deal with this issue. For example, Lall and Sharma [22] suggested choosing the fixed optimal $k = \sqrt{n}$ for training a dataset when the sample size $n > 100$. However, this approach has been criticised due to lack of theoretical guarantee. Many more advanced approaches to determine k-value have been proposed. For example, the kTree method is used to learn different k-value for different test sample [23], sparse-based kNN method [24], and using reconstruction framework [25]. For more details, we refer the readers to [23], and the references therein.

The cross-validation is one of the approaches that has gained popularity in machine learning applications to tune the hyperparameters and that has also been considered in kNN. Recall that the cross-validation splits the data into two folds, where one fold is used to train the model by learning suitable values for the hyperparameters and the other fold is used to validate the model. The m-fold cross-validation method extends this approach by randomly dividing data into m equally (or nearly equally) folds. In other words, the process of cross-validation is repeated m times, such that in the each iteration a different fold is held-out for model validation and the remaining m-1 folds are used to learn the hyperparameters.

Procedure 2 Computing Best k

Input: $\tau \in (0, 1)$, k_{max}, data
split data into m-folds
for $i = 1$ to k_{max} **do**
  for $j = 1$ to m **do**
    Obtain $k = i$ neighbours against each query point of j-th fold using distance measure $d(\cdot, \cdot)$
    Compute expectiles from $k = i$ neighbours using Procedure 1
    Compute error for j-th fold using L_τ
  end for
  For each $k = i$ choose minimum error from all folds
end for
Choose k with minimum cross-validation error
return Best k

Note that the algorithms, such as er-boost and ex-svm, use cross-validation approach to tune the hyperparameters. We consider the same approach in this study to select the

best value of k, despite the existence of advanced strategies so that a fair comparison of ex-kNN with these algorithms can be made in terms of training time. Procedure 2 provides the pseudo code of selecting best k value by cross-validation.

2.3. The ex-kNN Algorithm

After considering a distance measures from the list that is given in Section 2.1, the best k-value for the training part of the data is attained using cross-validation approach following Procedure 2. The selected best k-value is used in testing part of data to compute the test error. The whole procedure of implementing the ex-kNN is described in the following Algorithm 1.

Algorithm 1 Compute test error

Training Phase:
Input: $\tau \in (0,1)$, k_{max}, train data
split data into m-folds
Compute best k-value using Procedure 2

Testing Phase:
Input: $\tau \in (0,1)$, best k, test data
Compute expectiles using Procedure 1
Compute test error using L_τ

3. Experimental Results

In this section, we conduct experiments using different datasets to compare the performance of ex-kNN with the existing algorithms of computing expectiles, like er-boost and ex-svm, in terms of training time and test error. All of the experiments have been performed on INTEL CORE i3-4010U (1.70 GHz) 4 GB RAM system under 64 bit version of WINDOWS 8. The run time during experiments is computed by using single core, whereas the running of other processes were minimized.

Recall that [14] have considered nine datasets in their study to compare the performance of ex-svm with er-boost. To make a fair comparison of ex-kNN with ex-svm and er-boost, we have downloaded the same datasets following the details that are given by [14]. That is, the data sets CONCRETE-COMP, UPDRS-MOTOR, CYCLE-PP, AIRFOIL-NOISE, and HOUR have been downloaded from UCI repository. Three data sets—NC-CRIME, HEAD-CIRCUM, and CAL-HOUSING—were extracted from R packages Ecdat, AGD and StatLib repository, respectively. Finally, one data set MUNICH-RENT was downloaded from the data archive of Institute of Statistics, Ludwig-Maximilians-University. These datasets were scaled componentwise, such that all of the variables, including the response variable, lie in $[-1,1]^{d+1}$, where d denotes the dimension of the data. Table 1 describes the characteristics of the considered data sets. All of the datasets were randomly divided into training and testing samples comprising 70% and 30%, respectively. The training sample is further divided into m-folds with randomly generated folds to implement cross-validation approach for determining the best k-value.

Table 1. Characteristics of data sets together with the training sizes and the test sizes that refer to the splits used in the run time experiments.

Data	Sample Sizes	Training Size	Test Size	Features Categorical	Features Continuous	Total
NC-CRIME	630	441	189	3	16	19
CONCRETE-COMP	1030	721	309	0	8	8
AIRFOIL-NOISE	1503	1052	451	1	4	5
MUNICH-RENT	2053	1437	616	8	4	12
UPDRS-MOTOR	5875	4112	1763	1	18	19
HEAD-CIRCUM	7020	4914	2106	0	4	5
CYCLE-PP	9568	6697	2871	0	5	5
HOUR	17,379	12,165	5214	7	5	12
CAL-HOUSING	20,639	14,447	6192	0	8	8

It is important to note that the algorithms er-boost and ex-svm are implemented in C^{++} to gain the computational advantages. For fair comparison regarding computational cost, we have implemented ex-kNN in the R 3.6.1 package using the libraries Rcpp [26] and ArmadilloRcpp [27]. The library Rcpp provides seamless integration of R and C^{++}, whereas ArmadilloRcpp is the templated C^{++} linear algebra library. The use of these libraries makes the implementation of ex-kNN algorithm close to the implementation of er-boost and ex-svm algorithms.

Firstly, we evaluate the effect of different distance measures that were considered in Section 2.1 on the performance of ex-kNN for computing expectiles. Note that the performance is measured in terms of test error and computational time, and the distance measure that leads to the minimum on these two evaluation factors is considered to be the best distance measure. In this context, for $\tau = 0.25, 0.50, 0.75$, the test error and computational time of ex-kNN is computed and presented in Tables A1–A6 in Appendix A. By giving rank to the test error and computation time of ex-kNN for each distance measure in each dataset, we observe that there is no single distance measure that performs well on all data sets in terms of test error and computational cost. For instance, by looking in Table A1 of test error for $\tau = 0.25$, we see that, for the dataset HOUR, the Euclidean distance provides the minimum test error, whereas the same distance measure behaves entirely opposite for dataset MUNICH-RENT. This leads us to conclude that the distance measure that plays vital role on the performance of ex-kNN depends on the characteristics of datasets. This observation is also noted by [21] in their investigation. Therefore, one need to consider the nature of datasets while choosing a distance measure for kNN-type methods.

In order to determine the overall performance of a distance measure on all datasets, we have computed the average of ranks assigned to individual distance measures on different datasets, see Tables A1–A6 in Appendix A. Clearly, the distance measures Canberra, Lorentzian, and Soergel can be labelled as the best three distance measures when the goal is to achieve the high accuracy of the results. However, on the other hand, when the objective is to attain a low computational cost of ex-kNN, the Euclidean, Canberra, and Average (L_1, L_∞) distance measures are ranked as the top three. Furthermore, these best three distance measures regarding test error and computational cost does not hold the said order for $\tau = 0.25, 0.5$ and 0.75. This indicates that the distance measure for the same data set behaves differently for different τ-level. To be more elaborative, we see, in Tables A2 and A3, that the Canberra distance measure that holds top position in providing minimum distance measure in most of the datasets when computing expectiles for $\tau = 0.50, 0.75$ attains third position for $\tau = 0.25$. Similar is the situation with other distance measures. It is interesting to note from the results of our experiments based on the considered datasets that no single distance measure can lead the ex-kNN towards achieving both goals of high accuracy and low computational cost at the same time. In other words, the choice of a distance measures with ex-kNN not only depends on the data set, but also the objective. It is also important to note that the Euclidean distance that has been considered more often with kNN in the literature shows poor performance in general when the goal is to achieve highly accurate predictions.

Finally, we compare the performance of ex-kNN with the existing packages of computing expectiles, like er-boost and ex-svm, with respect to the test error and computational cost. To perform experiments by er-boost, we set the default value of boosting steps ($M = 10$) and use 5-fold cross-validation to choose the best value of interaction (L) between variables. For more details regarding experimental setting with er-boost, we refer the interested readers to [28]. To perform the ex-svm, which is the part of package liquidSVM, we downloaded the terminal version of liquidSVM for Windows 64 bit. The default setting of 10 by 10 grid search of hyperparameters together with 5-fold cross validation is used to tune these hyperparameters. For more details on ex-svm and liquidSVM, we refer the readers to [14,29], respectively. Finally, to compute the test error and computation cost for ex-kNN we consider the distance measures that attain the top average rank for τ-expectile, which is, Canberra distance for test error and Euclidean distance for computational time.

Furthermore, we also use five-fold cross-validation to determine the best k-value. Based on the aforementioned settings, the results for test error and computational cost of ex-kNN, er-boost, and ex-svm for different datasets are presented in Tables 2 and 3. By comparing ex-kNN with er-boost regarding test error at $\tau = 0.25, 0.50$, and 0.75, we see that ex-kNN, depending on the nature of the data, shows between two to eight times better performance, see Figure 1. On the other hand, the performance of ex-kNN in terms of test error is comparable with ex-svm on some examples. Regarding computational cost, similar to the findings of [14] for ex-svm, we observe that ex-kNN is also more sensitive to the training set size and less sensitive to data dimensions. However, it is interesting to note that ex-kNN, based on the datasets, is up to five times more efficient on some examples than ex-svm. Moreover, ex-kNN is found to be considerably time efficient on the datasets, particularly when the data sets are high dimensional, as we see in Figure 1.

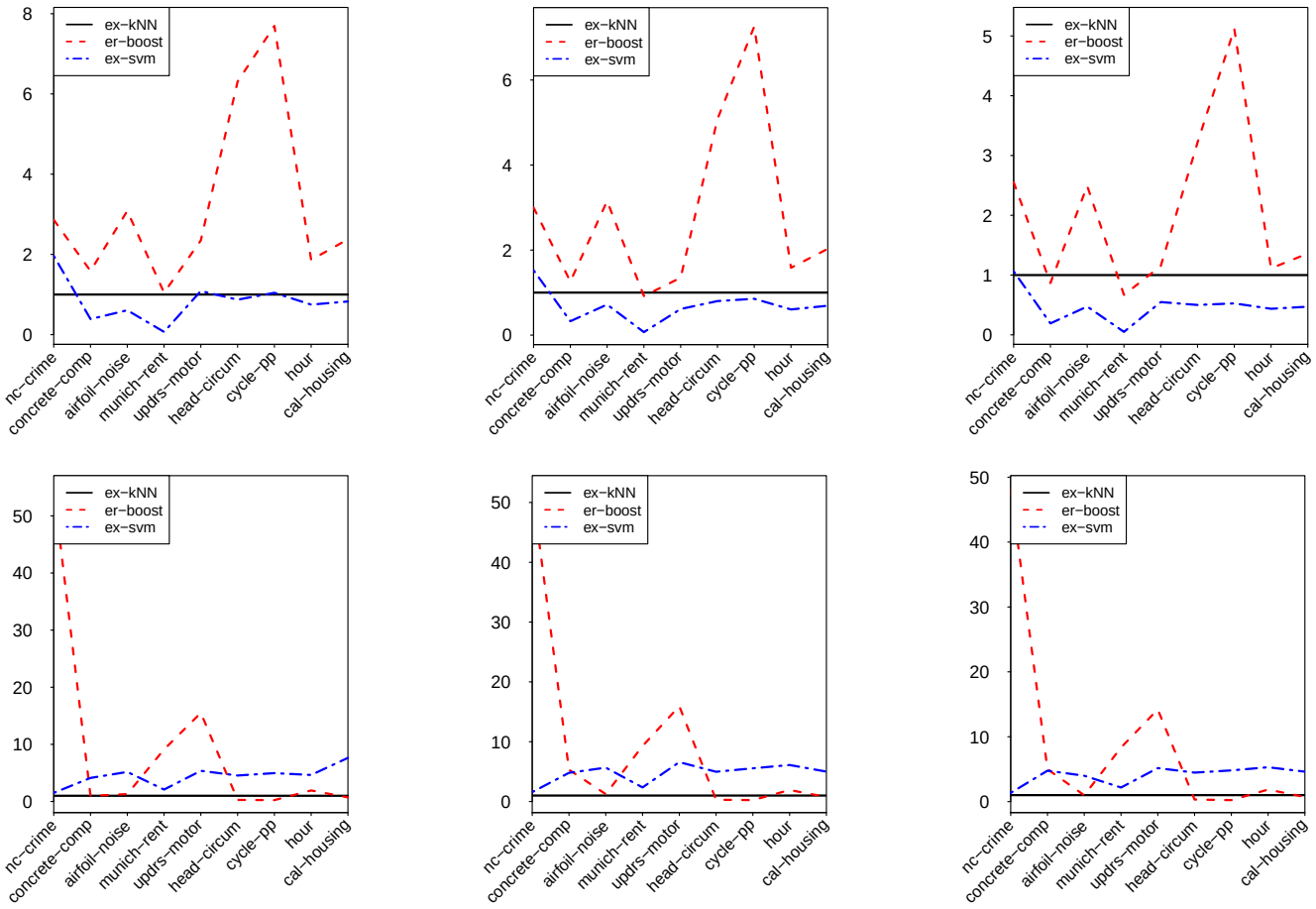

Figure 1. Test error ratio of er-boost and ex-svm to ex-kNN(**top**) and computational time ratio of er-boost and ex-svm to ex-kNN (**bottom**). The graphs comprise of $\tau = 0.25$ (**left**), $\tau = 0.50$ (**middle**) and $\tau = 0.75$ (**right**).

Table 2. Test error obtained by performing ex-kNN, er-boost, and ex-svm while computing expectiles for $\tau = 0.25, 0.50, 0.75$.

Data Set	$\tau = 0.25$			$\tau = 0.50$			$\tau = 0.75$		
	ex-kNN	**er-boost**	**ex-svm**	**ex-kNN**	**er-boost**	**ex-svm**	**ex-kNN**	**er-boost**	**ex-svm**
NC-CRIME	0.0029	0.0083	0.0057	0.0040	0.0120	0.0061	0.0047	0.0120	0.0050
CONCRETE-COMP	0.0250	0.0397	0.0097	0.0389	0.0490	0.0127	0.0522	0.0452	0.0100
AIRFOIL-NOISE	0.0130	0.0401	0.0079	0.0149	0.047	0.0107	0.0169	0.0462	0.0080
MUNICH-RENT	0.0131	0.0137	0.0009	0.0172	0.0158	0.0012	0.0202	0.0135	0.0010
UPDRS-MOTO	0.0229	0.0538	0.0249	0.0474	0.0640	0.0291	0.0474	0.0545	0.026
HEAD-CIRCUM	0.0039	0.0246	0.0034	0.00500	0.0254	0.0040	0.0060	0.0193	0.0030
CYCLE-PP	0.0046	0.0354	0.0048	0.0062	0.0450	0.0053	0.0076	0.0389	0.0040
HOUR	0.0155	0.0289	0.0116	0.0252	0.0399	0.0152	0.0344	0.0383	0.0150
CAL-HOUSING	0.0231	0.0549	0.0191	0.0375	0.0761	0.0259	0.0552	0.0751	0.0260

Table 3. Train time (in seconds) obtained by performing ex-kNN, er-boost and ex-svm while computing expectiles for $\tau = 0.25, 0.50, 0.75$.

Data Set	$\tau = 0.25$			$\tau = 0.50$			$\tau = 0.75$		
	ex-kNN	**er-boost**	**ex-svm**	**ex-kNN**	**er-boost**	**ex-svm**	**ex-kNN**	**er-boost**	**ex-svm**
NC-CRIME	0.545	29.790	0.900	0.571	29.819	0.830	0.617	29.730	0.820
CONCRETE-COMP	0.584	2.924	2.740	0.572	3.032	2.740	0.575	2.867	2.410
AIRFOIL-NOISE	0.132	1.091	5.060	0.893	1.105	4.510	1.130	1.072	4.510
MUNICH-RENT	1.619	14.760	3.82	1.619	15.026	4.070	1.872	15.604	3.380
UPDRS-MOTO	10.524	163.30	6.91	10.495	167.24	59.52	11.516	163.63	56.49
HEAD-CIRCUM	9.064	2.512	44.320	8.908	2.664	39.43	8.821	2.819	41.20
CYCLE-PP	16.163	3.833	88.670	15.961	3.728	77.06	16.018	3.680	80.250
HOUR	65.435	126.30	386.160	63.252	121.820	343.23	64.775	120.270	303.550
CAL-HOUSING	75.757	54.780	381.12	76.221	57.433	342.99	74.301	53.620	340.69

4. Conclusions

In this study, an algorithm, called ex-kNN, is proposed by combining the idea of the k-nearest neighbours approach and the asymmetric least squares loss function in order to compute expectiles. Because the performance of ex-kNN depends on the distance measure used to determine the neighbourhood of the query point, various distance measures are considered and their impact is evaluated in terms of test error and computational time. It is observed that there exists no single distance measures that can be associated with ex-kNN to achieve high performance for different kinds of datasets. To be more precise, it is found from the list of considered distance measures that Canberra, Lorentzian, and Soergel lead to minimum test error, whereas Euclidean, Canberra, and Average of (L_1, L_∞) provide a low computational cost of ex-kNN. Furthermore, using nine real-world datasets, the performance of ex-kNN is compared with existing packages for computing expectiles, namely er-boost and ex-svm. The results showed that the ex-kNN, depending on the nature of data, performs between two to eight times better than ex-kNN in terms of test error and it showed comparable performance with ex-svm on some datasets. Regarding computational cost, it is found that ex-svm is up to five times more efficient than ex-svmm and much more efficient than er-boost.

To make a fair comparison of ex-kNN with existing packages, this study is limited to using the cross-validation approach with ex-kNN to select the best value of k-neighbours. However, more advanced and efficient approaches for this purpose can be considered with ex-kNN for a further reduction of computational cost. Moreover, some other loss functions to compute expectiles can be investigated, with the results compared to the one considered in this study.

Author Contributions: Each author's contribution is equal. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request due to restrictions, e.g., privacy or ethical. The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors would like to thank the three referees for their constructive comments on the article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Impact of Different Distance Measures on Test Error and Computational Cost of ex-kNN

Table A1. Test error of ex-kNN for $\tau = 0.25$ under different distance measures for different datasets. The result in $(\cdot)$ indicates the rank of a distance measure for a specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.0047 (4)	0.0246 (3)	0.0125 (2)	0.0142 (8)	0.0277 (6)	0.0039 (2)	0.0047 (3)	0.0150 (1)	0.0237 (5)	3.78
Manhattan	0.0036 (3)	0.0257 (6)	0.0130 (5)	0.0135 (6)	0.0245 (4)	0.0039 (2)	0.0045 (1)	0.0157 (6)	0.0229 (1)	3.78
Chebychev	0.0066 (7)	0.0299 (8)	0.0121 (1)	0.0152 (9)	0.0347 (10)	0.0039 (2)	0.0050 (5)	0.0165 (7)	0.0250 (8)	6.33
Canberra	0.0034 (2)	0.0250 (6)	0.0130 (5)	0.0131 (4)	0.0229 (1)	0.0039 (2)	0.0046 (2)	0.0155 (4)	0.0231 (3)	3.22
Soergel	0.0034 (2)	0.0250 (6)	0.0130 (5)	0.0131 (4)	0.0229 (2)	0.0039 (2)	0.0046 (2)	0.0155 (4)	0.0231 (3)	3.33
Lorentzian	0.0029 (1)	0.0270 (7)	0.0129 (4)	0.0132 (5)	0.0236 (3)	0.0038 (1)	0.0045 (1)	0.0155 (4)	0.0230 (2)	3.11
Cosine	0.0060 (6)	0.0249 (5)	0.0175 (6)	0.0114 (1)	0.0323 (8)	0.0042 (3)	0.0069 (6)	0.0152 (3)	0.0260 (9)	5.22
Contracted JT	0.0059 (5)	0.0248 (4)	0.0121 (1)	0.0140 (7)	0.0287 (7)	0.0039 (2)	0.0048 (4)	0.0150 (1)	0.0237 (5)	4
Clark	0.0746 (8)	0.0396 (10)	0.0223 (9)	0.0352 (11)	0.0981 (12)	0.0038 (1)	0.0048 (4)	0.0540 (9)	0.0248 (7)	7.89
Squared Chi-squared	0.0692 (7)	0.0366 (9)	0.0215 (8)	0.0352 (11)	0.1037 (13)	0.0038 (1)	0.0047 (3)	0.0477 (8)	0.0241 (6)	7.33
Average (L_1, L_∞)	0.0036 (3)	0.0233 (2)	0.0128 (3)	0.0129 (2)	0.0257 (5)	0.0039 (2)	0.0047 (3)	0.0151 (2)	0.0235 (4)	2.88
Divergence	0.0746 (8)	0.0366 (9)	0.0223 (9)	0.0352 (11)	0.0981 (12)	0.0038 (1)	0.0048 (4)	0.0540 (9)	0.0248 (7)	7.78
Hassanat	0.0155 (9)	0.0667 (11)	0.0592 (10)	0.0349 (10)	0.0896 (11)	0.0563 (4)	0.0772 (8)	0.0604 (10)	0.0864 (11)	9.33
Whittaker's Index association	0.0036 (3)	0.0210 (1)	0.0178 (7)	0.0130 (3)	0.0333 (9)	0.0042 (3)	0.0070 (7)	0.0156 (5)	0.0263 (10)	5.33

Table A2. Test error of ex-kNN for $\tau = 050$ for nine data sets of different sizes under eighteen distance measures. The result in (·) indicates the rank of a distance measure for a specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.0063 (6)	0.0350 (4)	0.0145 (3)	0.0177 (7)	0.0535 (5)	0.0050 (2)	0.0064 (3)	0.0236 (1)	0.0385 (5)	3.89
Manhattan	0.0041 (3)	0.0405 (8)	0.0145 (3)	0.0174 (5)	0.0470 (1)	0.0050 (2)	0.0062 (1)	0.0254 (7)	0.0378 (2)	3.56
Chebychev	0.0085 (8)	0.0398 (7)	0.0142 (1)	0.0192 (8)	0.0624 (6)	0.0051 (3)	0.0067 (5)	0.0256 (8)	0.0402 (6)	5.78
Canberra	0.0040 (2)	0.0389 (6)	0.0149 (5)	0.0172 (4)	0.0474 (2)	0.0050 (2)	0.0062 (1)	0.0252 (6)	0.0375 (1)	3.22
Soergel	0.0040 (2)	0.0389 (6)	0.0149 (5)	0.0172 (4)	0.0474 (2)	0.0050 (2)	0.0062 (1)	0.0252 (6)	0.0375 (1)	3.22
Lorentzian	0.0037 (1)	0.0419 (9)	0.0143 (3)	0.0169 (3)	0.0470 (1)	0.0050 (2)	0.0062 (1)	0.0256 (8)	0.0378 (2)	3.33
Cosine	0.0067 (7)	0.0343 (2)	0.0219 (7)	0.0156 (1)	0.0663 (7)	0.0054 (4)	0.0100 (7)	0.0246 (4)	0.0429 (8)	4.78
Contracted JT	0.0067 (7)	0.0345 (3)	0.0141 (2)	0.0175 (6)	0.0527 (3)	0.0050 (2)	0.0066 (4)	0.0237 (2)	0.0382 (4)	3.67
Clark	0.0550 (11)	0.0435 (10)	0.0304 (9)	0.0560 (10)	0.1505 (10)	0.0049 (1)	0.0063 (2)	0.0716 (11)	0.0407 (7)	7.89
Squared Chi-squared	0.0506 (10)	0.0459 (11)	0.0231 (8)	0.0560 (10)	0.1268 (11)	0.0050 (2)	0.0064 (3)	0.0685 (9)	0.0392 (5)	7.67
Average (L_1, L_∞)	0.0046 (5)	0.0367 (5)	0.0146 (4)	0.0175 (6)	0.0533 (4)	0.0050 (2)	0.0064 (3)	0.0245 (3)	0.0379 (3)	3.88
Divergence	0.0550 (11)	0.0435 (10)	0.0304 (9)	0.0560 (10)	0.1505 (10)	0.0049 (1)	0.0063 (2)	0.0716 (11)	0.0407 (7)	7.89
Hassanat	0.0266 (9)	0.0963 (12)	0.0679 (10)	0.0558 (9)	0.0982 (9)	0.0582 (5)	0.1019 (7)	0.0701 (10)	0.1173 (11)	9.11
Whittaker's Index association	0.0042 (4)	0.0312 (1)	0.0214 (6)	0.0159 (2)	0.0678 (8)	0.0050 (2)	0.0099 (6)	0.0247 (5)	0.0435 (9)	4.78

Table A3. Test error of ex-kNN for $\tau = 0.75$ for nine data sets of different sizes under eighteen distance measures. The result in (·) indicates the rank of a distance measure for a specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.0073 (6)	0.0458 (3)	0.0165 (5)	0.0213 (6)	0.0535 (5)	0.0062 (3)	0.0078 (4)	0.0323 (1)	0.0521 (3)	3.67
Manhattan	0.0050 (3)	0.0547 (9)	0.0160 (2)	0.0211 (5)	0.0470 (1)	0.0061 (2)	0.0076 (2)	0.0349 (7)	0.0502 (1)	3.44
Chebychev	0.0103 (8)	0.0498 (5)	0.0164 (4)	0.0245 (8)	0.0624 (6)	0.0062 (3)	0.0081 (6)	0.0344 (5)	0.0552 (6)	5.67
Canberra	0.0047 (2)	0.0522 (7)	0.0169 (6)	0.0202 (3)	0.0474 (2)	0.0060 (1)	0.0076 (2)	0.0345 (6)	0.0502 (1)	3.33
Soergel	0.0047 (2)	0.0522 (7)	0.0169 (6)	0.0202 (3)	0.0474 (2)	0.0060 (1)	0.0076 (2)	0.0345 (6)	0.0502 (1)	3.33
Lorentzian	0.0045 (1)	0.0569 (11)	0.0156 (1)	0.0206 (4)	0.0470 (1)	0.0061 (2)	0.0075 (1)	0.0351 (8)	0.0502 (1)	3.33
Cosine	0.0072 (5)	0.0449 (2)	0.0263 (8)	0.0185 (1)	0.0633 (7)	0.0067 (5)	0.0129 (8)	0.0326 (3)	0.0591 (8)	5
Contracted JT	0.0075 (7)	0.0456 (10)	0.0162 (3)	0.0211 (5)	0.0527 (3)	0.0061 (2)	0.0079 (5)	0.0325 (2)	0.0524 (4)	4.56
Clark	0.0291 (9)	0.0518 (6)	0.0384 (10)	0.0803 (10)	0.1505 (9)	0.0060 (1)	0.0078 (4)	0.0804 (10)	0.0552 (6)	7.22
Squared Chi-squared	0.0303 (10)	0.0544 (8)	0.0275 (9)	0.0803 (10)	0.1268 (10)	0.0061 (2)	0.0079 (5)	0.0792 (9)	0.0534 (5)	7.56
Average (L_1, L_∞)	0.0057 (4)	0.0487 (4)	0.0165 (5)	0.0220 (7)	0.0533 (4)	0.0062 (3)	0.0077 (3)	0.0335 (4)	0.0511 (2)	4.0
Divergence	0.0291 (9)	0.0518 (6)	0.0384 (10)	0.0803 (10)	0.1505 (9)	0.0060 (1)	0.0078 (4)	0.0804 (10)	0.0552 (6)	7.22
Hassanat	0.0376 (11)	0.1271 (12)	0.0630 (11)	0.0801 (9)	0.0982 (8)	0.0548 (6)	0.0127 (7)	0.1330 (11)	0.1554 (9)	9.33
Whittaker's Index association	0.0047 (2)	0.0415 (1)	0.0251 (7)	0.0188 (2)	0.0678 (7)	0.0066 (4)	0.0076 (2)	0.0335 (4)	0.0590 (7)	4

Table A4. Computational time (in seconds) of ex-kNN for $\tau = 0.25$ for different datasets under different distance measures. The result in (·) indicates the rank of a distance measure for a specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.5446 (2)	0.5844 (1)	0.1316 (1)	1.6198 (4)	10.524 (2)	9.0639 (7)	16.1634 (3)	65.4357 (9)	75.7579 (1)	3.33
Manhattan	0.5498 (3)	0.818 (12)	0.1045 (12)	1.8028 (6)	10.364 (1)	8.5893 (2)	18.51 (10)	63.3749 (4)	77.5216 (2)	5.78
Chebychev	0.6044 (9)	0.5906 (2)	0.9668 (7)	3.0618 (13)	13.593 (10)	8.9278 (6)	17.4414 (7)	63.6503 (5)	86.1188 (10)	7.67
Canberra	0.5899 (8)	0.5954 (3)	0.9402 (4)	1.8594 (8)	10.826 (3)	8.5108 (1)	16.5988 (4)	65.7512 (10)	80.164 (5)	5.11
Soergel	0.5227 (1)	0.6073 (4)	0.8939 (2)	1.8247 (7)	12.143 (8)	9.6888 (10)	17.7741 (8)	65.3165 (8)	83.2344 (8)	6.22
Lorentzian	0.7613 (14)	0.7697 (11)	1.2644 (14)	3.2461 (14)	26.761 (13)	13.4067 (14)	24.3586 (14)	149.57 (14)	156.521 (14)	13.56
Cosine	0.5604 (6)	0.6299 (5)	1.0400 (13)	1.946 (10)	11.805 (6)	10.1804 (12)	19.8549 (12)	78.3711 (11)	94.7818 (13)	9.78
Contracted JT	0.5682 (7)	0.6525 (9)	0.9802 (9)	1.8632 (9)	11.362 (4)	8.7091 (4)	16.7185 (5)	65.1808 (7)	84.1575 (9)	7.00
Clark	0.6907 (12)	0.6334 (6)	0.9655 (5)	1.5661 (3)	12.752 (9)	9.4176 (9)	17.2851 (6)	56.4671 (3)	80.421 (6)	6.33
Squared Chi-squared	0.554 (4)	0.6351 (7)	0.9660 (6)	1.4994 (1)	11.9517 (7)	8.7754 (5)	15.9783 (2)	47.3696 (2)	78.1152 (3)	4.67
Average	0.5543 (5)	0.607 (4)	0.9069 (3)	1.7472 (5)	11.6386 (5)	9.1459 (8)	15.9075 (1)	64.6276 (6)	82.6535 (7)	4.89
Divergence	0.608 (10)	0.6077 (4)	0.9719 (8)	1.5152 (2)	12.14 (8)	8.6737 (3)	18.1009 (9)	47.3471 (1)	78.7135 (4)	5.44
Hassanat	0.7566 (13)	0.6872 (10)	1.0079 (13)	2.2696 (12)	18.315 (12)	10.7503 (13)	21.5964 (13)	93.594 (13)	92.7772 (12)	12.33
Whittaker	0.6154 (11)	0.6385 (8)	1.0025 (9)	1.9555 (11)	14.668 (11)	10.1414 (11)	19.4346 (11)	81.7056 (12)	91.5862 (11)	10.56

Table A5. Computational time (in seconds) of ex-kNN for $\tau = 0.50$ for different datasets under different distance measures. The result in (·) indicates the rank of a distance measures for specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.5708 (2)	0.5709 (1)	0.8935 (2)	1.6198 (4)	10.495 (1)	8.9081 (5)	15.9607 (2)	63.2521 (5)	76.2212 (1)	2.6
Manhattan	0.6034 (6)	0.594 (2)	0.9196 (4)	1.8028 (6)	10.517 (2)	8.2911 (1)	17.2263 (9)	66.1123 (9)	77.8895 (2)	4.6
Chebychev	0.539 (1)	0.6171 (5)	1.1711 (10)	2.0618 (11)	14.732 (9)	9.3347 (8)	17.9897 (10)	61.1169 (4)	81.1524 (6)	7.1
Canberra	0.5395 (1)	0.5937 (2)	0.9097 (3)	1.8594 (8)	10.806 (3)	8.6402 (2)	16.628 (4)	65.9869 (8)	82.4209 (8)	4.3
Soergel	0.5878 (3)	0.596 (3)	0.9583 (5)	1.8247 (7)	11.584 (5)	8.9926 (6)	16.9919(8)	65.3066 (6)	81.9699 (7)	5.6
Lorentzian	0.75 (9)	0.7826 (10)	2.0282 (11)	3.2461 (13)	25.827 (12)	13.44 (14)	23.1625 (14)	148.494 (14)	156.957 (13)	12.2
Cosine	0.6181 (7)	0.6827 (8)	0.9902 (6)	1.946 (9)	11.799 (7)	10.179 (11)	19.1273 (12)	78.2216 (11)	97.3233 (12)	9.2
Contracted JT	0.6053 (5)	0.5958 (3)	0.9215 (4)	1.8632 (8)	11.209 (4)	8.7274 (4)	16.9739 (7)	65.8887 (7)	190.639 (14)	6.2
Clark	0.6513 (8)	0.6448 (6)	0.9245 (4)	1.5661 (3)	12.109 (8)	9.343 (9)	16.8576 (6)	56.6508 (3)	80.3316 (5)	5.8
Squared Chi-squared	0.8069 (10)	0.6165 (5)	0.9978 (7)	1.4994 (1)	11.734 (6)	9.5527 (10)	15.9735 (3)	51.4523 (2)	78.0933 (3)	5.2
Average	0.57 (2)	0.6041 (3)	0.8823 (1)	1.7472 (5)	11.5787 (5)	9.0538 (7)	15.846 (1)	66.4976 (10)	83.7506 (9)	4.8
Divergence	0.5989 (4)	0.6124 (4)	1.0063 (8)	1.5152 (2)	12.115 (8)	8.665 (3)	16.7336 (5)	47.2858 (1)	78.5157 (4)	4.3
Hassanat	0.6543 (8)	0.7225 (9)	1.0487 (9)	2.2696 (12)	18.6056 (11)	11.2559 (13)	19.8263 (13)	95.3304 (13)	92.7626 (11)	11.0
Whittaker	0.5744 (2)	0.6718 (7)	0.9935 (6)	1.9555 (10)	15.3884 (10)	10.4403 (12)	18.2015 (11)	87.7859 (12)	91.691 (10)	8.9

Table A6. Computational time (in seconds) for ex-kNN for $\tau = 0.75$ for different data sets under different distance measures. The result in (·) indicates the rank of a distance measure for a specific dataset.

Distance Measure	Data Set									Average Rank
	NC-CRIME	CONCRETE-COMP	AIRFOIL-NOISE	MUNICH-RENT	UPDRS-MOTOR	HEAD-CIRCUM	CYCLE-PP	HOUR	CAL-HOUSING	
Euclidean	0.6173 (9)	0.5747 (1)	1.1302 (12)	1.8723 (8)	11.5164 (4)	8.8205 (6)	16.0187 (2)	64.775 (6)	74.3069 (1)	5.4
Manhattan	0.5877 (7)	0.627 (6)	1.0535 (9)	1.7588 (5)	11.1217 (2)	8.2967 (1)	16.273 (6)	61.1129 (4)	77.543 (2)	4.7
Chebychev	0.6173 (9)	0.6042 (3)	0.9624 (5)	1.7797 (6)	12.2382 (8)	9.296 (9)	17.5824 (9)	67.123 (9)	82.936 (6)	7.1
Canberra	0.514 (1)	0.608 (4)	1.0633 (10)	1.905 (9)	10.8455 (1)	8.4855 (2)	16.9752 (8)	65.7854 (8)	79.529 (4)	5.2
Soergel	0.6495 (10)	0.605 (4)	0.891 (2)	2.1117 (12)	11.6266 (6)	8.7794 (5)	17.6685 (10)	65.14 (7)	83.0972 (7)	7.0
Lorentzian	0.7376 (11)	0.8223 (11)	1.2033 (13)	3.1173 (14)	25.8591 (14)	13.592 (14)	24.2913 (13)	147.046 (14)	158.0938 (14)	13.1
Cosine	0.5593 (4)	0.6235 (5)	1.1075 (11)	1.9934 (11)	12.3925 (9)	10.141 (12)	18.8673 (14)	76.3668 (11)	98.0471 (12)	9.9
Contracted JT	0.522 (2)	0.5938 (2)	0.9526 (4)	1.7922 (7)	11.2751 (3)	8.6923 (4)	17.2228 (4)	69.6091 (10)	127.1401 (13)	5.4
Clark	0.5612 (4)	0.6482 (7)	1.0328 (8)	1.562 (3)	12.7084 (11)	8.9228 (7)	16.664 (7)	60.9276 (3)	80.3319 (5)	6.1
Squared Chi-squared	0.5808 (6)	0.6243 (5)	1.0119 (7)	1.5108 (2)	11.7821 (7)	9.6469 (10)	16.0538 (3)	50.4478 (1)	88.5031 (9)	5.6
Average	0.5702 (5)	0.6105 (4)	0.8769 (1)	1.7392 (4)	11.5745 (5)	9.1333 (8)	16.0031 (1)	62.9529 (5)	88 (8)	4.6
Divergence	0.5354 (3)	0.6829 (9)	0.991 (6)	1.4968 (1)	12.6362 (10)	8.6422 (3)	16.2329 (5)	50.6745 (2)	78.5331 (3)	4.7
Hassanat	0.6535 (9)	0.7815 (10)	1.0523 (9)	2.2169 (13)	18.0316 (13)	11.0908 (13)	18.2871 (12)	89.8852 (13)	92.8332 (11)	11.4
Whittaker	0.5924 (8)	0.6597 (8)	0.9252 (3)	1.926 (10)	15.1091 (12)	9.9105 (11)	17.8684 (11)	83.5628 (12)	91.9307 (10)	9.4

References

1. Koenker, R.; Bassett, G., Jr. Regression quantiles. *Econometrica* **1978**, *46*, 33–50. [CrossRef]
2. Newey, W.K.; Powell, J.L. Asymmetric least squares estimation and testing. *Econometrica* **1987**, *55*, 819–847. [CrossRef]
3. Efron, B. Regression percentiles using asymmetric squared error loss. *Statist. Sci.* **1991**, *1*, 93–125.
4. Abdous, B.; Remillard, B. Relating quantiles and expectiles under weighted-symmetry. *Ann. Inst. Statist. Math.* **1995**, *47*, 371–384. [CrossRef]
5. Schnabel, S.; Eilers, P. An analysis of life expectancy and economic production using expectile frontier zones. *Demogr. Res.* **2009**, *21*, 109–134. [CrossRef]
6. Sobotka, F.; Radice, R.; Marra, G.; Kneib, T. Estimating the relationship between women's education and fertility in Botswana by using an instrumental variable approach to semiparametric expectile regression. *J. R. Stat. Soc. C App.* **2013**, *62*, 25–45. [CrossRef]
7. Wang, Y.; Wang, S.; Lai, K.K. Measuring financial risk with generalized asymmetric least squares regression. *Appl. Soft Comput.* **2011**, *11*, 5793–5800. [CrossRef]
8. Kim, M.; Lee, S. Nonlinear expectile regression with application to value-at-risk and expected shortfall estimation. *Comput. Stat. Data Anal.* **2016**, *94*, 1–19. [CrossRef]
9. Bellini, F.; Klar, B.; Müller, A.; Gianin, R.E. Generalized quantiles as risk measures. *Insur. Math. Econ.* **2014**, *54*, 41–48. [CrossRef]
10. Keating, C.; Shadwick, W.F. A universal performance measure. *J. Perform Meas.* **2002**, *6*, 59–84.
11. Schnabel, S.K.; Eilers, P.H. Optimal expectile smoothing. *Comput. Statist. Data Anal.* **2009**, *53*, 4168–4177. [CrossRef]
12. Sobotka, F.; Kneib, T. Geoadditive expectile regression. *Comput. Statist. Data Anal.* **2012**, *56*, 755–767. [CrossRef]
13. Yang, Y.; Zou, H. Nonparametric multiple expectile regression via ER-Boost. *J. Stat. Comput. Simul.* **2015**, *85*, 1442–1458. [CrossRef]
14. Farooq, M.; Steinwart, I. An SVM-like approach for expectile regression. *Comput. Stat. Data Anal.* **2017**, *109*, 159–181. [CrossRef]
15. Yao, Q.; Tong, H. Asymmetric least squares regression estimation: A nonparametric approach. *J. Nonparametr. Statist.* **1996**, *6*, 273–292. [CrossRef]
16. Taunk, K.; De, S.; Verma, S.; Swetapadma, A. A brief review of nearest neighbor algorithm for learning and classification. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 15–17 May 2019; pp. 1255–1260.
17. Mulak, P.; Talhar, N. Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. *Int. J. Sci. Res.* **2015**, *4*, 2101–2104.
18. Lopes, N.; Ribeiro, B. On the impact of distance metrics in instance-based learning algorithms. In Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis, Santiago de Compostela, Spain, 17–19 June 2015; pp. 48–56.
19. Chomboon, K.; Chujai, P.; Teerarassamee, P.; Kerdprasop, K.; Kerdprasop, N. An empirical study of distance metrics for k-nearest neighbor algorithm. In Proceedings of the 3rd International Conference on Industrial Application Engineering, Kitakyushu, Japan, 28–31 March 2015.
20. Todeschini, R.; Ballabio, D.; Consonni, V.; Grisoni, F. A new concept of higher-order similarity and the role of distance/similarity measures in local classification methods. *Chemom. Intell. Lab. Syst.* **2016**, *157*, 50–57. [CrossRef]
21. Prasath, V.B.; Alfeilat, H.A.A.; Lasassmeh, O.; Hassanat, A. Distance and similarity measures effect on the performance of k-nearest neighbor classifier-a review. *arXiv* **2017**, arXiv:1708.04321.
22. Lall, U.; Sharma, A. A nearest neighbor bootstrap for resampling hydrologic time series. *Water Resour. Res.* **1996**, *32*, 679–693. [CrossRef]
23. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Wang, R. Efficient kNN classification with different numbers of nearest neighbors. *IEEE Trans. Neural Networks Learn. Syst.* **2017**, *29*, 1774–1785. [CrossRef]
24. Zhang, S.; Cheng, D.; Deng, Z.; Zong, M.; Deng, X. A novel kNN algorithm with data-driven k parameter computation. *Pattern Recognit. Lett.* **2018**, *109*, 44–54. [CrossRef]
25. Zhang, S.; Zong, M.; Sun, K.; Liu, Y.; Cheng, D. Efficient kNN algorithm based on graph sparse reconstruction. In Proceedings of the International Conference on Advanced Data Mining and Applications, Guilin, China, 19–21 December 2014; Springer: Cham, Switzerland, 2014; pp. 356–369.
26. Eddelbuettel, D.; François, R. Rcpp: Seamless R and C++ integration. *J. Stat. Softw.* **2011**, *40*, 1–18. [CrossRef]
27. Eddelbuettel, D.; Sanderson, C. RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Comput. Stat. Data Anal.* **2014**, *71*, 1054–1063. [CrossRef]
28. Yang, Y. Erboost: Nonparametric Multiple Expectile Regression via ER-Boost. 2015. R Package Version 1.3. Available online: https://CRAN.R-project.org/package=erboost (accessed on 1 March 2021).
29. Steinwart, I.; Thomann, P. LiquidSVM: A fast and versatile SVM package. *arXiv* **2017**, arXiv:1702.06899.