TimeStacking: An Improved Ensemble Learning Method for Continuous Time Series Classification

Victor Ribeiro, Gilberto Reynoso-Meza

TimeStacking: An Improved Ensemble Learning Method for Continuous Time Series Classification *

Victor Henrique Alves Ribeiro[1,2][0000−0002−9196−9890] and Gilberto Reynoso-Meza[1][0000−0002−8392−6225]

[1] Industrial & Systems Engineering Graduate Program (PPGEPS), Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, Brazil.
vhrique@pucpr.edu.br,g.reynosomeza@pucpr.br
[2] Hilab, Curitiba, Brazil.
victor.ribeiro@hilab.com.br

Abstract. Machine learning has gained great attention for solving time series classification problems. However, usual machine learning algorithms rely on learning from tabular data, and additional signal processing and data manipulation are necessary. Ensemble learning algorithms are famous for improving the performance in machine learning tasks by combining multiple predictors, but the usual techniques only take into account a single prediction from each base model. To improve the performance in time series classification tasks, this work proposes TimeStacking, a novel algorithm based on the famous ensemble learning technique stacked generalization (Stacking). Such an algorithm also takes into account the previous predictions of the base models to improve continuous time series classification tasks. Experiments are performed on a real-world dataset for drinking water quality monitoring, where TimeStacking achieves superior performance in comparison to Stacking and two other ensemble learning models, with over 10% improvement in terms of range-based F_1 score and over 30% in terms of range-based precision. Therefore, results show the effectiveness of TimeStacking for solving continuous time series classification problems.

Keywords: Machine learning · ensemble learning · blending · time series classification · drinking water quality.

1 Introduction

Machine learning has gained attention for solving time series classification problems. Different from usual classification tasks, time series classification attributes

* This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES), the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq), and the *Fundação Araucária* (FAPPR) - Brazil - Finance Codes: 159063/2017-0-PROSUC, 310079/2019-5-PQ2, 437105/2018-0-Univ, 51432/2018-PPP and PRONEX-042/2018.

are ordered. Therefore, attributes from previous samples influence the results of future ones [6].

Nevertheless, usual learning algorithms for tabular data are not recommended for such type of problems, given that it is necessary to take into account the dependency between the ordered samples. To this end, researchers have focused on developing different classification methods based on distance, dictionary, shapelets, and intervals [1]. Despite this, to the best of the authors knowledge, current time series classification methods only focus on the current and previous attributes.

Many real-world time series classification problems are range-based. That is, not only attributes, but also output targets occur over a period of time. One of such type of problems is anomaly detection, where an anomaly can occur at not only a single point in time, but for a continuous period [14]. Using machine learning techniques for such instances is aligned with the idea of industrial artificial intelligence [8].

Therefore, this work discusses that the usage of previous predictions of machine learning models can be used as additional features to improve the results of time series classification. With such additional information, the predictive models can filter incorrect predictions that could arise from noisy sequences of samples. To this end, this work proposes TimeStacking, a novel method for time series classification.

TimeStacking is an improvement of the stacked generalization (Stacking) method [15] for time series classification. Stacking is an ensemble method that trains a meta-learner to combine the outputs from several trained machine learning models, improving the results over single models. Different from the original algorithm, TimeStacking also combines the past predictions of the base models to better generalize range-based problems.

To evaluate the effectiveness of the proposed method, experiments are performed on a problem related to drinking water quality monitoring [9]. Such a problem involves the detection of quality events in water quality using a multivariate time series composed of different water and environmental data. TimeStacking is compared with three other ensemble combination methods, namely, Stacking, weighted voting, and weighted voting with windowed moving average, where statistical analysis confirms the effectiveness of the proposed method.

This work is organized as follows: Section 2 brings the background on time series classification and ensemble learning; Section 3 introduces the proposed TimeStacking method. Section 4 details the experimental procedure and the drinking water quality monitoring problem. Section 5 shows and discusses the results. Finally, the paper is concluded with some final remarks and future research directions.

2 Background

This section brings the background on time series classification and ensemble learning. First, the basics and recent literature on time series classification is presented. Later, the ensemble learning framework is detailed.

2.1 Time Series Classification

Time series classification involves the task of assigning correct classes to a series of time-dependent signals. To this end, it is important to define how the time series will be structured to enable a mapping from time series data to discrete classes. Nevertheless, the literature shows multiple different approaches, such as the use of the whole time series to compute distance measures, the comparison of smaller intervals of the time series, the use of phase-independent shapelets, dictionary-based algorithms, model-based algorithms and combinations of the previous methods [1].

Other approaches for time series analysis involve the extraction of relevant features [7]. Some examples of features that can be extracted from time series are global features, such as data distribution, stationary characteristics, auto-correlation, frequency-based transformations, and statistical model fitting. Moreover, statistical features can be extracted from the series.

Given that this work focuses on continuous time series classification, the time series is analyzed using intervals and manually curated features. The time series is split into equal sizes of sub series, or intervals. Finally, statistical and manually curated features are generated to extract relevant information from the series.

2.2 Ensemble Learning

To improve the predictive performance in classification tasks, it is possible to combine the output of multiple different classifiers. This is also known as ensemble learning [16]. The following steps are performed when building such ensembles: (a) member generation, where a pool of diverse base models is trained; (b) member selection, where a heuristic method is used to filter the most promising base models;[3] and (c) member combination, where the base models predictions are combined, thereby creating the ensemble's final prediction [12].

There are multiple different techniques to build the pool of base models, where the most important factor is to build diverse models [12]. To this end, diversity can be generated by using different learning algorithms or by modifying the training data when training each base model. The former generates what are called heterogeneous ensembles, while the latter can be used to generate homogeneous ensembles. This work focuses on using homogeneous ensembles of decision trees trained with random undersampling boosting (RUSBoost) [13].

Finally, it is essential to combine the base models' predictions. There are also multiple different approaches to do so, which can be split into trainable and non-trainable approaches. On the one hand, non-trainable approaches include using

[3] Ensemble member selection is an optional step, which is not used in this work.

the prediction with higher confidence, the mean prediction value, or the majority voting scheme. Moreover, it is also possible to include weights into each base model to improve the classification performance. On the other hand, trainable approaches use learning algorithms to combine the multiple predictions [12]. The most famous trainable approach is Stacking [15]. Such a method uses the predictions of the base models as input to a meta-learner, which automatically learns how to map the predictions to the expected output using machine learning or statistical techniques.

3 TimeStacking

This work proposes TimeStacking, a new ensemble method to improve continuous time series classification. The proposed method focuses on the combination of various trained base learners. It is based on the famous Stacking [15], which trains a meta-learner to generalize from the base learners' outputs. Different from the traditional method, where single outputs from the base learners are used as input for the meta-learner, TimeStacking also uses the previous predictions from the base learners to improve the performance on continuous time series classification problems (Figure 1).

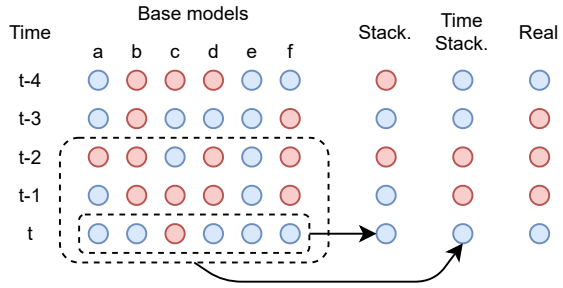

Fig. 1. Comparison of input data used by Stacking and TimeStacking methods to predict the outputs for binary problems.

The procedure to perform TimeStacking is similar to the one presented by Stacking. Such procedure is presented below:

1. First, the training dataset is split in two parts.[4]
2. Next, multiple base models are trained on the first split.
3. Subsequently, the base models predict the outputs for the other split.

[4] Stacking method can also be performed with k-fold cross validation, but this work only employs holdout validation.

4. Given the base models' outputs as input and the expected outputs, the meta-learner can be finally trained. At this step, it is important to notice that the number of previous outputs from the base-models can be selected as a hyperparameter for TimeStacking.[5] All the base model's current and previous outputs are then converted to a single array, which is used as input when training the meta-learner.
5. Finally, once new data is available, the base models can predict their outputs, which are used by TimeStacking to predict a more accurate output.

4 Drinking Water Quality Monitoring

To evaluate the effectiveness of the proposed method, this work compares TimeStacking with other ensemble combination methods in a dataset for drinking water quality monitoring. First, the dataset is introduced. Next, the ensemble pool generation step is detailed. Subsequently, the compared ensemble combination methods are shown. Finally, the evaluation metrics for the comparison are presented.

4.1 Dataset

The drinking water quality monitoring dataset has been proposed as an industrial challenge in 2018 [9]. It is composed of a training set and a separate test set. The training dataset presents an imbalance ratio of 67.8365, since it is a binary classification problem. Each split is composed of a time series with $139,566$ instances and the following features:

1. Time (yyyy-mm-dd HH:MM:SS format string);
2. Temperature (°C);
3. Chlorine dioxide amount (point 1) ($\mathrm{mg\,L^{-1}}$);
4. Chlorine dioxide amount (point 2) ($\mathrm{mg\,L^{-1}}$);
5. PH value (dimensionless);
6. Redox potential (mV);
7. Electric conductivity ($\mathrm{\mu S\,cm^{-1}}$);
8. Turbidity (NTU);
9. Flow rate (point 1) ($\mathrm{m^3\,h^{-1}}$);
10. Flow rate (point 2) ($\mathrm{m^3\,h^{-1}}$);

This work follows the preprocessing steps presented by Ribeiro et. al [10]. First, imputing and detrending operations mitigate issues related to missing data and concept drift. Next, feature extraction computes and combines statistical features. Finally, feature selection focuses on dimensionality reduction. Such steps are listed below:

[5] It is also interesting to notice that, if only the current outputs from the base models are used, TimeStacking is similar to Stacking.

1. To handle missing data, the previous available values $x(t-1)$ are imputed to future missing samples $x(t)$ at each feature f (Equation 1).

$$x_f(t) = \begin{cases} x_f(t), & x_f(t) \neq \emptyset \\ x_f(t-1), & \text{otherwise} \end{cases} \tag{1}$$

2. Detrending is performed by removing the simple moving average ($\overline{x}_f^{sm}(t,w)$) with a time window of $w = 1440$ minutes, or 24 hours (Equation 2).

$$x_f(t) = x_f(t) - \overline{x}_f^{sm}(t,w) \tag{2}$$

where

$$\overline{x}_f^{sm}(t,w) = \frac{1}{w}\sum_{i=0}^{w-1} x_f(t-i) \tag{3}$$

3. Additional features are computed using signal processing and statistical methods on a window of l samples, such as the differences (Equation 4), mean (Equation 5), standard deviation (Equation 6), maximum (Equation 7), minimum (Equation 8), and median (Equation 9).

$$x_{F+f}(t) = \Delta x_f(t) = x_f(t) - x_f(t-1) \tag{4}$$

$$\overline{x_f}(t) = \sum_{i=0}^{l-1} x_f(t-i)/l \tag{5}$$

$$x_f^{\sigma}(t) = \sqrt{\frac{\sum_{i=0}^{l-1}\left(x_f(t-i) - \overline{x_f}(t)\right)^2}{l-1}} \tag{6}$$

$$x_f^{max}(t) = \max\{x_f(t), \ldots, x_f(t-l+1)\} \tag{7}$$

$$x_f^{min}(t) = \min\{x_f(t), \ldots, x_f(t-l+1)\} \tag{8}$$

$$x_f^{med}(t) = \text{med}\{x_f(t), \ldots, x_f(t-l+1)\} \tag{9}$$

4. Moreover, the previous statistical features are combined to produce higher level features, including the standard deviation, the difference between the signal at current time and 30 minutes before, the difference between the signal at current time and its mean, the total signal amplitude, the difference between the maximum and current value, the difference between the current value and the minimum, the difference between the median and mean values. (Equation 10).

$$X_f(t) = \begin{bmatrix} x_f^{\sigma}(t) \\ x_f(t) - x_f(t-l+1) \\ x_f(t) - \overline{x_f}(t) \\ x_f^{max}(t) - x_f^{min}(t) \\ x_f^{max}(t) - x_f(t) \\ x_f(t) - x_f^{min}(t) \\ x_f^{med}(t) - \overline{x_f}(t) \end{bmatrix} \tag{10}$$

5. Since the previous step results in a set of 126 features, random forest (RF) [2] is employed to select the most valuable ones. The forest is configured with 100 trees, and the only the most relevant features are selected (Equation 11).

$$f_k^s = \begin{cases} 1, & f_k^i \geq \sum_{k=1}^{K} f_k^i / K \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

4.2 Ensemble Pool Generation

This work employs RUSBoost [13] to generate the pool of base models. The algorithm is a modification of adaptive boosting (AdaBoost) [5] for imbalanced problems. At each learning cycle, data from the majority class is randomly under sampled to balance the classes. Therefore, RUSBoost trains a pool of base models trained in a balanced manner.

Following the configurations that achieved high classification scores in previous works [10,11], RUSBoost is configured with 200 shallow decision trees, using a maximum of 10 decision splits and a ratio of 2 majority class observations for each minority class observation. Moreover, the base models are trained using the first 70% of the available training data.

4.3 Ensemble Combination

With the trained base models, the next step of the ensemble learning framework focuses on the combination of such methods.Therefore, four different methods are compared in this work, namely, weighted majority voting, weighted majority voting with additional moving average, Stacking, and TimeStacking.

The weighted majority voting is the default combination performed by RUSBoost (Equation 12). Similar to AdaBoost, RUSBoost computes a weight for each of its base models, which makes some base learners' outputs more valuable than others.

$$class(x) = \underset{c_i \in dom(y)}{\arg\max} \left(\sum_k h(y_k(x), c_i, w_k) \right) \tag{12}$$

where

$$h(y, c, w) = \begin{cases} w, & y = c \\ 0, & y \neq c \end{cases} \tag{13}$$

The weighted majority voting with moving average is performed to filter noisy predictions [10]. This model uses a window of $W = 3$ samples (current plus 2 previous predictions), where the filtered output $yf(t)$ is computed based on the majority of the 3 predictions. Such an output is computed as follows for a binary classification problem.

$$yf(t) = \left(\sum_{w=0}^{W-1} y(t-w)/W \right) \geq 0.5 \tag{14}$$

Finally, both Stacking and TimeStacking use RUSBoost to train the meta-learner. This is performed with the remaining 30% of the training dataset and using the same RUSBoost configurations from the ensemble pool generation step. However, while Stacking only uses the predictions $y(t)$ from each base model, TimeStacking uses the predictions $y(t)$, $y(t-1)$ and $y(t-2)$.

4.4 Evaluation

One of the main evaluation metrics for imbalanced data sets is the F_1 score [11], the harmonic mean between precision and recall (Equation 15). Such a metric is mainly concerned with point-based data. That is, the F_1 score does not take into account the relation between subsequent data, or ranges. However, continuous time series classification problems do not present point-based classes, but rather ranges where the classes can be found.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{15}$$

where

$$precision = TP/(TP + FP) \tag{16}$$

$$recall = TPR = TP/(TP + FN) \tag{17}$$

To handle range-based problems, this work employs time series adaptations of F_1 score, precision, and recall [14]. Different from the point-based precision and recall, the range-based metrics consider full sequences of classes when computing the scores. Moreover, such metrics can be configured to reward optimistic detections (any detection is rewarded regardless of how long it takes to occur) or early warning detection (earlier detections are more valuable) [10]. This work considers both scenarios equally by configuring the parameter for existence $\alpha = 0.5$, the consideration of front-end bias for recall (δ_{recall}), flat bias for precision ($\delta_{precision}$), and cardinality term $\gamma = 1$ [14].

Finally, the experiments are executed 31 times to enable statistical comparison. To this end, this work analyses the range-based F_1 score with the Friedman test and the post hoc Nemeniy test [3]. Moreover, the critical difference plot is employed to visualize the comparisons [4].

5 Results and Discussion

This section presents and discusses the results for all tested models. In total, 31 executions were performed to enable statistical comparison given the stochastic characteristic of the RUSBoost algorithm. Results are presented in tables, distribution plot, and the critical difference plot.

The original method (only RUSBoost) presents no variance across the 31 executions, since it is used as the baseline when constructing the Stacking and TimeStacking ensembles. The same occurs for the moving average method, since it does not present additional stochastic behavior. In comparison with the baseline method, which achieves F_1, precision, and recall scores of 0.31, 0.20, and 0.66, respectively, the moving average attains better F_1 (0.40) and precision (0.29) scores, while performing worse in terms of recall (0.64). Stacking obtains even better F_1 (0.48) and Precision scores (0.45), but worse recall (0.53). It is interesting to notice that Stacking presents small variance for all scores given the stochastic behavior of RUSBoost, with 0.03 for F_1 and recall, and 0.07 for precision. Among all models, TimeStacking achieves the highest F_1 (0.54) and precision (0.59) scores, with the worst recall (0.50). Similar to Stacking, TimeStacking also presents a variance in its results, with 0.01 for F_1 and 0.03 for precision. The variance for recall is close to zero (Table 1).

Table 1. Mean scores for each model in the drinking water quality monitoring problem.

Method	F_1 Score	Precision	Recall
Original	0.31	0.20	0.66
Moving Average	0.40	0.29	0.64
Stacking	0.48 (+/-0.03)	0.45 (+/-0.07)	0.53 (+/-0.03)
TimeStacking	0.54 (+/-0.01)	0.59 (+/-0.03)	0.50 (+/-0.00)

In addition, it is possible to analyze the distribution of the results. As previously discussed, the original ensemble and the moving average present fixed results. However, both Stacking and TimeStacking present variance in the results due to the stochastic behavior of the meta-learner RUSBoost. Nevertheless, neither methods achieve normal distribution.

In addition to the mean results (Table 1), Stacking has maximum and minimum F_1 scores of 0.56 and 0.41, respectively. TimeStacking, on the other hand, has maximum and minimum F_1 scores of 0.57 and 0.52, respectively (Figure 2). In terms of precision, Stacking achieves maximum and minimum values close to 0.65 and 0.35, while TimeStacking achieves 0.68 and 0.53, respectively (Figure 3). Finally, Stacking achieves maximum and minimum recall values close to 0.65 and 0.49, while TimeStacking achieves lower maximum and minimum recall scores of 0.52 and 0.49, respectively (Figure 4).

Finally, given the nonparametric characteristic of the results distributions, nonparametric statistical tests are performed. To this end, the Friedman's test with the post hoc Nemeniy test for multiple comparisons are performed [3]. Nonetheless, critical differences are analyzed with the critical differences plot [4] (Figure 5). Given the 31 runs and the 4 different compared models, a critical difference (CD) value of 0.84 is considered. On the one hand, the original and the moving average present mean ranks of 4 and 3, respectively. This occurs given their fixed results. On the other hand, TimeStacking and Stacking achieve mean ranks of 1.0323 and 1.9677, respectively. This occurs because there were few

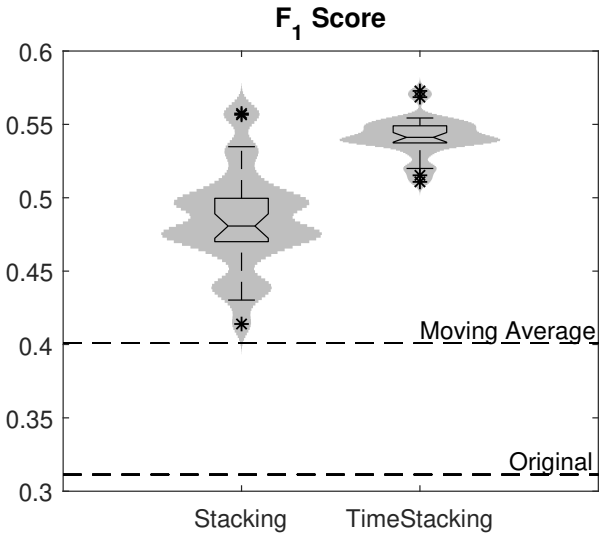

Fig. 2. Distribution plots with F_1 score results for each tested model.

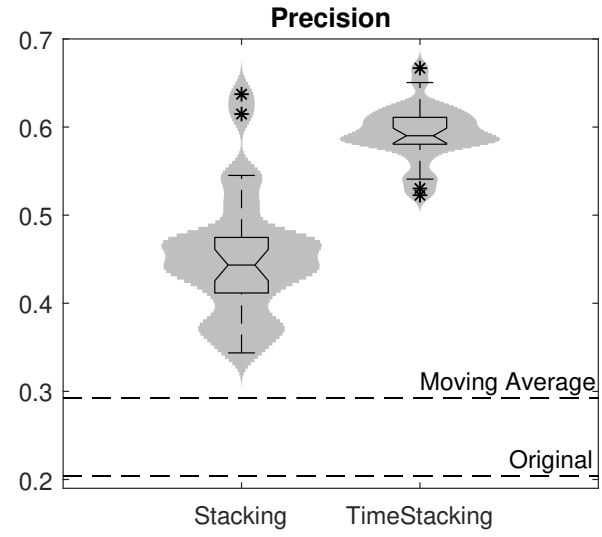

Fig. 3. Distribution plots with precision results for each tested model.

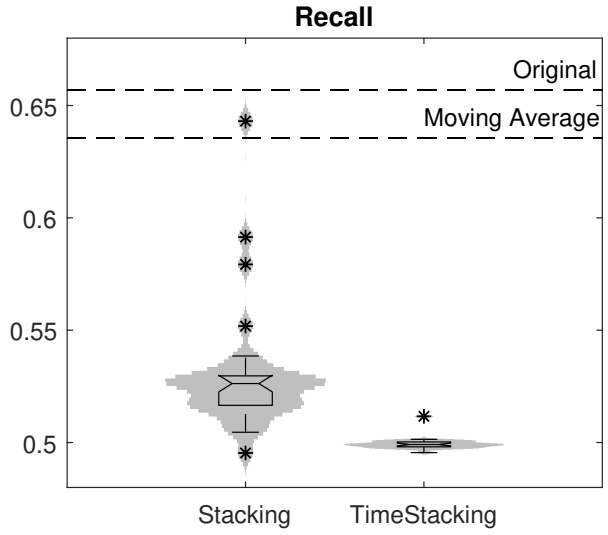

Fig. 4. Distribution plots with recall results for each tested model.

executions where Stacking achieved a better performance than TimeStacking. Therefore, since all differences between models are greater than the CD value, no results are statistically similar. Nevertheless, since TimeStacking achieved the best mean rank, it can be considered as the best solution for the drinking water quality monitoring problem.

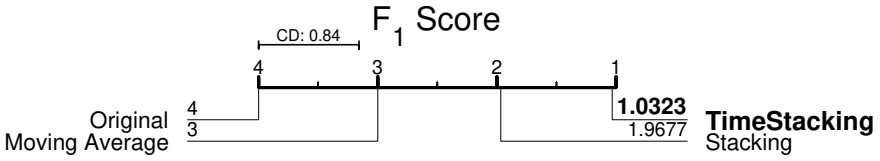

Fig. 5. Critical difference plot for the F_1 mean ranks of each tested model.

Similar result as the F_1 score has been achieved by the precision metric (Figure 6). However, different results have been achieved with recall. The original weighted majority voting method achieved the best mean rank for recall (1), followed by the moving average (2.0323). Finally, Stacking and TimeStacking achieved the worst ranks for such a metric, with mean ranks of 3 and 3.9677, respectively (Figure 7).

The results confirm the advantages of using TimeStacking to perform continuous time series classification. However, it is interesting to notice that, despite

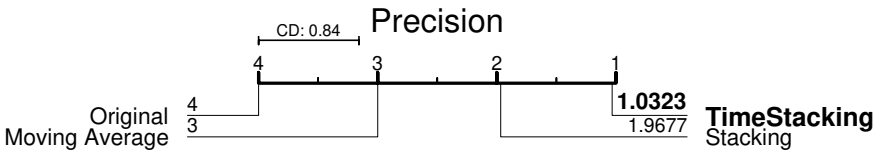

Fig. 6. Critical difference plot for the precision mean ranks of each tested model.

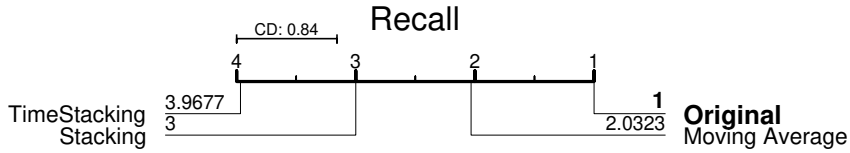

Fig. 7. Critical difference plot for the recall mean ranks of each tested model.

achieving the best F_1 scores, there has been a significant drop in the recall. This occurs because there is usually an inherent trade-off between precision and recall in binary classification problems. By using knowledge from previous predictions, TimeStacking achieves much higher precision at the cost of a lower recall. Nevertheless, as F_1 score is the harmonic mean between the previous scores, and it has also improved, the advantages of using TimeStacking surpass the disadvantages for the given task.

6 Conclusions

This work proposed TimeStacking as a new ensemble learning method to improve classification performance in continuous time series tasks. The proposed algorithm is an extension to the famous Stacking algorithm, but it includes the previous predictions of the base models to achieve better performance in time series classification.

To evaluate the effectiveness of TimeStacking, statistical comparison has been performed with Stacking, weighted majority voting, and weighted majority voting with moving average in a real-world problem related to drinking water quality monitoring. Results confirmed the superiority of TimeStacking for the given task, where improvements of over 10% in range-based F_1 score and over 30% in range-based precision have been achieved in comparison to Stacking.

Therefore, future work shall include the application of TimeStacking in different time series classification problems within the industrial domain. Nevertheless, it is of extreme importance to perform comparison on multiple problems to evaluate if the proposed model can outperform other techniques in not only a single dataset.

References

1. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)
2. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
3. Corder, G.W., Foreman, D.I.: Nonparametric statistics for non-statisticians: a step-by-step approach. John Wiley & Sons (2009)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research **7**(Jan), 1–30 (2006)
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences **55**(1), 119–139 (1997)
6. Fu, T.c.: A review on time series data mining. Engineering Applications of Artificial Intelligence **24**(1), 164–181 (2011)
7. Fulcher, B.D.: Feature-based time-series analysis. In: Feature engineering for machine learning and data analytics, pp. 87–116. CRC Press (2018)
8. Immerman, D.: An introduction to industrial artificial intelligence. InTech **July/August**, 34–38 (2020)
9. Rehbach, F., Moritz, S., Chandrasekaran, S., Rebolledo, M., Friese, M., Bartz-Beielstein, T.: Gecco 2018 industrial challenge: Monitoring of drinking-water quality (2018)
10. Ribeiro, V.H.A., Moritz, S., Rehbach, F., Reynoso-Meza, G.: A novel dynamic multi-criteria ensemble selection mechanism applied to drinking water quality anomaly detection. Science of The Total Environment **749**, 142368 (2020)
11. Ribeiro, V.H.A., Reynoso-Meza, G.: Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. Expert Systems with Applications **147**, 113232 (2020)
12. Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(4), e1249 (2018)
13. Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **40**(1), 185–197 (2009)
14. Tatbul, N., Lee, T.J., Zdonik, S., Alam, M., Gottschlich, J.: Precision and recall for time series. Advances in neural information processing systems **31**, 1920–1930 (2018)
15. Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)
16. Zhou, Z.H.: Ensemble learning. Encyclopedia of biometrics **1**, 270–273 (2009)