Time series extrinsic regression algorithms for forecasting long time series with a short horizon

Alexander Banwell and Anthony Bagnall

School of Electronics and Computer Science, University of Southampton

Abstract. Time series forecasting has a rich research history, with a variety of use cases across diverse domains. Many machine learning and deep learning forecasting algorithms approach the problem as regression by windowing. Given a univariate time series $Y = (y_1, ..., y_T)$, a training set of input-output pairs (x_i, y_i) is constructed where the input window $x_i = (y_{i-p+1}, ..., y_i)$ contains the most recent p observations, and the target variable z_i is a future value y_{i+h}, with h as the forecasting horizon. We call this approach time series forecasting regression (TSFR).
Recently, an alternative type of problem, time series extrinsic regression (TSER), has been described. In TSER, each training instance is a complete time series $\boldsymbol{X}_i \subset \Re$, assumed independent of the others, paired with an external response variable $z_i \subset \Re$. Unlike TSFR the response is not a future point of the same series but an exogenous variable to be predicted from the overall dynamics of the series. An archive of 63 TSER datasets was released in 2024. Our aim is to bridge the gap between TSER and TSFR by providing a common framework for both. We reformat a selection of diverse series used in forecasting research as TSER problems through windowing and create train/test splits. We then compare state-of-the-art TSER algorithms on these data and benchmark them against standard statistical and deep learning approaches. We find that the pattern of results seen on the TSFR archive does not mirror that on the TSER data. We propose a simple reformulation that allows the best TSER algorithm to achieve performance equivalent to statistical forecasters.

Keywords: Time series forecasting; time series regression; time series machine learning

1 Introduction

Forecasting time series has a rich and deep research history. There are a huge number of variations of use cases across diverse application domains. There are three main strands to the research into algorithms for forecasting: statistical, machine learning and deep learning, although there is a large degree of overlap [13]. One distinguishing feature is that a large number of the machine learning and (non-generative) deep learning forecasting algorithms reduce the problem to regression through windowing to form a training set collection of time series where a case is a historical window of the series to be forecast and the response variable is a number of steps ahead based on a forecasting horizon. We call this approach time series forecasting regression (TSFR) to differentiate it from an alternative time series regression formulation, time series extrinsic regression (TSER). In TSER, the problem is the same as traditional regression, except the explanatory variables are an ordered series of data equivalent to a time series. Each time series is assumed to be independent rather than windowed, and the response variable is not a future value of the series but an external variable. For example, one problem (BarCrawl16) in the archive is to predict how drunk someone is based on their motion. The explanatory variables are a time series of motion data (the coordinates derived from a wearable device) and the response/dependent variable is the blood alcohol level of the subject. Each case is a separate recording on one of 13 subjects. The data was originally published in [14] and donated to the UCI archive[1].

Although TSER is not itself a forecasting problem, the two formulations are closely connected. If we treat each window of past observations in a forecasting task as an independent series, then the prediction target (a future value) becomes just another external response variable. This shows that TSFR can be seen as a special case of TSER, where the "extrinsic" response happens to be a future continuation of the same process. Establishing this link provides the intuition for why TSER methods might be effective for forecasting and motivates our attempt to assess TSER methods for the specific case of forecasting.

The TSER problem was introduced in 2019 [24] with a collection of 19 problems. The archive was recently expanded to 63 problems from a range of domains including economics; energy, environmental and equipment monitoring; and sentiment analysis [11]. A range of new time series machine learning (TSML) algorithms were proposed based on algorithms from the more mature field of time series classification (TSC). These regressors follow a pipeline model of time series specific unsupervised transform to tabular format followed by a standard regression algorithm. Two TSML algorithms, FreshPRINCE [17] and DrCIF [20] were found to outperform a wide range of statistical and deep learning approaches on this TSER archive.

Standard regression techniques perform well on regression problems, with comparative studies often finding they outperform much more complex techniques. For example, it is observed in [22] that *"when considering Ranks, the (non-deep) machine learning methods LinearRegression and RandomForest (RF) outperform all competitors"*.

Our hypothesis is that regressors designed specifically for time series may be more effective than linear regression, Our aim is then to assess the effectiveness of TSER algorithms at TSFR. Forecasting is a huge research field with over 100 years of research. According to web of science[2], there were over 28,000 papers matching the keyword forecasting in 2024 and over 300,000 since 1970. It is also growing in popularity: 12,000 papers were recorded in 2015. For context, approximately 3000 matched the term TSC and just 7 for TSER in 2024. TSML is

[1] https://perma.cc/7VP5-79LN
[2] `https://www.webofscience.com/`

a relatively new field, and our goals in relation to forecasting must be realistic. We constrain our attention to operational rather than strategic forecasting [13]. We select a set of forecasting problems with characteristics we think are most suitable for TSML algorithms: relatively long series and a single step ahead forecasting horizon. We then evaluate a range of algorithms on these problems and compare performance with standard forecasting metrics, using classical statistical forecasters and standard deep learning algorithms for comparison.

We find that, for this scenario, no algorithm outperforms the benchmark naive forecasting strategy of predicting the last observed value. However, when we reformulate the problem so that the regressor predicts the change from the previous value (rather than the actual next value) the best TSER regressors outperform both naive forecasting and standard regression based algorithms such as XGBoost. The remainder of this paper is structured as follows. Section 2 describes the forecasting problems we use in experiments. Section 3 introduces the forecasting algorithms we evaluate, with more detail on standard statistical forecasters aimed at a machine learning audience. We present our results in Section 4 and analyse performance in Section 5, before concluding and describing future direction.

2 Forecasting Datasets

There are a very large number of forecasting datasets used in research and a range of forecasting competitions [16]. The Monash forecasting repository [10] contains hundreds of thousands of time series[3]. These are used in evaluations that typically aggregate performance over a hundreds of thousands of series with different characteristics to produce a single performance metric [1][4]. Our goals are more modest: we constrain our attention to series we believe are best suited to TSER and adopt a transformation/evaluation approach aligned with TSER.

2.1 Selection Criteria

We select 99 long time series and focus on one-step-ahead forecasting, due to computational constraints. When selecting datasets for our evaluation we are guided by the fact that we want to assess algorithms with specific data characteristics but without domain constraints. We prefer longer time series since we believe more complex models have more potential in this context. We use the Monash forecasting repository [10][5] as our starting point for datasets. We then select all series containing over 1000 data points. We do not want too many similar series. We filter out most energy demand/production datasets, and other weather datasets, since a portion of the mixed datasets will most likely contain weather-related data. We focus on univariate series only, because the TSER research has yet to address the multivariate case in detail. Of the remainder, we

[3] https://forecastingdata.org/
[4] https://huggingface.co/spaces/Salesforce/GIFT-Eval
[5] https://forecastingdata.org/

select 99 series at random for our preliminary evaluation. Table 1 shows the different categories of the time series used in this study.

Category	Number of Series	Series

Demographic	10	m4_daily_dataset_T{1604–1607}
		m4_monthly_dataset_T{26710}
		m4_weekly_dataset_T{224–227}
		us_births_dataset_T{1}

Energy Demand	10	australian_electricity_demand_dataset_T{1–3}
		elecdemand_dataset_T{1}, electricity_hourly_dataset_T{1–3}
		london_smart_meters_dataset_without_missing_values_T{1–3}

Energy Production	9	solar_10_minutes_dataset_T{1–5}
		wind_farms_minutely_dataset_without_missing_values_T{1,3–5}

Finance	10	m4_daily_dataset_T{2036,2037,2041}
		m4_monthly_dataset_T{37009,37070,37238,37248}
		m4_weekly_dataset_T{60–62}

Industry	10	m4_daily_dataset_T{1614,1615,1634,1650}
		m4_monthly_dataset_T{27138,27170,27175,27186}
		m4_weekly_dataset_T{55,56}

Macro	10	m4_daily_dataset_T{1,2,6}
		m4_monthly_dataset_T{122,145,180,186}
		m4_weekly_dataset_T{19–21}

Micro	10	m4_daily_dataset_T{130,131,145}
		m4_monthly_dataset_T{17051,17088,17132,17146}
		m4_weekly_dataset_T{248–250}

Other	10	kdd_cup_2018_dataset_without_missing_values_T{1}
		m4_daily_dataset_T{3595,3597}
		m4_hourly_dataset_T{170–172}
		m4_monthly_dataset_T{47915}
		m4_weekly_dataset_T{1,2}
		sunspot_dataset_without_missing_values_T{1}

Transportation	10	pedestrian_counts_dataset_T{1–5}
		traffic_hourly_dataset_T{1–5}

Weather	10	oikolab_weather_dataset_T{1–4}
		saugeenday_dataset_T{1}
		weather_dataset_T{1–5}

Table 1. Quantity of datasets from each field used in this study

2.2 Dataset Formatting

All series are taken from the Monash Forecasting Archive [10] in their .tsf format and formatted to be compatible with the `aeon` toolkit [18] which contains implementations of the TSER algorithms we use in evaluation. For practical purposes, we truncate each series to a maximum length of 10,000. We then split the series, using the first 70% of points to form the training data, with the last 30% being the test data. We transform the single series train and test data into `aeon` collections through windowing: we slide a 100 data point long window along each, using the first 99 as the regressors and the last variable as the response. We do this independently for the train and test partitions. Thus, for a series of 10,000 points, we generate collections of 6900 train cases and 2900 test cases. There is no overlap between the training and test sets at all.

This is an unusual formulation for forecasting and significantly restricts the information available to the regressor. Firstly, we do not produce predictions for the first 99 observations in the test series. We do this to ensure the train and test are completely independent. Secondly, we use the same model to predict the 100th test point as we do to predict the 3,000th. This is in contrast to the more commonly adopted strategies of giving all the data prior to the test point or adopting a direct or recursive strategy. In regression terms, we are making the standard assumption that there is no concept drift, i.e. that the underlying generative model does not change. However, for forecasting, we are assuming a prediction horizon from 100 to up to 3000, which is an unusual formulation.

We do it this way to make the problem as simple as possible to phrase as a TSER problem and to avoid any contamination between train and test data. In future work we will further bridge the gap to forecasting by looking at more commonly used formulations. In later experiments we compare to statistical models where this formulation does not work, since they are designed to predict only on time step ahead after being fitted. For these experiments we give the statistical models an advantage by giving the algorithm access to all the data prior to the test data point and producing a one ahead prediction. We do this as refitting a model for every test point would be computationally prohibitive, whereas with this formulation we can simply roll the models forward to produce predictions.

One advantage of the approach of making non overlapping train and test collections of series is that it makes reproducibility easier. We have made the 99 datasets available in the `aeon` .ts format as train/test data in addition to providing a single file .tsf with all 99 series used to generate the train test in one file on the accompanying website[6]. We have also provided simple code examples to perform the transforms and run a regression experiment.

To synchronise with TSER, we primarily use mean squared error (MSE) for comparison, comparing algorithms using adapted critical difference diagrams [6] that compare by ranks. Algorithms that are not significantly different from each other are grouped into cliques, within which there is no significant difference. We test significance using a pairwise Wilcoxon sign rank test ($\alpha = 0.05$) with Holm correction for multiple testing. We acknowledge that there are a large range of metrics used in forecasting, and we perform our final assessment with alternatives to MSE.

[6] `https://anonymous.4open.science/r/paper_2025-3B73`

3 Time Series Forecasting Algorithms

We have experimented with a diverse set of regressors to evaluate TSER for TSFR problems. We have compared with the full range of regressors reported in [11], but we omit several because they performed very poorly. These include nearest neighbour approaches and function regressors. Full results for these are available on the accompanying website.

Standard Regressors Standard regressors often perform well at forecasting tasks. We use regression versions of random forest (RandF) [2], extreme gradient boosting (XGBoost) [3] and Rotation Forest (RotF) [23] as standard machine learning benchmarks. These are trained and tested on the formatted tabular data formed by windowing. We also use a linear ridge regression algorithm (Ridge).

TSML Regressors Regression versions of the FreshPRINCE [17] (Fresh Pipeline with RotatIoN forest Classifier) and DrCIF [19] (Diverse Representation of Canonical Interval Features) were the best performing TSML algorithms for TSER [11]. FreshPRINCE is a pipeline algorithm for regression with two components: the TSFresh [4] feature extraction algorithm that transforms the input time series into a feature vector, and then a Rotation Forest estimator that builds a model and makes target predictions. The Diverse Representation of Canonical Interval Features (DrCIF) is an interval based tree ensemble regressor. Interval-based techniques select phase-dependent intervals of fixed offsets from which to extract summary features. These intervals share their position for all time series, with the aim of discovering discriminatory features from particular locations in time. Most interval techniques take the form of a forest of decision trees, using different intervals to achieve diversity in the ensemble. The first interval based approach, Time Series Forest (TSF) [7] and DrCIF [21] are both tree ensembles that use unsupervised transforms to extract features. TSF uses simple summary features (mean, standard deviation and slope) over the original time series. DrCIF uses three data representations: the original time series; the first order differences; and the periodogram. A different transform based on the CAnonical Time series CHaracteristics (Catch22) features [15] is created for each base regressor over these diverse representations. Catch22 is a set of 22 features filtered from the 7000+ available in the Highly Comparative Time Series Analysis (HCTSA) toolbox [9]. The Catch22 features were selected for use on normalised data, but we do not make that assumption. Hence, seven additional summary statistics are also candidates: the mean, standard-deviation, slope, median, interquartile range, min, and max. For each data representation, a set of k random intervals are selected, and the a unsupervised features are calculated and concatenated. Finally, a CART tree is trained on the feature set unique to each ensemble member.

Another popular family of TSML algorithms is based on random convolutions. The ROCKET family of classifiers [5] all involve unsupervised transformations using randomised convolutions and a pooling operation followed by a linear or ridge classifier or regressor. The original ROCKET was converted to TSER by switching the classifier for a ridge regressor. We extend this to consider a more recent ROCKET variant, MultiROCKET [25].

Deep learning regressors Deep learning algorithms are simple to adapt for regression tasks. To mirror the TSER study, we include a standard convolutional neural network adapted for time series (CNN) [26], Residual Network (ResNet) [12] and two versions of InceptionTime [8], a single InceptionTime model and an ensemble of five base models.

Statistical forecasters For benchmarking, we also compare with standard statistical forecasting techniques. We use bespoke implementations benchmarked for correctness against other open source alternatives, and are significantly faster implementations than those available in other common toolkits. These have been refactored into the aeon toolkit.

Naive Forecaster. This simply predicts the last value in the series as the one-step ahead forecast. This is one of the simplest methods and is included as a benchmark for more complicated methods.

Exponential Smoothing. At its simplest, exponential smoothing involves applying a weighted average to previous observations, where these weights decrease exponentially as the observations go further back in time. Different exponential smoothing algorithms take into account features of the data such as seasonality and trend, including damping the trend, as undamped trend often produces unreliable longer-horizon forecasts. The forecasting parameters are then usually estimated by minimising the sum of squared errors (SSE) $\text{SSE} = \sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2$.

It is also possible to translate these equations into statistical models by defining the error as additive error as in 1 or multiplicative error as in 2. We then assume that these errors are normally and independently distributed (NID) white noise with mean 0 and variance σ^2 i.e. $e_t = \varepsilon_t \sim NID(0, \sigma^2)$.

$$e_t = y_t - \hat{y}_{t|t-1} \tag{1}$$

$$e_t = \frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}} \tag{2}$$

We can then generate the state space error equations relating the observed values of the forecast variable and new values of level, trend and seasonality with previous values of level, trend and seasonality as well as the one-step errors. An example is given in equation 3 for a model with additive errors, trend and seasonality. This model would be described in shorthand as ETS(A,A,A)

$$\begin{aligned} y_t &= l_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t \\ s_t &= s_{t-m} + \gamma\varepsilon_t \\ \text{Where } \beta &= \alpha\beta^* \text{ and } \gamma = (1-\alpha)\gamma^* \end{aligned} \tag{3}$$

This model can then be used to estimate prediction intervals as well as produce forecasts. It also allows the estimation of the smoothing parameters by maximising the likelihood. Algorithms exist for automatically selecting the best ETS model base by maximising the likelihood.

ARIMA. An auto-regressive (AR) model uses a weighted combination of p values of the series. A qth order moving average MA(q) model takes a constant *c* and adds this to the weighted average of the errors of previous forecasts. These models require the data to be stationary. The ARIMA(p,d,q) model removes the requirement of the data to be stationary by applying differencing to the data. Differencing is applied repeatedly until the data are stationary. The degrees of differencing applied by the model are given by d. The ARIMA(p,d,q) model is then the combination of an AR(p) and a MA(q) component of the form given in equation 4 combined with differencing the data.

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + c + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{4}$$

Seasonal ARIMA uses a basic ARIMA model plus an ARIMA model where the lagged values are at the seasonal period. It has a different set of coefficients for the seasonal and non-seasonal lags as given in equation 5.

$$\begin{aligned} y_t = {} & \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + c + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \\ & + \varPhi_1 y_{t-s} + ... + \varPhi_P y_{t-P*s} + \varTheta_1 \varepsilon_{t-s} + ... + \varTheta_Q \varepsilon_{t-Q*s} + \varepsilon_t \end{aligned} \tag{5}$$

Differencing is applied using both a lag of one and also the seasonal lag. The degrees of normal differencing is given by d, and the degrees of seasonal differencing is given by D. The overall model is denoted as ARIMA(p,d,q)(P,D,Q).

The seasonal ARIMA model has many parameters, and hyperparameters which need to be tuned for the data. It is not practical to do this by hand for a large number of datasets, so an algorithm for selecting these parameters is required. The parameters ϕ, θ, $\varPhi$, $\varTheta$ and c are selected using a numerical optimisation technique such as the Nelder-Mead algorithm to minimise the Akaike's Information Criteria (AIC). A variation on the Hyndman-Khandakar algorithm is used to select the hyperparameters.

Algorithm: Modified Hyndman-Khandakar algorithm for automatic seasonal ARIMA modelling

1. The seasonal period of the data is estimated using the autocorrelation of the series
2. The number of differences d is determined using repeated KPSS tests.
3. If the seasonal period is not one, the data is seasonally differenced once (i.e. $D = 1$) else it is not ($D = 0$).
4. A stepwise search is then used to search for suitable values of p, q, P and Q.
   (a) The best initial model (with the smallest AIC) is selected from:
       ARIMA(2,d,2)(0,D,0)
       ARIMA(0,d,0)(0,D,0)

ARIMA(1,d,0)(0,D,0)
ARIMA(0,d,1)(0,D,0)
If d = 0 then c is included in the model, otherwise it is set to 0.

(b) The model is then varied 1 step at a time. One of p, q, P and Q is varied by ± 1 and this is tested both with c included in the model and $c = 0$.

(c) The previous step is repeated until no adjustment, either way, of any of the parameters yields a lower AIC.

Table 2. List of all forecasting algorithms used in evaluation

Algorithm	Acronym	Implementation
Machine Learning		
Ridge Regression	Ridge	scikit-learn
Rotation Forest [23]	RotF	aeon
Random Forest [2]	RandF	scikit-learn
Extreme Gradient Boosting [3]	XGBoost	scikit-learn
Deep Learning		
Convolutional Network [26]	CNN	aeon
Residual Network [12]	ResNet	aeon
Inception Time [8]	Inception	aeon
Inception Time Ensemble [8]	InceptionE	aeon
Time Series Machine Learning		
Time Series Forest [7]	TSF	aeon
Randomised convolutions [5]	ROCKET	aeon
MultiROCKET [25]	ROCKET	aeon
Diverse representations canonical intervals [19]	DrCIF	aeon
TSFresh/rotation forest pipeline	FreshPRINCE	aeon
Statistics		
Predict last	Naive	bespoke
Exponential Smoothing	ETS	bespoke
Automatic ETS	AutoETS	bespoke
AutoARIMA	AutoARIMA	bespoke
AutoSARIMA	AutoSARIMA	bespoke

Table 2 summarises the algorithms used in evaluation. All estimators are used with the default configuration from the associated open source toolkit.

4 Results

Our first experiment tests whether the pattern of performance observed with TSER algorithms on TSER data is recreated with the 99 TSFR datasets. The final experiment in [11] compared 13 regression algorithms (acronyms explained in table 2). Figure 1 shows the averaged rank performance of these 13 algorithms

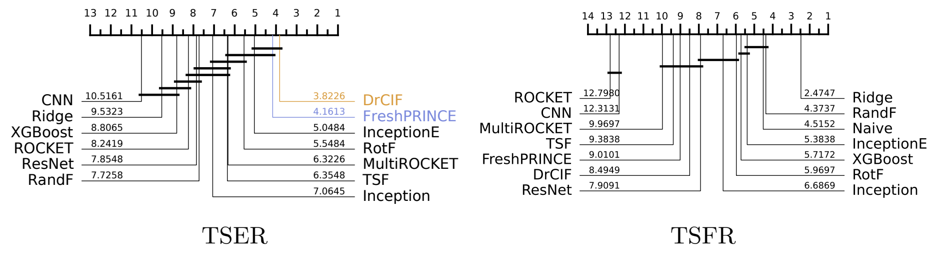

Fig. 1. (a) Ranks of 13 regressors on 63 TSER data published in [11] and reproduced with permission and (b) equivalent ranks on the 99 TSER problems, estimators ranked by MSE.

on (a) the TSER data and (b) the TSFR, with the addition of a naive forecaster to provide context.

The pattern of results is very different. The most notable difference is that ridge regression is promoted from the second worst to the best algorithm and the two best performers at TSER, DrCIF and FreshPRINCE, are now in the bottom half. We have included a Naive forecaster for context. Naive simply predicts the next value to be the same as the last seen. Ridge is the only algorithm to significantly outperform the naive strategy of predicting the last seen value. In fact, random forest and inception time are the only algorithms that are not significantly worse than the naive forecaster. These results are surprising and may be due to the fact the problem formulation may just be too hard. Naive forecasting is often a strong benchmark, and the experimental framework is of little value if it is too hard to outperform a naive forecaster. We are creating a large degree of separation between the train and test series, meaning models would need to find a strong representation of the underlying process driving the time series to make significant predictions.

We believe that one reason the TSER algorithms do badly is that they do not explicitly encode a key characteristic of forecasting, namely, that the next value is directly related to the previous value. Naive does well because this constraint makes it unlikely that one step ahead is that different to the last observed value. This constraint is hard coded into Naive and it could be argued that linear models like Ridge are designed for this scenario too. TSER do not encode this constraint.

To test the effect of explicitly encoding the relationship between observations at t and $t+1$ we can simply reformulate the problem so that the regressor is now the difference between observations rather than the actual value. We are changing the problem from ”predict the next value” to ”predict the change from the previous value”. This does not introduce any bias, since it is in the very nature of the forecasting problem. We leave the response variables the same (i.e. windows of the original values) and repeat our experiments. Figure 2 shows the relative performance of machine learning and TSML algorithms both with and without differencing the response variable (A prefix $d-$ indicates differenced). We do not include d-Ridge because it performed worse than the original.

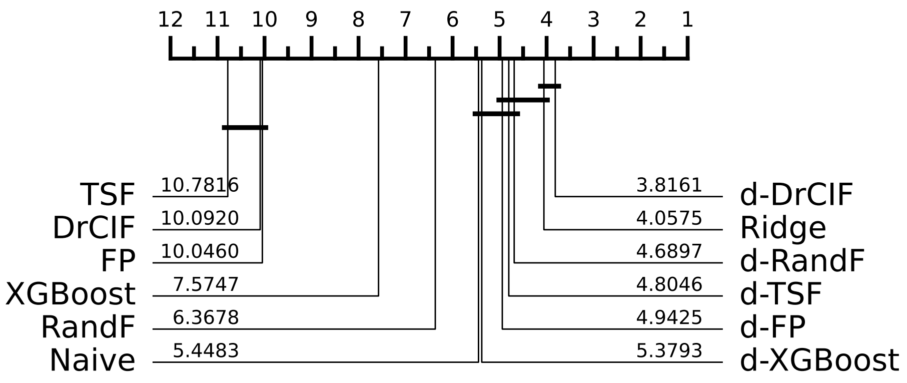

Fig. 2. Relative ranks of 12 regressors when predicting the next value or the change in the next value from the previous (prefix d). Estimators are ranked by MSE.

Predicting the change in y rather than the actual value improves performance of all of the regression based algorithms, but the effect is greater with the TSER approaches. DrCIF now joins Ridge as the only estimator significantly better than Naive and it is also significantly better than standard regressors RandF, RotF and XGBoost. This suggests there is at least some promise from a TSER approach.

To calibrate how hard our forecasting formulation is, we compare performance to standard statistical forecasting algorithms exponential smoothing (ETS) and auto regressive moving average (ARIMA) models. We cannot make a direct comparison because the statistical algorithms are designed to forecast one step ahead. To be directly comparable, we train standard forecasters on the train series then recursively make predictions across the test horizon, allowing the statistical forecaster to retain information about the time step so it can use the correct seasonal component. For the ARIMA models, we also allow the model to fit know the true residual at each step, otherwise it will degrade to an AR model. We would not expect ARIMA to do well with such a large prediction horizon, and this is borne out by the results. Figure 3 shows the relative performance of the time series regressors in comparison to the stats models. DrCIF, Ridge and AutoETS are better than Naive. This would imply that projecting the trend and seasonality across a long prediction horizon is better than the simple benchmark.

It is quite possible a more recent deep learning or large language model could do better. However, Figure 4 shows that standard deep learners are worse than

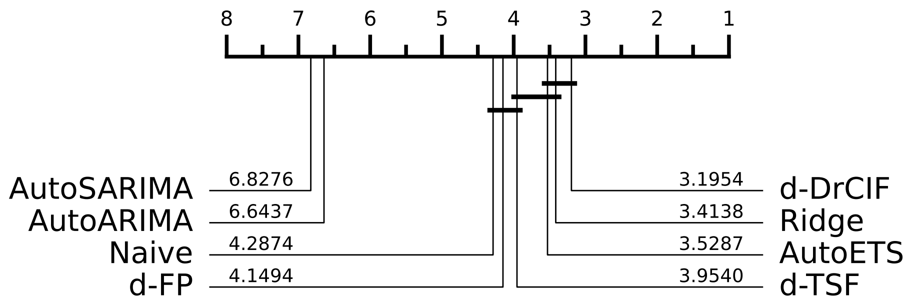

Fig. 3. Relative ranks of regression based forecasters and standard statistical forecasters.

DrCIF on differenced response data, indicating the problem is non trivial (we have not had the computational resources to run InceptionTime).

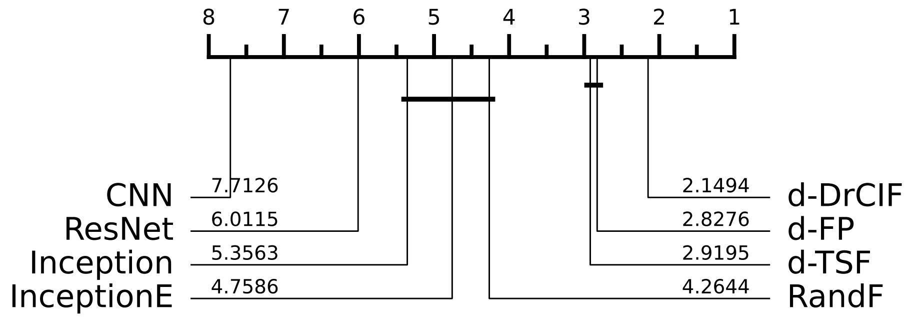

Fig. 4. Relative ranks of differenced TSER algorithms and deep learners.

5 Analysis

The performance of AutoETS suggests that the improvement over Naive is achieved through retaining the seasonality and trend. We investigate whether DrCIF is simply recreating this effect rather than modelling some other underlying process. We look in more detail at the performance of DrCIF relative to

AutoETS in order to determine the potential utility of the approach and its strengths and weaknesses. Figure 5 shows the scatter plots of DrCIF and AutoETS against the Naive forecaster. DrCIF shows much greater variation from Naive compared with ETS, suggesting there might be occasions where it discovers something more than trend and seasonality. To demonstrate this we examine

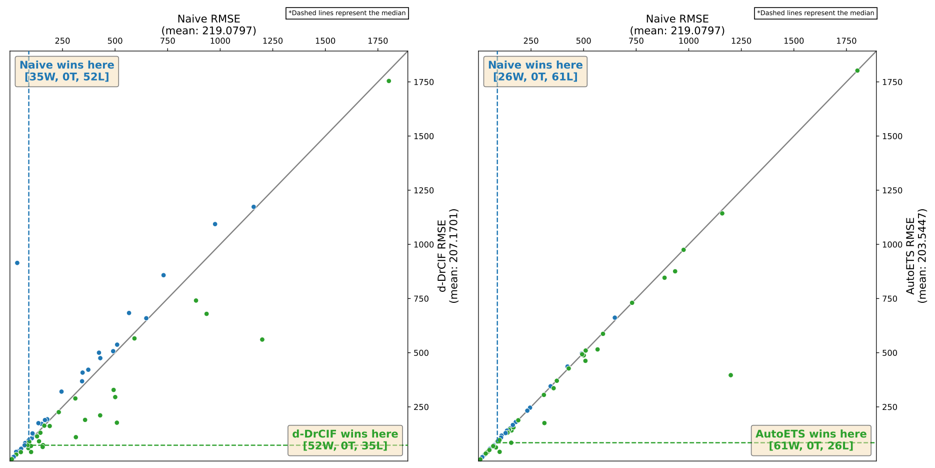

Fig. 5. RMSE scatter plots of Naive against DrCIF and ETS.

some series where the performance of DrCIF and AutoETS is most different. Table 5 breaks down wins and losses by problem type. AutoETS does better at the daily and monthly data, whereas DrCIF wins for hourly and the pedestrian class. These data are less likely to exhibit steady trends and seasonality.

Row Labels	**DrCif wins**	**AutoETS win**
australian	3	0
Births	0	1
Daily	5	14
Hourly	8	0
London	4	1
Monthly	3	15
Pedestrian	5	0
Solar	2	3
Weather	3	3
Weekly	11	6
Grand Total	**44**	**43**

Table 3. Comparison of DrCIF and Auto wins across datasets

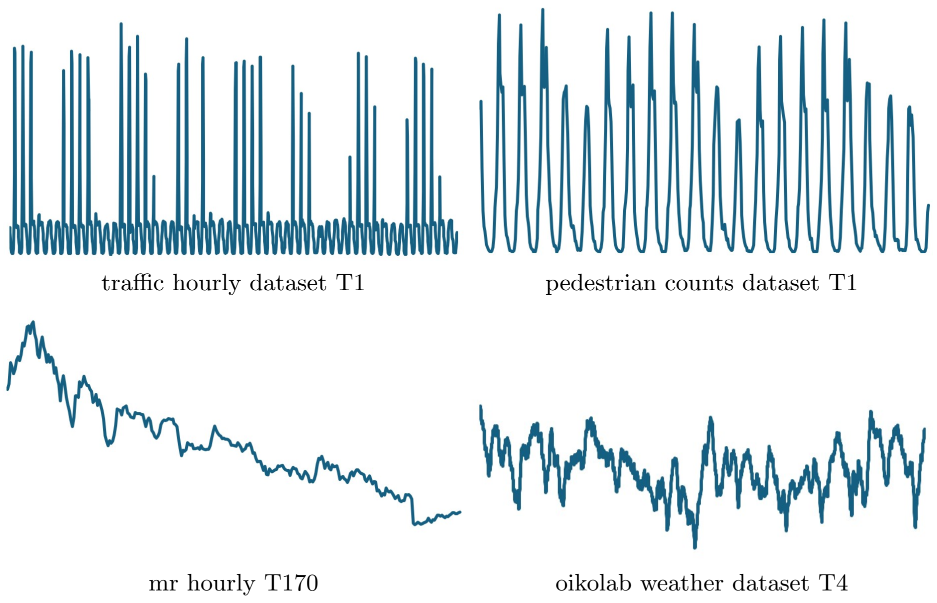

Fig. 6. Four example segments of test data where DrCIF performed better than AutoETS.

Figure 6 shows four problems where DrCIF performed relatively well and AutoETS did poorly,.

6 Conclusions

We have addressed the question of whether there is a place for TSER algorithms in TSFR. Our specification would be considered challenging. Our conclusion is there is some promise, but also room for improvement.

We have directly reformulated operational forecasting problems as time series regression problems in order to bridge the gap between TSER and TSFR research fields. We compared TSML algorithms to alternative approaches from machine learning, deep learning and statistics. The general conclusion is that with this formulation it is hard to beat Naive forecasting with any approach, and the only algorithm to do so is Ridge regression. We argue this is because forecasting inherently encodes the problem as "how will the value change from the time before", whereas the first TSER formulation is unconstrained. By making the simple transform of using the change in series as the response variable, we significantly improve all the regression algorithms. The best performer, DrCIF, is significantly better than Naive forecasting and no worse than baselines ridge regression and AutoETS, where AutoETS has an inherent advantage of internally recording the true time step. This is achieved without any alteration to DrCIF itself, and using a relatively short window of past values. The fact

AutoETS outperforms Naive implies there is some persistence of trend and seasonality over these large prediction intervals. That both ridge and DrCIF are equivalent to AutoETS suggests that they are successfully internally modelling this, albeit in a much more complex way than ETS. This is the first phase of the investigation and is a benchmarking exercise. We will repeat these experiments with a longer window for the regression problems before switching to a more familiar expanding window formulation. We will also evaluate regressors using direct and iterative forecasting over a shorter predictive horizon. It would also be interesting to evaluate the performance of predicting the percentage difference of the series. We will then expand our evaluation to consider more recently proposed machine learning and deep learning forecasting algorithms, whilst extending the scope of TSER ensembles to incorporate meta-ensemble structures.

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) (grant ref.: EP/W030756/2).

References

1. T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo. GIFT-eval: A benchmark for general time series forecasting model evaluation. 2024.
2. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
3. T. Chen. XGBoost: A Scalable Tree Boosting System. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
4. M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.
5. A. Dempster, F. Petitjean, and G. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34:1454–1495, 2020.
6. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
7. H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.
8. H. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. Schmidt, J. Weber, G. Webb, L. Idoumghar, P. Muller, and F. Petitjean. InceptionTime: finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.
9. B. Fulcher and N. Jones. hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell Systems*, 5(5):527–531, 2017.
10. R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

11. D. Guijo-Rubio, M. Middlehurst, G. Arcencio, D. F. Silva, and A. Bagnall. Unsupervised feature based algorithms for time series extrinsic regression. *Data Mining and Knowledge Discovery*, 38(4):2141–2185, 2024.
12. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
13. T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 2019.
14. J. A. Killian, K. M. Passino, A. Nandi, D. R. Madden, J. D. Clapp, N. Wiratunga, F. Coenen, and S. Sani. Learning to detect heavy drinking episodes using smartphone accelerometer data. In *KHD@ IJCAI*, pages 35–42, 2019.
15. C. Lubba, S. Sethi, P. Knaute, S. Schultz, B. Fulcher, and N. Jones. catch22: canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6):1821–1852, 2019.
16. S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.
17. M. Middlehurst and A. Bagnall. The FreshPRINCE: A simple transformation based pipeline time series classifier. In *International Conference on Pattern Recognition and Artificial Intelligence*, pages 150–161. Springer, 2022.
18. M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, and A. Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024.
19. M. Middlehurst, J. Large, and A. Bagnall. The canonical interval forest (CIF) classifier for time series classification. In *IEEE International Conference on Big Data*, pages 188–195, 2020.
20. M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall. HIVE-COTE 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110:3211–3243, 2021.
21. M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *arXiv preprint arXiv:2104.07551*, 2021.
22. X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, and B. Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. 17(9), 2024.
23. J. Rodriguez, L. Kuncheva, and C. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.
24. C. W. Tan, C. Bergmeir, F. Petitjean, and G. Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35:1032—1060, 2021.
25. C. W. Tan, A. Dempster, C. Bergmeir, and G. Webb. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36:1623–1646, 2022.
26. B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.