Improving the Forecast Accuracy of Protected Data Using Time Series Features

Cameron D. Bale, Matthew J. Schneider, Jinwook Lee

Drexel University

Abstract

Existing data privacy methods degrade forecast accuracy to unusable levels. To overcome this problem, we investigate the similarity between time series features that are predictive of forecast accuracy. We develop a matrix-based privacy method called k-nearest time series + (k-nTS+) swapping tailored to maintain forecast accuracy. We apply our privacy method to a forecasting competition data set where the identities of the time series are hidden but an adversary seeks to identify them. Using only six time series features, we find that k-nTS+ swapping maintains forecast accuracy and preserves the distribution of time series features much better than competitor methods at similar privacy levels. The k-nTS+ protected time series are also more representative of the original data, potentially leading to increased trust between data owners and forecasters. By preserving the original time series features and improving the privacy of sensitive data, our k-nTS+ method could produce data for multiple applications and increase data availability for the forecasting community.

Keywords: data privacy; time series features; forecast accuracy; identification disclosure risk

1. Introduction

Personally identifiable time series data are abundant and require protection (Boone et al., 2019). Recently, the General Data Protection Regulation (GDPR)[1] forced data owners to either anonymize their personal data or place strict limitations on data transfers and processing[2]. However, past research found that anonymizing time series data significantly reduced forecast accuracy (Gonçalves et al. 2021a). As a result, forecasters (Gonçalves et al., 2021b) used complex data processing solutions to share accurate forecasts without sharing the adjoining time series data. In this paper, we propose a privacy solution for data owners to anonymize time series data directly with good forecast accuracy.

Time series data are either stored in a single data set (centralized) or spread across multiple data owners and/or data sets (decentralized). In the decentralized scenario, multi-party computation and federated learning enable privacy-preserving collaborative forecasting to ensure accurate forecasts while protecting sensitive data (Gonçalves et al., 2021a; Gonçalves et al., 2021b; Sommer et al., 2021). For example, in a decentralized scenario, data owners can sell time series data to a market operator who then sells forecasts of the time series data to multiple buyers. This approach has the advantage of creating a market of economic incentives for data sharing while limiting data transfer and protecting privacy. However, there are still privacy risks including potential data breaches with the transfer and storage of the time series data to the market operator.

Our paper focuses on the centralized scenario where a single data owner uses privacy methods to protect the original time series data set. These privacy methods anonymize the time series data by

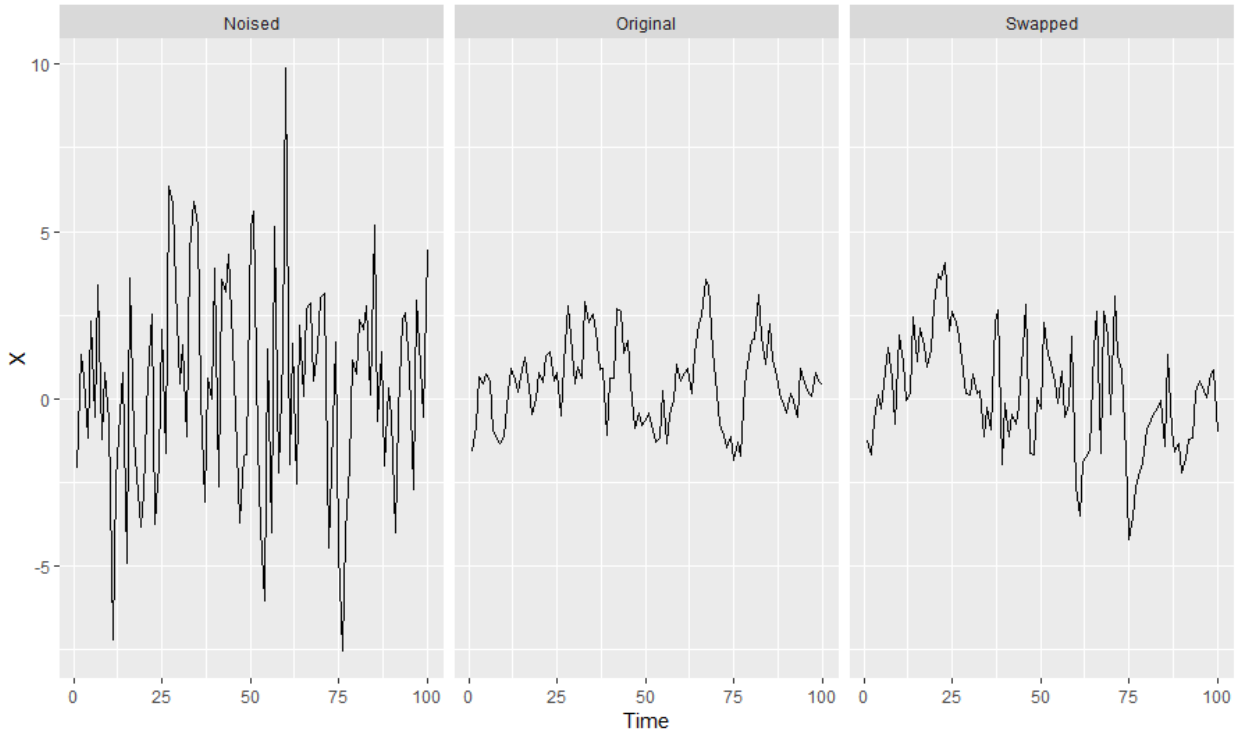
1 For legislation examples in the United States, see this map.

2 See articles 6, 45, and 46 of the GDPR.

directly altering the values within the data set to increase the privacy level of the protected data set. This approach assumes the data gets out eventually through an internal employee (or data breach) and protects against this worst-case scenario. The primary goal of the privacy methods is to limit the ability of an adversary to identify data subjects (identification disclosure, in our case, discovering the true identity of a time series) and learn sensitive information about them (attribute disclosure, learning sensitive values after an identification is made). The secondary goal of the privacy methods and the primary concern for forecasters is that these privacy methods do not significantly reduce forecast accuracy.

For illustration of the centralized scenario, consider the example shown in Figure 1. The time series in the middle plot is a simulated AR(1) process with autoregressive parameter $\phi_1 = 0.8$. The series on the left is the same simulated series with random noise added to each time period proportional to the standard deviation of the simulated series. The series on the right is generated by randomly swapping the values of the middle series with values from two other simulated AR(1) processes, both with $\phi_1 = 0.8$.

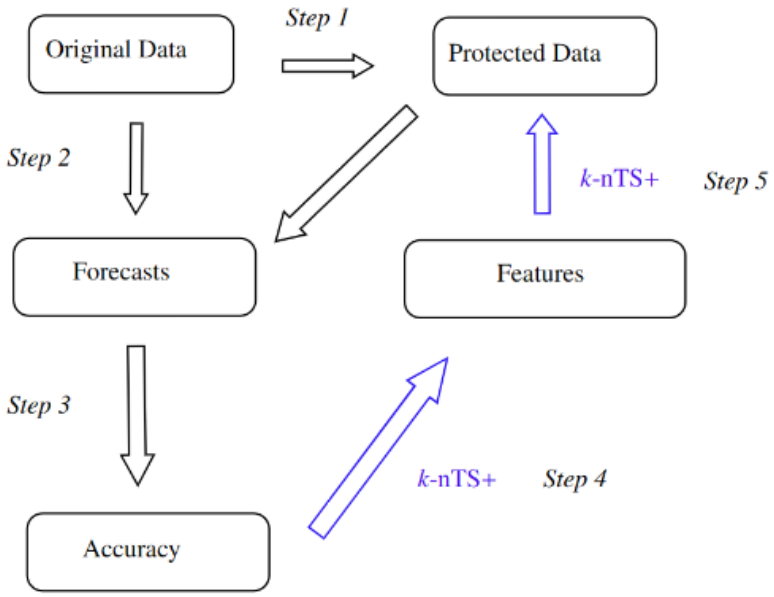
Figure 1: Comparison of protected AR(1) processes to the original AR(1) process.



Estimating an ARIMA(1, 0, 0) model on the simulated series in the middle yields an estimate of $\widehat{\phi_1} = 0.73$ with a standard error of 0.07, while the noised series on the left yields an estimate of $\widehat{\phi_1} = 0.16$ with a standard error of 0.10. The swapped series on the right yields an estimate of $\widehat{\phi_1} = 0.56$ and a standard error of 0.08. Figure 1 shows the series on the right is more representative than the noised version, but in both cases, the time series features (*e.g.*, AR (1) parameter, variance, spike, maximum variance shift, spectral entropy, etc.) from the middle series change. However, it is not immediately clear how these changes affect forecast accuracy.

The proposed method in this paper improves the series on the right by swapping time series values with each other only if their underlying features are likely to maintain forecast accuracy. The k-nTS+ swapping method is illustrated in Figure 2 and begins by generating protected data using baseline privacy methods (*e.g.*, additive noise, differential privacy) commonly used in practice (*Step 1)*. Then, the method generates forecasts for the original and protected series (*Step 2*) and compares their accuracies (*Step 3*). To improve forecast accuracy (blue arrows in Figure 2) of the protected data, k-nTS+ swapping uses a machine learning-based feedback loop with RReliefF (Robnik-Sikonja & Kononenko, 2003) and Recursive Feature Elimination (RFE) (Gregorutti et al., 2017) on the accuracy results to rank (RReliefF) and select (RFE) the time series features most predictive of forecast accuracy (*Step 4*). For privacy protection, it computes a feature-based distance matrix to randomly choose a time series to swap values with (*Step 5*). As a result, the k-nTS+ swapping method produces protected time series that are more likely to preserve useful time series features for forecasting.

Figure 2: k-nTS+ swapping method (Blue arrows indicate the feedback loop which informs the swapping).



Our contributions to the literature are two-fold. First, we measure the changes in forecast accuracy from multiple forecasting models and privacy methods. The literature demonstrated that differential privacy degrades forecast accuracy for VAR models and recurrent neural networks (RNNs) (Gonçalves et al., 2021a), but little work explains why different forecasting models produce different accuracies on protected data. We newly analyze how time series features change vis-à-vis accuracy and show which forecasting models are more robust to changes in time series features after privacy protection. We also examine whether the magnitude, direction, or volatility of these privacy adjusted forecasts improve forecast accuracy.

Second, we propose a matrix-based method that limits the degradation in forecast accuracy due to privacy protection. To the best of our knowledge, previous research does not improve the forecast

accuracy of protected data in a centralized scenario where the entire data set is shared. In the privacy literature, the usefulness of protected data is often overlooked (Blanco-Justicia et al., 2022), but is of utmost concern to forecasters. Recent research (Schneider et. al., 2018) maintained marketing metrics within 10-15% of the original by including a marketing loss function in their privacy method. Similarly, we use time series features predictive of forecast accuracy in our k-nTS+ swapping method. However, including all of the features would significantly increase the dimensionality and reduce the efficiency of a swapping process. To address this problem for our k-nTS+ swapping method, a random forest-based recursive feature elimination (RFE) algorithm can be applied to the features selected by RReliefF. Prior work has shown that random forest-based RFE is efficient when applied to sets of highly correlated features (Gregorutti et al. 2017).

Results show that our method provides significantly better forecast accuracy (+13.9% worse than the original forecasts) at similar levels of privacy to baseline privacy methods. Furthermore, using the *performance gap* from Petropoulos & Siemsen (2022), we show that k-nTS+ protected time series are more representative of the original series, leading to improved trust between data owners and forecasters.

In Section 2, we review the relevant literature. Section 3 describes the *k*-nTS swapping method and proposes the k-nTS+ swapping method with a feedback loop of the time series features. Section 4 presents the empirical application and Section 5 concludes.

2. Literature Review

2.1. Privacy Methods

In a decentralized scenario, Goncalves et al. (2021c) modeled a data market where data owners are compensated for sharing their time series data and purchase only forecasts based on the data from other parties. However, the original time series were still shared with a central party which discourages data owners from sharing time series due to privacy concerns such as a data breach. Other privacy-preserving solutions for collaborative forecasting include secure multi-party computation, decomposition-based methods, and data transformation techniques (see Gonçalves et al. 2021a).

In a centralized scenario, the data owner uses privacy methods to generate protected data sets for forecasting. Gonçalves et al. (2021a) showed that differential privacy reduces the forecast accuracy of VAR models under very high values of the privacy parameter ϵ (weak privacy protection). Others have also studied the application of differential privacy to time series (Imtiaz et al., 2020; Liyue Fan & Li Xiong, 2014). Luo et al. (2018) simulated data integrity attacks and found that multiplicative noise reduces forecast accuracy by over 21% when only half the data points are altered. Their results likely understate the reduction in forecast accuracy from privacy methods because only half the data points were altered.

Other privacy methods include generalization where the structure of the original data set is changed. Data records can be aggregated or combined to make every record (or time series) identical to at least $k - 1$ other records (or time series). For example, daily time series data can be aggregated to weekly time series data (frequency aggregation), or each time series can be averaged with its most similar time series ($k = 2$ anonymity). Using $k = 2$ anonymity (weak privacy), Nin & Torra (2009) evaluated the change in forecast accuracy for simple exponential smoothing, double exponential smoothing, linear regression, multiple linear regression, and polynomial regression.

They found an overall reduction in forecast accuracy but did not provide the accuracy of each model individually. Also, top- and bottom-coding can be used to replace the tails of distributions with a threshold value, such as $150,000 for income or 10 kilowatts-hours for household energy usage). Top- and bottom-coding limit attribute disclosure risk (*i.e.*, preventing knowledge of specific values within a time series), but may not be effective at limiting identification disclosure risk (*i.e.*, preventing the identification of an entire time series). Top- and bottom-coding could have an effect similar to adjusting for outliers which improves forecast accuracy when the outliers are close to the forecast origin (Chen & Liu, 1993).

2.2. Adjusted Forecasts

Privacy methods adjust forecasts by altering the original time series data. Similar to judgmental adjustments, this presents the forecaster with multiple forecasts to choose from. We reference the long history on judgmental forecasting (Petropoulos et al., 2022, see sections 2.11.2 and 3.7.3) investigating how the direction and magnitude of adjustments, and the volatility of forecasts affect forecast accuracy.

There are two critical differences between privacy adjustments and judgmental adjustments. First, judgmental adjustments alter a forecast after it is output from a forecasting model. The underlying time series and their features are not changed. For the direction of the adjustment, Davydenko & Fildes (2013) found that both positive and negative adjustments can improve accuracy, but positive adjustments tend to give only a marginal improvement. Khosrowabadi et al. (2022) similarly found that beneficial positive adjustments tended to be small, and beneficial negative adjustments tended to be large. Fildes et al. (2009) showed that negative adjustments reduce forecast bias, whereas positive adjustments maintain bias or exacerbate it. The magnitude of judgmental adjustments is also positively associated with the size of accuracy improvements when adjustments are based on reliable information. For volatility, accuracy improvements are greater for time series that have forecast errors with low volatility, presumably because adjusters struggle to assess the effect of future events accurately when a time series is more volatile (Fildes et al., 2009).

Second, the motivation for judgmental adjustments is different. Motivations include gaining control of the forecasting process, incorporating practitioner expectations, and compensating for judgmental biases (Petropoulos et al., 2022, sec. 3.7.3). The goal is to incorporate the intuition and experience of the adjuster, knowledge of special events, or insider or confidential information to improve forecast accuracy (Fildes et al., 2019). Despite varying motivations, judgmental adjustments have been shown to improve forecast accuracy by 5-10% on average (Davydenko & Fildes, 2013; Khosrowabadi et al., 2022). For privacy adjustments, the goal is to improve privacy by blurring the data. The assumption is that forecast accuracy will not improve – instead, utility (forecast accuracy) will tradeoff with privacy (Duncan & Stokes, 2004).

2.3. Time Series Features for Forecast Accuracy

Thousands of features have been used for time series classification (Fulcher & Jones, 2014) and a subset of those are useful for forecast accuracy. Bandara et al. (2018) clustered similar time series based on eighteen interpretable features, including the mean, variance, and strength of seasonality to improve the accuracy of RNNs between 2 and 11%. The initial results from the M4 competition suggested that the randomness and linearity of time series were the most important determinants of forecast accuracy and that seasonal time series (typically less noisy) are easier to forecast (Makridakis et al., 2018). In a follow-up study, Spiliotis et al. (2020) used multiple linear regression

to confirm the importance of randomness, linearity, and seasonal strength in predicting mean absolute scaled error (MASE) values of the ETS, ARIMA, Theta, and Naïve 2 (random walk applied to seasonally adjusted data) models from the M4 competition. They found that increasing the frequency, kurtosis, linearity, and seasonal strength of time series improved forecast accuracy, but increasing skewness, self-similarity, and randomness degraded forecast accuracy.

Time series features are also used for model selection and forecast combination. Qi et al. (2022) found that forecasts using the strength of trend and seasonality for exponential smoothing model selection had lower errors across multiple forecast accuracy metrics than information-based selection methods for the majority of forecast horizons. Talagala et al. (2022) applied a meta-learning algorithm based on Bayesian multivariate surface regression to 37 features, including spectral entropy and the Hurst exponent, to predict the model combination that would yield the minimum forecast error for the M4 competition data. This approach achieved forecast accuracy on par with the top M4 competition methods with less computational cost. Li et al. (2022) used features such as the first ACF value to propose an interpretable Bayesian forecast combination framework with time-varying weights. In experiments using the M3 competition data, this method reduced the MASE by approximately 1.1% relative to the next-best forecast combination method. Petropoulos & Siemsen (2022) created a representativeness metric that selects models with trend and seasonality components when the respective signals of these components are strong. For most data frequencies, their approach lowered MASE on the M, M3, and M4 competition data and selected the best forecasting model approximately 3% more often than the other selection methods.

3. The k-nearest Time Series (k-nTS) Swapping Method

We solve the data protection problem for the data owner using a matrix-based k-nTS (k-nearest time series) swapping method, where the data owner releases a set of protected time series $\mathbb{X}' = \{x_1', \dots, x_J'\}$ where $x_j' = \left(P_{j,1}, \dots, P_{j,t}\right)^T$ is based on $\mathbb{X} = \{x_1, \dots, x_J\}$, the original values of all series through time t. To create a protected series x_j', the k-nTS swapping method finds the k most similar time series to x_j where similarity is based on the time series features. For each period t, it randomly chooses one of the k similar series to x_j and replaces $A_{j,t}$ with the original value at time t from the randomly chosen series.

Depending on the quantity of available data, k-nTS swapping can use rolling windows of data that adjust for dynamic changes in time series features. For example, if we choose a rolling window of size n, then $x_j = \left(A_{j,t-n+1}, A_{j,t-n+2}, \dots, A_{j,t-1}, A_{j,t}\right)^T$ where $x_j \in \mathbb{R}^n$. Protection in subsequent periods from $t + 1$ to T rolls x_j forward by one time period. We label the time series features for the current window as $f_{j,t}$ which we refer to as the feature vector for time series j in time period t based on the n values in x_j. For simplicity, we omit the t subscript for the feature vectors and write f_j.

For each time series $x_j \in \mathbb{R}^n$, the data owner computes the feature vector $f_j \in \mathbb{R}^m$. This vector can contain any single-valued feature calculated from the values in x_j, such as the strength of the trend and seasonality, the spectral entropy, or the mean value of the current window. Let $\mathbb{C} = \{f_1, \dots, f_J\}$ be the set of m-vectors containing the features from each of the J time series windows. For each f_j, the data owner computes a set of squared distances of the elements of $\mathbb{C}$. We define $dist\left(f_j, f_i\right) = d_{j,i}$ as the distance between f_j and f_i, *i.e.*, the feature vectors corresponding to two distinct time

series from $\mathbb{X}$. Without loss of generality, we use the Euclidean norm, or ℓ_2-norm, as a distance metric[3]. Since our case is multivariate and partially ordered, we can get a totally ordered set based on the Euclidean distance.

We define $x_j^{(k)}$ as the kth nearest neighbor of x_j, with the corresponding feature vector $f_j^{(k)}$. Then, for a time series x_j, we have $\{d_{j,(1)}, d_{j,(2)}, \dots, d_{j,(J-1)}\}$ such that $d_{j,(k)} \leq d_{j,(l)}$ for any integers $k < l$ where $d_{j,(k)} = \left\| f_j - f_j^{(k)} \right\|$. Note that $x_j^{(i)} \in \mathbb{X}\backslash\{x_j\}$ and the superscript (i) means the ith order statistic of the related Euclidean distances of all $f_j^{(i)} \in \mathbb{C}\backslash\{f_j\}$ from f_j. Thus, for a given time series vector x_j, its k-nearest time series can be represented as the set $K_j = \{x_j^{(1)}, \dots, x_j^{(k)}\}$ based on an ordered set $\{d_{j,(1)}, d_{j,(2)}, \dots, d_{j,(k)}\}$.

For more efficient computation, we introduce a symmetric distance matrix D containing the squared distances between time series feature vectors. The squared distance between f_i and f_j is given by $d_{i,j}$, that is the (i, j)th entry of D (also note that rank$(D) \leq m + 2$). Suppose we have an original data matrix $X = [x_1, x_2, \dots, x_J]$, where $x_j \in \mathbb{R}^n$ (*i.e.*, $X \in \mathbb{R}^{n \times J}$). We calculate the desired features based on each x_j and construct a feature matrix C (where $C \in \mathbb{R}^{m \times J}$) as follows:

$$C = [f_1, f_2, \dots, f_J] = \begin{pmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & f_{1,m-1} & f_{1,m} \\ f_{2,1} & f_{2,2} & f_{2,3} & \cdots & f_{2,m-1} & f_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ f_{J,1} & f_{J,2} & f_{J,3} & \cdots & f_{J,m-1} & f_{J,m} \end{pmatrix}^T.$$

Algorithm 1: The k-nTS Swapping Method

Require [Initialization]:

$X = [x_1, \dots x_J]$: the $n \times J$ matrix of original time series.
$C = [f_1, \dots, f_J]$: the $m \times J$ matrix of time series features.
$D = \mathbf{1}diag(C^TC)^T - 2C^TC + diag(C^TC)\mathbf{1}^T$: the feature distance matrix.

for $j = 1, 2, \dots, J$ **do**

i. Find the set K_j for x_j by sorting the jth column of D from smallest to largest and finding the kth smallest component.
ii. Replace the last component of x_j with the last component of $x_j^{(i)}$ for a randomly chosen $i \in \{1, \dots, k\}$.

end for

We calculate the matrix D using the fact that $\|f_i - f_j\|^2 = (f_i - f_j)^T(f_i - f_j) = f_i^T f_i - f_i^T f_j - f_j^T f_i + f_j^T f_j$, which can be written as the following:

3 All norms on $\mathbb{R}^n$ are equivalent to the Euclidean norm.

$$D = \mathbf{1}\text{diag}(C^T C)^T - 2C^T C + \text{diag}(C^T C)\mathbf{1}^T,$$

where $\mathbf{1}$ denotes a column vector of J ones[4]. It is easy to see that the column vector $diag(C^T C) = \left(\|f_1\|^2, \ldots, \|f_J\|^2\right)^T$. Let d_j denote the jth column of D. Then we can write the $J \times J$ distance matrix $D = [d_1, \ldots, d_J]$, where $d_j \in \mathbb{R}^J$. In the general case where $k \ll J$, for each time series x_j we sort d_j and take the k smallest components so that we have

$$K_j = \{x_j^{(1)}, \ldots, x_j^{(k)}\}.$$

That is, the data owner selects a value of k from 1 to a maximum of $J - 1$ and selects the k-nearest time series to x_j based on the m features. Let the ith most similar time series to x_j be $x_j^{(i)}$. Swapping the last component of x_j with the last component of one of its k-nearest time series $x_j^{(i)}, i = 1, \ldots, k$, is:

$$P_{j,t} = A_{j,t}^{(i)} \text{ with probability } 1/k \text{ for } i = 1, \ldots, k.$$

By Algorithm 1, we can obtain X', a matrix of protected time series data through time t for all J time series. All of the first n values of each time series are swapped based on the k-nearest time series to the first rolling window. For each successive time period, the window is rolled forward, the k-nearest neighbors are re-calculated, and swapping is performed on a rolling basis. The output of the k-nTS swapping method can be written as the following protected data matrix,

$$X' = [x'_1, x'_2, \ldots, x'_J] = \begin{pmatrix} P_{1,1} & P_{1,2} & P_{1,3} & \cdots & P_{1,T-1} & P_{1,T} \\ P_{2,1} & P_{2,2} & P_{2,3} & \cdots & P_{2,T-1} & P_{2,T} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{J,1} & P_{J,2} & P_{J,3} & \cdots & P_{J,T-1} & P_{J,T} \end{pmatrix}^T.$$

3.1. The *k*-nearest Time Series + (k-nTS+) Swapping Method

The k-nTS+ swapping method adds a feature selection process to k-nTS swapping which selects features that are good predictors of forecast accuracy. The goal is to obtain a small set of features that predict forecast accuracy well. k-nTS+ swapping can be used collaboratively between a data owner and the forecaster. The forecaster specifies their preferred forecasting model $\mathcal{F}$, and the data owner applies the model to the original and protected data up through time period $T - 1$, assesses which features are most predictive of accuracy for the specified model, and releases protected data to the forecaster using k-nTS+ based on these features up through time period T. The data owner can repeat this process over successive time periods for multiple data releases at times $T + 1$ and beyond. Algorithm 2 specifies the k-nTS+ swapping method below. A version of Algorithm 2 with more mathematical detail can be found in the Appendix.

---

4 Note that we could also define a distance matrix based on the actual time series values x_j, where D would become a function of X rather than C.

Algorithm 2: The k-nTS+ Swapping Method

Require [Initialization]:

$X = [x_1, \dots x_J]$: the $T \times J$ matrix of original time series.
$P = \{\mathcal{P}_1, \dots, \mathcal{P}_B\}$: the B baseline privacy methods.
$\mathcal{F}$: the desired forecasting model.
$\mathcal{M} = \{m_1, \dots m_N\}$: the initial set of time series features' names.
$\mathcal{K}$: the number of nearest neighbor time series to consider for RReliefF.
N_{rfe}: the number of recursive feature elimination iterations
τ^*: recursive feature elimination prediction error threshold
k: the number of nearest neighbor time series to consider for swapping.

Step 1: Create Baseline Protected Datasets

Use baseline privacy methods P to create protected data sets through time $T - 1$

Step 2: Generate Baseline Forecasts

Generate forecasts using $\mathcal{F}$ for time T for the original and baseline protected data sets

Step 3: Measure Forecast Accuracy

Compute the accuracy of forecasts at the series level for the original and protected data sets

Step 4: Compute and Select Time Series Features

i. Compute time series features $\mathcal{M}$ for the original and protected data sets through time $T - 1$
ii. Use RReliefF (Robnik-Sikonja & Kononenko, 2003) with nearest-neighbor parameter $\mathcal{K}$ to weight the ability of features to discriminate between series with different forecast accuracies
iii. Drop features with RReliefF weights $W_n \leq 0$
iv. **for** $\boldsymbol{iter} = \mathbf{1}, \dots N_{rfe}$ **do** [*use RFE to select most important features*]
    a. Train a random forest to predict forecast accuracy using the features from RReliefF
    b. Drop the least important feature
    c. Repeat (a.) and (b.) until one feature remains

    end for
v. Rank the RReliefF features from least to most important using the average order of elimination across the N_{rfe} iterations of RFE
vi. Select the minimum number of features required in the random forest model to attain prediction accuracy within $\tau^*\%$ of the random forest model with the best predictive accuracy

Step 5: Create Protected Data Set using k-nTS+ Swapping

Use input k and the features selected in *Step 4* to perform swapping through time T using **Algorithm 1: The k-nTS Swapping Method.**

Output: k-nTS+ protected time series data set X'.

4. Empirical Application

4.1. Data

The organizers of the early M competitions did not disclose the true identity of the time series used in their competitions (Makridakis & Hibon, 2000). For our application, this provides a natural connection to privacy because we can compute the identification disclosure risk of each protected time series. We define identification disclosure risk as the probability of matching a protected time series to its original time series in the original data set. Good privacy implies the identification disclosure risk is low or similar to random guessing. To be conservative, we assume that an adversary (possibly a forecaster) has external data on at least one original time series and attempts to match it to the protected time series. The data owner seeks to alter the time series with privacy methods to reduce the identification disclosure risk while maintaining as much forecast accuracy as possible.

Recent work by Spiliotis et al. (2020) shows that the M3 competition data contain time series features representative of the real world data which also makes it suitable for our feature-based k-nTS+ swapping method. As a result, we use the monthly micro dataset from the M3 competition, which includes 474 strictly positive time series with values ranging from 120 to 18,100. Of the 474 series, 18 consist of 67 time periods, 259 consist of 68 time periods, and 197 consist of 125 time periods. The data owner protects every single time series value from time period 1 to T. The protected time series are given to forecasters to produce one-step ahead forecasts for time T + 1. The data owner then measures forecast accuracy using the protected and original data against the actual values from T + 1.

We assume the forecaster may be an adversary attempting to identify an original time series by using the protected time series. For calculating identification disclosure risk (see subsection 4.3.4 for further details), we take the most conservative approach and assume that the adversary knows the length of each original series which makes identification easier. Thus, we separate the privacy analysis and protection into three groups of 18, 259, and 197 time series. Random guessing averages out to approximately 0.6% ($\frac{1}{18}$ for the 18 series, $\frac{1}{259}$ for the 259 series, and $\frac{1}{197}$ for the 197 series).

The rest of the empirical application is outlined as follows. Subsection 4.2 defines the time series features used for the k-nTS+ swapping method, subsection 4.3 describes the privacy methods and defines the identification disclosure risk, subsection 4.4 presents the privacy and forecast accuracy results, and subsection 4.5 analyzes how the time series features change after privacy protection. Subsection 4.5 also analyzes whether the volatility of the original time series and magnitude or direction of privacy adjustments maintained forecast accuracy.

4.2. Time Series Features for Forecast Accuracy

Table 1 displays the time series features selected for k-nTS and k-nTS+ swapping. For k-nTS swapping, we selected time series features that had a relationship with forecast accuracy based on the literature review in Section 2.3. We omit stability and non-linearity since these features had little to no effect on accuracy. We also omit frequency because none of the privacy methods we consider change the frequency (monthly) of the original data. For k-nTS+, we include many additional features from the *tsfeatures* package in R including *Spike*, *Max Variance Shift*, and *Max*

Level Shift. We refer the reader to Hyndman et al. (2022) for a detailed explanation of these features and further mathematical detail on the time series features is provided in the Appendix.

Table 1: Time series feature descriptions and value ranges.

Feature	Description	Value Range	Selected (Literature)	Selected[5] (k-nTS+)
Spectral Entropy	Signal-to-noise ratio of the time series.	[0, 1]	X	
Hurst	Long-range dependence (self-similarity) of a time series.	[0, 1]	X	
Skewness	Symmetry of the distribution of time series values.	$(-\infty, \infty)$	X	
Kurtosis	Weight of the tails of the distribution of time series values.	$(-\infty, \infty)$	X	
Error ACF	First autocorrelation coefficient of the error component of the decomposed series.	[-1, 1]	X	
Trend	Strength of the trend.	[0, 1]	X	X
Seasonality	Strength of the seasonality.	[0, 1]	X	
Mean	Mean of the time series.	$[0, \infty)$	X	X
Variance	Variance of the time series.	$[0, \infty)$	X	X
Spike	Variance of the leave-one-out variances of the remainder component of the decomposed series.	$[0, \infty)$		X
Max Variance Shift	Largest variance shift between two consecutive sliding windows.	$[0, \infty)$		X
Max Level Shift	Largest mean shift between two consecutive sliding windows.	$[0, \infty)$		X

4.3. Privacy Protection

4.3.1. Differential Privacy

We follow the interpretation and implementation of differential privacy from Gonçalves et al. (2021a). A mechanism M satisfies ϵ-differential privacy by guaranteeing that, for every output x' of M and every pair of series x_j and x_i which differ on at most one observation,

$$Pr\big(M(x_j) = x'\big) \leq exp(\epsilon) Pr(M(x_i) = x').$$

[5] These features were selected from the results in Section 4.4.

A differentially private time series can be created using a randomized mechanism $M(x_j) = x_j + \eta$ that adds a vector of random noise values, each of which is drawn from a Laplace distribution with scale parameter $\Delta f_1/\epsilon$, to an original time series x_j. The sensitivity Δf_1 is determined as the maximum absolute difference between two time series x_j and x_i, which differ in at most one observation, where $\Delta f_1 = max\|x_i - x_j\|_1$. We use the values of $\varepsilon = 20.0, 10.0, 4.6, 1.0, 0.1$ which ranges from least private to most private.

4.3.2. Additive Noise

Additive noise adds a normally distributed random number with mean zero and standard deviation σ to each value in an original time series x. Protected values can be written $P_t = A_t + r$, where $r \sim N(0, \sigma^2)$ and $\sigma = s * \sigma_{x_j}$. The protection parameter s denotes the number of standard deviations of x_j and we set $s = 0.25, 0.50, 1.0, 1.5, 2.0$ which ranges from least private to most private.

4.3.3. k-nTS and k-nTS+

The k-nearest series for k-nTS and k-nTS+ swapping are determined using features computed by the data owner from the original data set. The data owner swaps original values to create the protected values from time 1 to T for each time series. We use the features described in Section 4.2.

To perform feature selection for *k*-nTS+, the data owner first creates protected versions of the original data using additive noise and differential privacy from time period 1 to $T - 1$. Then, using this protected data from time period 1 to $T - 1$, the data owner generates forecasts for each of the protected data sets at time T and computes the absolute error of each forecast for each series. The k-nTS+ swapping method is applied to each forecasting model separately in order to detect the variation in forecast accuracy due to changes in time series features (and not the forecasting model).

We select $\tau = 5\%$ so the features are within 5% of the minimum average prediction error from the best random forest model. For the k-nTS+ protected data, we use the six features (last column of Table 1) with the highest average rank across the RFE iterations for all forecasting models with $k = 3, 5, 7, 10, 15$. Next, the data owner uses these six features selected by this feedback to swap all series for time periods 1 to T. The data owner shares this protected data with forecasters who forecast time period $T + 1$.

4.3.4. Identification disclosure risk

As previously mentioned, the forecasters of the M3 competition did not know the identities of the original time series. For our privacy metric, we assess the ability of each privacy method to protect against *identification disclosure*, which occurs when an adversary correctly predicts the identity of a protected time series. Each protected data set consists of the protected series along with a pseudo identifier, *i.e.*, $X = \left[(PID_1, x_1), (PID_2, x_2), \ldots (PID_J, x_J)\right]^T$. The pseudo identifier in our application is the `Series` column from the original M3 data, which contains a PID for each time series, *e.g.*, `N1402`. Identification disclosure occurs if an adversary (or forecaster) correctly predicts the identity of one or more of the time series in the M3 data set based on the protected time series and some outside information the adversary possesses. For example, identification disclosure occurs when an adversary correctly states, "Series N1402 comes from the monthly sales of the Roseville, Minnesota Target store."

We perform $S = 20$ simulations of a privacy attack in which an adversary uses original time series values to identify the protected time series. In each simulation, we sample ten sequential values from each original time series and treat these as external information available to the adversary. The adversary predicts the identity of each protected series based on which original values are closest to the protected values from the same time periods.

The metric we use is identification disclosure risk for time series (Nin & Torra, 2006, 2009), $\bar{P}$, the average proportion of the J time series which are correctly identified across the S simulated privacy attacks,

$$\bar{P} = \frac{1}{J \times S} \sum_{s=1}^{S} \sum_{i=1}^{J} I\left(\widehat{M}_i^s = j^*\right)$$

where $\widehat{M}_i^s$ is the adversary's prediction of the identity of the ith protected time series. We use $I(*)$ to denote the indicator function which is equal to one when identification disclosure occurs, *i.e.*, when the predicted identity is equal to the true identity j^*. We refer the reader to the Appendix for added mathematical details.

4.4. Results

For all privacy methods, we generate one-step ahead forecasts for time T+1 using off-the-shelf models in R and Python shown in Table 2. Similar to the M3 Competition, all reported forecast accuracy and standard deviation results are derived from comparing the forecasts for T + 1 to the actual data from T + 1. Reported privacy results are derived from calculating the identification disclosure risk using the protected data from time period 1 to T. Also, the LGBM and RNN forecasting models and the VAR model are trained separately on the three subsets of 18, 259, and 197 time series. We perform minimal data pre-processing and use the standard settings in the off-the-shelf packages. [6]

Table 2: Univariate and Multivariate Forecast Models

Model Name	**Variant**
SES	-
DES	Additive trend
TES	Additive trend/seasonality
Auto-ARIMA	Seasonal
VAR	-
LGBM	-
RNN	LSTM

Table 3 displays the average MAE of one-step ahead point forecasts across all models and series, the identification disclosure metric $\bar{P}$, and the average performance gap across all series. The percentages in parentheses are the increase in average MAE relative to the average MAE from the

[6] Implementation details can be found in the appendix.

original data. The results show an inverse relationship between forecast accuracy and the strength of privacy protection. While strong differential privacy provides the lowest identification disclosure risk at 1.85% (random guessing is 0.6%), it nearly quintuples (+383%) the average forecast error relative to the original data resulting in unusable forecasts. Under weak differential privacy with $\epsilon = 10$, over 49% of series are identified correctly on average, which is poor identification disclosure risk. Protection against identification disclosure is better under additive noise with $s = 1$ where 22.5% of series are correctly identified on average. However, this comes at a cost to forecast accuracy, which degrades by nearly 45%.[7]

[7] The averages for additive noise and differential privacy excludes the VAR model error for AN (s = 1) and DP ($\epsilon = 0.1$) since the errors were over 1000% larger than the error of any other model. Due to the large noise infused from these privacy methods, the VAR could not fit small enough coefficients to smooth out the noise, resulting in extremely poor forecast accuracy. For example, the magnitude of the first lag coefficient for an AN (s = 1) protected time series increased from -0.372 in the original data to -0.679 in the protected data. This coefficient was multiplied by an extreme outlier at time T causing the forecast at time $T + 1$ to explode and skew the overall average forecast error. This problem did not occur for the other forecasting models, which did a better job smoothing out the random noise.

Table 3: Identification disclosure risk, forecast accuracy, and representativeness for original and protected data sets.

Privacy Method	Parameter Value	Privacy (Identification Disclosure Risk)	Accuracy (MAE)	Representativeness (Performance Gap)
Original Data	**-**	100.0%	685.71 (0.0%)	42.7
***k*-nTS+**	**15**	2.7%	839.8 (+22.5%)	73.0
	7	3.5%	822.3 (+19.9%)	66.5
	3	3.3%	781.0 (+13.9%)	62.1
***k*-nTS**	**15**	1.6%	1066.2 (+55.5%)	106.3
	7	2.1%	987.0 (+43.9%)	100.4
	3	2.1%	956.9 (+39.6%)	92.9
Differential Privacy	**1.0**	1.9%	3310.3 (+382.8%)	1,826,437.0
	4.6	13.6%	1401.0 (+104.3%)	311,037.7
	10	49.0%	899.4 (+31.2%)	78,456.4
Additive Noise	**2.0**	5.8%	1821.4 (+165.6%)	503,658.2
	1.5	10.4%	1343.3 (+95.9%)	326,834.5
	1.0	22.5%	994.0 (+45.0%)	166,171.2

k-nTS swapping with $k = 3$ offers a good identification disclosure risk of 2.1%, but forecast accuracy degrades by 39.6%. Our proposed method of *k*-nTS+ swapping with $k = 3$ provides similar levels of protection against reidentification (3.3%) with a reduction in forecast accuracy of only 13.9%. Part of this improvement in forecast accuracy at a minimal tradeoff to identification disclosure risk is due to the incorporation of the accuracy feedback loop for selecting time series features. Thus, we recommend data owners to use our k-nTS+ swapping method (k=3) with the selected time series features to balance the tradeoff between privacy and forecast accuracy.

Forecasters also prefer protected data that are representative of the original time series. Representativeness improves trust between data owners and forecasters and makes it more likely for forecasters to use protected data. Table 3 displays the *performance gap* of Petropoulos & Siemsen (2022) to measure the distance between the protected and original time series values,

performance gap $= \left\| x_j - x_j' \right\|_1$, which is calculated after applying a Box-Cox transformation and scaling the original and protected series. Note that our results in Table 3 differ from Petropoulos & Siemsen (2022) where the performance gap is calculated using the fitted values of forecasting models relative to the training data (which we include in the first row of Table 3). The results show that *k*-nTS and k-nTS+ swapping produce protected time series with the smallest performance gaps by a large margin. However, we note that the average performance gap across series (62.1 for k-nTS+ with k=3) is significantly larger than the average performance gap (42.7) of the fitted values across all series and forecasting models.

Table 4 displays the ranks of the MAE and forecast error variance across all forecasting models using the original data and k-nTS+ swapping with $k = 3$. Past research found that complex forecasting models forecast more accurately than simple models using the monthly micro data (Koning et al., 2005). The results show that k-nTS+ swapping preserves the ranking of the best and worst models on MAE. Univariate models (SES and DES) moved up in the ranking and more complex models (Auto-ARIMA and RNN) moved down.

Table 4: Ranks of MAE and standard deviation of forecast error for the original data and the k-nTS+ swapping (*k*=3) data.

	MAE Ranks		Standard Deviation of Forecast Error Ranks	
Model	Original	Protected	Original	Protected
TES	1 (637.90)	1 (731.30)	2 (859.30)	4 (920.57)
Auto-ARIMA	2 (646.07)	4 (764.83)	1 (834.78)	1 (897.67)
RNN	3 (665.38)	5 (783.15)	5 (883.86)	5 (966.35)
DES	4 (680.54)	2 (743.68)	3 (866.35)	2 (901.22)
SES	5 (686.71)	3 (752.08)	4 (867.13)	3 (914.20)
LGBM	6 (709.48)	6 (809.00)	7 (919.67)	6 (982.35)
VAR	7 (773.90)	7 (883.07)	6 (892.62)	7 (998.08)

4.5. Analysis of Time Series Features

4.5.1. Importance of Time Series Features

Let f_j^{κ} denote one of the $\mathcal{K}$ nearest neighbor feature vectors to f_j, where $\mathcal{K}$ is the number of nearest neighbors considered by RReliefF, and let ϵ_j^{κ} and ϵ_j denote the forecast errors for the corresponding time series. Let π_ϵ and π_m denote the events that series x_j and x_j^{κ} have different forecast errors and different values for feature m, respectively, conditional on being nearest neighbors. The RReliefF weight for feature m approximates the difference in conditional probabilities,

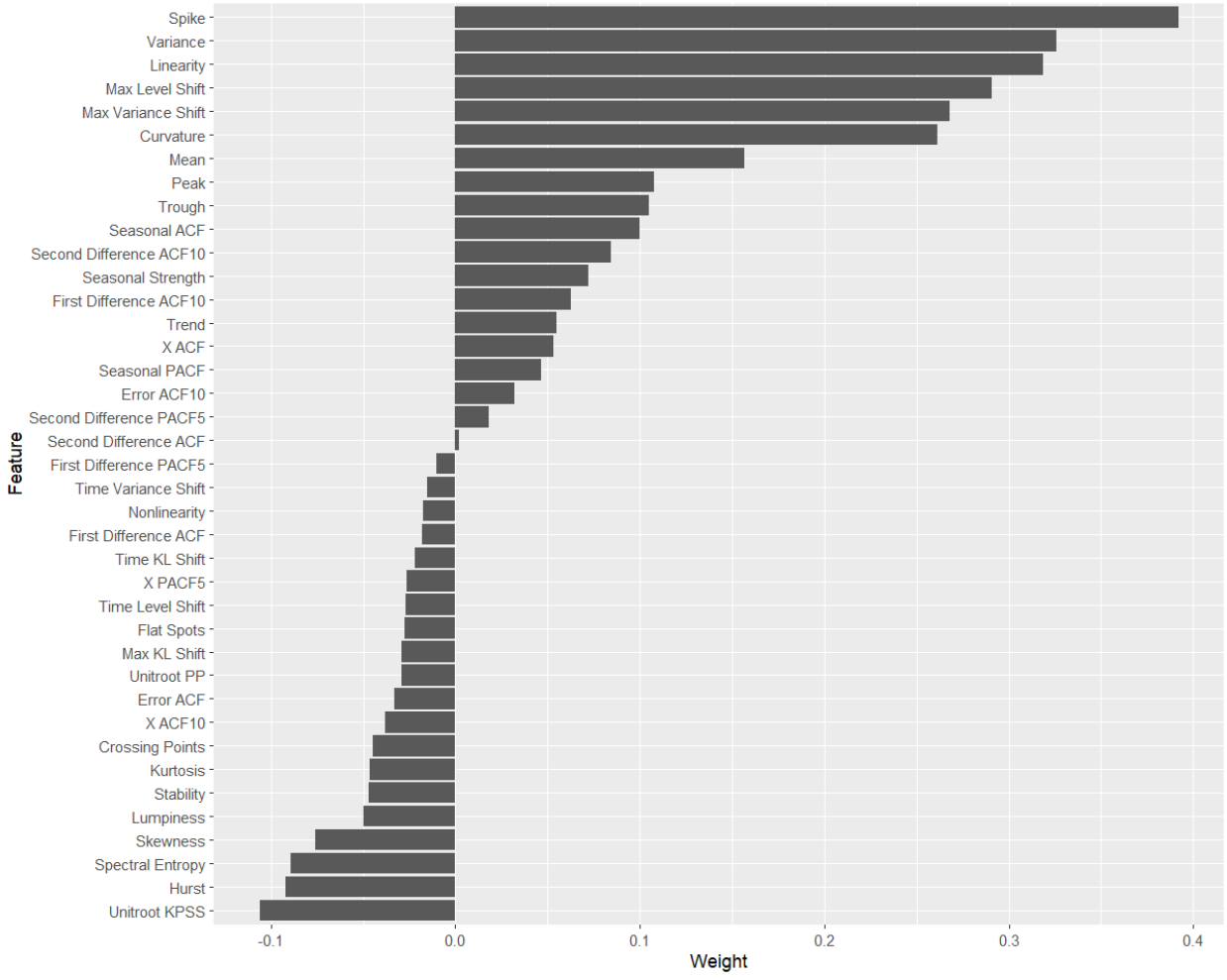
$$W_m = p(\pi_m|\pi_\epsilon) - p(\pi_m|\pi_\epsilon^c).$$

The RReliefF weights approximate the difference between the probability that feature m discriminates between series with different forecast errors, and the probability that feature m discriminates between series with the same forecast error. Features with $W_m > 0$ have a higher probability of varying across series with different forecast errors than varying across series with

similar forecast errors. If we swap using features with $W_m > 0$, we will maintain the values of these features throughout the swapping process and maintain forecast accuracy.

Figure 3 shows the RReliefF weights for each of the 39 features averaged across all forecasting models. RReliefF was used to predict the absolute forecast errors for each model and series across the original and protected data sets. Surprisingly, *Hurst* and *Spectral Entropy* had negative weights which implied they were not useful to maintain forecast accuracy for swapping in the protected data. On the other hand, *Spike*, *Variance*, *Linearity*, *Max Level Shift*, *Max Variance Shift,* and *Curvature* had large positive weights and were important to maintain forecast accuracy.

Figure 3: RReliefF weights averaged across the results of each forecasting model.



4.5.2. Selection of Time Series Features

Figure 4 presents the number of features included for each forecasting model after k-nTS+ eliminated features with negative weights that were poor predictors of forecast error. Over 50 iterations, most of the reduction in OOB MSE occurred using five or fewer features for all forecasting models.

Fig. 4: Average OOB MSE across feature subset sizes when predicting the MAE of each forecasting model.

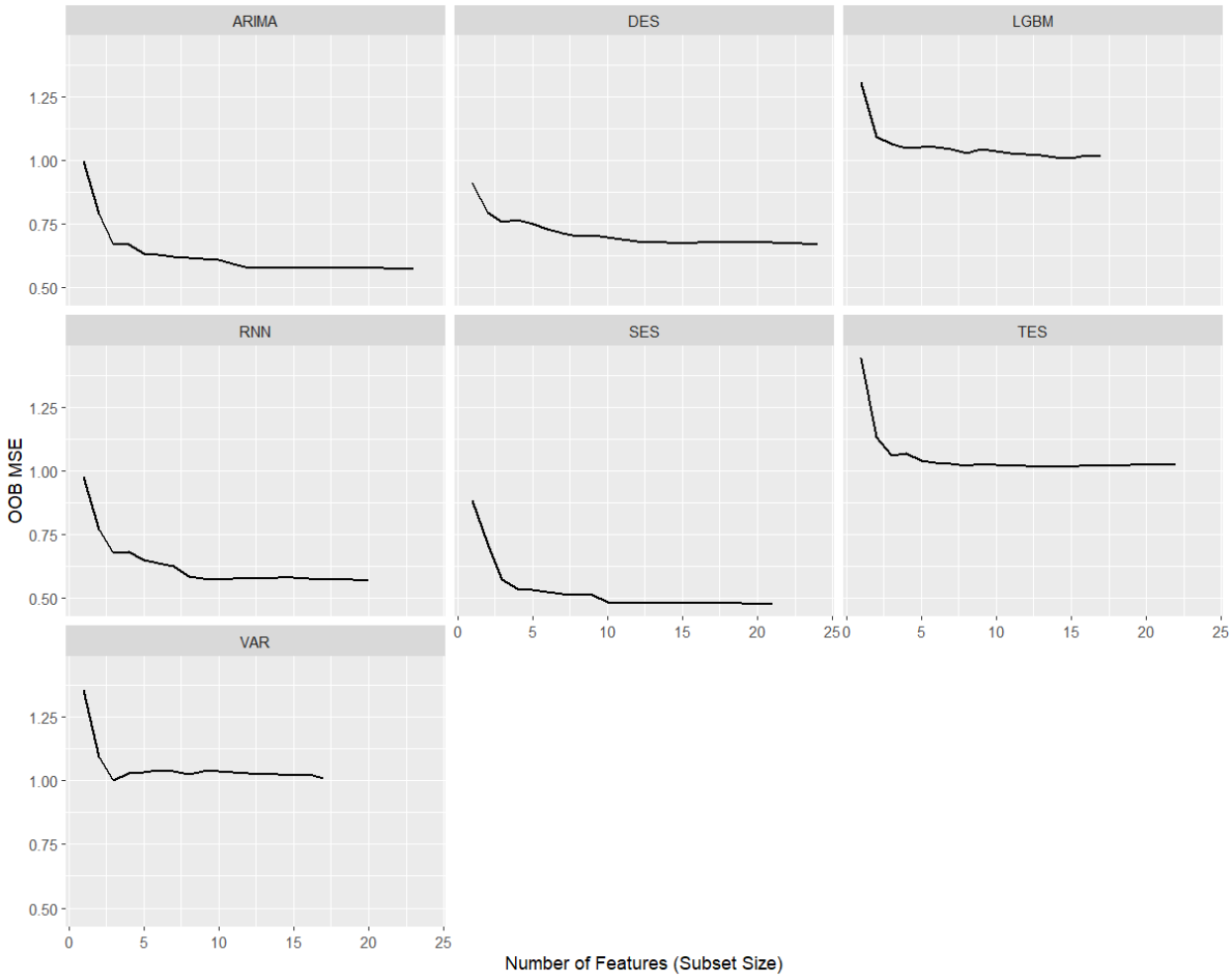
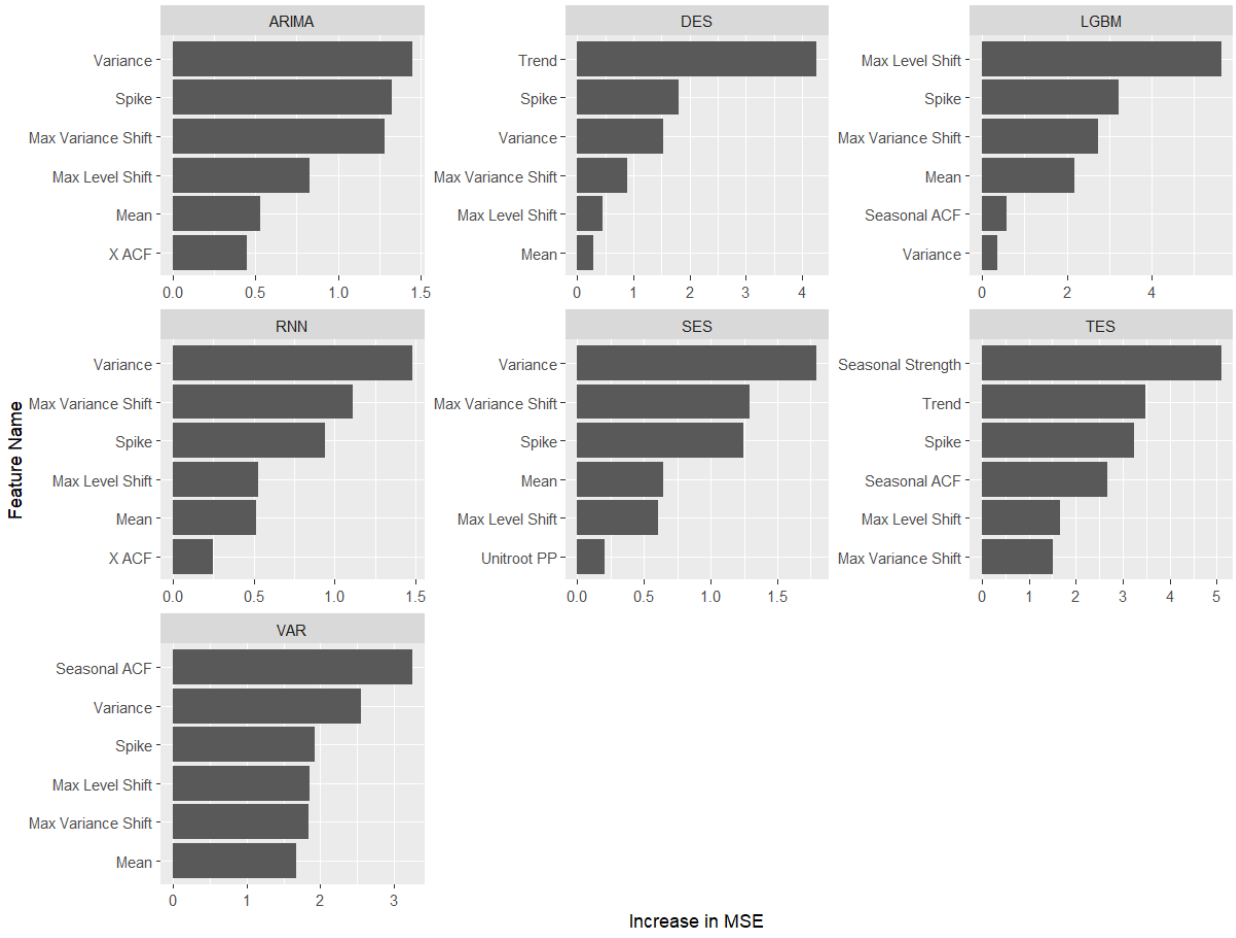


Figure 5 summarizes the results of RFE in Figure 4 and displays the permutation-based importance values for each forecasting model's six most highly ranked features. Some features, such as *spike, max variance shift, max level shift, mean,* and *variance* are highly ranked across most or all forecasting models. Other features appear to be highly important only for specific forecasting models. Examples include *trend*, which is required for DES and TES, *seasonal strength*, which is required for TES, and *X ACF* (the first autocorrelation coefficient of the time series), which is important for Auto-ARIMA and RNN.

Figure 5: Permutation-based importance for the top six features for each forecasting model.



4.5.3. Illustration of Changes in Time Series Features After Protection

Figure 6 displays two monthly time series from the M3 monthly micro data with desirable and undesirable features. After applying the privacy methods to the original time series, Figure 6 illustrates the results using k-nTS+ with $k = 3$ and additive noise with $s = 1$. We can see that for k-nTS+ with k=3, there is little visual change for the undesirable series. For additive noise, there are drastic changes to both series.

Table 5 displays the values of the time series features before and after protection. Table 5 shows that the low spectral entropy and high Hurst coefficient values of the desirable time series indicate good forecastability. Table 5 shows that the undesirable series is essentially a random walk as indicated by the 0.50 value of the Hurst coefficient. Furthermore, the undesirable series has a spectral entropy of 1 indicating a low signal-to-noise ratio. When comparing the two series, the variance of the desirable series is due to a forecastable trend, whereas the variance of the undesirable series is due to randomness. The desirable series also has low *Kurtosis* with a light tailed distribution compared to the undesirable series. One interesting finding is that the k-nTS+ (k=3) version of the desirable series has a lower *Variance* than the original series. However, the higher (long run) variance of the original series is due to the strong trend. Figure 7 shows the short

run month-to-month variance of the k-nTS+ protected series is higher than the original series, as indicated by the values of *Max Variance Shift* in Table 5.

Figure 6: Comparison of original, AN (s = 1), and *k*-nTS+ (*k* = 3) protected series with desirable and undesirable features.

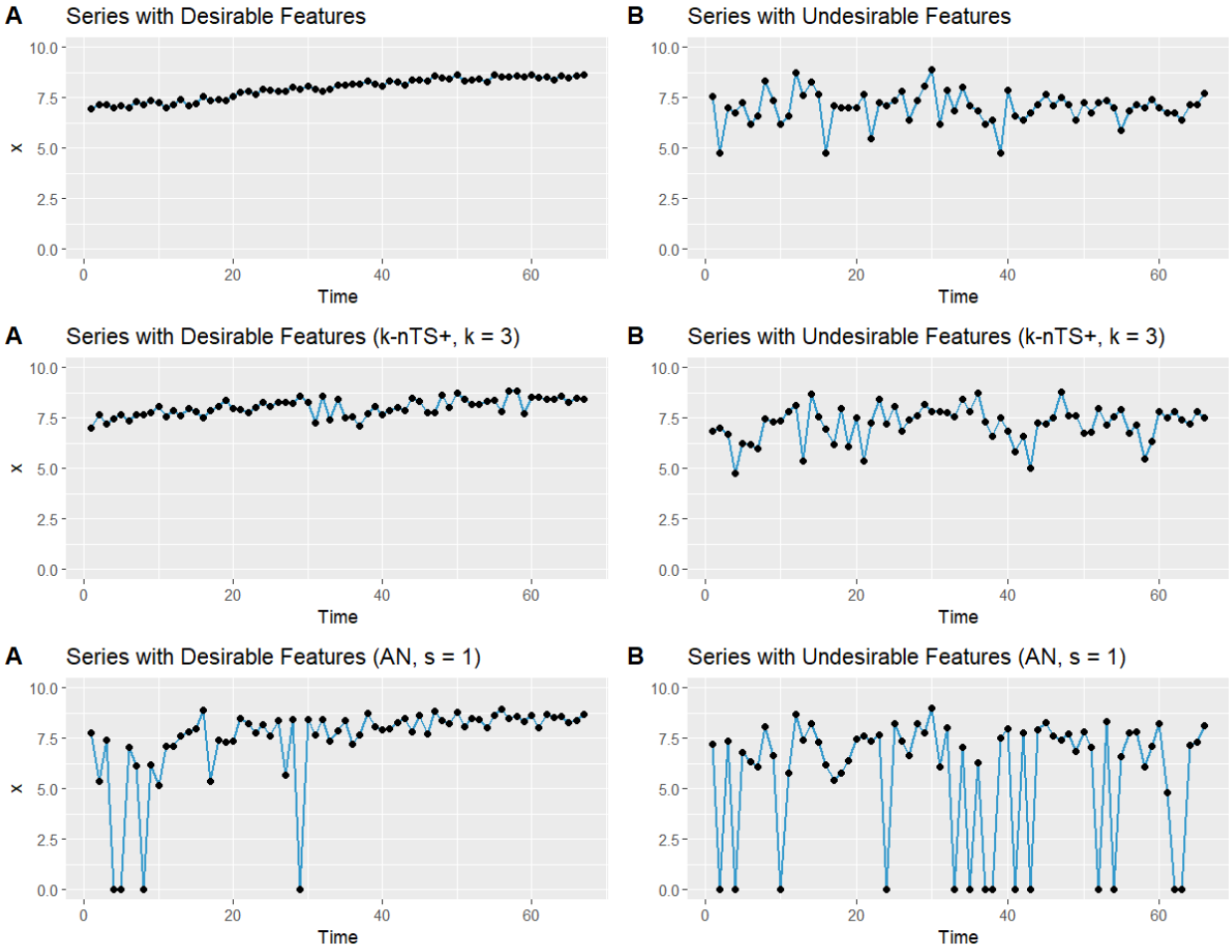


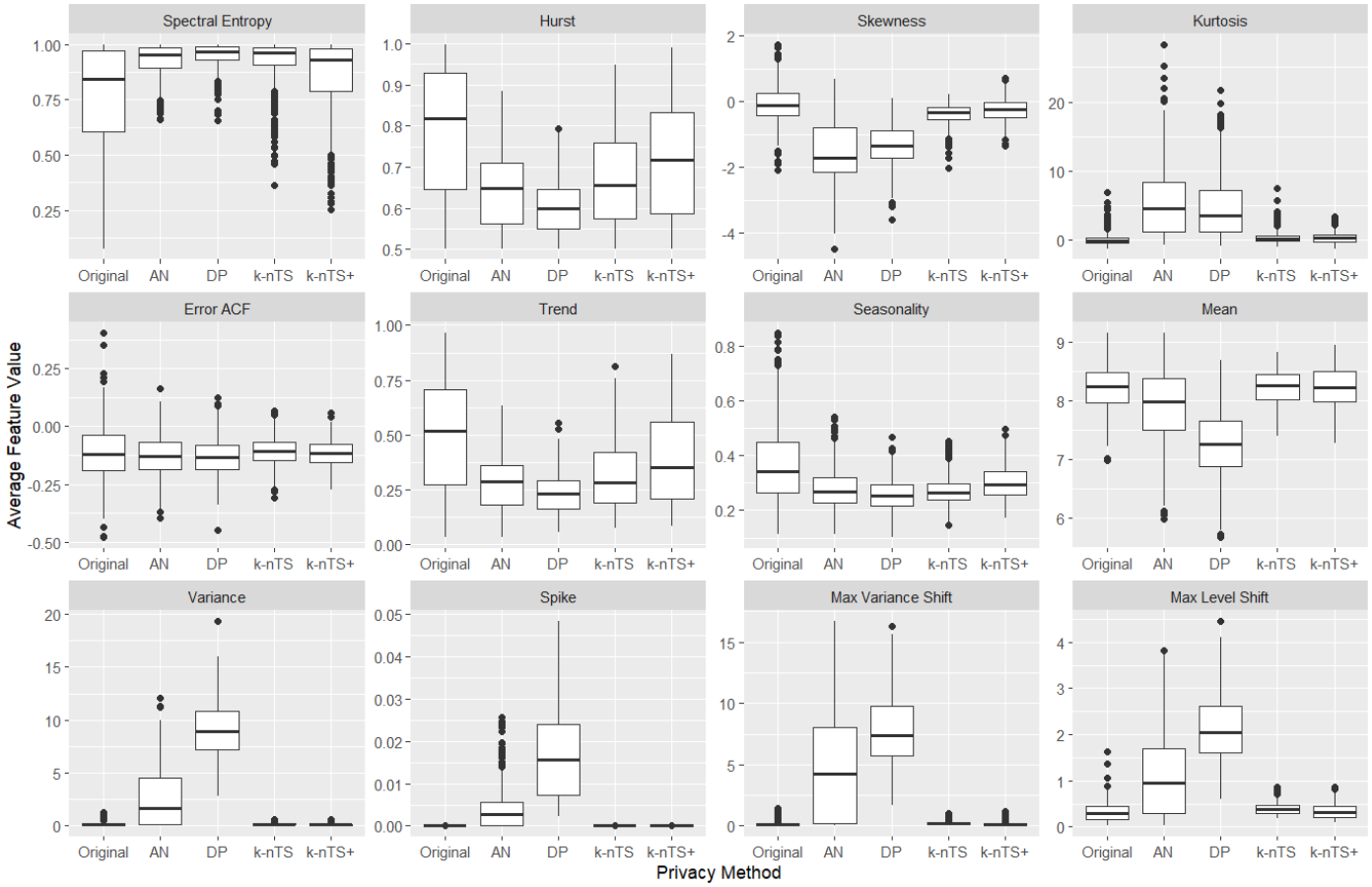
Table 5: Time series feature values from undesirable and desirable time series.

Feature	Desirable Time Series (left Fig. 7)			Undesirable Time Series (right Fig. 7)		
	Original	k-nTS+ (k=3)	AN (s=1)	Original	k-nTS+ (k=3)	AN (s=1)
Spectral Entropy	0.07	0.89	0.92	1.00	0.98	1.00
Hurst	0.99	0.81	0.76	0.50	0.66	0.50
Skewness	-0.41	-0.18	-2.74	-0.57	-0.71	-1.17
Kurtosis	-1.24	-0.74	6.99	1.16	0.25	-0.37
Error ACF	-0.09	-0.22	-0.20	-0.19	-0.06	-0.21
Trend	0.97	0.58	0.49	0.12	0.22	0.11
Seasonality	0.16	0.25	0.39	0.23	0.24	0.13
Mean	7.96	8.02	7.41	7.01	7.21	5.73
Variance	0.29	0.19	4.27	0.65	0.76	9.57

Spike	0.0000	0.0000	0.0037	0.0001	0.0001	0.0268
Max Variance Shift	0.05	0.24	9.37	1.10	1.12	11.44
Max Level Shift	0.57	0.51	2.77	0.70	0.84	3.29

Figure 7 displays boxplots of the time series feature values before and after protection across all time series in our application. Random noise privacy methods (AN and DP) increase the randomness and significantly change distributional characteristics of all features except *Error ACF*, leading to poor forecast accuracy. Random noise also produces a negative bias in the means of the protected series and significantly increases the variance. On the other hand, the *k*-nTS swapping method increases the spectral entropy but better preserves most feature distributions. The feature distributions of *k*-nTS+ swapping are much closer to the original distributions for those features important for forecast accuracy (*Spike, Max Variance Shift, Max Level Shift, Mean, Variance,* and *Trend*), which led to improved forecast accuracy results.

Fig 7: Distributions of time series features for each privacy method.



We note that while *k*-nTS performed swapping based on the values of *Spectral Entropy, Hurst*, and *Seasonality*, it does not preserve these feature distributions as well as *k*-nTS+. One reason could be that all three of these features are based on autocorrelation and we found that k-nTS swapping degrades *Seasonal ACF*, *X ACF*, *X ACF10*, and *X PACF5* more than k-nTS+ (mathematical details and a figure can be found in the Appendix). *k*-nTS+ did not explicitly swap based on the values of *Spectral Entropy, Hurst*, and *Seasonality* which demonstrates the importance of the k-nTS+ feedback loop. Although *Spectral Entropy* and *Hurst* were correlated with forecast accuracy across series, they were eliminated in the first stage of the *k*-nTS+ feature selection process using RReliefF.

4.5.4. Privacy Adjusted Forecasts

Similar to Fildes et. al. (2009) and Khosrowabadi et al. (2022), we compare the percentage of forecast adjustments that improved accuracy across adjustment direction, magnitude, and the coefficient of variation of the original time series. We use the adjusted forecasts using the k-nTS+ (k = 3) protected data set which was the top performing privacy method in our application.

To compute adjustment magnitude, we normalize the absolute difference between the adjusted and original forecasts using the mean of the original series,

$$Magnitude_j = \frac{\left|\hat{A}^o_{j,T+1} - \hat{A}^p_{j,T+1}\right|}{\bar{x}_j}$$

where the o and p superscripts denote a forecast based using the original and protected data, respectively. Using the approach of Khosrowabadi et al. (2022), we bin the magnitudes into high ($>$ 0.75 quantile), low ($<$ 0.25 quantile) and medium ($\geq$ 0.25 quantile and $\leq$ 0.75 quantile) intervals.

We also compute the average relative absolute error (AvgRelAE, see Fildes et al., (2013)) to compare the relative accuracy of the adjusted and original forecasts. The AvgRelAE of the adjusted forecasts is computed as

$$AvgRelAE \;=\; exp\left[\frac{1}{J}\sum_{j=1}^{J} log\;\frac{AE^p_j}{AE^o_j}\right],$$

where AE^p_j and AE^o_j are the absolute forecast error for the protected and original versions of series j, respectively. An $AvgRelAE$ less than one indicates an average improvement in accuracy and an $AvgRelAE$ greater than one indicates an average reduction in accuracy.[8] We remove the forecasts with the 5% smallest and 5% largest ratios (AE^p_j / AE^o_j) to prevent extreme outliers from affecting AvgRelAE (Fildes et al., 2013).

Using the k-nTS+ (k=3) protected data, we find that less than half (43%) of the adjusted forecasts improved forecast accuracy (lower absolute error), which is less than the reported 49.9% of judgmentally adjusted forecasts that improved accuracy in Khosrowabadi et al. (2022). Table 6 breaks down the results by adjustment magnitude and direction and displays the AvgRelAE and percentage of adjusted forecasts that improved accuracy. The results show that most privacy adjusted forecasts degraded accuracy and the AvgRelAE is greater than one in five out of six cases.

---

8 AvgRelAE can be generalized to accommodate multiple forecasts for each series. See Fildes et al. (2013) for the AvgRelMAE.

Also, our results are contrary to the findings in the judgmental literature which shows that large adjustments and negative adjustments improve forecast accuracy. We find that small privacy adjustments improved (47.9% of cases) forecast accuracy more frequently than large privacy adjustments (35.6% of cases). Furthermore, positive adjustments improved (44.6% of cases) forecast accuracy more than negative adjustments (40.2% of cases). However, none of these cases improved forecast accuracy overall which is expected due to privacy protection.

Table 6: AvgRelAE (and percentage of adjustments that improved accuracy) by magnitude and direction.

		Direction		
		Positive	Negative	**Total**
Magnitude	Large	1.35 (40.5%)	1.47 (30.4%)	**1.41 (35.6%)**
	Medium	1.12 (44.4%)	1.17 (41.9%)	**1.14 (43.2%)**
	Small	0.99 (49.1%)	1.06 (46.8%)	**1.03 (47.9%)**
	Total	**1.14 (44.6%)**	**1.21 (40.2%)**	**1.17 (42.5%)**

One issue with our data is that 73% of the series have negative slopes, which could cause positive adjustments to have a dampening effect on forecasts, and negative adjustments to overestimate the impact of the trend (Hyndman & Athanasopoulos, 2021). Table 7 displays the AvgRelAE and the percentage of adjustments for time series with positive slopes vs. negative slopes. To measure the slope, we calculate the slope coefficient of a simple linear regression that regresses the time series values on a continuous time variable. Our results show that time series with negative slopes and negative adjustments (32% of all time series) tended to degrade forecast accuracy the most.

Table 7: AvgRelAE (and the percentage of adjustments that improved accuracy) by slope and direction.

		Direction		
		Positive	Negative	**Total**
Slope	Positive	1.13 (42.9%)	1.14 (41.6%)	**1.14 (42.2%)**
	Negative	1.14 (45.0%)	1.24 (39.6%)	**1.18 (42.6%)**
	Total	**1.14 (44.6%)**	**1.21 (40.2%)**	**1.17 (42.5%)**

Table 8 measures the percentage of adjustments that improved accuracy and AvgRelAE categorized by the coefficient of variation of the original series and whether k-nTS+ (k=3) swapping increased, decreased, or maintained (within five percent) the coefficient of variation. We measure the coefficient of variation using the original time series values since there was only one forecast horizon. We bin the coefficients of variation into high (> 0.75 quantile), low (< 0.25 quantile) and medium (≥ 0.25 quantile and ≤ 0.75 quantile). We find that none of the coefficient of variation categories improve forecast accuracy compared to the original data. However, forecast accuracy degraded the most when k-nTS+ (k=3) was applied to time series with small coefficients of variation.

Table 8: AvgRelAE (and percentage of adjustments that improved accuracy) by coefficient of variation of the original series and the change in coefficient of variation in the protected series.

		Change in Coefficient of Variation			
		Decreased	Maintained (+/- 5%)	Increased	Total
Original Coefficient of Variation	Large	1.06 (49.1%)	1.14 (40.5%)	1.30 (44.7%)	**1.09 (46.2%)**
	Medium	1.18 (42.5%)	1.19 (40.3%)	1.12 (49.1%)	**1.16 (44.1%)**
	Small	1.15 (38.2%)	1.36 (31.6%)	1.27 (36.1%)	**1.27 (35.7%)**
	Total	**1.13 (45.0%)**	**1.19 (39.4%)**	**1.21 (41.7%)**	**1.17 (42.5%)**
		Proportion of Time Series			
		Decreased	Maintained (+/- 5%)	Increased	Total
Original Coefficient of Variation	Large	15.9%	7.8%	1.3%	**25.0%**
	Medium	22.7%	11.7%	15.6%	**50.0%**
	Small	1.1%	2.5%	21.4%	**25.0%**
	Total	**39.7%**	**22.0%**	**38.3%**	**100.0%**

Overall, our empirical results show that privacy adjustments affect forecast accuracy differently than judgmental adjustments. Specifically, we found that privacy adjustments had better forecast accuracy when the adjustments were small or positive, or when the coefficient of variation of the original series was large. However, on average, forecast accuracy worsened for nearly every combination of magnitude, direction, and coefficient of variation. This is not surprising since a major motivation of judgmental adjustments is to improve forecast accuracy (Fildes et al., 2019) and judgmental adjustments have been shown to improve forecast accuracy by 5-10% on average (Davydenko & Fildes, 2013; Khosrowabadi et al., 2022). For our application, privacy adjustments blur the data for privacy reasons and are expected to reduce forecast accuracy. The secondary goal of our proposed privacy method is to maintain forecast accuracy, which the top performing method (k-nTS+ (k=3) swapping) did with only a +13.9% average degradation. Furthermore, the average coefficient of variation of k-nTS+ (k=3) protected data is approximately 2% less than the average coefficient of variation in the original data. However, the average coefficients of variation under DP ($\epsilon = 10$) and AN ($s = 1$) were 18% and 35% larger than the average from the original data, respectively.

5. Conclusions

This paper examined the impact of data privacy on forecast accuracy in a centralized scenario where a data owner shares a protected data set with forecasters. Our proposed *k*-nTS+ swapping method used time series features to swap the values between time series to maintain forecast accuracy. We demonstrated the effectiveness of our privacy method using data from a well-known forecasting competition where the identities of the time series needed to be kept confidential. The proposed method limited the average reduction in forecast accuracy to +13.9% of the original forecast accuracy. Nearly all other privacy methods we studied degraded forecast accuracy to unusable levels (over 100%) at similar levels of privacy.

To the best of our knowledge, this paper is the first to create a protected time series data set tailored to maintain forecast accuracy. We did this by carefully investigating the similarity between

time series features rather than time series values. The protected data also preserved important features for forecasting such as spike, max variance shift, max level shift, mean, variance, strength of trend, and strength of seasonality. Furthermore, we showed that our k-nTS+ protected data was more representative of the original time series, potentially leading to increased trust and adoptability among organizations.

A substantial portion of the privacy literature is focused on theoretical privacy guarantees such as differential privacy. Our findings agree with past research (Goncalves et al. 2021a) and show that differential privacy (and additive noise) generates unusable forecasts at reasonable levels of privacy. This undesirable privacy-utility tradeoff has also been demonstrated in contexts other than forecasting. For example, a recent paper by Blanco-Justicia et al. (2022) found that much of the work on differential privacy and deep learning utilized relaxed versions of differential privacy with large values of ϵ that theoretically do not provide meaningful levels of privacy protection. Their experiments found that model regularization (e.g., L2-regularization) provided comparable privacy protection with better accuracy and lower model learning cost than differential privacy. In our application, we found that our k-nTS+ swapping method had better forecast accuracy at comparable levels of identification disclosure risk with differential privacy.

Although we showed that *k*-nTS+ swapping balanced the tradeoff between forecast accuracy and privacy well, future work could also examine the utility of use cases beyond forecasting. Since many of the time series features were preserved and the entire protected data set was shared, forecasters could use this time series data for other applications. One limitation of our study was that we did not consider privacy metrics other than identification disclosure risk, such as attribute disclosure risk. Further research could pursue this area or address whether combinations of forecasts using multiple protected data sets improve the privacy-utility tradeoff.

References

Bandara, K., Bergmeir, C., & Smyl, S. (2018). Forecasting Across Time Series Databases using Recurrent Neural Networks on Groups of Similar Series: A Clustering Approach. ArXiv:1710.03222 [Cs, Econ, Stat]. http://arxiv.org/abs/1710.03222

Blanco-Justicia, A., Sanchez, D., Domingo-Ferrer, J., & Muralidhar, K. (2022). A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning. arXiv preprint arXiv:2206.04621.

Boone, T., Ganeshan, R., Jain, A., & Sanders, N. R. (2019). Forecasting sales in the supply chain: Consumer analytics in the big data era. *International Journal of Forecasting*, *35*(1), 170–180. https://doi.org/10.1016/j.ijforecast.2018.09.003

Chen, C., & Liu, L.-M. (1993). Forecasting time series with outliers. *Journal of Forecasting*, *12*(1), 13–35. https://doi.org/10.1002/for.3980120103

Davydenko, A., & Fildes, R. (2013). Measuring forecasting accuracy: The case of judgmental adjustments to SKU-level demand forecasts. *International Journal of Forecasting*, *29*(3), 510–522. https://doi.org/10.1016/j.ijforecast.2012.09.002

Duncan, G. T., & Stokes, S. L. (2004). Disclosure risk vs. data utility: The RU confidentiality map as applied to topcoding. *Chance*, 17(3), 16-20.

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2009). Effective forecasting and judgmental adjustments: An empirical evaluation and strategies for improvement in supply-chain planning. *International Journal of Forecasting*, *25*(1), 3–23. https://doi.org/10.1016/j.ijforecast.2008.11.010

Fildes, R., Goodwin, P., & Önkal, D. (2019). Use and misuse of information in supply chain forecasting of promotion effects. *International Journal of Forecasting*, *35*(1), 144–156. https://doi.org/10.1016/j.ijforecast.2017.12.006

Fulcher, B. D., & Jones, N. S. (2014). Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, *26*(12), 3026–3037. https://doi.org/10.1109/TKDE.2014.2316504

Goerg, G. (2013). Forecastable component analysis. In *International conference on machine learning* (pp. 64-72). PMLR.

Gonçalves, C., Bessa, R. J., & Pinson, P. (2021a). A critical overview of privacy-preserving approaches for collaborative forecasting. *International Journal of Forecasting*, *37*(1), 322–342. https://doi.org/10.1016/j.ijforecast.2020.06.003

Gonçalves, C., Bessa, R. J., & Pinson, P. (2021b). Privacy-Preserving Distributed Learning for Renewable Energy Forecasting. *IEEE Transactions on Sustainable Energy*, *12*(3), 1777–1787. https://doi.org/10.1109/TSTE.2021.3065117

Gonçalves, C., Pinson, P., & Bessa, R. J. (2021c). Towards Data Markets in Renewable Energy Forecasting. IEEE Transactions on Sustainable Energy, 12(1), 533–542. https://doi.org/10.1109/TSTE.2020.3009615

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, 27(3), 659-678.

Herzen, J. et al. (2022). Darts: User-friendly modern machine learning for time series. Journal of Machine Learning Research, vol. 23, num. 124, pg. 1-6. URL:http://jmlr.org/papers/v23/21-1177.html.

Hewamalage, H., Bergmeir, C., & Bandara, K. (2022). Global models for time series forecasting: A Simulation study. *Pattern Recognition*, *124*, 108441. https://doi.org/10.1016/j.patcog.2021.108441

Hyndman, R.J., & Athanasopoulos, G. (2021) *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. OTexts.com/fpp3. Accessed on Feb. 23rd, 2023.

Imtiaz, S., Horchidan, S.-F., Abbas, Z., Arsalan, M., Chaudhry, H. N., & Vlassov, V. (2020). Privacy Preserving Time-Series Forecasting of User Health Data Streams. *2020 IEEE International Conference on Big Data (Big Data)*, 3428–3437. https://doi.org/10.1109/BigData50022.2020.9378186

Kang, Y., Hyndman, R. J., & Smith-Miles, K. (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, *33*(2), 345–358. https://doi.org/10.1016/j.ijforecast.2016.09.004

Khosrowabadi, N., Hoberg, K., & Imdahl, C. (2022). Evaluating human behaviour in response to AI recommendations for judgemental forecasting. *European Journal of Operational Research*, *303*(3), 1151–1167. https://doi.org/10.1016/j.ejor.2022.03.017

Koning, A. J., Franses, P. H., Hibon, M., & Stekler, H. O. (2005). The M3 competition: Statistical tests of the results. International Journal of Forecasting, 21(3), 397–409. https://doi.org/10.1016/j.ijforecast.2004.10.003

Li, L., Kang, Y., & Li, F. (2022). Bayesian forecast combination using time-varying features. International Journal of Forecasting, S0169207022000930. https://doi.org/10.1016/j.ijforecast.2022.06.002

Liyue Fan & Li Xiong. (2014). An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. *IEEE Transactions on Knowledge and Data Engineering*, *26*(9), 2094–2106. https://doi.org/10.1109/TKDE.2013.96

Loning, M., et al. (2022). sktime/sktime: v0.13.4. doi: 10.5281/zenodo.7117735.

Luo, J., Hong, T., & Fang, S.-C. (2018). Benchmarking robustness of load forecasting models under data integrity attacks. *International Journal of Forecasting*, *34*(1), 89–104. https://doi.org/10.1016/j.ijforecast.2017.08.004

Makridakis, S., & Hibon, M. (2000). The M3-Competition: Results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476. https://doi.org/10.1016/S0169-2070(00)00057-1

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

Nin, J., & Torra, V. (2009). Towards the evaluation of time series protection methods. *Information Sciences*, *179*(11), 1663–1677. https://doi.org/10.1016/j.ins.2009.01.024

Nogueira, Fernando (2014). Bayesian Optimization: Open source constrained global optimizatio tool for Python. URL: https://github.com/fmfn/BayesianOptimization.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Taieb, S. B., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M.

P., Cordeiro, C., Oliveira, F. L. C., Baets, S. D., Dokumentov, A., … Ziel, F. (2022). *Forecasting: Theory and practice*. 167.

Petropoulos, F., & Siemsen, E. (2022). Forecast Selection and Representativeness. *Management Science*, mnsc.2022.4485. https://doi.org/10.1287/mnsc.2022.4485

Qi, L., Li, X., Wang, Q., & Jia, S. (2022). fETSmcs: Feature-based ETS model component selection. *International Journal of Forecasting*, S0169207022000954. https://doi.org/10.1016/j.ijforecast.2022.06.004

Rose, O. (1996). *Estimation of the Hurst parameter of long-range dependent time series* (Vol. 137). Report.

Schneider, M. J., Jagpal, S., Gupta, S., Li, S., & Yu, Y. (2018). A flexible method for protecting marketing data: An application to point-of-sale data. Marketing Science, 37(1), 153-171.

Seabold, S. & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Sobolev, D. (2017). The effect of price volatility on judgmental forecasts: The correlated response model. *International Journal of Forecasting*, *33*(3), 605–617. https://doi.org/10.1016/j.ijforecast.2017.01.009

Sommer, B., Pinson, P., Messner, J. W., & Obst, D. (2021). Online distributed learning in wind power forecasting. *International Journal of Forecasting*, *37*(1), 205–223. https://doi.org/10.1016/j.ijforecast.2020.04.004

Spiliotis, E., Kouloumos, A., Assimakopoulos, V., & Makridakis, S. (2020). Are forecasting competitions data representative of the reality? *International Journal of Forecasting*, *36*(1), 37–53. https://doi.org/10.1016/j.ijforecast.2018.12.007

Talagala, T. S., Li, F., & Kang, Y. (2022). FFORMPP: Feature-based forecast model performance prediction. *International Journal of Forecasting*, *38*(3), 920–943. https://doi.org/10.1016/j.ijforecast.2021.07.002

Wang, X., Smith, K., & Hyndman, R. (2006). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*, *13*(3), 335–364. https://doi.org/10.1007/s10618-005-0039-x

Willinger, W., Paxson, V., & Taqqu, M. S. (1998). Self-similarity and heavy tails: Structural modeling of network Traffic. R. Adler, R. Feldman, and MS Taqqu, editors, *A Practical Guide to Heavy Tails: Statistical Techniques for Analyzing Heavy Tailed Distributions*, Birkhauser Verlag, Boston.

6. Appendix

6.1. Implementation Details

Preprocessing

Time series were pre-processed by taking the log of the original series. For the VAR model, we took the first difference of the log series. For the global models (RNN and LGBM), we divided each log series by its respective mean.

Model Implementations

All local models (SES, DES, TES, and AutoARIMA) were implemented using the *sktime* forecasting module (Loning et al., 2022). We used additive trend and seasonality components for DES and TES. We set AutoARIMA to apply a seasonal model (SARIMA) with a maximum of twenty-five iterations.

The VAR model from the *statsmodels* (Seabold & Perktold, 2010) module was applied to subsets of approximately five time series at a time, ensuring that each subset consisted only of time series with the same length.

The LGBM and RNN models were implemented using the *darts* module (Herzen et al., 2022). For both models, we reserved a validation time period immediately prior to the desired forecast horizon. We used Bayesian optimization (Nogueira, 2014) to optimize the hyperparameters of each model to minimize the absolute forecast error (L1 loss) in the validation time period. We retrained each model using the optimized hyperparameters and the full training data (including the validation period) prior to generating forecasts for the desired forecast horizon.

The Bayesian optimizer was initialized for the LGBM models using ten starting points and run for 50 iterations. For the RNN the optimizer was initialized using five starting points and run for 25 iterations.

The hyperparameters for the LGBM and RNN models and their corresponding ranges are shown in Tables 9 and 10. We limited the RNN to the last ten input-output window samples from each time

series for computational efficiency and trained ten RNN models taking the median of the forecasts as the final forecast (Hewamalage et al., 2022).

Table 9: LGBM hyperparameter ranges.

Hyperparameter	**Range**
Input Window Length	25
Learning Rate	[0.01, 0.1]
Number of Boost Rounds	[50, 1000]
Number of Leaves	[10, 100]
Bagging Frequency	[1, 5]
Bagging Fraction	[0.01, 1]
L2 Regularization Parameter	[0, 0.5]
Minimum Observations in Leaf	[10, 100]

Table 10: RNN hyperparameter ranges.

Hyperparameter	**Range**
Input Window Length	25
Training Length	30
Learning Rate	[0.001, 0.1]
Weight Decay	[0.0001, 0.0008]
Number of Layers	[1, 2]
Hidden Dimension	[20, 50]
Batch Size	[200, 700]
Number of Epochs	[3, 30]
Dropout Rate	[0.1, 0.5]

6.2 Mathematical Details of Identification and Attribute Disclosure

To perform identification disclosure, we assume a third party possesses some original data on a unit of interest in the protected dataset. Denote this original data $\boldsymbol{c_i} = (ID_i, c_i)$, which contains a direct identifier ID_i (e.g., the identity of retailer i) and original data $c_i = \left(A_{i,t'}, \dots, A_{i,t'+E}\right)$ which contains a sequence of values that are components of the original time series x_j.

We let M_i denote the random variable (from the perspective of the third party) that indicates the corresponding PID_j for ID_i, i.e., $M_i = j$ when the values in $\boldsymbol{c}_i$ are components of the original version of the protected series j. Since the true value $M_i = j^*$ is unknown, the third party predicts the value of M_i to be the series j with the highest match probability, conditional on the known values, as follows,

$$\widehat{M}_i = argmax_j P(M_i = j|c_i), \qquad (1)$$

where identification disclosure occurs when $\widehat{M_i} = j^*$. The probability $P(M_i = j|c_i)$ is calculated as follows. Let $\tilde{x}_j = \left(P_{j,t'}, \dots, P_{j,t'+E}\right), j = 1, \dots, J$ denote the protected values of each time series j that occur in the same time periods as c_i. The third party computes the similarity between c_i and the protected values $\tilde{x_j}, j = 1, \dots, J$ using the Euclidean distance,

$$s(c_i, \tilde{x}_j) = \frac{1}{\|c_i - \tilde{x}_j\|_2}, j = 1, \dots, J.$$

Using these similarities, the third party builds a probability mass function for M_i over all protected series in X' as

$$P(M_i = j \mid c_i) = \frac{s(c_i, \tilde{x}_j)}{\sum_{j=1}^{J} s(c_i, \tilde{x}_j)},$$

and predicts $\widehat{M_i}$ as in (1).

To estimate the risk of identification disclosure, we perform simulations in which we sample E sequential values from each original time series x_j, and we measure the average proportion of series which are identified. The sampled values are denoted $C = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_J]^T$. Each of the vectors $\boldsymbol{c}_i$ corresponds to one of the J original time series, and we compute r_j conditional on the sampled c_i from series j. We repeat this simulation S times to obtain $\boldsymbol{C} = \{C_1, \dots, C_S\}$, and compute the average proportion of correctly identified time series across all external data samples and original time series,

$$\bar{P} = \frac{1}{J \times S} \sum_{s=1}^{S} \sum_{i=1}^{J} I\left(\widehat{M}_i^s = j^*\right)$$

These simulations assume that the third party in possession of C predicts the match for each vector $\boldsymbol{c}_i$ independently of the predicted matches for other vectors. The risk estimate from a given simulation is equivalent to the identification risk when J independent third parties are each in possession of one of the vectors $\boldsymbol{c}_i$ and each attempts identification risk as described above. Overall, multiple vectors may be matched to the same protected time series.

6.3 Equations for Important Time Series Features from the Literature

Spectral Entropy

Suppose x_j is a univariate stationary time series with a finite mean and constant variance. The spectral density $f_x(\lambda)$ of x_j is estimated as the scaled Fourier transform of the autocovariance function $\gamma_x(k)$ of x_j. The spectral density can be thought of as the probability density function of a random variable Λ on the unit circle (Goerg, 2013), where for a non-zero integer k, when $\gamma_x(k) \neq 0$, the spectral density $f_x(\lambda)$ will have a peak at the corresponding frequency λ. The forecastability, or spectral entropy, of x_j is measured using the Shannon entropy of $f_x(\lambda)$, given by

$$Spectral\ Entropy = -\int_{-\pi}^{\pi} \widehat{f}_x(\lambda) log\, \widehat{f}_x(\lambda) d\lambda,$$

where the maximum entropy occurs when $\Lambda \sim U(-\pi, \pi)$. In practice, estimates of F_1 range from 0 to 1, where high values represent a low signal-to-noise ratio, indicating that x_j is difficult to forecast (Kang et al., 2017).

Hurst

Next, we consider a self-similarity feature quantified using the Hurst parameter (Wang et al., 2006), which measures the long-range dependence of a time series. Spiliotis et al. (2020) found this feature had the largest effect on forecast accuracy. We use the definition of self-similarity of a time series described by (Willinger et al., 1998). Suppose that x_j is the increment process of y_j, *i.e.*, $x_{j,t} = y_{j,t+1} - y_{j,t}$. An aggregate sequence, denoted $x_j^{(m)}$, is created by averaging x_j over non-overlapping blocks of size m, where

$$x_{j,k}^{(m)} = 1/m, \sum_{i=(k-1)m+1}^{km} x_{j,i} \quad k = 1,2, \ldots$$

and k indexes the block. If y_j is a self-similar time series, then

$$x_j = m^{1-H} x_j^{(m)}$$

for all integers m. We use the definition of second-order self-similarity, where x_j *i*s exactly second-order self-similar if $m^{1-H} x_j^{(m)}$ has the same variance and autocorrelation as x_j for all values of m, or is asymptotically second-order self-similar if this holds as $m \to \infty$ (Rose, 1996). The parameter H is the Hurst exponent, which is estimated using the differencing term d from a fractional ARIMA model, i.e., FARIMA(0, d, 0) (Wang et al., 2006; Hyndman et al., 2022), where

$$Hurst = H = d + 0.5.$$

Estimates of H range from 0 to 1, where $H = 0.5$ corresponds to a random walk (Sobolev, 2017), $H < 0.5$ corresponds to anti-persistent or mean-reverting series, and $H > 0.5$ corresponds to persistent time series that are more likely to maintain their current trend.

Skewness

Skewness measures the lack of symmetry in the distribution of the values of x_j (Wang et al., 2006), where positive (or negative) values are associated with a right- (or left-) skewed data distribution,

$$Skewness = \frac{1}{n\sigma^3} \sum_{t=1}^{n} \left(x_j - \bar{x_j}\right)^3.$$

Kurtosis

We measure *Kurtosis* relative to the standard normal distribution (Wang et al., 2006). Positive *Kurtosis* corresponds to distributions that tend to have a distinct peak near the mean with heavy tails, whereas negative *Kurtosis* corresponds to distributions that are relatively flat near the mean,

$$Kurtosis = \frac{1}{n\sigma^4}\sum_{t=1}^{n}\left(x_j - \bar{x_j}\right)^4 - 3,$$

where 3 is the *Kurtosis* of the standard normal distribution.

Error Autocorrelation Function (Error ACF)

Next, we perform STL decomposition (Cleveland et al. 1990) to obtain the trend, seasonal, and remainder components of x_j. We use the approach of Hyndman et al. (2019) to obtain

$$x_j = b_j + s_{1,j} + \cdots + s_{M,j} + e_t,$$

where b_j, $s_{i,j}$, and e_j are the trend, ith seasonal, and remainder components, respectively.

We extract the first-order autocorrelation coefficient of the detrended and deseasonalized series, referred to as 'linearity' by Spiliotis et al. (2018):

$$Error\ ACF = \frac{\sum_{t=2}^{T}(e_{j,t}-\bar{e})(e_{j,t-1}-\bar{e})}{\sum_{t=1}^{T}(e_{j,t}-\bar{e})^2}.$$

Error ACF is a measure of the predictability of a time series after the trend and seasonality have been accounted for (Kang et al. 2017).

Trend and Seasonality

We also compute the strength of trend (*Trend*) and strength of the ith seasonal component ($Seasonality_i$) as follows,

$$Trend = 1 - \frac{Var(e_j)}{Var(f_j+e_j)},$$

and

$$Seasonality_i = 1 - \frac{Var(e_j)}{Var(s_{i,j}+e_j)}.$$

In practice, the values of $Trend$ and $Seasonality_i$ range from 0 to 1 (Hyndman 2022).

Mean and Variance

The next two features are the *Mean* and *Variance*, also used by Bandara et al. (2018) to cluster similar time series for forecasting, which are written as follows,

$$Mean = \frac{1}{T}\sum_{t=1}^{T} x_{j,t},$$

$$Variance = \frac{1}{T-1}\sum_{t=1}^{T}\left(x_{j,t} - \overline{x_j}\right)^2.$$

We also included many other features from the *tsfeatures* package in R. We refer the reader to (Hyndman et al., 2022) for explanation of these features.

6.4 Detailed k-nTS+ Swapping Algorithm

Algorithm 2: The k-nTS+ Swapping Method [Detailed]

Require [Initialization]:

$X = [x_1, \dots x_J]$: the $J \times T$ matrix of original time series.
$P = \{\mathcal{P}_1, \dots, \mathcal{P}_B\}$: the B baseline privacy methods.
$\mathcal{F}$: the desired forecasting model.
$\mathcal{M} = \{m_1, \dots m_N\}$: the initial set of time series features' names.
k: the number of nearest neighbor time series to consider for swapping.
$\mathcal{K}$: the number of nearest neighbor time series to consider for RReliefF.
N_{rfe}: the number of recursive feature elimination iterations
τ^*: recursive feature elimination prediction error threshold

Step 1: Create Baseline Protected Datasets

i. Store the data values from all series from time T in X as a test set: $X^{test} = X[T]$
ii. Create protected data set $\mathcal{P}_b(X[1:T-1]) = X^b$ for each baseline privacy method $\mathcal{P}_b \in P$

Step 2: Generate Baseline Forecasts

i. Generate forecasts using $\mathcal{F}$ for time T based on the original data $X[1:T-1]$ and each protected data set $\mathcal{P}_b, b = 1, \dots, B$

Step 3: Measure Forecast Accuracy

i. Compute forecast errors at the series level $\epsilon = \left(\epsilon_1, \dots, \epsilon_J\right)^T$ for the original data forecasts and $\epsilon^b = \left(\epsilon_1^b, \dots, \epsilon_J^b\right)^T, b = 1, \dots, B$ for the protected data forecasts for time T

Step 4: Extract and Select Time Series Features

i. Extract cross-sectional time series features matrices C from $X[1:T-1]$ and $C^b, b = 1, \dots, B$, from $X^b, b = 1, \dots, B$

ii. Create cross-sectional feature matrix $\mathcal{C} = [C^1, \dots, C^B, C]$ and forecast error vector $\mathcal{E} = (\epsilon^b, \dots, \epsilon^B, \epsilon)$ by concatenating the feature matices and error vectors from the baseline protected and original data sets

iii. Treat forecast errors $\mathcal{E}$ as the target and time series features $\mathcal{C}$ as the predictors. Generate weight W_n for each feature m_n using RReliefF algorithm (Robnik-Sikonja & Kononenko, 2003) with nearest neighbor parameter $\mathcal{K}$.

iv. Select features $\mathcal{M}' = \{m_n \in \mathcal{M} \ : \ W_n > 0\}$ which contains the names of the S features with RReliefF weights greater than zero.

v. Create cross-sectional feature matrix $\mathcal{C}'$ from $\mathcal{C}$ such that $\mathcal{C}'$ contains the features $\mathcal{M}'$

vi. **for** $i = 1, \dots N_{rfe}$:
    a. Train a random forest to predict $\mathcal{E}$ using $\mathcal{C}'$.
    b. Calculate $e_{i,S}$, the mean-squared error of the random forest out-of-bag predictions for iteration i and number of features S.
    c. Calculate importance I_n of each feature $m_n \in \mathcal{M}'$as the change in mean-squared error of the out-of-bag predictions after permuting the feature m_n in $\mathcal{C}'$.
    d. **for** subset size $s = S - 1, \dots, 0$:
        i. Drop feature m_n with the lowest importance I_n from $\mathcal{C}'$ such that s features remain.
        ii. Assign rank $r_{i,s} = s + 1$ to m_n for iteration i.
        iii. **if** $s > 0$: Repeat steps (b.) and (c.)
    end for
end for

vii. Compute $\bar{e}_s = \sum_{i=1}^{N_{rfe}} e_{i,s}$ the average mean-squared error of the out-of-bag predictions using s features for $s = 1, \dots, S$.

viii. Compute $\bar{r}_n = \sum_{i=1}^{N_{rfe}} r_{i,s}$, the average rank of each feature m_n

ix. Identify the number of features s_{min} with the minimum $\bar{e}^{min}$

x. Calculate $\tau_s = (\bar{e}_s - \bar{e}^{min})/\bar{e}^{min}$ for $s = 1, \dots, S$, the percentage increase in the average out-of-bag mean squared error from using s features in the random forest model

xi. Set s^* to the smallest value of s with $\tau_s \leq \tau^*$

xii. Select the s^* features with the best (lowest) average ranks $\bar{r}_n$

Step 5: Create Protected Data Set using k-nTS+ Swapping

i. Use input k and the features selected in *Step 4* to perform swapping through time T using **Algorithm 1: The k-nTS Swapping Method.**

6.5 Autocorrelation Feature Results

In Section 4.5.3, we noted that while *k*-nTS performed swapping based on the values of *Spectral Entropy, Hurst*, and *Seasonality*, it does not preserve these feature distributions as well as *k*-nTS+. To help explain this difference, in Figure 8 we plot the distributions of four autocorrelation-based features, *Seasonal ACF* (first coefficient of the seasonal autocorrelation function), *X ACF* (first coefficient of the autocorrelation function), *X ACF10* (sum of the first ten coefficients of the autocorrelation function), and *X PACF5* (sum of the first five coefficients of the partial autocorrelation function). While neither k-nTS or k-nTS+ used these autocorrelation features for

swapping, k-nTS+ preserves the distributions of these features much better than the other privacy methods which again demonstrates the importance of the k-nTS+ feedback loop.

Figure 8: distributions of autocorrelation-based features in the original and protected data sets.