Certain time series characteristics, such as the strength of trend and spectral entropy, are related to forecast accuracy. These characteristics are useful for predicting the performance of forecasting models and predicting optimal forecast combinations (Li et al., 2022; Talagala et al., 2022). Due to their relationship with forecast accuracy, we analyze changes in these characteristics resulting from data protection to give insights into why forecasts change.

Our contributions are two-fold. First, we analyze privacy adjusted forecasts for multiple forecasting models and privacy methods, giving detailed explanations as to why model performance changes on protected data. To explain the improvement and/or degradation of forecast accuracy from data protection, we analyze privacy adjusted forecasts from two perspectives: (1) How data protection changes time series characteristics which translate into changes in forecast accuracy, and (2) How changes to forecasts are related to changes in forecast accuracy. To address (1), we use the approach outlined by Kang et al. (2017) to extract measures of forecastability such as spectral entropy and first order autocorrelation parameters which show fundamental changes to the data from data protection, and help explain why certain models perform better than others on protected data. To address (2), we measure whether forecasts are positively or negatively adjusted under data protection as well as the magnitude of each adjustment. Similar to Fildes et al. (2009) and Khosrowabadi et al. (2022), we classify privacy adjusted forecasts as either improving or degrading forecast accuracy. We use the random forest approach of Khosrowabadi et al. (2022) to identify the time series characteristics and adjustment features that are predictive of whether privacy adjusted forecasts improve or degrade accuracy.

For our second contribution, we propose a novel privacy method designed with forecasters in mind. Motivated by the random forest results, we implement a matrix-based privacy method which swaps the values of time series with similar characteristics to help maintain forecast accuracy. Results show that our method... We describe our proposed method next.

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

Let $\mathbb{X} = \{x_1, \dots, x_J\}$ be a given set of time series data (n-vectors). Using the standard privacy methods shown later in Table 1, a data provider releases protected data $P_{j,t}$ for each time series x_j based on the confidential values $A_{j,t}$ up until time t. The main issue with these methods is that they choose protected values based on predefined rules and not changes in forecast accuracy. However, the goal of the data provider should be to change $A_{j,t}$ to $P_{j,t}$ with minimal reductions in forecast accuracy while increasing privacy to an acceptable threshold.

We solve the data protection problem for the data provider using a matrix based k-nTS (k-nearest time series) swapping method, where the data provider releases a set of protected time series $\mathbb{X}' = \{x'_1, \dots, x'_J\}$, where $x'_j = (P_{j,1}, \dots, P_{j,t})^T$ is based on $\mathbb{X}$, the confidential values of *all* series up through time t. The k-nTS method matches time series that have similar characteristics within a rolling window of their past values. Then, it uses randomization to replace each $A_{j,t}$ with a confidential value from another time series with similar characteristics to balance the trade-off between forecast accuracy and privacy.

Depending on the quantity of available data, k-nTS can use rolling windows of data which adjust for dynamic changes in the relationships between time series. For example, if we choose a rolling window size, say n, then $x_j = (A_{j,t-n+1}, A_{j,t-n+2}, \dots, A_{j,t-1}, A_{j,t})^T$ where $x_j \in \mathbb{R}^n$. Protection in subsequent time periods from $t+1$ to T rolls x_j forward from $x_j = (A_{j,t-n+2}, A_{j,t-n+3}, \dots, A_{j,t}, A_{j,t+1})^T$ to $(A_{j,T-n+1}, A_{j,T-n+2}, \dots, A_{j,T-1}, A_{j,T})^T$, respectively. We label the time series characteristics 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 $x_j = (A_{j,t-n+1}, A_{j,t-n+2}, \dots, A_{j,t-1}, A_{j,t})^T$. For simplicity, we omit the

t subscript for the feature vectors, and write f_j. See Li et al. (2022) for another example of the computation of time series features based on a rolling, fixed-length window.

For each time series $x_j \in \mathbb{R}^n$, the data provider computes the feature vector $f_j \in \mathbb{R}^m$. This vector can contain any single-valued feature calculated based on the values in x_j, such as the strength of the trend and seasonality, the spectral entropy, or the mean of the values in the current window. This produces a set $\mathbb{C} = \{f_1, \ldots, f_J\}$ of m-vectors containing the characteristics of each of the J time series windows. For each of the feature vectors $f_j \in \mathbb{R}^m$, the data provider computes a set of squared distances of the elements of the set $\mathbb{C}$. We define $\mathbf{dist}(f_j, f_i) = d_{j,i}$ as the distance between f_j and f_i, i.e., the feature vectors corresponding to two distinct time series from a given set $\mathbb{X}$. Without loss of generality, we use the Euclidean norm, or ℓ^2-norm as a distance metric[8]. Since our case is multivariate and partially ordered, we can get a totally ordered set based on the Euclidean distance.

Let us define $x_j^{(k)}$ as the kth nearest neighbor of x_j, with corresponding feature vector $f_j^{(k)}$. Then, for a time series x_j, we have $\{d_{j,(1)},\ d_{j,(2)}, \ldots, d_{j,(J-1)}\}$ such that $d_{j,(k)} \leq d_{j,(l)}$ for any integers $k < l$ where $d_{j,(k)} = \|f_j - f_j^{(k)}\|$. Note that $x_j^{(i)} \in \mathbb{X}\backslash\{x_j\}$ and the superscript (i) means the i^{th} 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-nTS (k-nearest time series) can be represented as the set $K_j = \{x_j^{(1)}, \ldots, x_j^{(k)}\}$ based on $\|f_j - f_j^{(1)}\| \leq \cdots \leq \|f_j - f_j^{(k)}\|$ or an ordered set $\{d_{j,(1)},\ d_{j,(2)}, \ldots, d_{j,(k)}\}$.

For more efficient computation of such ordering, 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 $d_{i,j} = \|f_i - f_j\|^2$, and $d_{i,j}$ is the (i, j)th entry of D (also note that $\operatorname{rank}(D) \leq m + 2$). Suppose that we are given a data matrix $X = [x_1, x_2, \ldots, x_J],\ x_j \in \mathbb{R}^n$ (i.e., $X \in \mathbb{R}^{n\times J}$). We can write X, i.e., a confidential data matrix, as the following.:

$$X = [x_1, x_2, \ldots, x_J] = \begin{pmatrix} A_{1,t-n+1} & A_{1,t-n+2} & A_{1,t-n+3} & \cdots & A_{1,t-1} & A_{1,t} \\ A_{2,t-n+1} & A_{2,t-n+2} & A_{2,t-n+3} & \cdots & A_{2,t-1} & A_{2,t} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ A_{J,t-n+1} & A_{J,t-n+2} & A_{J,t-n+3} & \cdots & A_{J,t-1} & A_{J,t} \end{pmatrix}^T, \tag{2}$$

where $x_j = (A_{j,t-n+1}, A_{j,t-n+2},\ \ldots, A_{j,t-1}, A_{j,t})^T$ where $x_j \in \mathbb{R}^n$ and $x_j \in \mathbb{X}$. 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,

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

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 up as the following:

$$D = \mathbf{1} diag(C^T C)^T - 2C^T C + diag(C^T C)\mathbf{1}^T, \tag{4}$$

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

```
Algorithm 1 The $k$-nTS Swapping method
Require: [Initialization]
  (i) [Time Series Matrix $X \in \mathbb{R}^{n \times J}$] $X = [x_1, x_2, \ldots, x_J]$, $x_j \in \mathbb{R}^n$ for $j = 1, \ldots, J$ as in (2).
  (ii) [Feature Matrix $C \in \mathbb{R}^{m \times J}$] $C = [f_1, f_2, \ldots, f_J]$, $f_j \in \mathbb{R}^m$ for $j = 1, \ldots, J$ as in (3).
  (iii) [Distance Matrix $D$] $D = \mathbf{1} diag(C^T C)^T - 2C^T C + diag(C^T C)\mathbf{1}^T$ as in (4).
  for $j = 1, 2, \ldots, J$ do
    [Finding a set $K_j$ for $x_j$] Let $d_j$ denote the $j$th column of $D$. Sort $d_j$ from the smallest to largest components and find the $k$th smallest component, followed by $K_j$ as in (5).
    [Random swapping] $x_j \leftarrow x_j^{(i)}$ (last components only) for some $i \in \{1, \ldots, k\}$ as in (6).
  end for
```

where the symbol **1** denotes a column vector of J ones[9]. It is easy to see that the column vector $diag(C^T C) = (\|f_1\|^2, \ldots, \|f_J\|^2)^T$. 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, the jth column of D from the smallest to largest components and find the kth smallest component so that we have

$$K_j = \{x_j^{(1)}, \ldots, x_j^{(k)}\}. \tag{5}$$

That is, the data provider then 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. In case of $k = J - 1$, random swapping is simply done by rearranging the components in the last row of matrix X. Let the ith most similar time series to x_j be $x_j^{(i)} = (A_{j,t-n+1}^{(i)}, A_{j,t-n+2}^{(i)}, \ldots, A_{j,t}^{(i)})^T$ where n is the length of the rolling window of past data. Then the swap of the last component of x_j with the last component of one of its k-nearest time series $x_j^{(i)}$, $i = 1, \ldots, k$ can simply be written as the following random swapping:

$$k - \text{nTS Swapping}: \; P_{j,t} = \begin{cases} A_{j,t}^{(1)} \text{ with probability } \frac{1}{k} \\ \vdots \\ A_{j,t}^{(k)} \text{ with probability } \frac{1}{k}, \end{cases} \tag{6}$$

which is equivalent to the following: the last component of x_j is randomly replaced by the last component of $x_j^{(i)} \in K_j$ with probability $\frac{1}{k}$ for $i = 1, \ldots, k$.

By Algorithm 1, we can obtain X': a matrix of protected time series data at time point t for all J time series for a given rolling window size n. The k-nearest time series data protection method can be written up as the following protected data matrix

$$X' = [x'_1, x'_2, \ldots, x'_J] = \begin{pmatrix} A_{1,t-n+1} & A_{1,t-n+2} & A_{1,t-n+3} & \cdots & A_{1,t-1} & P_{1,t} \\ A_{2,t-n+1} & A_{2,t-n+2} & A_{2,t-n+3} & \cdots & A_{2,t-1} & P_{2,t} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ A_{J,t-n+1} & A_{J,t-n+2} & A_{J,t-n+3} & \cdots & A_{J,t-1} & P_{J,t} \end{pmatrix}^T. \tag{7}$$

[9]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.

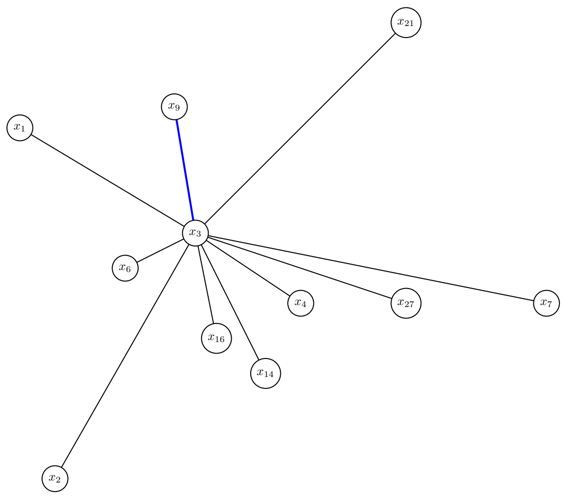

Figure 1: Random edge selection for k-nTS Swapping

As an example, consider a given time series x_3 where its last component is replaced by the last component of x_9 which was randomly selected among x_3's 10-nearest time series. Using our notation, we can write

$$K_3 = \{x_1, x_2, x_4, x_6, x_7, x_9, x_{14}, x_{16}, x_{21}, x_{27}\}.$$

We can represent each time series $x_j,\ j = 1, \ldots, J$ as a vector, and then put them in a graph $G = (V, E)$, which consists of a set V of vertices (or nodes) and a set E of undirected edges. In our case, we can use weighted edges to represent the Euclidean distance between associated time series feature vectors: $w_{i,j} = \mathbf{dist}(f_i, f_j)$. If we put all the nodes on the graph and assign weight on every edge (every pair of nodes), e.g., $w_{i,j} = \mathbf{dist}(f_i, f_j)$ for all $i \neq j$, then we will have a complete graph. The k-nearest time series swapping method can be considered as a random edge selection problem of the graph. Figure 1 depicts the case of $k = 10$ for the k-nearest time series of x_3, where the last component of x_3 is swapped with that of x_9.

4. Empirical Application

4.1. Standard Privacy Methods

4.2. Data

Complicated forecasting models are known to forecast more accurately than simple models using the unprotected version of the M3 competition monthly micro data (Koning et al., 2005), and models that explicitly capture trend and seasonality performed the best in the overall M3 competition (Makridakis & Hibon, 2000). We are interested in whether these results hold when forecasting using protected versions of the data. For our analyses, 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 (Makridakis & Hibon, 2000). Of the 474 series, 18 consist of 67 time periods, 259 consist of 68 time periods, and 197 consist of 125 time periods.