Medical Research Archives

ESMED
European Society of Medicine

OPEN ACCESS

Published: February 28, 2023

Citation: Simmons S.J., Bastin K., et al., 2023. A Comparison of the Prediction Capabilities of Large Scale Time Series Algorithms, Medical Research Archives, [online] 11(2). https://doi.org/10.18103/mra.v11i2.3620

DOI:
https://doi.org/10.18103/mra.v11i2.3620

ISSN: 2375-1924

RESEARCH ARTICLE

A Comparison of the Prediction Capabilities of Large Scale Time Series Algorithms

Susan J. Simmons[1], Kornelia Bastin[1], Aric LaBarr[1], and *Christopher G. Healey[2]

[1]Institute for Advanced Analytics, North Carolina State University, Raleigh, USA, 276061

[2]Department of Computer Science and Institute for Advanced Analytics, North Carolina State University, Raleigh, USA, 27695-8206

*healey@ncsu.edu

Abstract

Since December 31, 2020, the world has closely monitored the progress and outcomes of the SARS-CoV-2 coronavirus (COVID). This paper focuses on two goals. First, we compare time series algorithms for predicting fatalities during the COVID pandemic. Second, we examine how domain affects algorithm choice by comparing our COVID results to historical and current weekly temperature data analyses. Critical interest revolves around tracking and predicting the effects of COVID. Throughout the past three years, many researchers have created models and built visualizations to observe this disease's progression and impact, both regionally and worldwide. Researchers have recently proposed using machine learning to forecast the progression of COVID. With the increased interest in time series methods and the different algorithms available, this paper explores these techniques' accuracy and computational expense. We compare time series analysis approaches for forecasting COVID fatalities from March 11, 2020, to December 28, 2021. The time series models we include are those that can be automatically created to scale to large datasets. Statistical analysis is used to identify significant differences in performance. To investigate generalizability, we apply the same algorithms to predict temperature data, a standard example dataset due to its seasonal and trend components. An analysis is performed both for historical data (1970s) and current data (2020s). Results allow us to: (1) identify significant differences in algorithm performance versus pandemic data with different time series patterns; (2) examine the performance of time series algorithms trained on shorter, constant-length training sets; and (3) determine whether variations in temperature due to climate changes affect how temperature data should now be predicted. We conclude by discussing how domain and data patterns inform the decision of which time series algorithms to consider when predicting future events from historical or existing data. Our results illustrate that no one method is always the best. Careful consideration of the data's domain, the time period in question, and the length of time to analyze must be considered when deciding which algorithm to choose.

Keywords: COVID, time series, temperature

1 Introduction

On March 11, 2020, the SARS-CoV-2 disease changed the world when it was declared a national pandemic by the World Health Organization[1]. Since then, many institutions, governments, agencies, and researchers have been tracking and attempting to predict the severity and spread of the virus[2,3,4]. The world's interest in following the pandemic and its forecasts have inspired many different types of models and analyses[5]. Historically, forecasting the spread of diseases and viral infections is done using epidemiological models such as SIR (Susceptible, Infectious, Recovered) and SEIR (Susceptible, Exposed, Infectious, Recovered)[6,7] or agent-based models[8]. Compartmental approaches such as SIR use differential equations to model the spread of disease. Agent-based models use simulations that allow for spatial information and interactions among individuals[9]. Many assumptions are required regarding population, exposure, and other factors in all of these models.

With the COVID pandemic several models have been utilizing statistical and machine learning (ML) techniques to forecast fatalities from this disease[10]. This increased interest in ML for forecasting provides an opportunity to review forecasting models in the context of the coronavirus disease. Past studies have explored and illustrated some of the differences in time series algorithms. One of the first reviews dates back to 1976[11] when the main focus was on autoregressive integrated moving average models (ARIMA), exponential smoothing models (ESM), and time series decomposition. In a recent survey by[12], linear regression, least absolute shrinkage and selection operator (LASSO) regression, support vector machines (SVM), and ESM are compared to forecast aggregated global COVID data. A more extensive illustration of a survey of methods in Ahmed[13] provides an interesting comparison of multilayer perceptrons, Bayesian neural networks, radial basis functions, generalized regression neural networks (also called kernel regression), k-nearest neighbor regression, classification and regression trees (CART), support vector regression (SVR), and Gaussian processes on an M3 competition dataset. For a recent literature review on numerous algorithms and studies in time series, see[14,15].

Due to the large number of forecasts that need to be made, we investigate the strengths and limitations of methods that can be created automatically. We examine nine-time series and machine learning algorithms to compare their performance in predicting fatalities due to COVID. Traditional methods, including linear regression, SVR, ESM, and seasonal ARIMA models with exogenous variables (SARIMAX), are considered, as well as newer machine learning approaches: Bayesian structural models (BSTS), Prophet, random forests, extreme gradient boosting (XGBoost), and long short-term memory recurrent neural networks (LSTM).

The paper is structured as follows: Section 2 introduces the different algorithms and provides references for a more in-depth understanding of each method. Section 3 introduces the COVID data, the summary statistics obtained, and any preliminary inputs needed for each algorithm. It then provides

the results from each algorithm for predicting fatalities on the date range March 11, 2020, to December 28, 2021, for twenty-eight countries with different COVID mitigation strategies, populations, and macroeconomic properties. We use our results to discuss the advantages and disadvantages of each algorithm. Section 4 applies the same algorithms to a meteorological domain over two time periods with different meteorological patterns. Section 5 provides concluding remarks and future considerations.

2 Time Series Algorithms

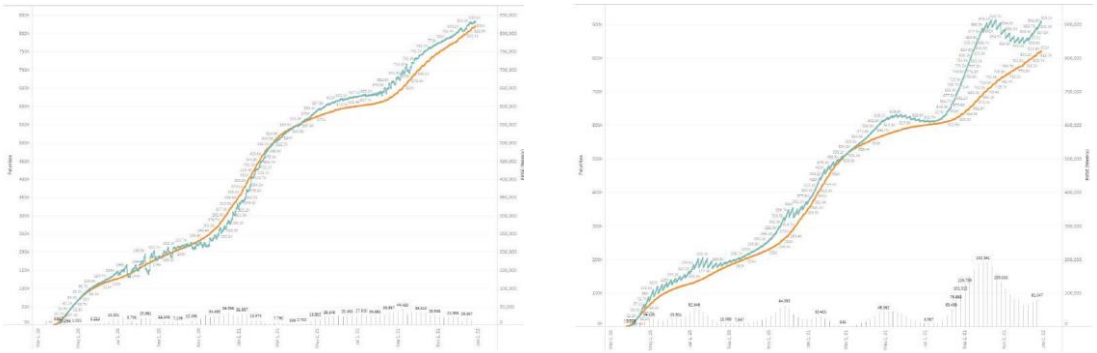
There are many statistical and ML algorithms used to forecast time series data. We focus on linear regression, SVR, ESM, SARIMAX, BSTS, Prophet, random forest, XGBoost, and LSTM. Each of these models is briefly discussed, and further reading is provided.

2.1 Linear Regression

Although linear regression is one of the oldest modeling techniques, it is still relevant in many analyses, both within and outside time series. In linear regression, a response variable $y_i, 1 \leq i \leq n$ exists for n observations. The model for y_i is a linear combination of k known predictor variables $X_1, X_2, \ldots, X_k$.

$$y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_k x_{i,k} + \epsilon_i,\ x_{i,j} \in X_j \quad (1)$$

Linear regression assumes that the response variable is a linear combination of the predictors X_j, together with residuals ϵ_i that are independent and normally distributed with mean $\bar{\epsilon} = 0$ and constant variance $\sigma^2(\epsilon) = c$[16]. Additionally, it is often assumed that there is no perfect multicollinearity between the predictors. The residual ϵ_i is the difference or error between the actual value y_i and the predicted value $\hat{y}_i, \epsilon_i = y_i - \hat{y}_i$. Due to the nature of time series data, there is a potential for correlation between residuals at subsequent points in time, which violates the independence assumption. However, given its simplicity and common use, linear regression is a good reference model[17]. Its closed-form solution means linear regression is a fast and efficient algorithm. Fig. 1a shows an example of linear regression applied to the COVID data we are investigating.



(a) (b)

Figure 1: Known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in the US, March 11, 2020, to December 28, 2021: (a) linear regression; (b) SVR

2.2 Support Vector Regression

Although support vector machines (SVM) are a classification method, they can be extended to regression, usually called support vector regression (SVR, Fig. 1b). First proposed by Vapnik in 1995[18], linear SVM is used for classification by locating a hyperplane that best separates different classes of n-dimensional points containing values for n features. From among all possible hyperplanes, SVM searches for the hyperplane that maximizes the marginal distance: the sum of the distances from the closest point in each class to the hyperplane. Each of these points is termed a support vector.

Mathematically a training set of samples $(x_1, \dots, x_n, y_j), x_i \in X_i$ with known classification y_j is used to generate a set of weights $\{w_1, \dots, w_n\}$. The weights are applied to an unlabeled sample X_k to estimate a corresponding classification $y_k = w_1 x_{k,1} + \cdots + w_n x_{k,n}$. Optimization reduces the number of non-zero weights to only a few corresponding to important features critical to choosing the separating hyperplane.

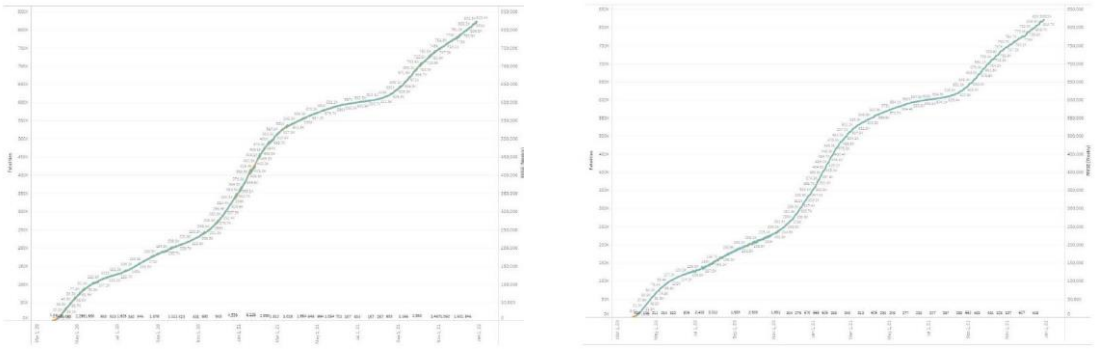
SVR can be seen as a non-linear SVM that applies a kernel function based on the dot product of the original data. The kernel function projects the original data to a higher dimension where it is separable. For example, a kernel function $z^2 = x^2 + y^2$ will add dimension z to convert a doughnutlike structure in $2\mathrm{D}$ into linearly separable data along the y-axis in 3D. The 3D representation corresponds to a circle enclosing the second class of points in the original $2d$ plane[1]. Several standard kernel functions exist: the polynomial kernel $k(x, y) = (x \cdot y + 1)^p$, the radial basis kernel $k(x, y) = \exp(-\| x - y \|^2 / 2\sigma^2)$, or the sigmoid kernel $k(x, y) = \tanh(\kappa x \cdot y - \delta)$.

2.3 Exponential Smoothing Models

Exponential smoothing models (ESM) were proposed shortly after World War II[19]. Basic ESM is called simple or single ESM. It models a single time series data set $y_1, y_2, \dots, y_n$. This model predicts the next observation at time $t + 1$ as a weighted average between the most recent observation y_t and the average $\bar{y}_t$ of the most recent observations (a prediction $\hat{y}_{t+1}$ of y_{t+1}, Fig. 2a)

$$\hat{y}_{t+1} = \theta y_t + (1 - \theta)\hat{y}_t \quad (2)$$

where θ weights the most recent observation versus the average of the previous observations[17,19,20]. Recursively applying this function exponentially decreases an observation's weight the further back in time it occurs. The drawback of single ESM is that predictions beyond one time point in the future $\hat{y}_{t+2}, \hat{y}_{t+3}, \dots$ are the same prediction as $\hat{y}_{t+1}$



(a) (b)

Figure 2: Known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in the US, March 11, 2020, to December 28, 2021: (a) ESM; (b) SARIMAX

Single ESM can be extended to capture trend and seasonality by adding a trend and seasonal component (see Holt[19] and Winters[20]). The seasonal component can be estimated in two ways: as an additive seasonal component or a multiplicative seasonal component. Using an additive seasonal component, the forecasting equation is

$$\begin{aligned} \hat{y}_{t+h} &= l_t + h \cdot b_t + s_{t+h-m} \\ l_t &= \theta(y_t - s_{t-m}) + (1-\theta)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - l_{t-1} - bt - 1) + (1-\gamma)s_{t-m} \end{aligned} \quad (3)$$

where m denotes the length of the season (for example, $m = 12$ for monthly data), β weights the trend component, and γ weights the seasonal component of the model. The multiplicative Holt-Winters model is defined by

$$\begin{aligned} \hat{y}_{t+h} &= (l_t + h \cdot b_t)s_{t+h-m} \\ l_t &= \theta(y_t / s_{t-m}) + (1-\theta)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma\left(\frac{y_t}{l_{t-1} + bt - 1}\right) + (1-\gamma)s_{t-m} \end{aligned} \quad (4)$$

The main difference between the two approaches to seasonality is how the seasonal term is integrated. The additive model adds the seasonal component to the main component. To remove the seasonal component in the level calculation, it is subtracted. The multiplicative model multiplies the seasonal component by adding the level and trend components. Removing this component in the level calculation requires dividing it out of the equation. Additional details around these and other variations of exponential smoothing models are found in Hyndman[21] and Montgomery[17].

2.4 Autoregressive Moving Average Models

The autoregressive integrated moving average (ARIMA) model is one of the most popular methods for analyzing time series data. ARIMA was first proposed by Box and Jenkins in their seminal book Time Series Analysis Forecasting and Control[22]. The ARIMA model combines two types of time series modeling structures: autoregressive (AR) terms and moving average (MA) terms. Both use data at previous time points to predict values at the current time point.

Autoregressive Models. AR models forecast a time series dataset based solely on a series's

past values (lags). For example, an $\mathrm{AR}(2)$ model would be

$$\hat{y}_t = \omega + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t \quad (5)$$

where y_{t-1} is the first lag of Y and y_{t-2} is the second lag. The coefficients ϕ_i represent how much weight to assign to each lag in the model. In AR models, the effects of a single time point diminish as you move further back in time.

Moving Average Models. MA models use previous errors rather than previous response variable values. These error lags are the difference (or residual) between the true value of the response variable at time t and the prediction at t. For example, an MA(2) model would be

$$\hat{y}_t = \omega + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} \quad (6)$$

where ϵ_{t-1} is the error for time $t-i, \epsilon_{t-i} = y_{t-i} - \hat{y}_{t-i}$. The coefficients θ_i represent how much weight to assign to each error lag. In MA models, the effects of an error at a single time point disappear if you move far enough away from that time point.

ARIMA. In an ARIMA model, we assume the series is stationary: the mean, variance, and autocorrelation do not change over time[22]. The data distribution depends only on the difference in time and not the location. Stationarity is the foundation of ARIMA models. Without it, AR and MA models will not produce reliable forecasts. Time series datasets typically need to be stationary before the AR or MA models are built. One common approach is to replace observations y_i with the difference of d neighboring observations $w_t = y_t - \cdots - y_{t-d}$. There are formal tests for the usefulness of differencing to achieve stationarity, such as the Augmented Dickey-Fuller test or the Phillips-Perron test[23,24]. Differencing impacts what the terms represent, while the number of samples and error lags define its structure. For example, if we had a single difference $d = 1$, then our response variable is $w_t = y_t - y_{t-1}$, and the lag terms are based on this difference.

$$\hat{w}_t = \omega + \phi_1 w_{t-1} + \cdots + \phi_p w_{t-p} - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q} + \epsilon_t \quad (7)$$

An ARIMA model is generally denoted by $\mathrm{ARIMA}(p,d,q)$, where p represents the number of AR terms, q represents the number of MA terms, and d represents the number of differencing operations. An ARIMA $(p,1,q)$ model is shown in Eqn. 7.

The ARIMA (p,d,q) model can be extended to include additional variables that do not depend on past values of the response variable. These predictors are analogous to the external predictors used in linear regression. ARIMA models with additional X variables are referred to as ARIMAX[25]. X can be included directly in the ARIMA model structure.

$$\hat{y}_t = \omega + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q} + \epsilon_t \quad (8)$$

The exogenous predictor variables may also be included ahead of time in a linear regression format, while the time series model moves over the residuals of the linear regression model. An example of this would be

$$\hat{y}_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + z_t$$
$$\hat{z}_t = \omega + \phi_1 z_{t-1} + \cdots + \phi_p z_{t-p} - \theta_1 \epsilon_{t-1} - \cdots - \theta_q \epsilon_{t-q} + \epsilon_t \quad (9)$$

where z_t represents the residual at time t and $z_{t-1}, \ldots, z_{t-p}$ are the lags of the residual.

These equations can be extended to include lags for the predictor variables. This is useful if a predictor is not related to the response variable at the same time point. If the predictor variable at previous time points predicts the response variable at the current time, then the predictor variable is referred to as a leading indicator. An example is COVID hospitalizations acting as a leading indicator of COVID fatalities.

SARIMA. Seasonal ARIMA (SARIMA) is an extension of ARIMA that considers seasonal patterns. It does this by including seasonal lags that occur on a seasonal cycle. For example, for monthly data with yearly seasonality, SARIMA uses both the regular lags of ARIMA and lags every twelve observations for the twelve months in a year. SARIMA includes its own P, D, and Q terms, as well as s, to define the seasonal length, $\text{SARIMA}(p, d, q, P, D, Q, s)$. SARIMA can also handle exogenous predictor variables by integrating with ARIMAX to form SARIMAX (Fig. 2b).

2.5 Decision Trees

Before discussing more complicated tree-based models, a brief review of decision trees is needed. A decision tree is built by recursively splitting data into successively purer subsets. These splits are done according to predetermined conditions such as the Gini statistic, entropy, or misclassification rate[26]. A more statistical approach would be to split the data using chi-squared tests. Whatever the decision mechanism, the algorithm searches for subsets of the data to better predict the response variable.

2.6 Random Forest

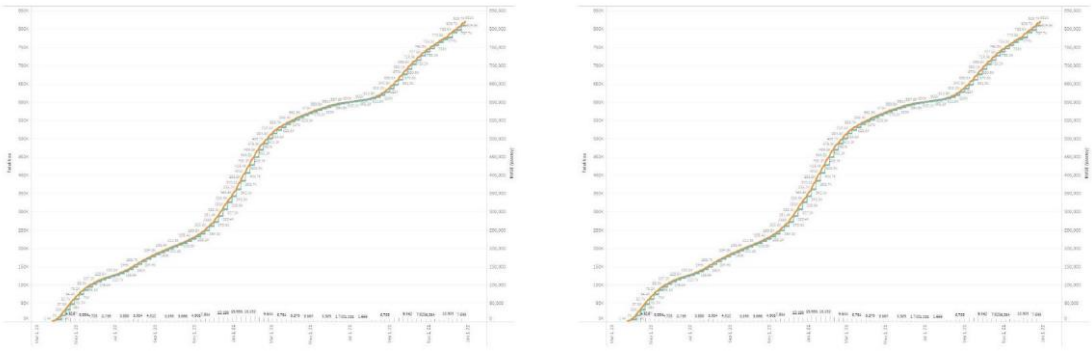
One disadvantage of decision trees is that once a split has been made, all decisions past that point depend on that previous split. A different set of training variables or observations could produce a different split. To account for this, ensembling decision trees can provide more reliable models.

A related approach to generalizing training data uses bootstrap samples. Here, random samples of the dataset with replacements are created. Replacement ensures that only a subset of the original observations will be present in the bootstraps. These observations are called out-of-sample observations. Taking many bootstrapped samples and aggregating the resulting models (bagging) simulates ensembling to provide a potentially better estimate.

Random forests were proposed by Leo Brieman in the early 2000s[27] and are an excellent example of bagged trees (Fig. 3a). Many decision trees (a forest of trees) classify or predict the response variable. In a classification setting, each tree "votes" for the class to which an observation belongs. In regression, each tree provides estimated values for an observation that are averaged together.

Each tree in the forest is created from a subset of observations based on the bootstrapped samples. Each split in the tree is determined from a random subset of predictor variables in the dataset. In a random forest model, cross-

validation is typically performed to optimize the best size of the random subset of predictor variables. Splits are determined by the best feature of the selected subset. One can also tune the number of trees or the bootstrapped sample in the forest and set limits for when splitting can occur and how deep each tree can grow.



(a) (b)

Figure 3: Known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in the US, March 11, 2020, to December 28, 2021: (a) random forest; (b) XGBoost

2.6.1 Gradient Boosting

Gradient boosted algorithms were developed in 1996 by Freund and Schapire[25]. Similar to bagging, boosting draws multiple samples from a dataset with replacement to ensemble multiple models. Boosted samples are not drawn at random. Boosting assigns weights to each sampled observation to use as a sampling distribution. Higher weighted observations are more likely to be drawn. Weights adaptively change in each round of samples, where a weight is increased for more difficult-to-classify observations. Unlike bagging, boosting considers samples sequentially to learn from mistakes in the previous model. In the first iteration, all observations have an equal chance of being selected using a standard decision tree. Observations that are predicted poorly are assigned higher weights for the following sample in the sequence.

Gradient boosting has been shown to excel at modeling both continuous and categorical response variables. There are many variations of gradient boosted algorithms: additive boosting (AdaBoost), gradient boosted machine (GBM), light GBM, and CatBoost for categorical variables. One of the most popular variations is XGBoost, developed by Chen and Guestrin[28] (Fig. 3b). Rather than boosted samples of the original data, XGBoost uses the residuals of the previous model. These models are added together to form a final prediction.

$$\hat{y}_i = f_1(x) + f_2(x) + \cdots + f_m(x) \quad (10)$$

$f_1(X)$ is a function of the predictor variables to estimate the response variable Y. $f_i(X), 2 \leq i \leq m$ does not predict Y, but instead, the residuals from model f_{i-1}. This fitting continues until convergence of the predictions is achieved. Unfortunately, XGBoost in its original form tended to overfit the data. To help prevent this, Chen and Guestrin[28] added penalty terms to f_i, revising its structure.

$$\hat{Y}_i = f_1(X) + \eta f_2(X) + \cdots + \eta f_m(X) \quad (11)$$

Table 1: XGBoost parameters and descriptions

Parameter	Description
booster	The overall structure of the model. gbtree and dart are tree-based models applying gradient descent and tree dropout, respectively, for optimization. gblinear uses a linear function with L1 and L2 regularization (equivalent to LASSO and ridge regression, respectively). The default is gbtree
η	The rate the learner is allowed to learn. The learning rate is in the range $0 \ldots 1$, where 1 imposes no penalty, Usually, η is less than 0.3, with a default of 0.3.
max_depth	The maximum depth limits the depth a tree can grow. Allowing trees to grow without bounds will overfit the data. The default maximum depth is six.
nrounds	The number of trees used in the final ensemble model.

There are numerous additional components of the XGBoost algorithm that can be tuned with cross-validation, similar to random forests. Some of the most impactful options are listed in Table 1[29].

2.7 Bayesian Structural Time Series

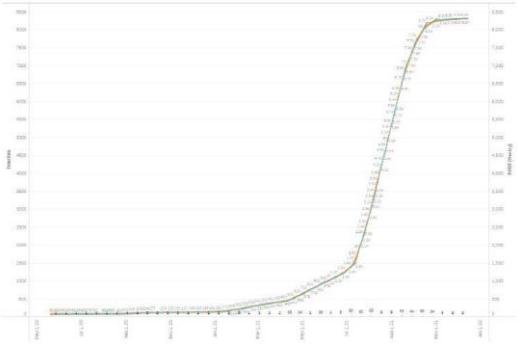


Figure 4. BSTS known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in the US, March 11, 2020, to December 28, 2021

ESM was the first state space model for time series data. Recently advances have been introduced to better model the evolution of a time series data set over time. Bayesian structural time series models (BSTS) were initially proposed by Harvey in 1989[30] and further refined by Durbin and Koopman in 2002[31]. BSTS creates a structural time series model that captures trend and seasonality in state equations (Fig. 4). Researchers at Google[32] developed an R program that is easy to implement and scales to large data sets. Eqn. 12 illustrates how the observed data y_t at time t is linked to the unobserved latent state μ_t and τ_t

$$y_t = \mu_t + \tau_t + \epsilon_t \quad (12)$$

where $\epsilon_t \sim N(0, \sigma_t^2)$. A transition equation is defined for the latent state of μ_t and τ_t.

$$\mu_t = \mu_{t-1} + \delta_{t-1} + u_t$$
$$\delta_t = \delta_{t-1} + v_t$$
$$\tau_t = -\sum_{s=1}^{S-1} \tau_{t-s} + w_t \quad (13)$$

where $u_t \sim N(0, \sigma_u^2), v_t \sim N(0, \sigma_v^2)$ and $w_t \sim N(0, \sigma_w^2)$. To fit the state space models, a Kalman filter and Kalman smoother are used to compute latent distributions throughout the series[30]. In addition to modeling trends and seasonality, the algorithm developed by Google's researcher can incorporate external regressors.

$$y_t = \mu_t + \tau_t + \sum_t X\beta + \epsilon_t \quad (14)$$

2.8 Prophet

Researchers at Facebook developed the Prophet algorithm[33] to forecast a wide variety of business problems. Prophet is similar to an exponential smoothing or Bayesian structural time series model, where a time series is subdivided into separate components that are combined.

$$y_t = g(t) + s(t) + h(t) + \epsilon_t \quad (15)$$

where $g(t)$ is a trend component, $s(t)$ is a periodic or seasonal component, and $h(t)$ is a holiday component.

The trend component uses time as a predictor variable to develop trend lines. This trend is broken into different neighboring subsets using knots (change points) if needed. The user can specify knots or allow the algorithm to automatically choose them based on minimizing the sum of squared errors. The trend can be non-linear, using a logarithmic function to generate a dampened trend similar to exponential smoothing.

A Fourier transformation is assumed for the seasonal component[34]. Fourier transformations allow the algorithm to handle multiple seasonal periods and lengths. By default, Prophet assumes daily data use and includes annual and weekly seasons. The annual seasonal pattern uses ten Fourier transformation predictor variables.

$$X_Y = \cos\left(\frac{2\pi t}{365.25}\right) + \sin\left(\frac{4\pi t}{365.25}\right) + \cos\left(\frac{6\pi t}{365.25}\right) + \cdots + \sin\left(\frac{20\pi t}{365.25}\right) \quad (16)$$

The weekly seasonal pattern uses three Fourier transformation variables.

$$X_W = \cos\left(\frac{2\pi t}{7}\right) + \sin\left(\frac{4\pi t}{7}\right) + \cos\left(\frac{6\pi t}{7}\right) \quad (17)$$

The 365.25 and 7 in the above equations control the annual and weekly seasons' lengths in days. These are only defaults and are adjustable in the algorithm. If the data is not seasonal, all seasonal components can be removed.

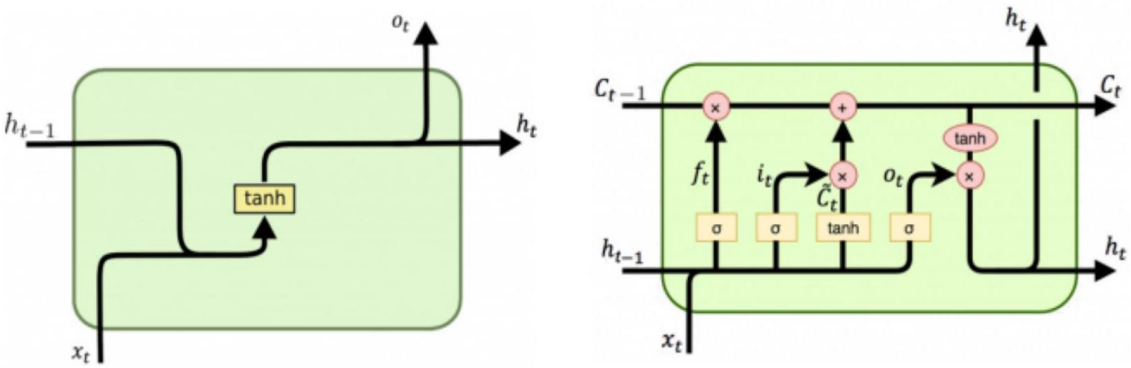
The holiday component consists of a series of binary intervention variables that take a value of one on days of interest to account for holiday effects. The Prophet algorithm has built-in national holidays for several countries. Users can also specify their own list of holiday dates to be used in the model.

2.9 LSTM

Recently, the area of deep neural networks and deep learning (DNNs and DL) have been applied in numerous domain areas with

noteworthy success[35]. Historically, deep neural networks are an extension of the original work on artificial neurons and perceptrons from the 1940s and 50s[36,37]. Different "deep" neural network structures contain two or more hidden layers (fully connected networks or FCNs), use convolutional kernels to extract features at different levels of abstraction (convolutional neural networks or CNNs), or adopt a recursive structure to analyze sequence data (recurrent neural networks or RNNs). Generative adversarial networks (GANs) were later presented to combine two DNNs: a generator that constructs artificial objects and a discriminator that tries to decide whether an object is artificial or real[38]. By carefully balancing the learning rates of the DNNs, they can act together to rapidly improve their performance. Most recently, the transformer model has been proposed to extend CNNs and RNNs using a mathematical model of attention, the ability to encode relational context between elements in a training sample[39].

Since time series data is an ordered sequence of samples, RNNs are an obvious DNN to apply. In a classic RNN, "memory" can be maintained over a sequence of data by providing two inputs to the deep neural network: the current element x_t in the sequence and the hidden state h_{t-1} of the previous execution of the RNN (Fig. 5a). The output $o_t = h_t$ is sent in two directions. First, o_t acts as the input layer to an FCN to calculate probabilities for predicting each possible category. Second, h_t is sent forward to serve as the previous hidden state for the next recurrence of the RNN.

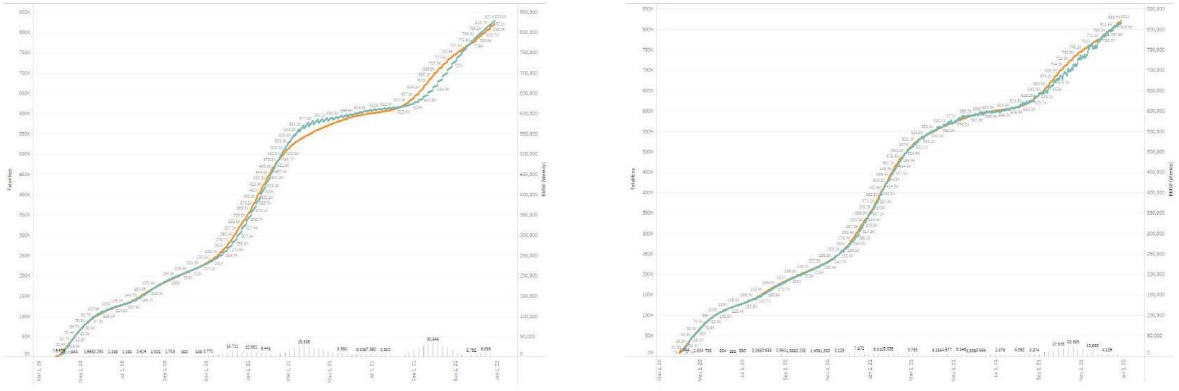


(a) (b)

Figure 5: Recurrent neural networks with previous hidden state h_{t-1}, current sequence element x_t, current hidden state h_t, previous cell state C_{t-1}, and current cell state C_t: (a) classic RNN; (b) LSTM showing the forget gate f_t, the input gate $i_t \odot \tilde{C}_t$, and the combination of long and short-term memory $\sigma(h_{t-1}, x_t) \odot C_t$

Unfortunately, there are various issues with classic RNNs. A critical question is how far back a classic RNN can maintain long-term memory. Due to the vanishing gradient problem during backpropagation of error, classic RNNs can only "remember" a limited number of previous steps in the input sequence. Because of this, their memory is often considered to be short-term only.

LSTMs address this issue using an additional cell state C_t representing long-term memory (Fig. 6). During each recurrence of an LSTM RNN h_{t-1} and x_t are combined as before. However, this result is fed through two gates: a forget gate and an input gate. The forget gate uses pointwise multiplication with C_{t-1} to determine which parts of long-term memory to forget. The input gate uses pointwise addition to select the parts of the current input element to add to long-term memory (Fig. 5b). Updated long and short-term memory are combined using pointwise multiplication to produce the cell's hidden state $h_t = \sigma(h_{t-1}, x_t) \odot C_t$. This process is repeated over the entire input sequence, classification error at the final step is calculated, and the error is backpropagated through the LSTM cell to update its weights and biases via optimization.



(a) (b)

Figure 6: Known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in the US, March 11, 2020, to December 28, 2021: (a) Prophet; (b) LSTM

3 COVID Investigation

We investigate the time series algorithms by using them to predict COVID fatalities based on three predictors: day of the week, confirmed COVID cases during the last week $(c_1, \ldots, c_7)$, and the number of days since the time series began (n). Since the day of the week needed to be one-hot encoded to avoid overweighting certain days, this produced a total of fifteen exogenous variables (Table 2). To examine the effects of different population sizes, locations throughout the world, socioeconomic status, and COVID mitigation strategies, we chose 28 separate countries for our study.

- G7: Canada, France, Germany, Italy, Japan, United Kingdom, United States
- BRIC: Brazil, Russia, India, China
- African Union subset: Algeria, Ethiopia, Egypt, Kenya, Morocco, Niger, Nigeria, South Africa
- Asia & Oceania: Australia, New Zealand, Philippines, South Korea
- Central America: Guatemala, Costa Rica, Cuba
- South America: Peru, Venezuela

Table 2: COVID fatality exogenous predictor variables

Date	M	Tu	W	Th	F	Sa	Su	c_1	c_2	c_3	c_4	c_5	c_6	c_7	n
03/11/20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
03/12/20	0	0	0	1	0	0	0	0	0	0	0	0	0	32	1
03/13/20	0	0	0	0	1	0	0	0	0	0	0	0	32	38	2
	...														
12/28/21	0	1	0	0	0	0	0	1217	1229	1318	1392	1425	1476	1509	656

Daily COVID data for March 11, 2020, through December 28, 2021, was obtained from the John Hopkins University Center for System Science Engineering COVID repository[9,40]. The data was divided into 94 weekly blocks. We started by using the first three weeks of data to predict fatalities for the fourth week and compare them to known fatalities. Average root mean squared error (RMSE) was used to evaluate the accuracy of the predictions. We then use the first four weeks to predict the fifth week, the first five weeks to predict the sixth, and so on, ending by using the first 93 weeks to predict the 94th. This approach produced 91 prediction weeks for each of the 28 countries we tested.

The day of the week and n (days since March 11, 2020) were determined directly during prediction. $c_1, \ldots, c_7$ needed to be estimated, however, since the predictions could not be based on known case counts. To do this, we used confirmed cases from the three weeks prior to the prediction period to "estimate" cases during the prediction week using XGBoost. XGBoost was selected for its accuracy and reasonable execution time for predicting COVID fatalities. Confirmed case predictions were needed for linear regression, LSTM, random forest, SVR, and XGBoost[2].

We started our analysis by investigating whether RMSE was significantly different across ML methods. Since we could not guarantee constant variance, we calculated the more conservative Kruskal-Wallis analysis of variance (ANOVA). For all countries $211.96 \leq \mathrm{H}(8) \leq 527.34, p <$ 0.001. Given this, we collapsed results across country to compute a single test for the difference in RMSE over ML method. The result was significant, $\mathrm{H}(8) = 2797.12, p = 0.0$. We continued by performing post hoc pairwise significance tests using Dunn's algorithm. Fig. 7 uses a heatmap to identify significant pairwise differences. Cells are colored dark red, light red, pink, and grey for $p < 0.001, p < 0.01, p < 0.05$, and $p \geq 0.05$, respectively. The cell's row identifies which of the two methods outperformed the other. For example, the pink cell with $p = 0.024$ occurs in the ESM row and the BSTS column. This indicates a significantly better RMSE for ESM versus BSTS.

Figure 7: Heatmap of pairwise significant RMSE differences by ML method: red for $p < 0.001$, light red for $p < 0.01$, pink for $p < 0.05$, grey for $p \geq 0.5$, the number in each cell represents Dunn's p-value

Examination of the heatmap shows that SARIMAX outperformed all other methods. ESM outperformed all methods except for SARIMAX. LSTM also showed numerous significant RMSE results. Since the method on a red or pink cell's column has a significantly higher RMSE, we can look down the columns to identify poor-performing techniques. Perhaps not surprisingly, linear regression and SVR produced significantly higher RMSEs than all other methods. Prophet also showed relatively poor performance.

The accuracy pattern across all countries does not extend to individual countries verbatim. Figures 8a-d show heatmaps for Australia, China, New Zealand, and the United States. Each heatmap shows a pattern that differs from the others. For Australia, BSTS and ESM perform best, with SARIMAX and LSTM producing good results (Fig 8a). In China and New Zealand, however, BSTS performs poorly (Fig. 8b, c). Performance could be due to the few fatalities reported in these countries. Most average RMSEs in China and New Zealand for each method, shown on the vertical axis, are still comparable to Australia and the United States. BSTS outperforms all other methods in the United States, with SARIMAX producing significantly better accuracy than all methods except BSTS (Fig. 8d). ESM and LSTM also perform well.

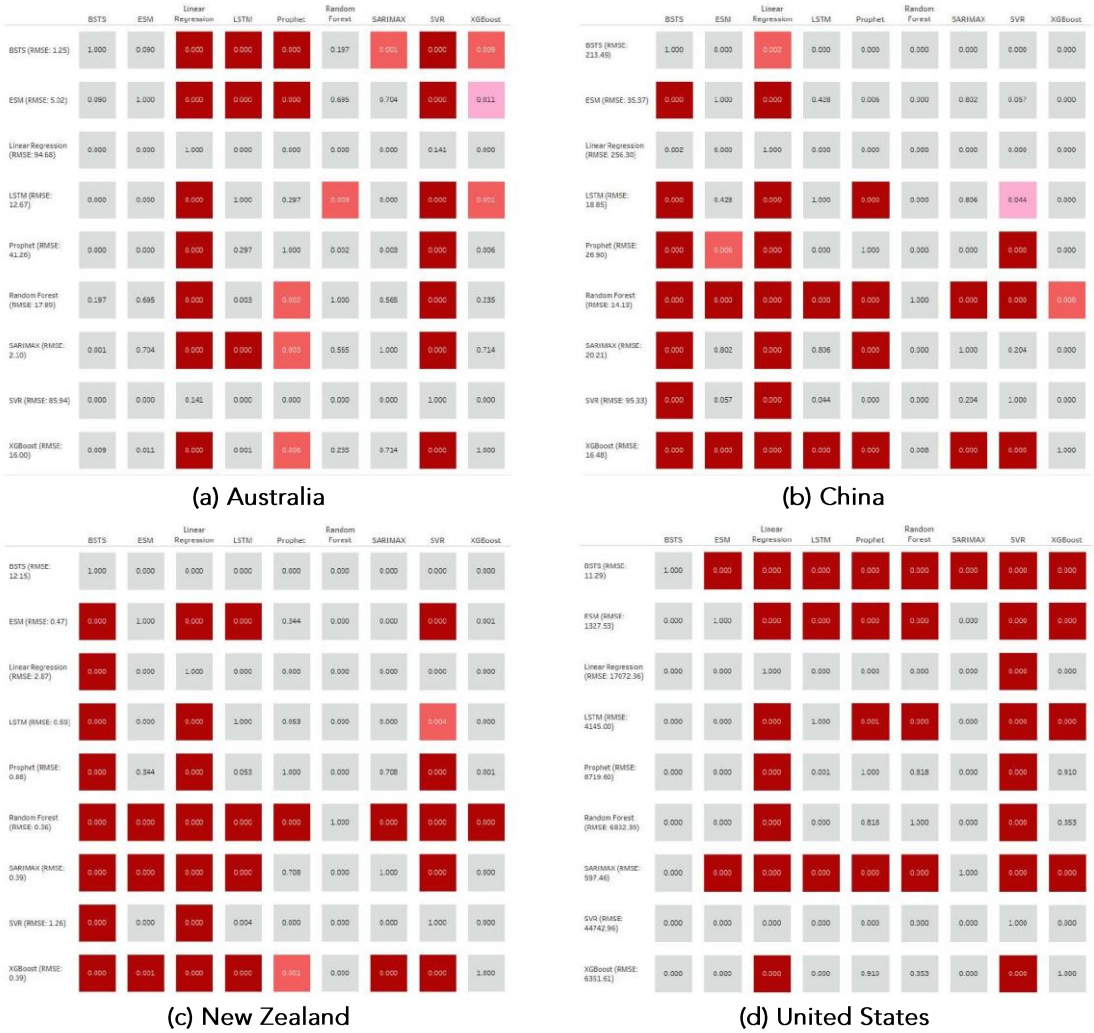


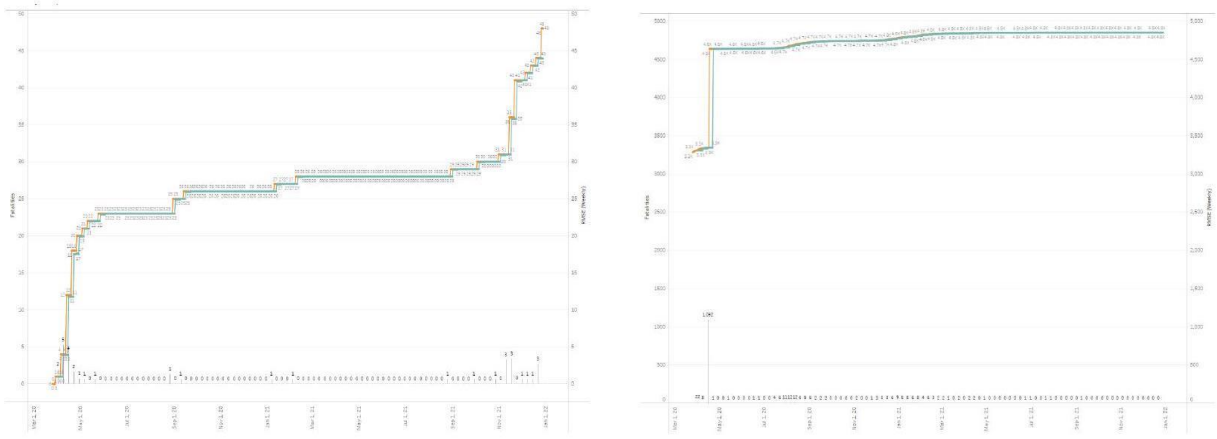
Figure 8: Heatmap of pairwise significant RMSE differences by ML method: red for $p < 0.01$, pink for $p < 0.05$, grey for $p \geq 0.5$, the number in each cell represents Dunn's p-value; (a) Australia; (b) China; (c) New Zealand; (d) United States

Tree-Based Prediction Patterns. A closer inspection of the random forest and XGBoost prediction graphs shows a weekly stair-step pattern (Fig. 3). Both algorithms predicted identical or nearly identical fatalities for each prediction week, anchored at the last known fatality value in the training set. Further investigation revealed this is due to the type of data in the COVID dataset, specifically the confirmed case counts. These increase monotonically over time. Because of this, known confirmed case counts in the training set will always be lower than estimated confirmed case counts in the test set. Since random forest and XGBoost never saw this level of confirmed case count during training, they default to predicting the highest target fatality value that was seen. Confirmed case counts outside the training set range explain why the seven predicted fatality values match the highest fatality value in the training set.

Tree performance demonstrates an important point when using ML algorithms on time

series data. Depending on the pattern of the data being analyzed, the direct application of certain ML techniques may not generate accurate results. In this case, we would need to enforce stationarity on both the training and test sets to improve the performance of tree-based algorithms. However, if this was done, it could be argued that other ML algorithms that either already do this (SARIMAX) or do not need this (ESM) are better choices.

Outlier Prediction Accuracies. Although both random forest and XGBoost struggled to predict fatalities due to the monotonically increasing nature of confirmed case counts, this is at odds with the strong performance shown by both algorithms for New Zealand and China (Fig 8b-c). Why are random forest and XGBoost performing well for these two countries? The answer lies in the number and reporting pattern of their fatalities. Fig. 9a shows New Zealand's very low COVID fatality rate, resulting in a flat curve with only a few upward shifts. In this environment, the flat predicted fatality rates produced by random forest and XGBoost match the known case counts, producing accurate results. China also has a relatively flat fatality rate, but it is due to the reporting granularity and China's "zero COVID" policy. Again, random forest and XGBoost predict flat fatality patterns well. This highlights how patterns within the time window under examination can favor certain ML algorithms over others.



(a) New Zealand (b) China

Figure 9: Random forest known (orange) and predicted (green) COVID fatalities and weekly RMSE (vertical lines) in (a) New Zealand; (b) China

3.1 Execution Time

The time to run each method may need to be balanced against the method's RMSE accuracy. Fig. 10a shows the time to run for all 28 countries by ML technique (2,548 individual runs per country). The relative differences are most important, since the compute environment affects execution. Times range from 11 seconds for linear regression to 13h 1m 38s for SARIMAX, a 4263× difference. Kruskal-Wallis ANOVA and Dunn's post hoc test confirmed significant

pairwise differences in runtime by ML method ($H(7) = 214.01, p < 0.001$). Fig 10 b shows pairwise significant differences using the same heatmap setup as before but comparing runtimes rather than average RMSE. Red in a method's row shows it was faster with $p < 0.001$ versus the corresponding column method. Light red and pink correspond to $p < 0.01$ and $p < 0.05$, respectively.

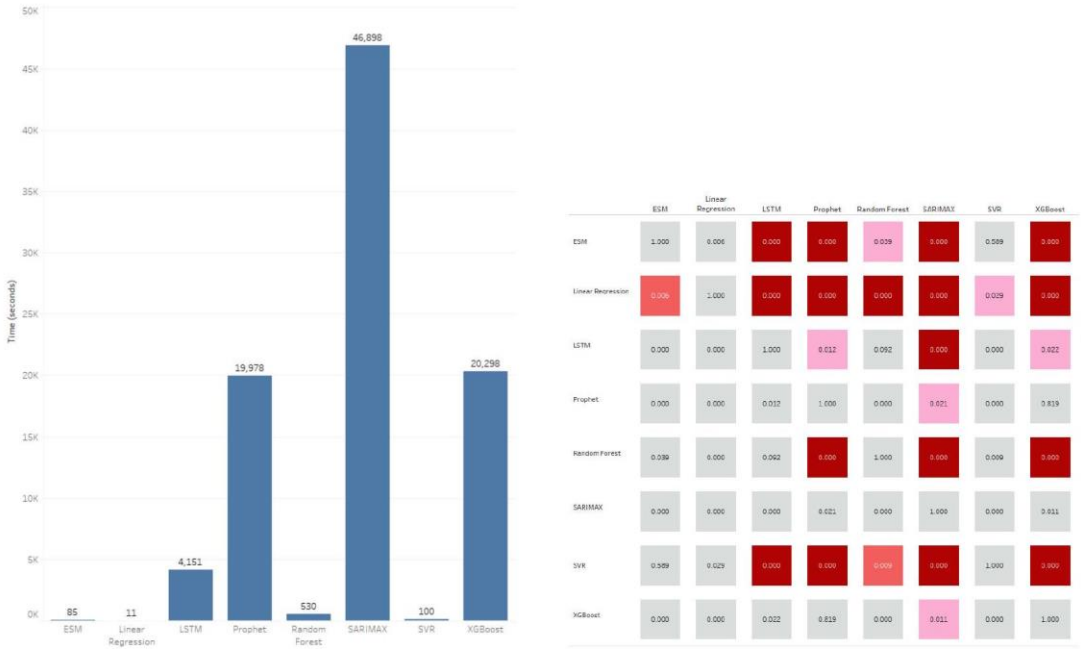


Figure 10: Time in seconds for each algorithm to complete 93 predictions over 28 countries † : (a) time; (b) methods with statistically significant time differences: red for much faster, light red for faster, pink for moderately faster

For the COVID data we tested, we conclude that ESM offers the best tradeoff between accuracy and runtime, although ESM's RMSE will increase if we predict out farther than one week.

Another question is whether longer training sets produce more accurate results. A Dickey-Fuller stationarity test on the sequence of RMSEs for each country should identify this as a non-stationary sequence due to the non-stationary long-term mean. Follow-on linear regression would show a negative slope for decreasing RMSEs or a positive slope for increasing RMSEs. Results showed no obvious advantage to longer training sets for the COVID data, however. Fig. 11 shows the number of non-stationarity ML sequences for each country we tested: red for a positive average μ, and blue for a negative average μ. For example, Russia had an overall slope for non-stationary sequences of $\mu = 72.78$ with six rising slopes (LSTM, linear regression, Prophet, random forest, SVR, and XGBoost), no falling slopes, and by inference, three stationarity slopes (BSTS, ESM, and SARIMAX).

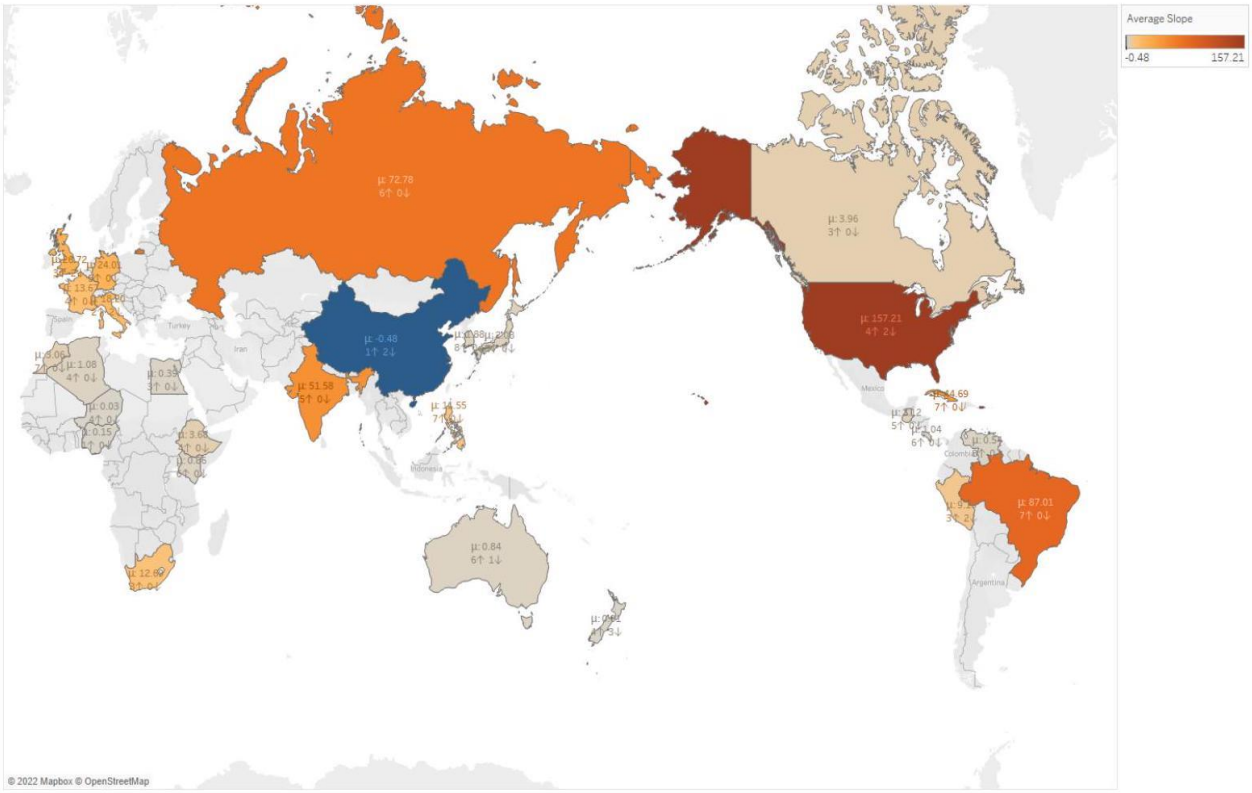


Figure 11: Stationarity by country: average slope for non-stationary ML technique μ in the center of each country, followed by the number of ML techniques with rising (up-arrow) and falling slope (down-arrow)

4 Meteorological Investigation

Following our study of COVID data, we decided to examine time series ML algorithms for predicting meteorological data. We chose meteorological data for several reasons:

- To investigate the generality of our findings for a different domain,
- to study whether a fixed training window produced results similar to the monotonically increasing training window for the COVID data,
- to see if historical temperature trends could be predicted with an accuracy comparable to current temperature trends (i.e., are the effects of climate change visible within time series prediction), and
- to work with data, particularly temperature data, which is a widespread "default" used to explain time series analysis since, at the proper granularity, it contains well-defined trends and seasonality.

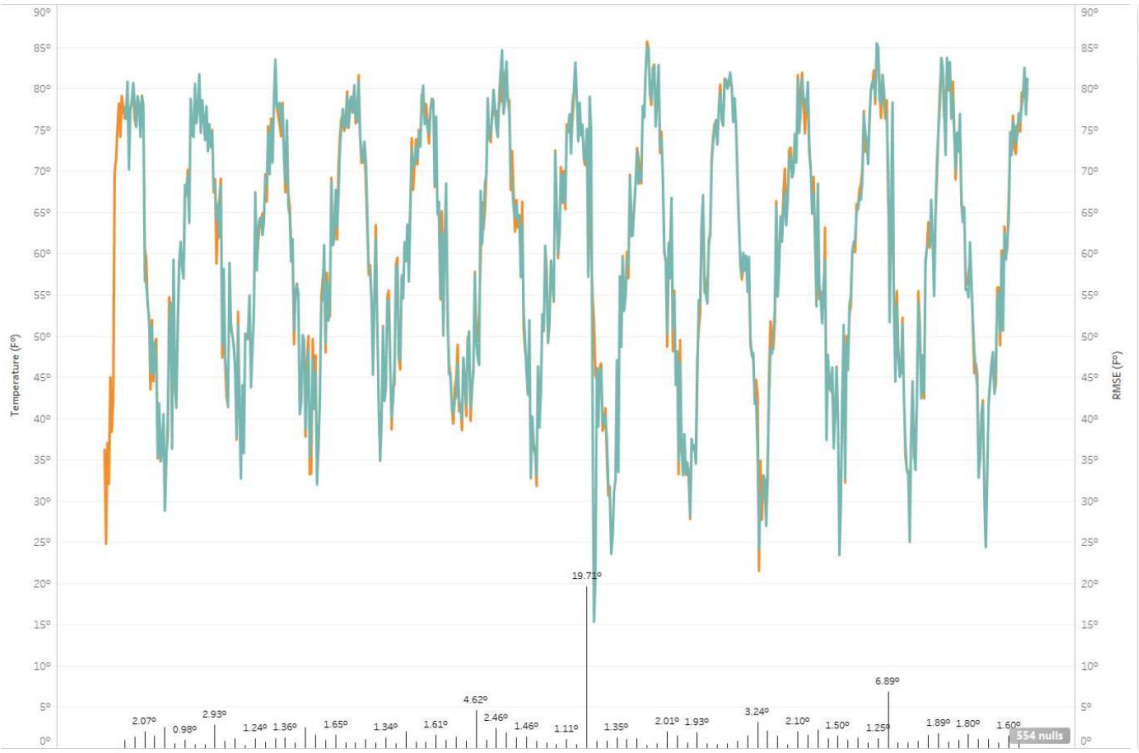


Figure 12: Known (orange) and predicted (green) weekly temperature medians using SARIMAX for the United States from January 7, 1970, to August 11, 1982

Since our interest was in generality and training set size, we focused on only one country, the United States. Choosing the United States also simplified data collection since meteorological data is readily available from the National Oceanic and Atmospheric Administration (NOAA, https://www.noaa.gov). We initially selected daily data, but this produced a temperature pattern with significant noise and no apparent seasonality. Because of this, we decreased the granularity to weekly steps. The data was still noisy but did exhibit the trend and seasonality we wanted (Fig. 12). Weekly data had the added advantage of allowing us to query approximately fourteen years of data to generate 91 executions per ML technique, identical to the number of executions for the COVID data. Fourteen years provides an extended timeframe to capture seasonal patterns.

From this starting point, we selected two time windows to investigate. The historical time window runs from January 7, 1970, to August 11, 1982. The current time window contains the most recent NOAA data, from November 16, 2009, to July 2, 2022. To study the effects of short, fixed-length time windows, each execution of an ML algorithm used three weeks of data to predict the upcoming fourth week. Constant training set length means, unlike the COVID data, we used the first three weeks from our dataset to predict the fourth week's data, weeks two through four to predict the fifth week's data and so on, finishing by using weeks 91 through 93 to predict the 94th week's results.

We used two exogenous variables: maximum average weekly temperature and month, to predict the median average weekly temperature. As with the COVID data, the month of the year was one-hot encoded to avoid overweighting months later in the year. XGBoost was again used to generate the predicted maximum average weekly temperature over the test window. We ran the same ML algorithms used during the COVID experiments, with median temperature RMSE (in Fahrenheit) as a measure of accuracy. A Kruskal-Wallis ANOVA showed a significant difference in RMSE between ML methods, $H(8) = 333.61, p < 0.001$ and $H(8) = 211.63, p < 0.001$ for the historical and current time periods, respectively. Dunn's post hoc analysis identified pairs of methods with significantly different RMSE. Fig. 13 shows significance tables for historical and current temperature medians in the top two heatmaps (Fig. 13a, b). For comparison, the significance heatmap for the United States COVID data is shown in the bottom image (Fig. 13c).

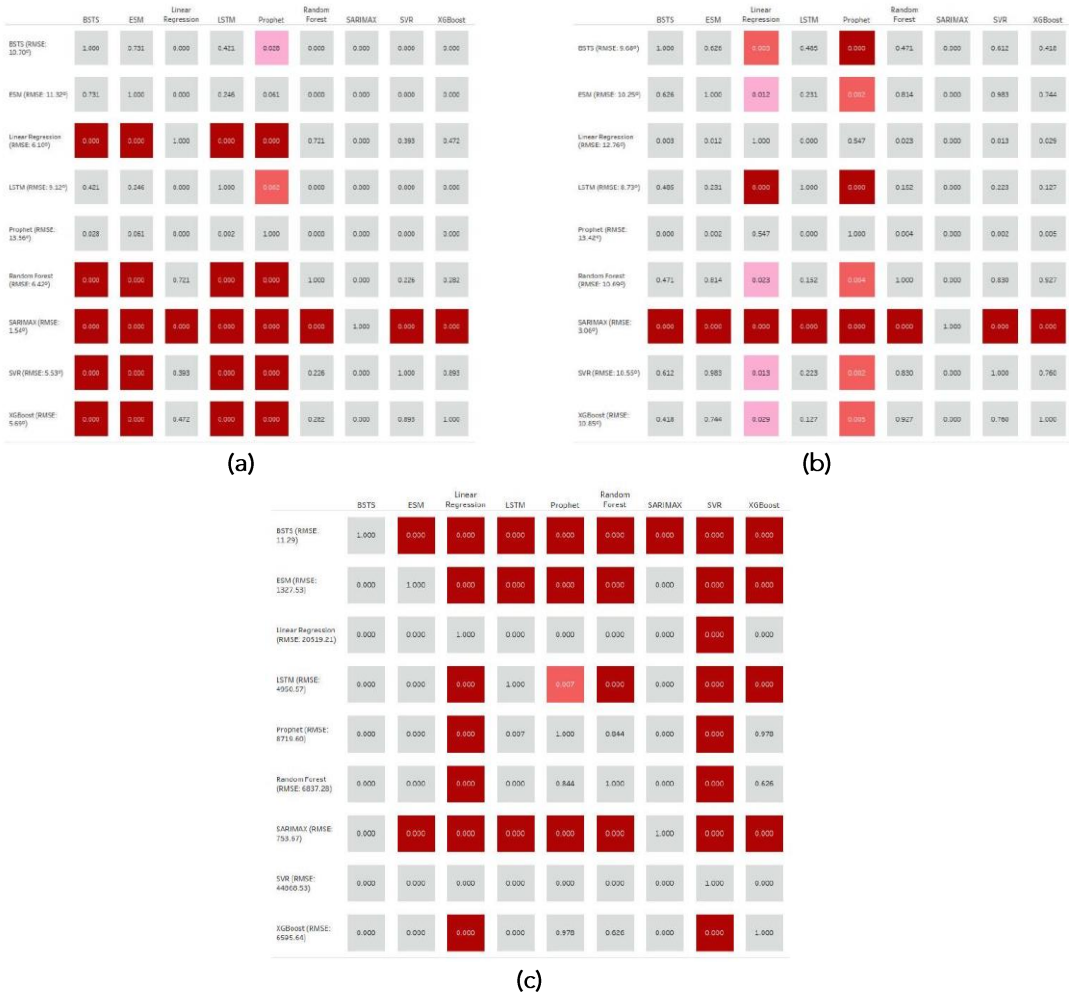
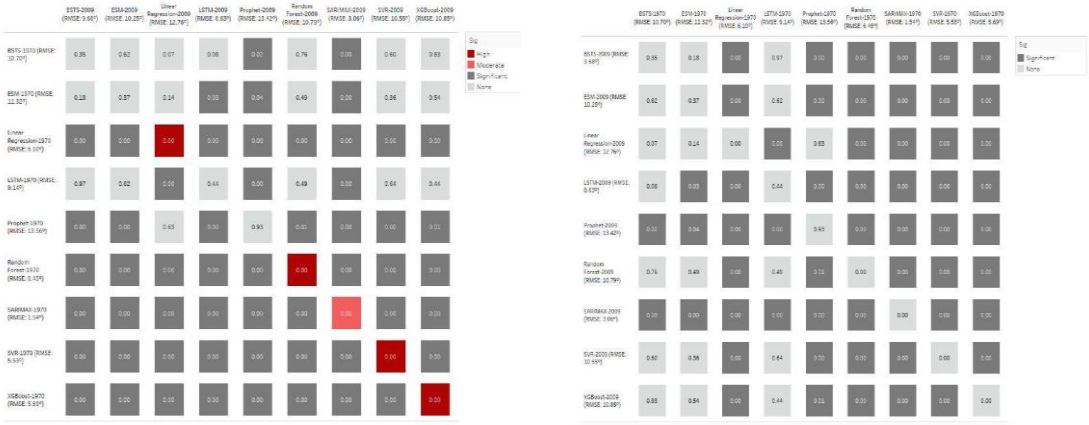


Figure 13: Pairwise significance heatmaps for the United States: (a) median weekly temperatures, January 7, 1970, to August 11, 1982; (b) median weekly temperatures, November 16, 2009, to July 2, 2022; (c) daily COVID fatalities, March 11, 2020, to December 28, 2021

XGBoost also performed comparatively better on historical versus current data. SARIMAX again performed best on both datasets. From a statistical significance perspective, for historical data SARIMAX performed best, linear regression, SVR, random forest, and XGBoost performed well, and LSTM and Prophet performed poorly. For current temperature data SARIMAX performed best, linear regression and Prophet performed poorly, and the remaining methods performed comparably.

To conclude, we conducted a statistical performance analysis on historical versus current temperature data. Kruskal-Wallis showed a significant difference in $\text{RMSE}, \text{H}(8) = 887.57, p \approx 0.00$. Dunn's post hoc probabilities were calculated for historical versus current temperature predictions (Fig. 14a) and current versus historical temperature predictions (Fig. 14b). As before, red, light red, and pink represent $p < 0.001, p < 0.01$, and $p < 0.05$, respectively. For the heatmaps in Fig. 14, the diagonal is most important since it compares performance for the same ML method: historical versus current in Fig. 14a and current versus historical in Fig. 14b. All other significant pairwise results are shown in dark grey to identify them but to reduce their emphasis versus results along the diagonal. Fig. 14 confirms that historical temperature predictions were significantly more accurate using linear regression, random forest, SARIMAX, SVR, and XGBoost. In no case were current temperature predictions significantly more accurate than their historical counterparts.



(a) (b)

Figure 14: Pairwise significance heatmaps for historical versus current temperature predictions: (a) 1970 versus 2009, rows represent historical results, columns represent current results; (b) 2009 versus 1970, rows represent current results, columns represent historical results

5 Summary of Results

Results from our investigation of machine learning techniques used for time series prediction showed that accurate performance could be achieved, even for short training windows (e.g., three weeks of training to predict one week into the future.) For the COVID data, the average per day RMSE was 77 cases across all 28 countries tested using SARIMAX. For historical and current

temperatures, the average per day RMSE for the United States was $1.54°$ and $3.06°$ using SARIMAX, respectively.

A summary of our results includes the following conclusions.

1 The domain and time window within the domain can influence the choice of ML prediction algorithm since different algorithms are better or worse at capturing different types of time series patterns.
2 Certain algorithms can perform poorly if their training set does not include examples of exogenous variables contained in the test set (e.g., random forest and XGBoost).
3 Time permitting, SARIMAX produces high-quality results across various domains and time windows.
4 For the smooth, monotonically increasing COVID fatality data, ESM performed comparably to SARIMAX but was two orders of magnitude faster.
5 For high variance, high temporal frequency temperature data, methods like linear regression, SVR, random forest, and XGBoost can produce significantly better accuracy than methods like LSTM and Prophet.
6 Five of the nine algorithms we tested (linear regression, random forest, SARIMAX, SVR, and XGBoost) produced significantly less accurate temperature predictions for current versus historical temperatures. Although further investigation is needed to determine precisely why this is, it suggests more randomness in the temperature patterns, something no algorithm can adjust for.

6 Conclusions

An important finding from our studies is that there is no obvious "one size fits all" method to apply. This choice will depend on the domain in question and the time window within that domain since both can affect the response variable patterns. Different techniques can be better or worse for certain patterns. For example, ESM did well with the smoothly changing COVID fatalities but was much less accurate for high temporal frequency changes in the median temperature values. Execution cost is also a factor. Time permitting, SARIMAX will usually generate high-quality results, but at the cost of significantly longer runtimes needed to identify optimal input parameters.

Items for future work are numerous, but three are being prioritized. First, we are curious to see if more recent supervised ML methods like transformer-based deep learning produce accurate time series predictions[39]. These new deep learning structures are replacing RNN LSTMs since they use self-attention mechanisms to choose where to focus their efforts over a data sequence rather than processing items in the sequence in the order they occur. Second, we would like to test the ML techniques on additional domains to see whether specific time series patterns favor particular approaches. This is similar to how high temporal frequency data was better managed by the tree and linear methods versus BSTS and ESM. Third, we plan to

investigate further whether differences in prediction accuracy can be used to highlight changes in real-world patterns, like the effects of climate change on meteorological conditions.

Corresponding Author:
Christopher G. Healey
Department of Computer Science and
Institute for Advanced Analytics
North Carolina State University
Raleigh, USA, 27695-8206
E-mail: healey@ncsu.edu

Conflicts of Interest:
The authors have no conflicts of interest to declare.

Acknowledgement:
None

Funding:
None

References:

1. WHO timeline—COVID-19. World Health Organization. 2020. Accessed January 21, 2023. https://www.who.int/news-room/detail-27-04-2020-who-timeline--covid-19/.

2. Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet* 2020;20:533–534.

3. Delphi Epidata API. Farrow, DC, Brooks LC, Rumack A, Tibshirani RJ, Rosenfeld R. 2015. Accessed January 21, 2023. https://github.com/cmu-delphi/delphi-epidata.

4. Wang Q, Xie S, Wang Y, Zeng D. Survival-convolution models for predicting COVID-19 cases and assessing effects of mitigation strategies. *Frontiers in Public Health* 2020;8:325.

5. The United States COVID-19 forecast hub dataset. Cramer EY, Huang Y, Wang Y, Ray EL, Cornell M, Bracher J, Brennen A, Castro-Rivadeneira AJ, Gerding A, House K, Jayawardena D, Kanji AH, Khandelwal, A, Le K, Niemi J, Stark A, Shah A, Wattanachit N, Zorn MW, Reich NG. 2021. Accessed January 21, 2023. https://www.medrxiv.org/content.10.1101/2021.11.04.21265886v1.

6. Blackwood JC, Childs LM. An introduction to compartmental modeling for the budding infectious disease modeler. *Letters in Biomathematics* 2018;5:195–221.

7. Smith D, Moore, L. The SIR model for spread of disease—The differential equation model. *Convergence* 2004.

8. Hunter E, Namee B, Kelleher J. An open-data-driven agent-based model to simulate infectious disease outbreaks. *PLoS One* 2018;13(12):e0208775.

9. Truszkowska A, Behring B, Hasanyan J, Zino L, Butail S, Caroppo E, Jiang ZP, Rizzo A, Porfiri M. High-resolution agent-based modeling of COVID-19 spreading in a small town. *Advanced Theory and Simulations* 2021;4(3):2000277.

10. Le M, Ibrahim M, Sagun L, Lacroix T, Nickel M. *Neural relational autoregression for high-resolution COVID-19 forecasting*. Facebook; 2021. Technical Report.

11. Makridakis S. A survey of time series. *International Statistical Review/Revue Internationale de Statistique* 1976;44:29–70.

12. Mojjada RK., Yadav A, Prabhu A, Natarajan Y. Machine learning models for COVID-19 future forecasting. *Materials Today: Proceedings* 2020.

13. Ahmed N, Atiya A, Gayar N, El-Shishiny H. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* 2010;29(5-6):594–621.

14. Hendikawati P. A survey of time series forecasting from stochastic method to soft computing. *Journal of Physics: Conference Series* 2020;16(1):012019.

15. Mahalakshmi G, Sridevi S, Rajaram S. A survey on forecasting of time series data. *International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE '16)*. January 2016;1–8. Piscataway, NJ.

16. Kutner M, Nachtsheim C, Neter J, Li W. *Applied Linear Statistical Models, 5th Edition.* McGraw-Hill/Irwin; 2004.

17. Montgomery DC, Jennings CL, Kulahci M. *Introduction to Time Series Analysis and Forecasting (Wiley Series in Probability and Statistics) 2nd Edition.* Wiley-Interscience; 2015.

18. Vapnik V. *The Nature of Statistical Learning Theory.* Springer; 1995.

19. Holt C. Forecasting trends and seasonal by exponentially weighted averages. *ONR Memorandum* 1957;52(52):5–10.

20. Winters PR. Forecasting sales by exponentially weighted moving averages. *Management Science* 1960;6(3):324–342.

21. Hyndman R, Athanasopoulos G. *Forecasting: Principles and Practice, 2nd Edition.* O Texts; 2018.

22. Box G, Jenkins G. *Time Series Analysis Forecasting and Control.* Wiley, 1970.

23. Dickey DA, Fuller WA. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association* 1979;74(366a):427–431.

24. Phillips PCB, Perron P. Testing for a unit root in time series regression. *Biometrika* 1988;75(2):335–346.

25. Freund Y, Schapire R. Experiments with a new boosting algorithm. *Proceedings of the 13th International Conference on Machine Learning (ICML '96).* July 1996;148–156. Bari, Italy.

26. Dinov, ID. Decision tree divide and conquer classification. In: Dinov ID, ed. *Data Science and Predictive Analytics.* Springer Professional; 2018:307–343.

27. Breiman L. Random forests. *Machine Learning* 2001;45:5-32.

28. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016).* August 2016; 785–794. San Francisco, CA.

29. Wikle CK, Datta A, Hari BV, Boone EL, Sahoo I, Kavila I, Castruccio S, Simmons SJ, Burr WS, Chang W. An overview of model agnostic explainability methods for machine learning applied to environmental data. *Environmetrics* 2023; 34(1):e2772.

30. Harvey, AC. *Forecasting, Structural Time Series Models and the Kalman Filter.* Cambridge University Press, 1989.

31. Durbin J, Koopman SJ. A simple and efficient simulation smoother for state space time series analysis. *Biometrika* 2002;89(3):603–616.

32. Scott SL., Varian H. Predicting the present with Bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation* 2014;5(1-2):4–23.

33. Taylor SJ, Letham B. Forecasting at scale. *The American Statistician* 2018;72(1):37–45.

34. Harvey AC, Shephard N. 10 structural time series models. In: Hrishikesh DV, ed. *Handbook of Statistics 11: Econometrics, 1st Edition.* Elsevier; 1993:261–302.

35. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* 2015;521(7553):436–444.

36. McCulloch WS, Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* 1943;5:115–133.

37. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 1958;65(6):386.

38. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. *Communications of the ACM* 2020;63(11):139–144.

39. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NIPS 2017)* 2017;30.

40. COVID-19 content portal. John Hopkins University Center for System Science Engineering. 2022. Accessed September 6, 2022. https://systems.jhu.edu/research/public-health/ncov/.