Article

COVID-19 Spread Prediction: A Comparative Study

Praveen Damacharla [1,*], Kanika Singhal [2], Ruthwik Reddy Junuthula [3], Colin Elkin [4], and Somesh Kumar[5]

1 KineticAI, Inc., The Woodlands, TX, 77380, USA; Praveen@KineticAI.com (P.D.)
2 Quantta Analytics, Kolkata, West Bengal, 700156, India; kanika.singhal22@gmail.com (K.S.)
3 Miracle Software Systems, Inc., Detroit, MI, 48374, USA; ruthwik.081@gmail.com (R.R.)
4 Xylem, Inc., Morton Grove, IL, 60053, USA; colin.elkin@gmail.com (C.E.)
5 Department of Mathematics, IIT Kharagpur, Kharagpur, West Bengal, 721302, India; smsh@iitkgp.ac.in (S.K.)
* Correspondence: Praveen@KineticAI.com; Tel.:+1-419-450-0357 (P.D.)

Version December 16, 2022 submitted to Mach. Learn. Knowl. Extr.

Abstract: COVID-19 is the latest infectious virus that has become a global pandemic and brought the global economies to their knees. Precise analysis and forecast of the disease spread can help with resource planning and create strategies to slow down the progress of this deadly virus. This paper explores a variety of machine learning models, from heuristic statistical techniques to advanced deep learning methods, to forecast the COVID-19 dynamic. To measure the daily spread of COVID-19, we opt for two target variables: the number of daily positive cases and the number of daily deaths. Although the chance of irregularities and reporting lags is high, it is more sensitive to short-term time series forecasting. These two variables look for stable and reliable estimates for COVID-19 spread. The peculiarity of the data is that it is time series but without one complete period, thereby preventing us from directly using established forecasting methods. Thus, our analysis uses some non-time series methods by including time factors and a few time series methods with the inclusion of exogenous variables by tailoring the data into the appropriate format. We aim to find an optimal model for each family of models where possible. To illustrate the results, India has been chosen for the case study, as this country presently recorded the fastest pace of COVID-19 spread in the first six months of the pandemic. A comparative study has been included with different evaluation metrics. The metrics such as Mean absolute error (MAE), Mean squared error (MSE), Median squared error (MEME), and Mean squared log error (MSLE) has been used for evaluating the spread of COVID-19. We have compared methods such as Liner Regression, Elastic net regularization, Random-forest regressor, XGBoost regressor, Simple exponential smoothing, and so on. Among these methods, the Random-forest regressor shows the highest MAE (11351.8833), MSE(11827.2160), MEME(9998.6333), and MSLE(0.0220) values than the other state-of-the-art methods. Our study indicates that more complex models may not be more reliable compared to simpler ones for forecast COVID-19 spread. We have used python to analyze our results.

Keywords: COVID-19; Machine Learning; Linear Regression; Random Forest; Time Series; XGBoost

1. Introduction

It has been nearly 15 months [5] since the world has noticed the most devastating pandemic of the 21st century – COVID-19. COVID-19 has cost more deaths and misery all over the world than anything else in the past century [2]. Major lifestyle changes have been observed as most of the world was shut down in the first year of the pandemic. For instance, schools were closed and people were stuck in their homes. It led to devastating changes in human civilization in which social interaction was forbidden, the global economy was on its knees, and people lost jobs in nearly every sector. In addition, first responders, medical professionals, and critical workers have been on their toes and continuously active in the war against COVID-19. The latter is true even in the present day when vaccines are readily available in many parts of the world, most economies have reopened, and many lives have returned mostly to normal. Despite currently being in a less intense phase of the pandemic, however, COVID-19 is still adversely affecting the world, through supply chain shortages and delays, worker shortages, and repeated stresses on healthcare systems due to waves caused by new variants of the virus. In addition, many individuals infected by COVID-19 suffer long-term effects of the virus long after the infection is over. For instance, this virus has caused many cardiovascular diseases [7].

The motivation of the other side of this research is the speed prediction in the growth of the COVID-19 virus. Prediction for the daily spread of COVID-19 in future days can be helpful for government and medical staff to be prepared for current and future waves of the pandemic. Furthermore, predictions of new confirmed cases and new deaths can help predict the daily spread of COVID-19. Because these two variables each have their own pros and cons, no one variable can be chosen over the other. The number of reported positive cases can be biased, as it is highly dependent on the number of tests done, which is highly dependent on the number of test centers available in the geographical unit. The cause of death, however, can be something other than COVID-19, i.e. dying of COVID versus DYING *with* COVID. In addition, the mortality rate of COVID-19 is not static, as there is often a lag in reporting both numbers. Despite all these pitfalls, these two variables are still the most ideal for our analysis, as there is a no better indicator to capture COVID-19 exposure [10–15,31]. The combined use of both metrics is a novel focus in this paper, as the prediction of COVID deaths has not been addressed in prior works.

Section 2 provides a literature review of similar work and their results. In Section 3, we discuss the data used to carry out our analysis. Sections 4, 5, and 6 are devoted to predication for daily confirmed cases. In Sections 4 and 5, we explore some predictive and time-series forecasting models respectively. We compare all models built in these sections in Section 6.1. Brief details on the results for the number of daily deaths are given in Section 6.2, and the final conclusions are made in Section 7.

2. Related work

A prediction model is used to analyze future conditions based on the data available. Many predictive modeling methods use statistics to predict events [17,18]. Forecasting always plays an important role in assisting the predictive outcomes of many models to analyze the accuracy of the prediction framework. This is estimated across different study populations, ecosystems, and locations for further improvement of the model [19]. Yang et al. [20] proposed a new method to identify the forecast of the COVID-19 virus using the SEIR and AI model and showed a good quality assessment of 95%. Liang et al., [21] used the Statistical software: LASSO, a logistic regression model to forecast the risk of critical illness of the patient who is affected by COVID-19. An accuracy of 88% was achieved by this method. Yan et al., [22] used the Machine learning tool: XG Boost to relieve the clinical burden and reduce the mortality rate of the people who are affected by COVID-19. Another interesting method proposed by Gong et al., [23] used statistical analysis for predicting the forecast of COVID-19. However, the accuracy achieved was not higher than the other methods. Chatterjee et al. [24] proposed a new method namely SEIR to predict the presence of COVID-19 in the people. Tomar and Gupta [25] used the LSTM method for prediction purposes. Another method that used the LSTM was proposed by Chimmula & Zhang [26]. IHME COVID-19 Health Service Utilization Forecasting Team & Murray [27] analyzed the presence of COVID -19 using the statistical model.

Many machine learning methods were used to predict the forecast of the spread of the COVID-19 virus. Pandey et al. [28] used SEIR and regression models for COVID-19 outbreak predictions. A machine learning forecasting model achieved high accuracy in predicting the outbreak [29]. Deep learning models for the prediction and analysis of COVID-19-positive cases were proposed by Ghosal et al [30]. Another yet interesting method that used LSTM and RNN for predicting and analyzing the COVID-19 positive cases proposed by Arora et al., [31] showed a better performance.

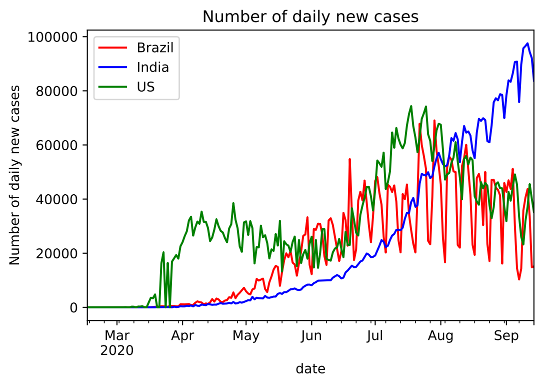
The prediction for the number of confirmed cases has been carried out by many researchers. [1] have proposed a mathematical model to predict the dynamic of COVID-19 for India. In the initial days of COVID-19 spreading, [3] used ARIMA, a wavelet-based forecasting model, and a hybrid implementation of both models. A deep learning model, LSTM, has been explored by [5,6,25] to predict the number of confirmed cases. To the best of our knowledge, no one has tried to predict the number of COVID-19 deaths so far. For predicting the number of daily deaths, number of daily positive cases, number of daily recovered cases, and cumulative number of confirmed cases, [10] used a support vector machine model. Similarly, [11] was able to predict cumulative daily counts of confirmed cases, deaths, and recoveries. A few other surveys have newly been divulged, but they did not cover much observation of many machine learning and deep learning uses.

3. Exploring the data and feature engineering

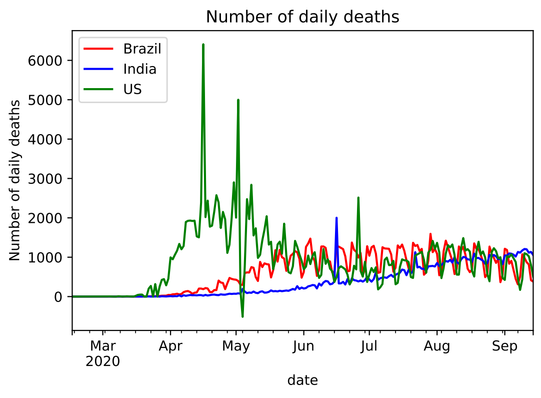
The main objective of this paper is to compare different models for forecasting COVID-19 spread. Thus, we require data from a geographical unit for the case study. We chose the following source for data - https://github.com/GoogleCloudPlatform/covid-19-open-data, as it is available for various countries and at different geographic levels. This source has multiple datasets such as epidemiology (COVID-19 data statistics), demographics, economy, weather, health, mobility, government response data, etc. We use a compiled version of all these datasets in our analysis.

First, we chose the top three most infected countries over the first six months of the pandemic - the US, India, and Brazil - for the analysis. More than 40 million positive cases were reported in each of these countries over such a time period. We have multiple choices to describe the COVID spread such as the number of reported positive cases and the number of deaths. Both numbers have drawbacks, as detailed in the previous section, so to avoid misleading results, we consider both features together.

The raw data is available from January 1, 2020, to the present day. For most of the countries, COVID-19 data was not updated for the initial days. Therefore, we consider data from February 15, 2020, as the starting point for our analysis. While there are presently over two years of data with which to work thereafter, we will only focus on data going up to September 1, 2020. This is due to the focus of this paper being on comparing machine learning algorithms in overall effectiveness in predicting the spread of COVID-19 rather than predicting present spread levels. Since we are trying to compare algorithms with the use of ground truth data, it is ideal to narrow the focus of the overall timeline to the first few months of the pandemic rather than every stage and wave encountered thus far.



(a) positive cases



(b) deaths

Figure 1. Time series plots for COVID-19 spread data

Figure **??** depicts plots of the number of positive cases and the number of deaths reported daily for all three countries. Over the first few months of the pandemic, the US was the most infected country, although the trend eventually gravitated towards Brazil. Towards the end of the timeline, there was exponential growth in India. Since India seemed to be the worst country in terms of daily COVID-19 cases at the end of the timeline, we have chosen India for the case study to compare various prediction techniques.

After filtering the data for India at the country level, we pre-processed the data for modeling. For the remainder of this paper, the word *data* stands for Indian data at the country level. There are a few drawbacks and limitations to the raw data. For instance, the population and related variables have static data over time across all the rows. Since we have exponential and substantial growth in the death poll, we should not consider the given demographic data as is. Thus, we distribute total deceased counts uniformly across gender and ten buckets of age data. This helps us to update the daily population and related variables in a meaningful way.

One of the most significant features we have available is mobility data. This tells us about the change in footfall and visitation patterns of consumers at different locations such as stores, parks, restaurants, cafes, workplaces, and homes.

Since we should evaluate the model performance on unseen data, we implement a supervised learning process by dividing our data into training and test data. For each class of models, we adopted the same train and test data for the sake of comparison of forecast values. We use the time frame leading up to September 1, 2020, as the training partition and the time frame thereafter as the test partition.

In the following sections, we showcase the usage of selected statistical and machine-learning models to predict the spread of COVID-19.

4. Non-time-series predictive models to predict number of daily positive cases

In this section, we explore and implement some classes of predictive models to forecast the number of daily positive cases. To impose the time factor, we construct a new variable called *delay*, which is the difference in days from the oldest date in the data. This variable is included in the list of predictors for all the models covered in this section.

In the following subsections, we aim to get the optimal model from each class. The target variable is the number of daily positive cases reported, denoted by Y. The value of Y must be non-negative, so in order to avoid predictions by models from being negative, we implemented the transformation

$$Y \to log(1+Y) \tag{1}$$

for the target variable.

Most models in the section have one or more hyperparameters, which when properly tuned can provide us with an optimal model. Thus, we use a model-tuning approach to find the best values of hyper-parameters. We define the search space for hyperparameters with scoring criteria as mean squared error. Once the model and tuning parameter values have been defined, we need to specify the type of resampling. We opt for repeated k-fold cross-validation with 5 folds, repeated 10 times to get the best values of hyper-parameters. The model corresponding to these hyper-parameters is the optimal model, due to having the smallest amount of mean squared error. Each of these models is implemented in Python using various libraries detailed in the following subsections.

4.1. Linear regression

Linear regression [32] can be used to find the linear relationship between a target variable and one or more independent variables. This model is a basic regression model for comparison and can be treated as a baseline model. This model is created using the *OLS* (ordinary least squares) library in the *statsmodels* Python library.

The standard regression model is represented in the equation.4.1 :

$$y_t = x_t^{'} \beta u_t (t = 1, 2, ...T) \tag{2}$$

Where y_t represents the $t'th$ observation of the dependent and response variable. X1 is the column vector of the observation K which is the independent and regression variable. The index t is the time series data. β is the $Kx1$ vector to be estimated and u_t is the stochastic term.

The first regression model is built by using all predictors. The importance of predictors is given in Figure 2.

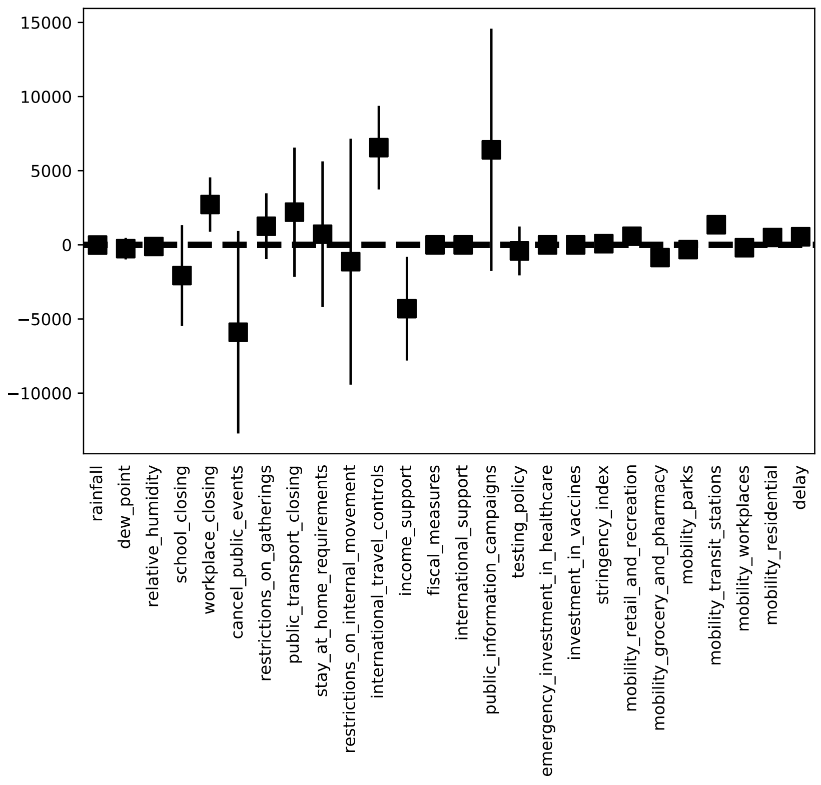


Figure 2. Coefficients of regression equations with 95% confidence interval

Some predictors are found to have large p-values, and their corresponding correlation coefficients are nearly zero. Such predictors are not significant. We choose the level of significance $\alpha = 0.05$ and skip the predictors with p-values greater than α. Table 1 shows the values of R^2 and adjusted R^2 for both regression models: one with all predictors and one with only significant predictors. Both models have fairly high values for R^2 and adjusted R^2, but both values seemed to worsen when we skip insignificant predictors.

	R^2	Adjusted R^2
Model with all predictors	0.989	0.987
Model with significant predictors	0.986	0.985

Table 1. R^2 and adjusted R^2 values for different linear regression models

Figure 3 compares the results of both models against the actual values. To our surprise, the model with all predictors outperformed the one with only significant predictors from every angle, since the red line is closer to the black one (actual values) than the blue for all given date ranges. Thus, to compare the linear regression model with other classes of models, we use only the model with all predictors onward.

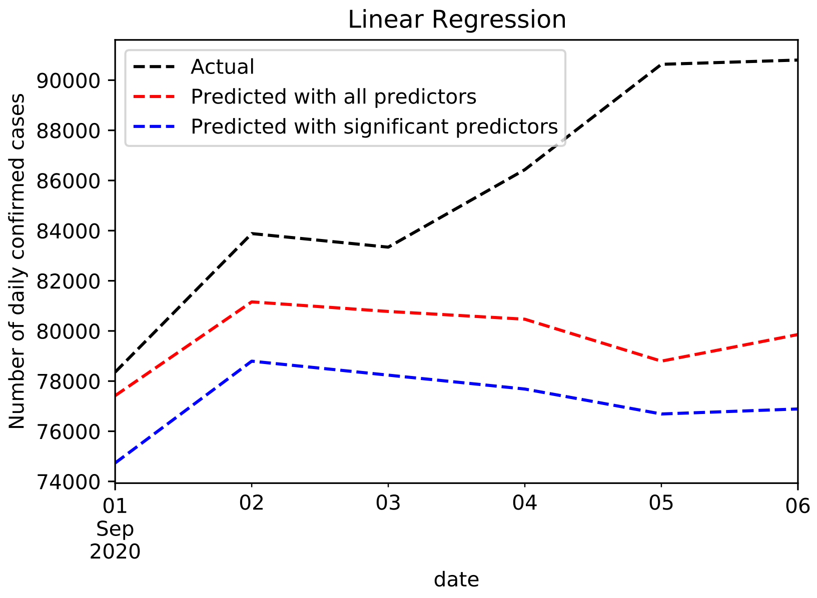


Figure 3. Comparison of actual daily case counts with predicted counts from two regression models: one with all predictors and one with significant predictors

4.2. Elastic net regularization

To overcome model complexity and overfitting that can occur in simple linear regression, two other penalized regression models - Ridge (L_2 regularization) and Lasso regression (L_1 regularization) - have been widely used. The overfitting occurs due to the large model parameters. The elastic net regularization is used as same as the ridge or Lasso. If the mixing parameter is zero, then we can use ridge regression. If the mixing parameter is one, then we can use the lasso regression [33].

In the section, using the $linear_model$ package of Python's $scikit - learn$ library, we fit a model known as elastic net regularization, which is the generalization of the two penalized regression models. This class of models has two hyper-parameters:

- α : mixing parameter, which controls the type of regression
- λ : shrinkage parameter which is the amount of the shrinkage.

The search space is chosen as

$$\begin{aligned}\alpha &\in \{0.1, 0.2, \ldots, 1\},\\ \lambda &\in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}, 1, 10^{1}, 10^{2}\}.\end{aligned}$$

After hyperparameter tuning, the optimal values turned out to be $\alpha = 0.2$ and $\lambda = 0.1$. Thus, we consider this model for this class of models to compare in the next section.

4.3. Random forest regressor

Random forest [34] is a supervised machine learning algorithm used for classification and regression. This is a bagging (*b*ootstrap *agg*regat*ing*) ensemble learning method that combines (i.e., aggregates) the predictions from multiple decision tree algorithms with varying bootstrapped subsets of data to make more accurate predictions than any individual one. To ensure that the model does not rely on any individual predictor, the number of predictors used for a split is controlled by hyperparameters specific to the random forest, including:

- n_estimators = number of trees in the forest,
- max_features = number of maximum features to consider at every split,
- max_depth = maximum number of levels in the tree,

- min_samples_split = minimum number of samples required to split a node,
- min_samples_leaf = minimum number of samples required at each leaf node, and
- bootstrap = method of selecting samples for training each tree.

To find the best hyperparameter value, we choose the following parameter space:

$$\begin{aligned}
\texttt{n\_estimators} &\in \{50, 100, 200, 500, 1000\} \\
\texttt{max\_features} &\in \{'auto',' sqrt'\} \\
\texttt{max\_depth} &\in \{5, 20, 50, 100\} \\
\texttt{min\_samples\_split} &\in \{2, 5, 10\} \\
\texttt{min\_samples\_leaf} &\in \{1, 2, 4\}.
\end{aligned}$$

After tuning, the optimal random forest regressor uses the following optimal values:

$$\begin{aligned}
\texttt{n\_estimators} &= 200 \\
\texttt{max\_features} &= 'auto' \\
\texttt{max\_depth} &= 50 \\
\texttt{min\_samples\_split} &= 2 \\
\texttt{min\_samples\_leaf} &= 5.
\end{aligned}$$

We consider this model from this class of models for comparison in Section 6.

4.4. XGBoost regressor

The XGBoost [35] is a widely used supervised machine learning model that is an implementation of the gradient boosting decision tree algorithm. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains a loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e how far the model results are from the real values. The most common loss function in XGBoost for regression problems is reg:linear, and that for binary classification is reg:logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancels out and better one sums up to form final good predictions. This algorithm has the following hyperparameters:

- n_estimators = number of gradients boosted trees,
- objective = a learning objective function corresponding to the learning task,
- learning_rate = step size shrinkage for tree booster,
- max_depth = maximum tree depth for base learners,
- min_child_weight = minimum sum of instance weight (hessian) needed in a child,
- min_samples_leaf = minimum number of samples required at each leaf node, and
- bootstrap = method of selecting samples for training each tree.

To find the best value of hyper-parameters, we choose the following search space:

$$\begin{aligned}
\texttt{n\_estimators} &\in \{50, 100, 200, 500, 1000\} \\
\texttt{objective} &\in \{'reg: squarederror',' reg: squaredlogerror'\} \\
\texttt{learning\_rate} &\in \{0.2, 0.5, 0.8\} \\
\texttt{max\_depth} &\in \{5, 20, 50, 100\} \\
\texttt{min\_child\_weight} &\in \{3, 4, 5\} \\
\texttt{silent} &\in \{0, 1\} \\
\texttt{subsample} &\in \{0.2, 0.7\} \\
\texttt{colsample \_bytree} &\in \{0.2, 0.7\}.
\end{aligned}$$

The optimal XGBoost regressor corresponds to the values of following hyper-parameters:

$$\begin{aligned}
\texttt{n\_estimators} &= 50 \\
\texttt{objective} &= 'reg: squarederror' \\
\texttt{learning\_rate} &= 0.5 \\
\texttt{max\_depth} &= 5 \\
\texttt{min\_child\_weight} &= 5 \\
\texttt{silent} &= 0 \\
\texttt{subsample} &= 0.7 \\
\texttt{colsample \_bytree} &= 0.7.
\end{aligned}$$

We consider this model for comparison in Section 6 using the *xgboost* Python library.

4.5. Recurrent neural network (RNN)

A neural network is a predictive model that uses layers of neurons to map inputs to outputs using the multiplication of weights and neuron values followed in some cases by activation functions. The weights are optimized using backpropagation. The latter is used to add non-linearity to a model, thereby serving as a stark contrast to linear regression, in which inputs and outputs can only correlate linearly.

A typical neural network has input, output, and hidden layers. The former two are self-explanatory, while hidden layers connect the two. A recurrent neural network is a variation of this that involves time. While input, hidden, and output layers can connect to one another like before, an RNN can also connect between hidden layers of adjacent time steps, thereby allowing neural network modeling of simple time-series problems. However, in our study, RNNs [36] are fairly limited in that a particular point in time only has a connection to adjacent time steps, and thus the information for one particular data can only be directly influenced by the most immediate previous day.

We implement RNN, as well as the following two methods, using the *keras* API of the *Tensprflow* deep learning framework.

4.6. Long short-term memory network (LSTM)

The long short-term memory (LSTM) [37] network is an advanced deep learning method based on RNN to forecast time-series data. Instead of neurons, LSTM networks have memory blocks that are connected through layers. A block has components that make it smarter than a classical neuron and a memory for recent sequences. A block contains gates that manage the block's state and output. A block operates upon an input sequence and each gate within a block uses sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block conditional.

Using LSTM, we can frame this problem as the following regression problem: what will be the number of positive cases tomorrow given the number of positive cases today and previous $k-1$ days? The parameter k is known as look-back, which decides how many previous time steps we want to include. For simplicity, we choose $k=1$. Therefore, we must convert our univariate data into bivariate, where the first variable indicates the number of the present day's positive cases and the second variable stands is the number of positive cases predicted on the next day. Since this method is sensitive to the scale of data, we, therefore, normalize the data to lie between 0 to 1. To build this model, we use the default settings.

4.7. *Gated recurrent unit (GRU)*

A

5. Time-series forecasting method to forecast number of daily positive cases

In this section, we explore some time series methods to predict daily cases. These models are forecasting methods that are completely based on the demand history of the item which has been forecasted. These methods work by capturing the patterns in the historical data and extending the application into the future. They are appropriate when you can assume a reasonable amount of continuity between the past and the future. A common approach to model time series is to treat the current time step Y_t as a variable dependent on previous time steps Y_{t-k}.

5.1. *Exponential smoothing*

Exponential smoothing [38] is a powerful time series forecasting method for univariate data. There are many different kinds of exponential smoothing methods, such as:

- Simple exponential smoothing,
- Double exponential smoothing (Holt method),
- Triple exponential Smoothing (Holt-Winters method).

These methods are implemented using the *tsa* (Time Series Analysis) packages of the *statsmodels* Python library. Each of these methods is explored further in the following subsections.

5.1.1. Simple exponential smoothing

As the name suggests, simple exponential smoothing is the simplest method. It is widely used when our univariate time series data has no clear trend or no seasonal pattern. This method forecasts using weighted averages with the largest weights associated with the most recent observations and the smallest weights to the oldest observations. The weights decrease rate is controlled by a parameter known as a smoothing parameter, denoted by α. The value of α lies between 0 to 1, where a larger value requires the model to pay close attention to the most recent past observations. The extreme cases are:

- $\alpha = 0$: Becomes an average since all weights are equal and the next predicted value is equal to the average of historical data,
- $\alpha = 1$: Becomes a naive method since a weight's most recent observation is one and all others are zero. Thus, the next predicted value is the same as the recent observation.

5.1.2. Double exponential smoothing (Holt method)

This is an extension of simple exponential smoothing. Double exponential smoothing was proposed by Holt in 1957. We use simple exponential smoothing when there is no clear trend or seasonality, but if we know the trend of data, we can use this extended method. Holt's method involves the following two parameters:

- α = smoothing parameter,
- β = trend smoothing parameter.

Both parameters take values between 0 to 1. There is also an option to choose a trend type. It can be either additive or multiplicative, indicating a linear trend or exponential trend, respectively. In Section 5, we found the admissible value for smoothing parameter α. Thus, we consider the fixed value of $\alpha = 0.8$ and then determine the optimal trend type with fixed values of α and β.

5.1.3. Triple exponential Smoothing (Holt-Winters method)

This is the most advanced exponential smoothing method, as it is ideal for data with clear trends and seasonality. It has the power to add support for seasonality in a model. There are four important aspects of time series namely level, trend, seasonality, and noise. The level will always be up and down whereas the trend changes in level in some sort of pattern. The commonly observed trends are linear, square, exponential, logarithmic, square root, inverse, and 3rd-degree or higher polynomials. Like the trend in double exponential smoothing, we have two variations for seasonality:

- Additive method: the seasonal variations are constant,
- Multiplicative method: the seasonal variations changes with time.

5.2. Auto Regressive Integrated Moving Average (ARIMA)

Auto-Regressive Integrated Moving Average (ARIMA) model [39] is one of the most widely used families of models for time series. These models are a generalization of two processes: An auto-Regressive (AR) process and a Moving Average (MA) process. Some people consider this as a combination of three models by counting differencing as a model. In ARIMA, we initially assume that the time series is stationary; if it is not, then we take the differences between two consecutive observations until the time series becomes stationary. An ARIMA model is classified by three following parameters:

- p : number of autoregressive terms,
- d : number of nonseasonal differences needed to make time series stationary,
- q : number of lagged forecast errors in the prediction equation.

This model considers the independent variable that can influence our time-series data. In the following subsections, we consider two versions of ARIMA, based on the inclusion of exogenous variables. Both versions are implemented using the *pmdarima* package in Python.

5.2.1. ARIMA without exogenous variables

Here, we build an ARIMA model with the count of daily positive cases as the only training data. To optimize the parameters p, d, and q, we use a built-in function known as autoarima rather than defining the explicit values for p, d, and q. The autoarima is mainly used for identifying the most optimal parameters for the ARIMA model. It settles on a single-fitted ARIMA model. This method is completely based on the commonly used R function.

5.2.2. ARIMA with exogenous variables

As exogenous variables, we use all the independent variables used in Section 4 except for *delay* variables. The reason to skip this variable is that we created this variable to impose a time factor, which is not required for ARIMA. Autoarima is used here as well.

5.2.3. Seasonal ARIMA

Seasonal ARIMA (SARIMA) is an ARIMA model in which

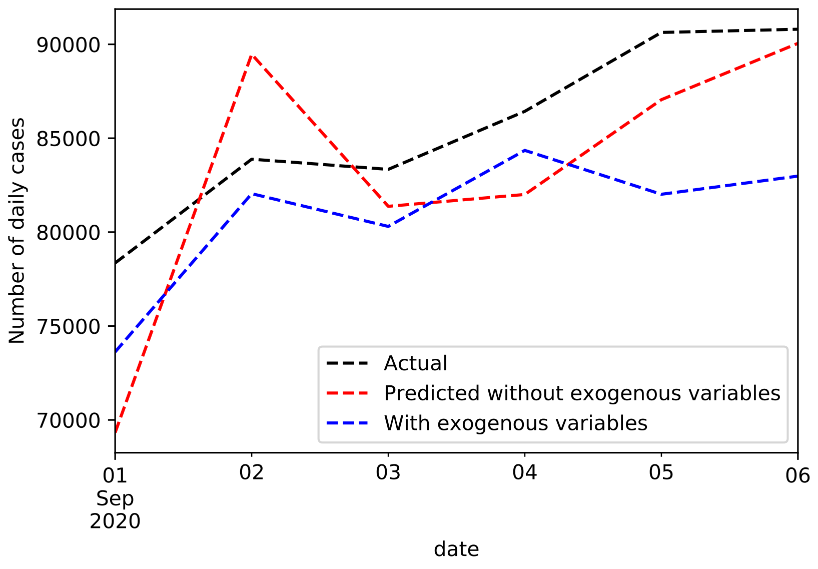


Figure 4. Comparison of SARIMA models

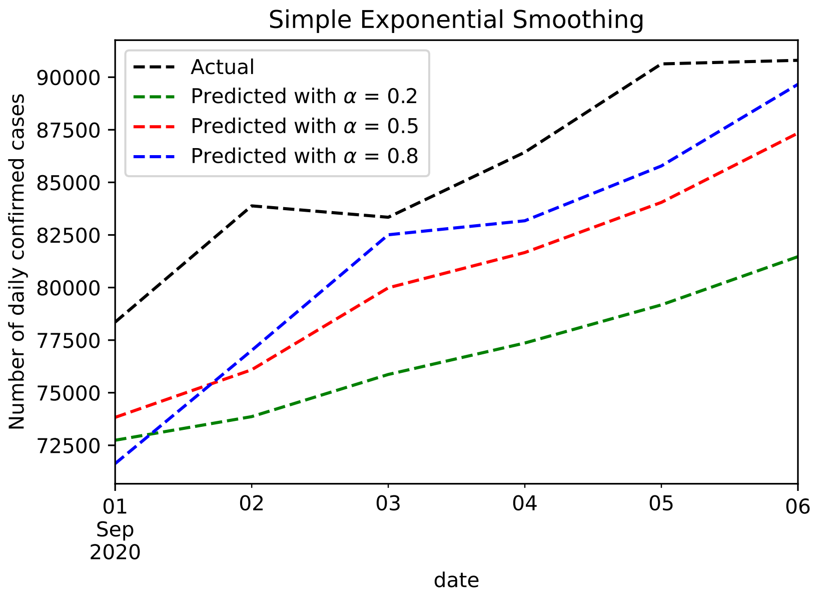


Figure 5. Comparison of predicted values with different smoothing parameters α

6. Results and analysis

In this section, we review the models with the following metrics for evaluating predictions and also the analysis for each method

- Mean absolute error (MAE): average of the absolute differences between predicted and actual values. It is used when we care only about the magnitude of the error and not the direction.
- Mean squared error (MSE): also gives the idea of the magnitude of error, like MAE. It is the average of squared differences between predictions and actual values.
- Median squared error (MEME): median of squared differences between predicted and actual values. Since the mean is not robust. The mean is much more sensitive to extreme values than the median. Therefore we consider MEME as an alternative evaluation metric.
- Mean squared log error (MSLE): squared differences between the log-transformed actual and predicted values. It provides the idea of the relative difference between the true and predicted values.

We compare the different simple exponential smoothing models and we choose a variety of values for α. The resultant predicted values are given in Figure 5.

For most of the dates, predicted values from the model with $\alpha = 0.8$ are the closest to actual values. Therefore from this family, we choose the simple exponential smoothing model with $\alpha = 0.8$ to compare it with other classes of models.

The double exponential smoothing method is implemented as shown in Figure 6. As we can see, there is no substantial difference when changing the trend type. So, we select additive trend type and plot for different values for β in Figure 7.

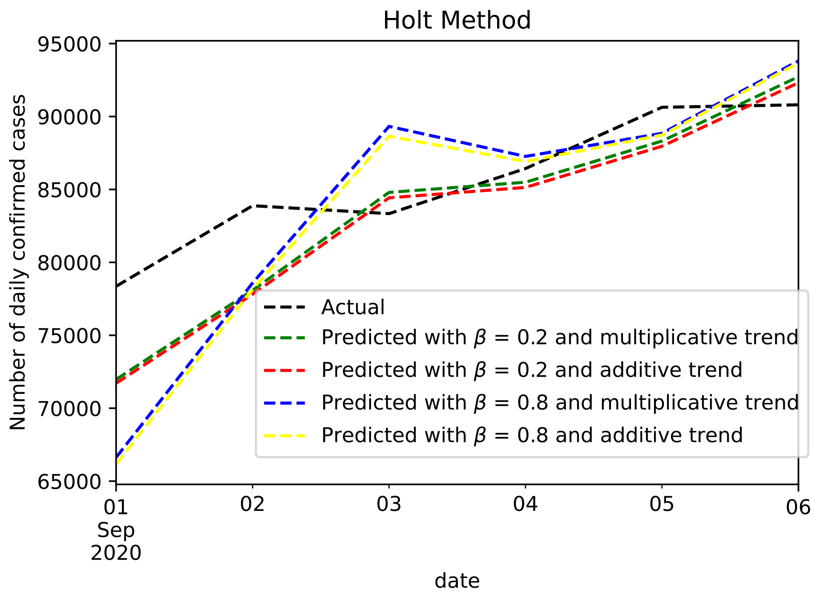


Figure 6. Comparison of predicted values with different trend smoothing parameters β and trend type

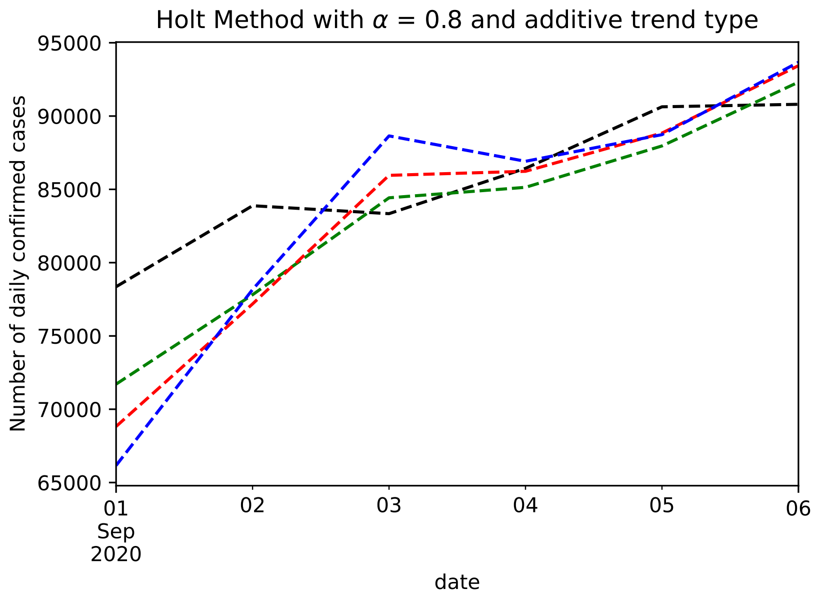


Figure 7. Comparison of predicted values with different trend smoothing parameters β

As indicated in the figure, there is no admissible choice for β. Therefore, we will consider all three methods with $\beta = 0.2, 0.5$, and 0.8 in Section 6.

The predicted values of the triple exponential smoothing method is plotted in Figure 8 for a different type of trend and seasonality.

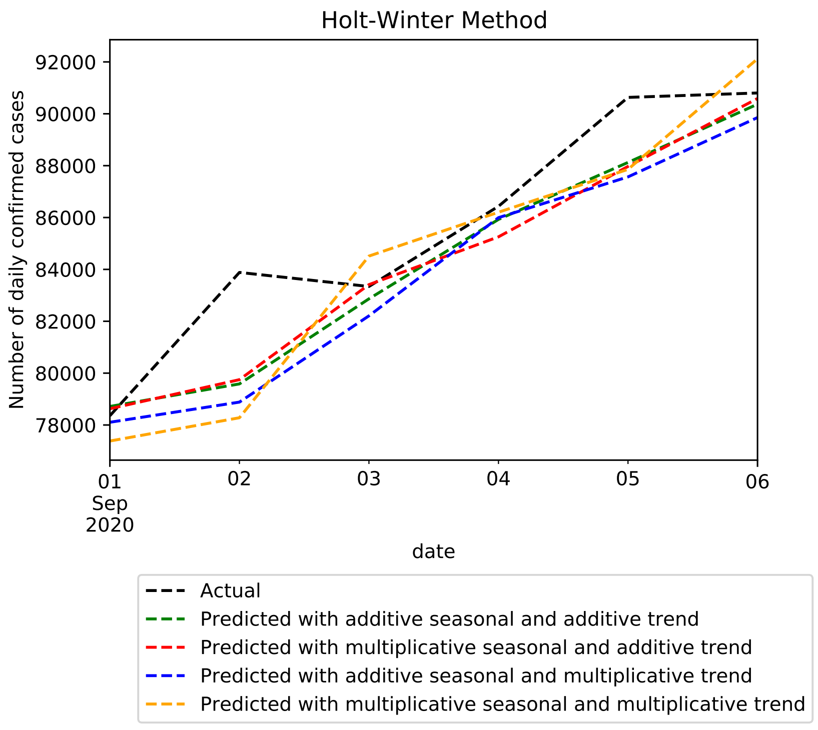


Figure 8. Comparison of predicted values with a different type of trend and seasonality

As the figure indicates, the Holt-Winters method with the additive trend and additive seasonality is found to be the best.

In Figure 9, we compare both ARIMA models, one without exogenous and one with, against ground truth values.

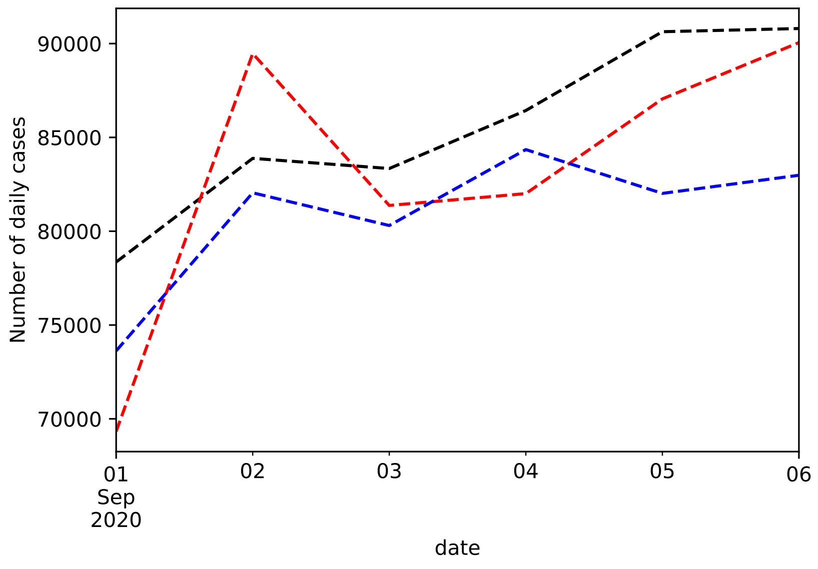


Figure 9. Comparison of ARIMA models

Model	Variables used for training	Configuration	Evaluation
Linear regression	Models with predictors and without predictors	$\alpha - 0.05$	Actual values with the predicted values
Elastic net regulation	Ridge Lasso	α = 0.2 $\gamma = 0.1$	Mean absolute error (MAE), Mean squared error (MSE), Median squared error (MEME), Mean squared log error (MSLE)
Random forest regressor	$n - estimators$, $max_features$, max_depth, $min_samples_split$, $min_samples_leaf$	$n_estimators$=200, $max_features = auto$, $max_depth = 50$, $min_samples_split = 2$, $min_samples_leaf$ = 5	Mean absolute error (MAE), Mean squared error (MSE), Median squared error $(MEME)$, Mean squared log error $(MSLE)$
Recurrent neural network (RNN)	uses layers of neurons to map inputs to outputs	Keras API	Mean absolute error (MAE), Mean squared error (MSE), Median squared error (MEME), Mean squared log error (MSLE)
Long short-term memory network (LSTM)	RNN	k = 1 and Normalize data = 0 and 1	Mean absolute error (MAE), Mean squared error (MSE), Median squared error (MEME), Mean squared log error (MSLE)
Gated recurrent unit (GRU)			

Table 2. Analysis of non-time-series predictive models to predict the number of daily positive cases

As indicated in the figure, there is no admissible choice between these two ARIMAs. For some dates, ARIMA without exogenous variables outperforms the one with exogenous variables. Therefore we will consider both models for comparison in Section 6.

6.1. A comparative study of models to predict the number of daily positive cases

In Sections 4 and 5, we have explored many methods to predict the number of daily positive cases. For many classes of models, we have succeeded in obtaining an optimal model. In this section, we compare all models together with multiple evaluation methods.

First, we compared two linear regression models and opted for the model with all predictors. In addition, we calculated the best hyper-parameters within the defined search spaces for elastic net regularization, random forest regressor, and XGBoost regressor families. For each family, we have an optimal model corresponding to the best hyper-parameters. We have also built an LSTM model, forming a total of five models from Section 4. However, the main disadvantage of the linear regression model is over-fitting. The elastic net regularization can cause a small bias in the model where the prediction is too dependent upon a particular variable. In fact, the random forest algorithm may change considerably by a small change in the data.

In Section 5, we explored some time-series forecasting methods. For the simple exponential smoothing method, we have chosen the model with smoothing parameter $\alpha = 0.8$. For the Holt method, we did not obtain anyone's admissible method. Thus, we decided to have three models with smoothing parameter $\alpha = 0.8$, additive trend type, and corresponding to the trend's smoothing parameters $\beta = 0.2, 0.5$, and 0.8. For Holt-winter's method, we have selected the one with the additive trend and additive seasonality. For ARIMA family, we have two models with and without exogenous variables. Thus, we have seven models from Section 5.

Model	Variables used for training	Configuration	Evaluation
Exponential smoothing	Simple exponential smoothing, Double exponential smoothing (Holt method), Triple exponential Smoothing (Holt-Winters method)	TSA (Time Series Analysis)	MAE, MSE, MESE, MSLE
Auto-Regressive Integrated Moving Average	p : number of autoregressive terms, d : number of nonseasonal differences needed to make time series stationary, q : number of lagged forecast errors in the prediction equation	ARIMA with and without exogenous variables. Seasonal ARIMA	MAE, MSE, MESE, MSLE

Table 3. Analysis of time-series predictive models to predict the number of daily positive cases

Model	MAE	MSE	MESE	MSLE
Linear regression	4804.8860	6172.9314	2723.2462	0.0054
Elastic net regularization	7265.5959	8245.1422	5342.1506	0.0100
Random forest regressor	11351.8833	11827.2160	9998.6333	0.0220
XGBoost regressor	10130.6125	10566.9168	9346.6719	0.0173
Simple exponential smoothing	4507.6726	5045.6480	4851.4896	0.0040
Holt with $\beta = 0.2$	3552.8030	4266.5536	2670.3701	0.0030
Holt with $\beta = 0.5$	4168.4262	5401.2516	2615.0862	0.0050
Holt with $\beta = 0.8$	5120.1373	6533.2962	5305.5930	0.0076
Holt-winters	1629.8258	2253.0399	506.1216	0.0007
Arima	4918.0511	5459.2333	4427.1078	0.0048
Arima with exogenous variables	4061.0362	4766.3267	3037.7827	0.0033
Sarima	4918.0511	5459.2333	4427.1078	0.0048
RNN	7604.9391	7895.1482	8395.9531	0.0098
GRU	4490.1203	5020.4703	5372.7188	0.0039
LSTM	6238.7969	6588.8430	7022.9141	0.0067

Table 4. Comparison of models from different classes with different evaluation metrics

379

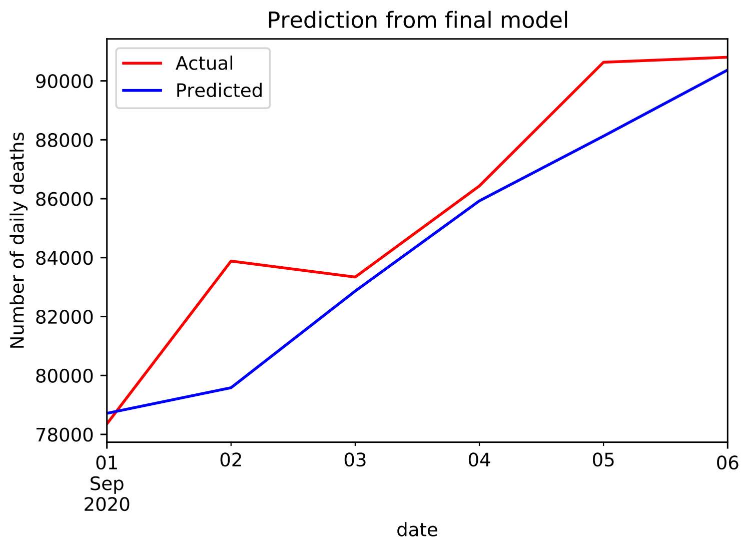


Figure 10. Prediction of daily confirmed cases for first six days of September 2020

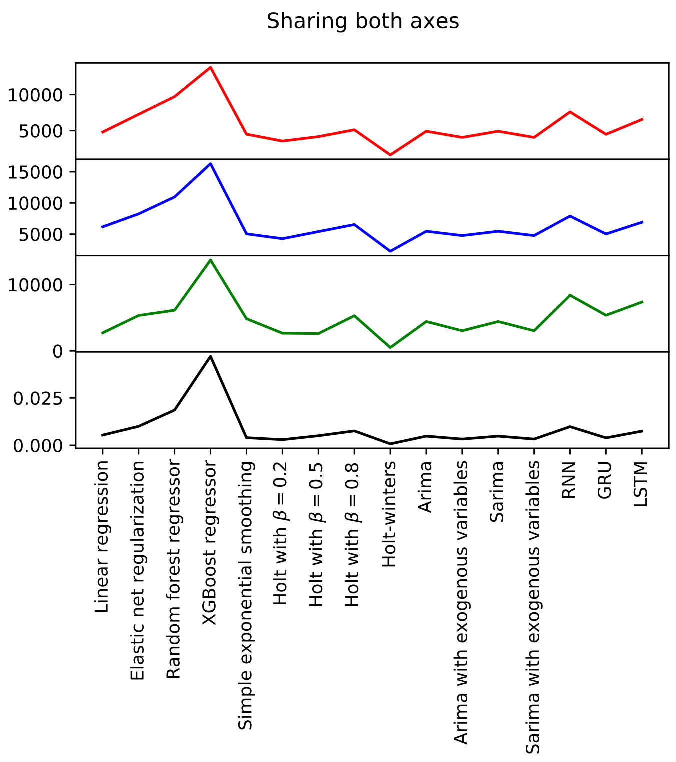


Figure 11. Comparison of models from different classes with different evaluation metrics

380 ## 6.2. Predicted number of daily deaths

381 In this section, we predict daily deaths on the same line using the methods from previous sections. 382 We provide the final results in the following table and graphs. There are different methods to handle 383 the computational cost and missing data. In these models such as XGBoost, and Random-forest, the 384 missing values are interpreted as data that contain information (ie, data that are missing for a reason) 385 instead of data that are missing at random.

Model	MAE	MSE	MESE	MSLE
predicted_lm1	75.9125	84.5736	87.1517	0.0067
predicted_el1	47.9260	55.1785	51.6010	0.0028
predicted_rf1	136.8500	152.6284	167.7500	0.0244
predicted_xgb1	196.8140	230.1556	222.5404	0.0661
predicted_ses_0.8	64.3146	93.7673	37.1779	0.0096
predicted_holt_0.2	73.1619	104.6138	44.9703	0.0123
predicted_holt_0.5	81.5152	125.5030	53.2248	0.0187
predicted_holt_0.8	94.6749	138.4354	49.2252	0.0234
predicted_hw1	29.8014	39.5107	18.9877	0.0014
predicted_autoarima	51.8795	55.8371	57.1838	0.0029
predicted_autoarima_ex	69.7804	75.0977	70.4612	0.0053
predicted_lstm	189.0365	191.7943	184.3370	0.0397

Table 5. Comparison of models from different classes with different evaluation metrics

386

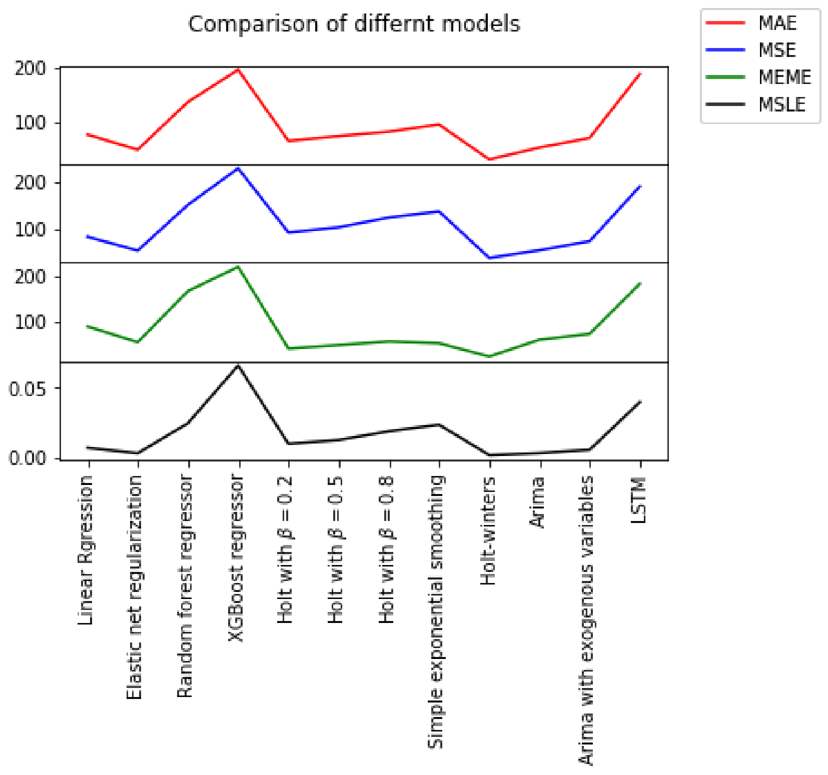


Figure 12. Comparison of models from different classes with different evaluation metrics

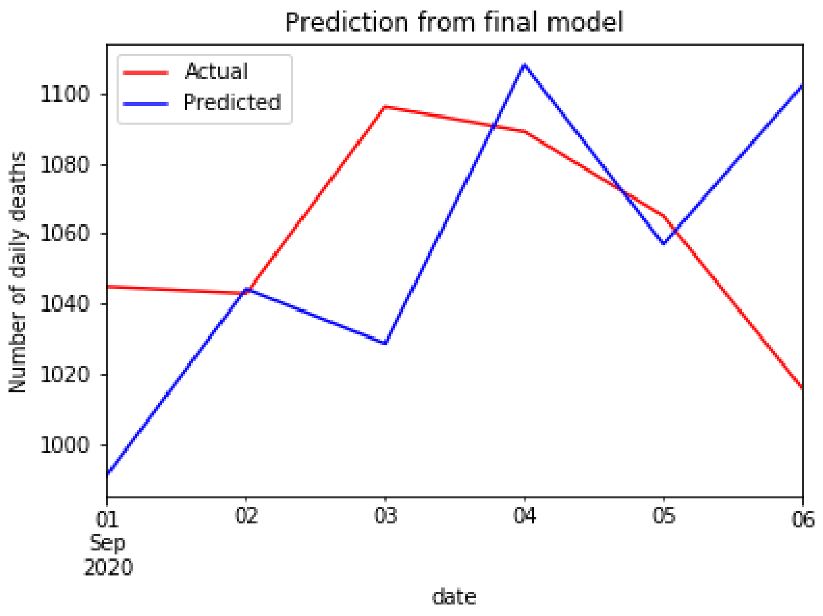


Figure 13. Prediction of daily deaths for first six days of September 2020

7. Conclusions

In this paper, we aimed to forecast COVID-19 spread by predicting the number of daily positive cases and daily deaths. For the case study, we considered the data of India at the national level. Any country with any geographic level can be analyzed in the same manner according to the availability of data. We used many different models to predict the number of daily positive cases and the number of daily deaths. From there, we compared with respect to different evaluation metrics. Our study indicates that more complex models do not outperform simpler ones for COVID-19 spread. Regardless, we have observed that the Holt-winters model is an optimal model for both the number of daily positive cases and the number of daily deaths.

References

1. Sarkar, K.; Subhas K.; Juan J. N. Modeling and forecasting the COVID-19 pandemic in India. *Chaos, Solitons & Fractals*, **2020**, 139, 110049.
2. Countries where coronavirus has spread, May 2020, [online] Available: https://www.worldometers.info/coronavirus/countries-where-coronavirus-has-spread/. **2021**
3. Chakraborty, T.; Ghosh, I. Real-time forecasts and risk assessment of novel coronavirus (COVID-19) cases: A data-driven analysis. *Chaos, Solitons & Fractals*, **2020**, 109850.
25. Tomar, A.; Gupta, N. Prediction for the spread of COVID-19 in India and effectiveness of preventive measures. *Science of The Total Environment*, **2020**, 138762.
5. Cucinotta, Domenico, and Maurizio Vanelli. "WHO declares COVID-19 a pandemic." Acta bio-medica: Atenei Parmensis 91.1 **2020**: 157-160.
6. P. Dhakal, P. Damacharla, A. Y. Javaid, H. K. Vege, and V. K. Devabhaktuni, "IVACS: Intelligent Voice Assistant for Coronavirus Disease (COVID-19) Self-Assessment," *2020 International Conference on Artificial Intelligence & Modern Assistive Technology (ICAIMAT)*, **2020**, pp. 1-6, doi: 10.1109/ICAIMAT51101.2020.9308013.
7. Zheng, Y.Y.; Ma, Y.T.; Zhang, J.Y.; Xie, X. COVID-19 and the cardiovascular system. *Nature Reviews Cardiology 17(5)*, **2020**, 259-260.
8. Wieczorek, Michał, Jakub Siłka, and Marcin Woźniak. "Neural network powered COVID-19 spread forecasting model." *Chaos, Solitons & Fractals* 140 **2020**: 110203.
9. Satu, Md, Koushik Chandra Howlader, Mufti Mahmud, M. Shamim Kaiser, Sheikh Mohammad Shariful Islam, Julian MW Quinn, Salem A. Alyami, and Mohammad Ali Moni. "Short-term prediction of COVID-19 cases using machine learning models." *Applied Sciences* 11, no. 9 **2020**: 4266.
10. Parbat, D.; Chakraborty, M. A python based support vector regression model for prediction of COVID-19 cases in India. *Chaos, Solitons & Fractals, 138*, **2020**, 109942.
11. Petropoulos, F.; Makridakis, S., Forecasting the novel coronavirus COVID-19. *PloS one, 15(3)*, **2020**, p.e0231236.
12. Zhao, Zebin, Xin Li, Feng Liu, Gaofeng Zhu, Chunfeng Ma, and Liangxu Wang. "Prediction of the COVID-19 spread in African countries and implications for prevention and control: A case study in South Africa, Egypt, Algeria, Nigeria, Senegal and Kenya." *Science of the Total Environment* 729 **2020**: 138959.
13. Sanchez-Caballero, Samuel, Miguel A. Selles, Miguel A. Peydro, and Elena Perez-Bernabeu. "An efficient COVID-19 prediction model validated with the cases of China, Italy and Spain: Total or partial lockdowns?." *Journal of clinical medicine* 9, no. 5 **2020**: 1547.
14. Ardabili, Sina F., Amir Mosavi, Pedram Ghamisi, Filip Ferdinand, Annamaria R. Varkonyi-Koczy, Uwe Reuter, Timon Rabczuk, and Peter M. Atkinson. "Covid-19 outbreak prediction with machine learning." *Algorithms* 13, no. 10 **2020**: 249.
15. Qin, Lei, Qiang Sun, Yidan Wang, Ke-Fei Wu, Mingchih Chen, Ben-Chang Shia, and Szu-Yuan Wu. "Prediction of number of cases of 2019 novel coronavirus (COVID-19) using social media search index." *International journal of environmental research and public health* 17, no. 7 **2020**: 2365.
31. Arora, Parul, Himanshu Kumar, and Bijaya Ketan Panigrahi. "Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India." *Chaos, Solitons & Fractals* 139 **2020**: 110017.
17. Geisser, Seymour. "Predictive inference" *Chapman and Hall/CRC*
18. Damacharla, P.; Javaid, A.Y.; Devabhaktuni, V.K. Human error prediction using eye tracking to improvise team cohesion in human–machine teams. In Proceedings of *the International Conference on Applied Human Factors and Ergonomics (AHFE 2018)*, Orlando, FL, USA, 22–26 July **2018**; pp. 47–57.
19. Allotey J, Snell KI, Chan C. "External validation, update and development of prediction models for pre-eclampsia using an Individual Participant Data (IPD) meta-analysis" *he International Prediction of Pregnancy Complication Network (IPPIC pre-eclampsia) protocol. Diagn Progn Res* **2017**: 1-16

20. Yang Z, Zeng Z, Wang K, et al. "Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions." *J Thorac Dis* no. 12 **2020**:165-74.
21. Liang W, Liang H, Ou L, et al."Development and validation of a clinical risk score to predict the occurrence of critical illness in hospitalized patients with COVID-19." *JAMA Intern Med* no.180 **2020**:1081-9.
22. Yan L, Zhang HT, Goncalves J, et al. "An interpretable mortality prediction model for COVID-19 patients." *Nat Mach Intell* no.2 **2020**:238-8.
23. Gong J, Ou J, Qiu X, et al. "A tool for early prediction of severe coronavirus disease 2019 (COVID-19): a multicenter study using the risk nomogram in Wuhan and Guangdong, China." *Clin Infect Dis* no. 71 **2020**:833-40.
24. Chatterjee K, Chatterjee K, Kumar A, et al."Healthcare impact of COVID-19 epidemic in India: a stochastic mathematical model." *Med J Armed Forces India*. no. 76 **2020**:147–55.
25. Tomar A, Gupta N."Prediction for the spread of COVID-19 in India and effectiveness of preventive measures." *Sci Total Environ* no.728 **2020**:138762
26. Chimmula VK, Zhang L. "Time series forecasting of COVID-19 transmission in Canada using LSTM networks." *Chaos Solitons Fractals* no 138 **2020**:109864.
27. IHME COVID-19 Health Service Utilization Forecasting Team. Murray CJ."Forecasting COVID-19 impact on hospital bed-days, ICU-days, ventilator-days and deaths by US state in the next 4 months" **2020** Mar 30.
28. Pandey G, Chaudhary P, Gupta R, et al. "SEIR and Regression Model based COVID-19 outbreak predictions in India" *arxiv* **2020**.
29. Sujath R, Chatterjee JM, Hassanien AE. "A machine learning forecasting model for COVID-19 pandemic in India." *Stoch Environ Res Risk Assess* no. 34 **2020**:959–72.
30. Ghosal S, Sengupta S, Majumder M, et al. "Linear Regression Analysis to predict the number of deaths in India due to SARS-CoV-2 at 6 weeks from day 0 (100 cases - March 14th 2020)" *Diabetes Metab Syndr* no.14 **2020**:311–5
31. Arora P, Kumar H, Panigrahi BK. "Prediction and analysis of COVID-19 positive cases using deep learning models: a descriptive case study of India." no.139 *Chaos Solitons Fractals* **2020**:110017.
32. Gupta, Amit Kumar and Singh, Vijander and Mathur, Priya and Travieso-Gonzalez, Carlos M."Prediction of COVID-19 pandemic measuring criteria using support vector machine, prophet and linear regression models in Indian scenario." no.24 *Journal of Interdisciplinary Mathematics* **2021**:89-108.
33. Yu, Lang and Ma, Xin and Wu, Wenqing and Wang, Yong and Zeng, Bo." A novel elastic net-based NGBMC (1, n) model with multi-objective optimization for nonlinear time series forecasting." *Communications in Nonlinear Science and Numerical Simulation*, no.96 **2021**:105696.
34. Khan, Mohsin Ali and Memon, Shazim Ali and Farooq, Furqan and Javed, Muhammad Faisal and Aslam, Fahid and Alyousef, Rayed. "Compressive strength of fly-ash-based geopolymer concrete by gene expression programming and random forest." *Advances in Civil Engineering*, **2021**.
35. Shehadeh, Ali and Alshboul, Odey and Al Mamlook, Rabia Emhamed and Hamedat, Ola."Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression." no.129, *Automation in Construction*, **2021**:103827.
36. Lalapura, Varsha S and Amudha, J and Satheesh, Hariramn Selvamuruga."Recurrent neural networks for edge intelligence: a survey." no.4. *ACM Computing Surveys (CSUR)*, **2021**:1-38.
37. Lindemann, Benjamin and Maschler, Benjamin and Sahlab, Nada and Weyrich, Michael."A survey on anomaly detection for technical systems using LSTM networks." no.131. *Computers in Industry*, **2021**: 103498.
38. Sulandari, Winita and Suhartono and Subanar and Rodrigues, Paulo Canas."Exponential smoothing on modeling and forecasting multiple seasonal time series: An overview." no.20. *Fluctuation and Noise Letters*, **2021**:2130003.
39. Schaffer, Andrea L and Dobbins, Timothy A and Pearson, Sallie-Anne."Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: a guide for evaluating large-scale health interventions." no.21. *BMC medical research methodology*, **2021**:1-12.
40. Damacharla, P.; Rao, A.; Ringenberg, J.; Javaid, A.Y. "TLU-net: A deep learning approach for automatic steel surface defect detection". In Proceedings of *the 2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, Suzhou, China, 15–17 October **2021**; pp. 1–6.