Forecasting COVID-19 cases for the next seven days and beyond

Junchol Park
yojcpark@gmail.com
https://github.com/parkjlearning/covid19_forecasting
Springboard Mentors: Dipanjan Sarkar, Ankur Verma

Summary

Coronavirus disease 2019 (COVID-19) is a contagious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Since the first known case was reported in Wuhan, China, in December 2019, more than 261 million total cases and 5 million total deaths were reported worldwide, leading to an ongoing pandemic. Accurate forecasting of COVID-19 cases is critical for epidemiological, economic and personal coping strategies, and thus it poses an important challenge for data scientists who work on time series analysis and forecasting. Here we took diverse approaches and built four different kinds of models to predict the number of COVID-19 cases for the next seven days in 23 countries individually. We trained the models based on the features that are readily available to maximize the usability of our models, namely we used Google mobility, weather, vaccination, previous cases, and temporal data as features. We compared performance of the models with cross validation. We conclude that the neural network model with LSTM layers outperform others, however, the XGBoost regressor model might be considered for a faster outcome with a modest compromise in performance.

Table of Contents

Introduction

The ongoing COVID-19 pandemic caused by SARS-CoV-2 has profoundly influenced numerous aspects of our lives. Epidemiological data have been nicely visualized and available on dashboards (e.g., https://coronavirus.jhu.edu/map.html), which tracks global and local cases in the US. These data are essential not just to track the past and present but more importantly to forecast the future. An accurate forecasting is critical for epidemiological, administrative, economic, and personal coping strategies. Many groups have been working on this time series forecasting problem, and their 7-day predictions are being posted in the CDC website on a weekly basis (https://www.cdc.gov/coronavirus/2019-ncov/science/forecasting/forecasts-cases.html). While these groups use variants of time series forecasting methods, little effort has been made to compare performance of different models. Here we created four different (traditional and modern) kinds of models - SARIMAX, XGBoost, MLP, LSTM. Their performance was measured using common metrics and compared. Importantly, the four models were trained using the same dataset that comprised readily available data that are relevant to the pandemic, e.g. Google community mobility data, weather, date, vaccination etc. Models were trained to predict

COVID-19 cases for the next 7 days (on any given day) in 23 different countries to gauge their generalizability to various time varying patterns in COVID-19 cases. We tested models on both validation and test sets with cross validation. This first ensured that model performance was measured on held out data that are unseen during training. Also importantly, this cross validation scheme enabled us to test how models perform when predicting cases far ahead in the future, as the validation set comprised data for the next three weeks from the tail end of the training set, and the test set comprised the final three weeks of the entire time series.

Dataset

Target variable

The dataset consists of daily COVID-19 cases, namely, the number of cases per million ('case_mil') in 23 different countries from 02/15/2020 to 09/17/2021. Here is the list of countries in alphabetical order: Argentina (AR), Australia (AU), Austria (AT), Belgium (BE), Canada (CA), Denmark (DK), Finland (FI), France (FR), Germany (DE), India (IN), Indonesia (ID), Ireland (IE), Israel (IL), Italy (IT), Japan (JP), Korea (KR), Mexico (MX), Netherlands (NL), Norway (NO), Russia (RU), Singapore (SG), UK (GB), US (US). COVID-19 case data was downloaded from Our World in Data (OWID) repository https://github.com/owid/covid-19-data. Each model was trained to predict the number of cases in the next 7 days based on features prepared for each day. Thus, the target variable was preprocessed as a 2D numpy array whose dimension was the-batch-size-by-7. Importantly, 'cases per million' was used rather than cases per se as it provides a nice normalization against different population sizes across countries. For convenience, we will use 'cases' and 'cases per million' interchangeably.

Features

Feature variables comprise three kinds:
1) Static Categorical refers to categorical features that do not vary with time like country, continent etc.
2) Temporal Categorical refers to categorical features that vary with time like week, month of year etc.
3) Temporal Continuous refers to continuous variables that vary with time like temperature, vaccination rates etc.

Country ID was included as a static categorical feature variable. Exploratory data analysis (see below) has revealed that different countries show distinct time varying patterns in

COVID-19 cases, and capturing this diversity is critical in this project. SARIMAX and autoregressive integrated moving average models in general have no capacity to 'learn' distinct time series in a unified manner, i.e. separate models must be trained per country. In the other three models, country ID was included as a categorical variable (XGBoost) or as an embedding vector (MLP, LSTM) so that the relevant weight parameters could be adjusted for different countries.

Temporal categorical variables could contribute to capture temporal changes in times series at different scales. See table 1 for the list of temporal categorical variables. Note that the temporal categorical variables are not included in the SARIMAX model, in which the temporal dependencies are meant to be represented innately by the autoregressive, moving-average, and seasonal components of the model.

Temporal continuous variables are time varying continuous features. Google community mobility data (https://www.google.com/covid19/mobility/) provide movement trends over time by geography, across different categories of places such as retail & recreation, groceries & pharmacies, parks, transit stations, work, and residential. Mobility data display characteristic time varying patterns by places, which is thought to reflect how people are responding to COVID-19 policies as well as trends in interaction types among people that could be informative of trends in cases.

Four weather related data - temperature, humidity, cloud cover, precipitation were included as temporal continuous features. The relationship between weather conditions and COVID-19 cases has not been systematically established, but it is thought that there exists some relationship like faster spread of virus in cold temperature. Weather in the capital city was used for each country, as these features were meant to reflect the general climate differences and seasonal rather than daily changes in cases.

The vaccination rate is also a critical component of the forecast, and is thought to be negatively correlated with cases. 'Vaccination per hundred' was used as it provides a nice normalization for different population sizes.

Finally, the recent trend in cases is highly informative of the future. SARIMAX model by design identifies the linear temporal dependency of the target onto its own past. For other models we explicitly fed cases in the previous 14 days as features to represent the temporal dependencies. Also note that previous 14 days of other temporal continuous variables, namely, Google community mobility and vaccination data were also used as features to extract any temporal dependencies between the future cases and those lagged variables.

Table 1. Feature list.

Static Categorical	● Country ID (country_region_code)
Temporal Categorical	● Year (year) ● Month (month)

	• Day (day) • Week of year (week_of_year) • Day of week (day_of_week) • Holiday (holiday)
Temporal Continuous	• Google Community Mobility Data ○ Retail & Recreation (rtrc) ○ Grocery & Pharmacy (grph) ○ Park (prks) ○ Transit stations (tran) ○ Work (work) ○ Residential (resi) • Weather Data ○ Temperature (tempC) ○ Humidity (humidity) ○ Cloud cover (cloudcover) ○ Precipitation (precipMM) • Vaccination (vac: the number of vaccinated people per hundred) • Previous cases (cases during previous 14 days)

Exploratory data analysis (EDA)

Our EDA showed that different countries display distinct time-evolving patterns in COVID-19 cases, and the ability to capture these diverse patterns in time series would be the key for better performance. This provides a useful testbed for models controlling for a low bias high variance model, as a model that can only capture a specific pattern would poorly perform for others. To depict distinct time series, we plotted cases as a function of time in five different countries in **Fig. 1**.

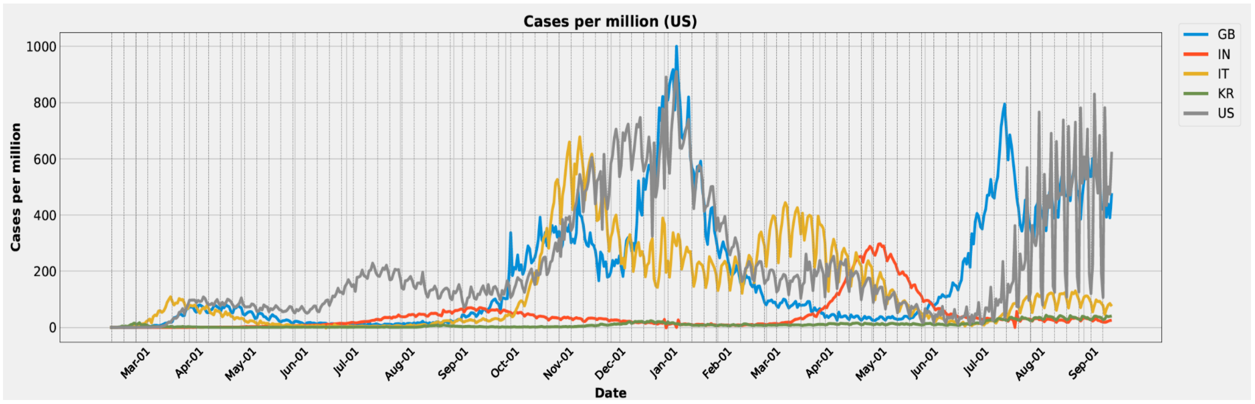

Figure 1. Distinct time varying patterns in COVID-19 cases across countries.

We then started to examine the relationship between the target and feature variables to make sure that our feature set includes variables that are relevant to the target. We first checked the relevance of Google mobility data. By plotting the mobility time series and cases on the same axes we could appreciate that mobility in different places display characteristic time-evolving patterns, which could be potentially informative in forecasting the future cases (**Fig. 2**). For instance, note that the mobility in parks (blue) shows an interesting negative correlation with the cases (black) in the US (**Fig. 2**). In addition, we observed a robust periodicity. Even with bare eyes mobility data fluctuated with a period of 7 meaning a weekly fluctuation. Given that the weekly rhythmicity also existed in the target (cases), it could leverage models' prediction power, if the regularity is modeled properly. Also, the widespread periodicity, which is nonlinear, implies that a linear regression would be a suboptimal model choice for this dataset.

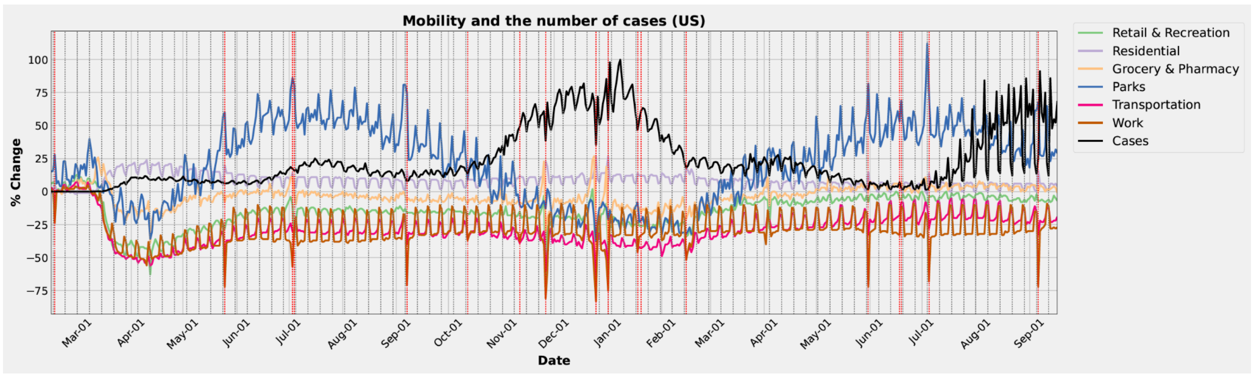

Figure 2. Distinct time varying patterns in mobility at places in the US.

We then visualized the correlation matrix to examine pairwise correlations amongst feature and target variables (**Fig. 3**). The correlogram revealed that many feature variables are positively or negatively correlated with the COVID-19 cases, justifying themselves as useful features. For instance, some mobility variables (retail & recreation, transit stations, work) showed negative correlation with cases as expected. Temperature was negatively correlated with cases consistent with the general notion. Thus, our EDA described characteristic time varying patterns in the temporal continuous features that are potentially correlated with the number of cases, suggesting that our models should be able to leverage the correlational structure in the dataset to make a prediction for the future cases.

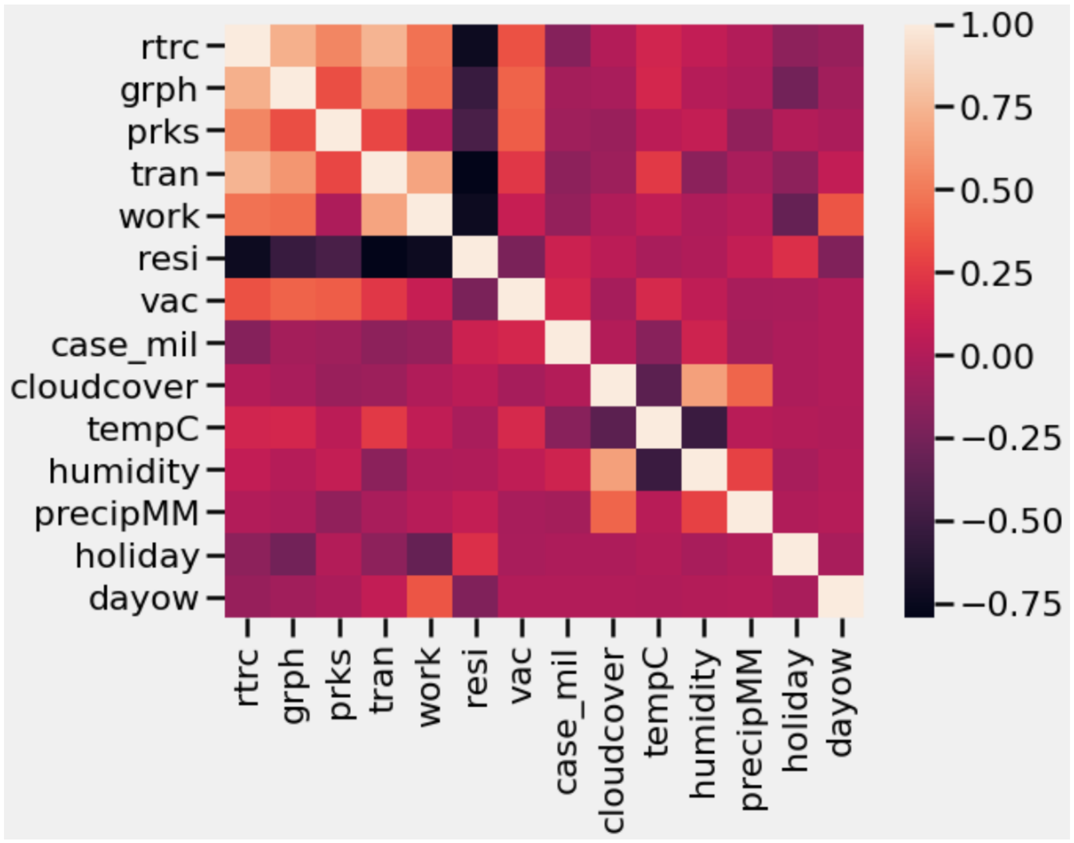

Figure 3. Correlations amongst pairs of feature and target variables.

Training and testing the model with cross validation

Before we dive into model performance we need to articulate how we trained, validated, and test our models. As described above we used a walk forward cross validation, which is widely used in time series forecasting. We first took the tail end chunk of data corresponding to the final 21 days of the dataset, which served as the test set. The test set was used to gauge model performance when predicting cases weeks or months ahead into the future (**Fig. 4**). After separating the test set the rest of the remaining dataset was split into five train and validation folds. At the end each train fold a validation fold (next 21 days) was apposed, and the model trained on the current train fold was tested on that validation fold. For the next iteration the validation fold was concatenated to the train fold to produce the new elongated train fold, and the new validation data (next 21 days) was apposed at the end, i.e. walk forward validation (**Fig. 4**).

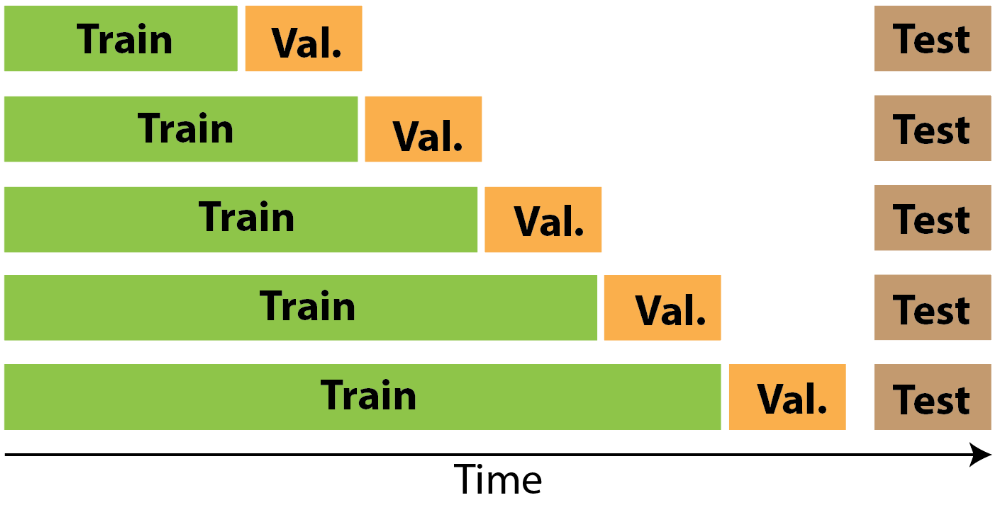

Figure 4. Walk forward validation.

SARIMAX model

Seasonal **A**uto-**R**egressive **I**ntegrated **M**oving **A**verage with e**X**ogenous factors is an extension of the ARIMA class of models that has long been used in univariate time series forecasting (https://phosgene89.github.io/sarima.html). In ARIMA time series is thought to comprise auto-regressive (AR) and residual errors (MA), and the linear dependency of the time series on these components is solved with application (the integrated term (I)) of a method(s) to ensure time series' stationarity, e.g., differencing. SARIMAX is based on ARIMA with two additional components; the seasonal component (S) is added to capture seasonality in the data, and exogenous (X) variables and their linear dependency on the time series are also modeled in SARIMAX. In **Fig. 3** we indicated that a number of exogenous variables are correlated with the number of cases. In **Fig. 5** we observe robust seasonality (period=7) in the example time series (cases in the US), which has also been observed in data from other countries. Thus, we reasoned that SARIMAX may be the best model choice than other variants.

A SARIMAX model was fit per country. There are six parameters to be determined: $(p, d, q), (P, D, Q)_m$ where p is the order of the auto-regressive model, d is the degree of differencing, and q is the order of the moving-average model, while upper case P, D, Q refers to corresponding terms but for the seasonal part of the model with m being the period of seasonality. We used a python wrapper 'Pyramid' that provides a grid search method to optimize the orders of AR and MA terms for seasonal and non-seasonal parts of the SARIMAX model based on Akaike information criterion (AIC).

We measured model performance on train, validation, and test sets, respectively. The root mean squared error (rmse) between actual and predicted future cases was used as the metric to evaluate performance. As expected the model performed well on the training set (**Fig. 6**). The prediction error increased on the validation set, i.e., when predicting cases farther ahead into the future upto 28 days.

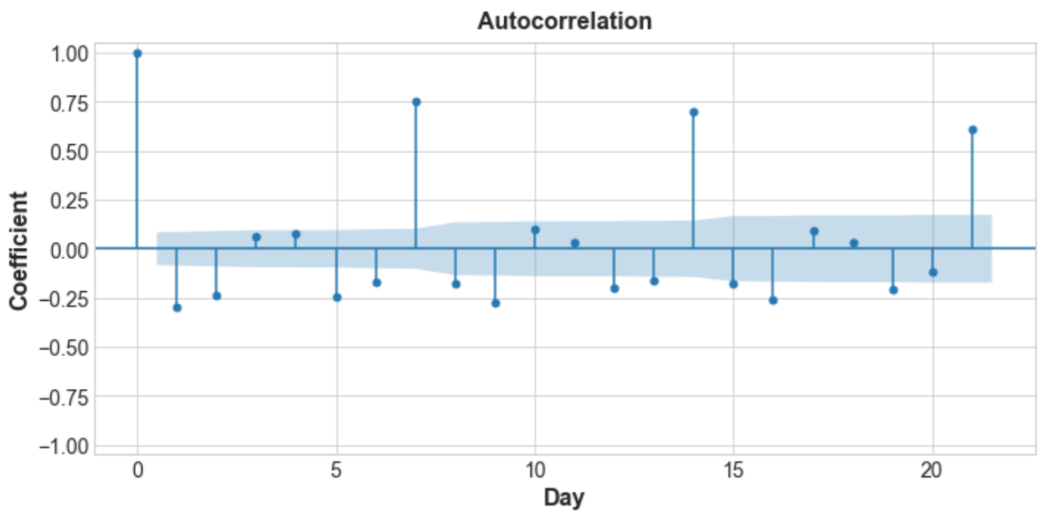

Figure 5. Autocorrelogram for COVID-19 cases in the US.

Note the peaks occurring on day 7 and its multiples, i.e. there is seasonality with a period of 7.

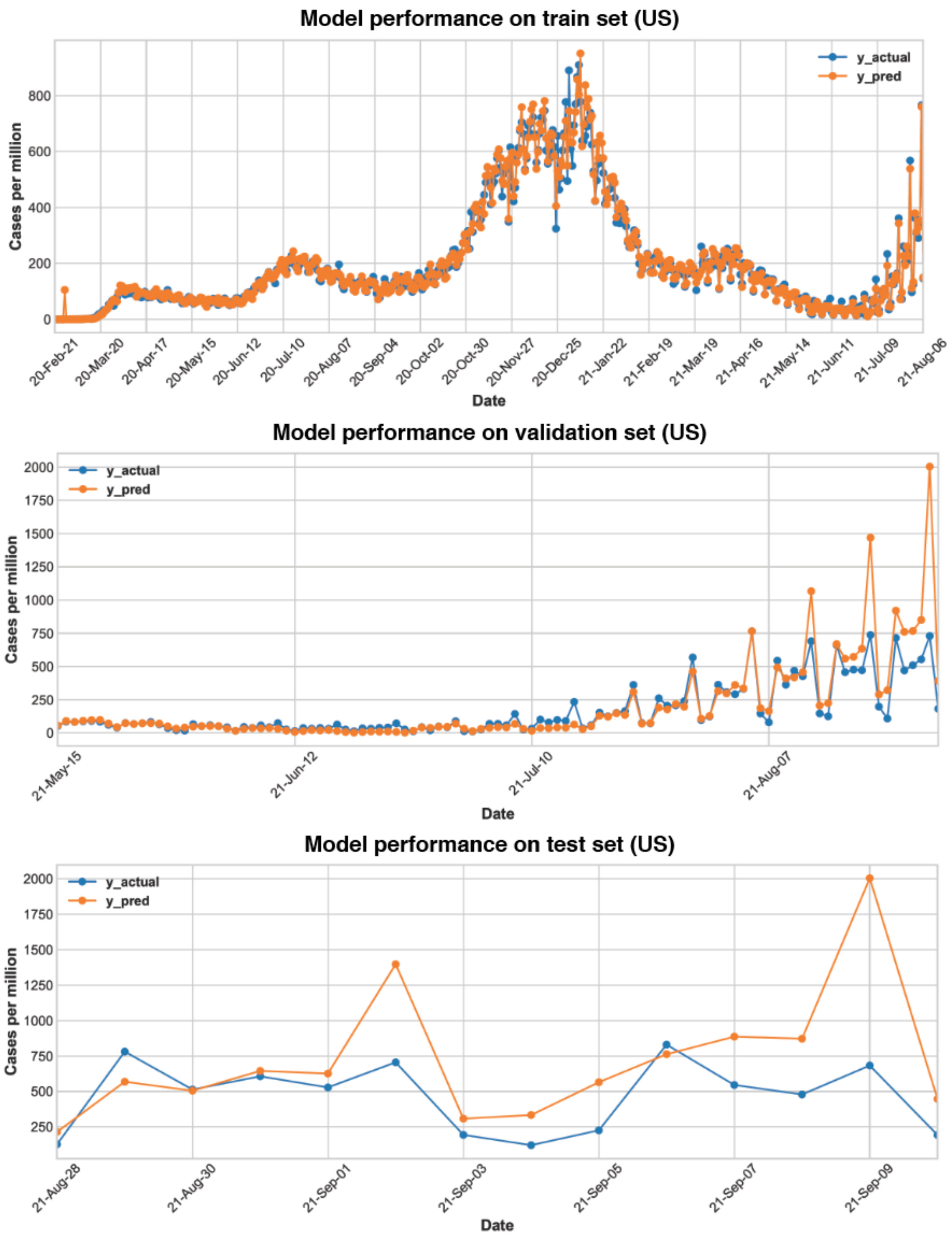

Figure 6. SARIMAX performance on the train, validation, and test sets (US data).

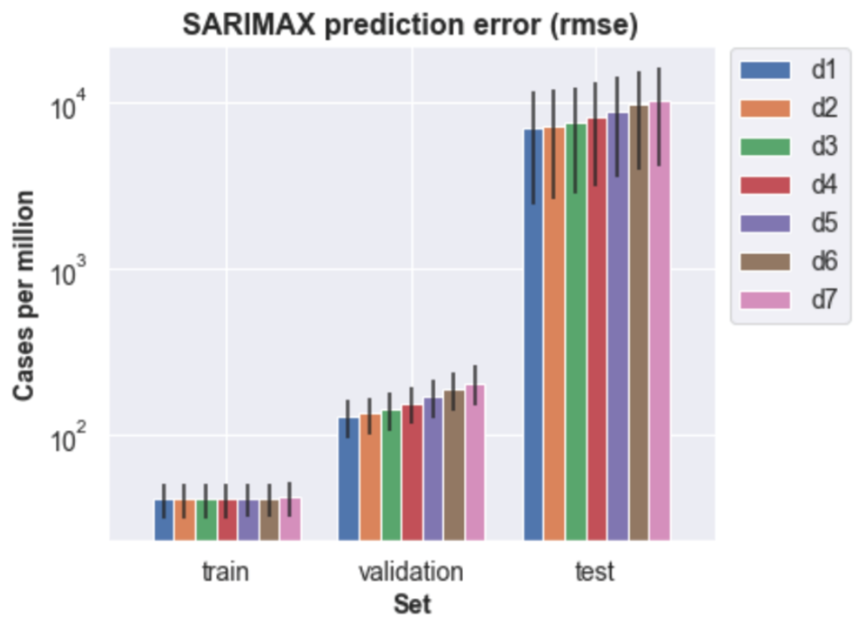

Figure 7. SARIMAX prediction errors on train, validation, and test sets averaged across countries (mean ± sem).

XGBoost regressor model

Gradient boosting is a machine learning method used in classification and regression tasks. It provides a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees. The model is optimized by iteratively growing a weak learner (e.g. a tree) in order to minimize the residual error. XGBoost is an open-source library that provides a regularizing gradient boosting [1]. XGBoost can be used for time series forecasting by transforming the time series into a supervised learning problem, meaning that lagged time series, e.g. cases in prior 14 days, are used along with other variables as features to predict the future of the time series, e.g. case in next 7 days.

We trained XGBoost models per country with the walk forward validation (**Fig. 4**). As described above model performance was measured on non-overlapping validation and test sets to measure models' predictability for near (up to 4 weeks ahead) and far (up to 4 months ahead), respectively (**Fig. 8**). A key hyperparameter in the XGBoost regressor model is the number of trees ('n_estimators') to grow for model fitting. Too many trees might lead to overfitting, while too few trees might lead to a highly biased model. We used a grid search method combined with the walk forward validation to optimize the number of trees, and concluded that 100 trees might be adequate.

XGBoost outperformed SARIMAX for both validation and test sets (**Figs. 8-9**). Notably, XGBoost achieved a very low degree of bias for the train set (**Fig. 8** *top*), suggesting that our models might have been overfitted to be potentially in the realm of low bias and high variance, meaning that their performance for validation and test sets could be improved if we further regularize the number of trees in the models.

Similar to other tree-based methods, XGBoost has a principled way to quantify feature importance, which is a score that indicates how useful or informative each feature was in constructing the boosted decision trees within the model. The more a feature is used to make key splits within the decision trees, the higher its relative importance, thus features can be ranked and their relative importance can be compared among one another. We observe that features like cases and vaccination in prior days tended to be highly ranked in our models. Interestingly, the number of cases reported 7 days back appeared to be most informative, consistent with the periodicity we observed when fitting the SARIMAX model. We note that feature importance was widely distributed over many of the features included in our models, suggesting that the vast majority of them were useful in predicting the future cases.

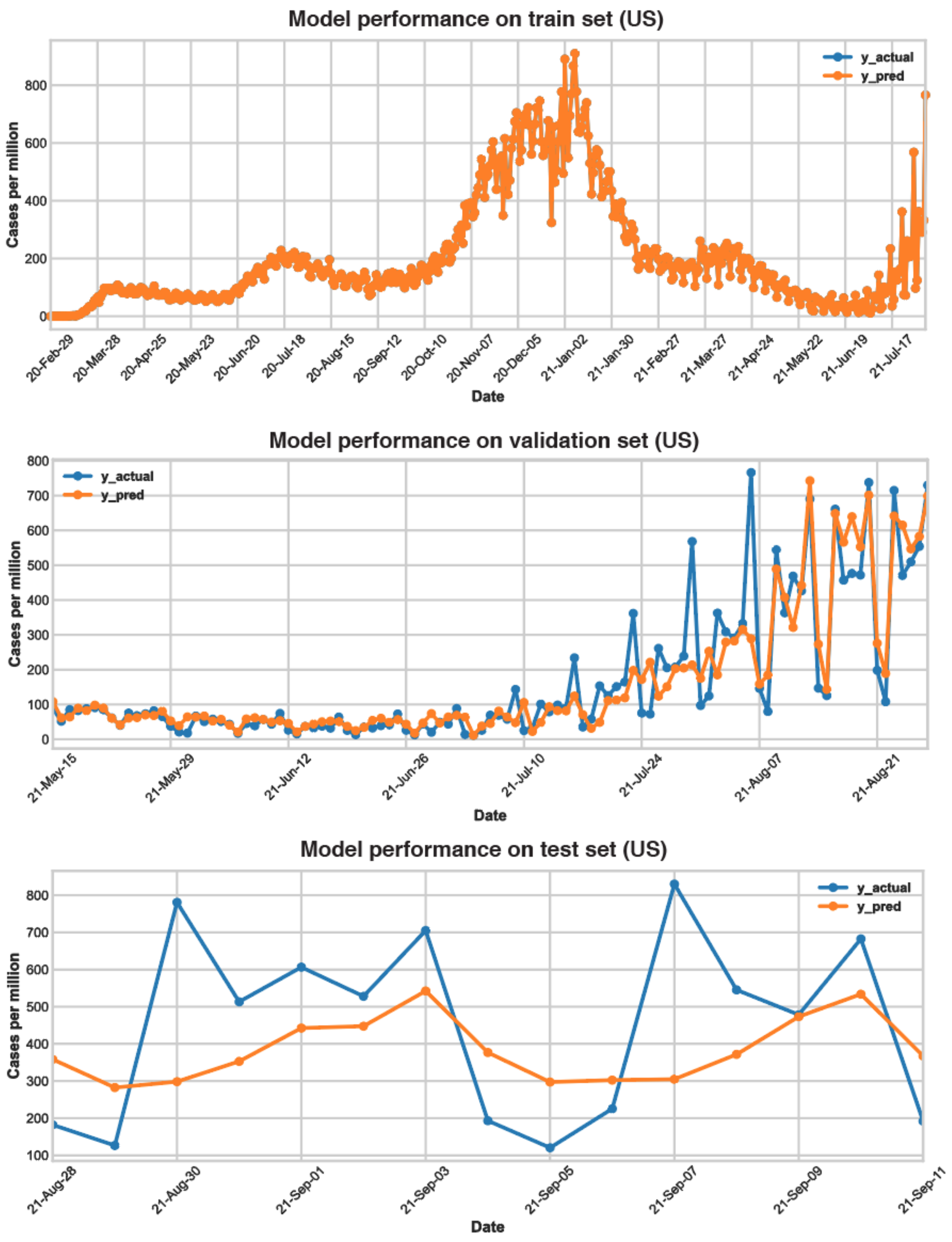

Figure 8. XGBoost performance on the train, validation, and test sets (US data).

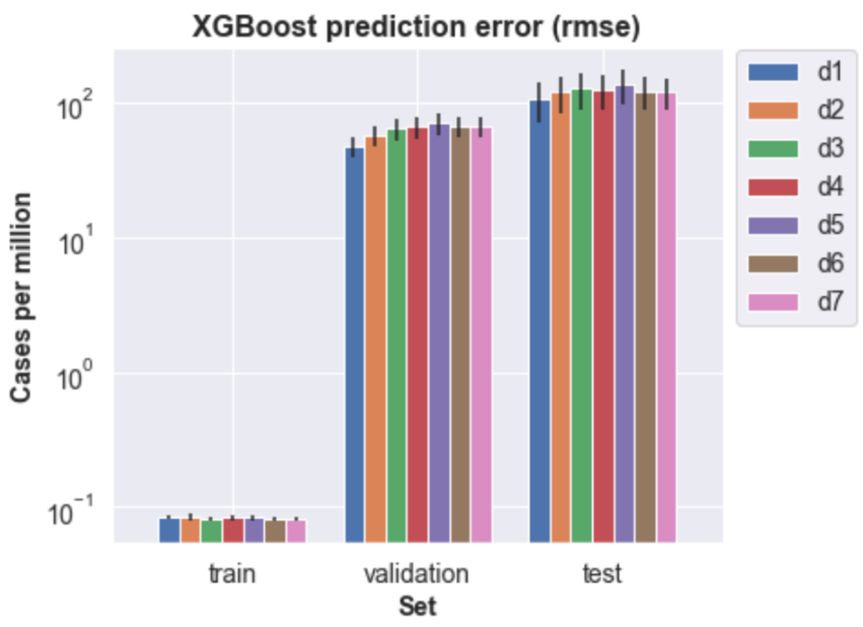

Figure 9. XGBoost prediction errors on train, validation, and test sets averaged across countries (mean ± sem).

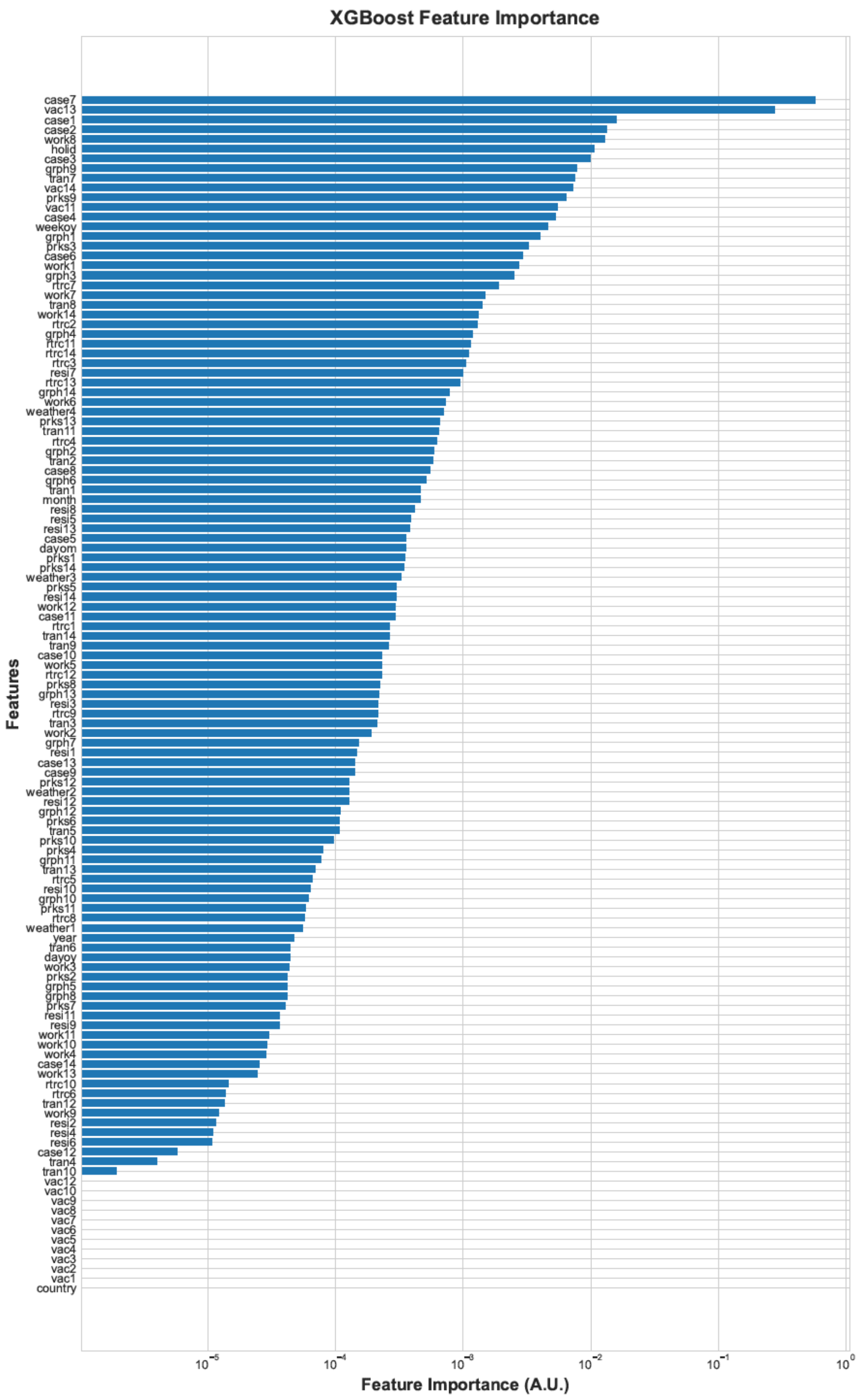

Figure 10. Feature importance in the XGBoost model fitted on the US dataset.

Multi Layer Perceptron (MLP) model

Time series forecasting can be represented using various neural network architectures, in which inputs are processed through one or more hidden layers comprising numerous units ("neurons") and activation functions. We first considered a relatively simple feedforward neural network, MLP comprising the input layer, embedding layers, a concatenation layer, and a series of densely connected layers with the activation function (ReLU). Detailed description of the architecture and model parameters can be found here: https://github.com/timeseriesanalysis_MLP_7d.ipynb.

In general, a neuron in the MLP (or in other architectures) computes a weighted sum of the inputs, followed by a non-linear activation of the weighted sum, as shown in this equation: $o_i = \varphi(\sum_{j=1}^{d} (x_j w_{ij} + b_j))$, where x is the input vector that is transformed by the weight matrix w, the bias b, and the non-linear activation function φ. While a linear model, mapping from features to outputs via matrix multiplication, can by definition represent only linear functions, the non-linear activation function, e.g. ReLUs, enables the feedforward networks with one or more hidden layers to represent theoretically any function [2]. The universal approximation theorem implies that regardless of what function we are trying to learn, we know that a large enough MLP will be able to represent this function, although there is no guarantee that the training algorithm will be able to learn that function. The latter point leads to an important discussion but it is not within the scope of this paper.

We trained and tested the MLP using the abovementioned walk forward validation (**Fig. 4**). MLP models were built using the Keras Functional API on GPUs powered by Google Colab. Categorical variables were mapped into Euclidean spaces using entity embeddings, and the mapping was learned during training the neural network. Use of entity embeddings is known to help the neural network to generalize better . The entity embedding layer was concatenated with the fully connected dense layer that receives temporal continuous variables, which consisted of mobility, weather, vaccination, and cases in the previous 14 days. The concatenated layer was then connected through four fully connected dense and dropout layers with their final output being a sequence (length = 7) as predictions were made for the next 7 days on each day.

Our results indicated that performance of the MLP might be suboptimal and it could be improved to reduce both bias and variance. This reasoning was based on the fact that the model performance on the training set was worse than the validation and test sets (**Fig. 11**), and the performance was highly variable as can be seen in the large error bars (**Fig. 12**). We attempted to improve performance of the neural network by capturing more complex temporal interactions amongst the temporal continuous features using the Long Short Term Memory network (LSTM).

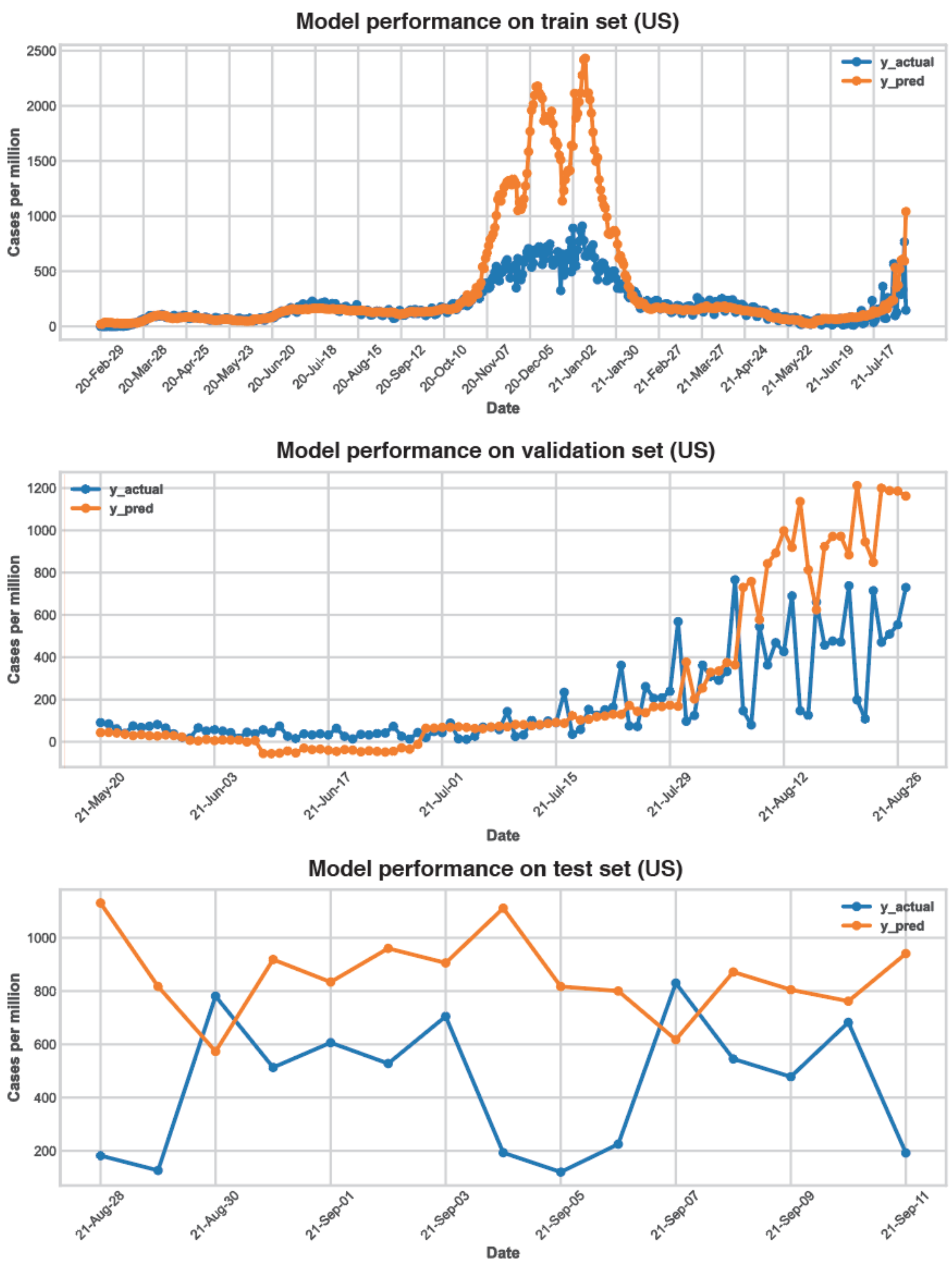

Figure 11. MLP performance on the train, validation, and test sets (US data).

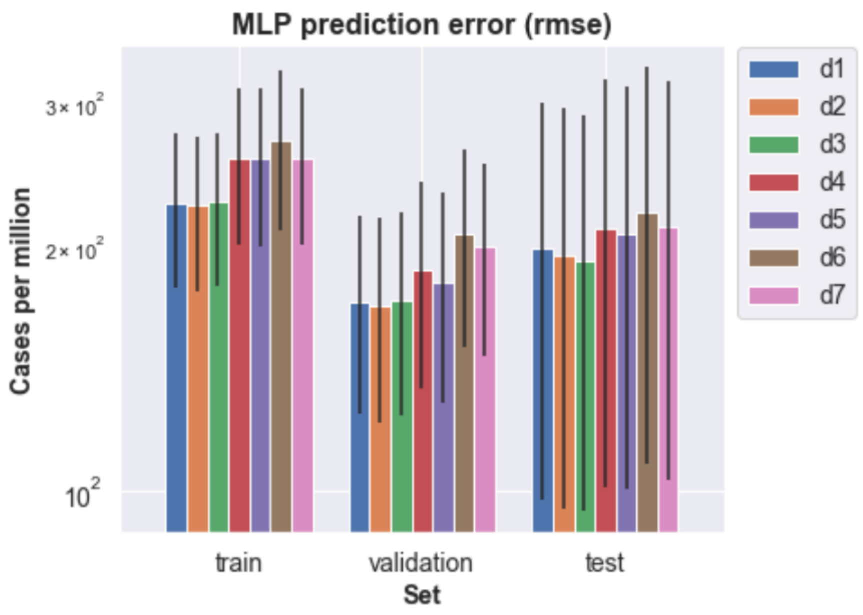

Figure 12. MLP prediction errors on train, validation, and test sets averaged across countries (mean ± sem).

Long Short Term Memory (LSTM) model

LSTMs are a special kind of recurrent neural network that is capable of learning long-term dependencies [3]. LSTMs are explicitly designed to selectively remember relevant information by adding or dropping information as the cell state progresses across time steps (https://colah.github.io/Understanding-LSTMs). They perform superbly in problems where long-term dependencies need to be solved, e.g., speech recognition and video analysis etc. Time series forecasting is another area wherein LSTMs can excel by capturing long-, short-term dependencies amongst various continuous features at different time scales, for example, seasonalities with different periods should be better represented by LSTMs than a simple feedforward architecture. Indeed, we found that a feedforward neural network (MLP) lacking any recurrency led to high bias and variance in forecasting. Therefore, we turned to LSTMs to capture temporal dependencies amongst variables at different scales. It has been shown that stacking multiple hidden LSTM layers on top of each other could further enhance the depth and could address even more complex temporal dependencies [4], thus we stacked dual LSTM layers.

Specifically, each of the eight temporal continuous (mobility (6) + vaccination (1) + case(1)) features was lagged to get a sequence of prior 14 days, and thus (batch size)-by-14-by-8 array was fed to the input layer. This input layer led to the dual stacked LSTM layers. The output of the second LSTM layer was connected to the time-distributed dense layers, which then was concatenated with the output of the rest of the neural network that processed the categorical features and non-lagged continuous features (weather) through entity embedding layers and dense layers. Details of the model and parameters can be found here: https://github.com/timeseriesanalysis_LSTM_7day.ipynb

As expected, model performance with the stacked LSTM layers improved when compared to the MLP, suggesting that LSTM's ability to selectively "remember" relevant information may help the model to capture complex interactions amongst features at different time scales. Model performance might be further enhanced by increasing the depths of the LSTM model, for example, stacking more LSTM layers, increasing the number of memory cells, and/or expanding each memory cell's dimension etc. While this is one intriguing direction for future endeavor, it should also be noted that this type of model has numerous parameters (the current LSTM model had 42,198 trainable parameters) and so takes many hours (or more) to train, precluding its usage in some user cases where speed and/or online training matter.

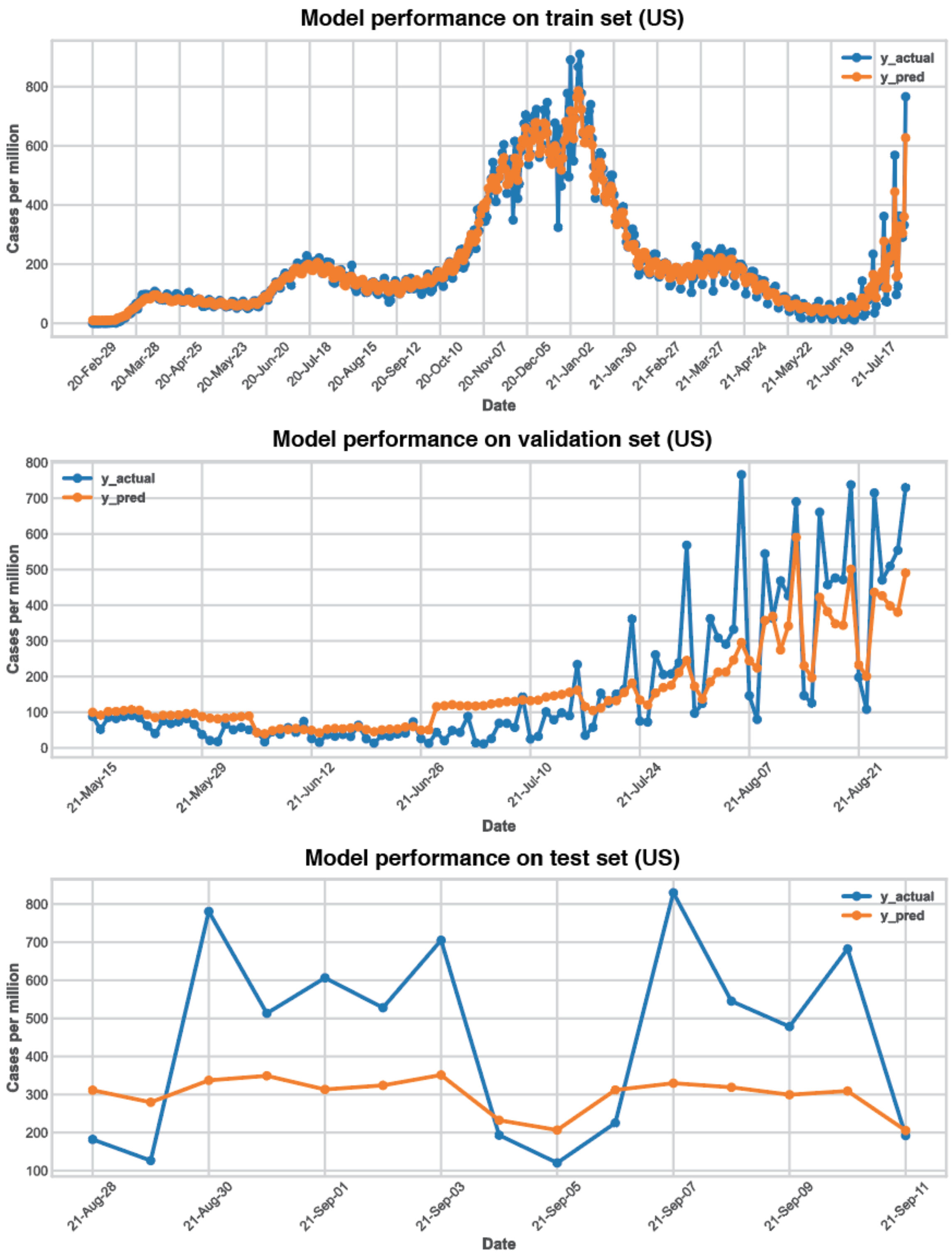

Figure 13. LSTM performance on the train, validation, and test sets (US data).

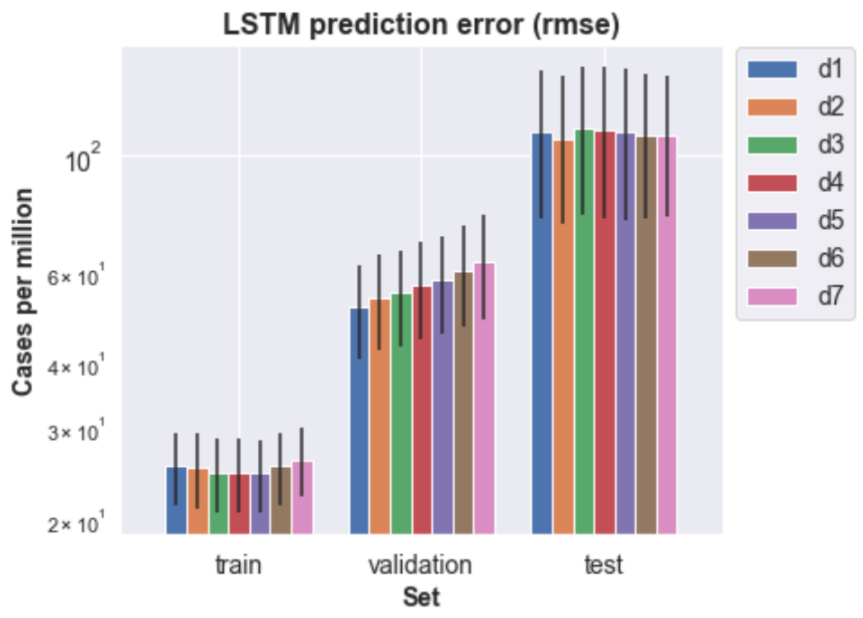

Figure 14. LSTM prediction errors on train, validation, and test sets averaged across countries (mean ± sem).

Comparing the four models

To compare performance of the four kinds of models we plotted their performance (rmse) on the train set (**Fig. 15**), the validation set (**Fig. 16**), the test set (**Fig. 17**). First, we note that the XGBoost model tended to overfit during training, possibly falling into the low bias, high variance regime (**Fig. 15**). A remedy for this would be to further regularize hyperparameters like the number of trees, which may increase bias but likely decrease the variance, leading to a better performance on unseen data (validation and test sets). We also noted that the MLP model might have suffered underfitting, which is observed as large errors in **Fig. 15**. While underfitting could be resolved by data-driven approaches e.g., adding more features or better feature engineering, we took it as a motivation for improving the model by leveraging the LSTM's strengths in selectively remembering relevant information on a large timescale. Indeed, adding stacked dual LSTM layers proved to be useful, as the LSTM model performed best on validation and test sets compared to the MLP and other models (**Fig. 16-17**). Performance of the SARIMAX model tended to decline as a function of the time interval between training and testing. It could be adequately fitted on the training data (**Fig. 15**), but it performed poorly in predicting cases far ahead in the future (**Fig. 16-17**). Thus, the model would be only appropriate for forecasting the near future. Based on these results, we conclude that the LSTM and XGBoost might be the most appropriate models for forecasting COVID-19 cases in the future up to weeks and even months ahead. We intended to use a set of features that are readily accessible to maximize the useability of the models. However, incorporating more features like social distancing policies and local events may further improve the model performance.

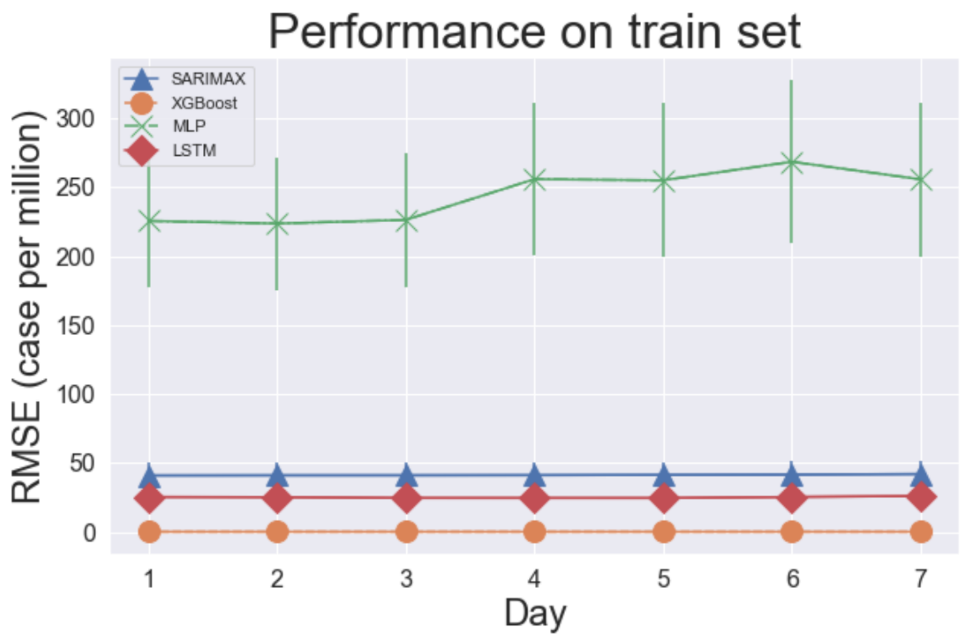

Figure 15. Performance of the four models on the train set (mean ± sem).

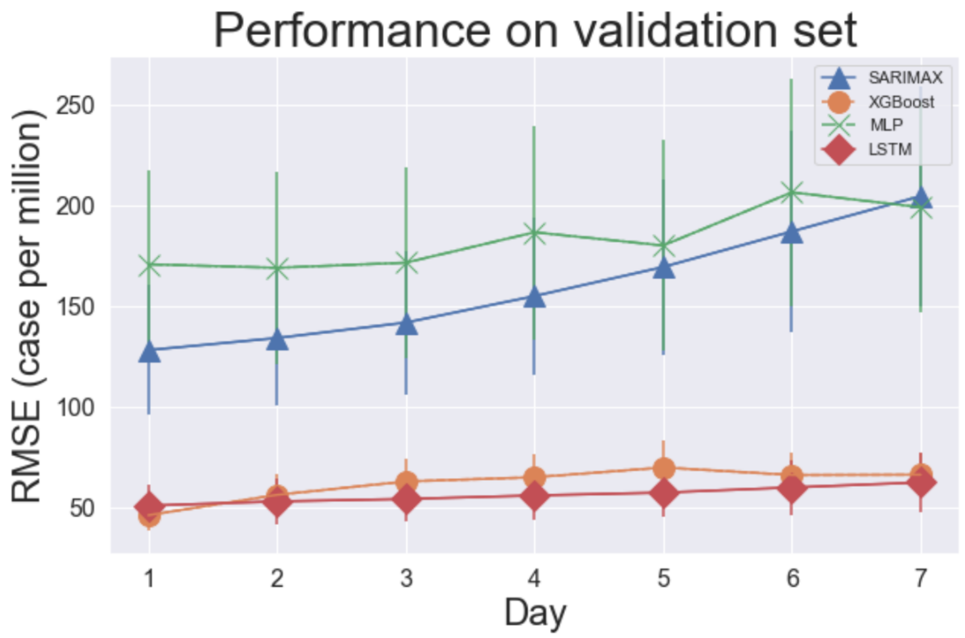

Figure 16. Performance of the four models on the validation set (mean ± sem).

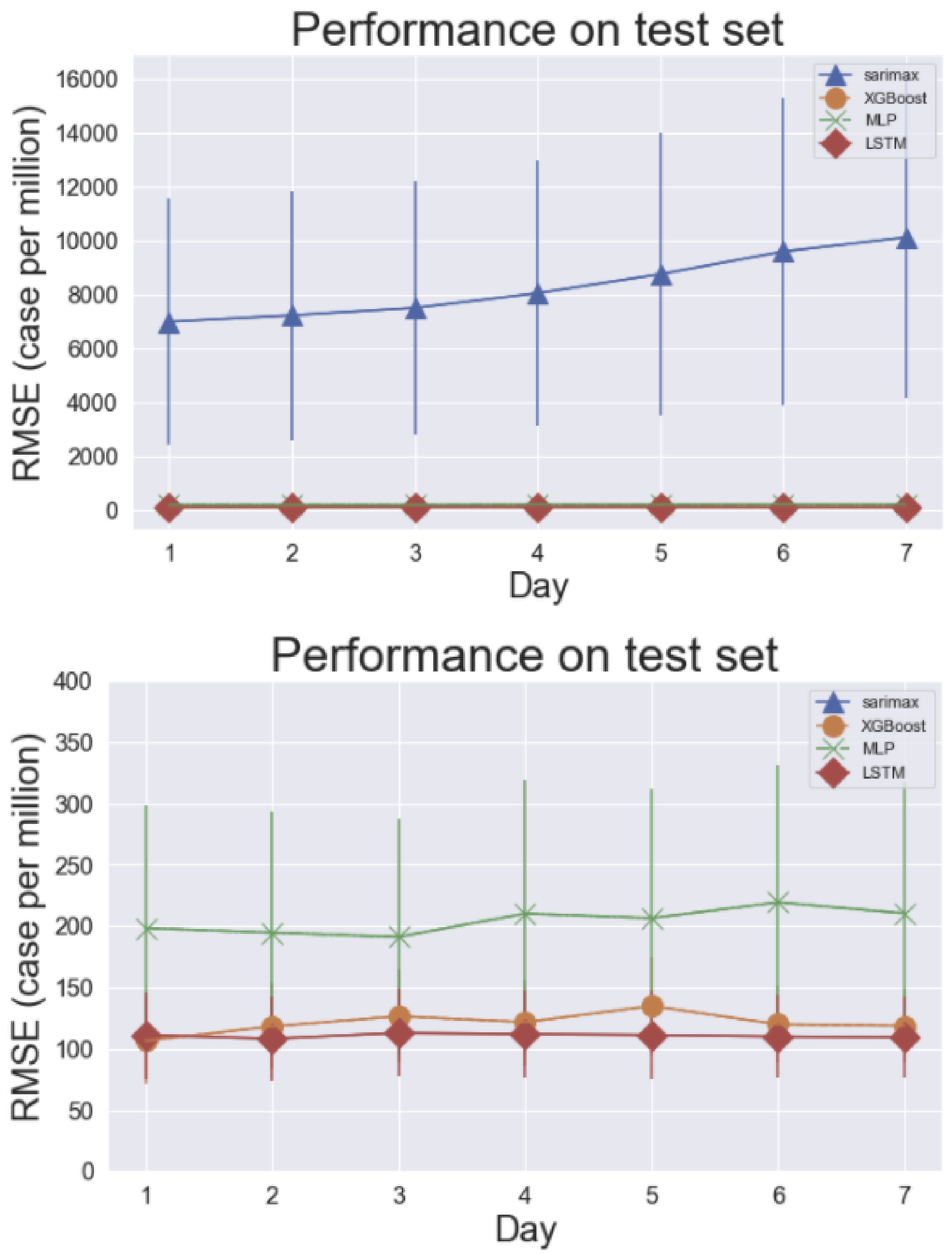

Figure 17. Performance of the four models on the test set (mean ± sem).

We plotted data on two different y-axes for a proper comparison amongst models.

1. Chen, T. & Guestrin, C. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016) doi:10.1145/2939672.2939785.
2. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning*. (MIT Press, 2016).
3. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
4. Graves, A., Mohamed, A.-R. & Hinton, G. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013) doi:10.1109/icassp.2013.6638947.