Willie X.
STAT 390 - Group 2

Time Series Prediction Project Final Report

I. Abstract

This data science project aims to accurately predict the number of new cases of COVID-19 for some of the G20 and G24 countries using regression with linear, tree-based, and neural network models. First, our group preprocessed the data focusing on issues like missingness and multicollinearity using solutions like clustering and correlation matrix as well as dealing with missingness differently for different types of machine learning models: ARIMA, AutoARIMA, Prophet (Univariate), Prophet (Multivariate), XGBoost, and LSTM. Next, each group member focused on a specific type of model to examine and create and preprocess training, testing, and validation sets for model tuning along with feature engineering and variable and/or model transformations. Overall, the best-performing model based on my results is the XGBoost model with a log transformation on the new cases of COVID-19.

II. Project and Data Overview

A. Project Overview

The goal of this time series projection is to predict new cases of COVID-19 for G20 and G24 countries on a country level using 6 regression models: ARIMA, AutoARIMA, Prophet (Univariate), Prophet (Multivariate), XGBoost, and LSTM. By determining the best model(s) in predicting new cases of COVID-19, health organizations will be able to determine where and when to combat emerging large cases of COVID-19.

The dataset used is originally from Kaggle titled *Our World in Data - COVID-19*[1], which originally contains 341,000 observations and 67 variables. The variable of interest as a response variable for the models will be **new_cases.** Some notable predictors other than **date** include:

COVID data:

- **total_deaths** / **new_deaths** = The amount of deaths due to COVID occurring
- **reproduction_date** = Real-time estimate of reproduction rate of COVID
- **positive_rate** = The share of COVID tests that are positive

Geographic data:

- **continent** = The continent where the new cases of COVID are occurring
- **location** = The country where the new cases of COVID are occurring

Country data

- **population_density** = Population of country divided by its land area
- **gdp_per_capita** = Gross domestic product of a country
- **extreme_poverty** = Share of population living in extreme poverty

Hospital data

[1] Dataset Source: Our World in Data. (2023). Our World in Data - COVID-19 [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/6559049

icu_patients / **hosp_patients** = Number of COVID patients in the icu / hospital
total_tests / **total_vaccinations** = Total tests and vaccinations for COVID taken

The models use information due to geography, countries, hospitals, and COVID in order to better accurately predict the trend of new cases of COVID as indicated by **new_cases.**

B. Data Overview and Preprocessing

From the initial 67 predictors, 2 hold no significant value:

iso_code = country code **tests_units** = units COVID is measured in.

Next, variable missingness is addressed. As shown in Table 1 in the Appendices, there are 34 predictors that have large missingess as indicated by the low completion rate. However, 4 predictors may be significant in predicting new_cases as indicated in Table 2 in the Appendices:

new_tests **total_vaccinations** **total_tests** **positive_rate**

Note there are other predictors in Table 2 with higher correlation not mentioned above since these variables have large missingness due to country related reasons and thus, are not usable such as **hosp_patients**. Next, the last data cleaning step is to remove predictors with multicollinearity issues. As shown in Table 3 in the Appendices, there are 8 predictors that have multicollinearity issues:

new_cases_smoothed **new_deaths_smoothed** **median_age**
aged_65_older **aged_70_older**

Finally, after removing some scaled predictors such as **_smooted** or **_per_million** predictors, there are a total of 38 predictors remaining along with 1 response = **new_cases** with 32,000 observations. There was only one main challenge in preprocessing which was that some predictors have large missingness due to country related issues as mentioned above so each large missingness was analyzed individually to determine if the missingness was random.

At this point for feature engineering, 4 new predictors are created: an indicator variable, **G20**, to determine if a country is in the G20 (1) or not (0) and another indicator variable for **G24**, **month** to indicate an observation's current month (1-12), and **day_of_week** to indicate an observation's current day in the week (Monday-Sunday). Then, the dataset is split into its training and testing set with training having observations before January 2023 and testing having observations starting January 2023. At this point, the data is also then differentiated into 3 datasets each for a different type of model: linear, tree-based, and neural nets.

Finally, to fix any remaining random missingness, a k-means clustering model was implemented using all the time-independent predictors, **life_expectancy, female_smokers,** and **male_smokers**, which were normalized in order to compare distances, to find the optimal number of clusters to group the data using the metric average silhouette[2]. Since most of the random missingness is between February 1, 2021 and March 1, 2022, then any missingness from a predictor is imputed with the median value of that predictor's cluster. Missingness outside the aforementioned date range will be handled differently by different models.

[2] "K-Means Cluster Analysis." K-Means Cluster Analysis · UC Business Analytics R Programming Guide, University of Cincinnati, uc-r.github.io/kmeans_clustering. Accessed 26 Nov. 2023.

For linear models, any predictors still with missingness are removed from the linear model dataset. Additionally, outliers are checked by using the Cook's distance method with a critical value of 0.5, which provides results that suggest there aren't any substantial outliers. For tree-based models, any predictors still with missingness are imputed with a very large value, 10^{15}, in the tree-based dataset. Finally, for neural network models, any predictors still with missingness will have an indicator predictor generated which indicates if value is present (TRUE) or missing (FALSE). Specifically, the only predictors with such indicators are **total_tests_b, new_tests_b, positive_rate_b, total_vaccinations_b,** and **people_vaccinated_b**.

Additionally, one main issue with the response, **new_cases**, is that some countries reported their new_cases differently or changed their reporting method. In the dataset, India, Italy, Japan, Pakistan, Saudi Arabia, Sri Lanka, and the United Kingdom always reported their new cases daily. France and Germany reported their new cases weekly while Argentina, Mexico, Australia, Canada, Colombia, Ecuador, Ethiopia, Morocco, Philippines, Russia, South Africa, South Korea, Turkey, and the United States all changed their daily new cases reporting to weekly reporting after some point in time. In order to be able to model each data comparably, **new_cases** is turned into daily data if it wasn't already. The solution implemented is to average the **new_cases** if weekly into daily **new_cases**. To check the preprocessing, refer to the preprocessing files located in **Preprocessing Code/Final Preprocessing** in the Github repository[3] in the Appendices. After all the preprocessing is done, each of the linear, tree-based, and neural network datasets have a training set with 25,000 observations and a testing set with 6,000 observations.

III: Methodology

A. ARIMA Model

1. Data Preparation

Using the linear dataset, first the data is filtered to only have observations after the first instance of COVID-19 for any country in the dataset which is January 4, 2020. Next, each country's daily data is transformed into weekly data using a weekly rolling average.

Next, check if a country has stationary or non-stationary data using the Augmented Dickey-Fuller Test with an alpha level of 0.05. As indicated in Table 4 of the Appendices, the only **non-stationary** countries are:

Australia	**Canada**	**France**	**Germany**	**Italy**
Russia	**Sri Lanka**	**United-Kingdom**		

For model parameter tuning, the training set of the linear data is used to create validations sets using a rolling origin backtesting method. For each validation set, the training part is a year worth of observations by country = 53 observations, the testing set is 2 months worth of observations by country = 8 observations, and the validation sets are incremented by 4 months by

[3] https://github.com/AzureAmber/STAT-390-Covid-Project/tree/main

country = 16 observations for a total of 6 validation sets. For the ARIMA model, 1 parameter is fixed and 6 parameters to be tuned with possible values:

Fixed	Tune
- seasonal_period = "auto"	- non_seasonal_ar = {0, 2, 4} - non_seasonal_differences = {0, 1, 2} - non_seasonal_ma = {0, 2, 4} - seasonal_ar = {0, 2, 4} - seasonal_differences = {0, 1, 2} - seasonal_ma = {0, 2, 4}

2. Model Building

Using R's *arima* package directly resulted in issues where the model was consistently a line regardless if the data was stationary or non-stationary. Thus, the solution was to implement the ARIMA process manually. First, to model any data in general is Response = Trend + Error. For ARIMA, specifically, is Response = **Linear Trend + Seasonality + Error**. The **linear trend** can be modeled using a simple linear regression from R's built-in lm() function. The **seasonality** can be modeled alongside lm() by treating seasonality as a categorical regressor. Finally, the **error** can be modeled using an ARIMA model where the parameters mentioned above will be tuned by minimizing the metric root mean squared error, RMSE. Finally, the predicted linear trend added with the predicted seasonality and predicted error from ARIMA will be the final prediction to the response variable, new_cases. The predictor used to predict the response **new_cases** is only **date**.

Another issue is that the first few weeks of any country generally has very low or 0 cases of COVID which skews the linear trend so a solution was that the linear trend will only be modeled using observations after the first 3 months of the training data. Finally, to check if the each country is stationary through this new process after removing linear trend and seasonality, refer to the results in Table 5 of the Appendices where the only **stationary** countries are:

Ecuador	**Japan**	**Mexico**	**Morocco**	**Pakistan**
Russia	**South Korea**	**Turkey**	**United States**	

Additionally, another major issue with the ARIMA model in R is that **seasonal_period** = "auto" else the model takes a much longer time to tune parameters. After applying a regular grid of 3 levels for each of the 6 parameters to be tuned resulting in $3^6 = 729$ parameter combinations on the 6 validation sets, the results of the best parameters for each country are (p,d,q,P,D,Q):

Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Parameters	4,1,2,2,0,0	2,0,4,2,0,2	2,0,2,1,0,2	4,2,0,2,2,2	4,2,4,1,0,1	4,0,2,2,0,1
Country	France	Germany	India	Italy	Japan	Mexico
Parameters	4,0,0,1,0,1	4,0,2,1,0,1	2,0,2,2,0,0	4,2,4,2,0,1	4,0,0,2,1,0	2,2,4,2,0,2
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa

Parameters	0,0,2,2,0,2	4,0,2,2,0,0	0,0,2,1,1,0	4,2,2,2,0,0	4,2,2,2,0,1	2,0,0,0,0,0
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Parameters	2,2,4,2,0,2	4,2,2,2,0,0	2,1,0,2,2,1	0,1,0,1,0,0	2,0,2,2,0,0	

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to **new_cases** and then applied the ARIMA process. An issue with log transformation is that log cannot be applied to **new_cases** = 0 so a solution is to convert all log(0) into 0. As indicated in Table 6 in the Appendices, the only non-stationary countries:

Australia Ethiopia France Sri Lanka Turkey

The results for the best parameters for log ARIMA:

Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Parameters	4,1,4,2,0,2	0,1,2,2,0,1	2,2,4,2,0,1	2,2,0,2,0,1	2,1,4,2,1,0	4,0,2,2,1,2
Country	France	Germany	India	Italy	Japan	Mexico
Parameters	4,2,4,2,0,0	2,0,0,2,1,0	0,1,0,2,0,2	2,1,0,2,1,1	2,0,4,2,0,2	4,0,4,2,1,1
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Parameters	4,2,2,2,1,0	4,1,0,2,1,1	2,2,4,2,0,2	2,0,4,2,0,1	4,0,2,1,1,2	4,0,2,2,0,0
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Parameters	0,0,4,0,0,1	4,2,0,2,1,1	2,1,0,2,0,1	4,0,4,1,0,2	4,2,2,2,1,1	

3. Model Performance

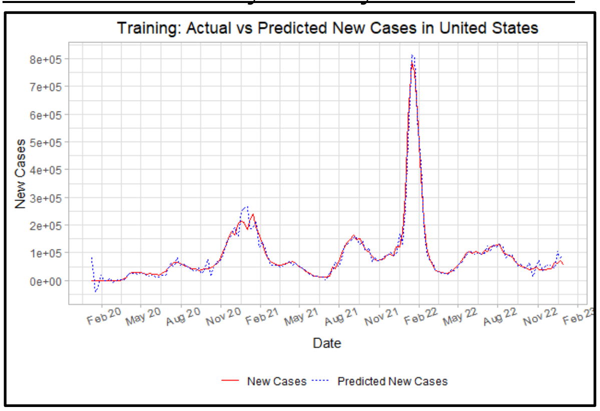
The RMSE for the ARIMA and log ARIMA models on the training and testing sets:

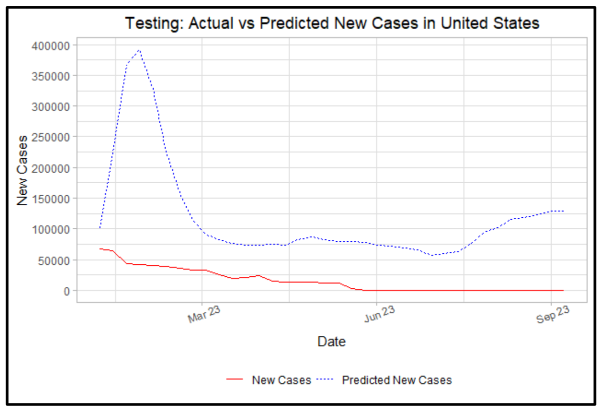
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	2751	2769	1247	1458	424	200
Test RMSE	16177	37152	7372	6219	1772	332
log Train RMSE	3880	13186	1553	3540	1168	708
log Test RMSE	43006	52424	546	4967	4967	113
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	7313	4521	11142	5033	9687	1774
Test RMSE	96537	95225	53727	32833	123645	8437
log Train RMSE	21455	10187	19202	9263	32591	3414
log Test RMSE	11917	52656	2736	2823	134905	4694
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa

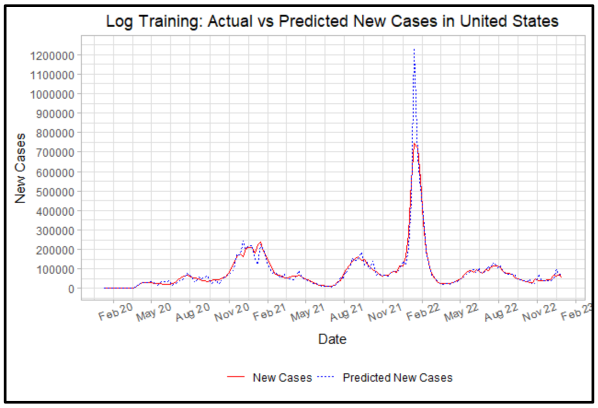
Train RMSE	470	384	2249	3255	236	1307
Test RMSE	1442	1012	4436	22070	1478	3380
log Train RMSE	668	2236	2807	3693	330	2075
log Test RMSE	23	33	341	3160	99	132
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	11581	162	5101	8679	20925	
Test RMSE	101029	487	34135	47814	126535	
log Train RMSE	20944	520	5809	6992	43808	
log Test RMSE	1022714	4	1	1629	16005	

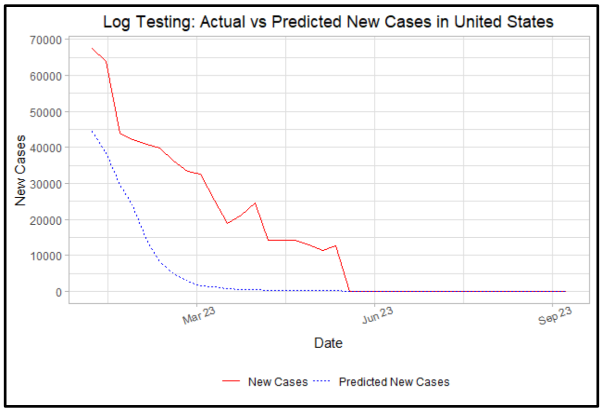
Generally, the log transformed ARIMA model isn't as accurate on the training set, but improves the testing set's accuracy. The R code for the ARIMA model is in Github Repository in **Models/regular/wx_arima.R** and **Models/log/wx_arima_log.R**.

Plots of a Stationary Country - United States

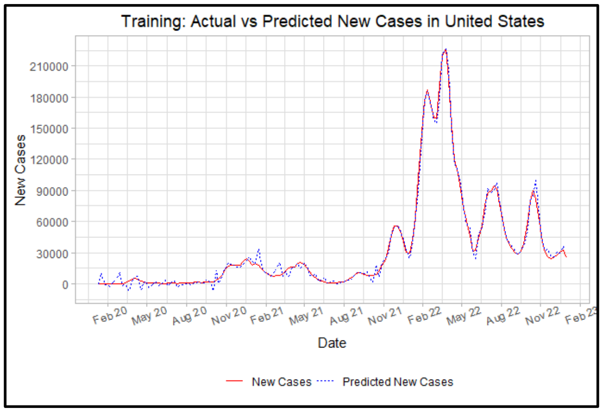


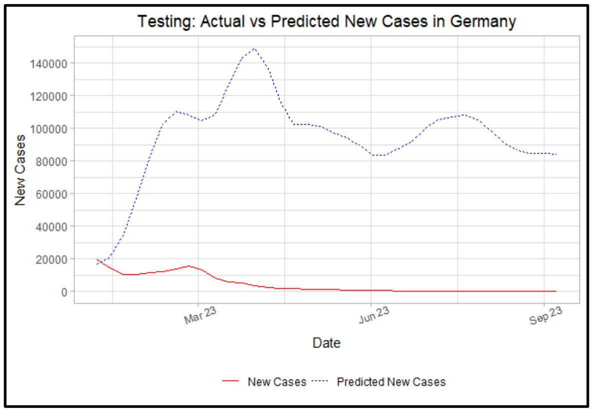


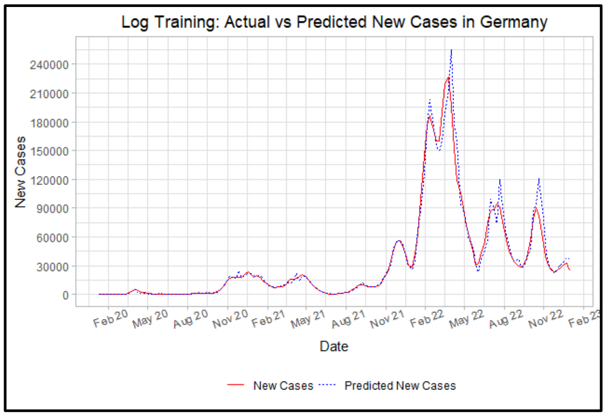


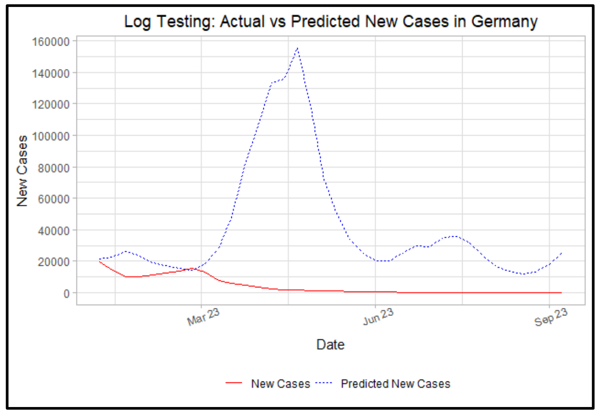


Plots of a Non-Stationary Country - Germany









B. AutoARIMA Model

1. Data Preparation

The process applied to the ARIMA model is applied to the AutoARIMA model. First, the linear dataset is filtered to only have observations after the first instance of COVID-19. Next, each country's daily data is transformed into weekly data using a weekly rolling average.

Similarly to ARIMA, check if a country has stationary or non-stationary data using the Augmented Dickey-Fuller Test with an alpha level of 0.05 as indicated in Table 4 of the Appendices. Similarly to ARIMA, the validations sets are constructed from the linear training set using a rolling origin backtesting method for a total of 6 validation sets. For the AutoARIMA model, 1 parameter is fixed and 6 parameters to be tuned with possible values:

Fixed	Tune
- seasonal_period = 53	- non_seasonal_ar = {0, 2, 4} - non_seasonal_differences = {0, 1, 2} - non_seasonal_ma = {0, 2, 4} - seasonal_ar = {0, 2, 4} - seasonal_differences = {0, 1, 2} - seasonal_ma = {0, 2, 4}

2. Model Building

Similarly to ARIMA, using R's *auto_arima* package directly resulted in issues where the model was consistently a line regardless if the data was stationary or non-stationary. Thus, the solution was to implement the ARIMA process manually as described in ARIMA. Also, all the issues mentioned in ARIMA are also issues for AutoARIMA which are addressed similarly. Thus, the stationary check remains the same as the results in ARIMA as presented in Table 5 of the Appendix. Thus, the only difference between the ARIMA and AutoARIMA models is the R packages used to implement each model. The predictor used to predict the **new_cases** is **date**.

Additionally, seasonal_period = 53 for the 53 weeks in a year accounting for leap years since R's auto_arima package is able to deal with seasonal_period ≠ "auto" without the time setbacks. Finally, applying a regular grid of 3 levels for each of the 6 parameters to be tuned resulting in 3^6 = 729 parameter combinations on the 6 validation sets, the results of the best parameters for each country are (p,d,q,P,D,Q):

Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Parameters	2,0,2,0,0,0	0,1,4,0,0,0	0,0,4,0,0,0	0,1,0,0,0,0	4,0,2,0,0,0	0,0,2,0,0,0
Country	France	Germany	India	Italy	Japan	Mexico
Parameters	2,0,2,0,0,0	0,1,4,0,0,0	2,0,4,0,0,0	2,1,0,0,0,0	0,0,4,0,0,0	0,0,4,0,0,0
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Parameters	0,0,4,0,0,0	4,0,0,0,0,0	0,1,2,0,0,0	4,0,2,0,0,0	0,0,4,0,0,0	4,0,0,0,0,0
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Parameters	2,1,0,0,0,0	4,1,0,0,0,0	2,1,0,0,0,0	0,0,4,0,0,0	0,1,2,0,0,0	

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to **new_cases** and then applied the AutoARIMA process. Since the AutoARIMA process is the same as ARIMA process, it has the same issues as ARIMA with the same solutions implemented. The stationary check is the same with Table 6 in the Appendices.

The results for the best parameters for log AutoARIMA:

Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Parameters	2,0,0,0,0,0	0,1,0,0,0,0	2,0,2,0,0,0	2,0,0,0,0,0	4,0,2,0,0,0	0,0,4,0,0,0
Country	France	Germany	India	Italy	Japan	Mexico
Parameters	4,0,2,0,0,0	2,0,0,0,0,0	4,0,0,0,0,0	0,1,4,0,0,0	0,0,2,0,0,0	0,1,2,0,0,0
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Parameters	0,0,4,0,0,0	2,0,2,0,0,0	0,1,0,0,0,0	4,0,0,0,0,0	0,0,4,0,0,0	2,0,2,0,0,0
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Parameters	0,0,4,0,0,0	2,2,0,0,0,0	2,0,2,0,0,0	4,1,2,0,0,0	4,0,2,0,0,0	

3. Model Performance

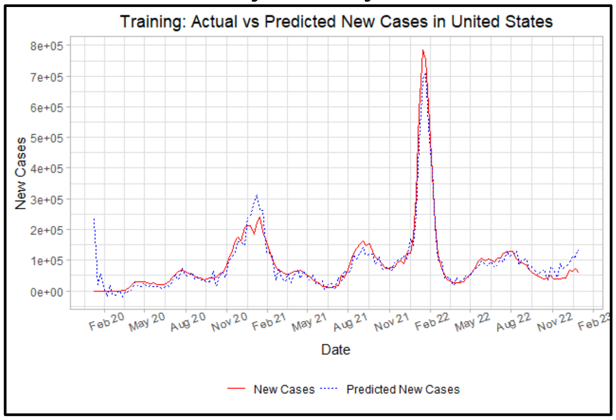
The RMSE for the AutoARIMA and log AutoARIMA models on the training and testing sets:

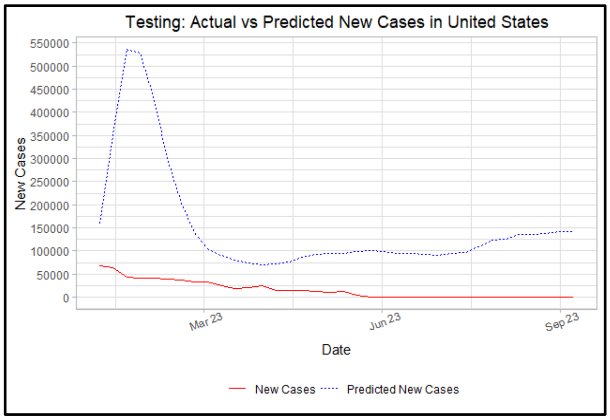
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	2789	2872	1297	1713	423	246
Test RMSE	19440	**17139**	8648	5958	1880	402
log Train RMSE	3916	13965	1674	8256	405	315

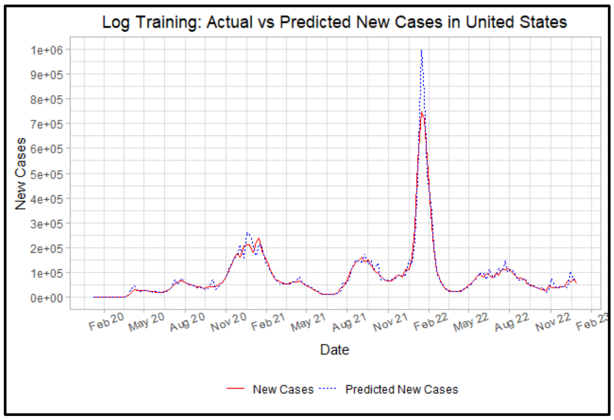
log Test RMSE	**8579**	140080	643	**1174**	1495	**37**
best ARIMA RMSE	16177	37152	**546**	4967	**1772**	113
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	7436	5883	10933	5237	10524	1981
Test RMSE	105095	112814	57545	72417	**96071**	12385
log Train RMSE	20123	7264	27470	7259	31111	2650
log Test RMSE	68979	646442	2838	8369	257184	22386
best ARIMA RMSE	**11917**	**52656**	**2736**	**2823**	123645	**4694**
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Train RMSE	361	392	1557	3163	269	1259
Test RMSE	1340	1155	5195	44662	793	3629
log Train RMSE	841	1237	2315	4658	381	2400
log Test RMSE	145	36	421	21085	139	145
best ARIMA RMSE	**23**	**33**	**341**	**3160**	**99**	**132**
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	11692	182	4354	9296	33525	
Test RMSE	**101376**	536	35506	43918	174562	
log Train RMSE	21274	490	5937	7933	27484	
log Test RMSE	987080	**4**	397	1929	16713	
best ARIMA RMSE	1022714	**4**	**1**	**1629**	**16005**	

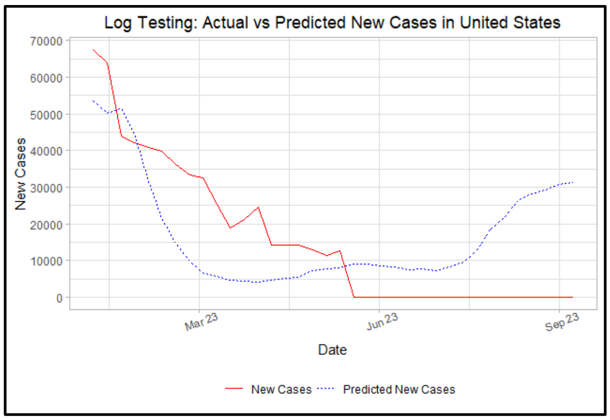
Similar to the ARIMA model, generally, the log transformed AutoARIMA model isn't as accurate on the training set, but improves the testing set's accuracy. Compared to the best test RMSE from the ARIMA model, generally AutoARIMA performs worse than the ARIMA model. A possible explanation for this result is that the seasonality for AutoARIMA is manually set which is not the most optimal seasonality as determined in the ARIMA model so the difference in performance in the model is solely due to the implementation of each model in each of their R packages. The R code for the AutoARIMA model is in Github Repository in **Models/regular/wx_autoarima.R** and **Models/log/wx_autoarima_log.R**.

<u>Plots of a Stationary Country - United States</u>

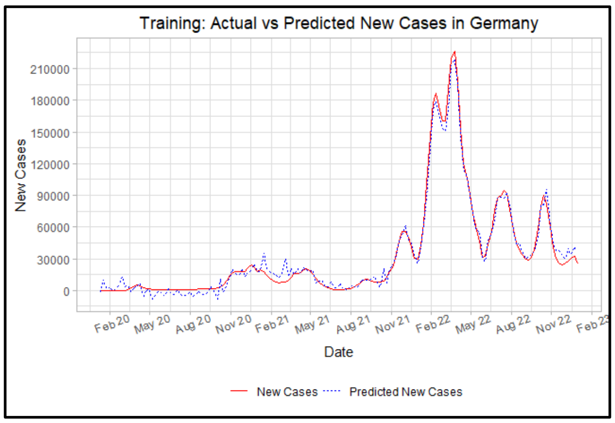


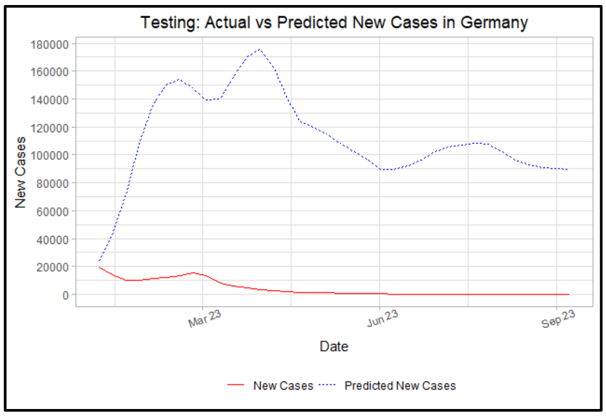


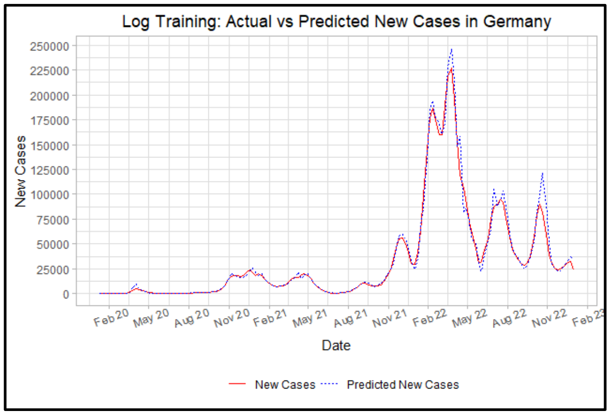


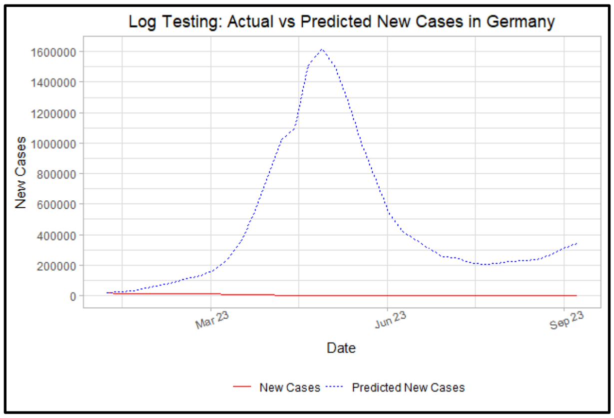


Plots of a Non-Stationary Country - Germany









C. Prophet (Univariate) Model

1. Data Preparation

Since the ARIMA model only uses one predictor, **date**, ARIMA is generally not as accurate in predicting the response as compared to Prophet which can use more than one predictor. For the Prophet (Univariate) model, the linear data is used since the Prophet model is a linear model in which there is only one predictor, **date**, for univariate. First, the data is filtered to only have observations after the first instance of COVID-19 for any country in the dataset which is January 4, 2020. Next, each country's daily data is transformed into weekly data using a weekly rolling average.

For model parameter tuning, the training set of the linear data is used to create validations sets using a rolling origin backtesting method. For each validation set, the training part is a year worth of observations per country = 23*53 observations, the testing set is 2 months worth of observations per country = 23*8 observations, and the validation sets are incremented by 4 months per country = 23*16 observations for a total of 6 validation sets. Also, Prophet (Univariate) has 3 fixed parameter and 5 parameters to be tuned with possible values:

Fixed	Tune
- growth = "linear" - season = "additive" - seasonality_weekly = TRUE	- changepoint_num = {0, 25, 50} - changepoint_range = {0.6, 0.75, 0.9} - prior_scale_changepoints = {0.001, 0.316, 100}

	- prior_scale_seasonality = {0.001, 0.316, 100} - prior_scale_holidays = {0.001, 0.316, 100}

2. Model Building

The Prophet (Univariate) model was implemented using R's *prophet* package directly following an example at R-bloggers[4]. A small issue with the prophet models at the beginning is that the model constantly returns saw-like graphs. However, this problem was due to the rolling origin backtesting validation sets were not implemented correctly for this model at the beginning since our group forgot to account for the repeated dates due to the dataset having daily observations from several countries. This issue was resolved by multiplying the lengths of each part of the validation sets by 23, the number of countries in the dataset, as noted above in Data Preparation. After applying model tuning for Prophet using a regular grid of 3 levels for each of the 5 parameters to be tuned resulting in $3^5 = 243$ parameter combinations on the 6 validation sets, the results of the best parameters that minimizes the metric RMSE are:

changepoint_num = 25 **changepoint_range** = 0.75
prior_scale_changepoints = 100 **prior_scale_seasonality** = 100
prior_scale_holidays = 0.001

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to **new_cases** and then, applied the Prophet model tuning with the best parameters that minimizes the metric RMSE are:

changepoint_num = 25 **changepoint_range** = 0.6
prior_scale_changepoints = 0.316 **prior_scale_seasonality** = 0.001
prior_scale_holidays = 0.316

3. Model Performance

The RMSE for the Prophet (Univariate) and log Prophet (Univariate) models on the training and testing sets:

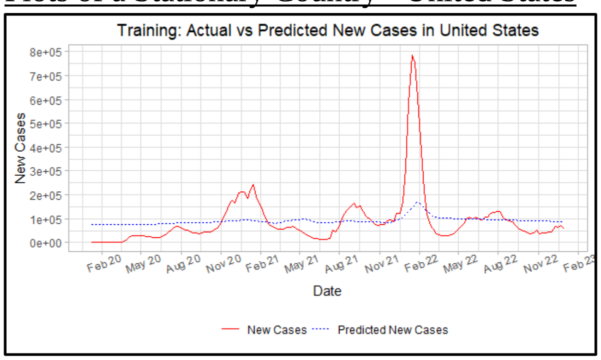
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	12162	12796	11518	14746	14536	15308
Test RMSE	4274	5091	8190	6422	10649	11066
log Train RMSE	14813	18875	5176	7621	1085	616
log Test RMSE	1408	3178	446	658	111	51
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	42687	39923	67365	20353	45607	11890

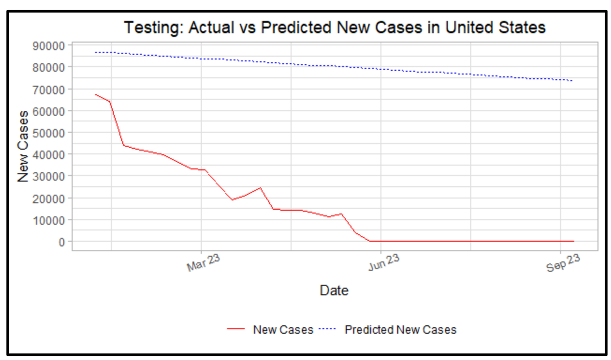
[4] Science, Business. "Introducing Modeltime: Tidy Time Series Forecasting Using Tidymodels: R-Bloggers." R-Bloggers, 29 June 2020, www.r-bloggers.com/2020/06/introducing-modeltime-tidy-time-series-forecasting-using-tidymodels/.

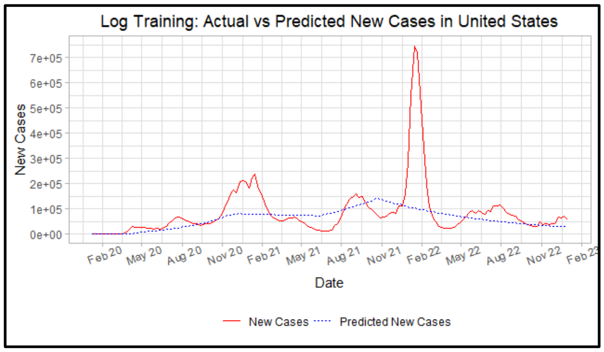
Test RMSE	20680	19346	29049	9394	36683	6405
log Train RMSE	59568	51398	75490	31564	47183	7833
log Test RMSE	2753	4712	3839	2618	40091	841
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Train RMSE	14922	14996	13735	21997	15037	15756
Test RMSE	10416	10155	8510	5311	10821	8136
log Train RMSE	1801	1700	5342	26569	1137	4878
log Test RMSE	124	192	400	3890	114	371
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	60603	15343	13284	23031	103184	
Test RMSE	17097	10913	6178	9756	66128	
log Train RMSE	65245	1002	20433	28293	101471	
log Test RMSE	21974	49	1861	1645	14828	

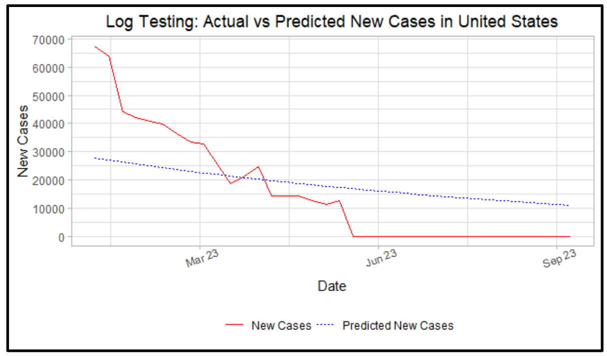
While the prophet model performs somewhat well on the untransformed **new_cases** as indicated by the somewhat low test RMSE for each country, the prophet model performs well on the log transformed **new_cases** as indicated by the low test RMSE for each country. The R code for Prophet (Univariate) is in Github Repository in **Models/regular/wx_prophet_single.R** and **Models/log/wx_prophet_single_log.R**.

<u>Plots of a Stationary Country - United States</u>

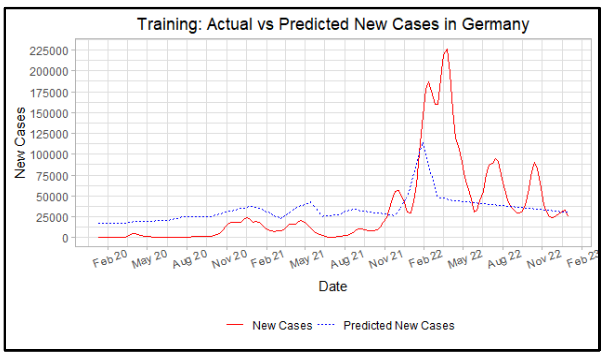


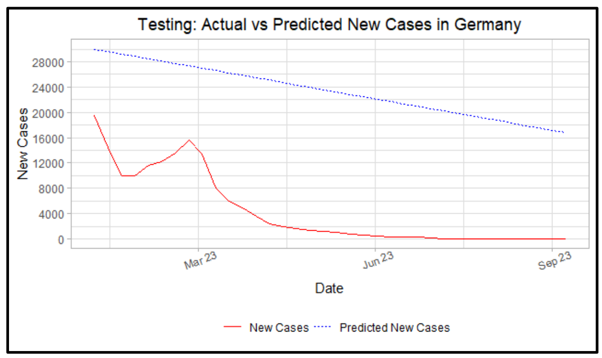


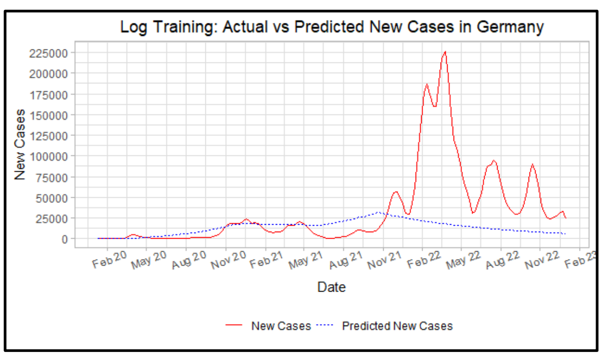


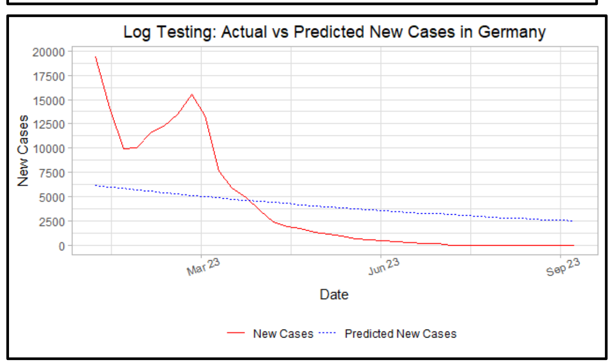


Plots of a Non-Stationary Country - Germany









D. Prophet (Multivariate) Model

1. Data Preparation

For the Prophet (Multivariate) model, the linear data is used since the Prophet model is a linear model in which all the predictors are used. First, the data is filtered to only have observations after the first instance of COVID-19 for any country in the dataset which is January 4, 2020. Next, each country's daily data is transformed into weekly data using a weekly rolling average. Additionally, any variables with high correlation are removed using R's **step_corr()** function with a critical value of 0.7. Finally, all categorical predictors are converted into dummy variables.

Similarly to Prophet (Univariate), the validations sets are constructed from the linear training set using a rolling origin backtesting method per country for a total of 6 validation sets. The Prophet (Multivariate) model has 3 fixed parameter and 5 parameters to be tuned with possible values:

Fixed	Tune
- growth = "linear" - season = "additive" - seasonality_weekly = TRUE	- changepoint_num = {0, 25, 50} - changepoint_range = {0.6, 0.75, 0.9} - prior_scale_changepoints = {0.001, 0.316, 100} - prior_scale_seasonality = {0.001, 0.316, 100} - prior_scale_holidays = {0.001, 0.316, 100}

2. Model Building

Similar to the Prophet (Univariate), the Prophet (Multivariate) model will be implemented using R's *prophet* package directly. There is still the constant issue with the prophet models constantly returning saw-like graphs in which I applied possible solutions as mentioned in Prophet (Univariate). With still this issue present, after applying model tuning for Prophet using a regular grid of 3 levels for each of the 5 parameters to be tuned resulting in $3^5 = 243$ parameter combinations on the 6 validation sets, the results of the best parameters that minimizes the metric RMSE are:

changepoint_num = 50 **changepoint_range** = 0.9
prior_scale_changepoints = 0.001 **prior_scale_seasonality** = 100
prior_scale_holidays = 0.001

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to new_cases and then, applied the Prophet model tuning with the best parameters that minimizes the metric RMSE are:

changepoint_num = 50 **changepoint_range** = 0.75
prior_scale_changepoints = 0.001 **prior_scale_seasonality** = 0.316
prior_scale_holidays = 0.001

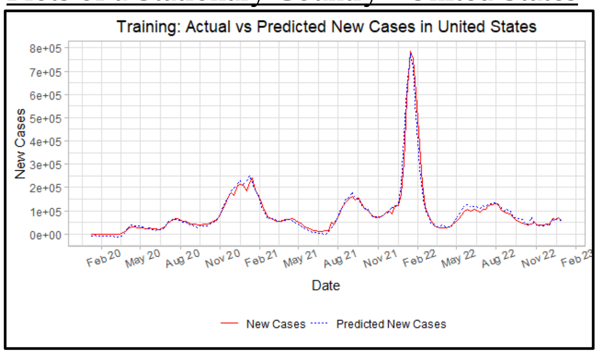
3. Model Performance

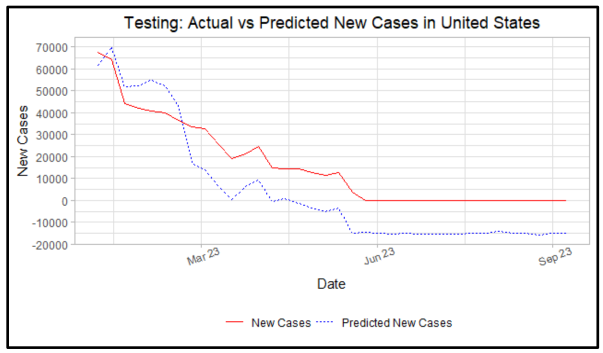
The RMSE for the Prophet (Multivariate) and log Prophet (Multivariate) models on the training and testing sets as well as the best Test RMSE for Prophet (Univariate) which is the Test RMSE for the log transformed Prophet (Univariate):

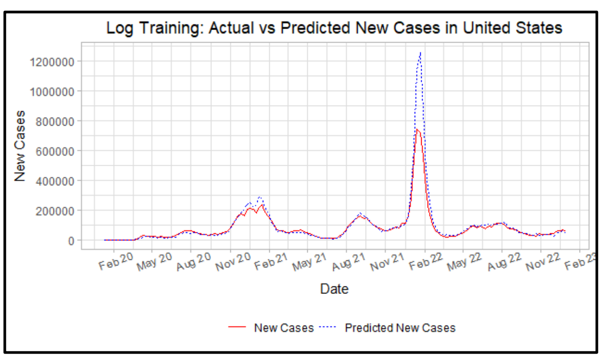
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	2927	2231	1898	1841	1528	1836
Test RMSE	3250	7145	3535	2813	3023	2381
log Train RMSE	2536	7344	1446	1241	511	244
log Test RMSE	**722**	**2220**	**373**	**84**	**48**	**12**
best Uni RMSE	5502	6351	4840	6237	4670	5957
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	6080	4784	10264	5860	12494	3161
Test RMSE	9746	8326	3127	4364	15119	2719
log Train RMSE	11370	9741	9205	4719	18593	2208
log Test RMSE	6136	7455	**683**	**2900**	**4719**	**465**
best Uni RMSE	**5808**	**7147**	5385	4836	40213	3181
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Train RMSE	1887	1685	1969	4999	1706	1751
Test RMSE	2959	3163	3106	2535	2992	2283
log Train RMSE	408	437	1276	5786	288	1176
log Test RMSE	**8**	**22**	**361**	**1391**	**32**	**97**
best Uni RMSE	4481	4162	2912	6077	2910	3508
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	11574	1652	3011	7249	18287	
Test RMSE	13515	2906	5533	3490	14778	
log Train RMSE	9071	312	3216	7092	62245	
log Test RMSE	21803	**5**	**4**	**1676**	**10949**	
best Uni RMSE	**21411**	3358	2434	2790	24392	

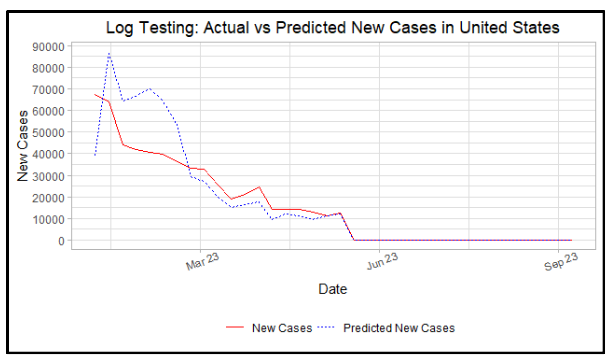
For the Prophet (Multivariate), it appears that the non transformed performs somewhat well as indicated by the somewhat low test RMSE, but the log transformed Prophet (Multivariate) is one of the best performing models as indicated by the extremely low test RMSE even for countries like the United States and Japan. In fact, the log transformed Prophet (Multivariate) basically outperforms the log transformed Prophet (Univariate) as indicated by the lower test RMSE in the log test RMSE compared to the Prophet (Univariate)'s lowest test RMSE. This is most likely due to the Prophet (Multivariate) having additional information such as country data like **gdp_per_capita** and **population_density** and hospital data like **icu_patients** and **total_tests** which helped it make better predictions compared to Prophet (Univariate) which only had information based on just **date**. The R code for the Prophet (Multivariate) model is in Github Repository in **Models/regular/wx_prophet_multiple.R** and **Models/log/wx_prophet_multiple_log.R**.

<u>Plots of a Stationary Country - United States</u>

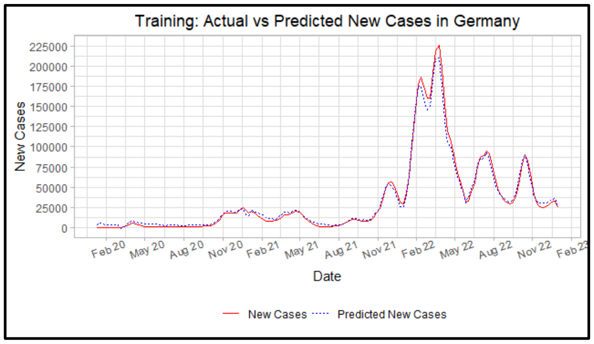


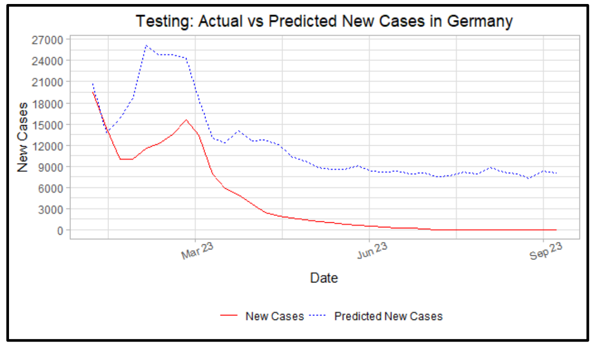


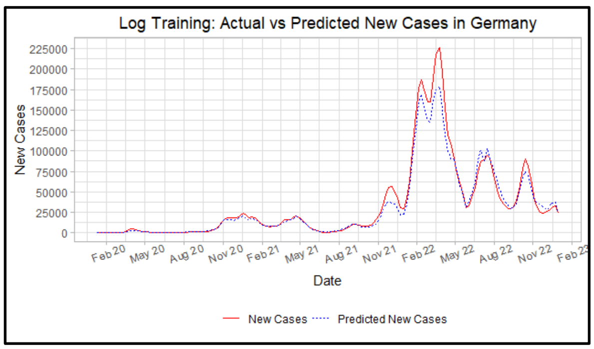


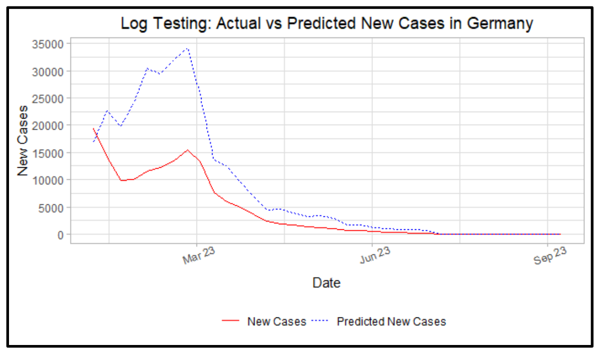


<u>Plots of a Non-Stationary Country - Germany</u>









E. XGBoost Model

1. Data Preparation

Using the tree-based dataset, first the data is filtered to only have observations after the first instance of COVID-19 for any country in the dataset which is January 4, 2020. Next, 3 new predictors are created: 1 week lagged variable for **new_cases**, 2 week lagged variable for **new_cases**, and 1 month lagged variable for **new_cases**. Finally, all categorical predictors are converted into dummy variables.

For model parameter tuning, the training set of the linear data is used to create validations sets using a rolling origin backtesting method. For each validation set, the training part is a year worth of observations per country = 23*366 observations, the testing set is 2 months worth of observations per country = 23*60 observations, and the validation sets are incremented by 4 months per country = 23*120 observations for a total of 6 validation sets. For the XGBoost model, no parameters are fixed and 5 parameters are to be tuned with possible values:

Fixed	Tune
	- trees = {500, 1000, 1500} - tree_depth = {2, 11, 20} - learn_rate= {0.001, 0.0178, 0.316} - min_n = {5, 10, 15} - mtry = {5, 15, 25}

2. Model Building

The XGBoost model was implemented using R's *xgboost* package directly using an example[5] by a Data Scientist for Posit, Julia Silge. An issue with the XGBoost model was that it was performing very poorly without information based on time. Thus, a solution implemented was to create several lagged variables of the response **new_cases** which vastly improved the performance of the XGBoost model. Applying model tuning for XGBoost using a regular grid of 3 levels for each of the 5 parameters to be tuned resulting in $3^5 = 243$ parameter combinations on the 6 validation sets, the results of the best parameters that minimizes the metric RMSE are:

trees = 1000 **tree_depth** = 2 **learn_rate** = 0.316
min_n = 10 **mtry** = 25

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to **new_cases** and then, applied the XGBoost model tuning with the best parameters that minimizes the metric RMSE are:

trees = 1000 **tree_depth** = 20 **learn_rate** = 0.0178
min_n = 10 **mtry** = 25

3. Model Performance

The RMSE for the XGBoost and log XGBoost models on the training and testing sets:

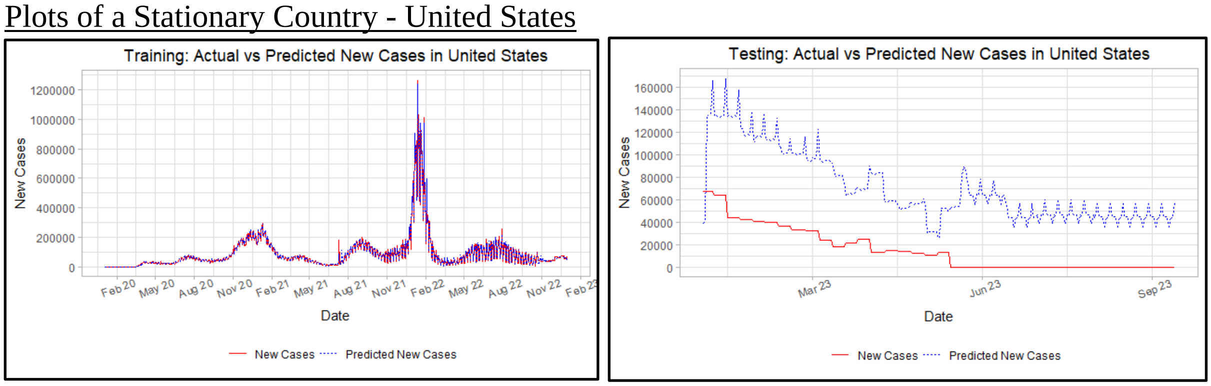
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	2564	2223	1549	1773	964	629
Test RMSE	5354	12856	6389	4100	2416	2056
log Train RMSE	91	59	30	35	17	4

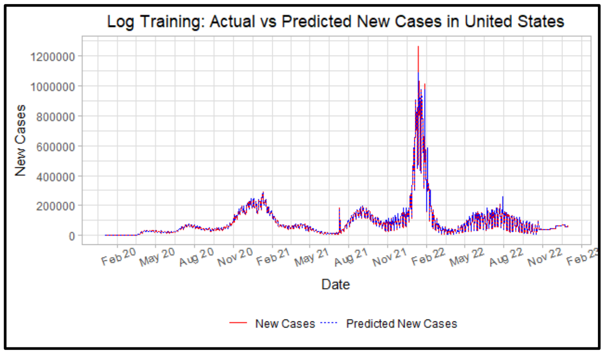
[5] Silge, Julia. "Tune XGBoost with Tidymodels and #tidytuesday Beach Volleyball." Julia Silge Blog, 21 May 2020, juliasilge.com/blog/xgboost-tune-volleyball/.

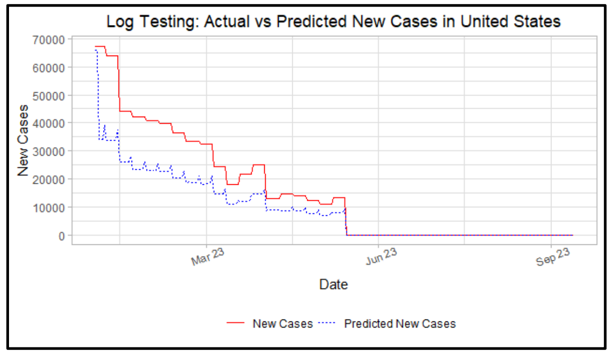
log Test RMSE	292	1611	279	44	47	2
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	4745	3744	4691	4898	6234	2190
Test RMSE	18441	22620	10799	14501	64125	9010
log Train RMSE	174	164	416	129	469	56
log Test RMSE	1358	2486	834	1407	20480	518
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Train RMSE	849	882	1358	2426	580	1786
Test RMSE	2319	2098	4579	13588	2039	2131
log Train RMSE	9	10	40	118	6	66
log Test RMSE	7	3	198	1970	17	52
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	4410	732	3011	4207	9378	
Test RMSE	56720	1555	1243	6367	58104	
log Train RMSE	795	5	99	124	6059	
log Test RMSE	18173	1	1	612	10273	

The XGBoost model without log transformation is one of the better performing models in predicting new cases of COVID as indicated by the low test RMSE relative to the other models mentioned above thus far. In fact, XGBoost with a log transformation is the best performing model compared to every other model thus far except for some of the Prophet models for a few countries. The R code for XGBoost model is in Github Repository in **Models/regular/wx_xgboost.R** and **Models/log/wx_xgboost_log.R**.

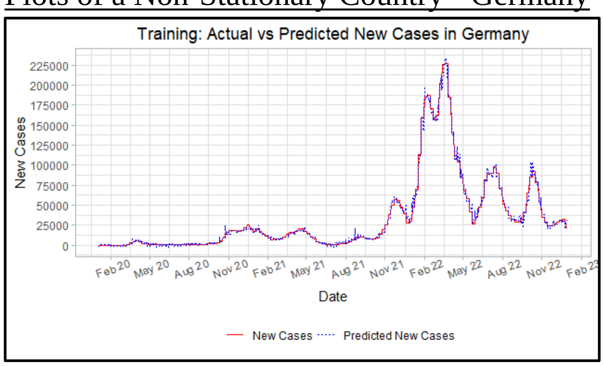
<u>Plots of a Stationary Country - United States</u>

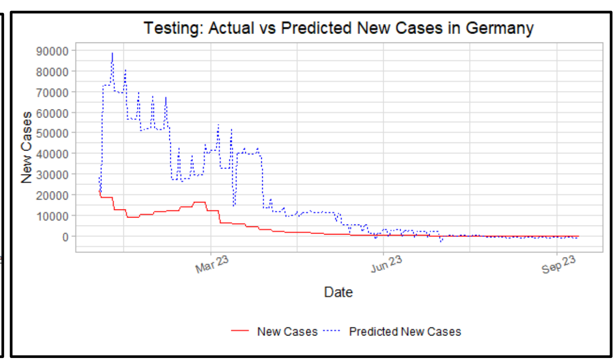


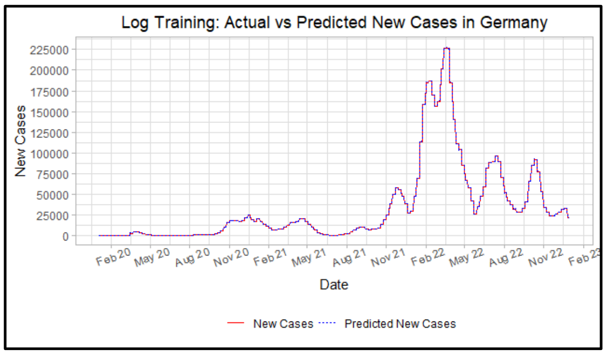


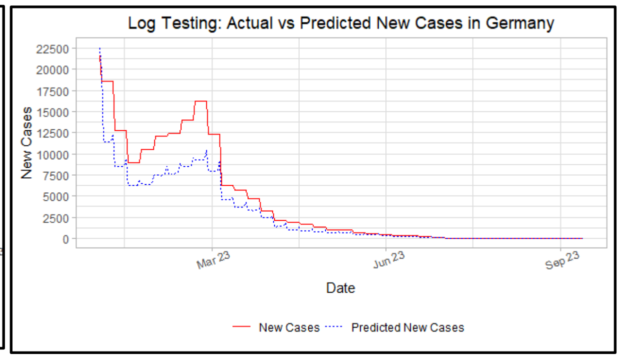


Plots of a Non-Stationary Country - Germany









F. LSTM Model

1. Data Preparation

For the LSTM model, the neural network dataset is used since LSTM is a recurrent Neural Network. First, the data is filtered to only have observations after the first instance of COVID-19 for any country in the dataset which is January 4, 2020. Next, I selected 2 predictors, **new_deaths** and **new_tests**, which have high correlation with the response **new_cases**, as indicated by the high correlation coefficient in Table 2 and 4 in the Appendices. Additionally, geographic and country data predictors are added which includes **location**, **gdp_per_capita**, and **new_tests_b**, the missingness indicator for new_tests, for a total of 5 predictors. All the predictors are normalized and new_tests_b is converted into a dummy variable.

2. Model Building

The LSTM model was implemented using R's **keras** package directly using an example[6]. First, a lagged predictor is created for the response new_cases, which I chose to be a lag of 7 for one week. Then, the lagged predictor and the 5 normalized predictors mentioned previously are merged into a 3d array. Next, a LSTM model was built with an initial layer of 5 **units**, in which the number of units will be a parameter tuned, along with fixed parameters **return_sequences** = TRUE and **stateful** = TRUE. A dropout layer is also added to lessen overfitting along with another LSTM layer with 5 units and finally, a dense layer to the output of the LSTM. Then, the LSTM model is compiled with a mean squared error loss function and using the Adaptive Moment Estimation, "adam", optimizer with the mean absolute error metric. Finally, the LSTM model is trained with the predictors mentioned previously converted into a 3d array with the number of **epochs** being a parameter tuned.

Fixed	Tune
- return_sequence = TRUE - stateful = TRUE	- units = {20, 30} - epochs = {10, 20}

The main challenge was finding out how to implement the LSTM model as the format in constructing the model and preparing the data was vastly different from classical machine learning models. I had to search through several examples to get somewhat of a grasp on how to prepare the data such as transforming it into a 3d array and determining how to set up the LSTM inputs such that it can use the predictors converted into a 3d array. Another challenge was that I was unable to utilize all predictors for the LSTM model since I had to normalize and convert each predictor into the 3d array so utilizing all predictors became difficult to implement and time consuming. Thus, only the 5 predictors mentioned above along with the 7 lagged **new_cases** predictor were the only predictors used to build the LSTM model. The best parameters that minimizes the metric RMSE are:

units = 30 **epochs** = 10

Additionally, in order to improve performance by lowering the metric RMSE, I applied a log transformation to **new_cases** and then, applied the LSTM model tuning with the best parameters that minimizes the metric RMSE are:

units = 20 **epochs** = 10

3. Model Performance

The RMSE for the LSTM and log LSTM models on the training and testing sets:

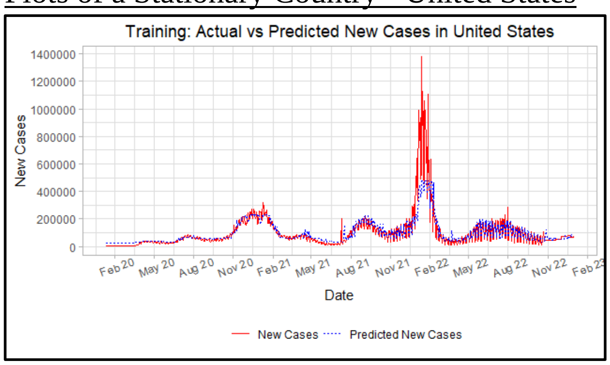
Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
Train RMSE	19497	22004	20217	18153	21913	22702
Test RMSE	21185	19880	21164	21663	21788	21964
log Train RMSE	9531	20014	19519	3895	1954	580

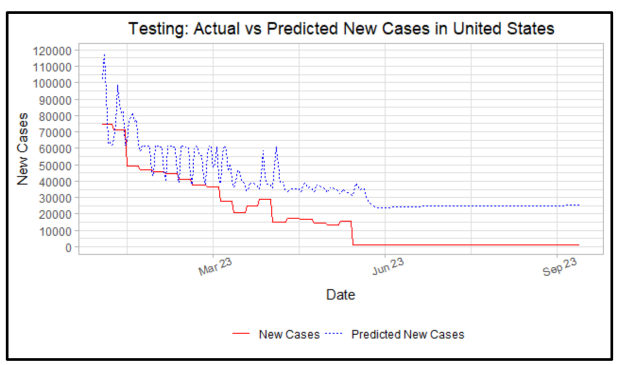
[6] "LSTM TIME SERIES PREDICTION IN R." Data Side of Life, 5 Jan. 2020, datasideoflife.com/?p=1171.

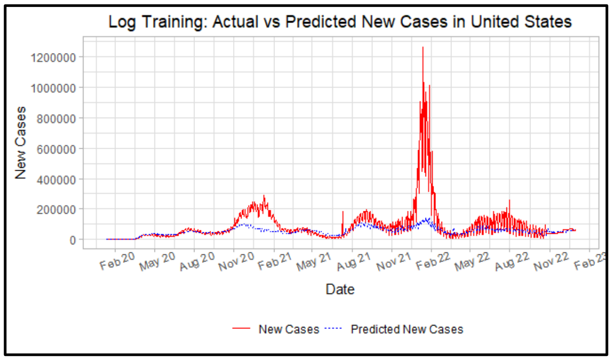
log Test RMSE	5359	21503	5223	231	144	37
Country	France	Germany	India	Italy	Japan	Mexico
Train RMSE	23915	18664	32535	22195	28535	16868
Test RMSE	19454	20202	21119	19960	28268	18992
log Train RMSE	47011	43566	57977	20620	35268	5653
log Test RMSE	15998	21472	1215	8663	29071	1511
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
Train RMSE	19456	19358	18087	16524	20327	18314
Test RMSE	20654	20707	20316	16515	20911	20946
log Train RMSE	1544	1251	3525	18495	5959	3428
log Test RMSE	48	53	619	3689	1071	363
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
Train RMSE	31106	20344	15580	20914	72201	
Test RMSE	25929	20588	20667	18839	21919	
log Train RMSE	45348	1782	10598	17104	115006	
log Test RMSE	25452	29	11	3970	13476	

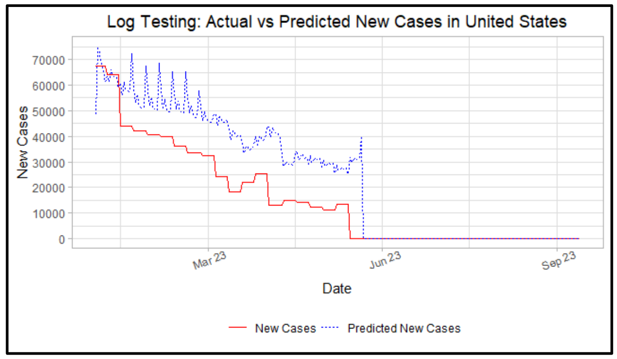
In general, the non-transformed LSTM model is one of the worst performing models alongside the non-transformed ARIMA and AutoARIMA models as indicated by the large train and test RMSE. However, the log transformation LSTM model is one of the best performing models as indicated by the low test RMSE. The R code for the LSTM model is in Github Repository in **Models/regular/wx_lstm.R** and **Models/log/wx_lstm_log.R**.

<u>Plots of a Stationary Country - United States</u>

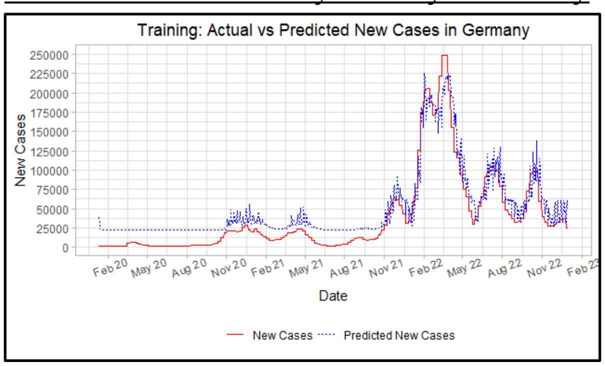


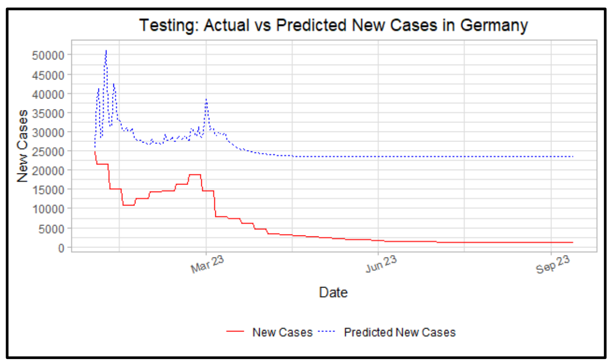


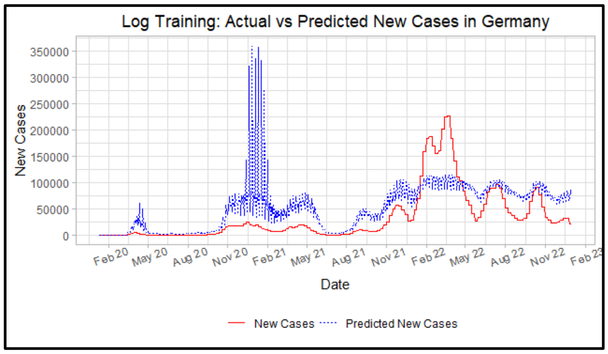


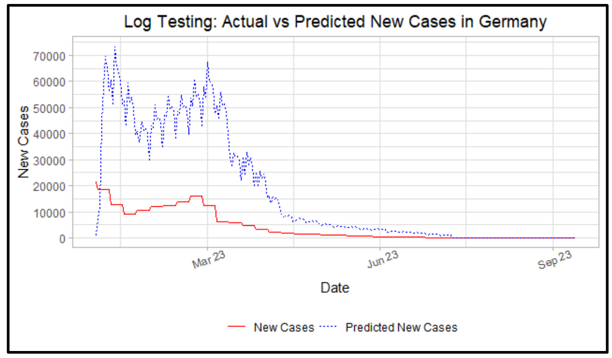


Plots of a Non-Stationary Country - Germany









IV. Results

A. Model Performance Comparison

A table of each country's test RMSE for each model and their log transformed version:

Country	Argentina	Australia	Canada	Colombia	Ecuador	Ethiopia
ARIMA	16177	37152	7372	6219	1772	332
AutoARIMA	19440	17139	8648	5958	1880	402
Prophet (Uni)	4274	5091	8190	6422	10649	11066
Prophet (Multi)	3250	7145	3535	2813	3023	2381
XGBoost	5354	12856	6389	4100	2416	2056
LSTM	21185	19880	21164	21663	21788	21964
log ARIMA	43006	52424	546	4967	4967	113
log AutoARIMA	8579	140080	643	1174	1495	37
log Prophet (Uni)	1408	3178	446	658	111	51
log Prophet (Multi)	722	2220	373	84	48	12
log XGBoost	292	1611	279	44	47	2

log LSTM	5359	21503	5223	231	144	37
Country	France	Germany	India	Italy	Japan	Mexico
ARIMA	96537	95225	53727	32833	123645	8437
AutoARIMA	105095	112814	57545	72417	96071	12385
Prophet (Uni)	20680	19346	29049	9394	36683	6405
Prophet (Multi)	9746	8326	3127	4364	15119	2719
XGBoost	18441	22620	10799	14501	64125	9010
LSTM	19454	20202	21119	19960	28268	18992
log ARIMA	11917	52656	2736	2823	134905	4694
log AutoARIMA	68979	646442	2838	8369	257184	22386
log Prophet (Uni)	2753	4712	3839	2618	40091	841
log Prophet (Multi)	6136	7455	683	2900	4719	465
log XGBoost	1358	2486	834	1407	20480	518
log LSTM	15998	21472	1215	8663	29071	1511
Country	Morocco	Pakistan	Philippines	Russia	Saudi Arabia	South Africa
ARIMA	1442	1012	4436	22070	1478	3380
AutoARIMA	1340	1155	5195	44662	793	3629
Prophet (Uni)	10416	10155	8510	5311	10821	8136
Prophet (Multi)	2959	3163	3106	2535	2992	2283
XGBoost	2319	2098	4579	13588	2039	2131
LSTM	20654	20707	20316	16515	20911	20946
log ARIMA	23	33	341	3160	99	132
log AutoARIMA	145	36	421	21085	139	145
log Prophet (Uni)	124	192	400	3890	114	371
log Prophet (Multi)	8	22	361	1391	32	97
log XGBoost	7	3	198	1970	17	52
log LSTM	48	53	619	3689	1071	363
Country	South Korea	Sri Lanka	Turkey	United Kingdom	United States	
ARIMA	101029	487	34135	47814	126535	
AutoARIMA	101376	536	35506	43918	174562	

Prophet (Uni)	17097	10913	6178	9756	66128	
Prophet (Multi)	13515	2906	5533	3490	14778	
XGBoost	56720	1555	1243	6367	58104	
LSTM	25929	20588	20667	18839	21919	
log ARIMA	1022714	4	1	1629	16005	
log AutoARIMA	987080	4	397	1929	16713	
log Prophet (Uni)	21974	49	1861	1645	14828	
log Prophet (Multi)	21803	5	4	1676	10949	
log XGBoost	18173	1	1	612	10273	
log LSTM	25452	29	11	3970	13476	

In general, the best performing model in predicting new cases of COVID-19 by country level is the log transformed XGBoost model as indicated by the majority of the best test RMSE in red highlight in the log XGBoost row. Additionally, for a few countries, the log transformed Prophet (Multivariate) model performs as well as or even sometimes outperforms the log transformed XGBoost Model as indicated by the highlighted oranges test RMSE.

B. Analysis of Results

Model	Strengths	Weaknesses
ARIMA	- Easy to implement - Able to easily use past data to predict new data	- Does not handle non-linear data well - Does not handle outliers well - Only one predictor = Date → Results not very accurate - Manually rerun model to apply to each country - Model tuning and training is somewhat slow
AutoARIMA	- Easy to implement - Able to easily use past data to predict new data - Model tuning and training is quick	- Does not handle non-linear data well - Does not handle outliers well - Only one predictor = Date → Results not very accurate - Manually rerun model to apply to each country
Prophet (Univariate)	- Apply model to every country at once	- One one predictor = Date → Results not very accurate - Model tuning and training is slow
Prophet (Multivariate)	- Easy to implement - Apply model to every	- Can tend to overfit the training data → Results are not as accurate on the

	country at once - Apply time series data easier with multivariate regressors	testing data - Model tuning and training is very slow
XGBoost	- Apply model to every country at once - Can accurately predict the general trend and outliers such as peaks - Handle missingness well using large imputations	- Can tend to overfit the training data → Results are not as accurate on the testing data - Model tuning and training is very slow
LSTM	- Apply model to every country at once - Can accurately predict the general trend - Handle missingness well using indicator predictors	- Hard to implement due to different format for inputs and different way to train model - Can sometimes handle outliers / peaks - Model tuning and training is slow

I believe that the XGBoost model performed the best because (1) it was able to use all the predictors available compared to some models like ARIMA and Prophet (Univariate), (2) it was able to handle missingness in the data very well since such missingness is imputed with a large value (i.e. 10^{15}), and (3) most models can predict the general trend, but not outliers like large peaks, but XGBoost are able to predict both the trend and large peaks.

V. Discussion

- The LSTM model is not as accurate in predicting COVID-19 cases as the other models since it isn't a classical machine learning model and is implemented with more complexity. Due to my lack of experience and knowledge on how neural networks work especially for recurrent neural networks, I am unable to determine even the issues that my LSTM model might be having. I can only visualize why the model is not performing well, but have no understanding on why the LSTM is performing the way it is.
- During my time in this class and working on this project, I think I gain some valuable experience in learning new types of machine learning models like Prophet and LSTM, but I think the amount of confusion and time spent on learning how to implement and constantly fixing the model such that it can work property was too time consuming.
- I liked how the project is structured using version control and weekly reports since it provides a clear way to communicate to group members while staying organized and the weekly reports gave me a concrete way to set personal deadlines and address issues in order to maintain progress on the project. I disliked how messy the dataset was such that I kept finding more and more issues with the dataset that isn't as obvious as missingness or collinearity which meant everytime I fixed the dataset, I had to rerun all my models again which was very frustrating and time consuming.

VI. Conclusion

- The best model to predict new cases of COVID-19 by country level is a log transformed XGBoost model followed by a log transformed Prophet (Multivariate) model.
- The worst models to predict new cases of COVID-19 by country level are the non-transformed ARIMA and AutoARIMA models.
- In general, log transformation on new cases of COVID-19 helps to improve a model's accuracy in predicting new cases of COVID-19.
- Consistent communication is an essential key part in a collaborative learning environment in order to address issues and provide solutions, maintain deadlines, and clear up misunderstandings about machine learning models and processes.
- It is also important to be organized and clear when making contributions to a group project such that member's tasks are clearly understood and beneficial in furthering the progress of the project rather than being stuck, disorganized, and fixing errors.

VII. References

1. Dataset Source: Our World in Data. (2023). Our World in Data - COVID-19 [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/6559049
2. "K-Means Cluster Analysis." K-Means Cluster Analysis · UC Business Analytics R Programming Guide, University of Cincinnati, uc-r.github.io/kmeans_clustering. Accessed 26 Nov. 2023.
3. Science, Business. "Introducing Modeltime: Tidy Time Series Forecasting Using Tidymodels: R-Bloggers." R-Bloggers, 29 June 2020, www.r-bloggers.com/2020/06/introducing-modeltime-tidy-time-series-forecasting-using-tidymodels/.
4. Silge, Julia. "Tune XGBoost with Tidymodels and #tidytuesday Beach Volleyball." Julia Silge Blog, 21 May 2020, juliasilge.com/blog/xgboost-tune-volleyball/.
5. "LSTM TIME SERIES PREDICTION IN R." Data Side of Life, 5 Jan. 2020, datasideoflife.com/?p=1171.

VIII. Appendices

- Source 1: Data Science Project Github Repository https://github.com/AzureAmber/STAT-390-Covid-Project/tree/main

- Table 1: Data Missingness

skim_type	skim_variable	n_missing	complete_rate
numeric	excess_mortality_cumulative_absolute	28693	7.64%
numeric	excess_mortality_cumulative	28693	7.64%
numeric	excess_mortality	28693	7.64%

numeric	excess_mortality_cumulative_per_million	28693	7.64%
numeric	weekly_icu_admissions	28193	9.25%
numeric	weekly_icu_admissions_per_million	28193	9.25%
numeric	weekly_hosp_admissions	24846	20.02%
numeric	weekly_hosp_admissions_per_million	24846	20.02%
numeric	hosp_patients	22647	27.10%
numeric	hosp_patients_per_million	22647	27.10%
numeric	total_boosters	21913	29.46%
numeric	total_boosters_per_hundred	21913	29.46%
numeric	handwashing_facilities	20258	34.79%
numeric	icu_patients	20250	34.82%
numeric	icu_patients_per_million	20250	34.82%
numeric	new_vaccinations	18229	41.32%
numeric	people_fully_vaccinated	17978	42.13%
numeric	people_fully_vaccinated_per_hundred	17978	42.13%
numeric	people_vaccinated	17661	43.15%
numeric	people_vaccinated_per_hundred	17661	43.15%
numeric	total_vaccinations	17191	44.66%
numeric	total_vaccinations_per_hundred	17191	44.66%
numeric	new_tests	14004	54.92%
numeric	new_tests_per_thousand	14004	54.92%
numeric	total_tests	13518	56.49%
numeric	total_tests_per_thousand	13518	56.49%
numeric	tests_per_case	13137	57.71%
numeric	positive_rate	13061	57.96%
numeric	new_people_vaccinated_smoothed	12484	59.81%
numeric	new_people_vaccinated_smoothed_per_hundred	12484	59.81%
numeric	new_vaccinations_smoothed	12408	60.06%
numeric	new_vaccinations_smoothed_per_million	12408	60.06%
numeric	new_tests_smoothed	12064	61.17%
numeric	new_tests_smoothed_per_thousand	12064	61.17%

character	tests_units	11903	61.68%
numeric	reproduction_rate	7474	75.94%
numeric	extreme_poverty	6750	78.27%
numeric	stringency_index	5900	81.01%
numeric	total_deaths	1495	95.19%
numeric	total_deaths_per_million	1495	95.19%
numeric	total_cases	911	97.07%
numeric	total_cases_per_million	911	97.07%
numeric	new_cases_smoothed	284	99.09%
numeric	new_cases_smoothed_per_million	284	99.09%
numeric	new_deaths_smoothed	259	99.17%
numeric	new_deaths_smoothed_per_million	259	99.17%
numeric	new_cases	161	99.48%
numeric	new_cases_per_million	161	99.48%
numeric	new_deaths	144	99.54%
numeric	new_deaths_per_million	144	99.54%
Date	date	0	100.00%
character	iso_code	0	100.00%
character	continent	0	100.00%
character	location	0	100.00%
character	day_of_week	0	100.00%
factor	month	0	100.00%
logical	G20	0	100.00%
logical	G24	0	100.00%
numeric	population_density	0	100.00%
numeric	median_age	0	100.00%
numeric	aged_65_older	0	100.00%
numeric	aged_70_older	0	100.00%
numeric	gdp_per_capita	0	100.00%
numeric	cardiovasc_death_rate	0	100.00%
numeric	diabetes_prevalence	0	100.00%

numeric	female_smokers	0	100.00%
numeric	male_smokers	0	100.00%
numeric	hospital_beds_per_thousand	0	100.00%
numeric	life_expectancy	0	100.00%
numeric	human_development_index	0	100.00%
numeric	population	0	100.00%

- Table 2: Correlation between new_cases and variables with large missingness

	new_cases
hosp_patients	0.51
weekly_hosp_admissions	0.45
new_tests	0.39
icu_patients	0.38
people_vaccinated	0.38
people_fully_vaccinated	0.37
total_vaccinations	0.36
new_vaccinations_smoothed	0.33
total_tests	0.32
total_boosters	0.30
new_vaccinations	0.29
new_people_vaccinated_smoothed	0.24
weekly_hosp_admissions_per_million	0.22
weekly_icu_admissions	0.19
positive_rate	0.14
excess_mortality_cumulative_absolute	0.13
hosp_patients_per_million	0.12
icu_patients_per_million	0.11
weekly_icu_admissions_per_million	0.08
new_tests_smoothed	0.06
new_tests_smoothed_per_thousand	0.06
total_vaccinations_per_hundred	0.06

total_tests_per_thousand	0.05
people_vaccinated_per_hundred	0.05
people_fully_vaccinated_per_hundred	0.05
new_tests_per_thousand	0.04
handwashing_facilities	0.02
new_vaccinations_smoothed_per_million	0.01
excess_mortality	0.01
excess_mortality_cumulative_per_million	0.00
new_people_vaccinated_smoothed_per_hundred	-0.01
tests_per_case	-0.01
excess_mortality_cumulative	-0.01
total_boosters_per_hundred	-0.03

- Table 3: Correlation Matrix for Multicollinearity

	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	new_cases_smoothed_per_million	median_age	aged_65_older	aged_70_older	gdp_per_capita	extreme_poverty	female_smokers	male_smokers	life_expectancy	human_development_index	population
total_cases	1.00	0.37	0.44	0.94	0.30	0.33	0.00	0.05	0.04	0.04	0.02	-0.04	0.00	0.00	0.02	0.05	0.65
new_cases	0.37	1.00	0.84	0.43	0.50	0.43	0.07	0.03	0.03	0.03	0.01	-0.03	0.00	0.01	0.02	0.03	0.37
new_cases_smoothed	0.44	0.84	1.00	0.50	0.48	0.51	0.08	0.04	0.03	0.03	0.02	-0.03	0.00	0.01	0.02	0.03	0.44
total_deaths	0.94	0.43	0.50	1.00	0.47	0.52	0.00	0.04	0.03	0.03	0.01	-0.05	-0.01	0.00	0.01	0.04	0.74
new_deaths	0.30	0.50	0.48	0.47	1.00	0.91	0.02	0.03	0.03	0.03	0.01	-0.04	-0.01	0.00	0.01	0.04	0.58
new_deaths_smoothed	0.33	0.43	0.51	0.52	0.91	1.00	0.02	0.04	0.03	0.03	0.01	-0.04	-0.01	0.00	0.01	0.04	0.64
total_cases_per_million	0.07	0.00	0.00	0.02	-0.04	-0.05	0.19	0.48	0.46	0.46	0.41	-0.35	0.41	0.06	0.44	0.50	-0.07
new_cases_per_million	0.00	0.09	0.04	0.00	0.03	0.01	0.52	0.14	0.14	0.13	0.11	-0.09	0.10	0.01	0.09	0.14	-0.01
new_cases_smoothed_per_million	0.00	0.07	0.08	0.00	0.02	0.02	1.00	0.25	0.25	0.25	0.19	-0.16	0.18	0.02	0.17	0.24	-0.02
total_deaths_per_million	0.10	0.02	0.03	0.11	0.01	0.01	0.11	0.49	0.48	0.48	0.19	-0.39	0.45	0.12	0.38	0.44	-0.06
new_deaths_per_million	-0.01	0.04	0.02	0.00	0.11	0.06	0.14	0.15	0.15	0.15	0.06	-0.12	0.13	0.03	0.09	0.13	-0.01

new_deaths_smoothed_per_million	-0.01	0.03	0.04	0.00	0.10	0.11	0.24	0.27	0.27	0.27	0.11	-0.21	0.24	0.06	0.17	0.24	-0.02
reproduction_rate	0.02	0.02	0.02	0.03	0.03	0.03	0.09	0.25	0.22	0.22	0.17	-0.21	0.14	0.00	0.25	0.24	0.05
stringency_index	-0.11	0.02	0.02	-0.05	0.14	0.16	-0.01	0.04	-0.01	-0.01	0.01	-0.11	-0.08	0.03	0.06	0.07	0.12
population_density	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	0.05	0.15	0.07	0.04	0.37	-0.03	-0.05	0.00	0.22	0.18	-0.02
median_age	0.05	0.03	0.04	0.04	0.03	0.04	0.25	1.00	0.91	0.90	0.65	-0.70	0.64	0.17	0.83	0.90	0.01
aged_65_older	0.04	0.03	0.03	0.03	0.03	0.03	0.25	0.91	1.00	0.99	0.50	-0.57	0.73	0.08	0.73	0.78	0.00
aged_70_older	0.04	0.03	0.03	0.03	0.03	0.03	0.25	0.90	0.99	1.00	0.49	-0.56	0.73	0.08	0.71	0.77	-0.01
gdp_per_capita	0.02	0.01	0.02	0.01	0.01	0.01	0.19	0.65	0.50	0.49	1.00	-0.50	0.31	-0.10	0.68	0.75	-0.03
extreme_poverty	-0.04	-0.03	-0.03	-0.05	-0.04	-0.04	-0.16	-0.70	-0.57	-0.56	-0.50	1.00	-0.41	-0.19	-0.75	-0.78	-0.03
cardiovasc_death_rate	-0.04	-0.03	-0.03	-0.04	-0.04	-0.04	-0.13	-0.34	-0.34	-0.36	-0.48	0.19	-0.14	0.43	-0.47	-0.42	-0.02
diabetes_prevalence	0.00	0.00	0.00	0.01	0.00	0.00	0.04	0.14	-0.05	-0.08	0.12	-0.38	0.09	0.20	0.20	0.20	0.00
female_smokers	0.00	0.00	0.00	-0.01	-0.01	-0.01	0.18	0.64	0.73	0.73	0.31	-0.41	1.00	0.23	0.44	0.55	-0.06
male_smokers	0.00	0.01	0.01	0.00	0.00	0.00	0.02	0.17	0.08	0.08	-0.10	-0.19	0.23	1.00	0.05	0.09	0.01
hospital_beds_per_thousand	0.02	0.01	0.02	0.00	0.00	0.00	0.13	0.63	0.60	0.60	0.29	-0.44	0.47	0.35	0.42	0.56	-0.02
life_expectancy	0.02	0.02	0.02	0.01	0.01	0.01	0.17	0.83	0.73	0.71	0.68	-0.75	0.44	0.05	1.00	0.91	-0.02
human_development_index	0.05	0.03	0.03	0.04	0.04	0.04	0.24	0.90	0.78	0.77	0.75	-0.78	0.55	0.09	0.91	1.00	0.00
population	0.65	0.37	0.44	0.74	0.58	0.64	-0.02	0.01	0.00	-0.01	-0.03	-0.03	-0.06	0.01	-0.02	0.00	1.00

- Table 4: Initial Stationary Check for ARIMA by ADF Values

Country	ADF	P-value	Result
Argentina	-4.21	0.01	Stationary
Australia	-2.02	0.57	Non-Stationary
Canada	-3.25	0.08	Non-Stationary
Colombia	-3.47	0.05	Stationary
Ecuador	-4.68	0.01	Stationary
Ethiopia	-4.19	0.01	Stationary

France	-2.45	0.39	Non-Stationary
Germany	-2.52	0.36	Non-Stationary
India	-3.78	0.02	Stationary
Italy	-2.73	0.27	Non-Stationary
Japan	-4.32	0.01	Stationary
Mexico	-4.47	0.01	Stationary
Morocco	-4.13	0.01	Stationary
Pakistan	-4.25	0.01	Stationary
Philippines	-3.59	0.04	Stationary
Russia	-3.40	0.06	Non-Stationary
Saudi Arabia	-3.85	0.02	Stationary
South Africa	-4.07	0.01	Stationary
South Korea	-3.46	0.05	Stationary
Sri Lanka	-2.83	0.23	Non-Stationary
Turkey	-3.56	0.04	Stationary
United Kingdom	-2.92	0.19	Non-Stationary
United States	-3.92	0.01	Stationary

- Table 5: Stationary Check with Manual ARIMA

Country	ADF	P-value	Result
Argentina	-3.34	0.07	Non-Stationary
Australia	-2.01	0.57	Non-Stationary
Canada	-3.13	0.11	Non-Stationary
Colombia	-2.52	0.36	Non-Stationary
Ecuador	-4.24	0.01	Stationary
Ethiopia	-3.21	0.09	Non-Stationary
France	-2.57	0.34	Non-Stationary
Germany	-2.82	0.23	Non-Stationary
India	-3.29	0.08	Non-Stationary
Italy	-3.32	0.07	Non-Stationary
Japan	-3.51	0.04	Stationary

Mexico	-4.41	0.01	Stationary
Morocco	-4.11	0.01	Stationary
Pakistan	-3.94	0.01	Stationary
Philippines	-3.07	0.13	Non-Stationary
Russia	-3.58	0.04	Stationary
Saudi Arabia	-3.22	0.09	Non-Stationary
South Africa	-2.92	0.19	Non-Stationary
South Korea	-3.80	0.02	Stationary
Sri Lanka	-1.67	0.71	Non-Stationary
Turkey	-3.75	0.02	Stationary
United Kingdom	-0.86	0.95	Non-Stationary
United States	-3.72	0.02	Stationary

- Table 6: Stationary Check for Manual log ARIMA

Country	ADF	P-value	Result
Argentina	-4.51	0.01	Stationary
Australia	-1.67	0.72	Non-Stationary
Canada	-5.09	0.01	Stationary
Colombia	-4.29	0.01	Stationary
Ecuador	-4.69	0.01	Stationary
Ethiopia	-3.11	0.11	Non-Stationary
France	-2.40	0.41	Non-Stationary
Germany	-6.07	0.01	Stationary
India	-4.57	0.01	Stationary
Italy	-6.63	0.01	Stationary
Japan	-4.47	0.01	Stationary
Mexico	-5.99	0.01	Stationary
Morocco	-3.90	0.02	Stationary
Pakistan	-3.52	0.04	Stationary
Philippines	-3.79	0.02	Stationary
Russia	-5.19	0.01	Stationary

Saudi Arabia	-4.95	0.01	Stationary
South Africa	-3.54	0.04	Stationary
South Korea	-5.98	0.01	Stationary
Sri Lanka	-2.08	0.54	Non-Stationary
Turkey	-2.37	0.42	Non-Stationary
United Kingdom	-4.59	0.01	Stationary
United States	-8.69	0.01	Stationary