Machine Learning Engineer Nanodegree

Capstone Project

Ashraf Hussain 22 June, 2020

Machine Learning Engineer Nanodegree: Forecasting COVID-19 Cases – A Time Series Forecasting Model

I. Definition

Project Overview

On 31 December, 2019, the World Health Organization (WHO) was informed of an outbreak of “pneumonia of unknown cause” detected in Wuhan City, Hubei Province, China. Initially identified as coronavirus disease 2019, it quickly came to be known widely as COVID-19 and has resulted in an ongoing global pandemic. As of 20 June, 2020, more than 8.74 million cases have been reported across 188 countries and territories, resulting in more than 462,000 deaths. More than 4.31 million people have recovered.[^1]

In response to this ongoing public health emergency, Johns Hopkins University (JHU), a private research university in Maryland, USA, developed an interactive web-based dashboard hosted by their Center for Systems Science and Engineering (CSSE). The dashboard visualizes and tracks reported cases in real-time, illustrating the location and number of confirmed COVID-19 cases, deaths and recoveries for all affected countries. It is used by researchers, public health authorities, news agencies and the general public. All the data collected and displayed is made freely available in a GitHub repository.

Problem Statement

This project seeks to forecast number of people infected (new and total case) caused by COVID-19 for a time duration of 30-days based on historical data from JHU.

In my `proposal`, I initially intended to use Time Series Forecasting with Linear Learner as the benchmark along with DeepAR which is an underutilized approach in this area. However after doing extensive research, I have come to the conclusion that any dataset based on epi curve like an epidemic (COVID-19), pandemic (MERS) and/or outbreaks (measles) will not be best suited for DeepAR and/or 14 other Classical Time Series Forecasting Methods (TSFMs) in Python. See `Appendix A`.

Instead, I will be using Matplotlib, numpy, pandas from the scipy ecosystem to help me with data analysis and visualization. I will also be using scipy.optimize from the same ecosystem to create an algorithm based on `curve_fit` function. It will best fit two curves: one for total cases for which I will be using Logistic Function; and another for new cases for which I will be using Gaussian Function.

This is a very common approach used in datasets that follow an epi curve [^10] for example Dr. Tim Churches who took the MERS virus epi curve, and `curve fitted` using the Hubei Province's COVID-19 cases.

I will be comparing my model to the benchmark model, Coronavirus 10-day forecast, designed by the University of Melbourne.

Metrics

The error represents random variations in the data that follow a specific probability distribution (usually Gaussian). The objective of curve fitting is to find the optimal combination of parameters that minimize the error. Here we are dealing with time series, therefore the independent variable is time. In mathematical terms[^4]

> E(error) = E(time) + error

II. Analysis

Data Exploration

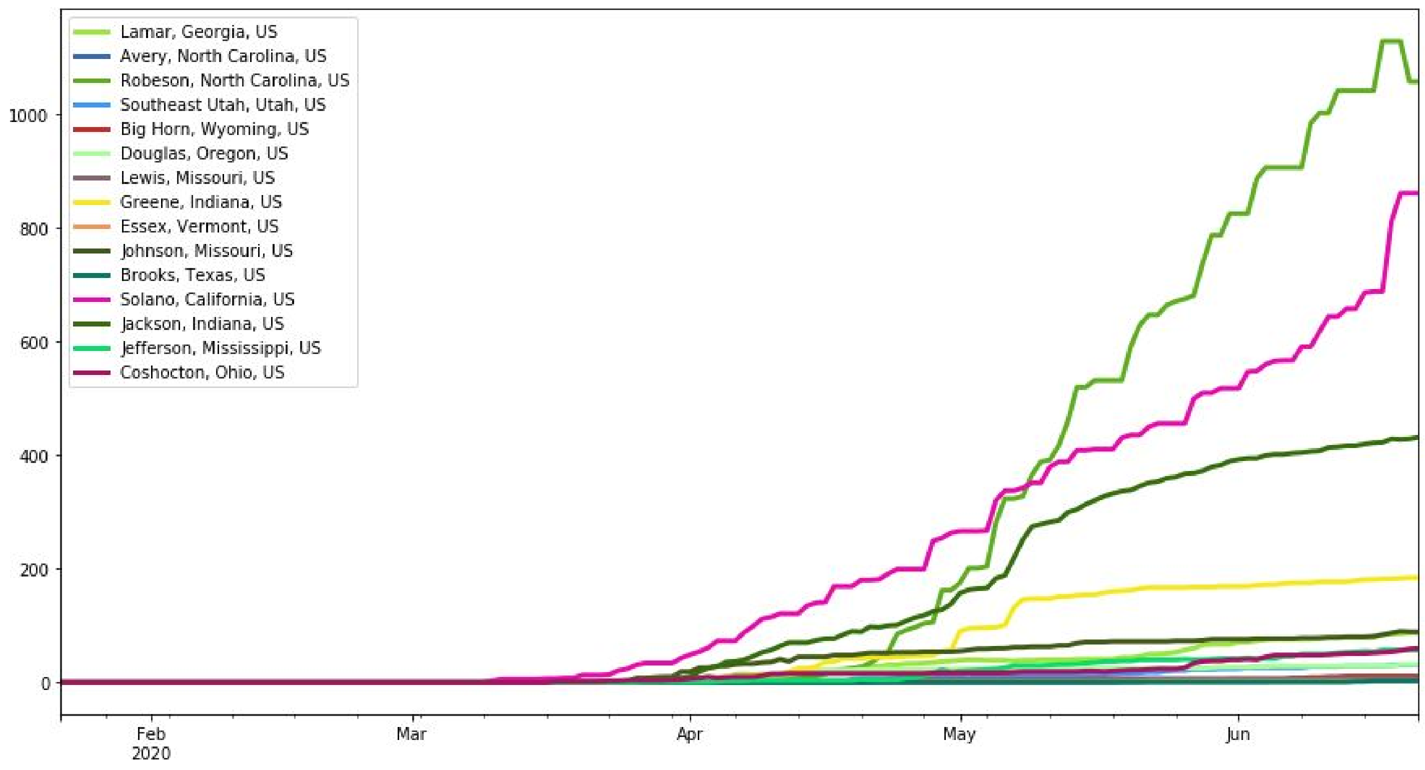
The datasets are accessed from files provided by the JHU GitHub repository time_series_covid19_confirmed_US.csv

The file have the same columns:

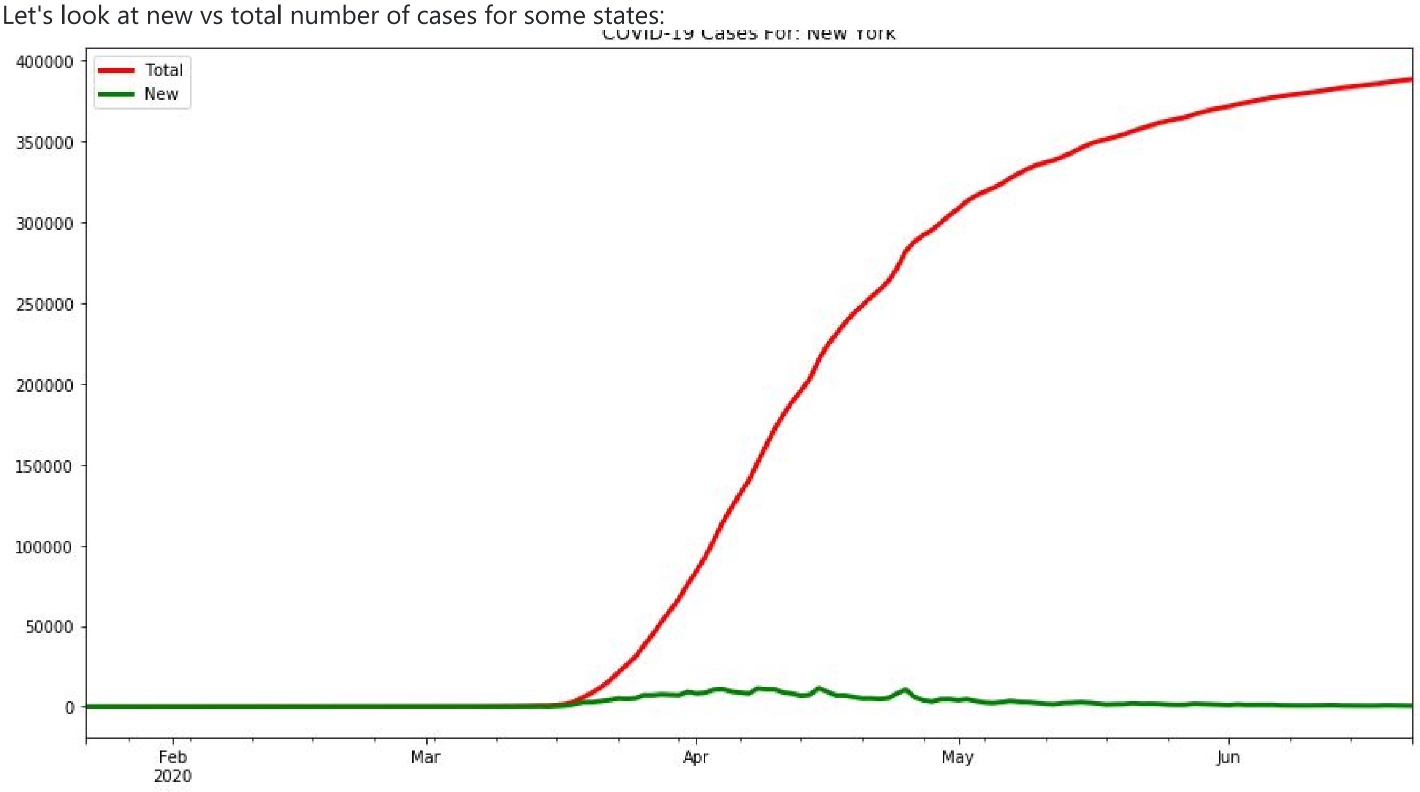
- UID - UID = 840 (country code3) + 000XX (state FIPS code). Ranging from 8400001 to 84000056.
- iso2- Officially assigned country code identifiers 2 Chr (US, CA, ...)
- iso3 - Officially assigned country code identifiers 3 Chr.(USA, CAN, ...)
- code3- country code USA = 840
- FIPS -Federal Information Processing Standards code that uniquely identifies counties within the USA.
- admin2 - County name. US only.
- Province_State - The name of the State within the USA.
- Country_Region - The name of the Country (US).
- Combined_Key - Province_State + Country_Region
- Population - Population
- Number of cases are is columns where each column is a day

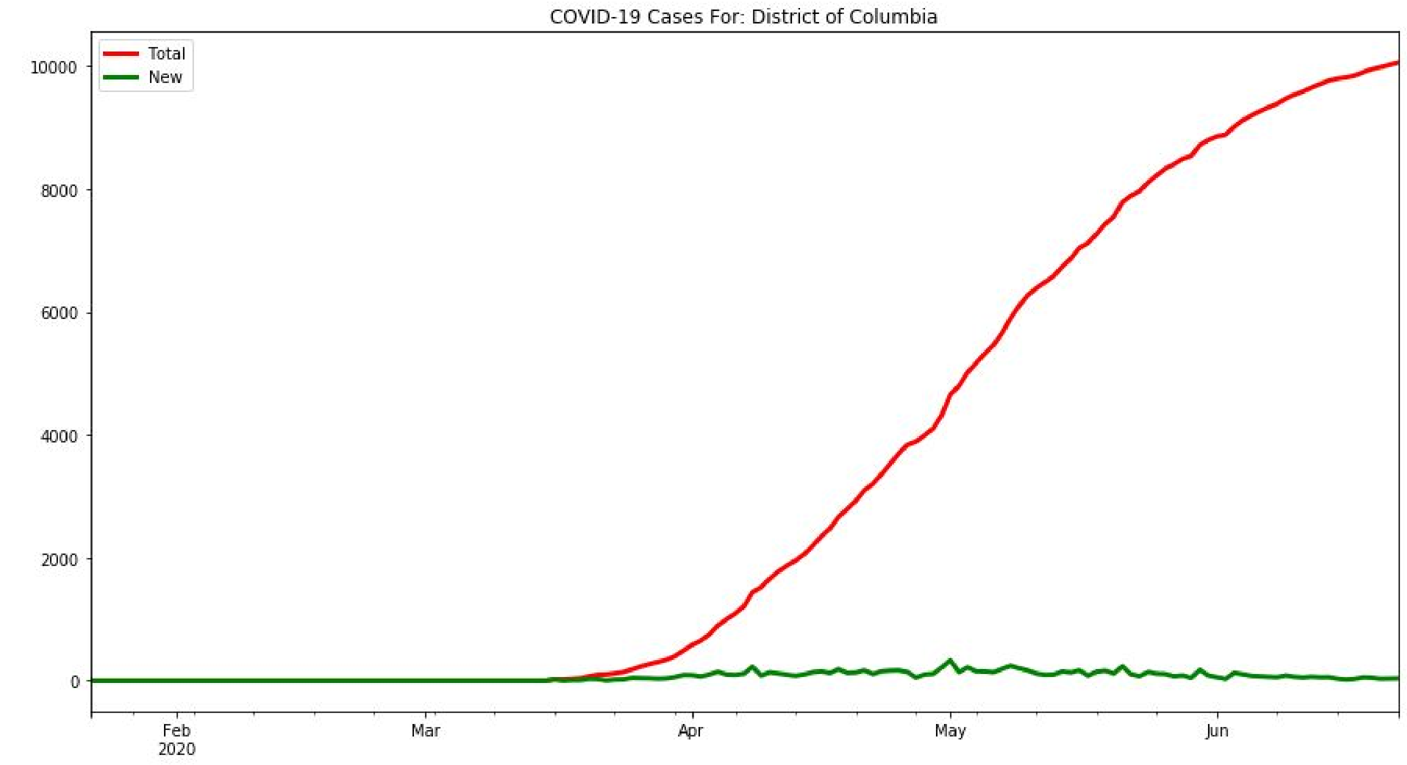
Exploratory Visualization

The plot below shows how the COVID-19 cases increase by city.



Let's look at new vs total number of cases for some states:





Algorithms and Techniques

Algorithms used for my model:

- curve_fit from scipy ecosystem Algorithms used for the benchmark model:
- projSimple form functions.R which is part of nCovForecast toolkit

Benchmark

I am using the University of Melbourne Coronavirus 10-day forecast. Established in 1853, the University of Melbourne is an internationally recognised, research-intensive university with a strong tradition of excellence in teaching, research and community engagement. According to its website, "It's consistently ranked among the leading universities in the world, with international rankings of world universities placing it as number 1 in Australia and number 32 in the world (Times Higher Education World University Rankings 2017-2018)."

The University of Melbourne Coronavirus 10-day forecast is available as an MIT License on (GitHub) (https://github.com/benflips/nCovForecast). This model was done using RStudio. I did not make any changes to this as this would be my benchmark to compare my model with. They have several models predicting different aspects of the dataset. I am only using their forecast function.

III. Methodology

Data Preprocessing

The time_series_covid19_confirmed_US.csv

UID	iso2	iso3	code3	FIPS	Admin2	Province_State	Country_Regio
16	AS	ASM	16	60	NaN	American Samoa	US
316	GU	GUM	316	66	NaN	Guam	US
580	MP	MNP	580	69	NaN	Northern Mariana Islands	US
630	PR	PRI	630	72	NaN	Puerto Rico	US
850	VI	VIR	850	78	NaN	Virgin Islands	US

The dataset was imported into a pandas Dataframe.

Data needed minimal data preprocessing because each Date was in one column and City and State were in other columns.

Province_State	Alabama	Alaska	American Samoa	Arizona	Arkansas	Californi
1/22/20	0	0	0	0	0	0
1/23/20	0	0	0	0	0	0
1/24/20	0	0	0	0	0	0
1/25/20	0	0	0	0	0	0
1/26/20	0	0	0	1	0	2
5 rows × 58 columns						

The dataframe Date was converted to datetime Index

Now we have a clean dataset which will have Date as Index, and sum of cases for each State. From here onwards, we can use the following function to get a cumulative number of cases (total) and new cases (new) for a given State in the dataframe.

.	total	new

.	total	new
count	153	153
mean	168552.3987	2539.137255
std	162319.6416	3209.162717
min	0	0
25%	0	0
50%	139875	1075
75%	345813	4073
max	388488	11434

Implementation

Once the data cleaning was preprocessed, I implemented two models. Both models used the same dataset from JHU:

a. Benchmark model was implemented in RStudio and designed by the University of Melbourne

b. My model was implemented in SageMaker using scipy ecosystem, and I followed these steps:

1. Grouping:
    - Removed unnecessary columns
    - Grouped the data by State
2. Fit the model:
    - Total Cases: Algorithm based on `curve_fit using Logistic Function f(x), defined below. This will fit f(x) curve to total number of COVID-19 cases within a 95% confidence index.

      > f(x) = capacity / (1 + e^-k*(x - midpoint))

    - Total New: Algorithm based on `curve_fit using Gaussian Function g(x), defined below. This will fit g(x) curve to new cases of COVID-19 within a 95% confidence index.

      > g(x) = a * e^(-0.5 * ((x-μ)/σ)**2)

    - Outputs: optimal parameters for a given function
3. Predictions:
    - The `forecast_curve` takes optimal parameters (models) from step 2 above and applies it to a new independent variable based on observations (cases) to forecast.
    - outputs: Graph of actual vs forecast

For the benchmark model, the nCovForecast toolkit comes with GUI, which I used after I ran the model locally.

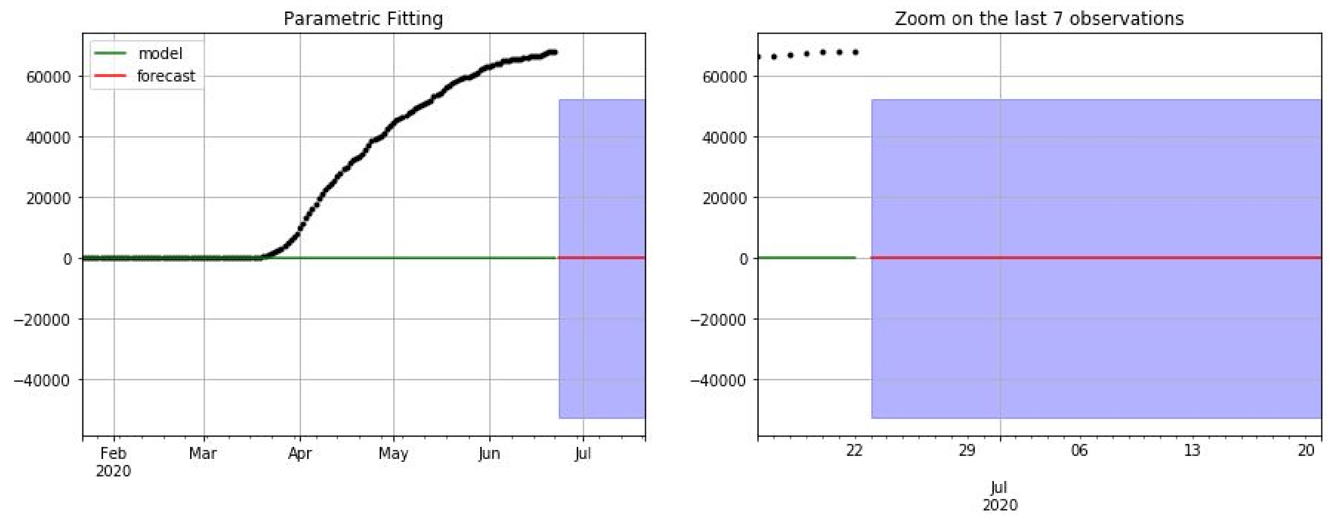
Refinement

The model and the forecast work well when applied to certain States. However, there are a few States that do not work because they are on a different epi curve, for example

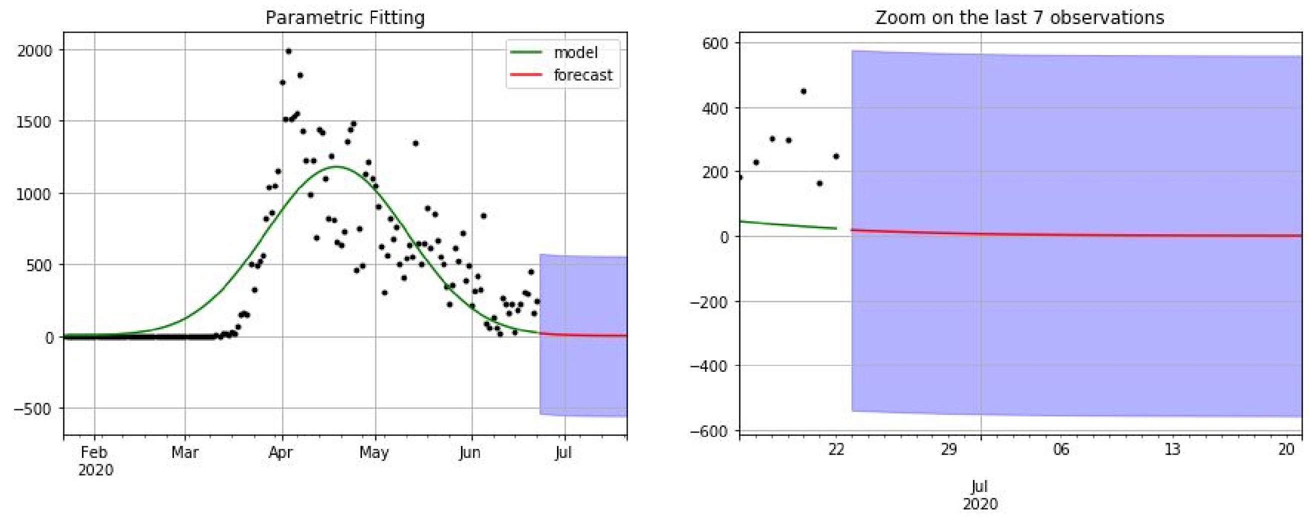
Michigan

My Model:

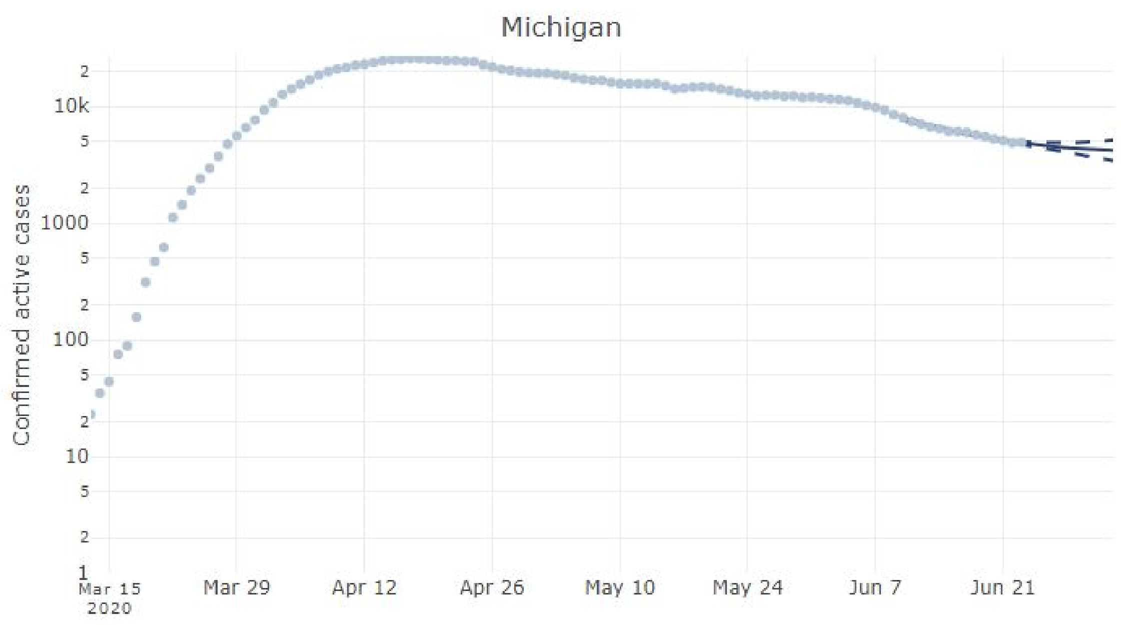
```
###Total Cases###
Making model for Michigan
forecast Michigan
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



```
###New Cases###
Making model for Michigan
forecast Michigan
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



Benchmark:



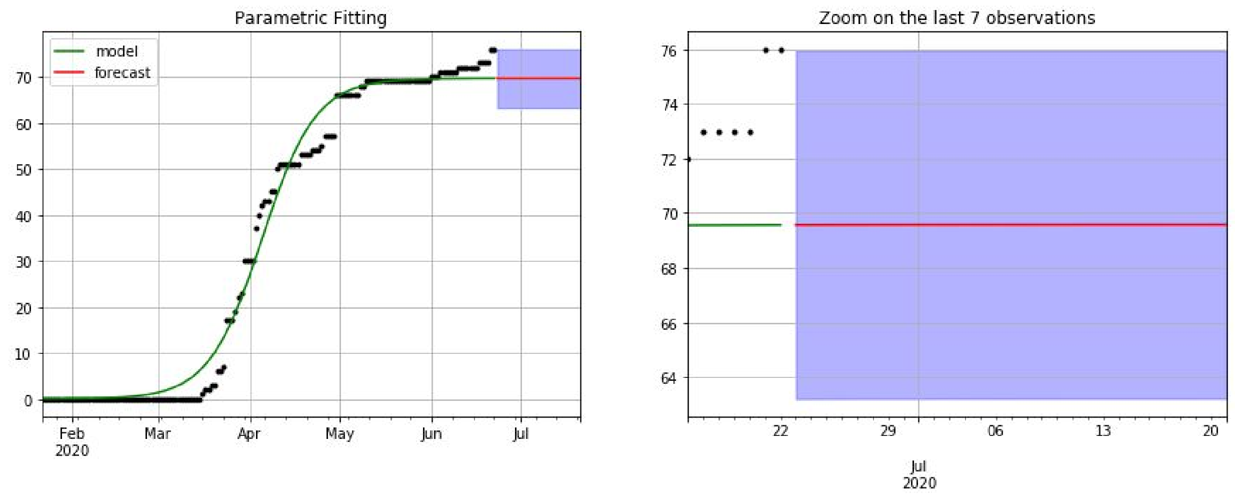
My Model vs Benchmark:

My model could not fit total cases to a Logistic Function, however the benchmark model did not have any problems predicting the forecast within 95% confidence. This was because Michigan is on a different epi curve. Since my model is fitting to a standard Logistic Function f(x), it was unable to predict the forecast. The benchmark model takes into account different epi curves, and hence it was able to predict better than my model.

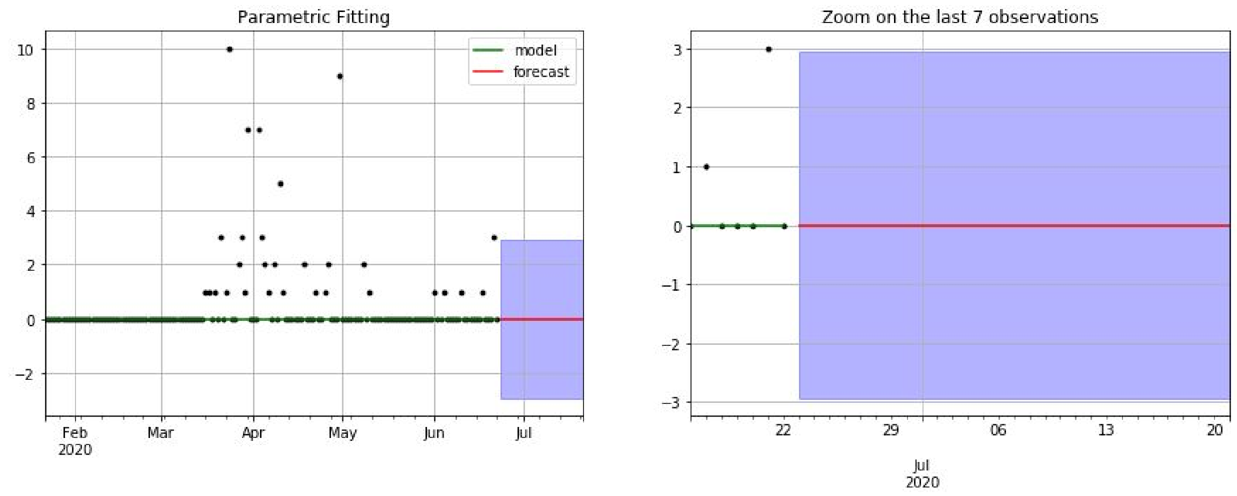
Virgin Islands

My Model:

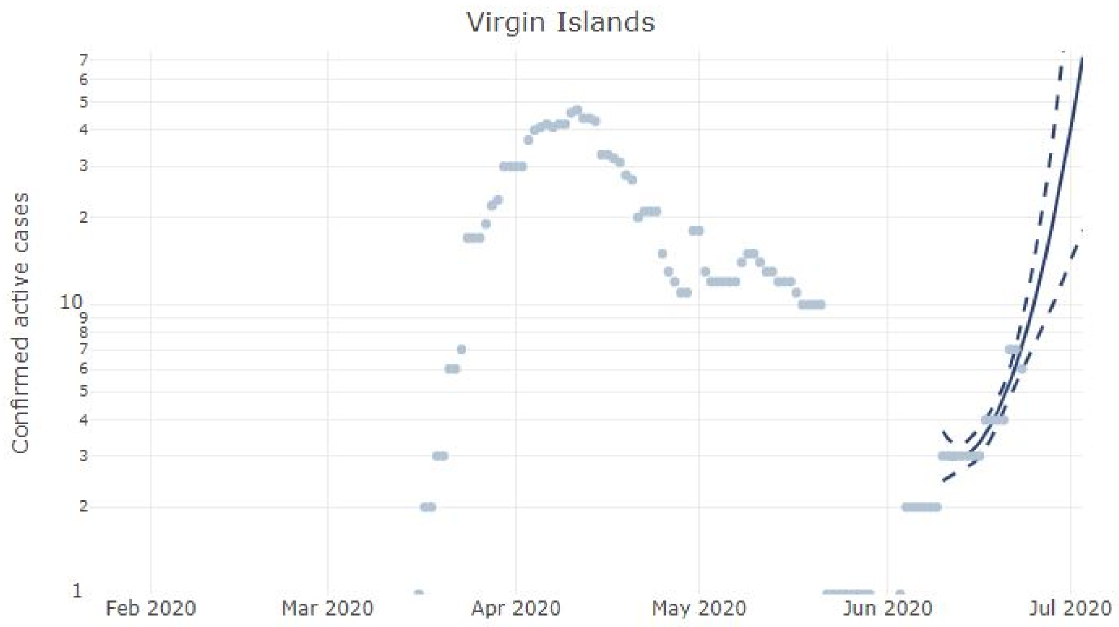
```
###Total Cases###
Making model for Virgin Islands
forecast Virgin Islands
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



```
###New Cases###
Making model for Virgin Islands
forecast Virgin Islands
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



Benchmark:



My Model vs Benchmark: My model could not fit new cases to a Gaussian Function, however the benchmark model did not have any problems predicting the forecast within 95% confidence. This was because Virgin Islands are on a different epi curve. Since my model is fitting to a standard Gaussian Function g(x), it was unable to predict the forecast. The benchmark model takes into account different epi curves, and hence it was able to predict better than my model.

Summary: This clearly shows the power of a better package such as `earlyR` and `EpiEstim` that are part of R when applied to an epidemic dataframe. There are three well-researched articles written by Tim Churches that talk about using R to predict COVID-19 cases.

> Tim Churches is a Senior Research Fellow at the UNSW Medicine South Western Sydney Clinical School at Liverpool Hospital, and a health data scientist at the Ingham Institute for Applied Medical Research, also located at Liverpool, Sydney. His background is in general medicine, general practice medicine, occupational health, public health practice, particularly population health surveillance, and clinical epidemiology.

- COVID-19 epidemiology with R by Tim Churches
- Analysing COVID-19 (2019-nCoV) outbreak data with R - part 1
- Analysing COVID-19 (2019-nCoV) outbreak data with R - part 2

IV. Results

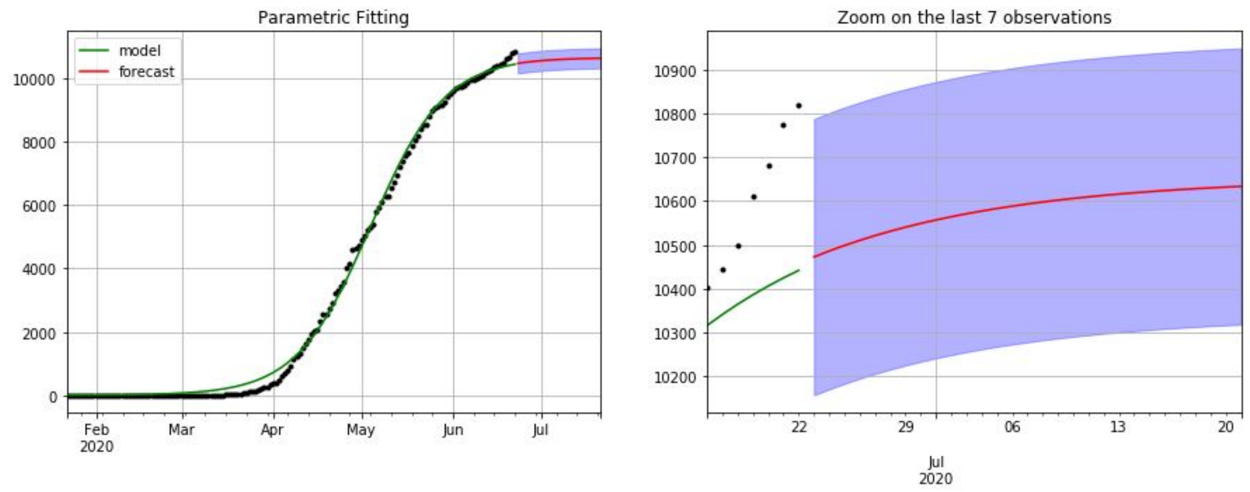
Model Evaluation and Validation

Given my limited understanding and knowledge in the field of epidemiology, this was the simplest model I was able to work with. The model was able to predict future cases for some States very well. However, for others it was unable to predict at all. I used a random generator to pick sample cities to plot the following graphs:

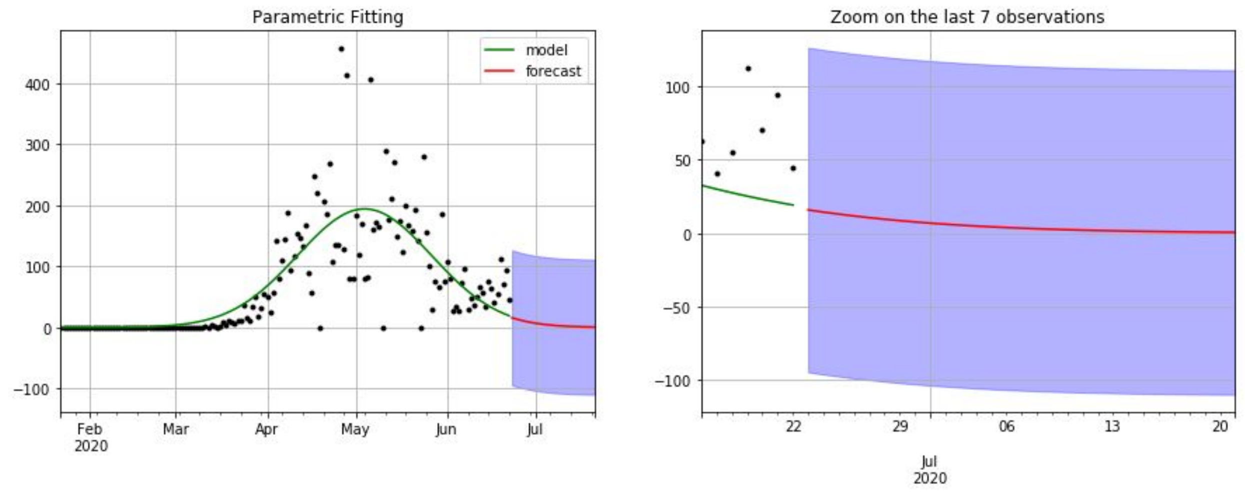
Delaware

My Model:

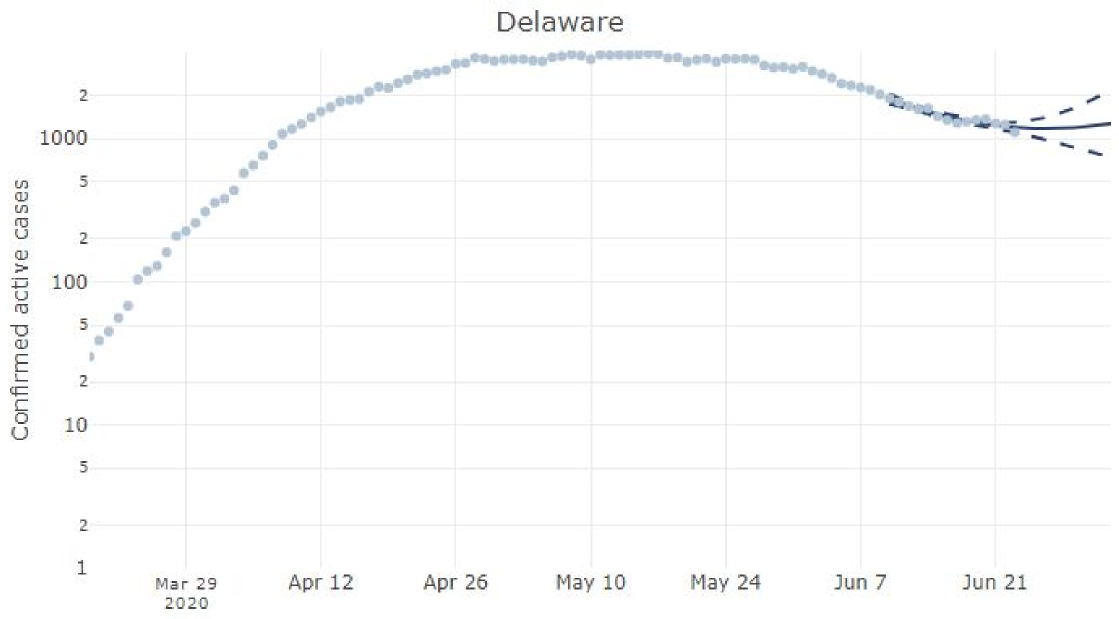
```
###Total Cases###
Making model for Delaware
forecast Delaware
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



```
###New Cases###
Making model for Delaware
forecast Delaware
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



Benchmark:

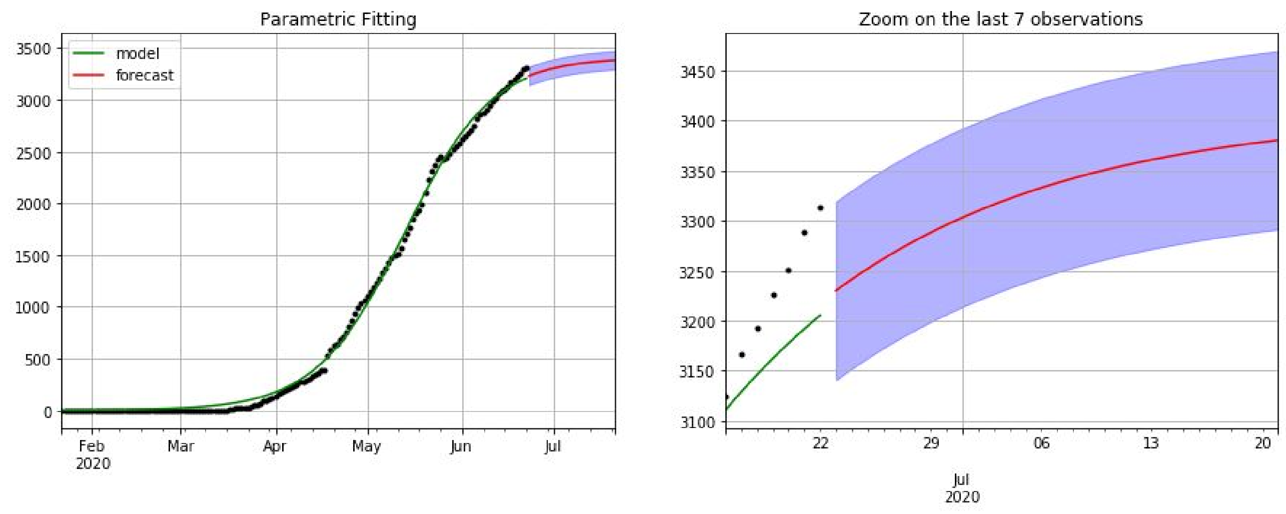


My Model vs Benchmark: In Delaware, the number of new cases seem to be following an epi curve very well. This indicates that the social distancing measures have been working well to reduce the number of new cases. There was a spike in the number of cases in the week of Memorial Day (May 25), however, they seemed to have recovered quickly falling into a standard epi curve pattern. Both my model and benchmark did not have any problems predicting the forecast within 95% confidence and they both showed the same trend.

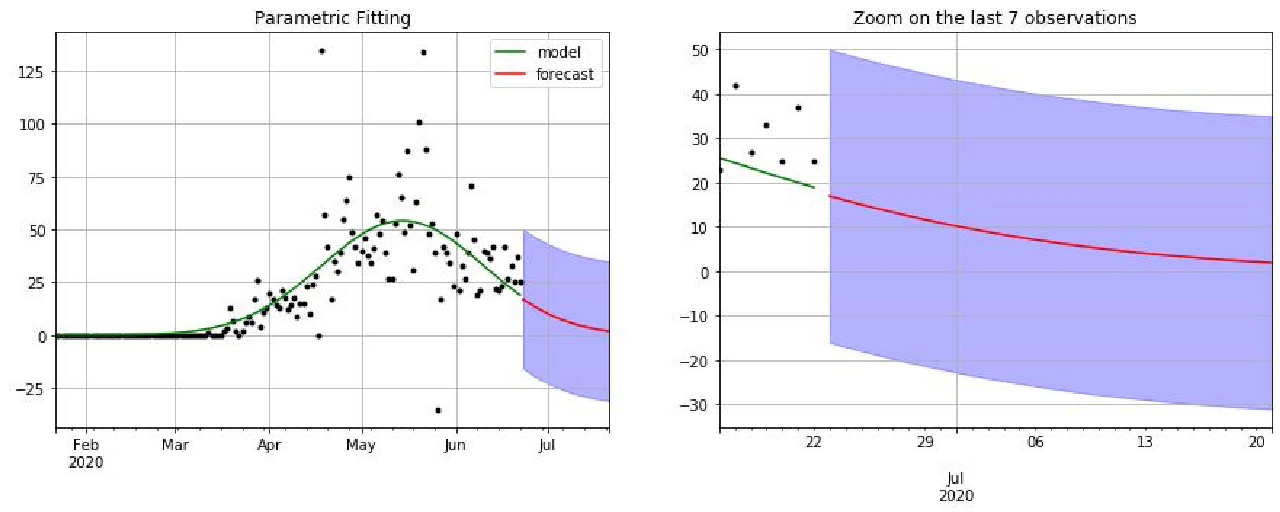
North Dakota

My Model:

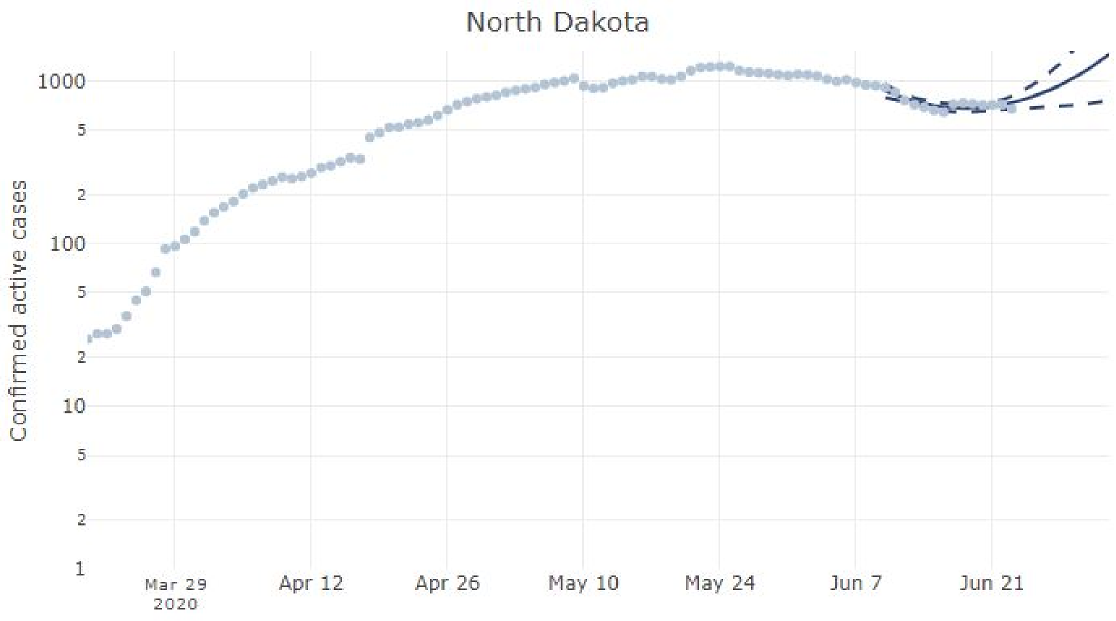
```
###Total Cases###
Making model for North Dakota
forecast North Dakota
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



```
###New Cases###
Making model for North Dakota
forecast North Dakota
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



Benchmark:

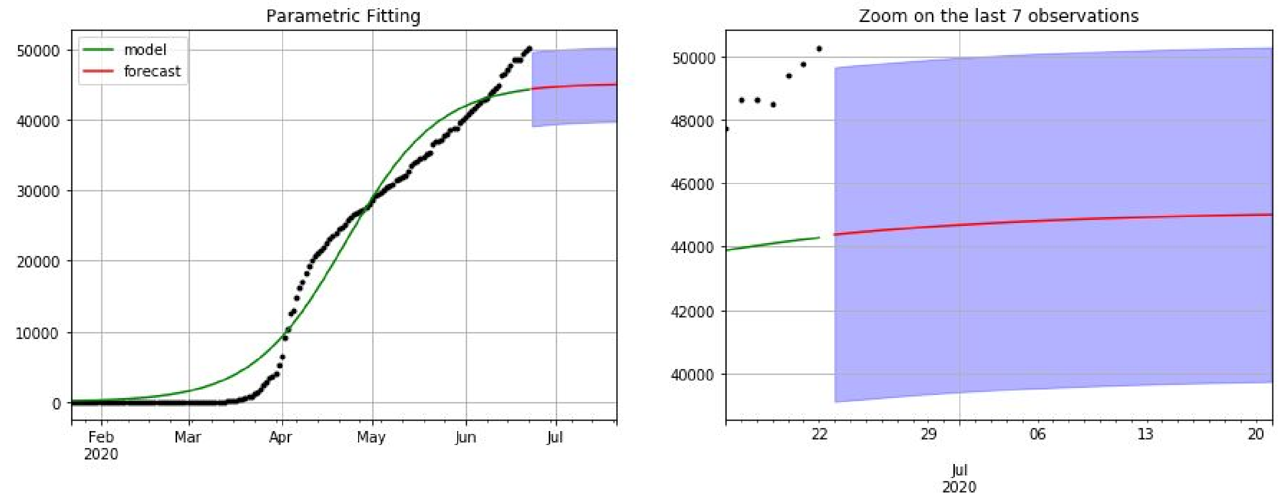


My Model vs Benchmark: New COVID-19 cases in North Dakota seem to be following the same pattern as that of Delaware indicating that social distancing measures are proving to be effective. Their cases peaked around Memorial Day, similar to Delaware, however they seem to have taken slightly longer to come back to a normal epi curve. Both my model and benchmark did not have any problems predicting the forecast within 95% confidence, however my model shows a downward trend and the benchmark shows an upward trend.

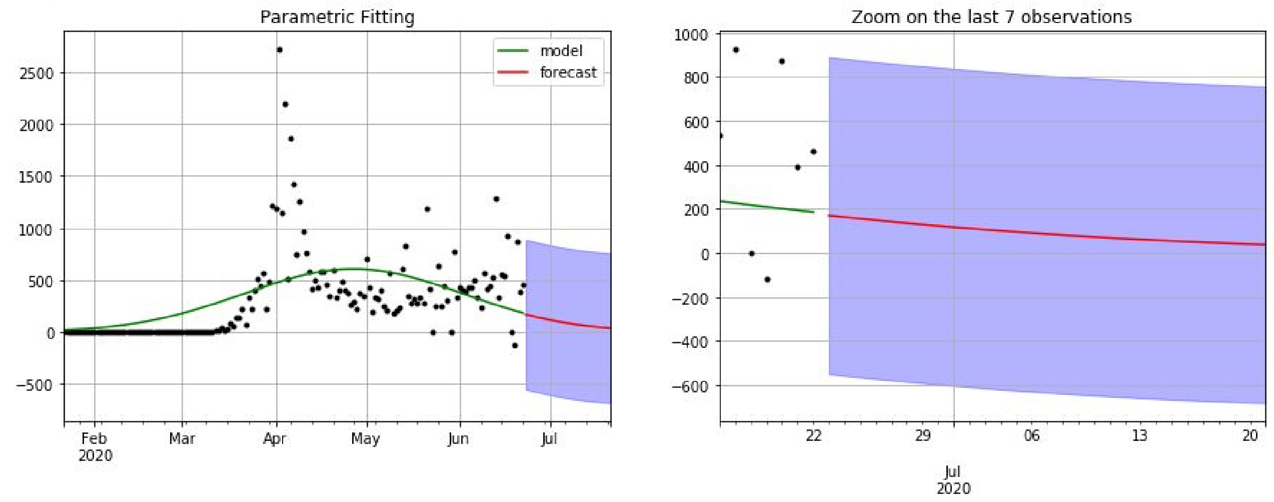
Maryland

My Model:

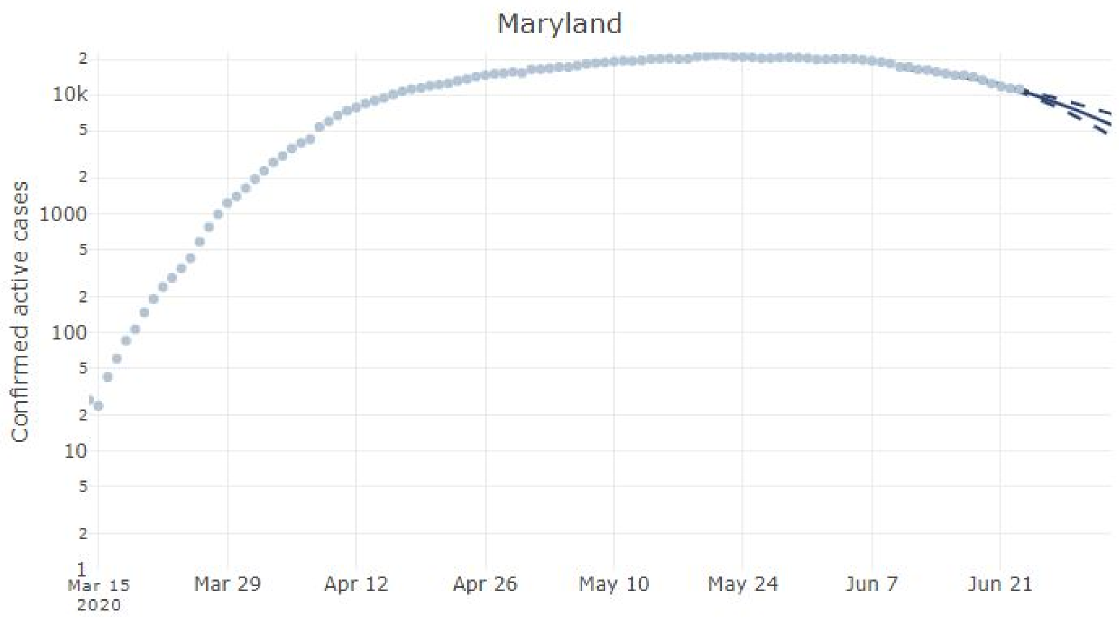
```
###Total Cases###
Making model for Louisiana
forecast Louisiana
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



```
###New Cases###
Making model for Louisiana
forecast Louisiana
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



Benchmark:



My Model vs Benchmark: Contrary to Delaware and North Dakota, new cases in Maryland peaked during the long weekend in April and seemed to stay within the mean of an epi curve. This irregular peaking may be attributed to people traveling between States for the long weekend and for other reasons. Both my model and benchmark did not have any problems predicting the forecast within 95% confidence however my model shows an upward trend and the benchmark shows a downward trend.

Justification

It is quite evident that `earlyR` is a better package that implements simple estimation of infectiousness, as measured by the reproduction number (R), in the early stages of an outbreak. The second package `EpiEstim` implements a Bayesian approach for quantifying transmissibility over time during an epidemic. More specifically, it allows estimating the instantaneous and case reproduction numbers during an epidemic for which a time series of incidence is available and the distribution of the serial interval (time between symptoms onset in a primary case and symptoms onset in secondary case) is _ more or less precisely _ known. `earlyR` and `EpiEstim` are part of R toolkit.

Source `earlyR` : https://www.repidemicsconsortium.org/earlyR/ Source `EpiEstim` : https://sites.google.com/site/therepiproject/r-pac/epiestim

Research is ongoing and better benchmarks for the COVID-19 epidemic are being developed every day by people like Dr. Tim Churches, a medically-trained epidemiologist. Epidemiologists are designing new models predominantly in R. Below are some examples:

- an introduction to R for epidemiologists by Dr. Charles DiMaggio (New York University Departments of Surgery and Population Health)
- R-software: A Newer Tool in Epidemiological Data Analysis by Dr. Amir Maroof Khan (Department of Community Medicine, University College of Medical Sciences and GTB Hospital, Delhi, India)
- Analysis of Epidemiological Data Using R and Epicalc by Prof. (Prince of Songkla University, Thailand)

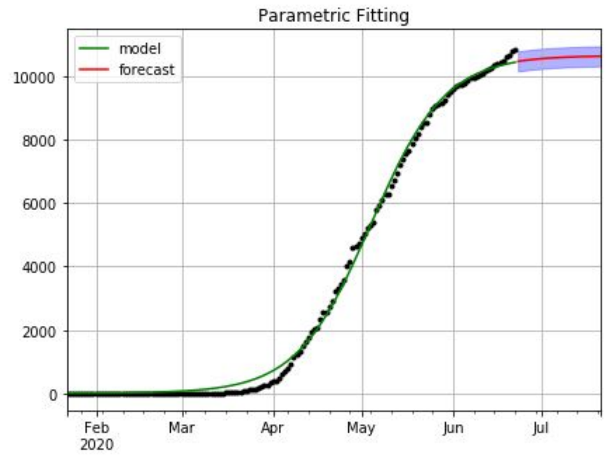
A Google Scholar search on `epidemiology and r` returns over three million articles, however a similar search with other keywords yielded: epidemiology and ARIMA (18k articles), epidemiology and SARIMA (46 articles), and epidemiology and DeepAR (0 articles).

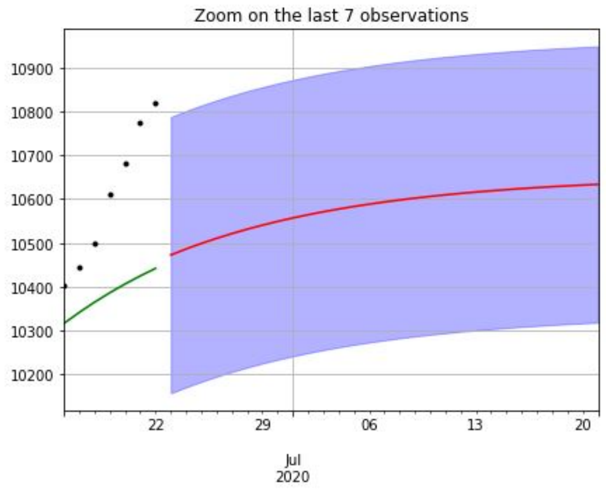
V. Conclusion

Reflection

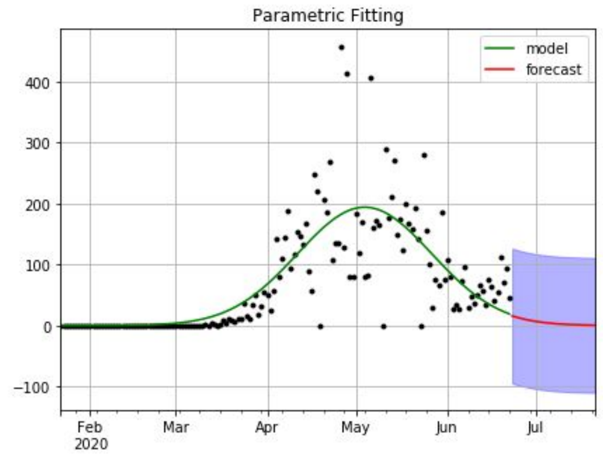
Based on my observations, the dataframes for **North Dakota** and **Delaware** fit the epidemic curve well.

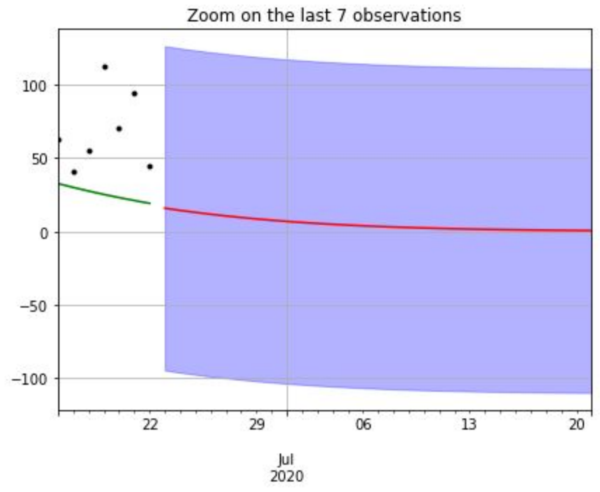
```
###Total Cases###
Making model for Delaware
forecast Delaware
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



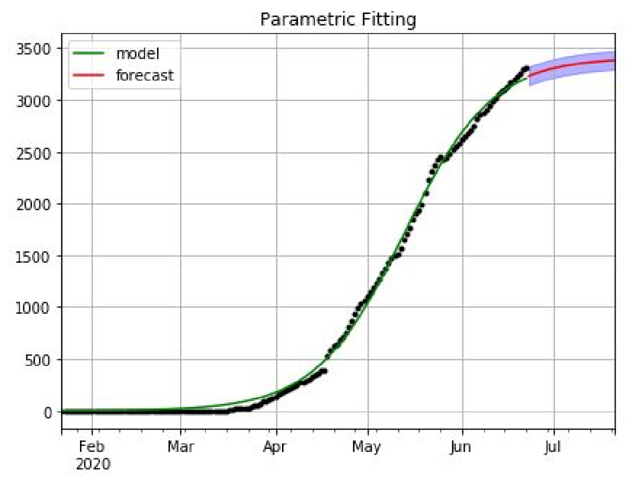


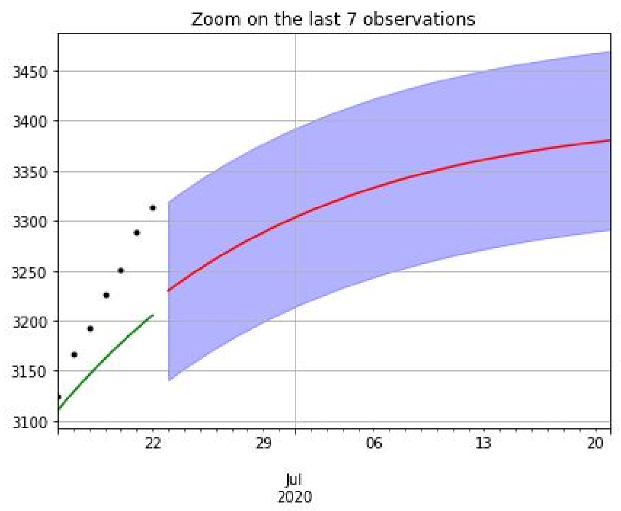
```
###New Cases###
Making model for Delaware
forecast Delaware
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```



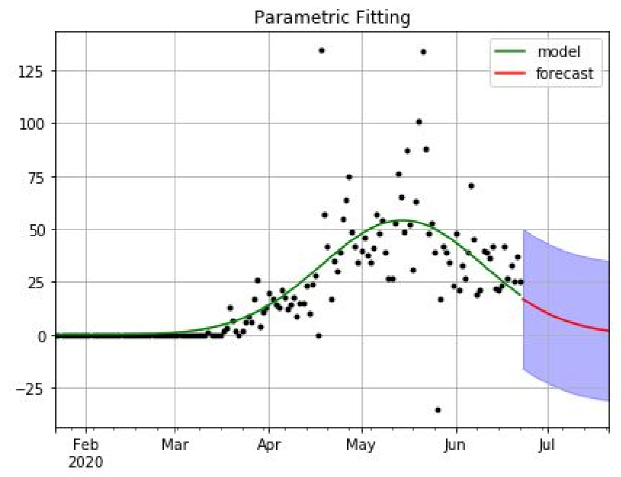


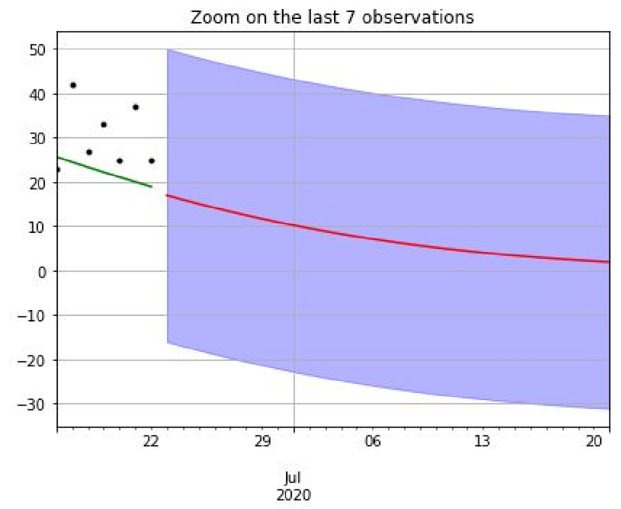
```
###Total Cases###
Making model for North Dakota
forecast North Dakota
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```





```
###New Cases###
Making model for North Dakota
forecast North Dakota
--- generating index date --> start: 2020-06-23 00:00:00 | end: 2020-07-21 00:00:00 | len: 29 ---
```





With respect to North Dakota, they have 3,313 total cases and 2,952 recovered cases with only 77 deaths. If they continue on this path, based on my model they are predicted to reach equilibrium (zero new cases) by the end of July or August 2020.

Delaware is another State that is following the epi curve closely. Though they had some peaks around Memorial Day week, they seem to have recovered well. If they continue on this trend, based on my model they are predicted to reach a minimum amount of cases by the end of July or August 2020.

The only challenge with these predictions is that it does not take into account the human factors. Human factors can be defined as human interactions in relation to their environment, such as not following social distance measures; not wearing masks when going out in public areas; not following proper safety and sanitization rules; non-essential travel from one city to another to visit family, friends or to go on a vacation; and more. This will increase the rate of transmission leading to peaks in new cases as was witnessed around the Memorial Day week.

Final Thoughts and Improvements

This project has taught me a lot about machine learning and how epidemiologists are using machine learning. There were several lessons to be learnt but the top five that stood out for me are:

1. Using `earlyR` and `EpiEstim` packages that are part of R would have been a better and more accurate option for my model. `earlyR` was designed for simple estimation of infectiousness, as measured by the reproduction number (R), in the early stages of an outbreak. It has been in epidemiologists' circles since 2017 and was desgined by Thibaut Jombart, Associate Proffessor in outbreak analytics at LSHTM / Imperial College, London, UK. `EpiEstim` implements a Bayesian approach for quantifying transmissibility over time during an epidemic. More specifically, it allows estimating the instantaneous and case reproduction numbers during an epidemic for which a time series of incidence is available and the distribution of the serial interval (time between symptoms onset in a primary case and symptoms onset in secondary case) is _ more or less precisely _ known.

2. Work with the right expert: My understanding of epidemiology is limited. I would have appreciated an opportunity to work with an epidemiologist to understand how an epidemic works in order to apply the nuances of the data to machine learning. It is really critical to understand the story behind the data to be able to build a good machine model.

3. Generalize the problem: My proposal was confined to a narrow dataframe and solution that did not leave any room for improvisation. This led to a tunnel vision when trying to build the data model. An alternative would have been generalizing the problem in the proposal which would have allowed me to manoeuvre in different ways to come up with innovative solutions.

4. Time management: There is a ton of published papers and research available that provide top-notch information. Going forward, I will allow myself sufficient time to go through available research before embarking on the solution. Some good sources of information are Google Scholar, Medium, GitHub repositories, and other open-source packages & libraries. Hindsight is indeed 2020. Armed with this knowledge, I am confident that I can continue to apply myself in the field of machine learning to find novel solutions to human challenges.

5. This dataset would be great for Udacity's course called Data Analysis with R and Data Analyst Nanodegree Program

To conclude, I would like to thank the following people without whom I would not have been able to complete my project and get an understanding of how COVID-19 is being used in machine learning:

- Dr. Tim Churches.
- Dr. Jason Brownlee
- Mauro Di Pietro
- Subhasree Chatterjee

And last but not least, my wife Shamsia Quraishi for supporting me during my training and being there for me.

Once again thanks and be safe.

Appendix A

Why DeepAR/Classical Time Series Forecasting Methods fails to work on epidemic(COVID-19), pandemic(MERS) and/or outbreak (measles) datasets.

To use DeepAR it needs to meet the following criteria [^5]:

1. Except for when splitting your dataset for training and testing, always provide the entire time series for training, testing, and when calling the model for inference. Regardless of how you set `context_length`, don't break up the time series or provide only a part of it. The model uses data points further back than the value set in `context_length` for the lagged values feature.[^5]

2. When tuning a DeepAR model, you can split the dataset to create a training dataset and a test dataset. In a typical evaluation, you would test the model on the same time series used for training, but on the future `prediction_length` time points that follow immediately after the last time point visible during training. You can create training and test datasets that satisfy this criteria by using the entire dataset (the full length of all time series that are available) as a test set and removing the last `prediction_length` points from each time series for training. During training, the model doesn't see the target values for time points on which it is evaluated during testing. During testing, the algorithm withholds the last `prediction_length` points of each time series in the test set and generates a prediction. Then it compares the forecast with the withheld values. You can create more complex evaluations by repeating time series multiple times in the test set, but cutting them at different endpoints. With this approach, accuracy metrics are averaged over multiple forecasts from different time points. For more information, see Tune a DeepAR Model.[^5]

3. Avoid using very large values (>400) for the `prediction_length` because it makes the model slow and less accurate. If you want to forecast further into the future, consider aggregating your data at a higher frequency. For example, use `5min` instead of `1min` .[^5]

4. Because lags are used, a model can look further back in the time series than the value specified for `context_length` . Therefore, you don't need to set this parameter to a large value. We recommend starting with the value that you used for `prediction_length` .[^5]

5. We recommend training a DeepAR model on as many time series as are available. Although a DeepAR model trained on a single time series might work well, standard forecasting algorithms, such as ARIMA or ETS, might provide more accurate results. The DeepAR algorithm starts to outperform the standard methods when your dataset contains hundreds of related time series. Currently, DeepAR requires that the total number of observations available across all training time series is at least 300.[^5]

[^5]:Best Practices for Using the DeepAR Algorithm

Based on the above requirements we cannot use DeepAR. I then looked into the 14 other Classical Time Series Forecasting Methods (TSFMs) in Python, list below, to see if we can use any of them. The first thing we need to do is to check for Stationarity. A common assumption in many time series techniques is that the data are stationary. A stationary process has the property that the mean, variance and autocorrelation structure do not change over time. Stationarity can be defined in precise mathematical terms, but for our purpose we mean a flat looking series, without trend, constant variance over time, a constant autocorrelation structure over time and no periodic fluctuations (seasonality).[^10] [^10]:6.4.4.2. Stationarity

To summarize a time-series to be Stationarity, the following should not change over time

mean(μ) standard deviation(σ) Autocorrelation structure (No seasonality)

There are a number of unit root tests we can do to check if a dataset is stationary or non-stationary. The Augmented Dickey-Fuller is one of the more widely used tests. It uses an autoregressive model and optimizes an information criterion across multiple different lag values.

According to Augmented Dickey-Fuller test the null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

`Null Hypothesis (H0):` If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure.

`Alternate Hypothesis (H1):` The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have time-dependent structure. We interpret this result using the p-value from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (stationary), otherwise a p-value above the threshold suggests we fail to reject the null hypothesis (non-stationary).

- p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
- p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

source: How to Check if Time Series Data is Stationary with Python

Want to know more about How to Check if Time Series Data is Stationary with Python

> If we fit a stationary model to data, we assume our data are a realization of a stationary process. So our first step in an analysis should be to check whether there is any evidence of a trend or seasonal effects and, if there is, remove them.

— Page 122, Introductory Time Series with R.

```
 The problem is that we cannot remove any data because we would end up with only 2 States that are deemed to pass the
Stationarity test.
```

Data output of Stationarity test based on the function below

```
  Testing Null Hypothesis
  Calculation Complete
  adfuller results:
  56 p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
  2 p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.
  ---------
  58 Total
```

University of Melbourne Forecasting Function

> We are interested in knowing how the number of active cases is going to change in the near term. We provide two alternative growth models. The growth dynamics of epidemics are complex, but we make the simplifying assumption that the epidemic is a long way from population saturation (i.e. that there is a ready supply of susceptible hosts) such that simple exponential growth provides a reasonable short-term approximation. The first model -- the "constant growth" model -- assumes a constant growth rate, in which case active cases follow an exponential growth model such that $E(A_t) = A_0e^{rt}$, where A_0 is the initial number of active cases, and r is the (constant) intrinsic growth rate. The second model -- "time-varying growth" -- assumes that the growth rate changes linearly over time. Empirically, linear change in growth is what we have been observing in this epidemic as populations enact physical distancing and quarantine measures. Under this model, growth follows a Gaussian function such that $E(A_t) = A_0e^{r_0t+\frac{b}{2}t^2}$. Here, r_0 is the initial growth rate, and b is the rate of change in growth rate over time.
>
> To fit both models, we take the natural logarithm of both sides, yielding $\ln A_t = rt + \ln A_0$ (constant growth) and $\ln A_t = \frac{b}{2}t^2+rt + \ln A_0$ (time-varying growth). This shows us that we can fit a simple linear regression of $\ln A_t$ against t in the constant growth scenario, and a quadratic function in the case of the time-varying model.
>
> These fits give us an estimate of intrinsic growth rate, r, or $r(t)$.
>
> We fit each model to the last n days of A_t data (where n is determined by the user with the input slider) and extrapolate the fitted model to capture ten day into the future from now. Fitting and extrapolation is effected with the log-transformed model (lower plot on "10-day forecast" tab, with 95% confidence intervals) and the log of expected active case numbers is back-transformed to the original scale to produce the top plot on the "10-day forecast" tab. We provide a larger number of days for fit input for the time-varying growth model because this model estimates an additional parameter, so requires more data.
>
> When public health interventions are rapidly changing the growth rate, this can be seen as deviations from the expected straight line on the log-plot. In these situations, when growth rate is declining rapidly (the curve is flattening), forecasts from the constant growth model (averaging growth over the last ten days) will be biased upwards. By altering the slider you can adjust the window over which growth rate is averaged, so you can get a sense of how recent shifts are affecting the forecast. The time-varying growth rate forecast should be less sensitive to changes in n, and is the better model when growth rates are changing in a steady linear manner.

Research: Definitions of 14 Classical Time Series Forecasting Methods (TSFMs) in Python

AR (autoregressive model): The autoregression (AR) method models the next step in the sequence as a linear function of the observations at prior time steps. The notation for the model involves specifying the order of the model p as a parameter to the AR function, e.g. AR(p). For example, AR(1) is a first-order autoregression model. The method is suitable for univariate time series without trend and seasonal components.[^6]

MA (moving-average model): The moving average (MA) method models the next step in the sequence as a linear function of the residual errors from a mean process at prior time steps. A moving average model is different from calculating the moving average of the time series. The notation for the model involves specifying the order of the model q as a parameter to the MA function, e.g. MA(q). For example, MA(1) is a first-order moving average model. The method is suitable for univariate time series without trend and seasonal components.[^6]

ARMA (autoregressive-moving-average model): The Autoregressive Moving Average (ARMA) method models the next step in the sequence as a linear function of the observations and residual errors at prior time steps. It combines both Autoregression (AR) and Moving Average (MA) models. The notation for the model involves specifying the order for the AR(p) and MA(q) models as parameters to an ARMA function, e.g. ARMA(p, q). An ARIMA model can be used to develop AR or MA models. The method is suitable for univariate time series without trend and seasonal components.[^6]

ARIMA (autoregressive integrated moving average model): The Autoregressive Integrated Moving Average (ARIMA) method models the next step in the sequence as a linear function of the differences observations and residual errors at prior time steps. It combines both Autoregression (AR) and Moving Average (MA) models as well as a differencing pre-processing step of the sequence to make the sequence stationary, called integration (I). The notation for the model involves specifying the order for the AR(p), I(d), and MA(q) models as parameters to an ARIMA function, e.g. ARIMA(p, d, q). An ARIMA model can also be used to develop AR, MA, and ARMA models. The method is suitable for univariate time series with trend and without seasonal components.[^6]

SARIMA (seasonal autoregressive integrated moving average model): The Seasonal Autoregressive Integrated Moving Average (SARIMA) method models the next step in the sequence as a linear function of the differenced observations, errors, differenced seasonal observations, and seasonal errors at prior time steps. It combines the ARIMA model with the ability to perform the same autoregression, differencing, and moving average modeling at the seasonal level. The notation for the model involves specifying the order for the AR(p), I(d), and MA(q) models as parameters to an ARIMA function and AR(P), I(D), MA(Q) and m parameters at the seasonal level, e.g. SARIMA(p, d, q)(P, D, Q)m where “m” is the number of time steps in each season (the seasonal period). A SARIMA model can be used to develop AR, MA, ARMA and ARIMA models. The method is suitable for univariate time series with trend and/or seasonal components.[^6]

SARIMAX (seasonal autoregressive integrated moving average model with exogenous variables): The Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX) is an extension of the SARIMA model that also includes the modeling of exogenous variables. Exogenous variables are also called covariates and can be thought of as parallel input sequences that have observations at the same time steps as the original series. The primary series may be referred to as endogenous data to contrast it from the exogenous sequence(s). The observations for exogenous variables are included in the model directly at each time step and are not modeled in the same way as the primary endogenous sequence (e.g. as an AR, MA, etc. process). The SARIMAX method can also be used to model the subsumed models with exogenous variables, such as ARX, MAX, ARMAX, and ARIMAX. The method is suitable for univariate time series with trend and/or seasonal components and exogenous variables.[^6]

VARMA (vector autoregressive moving average model): The Vector Autoregression Moving-Average (VARMA) method models the next step in each time series using an ARMA model. It is the generalization of ARMA to multiple parallel time series, e.g. multivariate time series. The notation for the model involves specifying the order for the AR(p) and MA(q) models as parameters to a VARMA function, e.g. VARMA(p, q). A VARMA model can also be used to develop VAR or VMA models. The method is suitable for multivariate time series without trend and seasonal components.[^6]

Vector Autoregression (VAR) The Vector Autoregression (VAR) method models the next step in each time series using an AR model. It is the generalization of AR to multiple parallel time series, e.g. multivariate time series. The notation for the model involves specifying the order for the AR(p) model as parameters to a VAR function, e.g. VAR(p). The method is suitable for multivariate time series without trend and seasonal components.[^6]

Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) The Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) is an extension of the VARMA model that also includes the modeling of exogenous variables. It is a multivariate version of the ARMAX method. Exogenous variables are also called covariates and can be thought of as parallel input sequences that have observations at the same time steps as the original series. The primary series(es) are referred to as endogenous data to contrast it from the exogenous sequence(s). The observations for exogenous variables are included in the model directly at each time step and are not modeled in the same way as the primary endogenous sequence (e.g. as an AR, MA, etc. process). The VARMAX method can also be used to model the subsumed models with exogenous variables, such as VARX and VMAX. The method is suitable for multivariate time series without trend and seasonal components with exogenous variables.[^6]

Simple Exponential Smoothing (SES) The Simple Exponential Smoothing (SES) method models the next time step as an exponentially weighted linear function of observations at prior time steps. The method is suitable for univariate time series without trend and seasonal components.[^6]

Holt Winters Exponential Smoothing (HWES)

The Holt Winters Exponential Smoothing (HWES) also called the Triple Exponential Smoothing method models the next time step as an exponentially weighted linear function of observations at prior time steps, taking trends and seasonality into account. The method is suitable for univariate time series with trend and/or seasonal components.[^6]

ARCH (autoregressive conditional heteroskedasticity model) Autoregressive conditional heteroskedasticity (ARCH) is a time-series statistical model used to analyze effects left unexplained by econometric models. In these models, the error term is the residual result left unexplained by the model. The assumption of econometric models is that the varianceof this term will be uniform. This is known as "homoscedasticity." However, in some circumstances, this variance is not uniform, but "heteroskedastic."[^7]

ARIMAX (autoregressive integrated moving average model with exogenous variables) A time series model using the Autoregressive Integrated Moving Average with exogenous variables (ARIMAX) function was developed to predict impacts from groundwater pumping on Silver Springs discharge in Ocala Florida. This effort was conducted to determine the effects of groundwater withdrawal using the statistical relationship between rainfall and spring discharge at Silver Springs. Other statistical models were developed in previous work by both Southwest Florida Water Management Districts and St Johns River Water Management District. However, there were several opportunities for improvement including using consistent data, model calibration period, and residual periods. Additionally previous statistical methods included Multiple Linear Regression and Line of Organic Correlation methods. These methods did not account for autocorrelation that is present in many time series analysis. Through inter-district collaboration, data was made consistent and new methods were explored. The ARIMAX model was explored in this paper and is useful for prediction when autoregressive patterns are present in model residuals that bias modeled coefficients.[^8]

GARCH (generalized autoregressive conditional heteroskedasticity model) A natural generalization of the ARCH (Autoregressive Conditional Heteroskedastic) process introduced in Engle (1982) to allow for past conditional variances in the current conditional variance equation is proposed. Stationarity conditions and autocorrelation structure for this new class of parametric models are derived. Maximum likelihood estimation and testing are also considered.[^9]

[^6]:11 Classical Time Series Forecasting Methods in Python (Cheat Sheet)[^7]:Autoregressive Conditional Heteroskedasticity (ARCH)

[^8]:Autoregressive Integrated Moving Average Model with exogenous variables (ARIMAX) transfer function model for Sharpes Ferry Well and Silver Springs

[^9]:Generalized autoregressive conditional heteroskedasticity

[^1]:COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)". ArcGIS. Johns Hopkins University. Retrieved 20 June 2020.

[^2]:Logistic Growth Model for COVID-19

[^3]:Mathematical prediction of the time evolution of the COVID-19 pandemic in Italy by a Gauss error function and Monte Carlo simulations

[^10]:Logistic growth modelling of COVID-19 proliferation in China and its international implications Covid-19 predictions using a Gauss model, based on data from April 2

[^4]:Time Series Forecasting with Parametric Curve Fitting

[^11]:Time Series Forecasting with Parametric Curve Fitting