Visualization of Covid-19 Data with Time Series Forecasting Models

Nick Abcarius, Yi-Hsuan Cheng, Lia Freed-Doerr, Emma Baier, Joshua Winter, Andrey Yakymovych, Zibo Zhang, Sabarno Islam, Kwan Tien Ooi

ECS 171 Spring Quarter 2021

https://github.com/nmabcarius/ECS171_Group_5

1 Introduction

1.1 Problem statement

Since the pandemic started, traveling has been an issue for many people. International travel has been out of the question for many but even traveling between states has been somewhat taboo due to Covid-19 safety protocols. Due to the limited Covid-19 data, the infection rates can be difficult to predict and visualize for different states. This poses a question: What if you want to travel to Santa Monica during summer and would like to see the new confirmed rate of Covid-19 in California? With the decrease in Covid-19 cases and the increase in vaccination supplies, many people want to travel in the summer while staying safe. Therefore, the objective of this research project is to predict and visualize Covid-19 related data including mortality rate, hospitalization rate, new confirmed rate, total tested rate, and more for a given state at a given time. This will allow users to be informed about the potency of Covid-19 through insightful visualization for any state, which will allow them to make more suitable decisions and be safer while traveling.

1.2 Challenges

The key challenge we faced was the lack of data in some states and also predicting the rates for missing entries. Since we were trying to predict mortality rate and other specific categories, the limiting factor on the prediction was that we could only predict it based on the features that we could find. While the dataset that we used was large enough to ensure the legitimacy of our prediction, there were some minor inconsistencies between categories when using them to predict. For example, in California it lacks many numbers of data in the `current_intensive_care` feature when compared to other states. In smaller states there are little to no data in the `current_ventilators` feature. Some of the other challenges we had include finding appropriate time series models to use on our dataset and also developing the web application while integrating the machine learning models given our limited computing power in CPUs. Since we were running locally, we had to clean the dataset and also use efficient algorithms to run our code in order to run the models in a timely manner. We also had to conduct extensive research to figure out and implement the appropriate time series model for our dataset.

2 Literature Review

Given the severity of Covid-19 in 2020, there are a number of published reports and papers that discuss, regress, display, and predict quantitative Covid-19 statistics. As this pandemic continues to spread and affect the lives of people worldwide, researchers initially used machine learning algorithms to track and predict Covid-19 outbreaks to have a better understanding and control over the situation [1]. In the early stages of Covid-19, most reports focused on the mortality rate for different states/counties in the United States, with some not even predicting future trends. These reports focused on simply tracking and illustrating Covid-19 data, so that individuals can be kept up to date with the latest quantitative updates. Some geographical displays of Covid-19 data, like the HealthMap AI application, have visual representations of high, medium, and low mortality rates indicated by size and color intensity and mainly focus on total cases and total deaths [1]. Once Covid-19 started to spread throughout the US, the mortality rate started to increase at an exponential rate. Researchers started to apply regression models to predict future trends on quantitative Covid-19 statistics [2]. As vaccinations started to disperse throughout the US, researchers started to include vaccination records as part of their Covid-19 statistical regression model and quantitative representations [3].

In order to best analyze the day-to-day fluctuations in Covid-19 data, time series analyses were used. Time series analyses are well-developed regression models that can be decomposed in order to represent trends in the data [4]. They have been used in a variety of settings, for example, in healthcare settings to predict risk of chronic kidney disease or to forecast sales information based on the season [5, 6]. Most importantly, certain time series models like ARIMA are useful in predicting disease progression in populations [7]. The more data is reported, the more accurate results these models tend to produce, and they can provide indications of variability in future predictions.

With all that has been said, however, every Covid-19 model is informative, useful, and helps individuals keep track of current Covid-19 updates. Finalizing our research, we referred to the best qualities of various reports and models to come up with our final solution.

3 Dataset Description

Our dataset was acquired from the Google Cloud Platform, Covid-19 Open-Data dataset. We chose to use this dataset because it is regularly updated with the latest information from relevant sites like the US Census, the CDC and others. But this dataset had too much data for our website to handle and would take too long to load. We ended up having to trim down this dataset for this project because we do not have access to the hardware required to create a dedicated system for processing this data and making our system widely available. We removed extra data that are not needed and had to resort to using a static dataset that didn't update as time went on so that we would not have issues with the dataset being too large for our computers to handle. To do our data cleaning, we first placed our dataset into a pandas dataframe. Then we dropped all countries except for the US and changed rows with null values to 0. This is an important step because issues pop up if dataset is not processed and clean properly. We also changed how the dates are expressed in the dataframe for easier use in our models. The cleaned dataset columns are date, confirmed, deceased, recovered, tested, hospitalized, intensive care, ventilator, population, gender, population density, and several age ranges. Once the dataset was small enough for our computers to handle and had been cleaned, we could use the data for training models.

4 Proposed Solution

4.1 Website Showcase

To showcase our solution, we created a website so that our data analysis would be available to the general public. Unfortunately, we could only perform data analysis on a local website because we could not find any free website services that would host such a complex real time data analysis. We were able to find a website service www.glitch.com that could do some limited data analysis like simply creating a bar graph of our Python data, but when we tried to use any of the other more complex data analysis tools in Python, the website would freeze up due to limited disk space and slow CPU timing and result in a website crash. Here is a link to a cut-down version of the website so that you can see a bar graph and how the US map, adjustable slider, and drop down tools on the website work. The full version of the website must be run locally. The code can be found on our Github repository linked above.

4.1.1 Website Structure

The website was constructed such that when a plot was made in Python, we could display that graph on the website. We used Python Flask for our web server and html templates for the structure of our web page. A mix of functional programming and classes in Python were used for all of our models. This way we could simply pass the selected state using a map, the day using an adjustable slider, and the selected graph parameter, and we would have our model graphs displayed in a clean way.

4.2 Forecasting Models

To perform time series forecasting on Covid-19 data in different states, we implemented two different regression models. Our first model utilized a Facebook-developed forecasting package called Prophet. Our second model relied on the Python module statsmodels implementation of the time series forecasting ARIMA class.

4.2.1 Prophet's Mathematical Model

The Prophet is a time series regression model composed of three major factors: trend, seasonality, and holidays [8]. This is represented by the following equation $y(t) = g(t) + s(t) + h(t) + \epsilon_t$, where $g(t)$ is the trend function which represents non-periodic changes, $s(t)$ is the seasonality function which models periodic changes, $h(t)$ models the effects of holidays and events, and ϵ_t represents the error term not accounted by the model [8]. The basic mathematical implementations of the trend, seasonality, and holidays are discussed below. The trend function is often modeled with logistic growth [8]: $g(t) = \frac{C(t)}{1+e^{(-k(t-m))}}$, where $C(t)$ is the carrying capacity. An example of carrying capacity applied to our dataset is for the total tested Covid-19 cases in California, the carrying capacity might be all the people who had access to the testing site. This number can always increase in time as more testing sites become available; thus, $C(t)$ is modeled as a time-variant parameter. k is the growth rate, and m is a parameter to offset the time series data [8]. The seasonality function is modeled with Fourier series [8]: $s(t) = \sum_{n=1}^{N}(a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P}))$, where P is the period. An example would be a yearly data would have 365 days as P, while a weekly data would have 7 days as P [8]. N is a parameter to change seasonal patterns [8]. This value is set to 10 and 3 by default for yearly and weekly seasonality [8]. The summation of a_n and b_n can be represented by a column vector β, $[a_1, b_1, a_2, b_2 \ldots a_N, b_N]^T$, where $\beta \sim Normal(0, \sigma^2)$, meaning that β is assumed to be normally distributed with a mean of 0 and σ^2 as the variance [8]. The holiday function is modeled as follows [8]: $h(t) = k[1(t \in D_1), 1(t \in D_2) \ldots, 1(t \in D_N)]$, where D_i represents all the dates for the holidays and commemorative events, and k is the parameter associated with each holiday, where $k \sim Normal(0, \sigma^2)$ [8]. The advantages of using Prophet include working well with large time series data with prominent seasonal effects, and a strong capability of handling missing data and outliers [8].

4.2.2 Picking Between ARMA & ARIMA

After building the Prophet model, we conducted research on other time-series forecast models. This lead us to the Auto-Regressive Moving Average (ARMA) family of models. This family has many variants of models that allow us to forecast the behavior of a variable over time based upon that variable's past values. We had to choose between building a general ARMA model and an Auto-Regressive Integrated Moving Average (ARIMA) model. The deciding factor between the two models was the stationarity of the data we are forecasting. Stationary data is defined as data that whose mean and variance are constant over time. ARMA models require stationary data to create an effective forecast. ARIMA models take non-stationary data and transform it via differencing to make the data stationary.

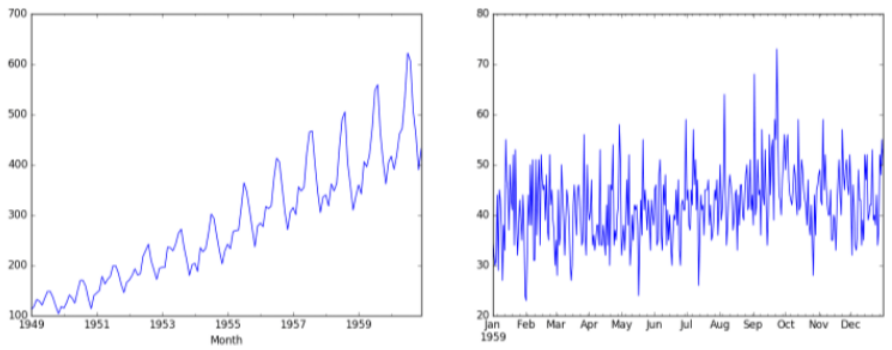


Figure 1: An example of non-stationary data (left) and stationary data (right) [9]

To test whether our data was stationary, we performed an Augmented Dickey-Fuller

(ADF) test. The equation for the test is as follows: $y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} + \ldots + \phi_p \Delta Y_{t-p} + \varepsilon_t$, where y_t is the lag in the time series, ΔY_{t-p} is the difference in the series at the time $t-p$, α is the coefficient for the first lag [10]. The null-hypothesis of the Augmented-Dickey Fuller Test is that the data is stationary which we reject if the data is non-stationary ($\alpha > 0.05$).

4.2.3 ARIMA's Mathematical Model

As discussed further below, we came to the conclusion that ARIMA models would work better than ARMA models. To establish results with the ARIMA model, we must discuss it further. The ARIMA model combines an AutoRegressive equation, which calculates predicted values based on the past values (the "lags" of the past values), and a Moving Average equation, which calculates predicted values based on the lags of previous forecast errors [4]. The modeling equation resembles: $y'_t = c + \phi_1 y'_{t-1} + \ldots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t$, where y'_t is the differenced series, the ϕ terms are the AutoRegressive lags, and the θ terms are the Moving Average lags [4]. The ARIMA model takes three parameters: p, d, q. p and q are the orders of the AutoRegressive and moving average equations respectively. d is the number of differencing terms required to make the data stationary.

5 Experimental Results

5.1 Prophet Model

Datetime from January 2020 to June 2021 is plotted against new confirmed Covid-19 cases for New York in Fig. 2. The forecast starts from the end-date of the dataset, labeled as black points in Fig. 2, 4/26/2021 to 6/15/2021. The blue curve is the prediction made by Prophet, while the light blue region is the lower and upper bounds of the prediction. From Fig. 2, some of the outliers are neglected, and the overall trend is captured by Prophet.

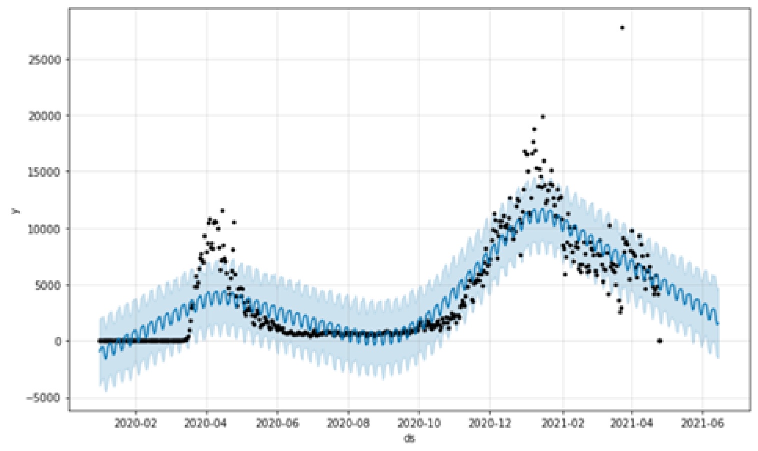
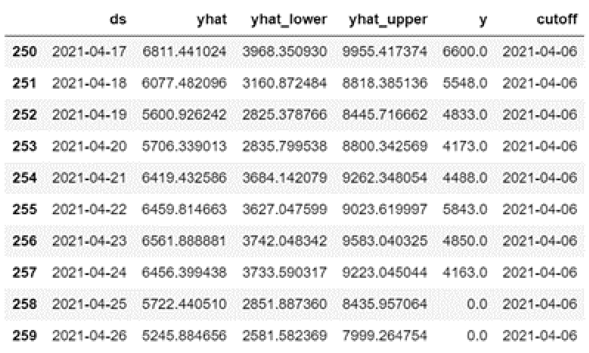
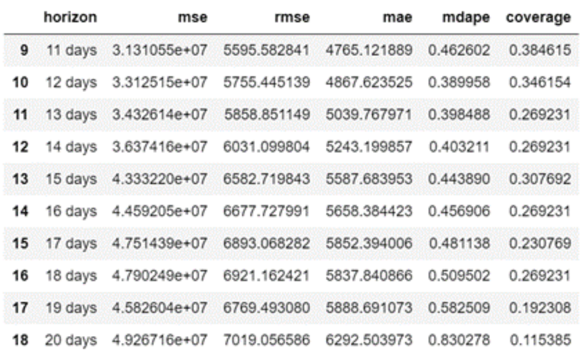


Figure 2: Forecast plot of Datetime vs New Confirmed Covid-19 Cases for New York



	ds	yhat	yhat_lower	yhat_upper	y	cutoff
250	2021-04-17	6811.441024	3968.350930	9955.417374	6600.0	2021-04-06
251	2021-04-18	6077.482096	3160.872484	8818.385136	5548.0	2021-04-06
252	2021-04-19	5600.926242	2825.378766	8445.716662	4833.0	2021-04-06
253	2021-04-20	5706.339013	2835.799538	8800.342569	4173.0	2021-04-06
254	2021-04-21	6419.432586	3684.142079	9262.348054	4488.0	2021-04-06
255	2021-04-22	6459.814663	3627.047599	9023.619997	5843.0	2021-04-06
256	2021-04-23	6561.888881	3742.048342	9583.040325	4850.0	2021-04-06
257	2021-04-24	6456.399438	3733.590317	9223.045044	4163.0	2021-04-06
258	2021-04-25	5722.440510	2851.887360	8435.957064	0.0	2021-04-06
259	2021-04-26	5245.884656	2581.582369	7999.264754	0.0	2021-04-06

Figure 3: Cross Validation Table



	horizon	mse	rmse	mae	mdape	coverage
9	11 days	3.131055e+07	5595.582841	4765.121889	0.462602	0.384615
10	12 days	3.312515e+07	5755.445139	4867.623525	0.389958	0.346154
11	13 days	3.432614e+07	5858.851149	5039.767971	0.398488	0.269231
12	14 days	3.637416e+07	6031.099804	5243.199857	0.403211	0.269231
13	15 days	4.333220e+07	6582.719843	5587.683953	0.443890	0.307692
14	16 days	4.459205e+07	6677.727991	5658.384423	0.456906	0.269231
15	17 days	4.751439e+07	6893.068282	5852.394006	0.481138	0.230769
16	18 days	4.790249e+07	6921.162421	5837.840866	0.509502	0.269231
17	19 days	4.582604e+07	6769.493080	5888.691073	0.582509	0.192308
18	20 days	4.926716e+07	7019.056586	6292.503973	0.830278	0.115385

Figure 4: Performance Metric Table

To access the accuracy of the Prophet's forecast, cross validation is performed and analyzed. The parameters of the cross validation include `initial`, `period`, and `horizon`. Initial is the number of data to be trained on and is set to 336 days, which is equivalent to 70 percent of the total dataset. Horizon is the number of days to forecast after the training data and is set to 20 days [8]. Period is the number of intervals to increment per iteration and is set to 10 days. In the first iteration, the model will train on 336 days and perform forecasting for 20 days. In the second iteration, the period is added to the training data. In other words, the model will train on 346 days and perform forecasting for another 20 days. This iteration process will repeat until the training data has reached 480 days, which is the maximum numbers of data available. The result of the cross validation is a dataframe including all the forecast values for 13 iterations. Only the last iteration and 10 rows will be displayed in Fig. 3.

From Fig. 3, the cutoff is the date that corresponds to the nth day of training data [8]. The cutoff starts from 12/7/2020, which is the 336th day and increments by the value of period to 12/17/2020 for the next iteration. The last iteration of the cutoff is 4/6/2021. The prediction, lower and upper bounds of the prediction, and the original data are all shown in Fig. 3 and denoted as `yhat`, `yhat_lower`, `yhat_upper`, and `y`.

Performance metric is implemented to assess the errors of the cross validation. The errors are averaged over 13 iterations, and a table displaying various errors for the last 10 rows of the 20-day forecast as well as a plot of root mean square error against forecast horizon are shown in Fig. 4 and Fig. 5.

The root mean square error (RMSE) is given by $RMSE = \sqrt{\frac{\sum_{t=1}^{N}(\hat{y}-y)^2}{N}}$ where $\hat{y}$ is the predicted value in Fig. 3, y is the original data in Fig. 3, and N is the number of iterations, which is 13. From Fig. 4, the RMSE is within the range of [4500, 7020]. This large RMSE can be attributed to the large values of the new confirmed Covid-19 cases as well as the uncertainty due to the fluctuations of the trend. Also, as mentioned in section 4.2.1, Prophet works best with large time series dataset which spans over several years, with changing seasonality. Since Covid-19 data is only out for a little over a year and external factors such as the effects of vaccines are not accommodated in the Prophet's regression

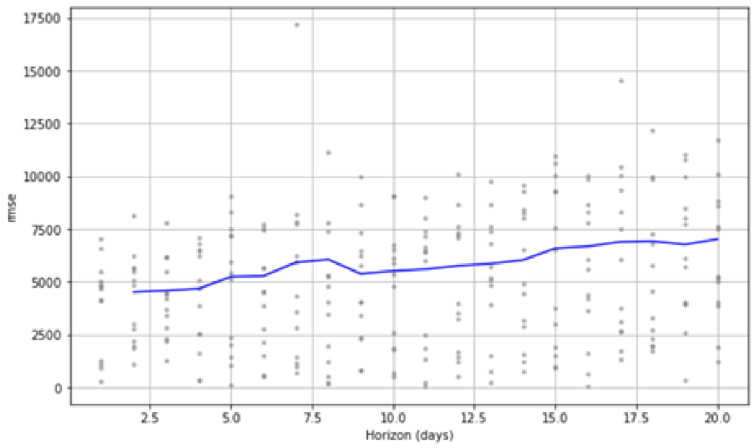


Figure 5: Plot of Forecast Horizon vs RMSE

model, a large RMSE can be comprehensible. Other errors including mean absolute error (MAE), mean square error (MSE), and median absolute percentage error (MDAPE) are illustrated in Fig. 4.

5.2 ARIMA Model

To first pick between the ARMA and ARIMA models, we perform the ADF test. We used a function built into the statsmodels python package to conduct the test on each of our possible input variables for the data, and we found that all of our data returned a $p > .05$. We rejected the null-hypothesis and we now know that our data is stationary, therefore we chose to use an ARIMA model.

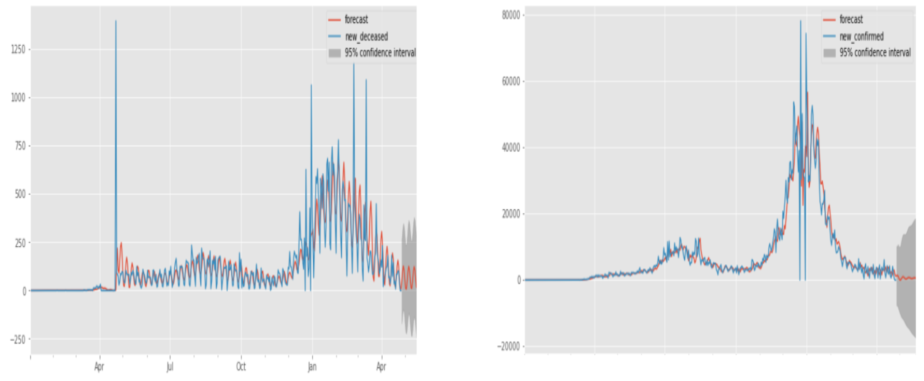


Figure 6: Example ARIMA plots of newly deceased patients (left) and newly confirmed cases (right) for California

To implement the ARIMA model, a pre-built package from Python's statsmodels was used. An automated fitting feature to test the best parameters was used to get values for p, d, and q, based on the lowest Akaike's Information Criterion (AIC) value. We were then able to input these into a forecasting building function that forecasts to the current date. In using forecasts, it is notable that in any function, the further from the data the prediction, the less accurate it is. The ARIMA is well-correlated to most of the data, achieving numbers above .8 or .9 most of the time. Unfortunately, the ARIMA implementation package did not allow for differencing more than twice or $d > 2$, which led to problems in exponential-resembling data, such as in `total_confirmed`. In these cases, ARIMA was not a suitable model and we did not use it for fitting.

6 Conclusion and Discussion

In this project, we chose to use data representing the prevalence of the Covid-19 virus in the United States throughout the pandemic to predict future rates of infection based on multiple factors indicated in our dataset. In order to create models that would give users meaningful insights about Covid-19 rates in different parts of the country, we chose to use the Covid-19 Open-Data dataset to graph Covid-19 data by state. Using the data we parsed down and cleaned from the original dataset, we constructed bar plots, linear regression models, and time series ARIMA and Prophet models that could be displayed on our website. Though we faced many challenges throughout our process, such as accounting for missing data, figuring out what time series models would work best for our data, and deciding which models would display best on our website given the capacity of our local systems, we were able to create a website using Python Flask that would display a number of predictive models. These models show trends that will guide users toward making safe travel decisions as pandemic conditions continue to ease and evolve.

In the future, we hope that this project could be expanded to not only provide predictions by state in the US, but also to show predictions by country around the world, and perhaps even by region or by major city. Then, those using this tool can get the best sense of which route is the safest to travel, given Covid-19 conditions, no matter where they're going. We were unable to do this due to the sheer quantity and scope of the data we were handling given the computers we had at our disposal, but showcasing predictive models like those we've created for the United States on a larger scale could take this practical use of machine learning principles to the next level.

References

[1] Tbilisi and U.S. Embassy. *Tracking Covid-19 with Artificial Intelligence*. July 2020. URL: `https://ge.usembassy.gov/tracking-covid-19-with-artificial-intelligence-july-22`.

[2] Yu. *Covid-19 Severity Prediction*. May 2021. URL: `https://covidseverity.com/`.

[3] Gu. *Covid-19 Projections using Machine Learning*. Mar. 2021. URL: `https://covid19-projections.com/`.

[4] Hyndman and Athanasopoulos. *Forecasting principles and practice*. 2nd. OTexts, 2018.

[5] Perotte et al. "Risk prediction for chronic kidney disease progression using heterogeneous electronic health record data and time series analysis". In: *Journal of the American Medical Informatics Association* 22.4 (Apr. 2015), pp. 872–880. ISSN: 1067-5027. DOI: `10.1093/jamia/ocv024`. eprint: `https://academic.oup.com/jamia/article-pdf/22/4/872/34145809/ocv024.pdf`. URL: `https://doi.org/10.1093/jamia/ocv024`.

[6] Kapoor, Madhok, and Wu. "Modeling and Forecasting Sales Data by Time Series Analysis". In: *Journal of Marketing Research* 18.1 (1981), pp. 94–100. DOI: `10.1177/002224378101800110`. eprint: `https://doi.org/10.1177/002224378101800110`. URL: `https://doi.org/10.1177/002224378101800110`.

[7] Allard. "Use of time-series analysis in infectious disease surveillance". In: *Bulletin of the World Health Organization* 76.4 (1998), pp. 327–333.

[8] Taylor and Letham. *Forecasting at Scale*. Sept. 2017. URL: `https://peerj.com/preprints/3190.pdf#section.1`.

[9] Brownlee. *How to Check if Time Series Data is Stationary with Python*. Aug. 2020. URL: `https://machinelearningmastery.com/time-series-data-stationary-python/`.

[10] Prabhakaran. *Augmented Dickey-Fuller (ADF) Test*. Mar. 2021. URL: `https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/`.