Udacity Machine Learning Nanodegree

Capstone Project

Projecting COVID-19 Contagion

John Marshall

August 1st, 2020

1. Definition

I. Project Overview

In late 2019, a novel infectious disease called Coronavirus disease 2019 (COVID-19) was first identified in Wuhan, China and has since spread and devolved into a global pandemic. The containment of COVID-19 has proved problematic not only because of the prolonged incubation period (10-14 days) but also the fact that asymptomatic individuals can be vectors for the contagion of COVID-19. To highlight the difficulties containing this virus, according to a report by Harvard Medical School, individuals who were exposed four days prior to getting tested still have a 40% chance of receiving a false negative test result (Harvard Medical School *If you've been exposed to the coronavirus*). This has prompted different regions to enact certain policies to prevent COVID-19's proliferation as much as possible. Some of these policies address social interactions and are aimed at prophylactically containing the virus, such as social distancing, mandatory mask wearing, and in some cases, a complete economic shutdown. From a medical perspective, certain regions have taken different steps to contain COVID-19 that include rapidly boosting testing capabilities, hiring and training contact tracing specialists, and setting up additional makeshift medical facilities to provide relief to overburdened medical facilities. I would argue that these policies either fall into two categories: prophylactic containment or diagnostic identification. Depending on the degree of emphasis on either of those two categories, there will be differences not only in actual COVID-19 contagion, but also in the way in which the virus is reported to spread, as represented by the time series.

For these reasons, there is quite a bit of heterogeneity in the data. Part of this comes from the vast array of policies enacted by different regions, but another more challenging component is the accuracy of the data due to discrepancies of testing rates and procedures across the world. I have accessed the data granularly tabulated by Johns Hopkins University and imported downloaded into a comma

separated values file from a Kaggle profile, credit to user Sudalair Rajkumar for providing the file. I have worked with two of the files provided – both are categorized by region, but one aggregates the quantity of confirmed cases while the other tallies COVID-19 deaths. For the duration of this report, I will focus on the machine learning applied to the confirmed cases dataset for three reasons. First and foremost, I think projecting the raw number of confirmed cases has more intrinsic value than projecting mortality totals. An accurate projection of the confirmed cases can allow both governmental agencies and non-governmental organizations to efficiently deploy their resources to limit the outbreak. The secondary rationale is that COVID-19 deaths can be expressed as a mortality coefficient applied to the quantity of confirmed cases, thereby introducing a second layering to the machine learning's estimation task that can obfuscate any fundamental underlying trend. And the final reason, is that many of the results of the analysis for confirmed cases were parallel to the COVID-19 deaths dataset, so to avoid redundancy I will explicate the findings of the confirmed cases dataset.

II. Problem Statement

At a high level, my goal is to accurately project the number of cases over the next several days given a time series for each region. More formally:

Given a time series $\{x^r_1, \ldots, x^r_t\}$ for time $T=(1, \ldots,t)$, and for regions $r=(1,\ldots,n)$, I want to map:

$\{x^r_{t-j}, \ldots, x^r_t\} \rightarrow \{x^r_{t+1}, \ldots, x^r_{t+k}\}$

Where j = context length (i.e. prior period to inform predictions), and k = prediction length.

In practice, this is somewhat problematic due to the data sparsity for this novel virus and the vast number of regions I am attempting to jointly estimate. Given these circumstances, I evaluated the efficacy of Amazon's DeepAR model in comparison to two 'benchmark' models. One is the persistence model, where the hypothesis is that tomorrow's observation will be equal to today's observation. This will suffice as a rudimentary benchmark that DeepAR should easily surpass. The second, slightly more

sophisticated method I employed is an ordinary least squares regression method. I think comparing these three methods will provide a more useful spectrum for evaluation than just a binary comparison.

III. Metrics

For this project, I am going to use the root mean squared error as the valuation metric for DeepAR and for the linear regression. Root mean squared error is calculated as follows:

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\sum_{i=1}^{n} \frac{(Observed_i - predicted_i)^2}{n}}$$

Amazon Sagemaker's 'DeepAR' model has two built-in metrics: root mean squared error and weighted quantile loss. Since regressions are designed to minimize the squared error, I thought that this would be a better metric than weighted quantile loss. Additionally, weighted quantile loss is not an applicable metric for the persistence model, whereas root mean squared error works for it. I did consider using either mean absolute percentage error or mean absolute scaled error, but there were problems with both metrics. The dataset, as you will see below, has a lot of observations of zero confirmed cases for the date range. Mean absolute percentage error is calculated as a percentage of the observations, which makes this infeasible. And mean absolute scaled error has similar problems with this characteristic of the dataset. Therefore, I decided to opt for root mean squared error because it works for all three models and allows me to compare apples to apples.

2. Analysis

I. Data Exploration

After examining the data, I wrote a function to condense the major cleaning steps to improve notebook readability. The original table had two columns for describing the region – one for the 'Province/State' and one for the 'Country/Region'. To condense the information, I filled the NA values

of the 'Province/State' with an empty string before concatenating the two columns into a single 'Country – State' column. After cleaning, my data had 266 distinct regions and 190 observations per region.

Due to the high dimensional nature of the data, it is difficult to condense information by region in a readable format. Therefore, I have summarized the data in two formats: the first method I simply aggregated all the datapoints indiscriminately of the regions and provided summary statistics. The second, slightly convoluted method, I gathered summary statistics for each region, and then treated that array attempt to convey some traits of the data, I initially calculated the summary statistics of each region, and subsequently a summary statistics of those summary statistics. This meta-summary, if you will, is provided below:

Stats	Confirmed_Cases
count	48573.0
mean	332.510365841
std	2636.82632071
min	0.0
25%	0.0
50%	0.0
75%	23.0
max	77255.0

Meta Stats Summary

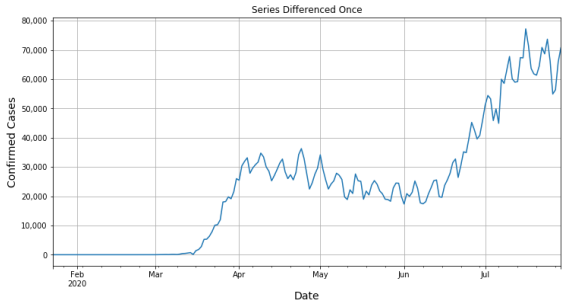
Stats	count	mean	std	min	25%	50%	75%	max
count	266.0	190.0	0.0	190.0	190.0	190.0	190.0	190.0
mean	266.0	17064.0	86010.0	0.0	129.0	675.0	4689.0	1226996.0
std	266.0	19313.0	96236.0	0.0	79.0	572.0	5308.0	1274057.0
min	266.0	2.0	27.0	0.0	0.0	0.0	0.0	444.0
25%	266.0	433.0	4237.0	0.0	0.0	1.0	32.0	67747.0
50%	266.0	11017.0	63904.0	0.0	73.0	415.0	1927.0	954674.0
75%	266.0	28659.0	144021.0	0.0	170.0	1001.0	8299.0	2042654.0
max	266.0	64020.0	335474.0	1.0	267.0	1833.0	17231.0	4426982.0

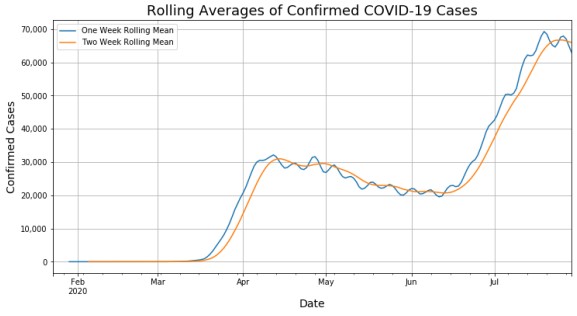
II. Exploratory Visualization

Due to the high-dimensional nature of the data, I opted to investigate a single region and induct upon that region for potential transformations I could subsequently apply to the entire dataset. I used the United States as my guinea pig only because I am more familiar with its trends as a current resident. I confess that I initially intended to utilize some type of ARIMA models to analyze the dataset. Therefore, I tried to use some of these visualizations to determine what transformations might be necessary to create sufficiently stationary time series for an ARIMA model. Despite this original

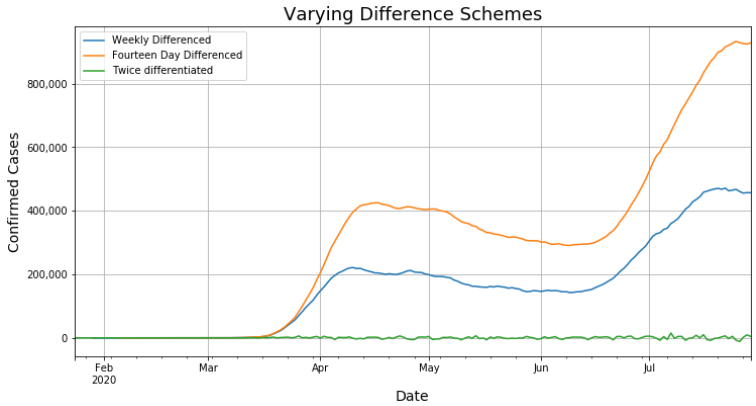
purpose, I left the charts in this report because I think they provide value as an informative graphical tool and helped provide context for setting the context and prediction interval of DeepAR.

I initially plotted the daily time series against the rolling weekly time series and noticed that they were both monotonically increasing. Clearly the data provided a running total, not daily totals, so I differenced the data once to obtain daily observations. After that, I experimented plotting different charts of rolling means for different time frames to get an intuitive feel for how the data is structured. The first chart below is of the one week rolling mean, and the second chart shows the two-week rolling mean in addition.

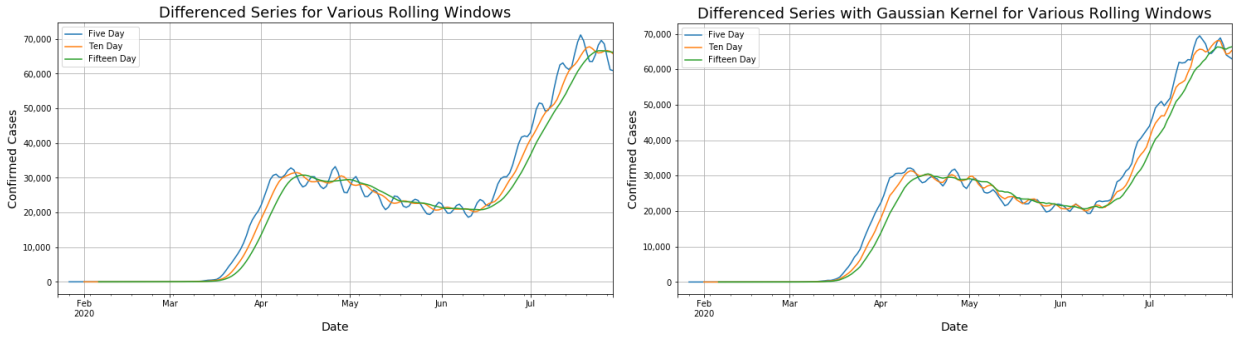




From these charts one can see there is still quite a bit of variability and nontrivial trends. The next chart plots the weekly differential, the fourteen-day differential, and the series twice daily differentiated:

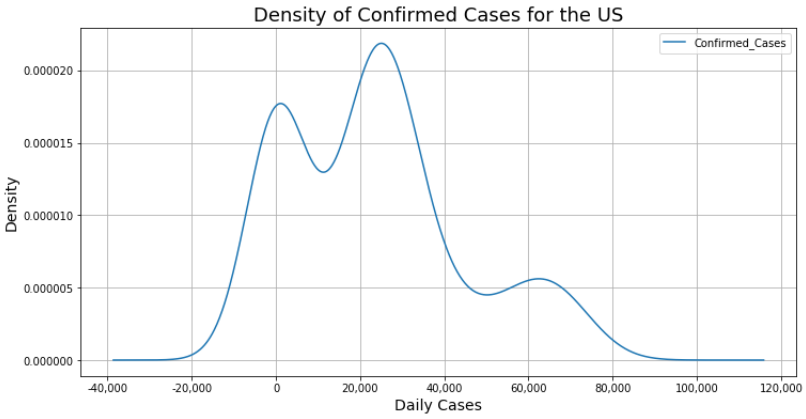


From there, I decided to investigate systematically spaced rolling windows and compare it to an identical chart with a Gaussian kernel:

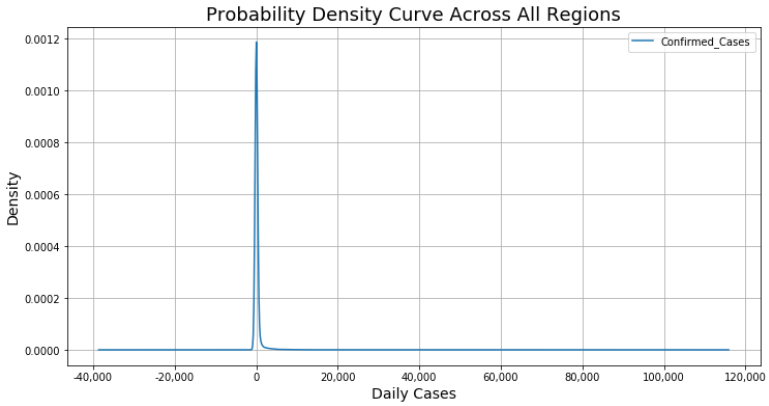


From here, it became clear that outside of differentiating the series twice, it would be difficult to create a stationary time series. And differentiating these time series twice really compromises the interpretability of any future model fits, so I abstained from using this transformation. The lack of stationarity makes intuitive sense given the nascent nature of COVID-19, nonetheless I thought it would be an endeavor worth exploring.

To provide additional visuals of the type of distribution these time series have, I have plotted a probability density function of the United States Confirmed Cases below:



When I generalized this for every observation in my dataset, the results were informative despite the absence of aesthetic appeal:



Despite spreading across the world, most regions appear to be only marginally affected. This could be due to prophylactic measures taken when the virus first spread, or because of a lack of testing and reporting. The proportion of observations that have zero confirmed cases represents 53.5% of the dataset. Since regions most acutely affected are the outliers of this dataset, I expect models to have a lot of difficulty fitting these regions and the coefficient estimates attributed to these regions to be nontrivial.

III. Algorithms and Techniques

After a few visualizations, I decided to dive into creating the benchmark model and the ordinary least squares linear regression. The persistence model is trivial to evaluate, but for the linear regression I did have to restructure my data. The data frame is not properly structured to run a linear regression that jointly estimates the coefficient estimates for observation at time *t-1* and for region *r*, so I had to fix this issue. To do so, I iterated through each column (region), then I sliced that column as a pandas' series. After that, I created an additional column equal to the country name and created a two-column data frame and then appended that data frame to a common data frame along the rows. That way, I was able to have a data frame of two columns, one column of the observations and a second column of the which country the observation corresponds. From there, I changed the column to a categorical label and appended an array of dummy variables corresponding to each unique country. After that, I ran my

linear regression that jointly estimated the value for the subsequent day and the linear impact each individual country contributes.

From here, I decided to partition 80% of the dataset for training and leave the remaining 20% for testing. Since there are only 190 observations per time series and the model is trying to fit 266 binary estimates along with the previous value coefficient, I thought it would be prudent to allocate more data to training and fitting the model correctly. I thought that I could create a validation dataset as time passes, so one wasn't completely necessary at this juncture.

IV. Benchmark

After splitting the dataset into a training set and a testing set, I fit a regression on the data and the root mean squared error of the training set was 63.43. But when I applied the predictor to the test set, the root mean squared error jumped to 602.31 which suggests overfitting. When I fit the persistence model on the entire dataset, the aggregate root mean square was 561.65, which did surprise me. Ultimately, I think this result can be attributed to the significant proportion of observations of no confirmed cases.

3. Methodology

I. Data Preprocessing

When I explored the data, I noticed that there were several instances where the differenced series yielded negative values for specific dates. There was a total of 78 observations of negative confirmed cases on particular dates, and a total of 38 unique regions contributed to this defect. DeepAR takes in observations over several days as a single datapoint, and projects values over subsequent days. So, a single missing date creates a void that is felt over several days, nontrivially shrinking the dataset. Therefore, I didn't want to discard these single observations, but I also didn't want to discard entire regions. Ultimately, I set a somewhat arbitrary cutoff value of negative fifty, and any region that

contained observations less than that I discarded. For all the negative observations greater than negative fifty, I created synthetic values by taking an average of the observation the day before and the day after. While less than ideal, I think it was the best compromise to not create too many synthetic values simultaneously while avoiding discarding too many regions. Spain, for example, had the largest negative difference in the dataset which came in at -10,034 confirmed cases on April 24th. I felt uncomfortable taking an average of the two adjacent datapoints since the discrepancy between the real number of observations and my estimate could significantly differ and have substantial impacts on fitting the training data. Ultimately, I dropped eight regions and replaced the remaining faulty datapoints.

For the DeepAR setup, I had to take a few additional steps to prepare the data. If you include a categorical variable, one of the requirements is that each categorical variable (region) is included in the training dataset. The reason that this is problematic for DeepAR, is that you need to provide a value for a context interval and a prediction interval. The context interval informs DeepAR as to how many days the model should use as inputs to make predictions. The prediction interval informs DeepAR how many days ahead it should predict values. Initially, I set the context interval for 21 days and the prediction interval for one week for two reasons. The primary reason is that COVID-19's incubation period is 10-14 days so I thought it would be ideal to allow DeepAR to look backwards for up to 150% of the incubation period in case testing limitations prevented the true number of cases to be reported. Secondly, I thought it would be best to provide longer periods for DeepAR to learn from and allow the algorithm to gradually narrow down the ideal context interval. The issue with this setup is that a single datapoint is represented by the sum of the context interval and the prediction interval, which means that a 28-day stretch represented a single datapoint. Therefore, there were only six datapoints per time series which meant that there was a nontrivial chance that a random sample of 80% of the dataset would miss an entire region, rendering the model unable to estimate that categorical variable or provide predictions

for the test set. Because of this issue, I ended up writing a function to iterate through each unique region and randomly sample 80% of that specific region's datapoints and subsequently append that to an aggregate training list.

The next step required me to transform these Pandas data frames into a JavaScript Object Notation (JSON) file. A JSON file basically requires the data to be stored in a dictionary format, so I iterated through each time series and matched the temporary category variable with the string 'cat', matched the first item in the datetime index with the 'start', and finally matched the datapoint with the string 'target'. After iterating through the entire training and testing set, I saved those files locally and prepared them for Sagemaker.

II. Implementation

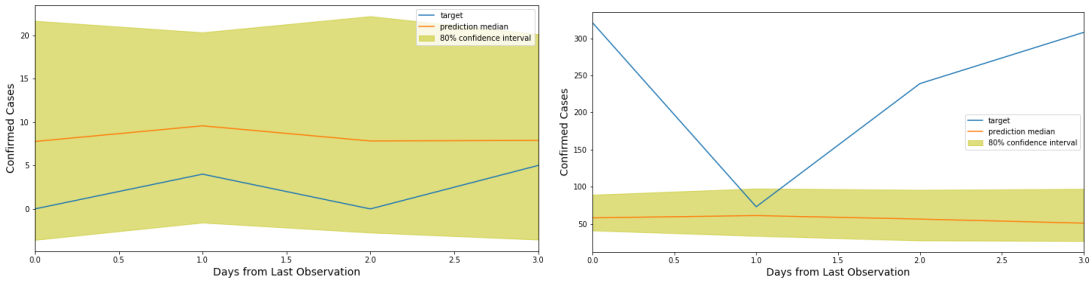
The next step required me to setup the necessary Sagemaker session, container, and estimators. After initializing these, I set a few hyperparameters for the initial estimator. Then I set the location for the training path and the testing path and finally fit my estimator. I figured it would be incredibly naïve if I thought I could a priori determine the optimal hyperparameters to arrive at the best model, so my next step was to employ Sagemaker's hyperparameter tuner. After some additional research and time reviewing Udacity's sample notebooks, I tried a reasonable range for each of the hyperparameters.

The next task was to evaluate the best model returned from the hyperparameter tuning of DeepAR, so I attached the best model to a new Sagemaker estimator and documented the hyperparameters, which are presented below:

Hyperparameters for Best Model

Metrics	Values
_tuning_objective_metric	test:RMSE
cardinality	[257]
context_length	6
early_stopping_patience	10
epochs	59
learning_rate	0.01
mini_batch_size	66
num_cells	93
num_layers	1
prediction_length	4
time_freq	D

Now that I had a record of the best job, I wanted to generate some predictions from the test set and generate some graphical displays. I have attached a sample for one datapoint below:



III. Refinement

After running twenty hyperparameter tuning models, I decided to evaluate predictions for specific intervals by plotting the observations along with the mean prediction and an 80% confidence interval filled in. With the initial parameterizations, the confidence intervals were either comically wide or grossly insufficient, as displayed above. Therefore, I decided I needed to rethink my strategy. I figured the lack of datapoints used to create an effective dataset for the DeepAR estimator was a likely culprit, so I significantly pared down both the context interval and the prediction interval.

After further tinkering, I settled on a context length of seven days and a prediction length of four days. This provided 17 'datapoints' (17 separate instances of 11 consecutive days of observations) per time series. After some additional experimentation, I ultimately fit 84 different models. I came to this figure by running four different experimentations fitting both the estimator and 20 concurrent hyperparameter tuning jobs.

4. Results

I. Model Evaluation and Validation

Despite some lackluster models, I thought it would be appropriate if I reported the ten best root mean squared error models along with the ten worst to give an idea for the spectrum. I provided it below:

DeepAR Best and Worst Estimators Summary

Training Job Name	Batch Size	RMSE
forecasting-deepar-200731-1633-010-ba333c5e	206	325.485656738
forecasting-deepar-200731-1633-015-4a552225	113	334.954925537
forecasting-deepar-200731-1633-014-7df3e0c9	85	359.503509521
forecasting-deepar-200731-1633-012-c25cefa2	97	394.54397583
forecasting-deepar-200731-1633-016-cdc478d1	163	406.174713135
forecasting-deepar-200731-1633-008-f77a4ad1	235	450.340789795
forecasting-deepar-200731-1633-003-f9fa698d	174	453.996643066
forecasting-deepar-200731-1633-013-d6253b4a	80	459.12677002
forecasting-deepar-200731-0244-008-97f9cc58	242	459.56060791
forecasting-deepar-200801-1551-017-2e207515	79	1118.19824219
forecasting-deepar-200731-1633-002-1a2ed9da	163	1254.03234863
forecasting-deepar-200731-0038-016-590bf6e5	207	1311.62976074
forecasting-deepar-200731-1633-011-1709b816	254	1371.70678711
forecasting-deepar-200731-1633-005-b733d101	155	1406.41052246
forecasting-deepar-200801-1551-003-f9cd2e26	168	1572.7980957
forecasting-deepar-200801-1551-004-5e3ae973	129	1955.58508301
forecasting-deepar-200731-0244-001-2fc0c305	156	2583.53417969
forecasting-deepar-200731-0244-003-ea2f0206	77	4837.50488281
forecasting-deepar-200731-0038-011-f91b00c0	176	5846.38330078

As you can see, there is an exceptionally wide range of outcome for these different results. The top ten models were all superior to both the persistence and the linear regression model. But at the other end, the worst models root mean squared error was ten times that of the persistence model's root mean squared error. I think there are several things to take away from all these results. The first thing I'll state is a reminder that garbage inputs results in garbage output, no matter how sophisticated the algorithm. I researched DeepAR for clues to figure out the optimal context and prediction interval, but there is an extremely limited amount of data on its optimization given how recently this algorithm was developed by Amazon researchers. I tried to rely on intuition and logic based on innate characteristics of COVID-19, but ultimately this proved foolhardy as this significantly handicapped DeepAR's optimization algorithm given the limited samples it was working with. I think this provides evidentiary support for the fact that it is more important to tailor the data appropriately for the algorithm instead of intuitions about the dataset.

II. Justification

I need to provide one significant caveat before I proceed. I want to remind the differences in the algorithm's I considered: both the linear regression and the persistence model projected one day forward whereas the DeepAR predicted four days forward. I understand that this is inconsistent, but I

wanted this project to replicate a real-world problem and predicting one day out has trivial value for organizations like the World Health Organization, since information over a longer time horizon significantly helps to contain outbreaks.

For the models, I think it is inappropriate to conclude that the DeepAR performed better or worse than the benchmark given the number of models I estimated. But upon review, about a quarter of the estimated DeepAR models outperformed the persistence model and roughly 30% outperformed the linear regression. But, the top DeepAR model performed at root mean squared error 40% less than that of the persistence model which suggests further refinement and more technical knowledge of the discipline might provide for clearer insights. My gut reaction is that with optimal use DeepAR would significantly outperform the persistence model and the linear regression model, but without a validation set this is difficult to prove. I considered rewriting the test train split into a test train validation split, but if I split it 60%/20%/20%, that only provides ten datapoints per region for training and three or four for testing and validation which seems rather meager for a machine learning algorithm, not only for testing but also for evaluation. In two months, I will have an additional five datapoints per region that I could use as validation, but I am unable to evaluate at this time.

5. Conclusion

I. Free-Form Visualization

I think one of the most telling visualizations about the dataset was the density plot of all the regional observations combined. It's clear that there is this bimodal distribution, those regions where the virus initially spread but was immediately contained, and those regions where the virus was able to contaminate different social pools and reverberate around the region. Despite the vast number of regions impacted by this virus, most cases are from a few select regions most afflicted by the outbreak.

The second thing that stood out to me, was when I tried to look for trends in the case study of the United States' confirmed case totals. It seemed like there were two different regimes, or waves if you will, that made it through the United States. There are several potentially confounding variables, not only from inadequate testing but also from the discrepancies between the true number of cases and the total number of reported cases based on positive test results. And when you combine that with the fact that a nontrivial more contagious 'G' strain emerged from Europe and quickly replaced the original strain from Wuhan as the predominant strain, it is difficult to attribute the second wave to one of these characteristics not explicitly stated in the dataset.

II. Reflection

This project idea materialized organically since the world became paralyzed in the midst of this pandemic, so it felt like a prime opportunity to apply a complex machine learning algorithm to a nascent real-world problem without a ton of guidance. I knew this would be a challenging project for me, not only because of my lack of infectious diseases expertise but also due to my limited experience working with machine learning models. Thus, this project required quite a bit of research for me on both these fronts.

After experimenting with the data to become more familiar, one of the first things I realized I needed to do, was create an appropriate counterfactual model to the DeepAR estimator. While I have a background in statistics, I wasn't familiar with methods forecasting longitudinal data. This naturally led to prolonged research that didn't always yield the model I looked for, but nonetheless I learned more about different families of estimators. I decided to use the linear regression because of its simple elegance and interpretability. When I started researching different estimators, I came across four different estimators that I considered using, but when I worked through my case study, the United States, it became clear that two of the metrics (mean absolute scaled error and mean absolute percentage error) would not be efficacious evaluators for the datasets I used. Since over half the all the

observations of confirmed cases were zero across all regions, this made me realize the issues these estimators posed. Thus, when I investigated objective tuning metrics used to evaluate Sagemaker's DeepAR hyperparameter tuner, root mean squared error seemed to be the optimal metric to be able to provide natural cross estimator comparisons.

As I stated earlier, I didn't want to conclude that DeepAR proved to be a better estimator due to the absence of a validation test set, but it is something that made me reevaluate how best to partition the dataset. Like most real-world problems, a limited dataset with underlying confounding variables required me to compromise on the prototypical steps to building and evaluating a machine learning model. While the results of DeepAR were promising, I can't consider this in a vacuum since the root mean squared error of the best model is monotonically decreasing with each additional model estimated. The only caveat that I will add, is that if I had more experience with DeepAR, then I might be able to produce better models without fitting as many as I did by more appropriately restricting the hyperparameter space.

All in all, I was slightly disappointed at first that my DeepAR models were not clearly superior to the benchmarks, but I had to remind myself that this is part of the machine learning process. Nonetheless, I think it is a great lesson and reminder of the limitations of machine learning models. After a great deal of research, cleaning, and refinement I was only able to slightly improve upon the benchmark just emphasizes the fact that the world we live in is not black and white but colored with so many different hues, and that it is a nontrivial endeavor to capture that reality in a constructed algorithm, no matter the level of sophistication of these algorithms. But most importantly, I think I developed skills and learned a lot from working through this process from start to finish.

III. Improvement

There are a lot of ways this model could be improved, but I think the most effective way would be to create additional categorical variables that correspond to the type of policies different sovereignties enforced, and another quantitative metric that describes how much testing is being done, potentially as a proportion of the population. This would require copious amounts of digging to assemble this data for every region in the dataset, if this data is even available. It wouldn't surprise me if certain governments restricted access to this information which would undermine this more robust dataset initiative. The only other way to build a more robust dataset is to wait for this pandemic to propagate around the world. But accumulating data without making attempts to estimate and make projections is a poor attitude given the time sensitive nature of this pandemic and the severity outbreaks to a vast number of regions around the globe. I had a thought when investigating the data that I should start these datasets when the first case presented itself, but I was concerned that I would shrink and already limited dataset.

Works Cited

Adhanom, Tedros. “WHO Director-General's Opening Remarks at the Media Briefing on COVID-19 - 11 March 2020.” *World Health Organization*, World Health Organization, 11 Mar. 2020, www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020.

Hopkins, Johns. “Covid Dashboard.” *ArcGIS Dashboards*, 30 May 2020, www.arcgis.com/apps/opsdashboard/index.html.

Srk, SRK. “Novel Corona Virus 2019 Dataset.” *Kaggle*, 29 May 2020, www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset?select=time_series_covid_19_confirmed.csv.

Srk, SRK. “Novel Corona Virus 2019 Dataset.” *Kaggle*, 29 May 2020, www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset?select=time_series_covid_19_deaths.csv.

Publishing, Harvard Health. “If You've Been Exposed to the Coronavirus.” *Harvard Health*, 30 July 2020, www.health.harvard.edu/diseases-and-conditions/if-youve-been-exposed-to-the-coronavirus.