Team 6 Final Report

Ziyan Wang
University of California, Los Angeles
805216468
jasonwang0817@outlook.com

Tianchen Wang
University of California, Los Angeles
905331972
tanwangtianchen@gmail.com

Xuan Peng
University of California, Los Angeles
705185066
syeaziran@gmail.com

Yuetian Sun
University of California, Los Angeles
605175082
sunyuetian1999@gmail.com

Wenxuan Liu
University of California, Los Angeles
805152602
allenliux01@gmail.com

ABSTRACT

Some traditional methods specializing on mining pandemic data predict the trend of the disease by exploring the differential relationships between the susceptible, the exposed, and the infected, and the recovered. However, these methods are generally based on the assumptions that is at odd with COVID-19, and require full exposure to data, for some of which are not reported by many states.

In our paper, we proposed some alternative but potentially more powerful approaches: instead of only focusing on the relationship between four time-dependent variables exploited by traditional methods, we smoothed the data in advance to make the model behave more properly, and then left out some data that we believed to be not reliable, but added certain features that are conducive to forecast the trend (i.e. mobility). Also, without assuming a differential correlation, we separated each feature as a time series sequential data, and attempted to explore it with different models. Finally, we found using other features as a verification for trend, in addition to the univariate regression on each predicted quantity after smoothing and selection, as the best method to describe the historical data and forecast the future.

Keywords. *COVID-19, LSTM, ARIMA, Prophet*

1 INTRODUCTION

COVID-19 is a kind of respiratory disease that may causes an infection in the nose, sinuses, or upper throat. While it may be less dangerous than other viruses, it is unprecedentedly contagious. Starting its first outbreak in December 2019, the COVID-19 has rapidly spread to nearly every corner of the world, especially to US. There are up to 16.8 million suffered from COVID-19 and over 300 thousand deaths. Nevertheless, it shows no signs of halting, but surges exponentially. The daily confirmed cases increase from several thousand in April and May, to around 200,000 recently. It impacts every aspect of people's lives, both on daily lives and the whole economy: schools and other public facilities are closed; people are required to wear masks; tourism and other industries are largely diminished; and world trade is disrupted. COVID-19 has a such great influence that we must pay attention to it.

Therefore, it is important to predict the trend of COVID-19, since accurately forecasting the progression of COVID-19 can help government monitor and take actions to combat it. If the rate of spreading exceeds expectations, government may enforce stricter lockdown and provide more facilities for prevention. On the other hand, if it does not spread rapidly, government may ease the restriction to revitalize the businesses.

Some of the most popular models on disease pattern discovery, including SIR and SEIR, consider the relative relationship between the susceptible groups, the exposed individuals, the infected, and the recovered. They generally work well, but it fails to produce the most desirable results in our cases. Firstly, these models depend on unrealistic assumptions that recovered population will not be infected again, but evidences show that it is not true. Also, many states do not report daily recovery and the susceptible and exposed population are hard to estimate, which further undermine the power of the model. Even though these problems can be solved, it fails to account for the surge of infected population in the beginning.

In order to compensate for the incompleteness of the data, and to design a model best fit for COVID-19, we attempted several models to explore the spreading of coronavirus, including Prophet (a time series data discovering model developed by Facebook), LSTM (Long short-term memory, a specially designed neural network for sequence data), and ARIMA (Auto Regressive Integrated Moving Average, a model featuring on the seasonality of the sequence data itself). These models focus on the Confirmed and Deaths themselves, and we added additional regressors to utilize other complete data. In the end, we concluded that linear model best utilized the data given and produces the most reasonable results.

The remaining of the paper is organized as follows. Section 2 introduces the related work associated with our topics, and section 3 illustrates the research topic in a formalized way. In section 4, we introduce how we pre-processed the data. Section 5 demonstrates the methodology of different models and section 6 evaluates their results respectively. We conclude our work in section 7.

2 RELATED WORK

2.1 Covid-19 Prediction Techniques

According to current data, there have been about 72.5 million cases and 1.62 million deaths due to COVID-19 all over the world. Governments have tried to contain the spread of virus by various surveillance and monitoring technologies as well as using collected data to do prediction to provide a basis for future epidemic prevention work. Data mining techniques has penetrated into our daily lives with many successes and it will continue to help people fighting against COVID-19.

To get a clearer picture of how we can do the Covid-19 prediction using data mining techniques, we explored some papers and one of

them gave us a direction. In " COVID-19 Outbreak Data Analysis and Prediction Modeling Using Data Mining Technique", Tajebe analyzed some COVID-19 prediction experiences and pointed out some models that relatively produce better results of COVID-19 prediction. So far, the most widely used models for COVID-19 are time series forecasting methods since it comprises methods for analyzing time-series data to extract meaningful statistics and other characteristics of the data. In Tajebe's research, he did prediction of COVID-19 confirmed, recovered and death cases worldwide with Prophet base model, which drove us to dig more on the timeseries prediction models like Prophet and Arima.

2.2 Arima

Due to the functionalities for prediction time series data, Arima has been widely used for detecting outbreaks of infectious diseases [1]. Also, previous researches have been done with Arima's prediction of COVID-19 trend, and we have the best hyper-parameters for U.S. in general which is p, q, d equal to 1, 2, and 1 as attempted by the Arima model for the prediction of confirmed and deaths data during June and July[2].

2.3 Prophet by Facebook

As we are dealing with a time-series regression, a lesser-known yet powerful library that we discovered was the Prophet model proposed by Facebook in an article in 2017, which was originally created for business usage on easy, automatic time-series forecasting for domain experts. According to the paper, the model is described as a Generalized additive model (GAM), that aims at reframing the time-series forecasting problem as a curve-fitting exercise (thus discriminative in nature), unlike other generative models (e.g. ARIMA) that are commonly used. More specifically, the Prophet model was constructed with three components in mind, Trend, Seasonality and Holidays, such that final predictions are given as: $y(t) = g(t) + s(t) + h(t) + \varepsilon_t$, where g(t) measures trend, s(t) measures seasonality, h(t) for holidays and ε_t for random errors/noise.

Together, the three major parts define the components used for the prediction task, in which we will go into details of how to treat and selectively utilize useful features from it. The model also adds a self-correction mechanism called "Simulated Historical Forecasts (SHF)", such that it partitions and evaluates parameter choices for couple candidate models created before a key date, in which data after the key date is used as a validation set to optimize its choice of hyper-parameters. The particular construct of this model has inspired us then to improve our existing models through techniques that tackle with trend verification and smoothing, for which we will introduce. Furthermore, we exploited and extracted its seasonality detection as an additional feature to be used for our model.

3 PROBLEM DEFINITION AND FORMULATION

We labeled $day_t = day_i | i = 0 \ldots k$ to represent the t_{th} day. day0 denotes the 4-12-2020, the start day that our data is based on. For round 1, we were given the data from 4-12-2020 to 8-31-2020 to predict the cases ranging from 9-1-2020 to 9-26-2020. More formally, we would like to make predictions on the data of day_{142} to day_{167} (since there are 142 days between 4-12-2020 and 8-31-2020), based on the data of $day_0 \ldots day_{142}$. Also, we used the data of $day_0 \ldots day_{224}$ (from 4-12-2020 to 11-22-2020) to forecast day_{248} to day_{254} (from 12-14-2020 to 12-20-2020) for round 2.

For each day, it contains the following feature vector:

Symbols	Meaning
C_t	number of confirmed cases at the t^{th} day.
D_t	number of deaths at the t^{th} day.
A_t	number of active cases at the t^{th} day.
R_t	number of recovered cases at the t^{th} day.
PT_t	people tested at the t^{th} day.
PH_t	people hospitalized at the t^{th} day.
MI_t	move in mobility at the t^{th} day.
MO_t	move out mobility at the t^{th} day.
IS_t	in state mobility at the t^{th} day.

Our objective is to make our prediction as close to real value as possible. The "affinity" is defined as MAPE (Mean Squared Average Error).

$$MAPE = \frac{|Predicted - Truth|}{Truth} (1)$$

The closer the predicted value to the the truth, the lower the MAPE. If predicted values completely matches with the truth, MAPE approaches to 0.

We scraped the real value for round 1 for evaluation purposes in the paper, and we aim to develop a model that effectively exploits the features of each day to lower the prediction MAPE while not be prone to overfitting. In contrast, the real value of round 2 is not given, so we attempted to modify the model of round 1 slightly in order to accommodate for the data after round 1.

4 DATA PROCESSING & TRANSFORMATION

During the pre-processing stage, we checked that for features such as Hospitalization_Rate, People_Hospitalized, and Recovered, there existed numerous NAs (i.e. missing values) that we have to deal with. We filled all these blanks with 0, as graphing these features show that the missing values would best correspond to values of 0. For Recovered, we also see that a simple calculation of total - active - death = recovered would show that the days of missing recovered data should have values of 0. Also, we had a glimpse on the relationship between cases of each state with related attributes (especially the date) to get a sense of which model may be appropriate for the dataset.

In addition to a crude lookup on the trends, we also noticed that some attributes are dependent on each other. For example, the "hospitalization rate" is simply "People_Hospitalized" divided by "Confirmed_cases". Therefore, we only need to put two of "hospitalization rate", "people_hospitalized", and "confirmed_cases" into our model. It is also true for other "rates". Therefore, we omitted some features when training the model.

Besides leaving out certain attributes, we also conducted a transformation on the mobility data. We simplified the large dataframe through aggregation, by separating its data into three categories: move-in daily mobility, move-out daily mobility, and in-state daily mobility. We calculated move-in mobility by summing up all mobility data with the same target state, and obtained the move-out mobility by adding the ones with the same source state (for both cases, the data

that target state is identical to source state is not counted). For in-state mobility data, we extracted the rows with the same source and target state. We did these because we do not see a clear correlation between people moving in, moving out, and moving inside the state, different from the linear relationship between hospitalization rate and people_hospitalized. Therefore, we considered these as three independent attributes. Also, we believed that the total confirmed or death cases is independent with from which state people come in and to which states people leave for. Thus, we treated them as the same, and added them to regard them as a single attribute, which decrease the complexity of the model and make it less prone to overfitting.

In this project, we are supposed to find the trend of the data. Noise from the training data will affect the correctness of models especially when dataset is small. Without affecting the trend of the data, we conducted smoothing on the data by applying Savitzky-Golay filter to remove noise and reduce the harm from over-fitting.

Besides smoothing, we also projected each feature into a polynomial degree (in our model it's 3) to enable us to find a polynomial relationship between cases and pertinent attributes. It will be introduced in detail in the following parts.

5 METHODS

In this section, we will briefly talk about some models in attempts to predict the future confirmed cases and deaths trends.

5.1 LSTM

Our LSTM models are trained with 3 features: date, confirmed, and deaths number, and all grouped by state. For each state and its corresponding confirmed or deaths number, we fit an independent model for it so that they have unique parameters to approximate the curve. With Keras sequential deep neural network framework, we tried hidden layers with number of layers ranging from 1 to 5 and each with units ranging from 10 to 1000. The best result with lowest MAPE score turned out to come from a model with two hidden LSTM layers each with 100 and 50 units, with a dense layer for generating output, and the time seq was set to 50 when packing the data for LSTM.

5.2 ARIMA

ARIMA, i.e. Auto Regressive Integrated Moving Average, is a model that can predict time series data based on the past values, including its own lags and the lagged forecast errors. Basically, it needs three vital hyperparameters: p as the order of the AR term, q as the order of the MA term, and d as the number of differencing required to make the time series stationary[3]. In order to find the best hyperparameter for each state and each features(confirmed and deaths number), we have constructed automatic trials of each hyperparameters on all the states with p, q, d each ranges from 1 to 3, which are 27 groups of parameters for 50 states in total, separately considered for round 1 and round 2. Afterwards, we used the average P-values for coefficients ($P >= ||z||$) as our evaluation criteria for selecting the best parameters such that the lower of average $P >= ||z||$, the better converge on the curve and the better performance of the prediction. Thus, we have the best parameters for each state and feature, and then use the parameter to fit the ARIMA model for each state again and make the prediction to get the final result for both round 1 and round 2.

5.3 Prophet

As Prophet is a GAM model, it can be separated in parts for easier interpretability. The first component, Trend, captures the trend of the data by utilizing two major components: i. a saturating-growth model, and ii. a piece-wise linear model. The saturating-growth model, as its name, measured the part of the data that is saturating, i.e. converging to a carrying capacity. But unlike simple saturating-growth, the model assumes a non-stationary carrying capacity that is recalculated after some periods by observing trends in the data. The model first calculates change points, defined as timestamps in history of which the target data has shown a changing carrying capacity. Then, it learns and models the frequency and magnitude of rate of changes of the data at those change points for later usage. Assuming that the future will see the same average frequency and magnitude of rate changes in history, the model later selectively generates change points for future prediction as a type of trend adjustment according to the learnt pattern. For the second part, the piece-wise linear model is simply a linear regression model that fits a best-fit line for periods of no saturating growth that fill in the gaps between the change points. Together, the two components define the Trend component, i.e. g(t), of the GAM.

Then, the Seasonality component uses Fourier series as a model of periodic effects seen in the data, with normally distributed parameters for adjusting the fitting of the Fourier series. The choices of the parameters are automated through calculation of AIC (Akaike information criterion). Furthermore, the Fourier series included easily an adjustable parameter (P) for learning and checking periodic trends (e.g., P = k, where k is user specified or fixed if chosen from weekly = 7, monthly = 30 or yearly = 365.5) that allows for a further breakdown and comparison of trends by week, month or year, which are often seen in real-world data generated by people.

The Holiday component included information about user-defined dates as "Holidays", or key special dates that the model pays specific attention to by drawing data from the same date from previous years or other special handling to model for holiday changes (e.g., Christmas sale for commercial data), which we did not use for our purposes here.

As the Prophet model frames the time-series prediction problem more as a curve-fitting exercise that was in line with our previously best-performing model of linear regression, but incorporated with additional details of adjustable seasonalities, and change points for trend adjustment, we were interested in seeing how this model might enlighten us on what to improve further and if seasonality can be better explored on a different angle than the ARIMA model.

For implementing the Prophet model, we utilized the ready-made package Prophet produced by Facebook for fitting a Prophet model on our training dataset. Noticing its advantage in capturing seasonalities, identifying segments of trend and convenience of decomposing seasonalities as features that can be further utilized, we incorporated the Prophet model as a "knowledge-feeder" model for improving others, besides making its own predictions. However, as it is more of a black-box algorithm despite the basic building blocks illustrated by the paper, we tried to first use it as a evaluating

criterion for our other models to identify trends that can be better captured (through our other models), and possibly detecting states that had more "problematic" seasonalities that can be identified and improved on by incorporating additional information or techniques (e.g. smaller states that saw a later wave of rising cases, which will expanded below), before finally incorporating it into our prediction pipeline.

5.4 Linear and Polynomial Models

The best result of our team so far comes from the linear model with L2 normalization (Ridge). By observing the plots of 'confirmed cases' and 'deaths' of each state, we found that most of the trends of the curves remain the same in the last month. Moreover, the variation of gradient is small, some of the curves even remain a straight line. Therefore, we proposed that only the last few data should be retained, and the Linear model and Polynomial model might work well. Since the size of training data decrease to a very small number, we conducted the smoothing on data and applied Ridge to relieve the overfitting issue. As a starting experiment, we applied the same setting (data, smoothing window size, hyperparameter of Ridge) to all states. The best results of Linear model and polynomial are close (2.28 for linear. 2.30 for polynomial). So far, the linear model is slightly better than the polynomial model. Intuitively, we believed that the polynomial model should fit the data better since the curves are not all straight lines. One of the problems with polynomial models with degree greater than 2 is that they are likely to drastically drop down which is also the reason why we smoothed the data. Besides, how do we select the data and adjust the parameters to form the curves that go as we wish is tough. On the other hand, linear model gives us a more controllable in terms of trend even though it might not perfectly fit the test data. Therefore, our strategy is to use linear model as our base prediction model supported by parameter adjustment functions and to use polynomial model for some cases that we could easily control the trend.

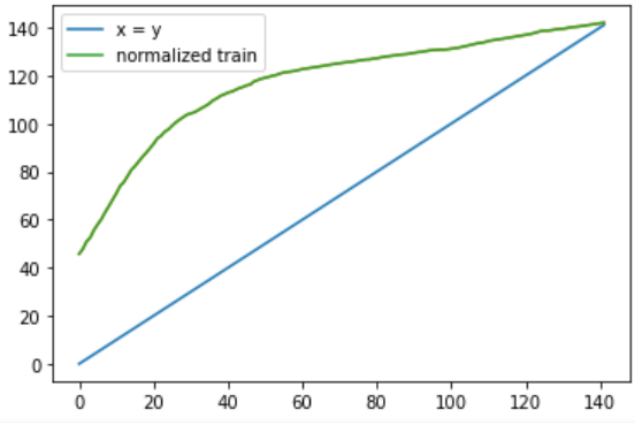

Figure 1: Fast-growing state sample (New Jersey)

Initially, we used linear model with L2 normalization and tuned it to the state with highest validation score (MAPE). In order to tune the model in state level, we were trying to find the patterns of states. We also normalized the data by dividing all data by the value of last day (Day 142 in round 1) so that we can classify them in a standard manner. We found that the states whose values grow too fast in the beginning will always end up with a flat and straight line. We called these states fast-growing states and we defined these states by comparing their values on 80th day with 100. If this value of a state is greater than 100, they are classified as fast-growing states. For the fast-growing states, we select the last ten days to train the liner model without any further adjustment.

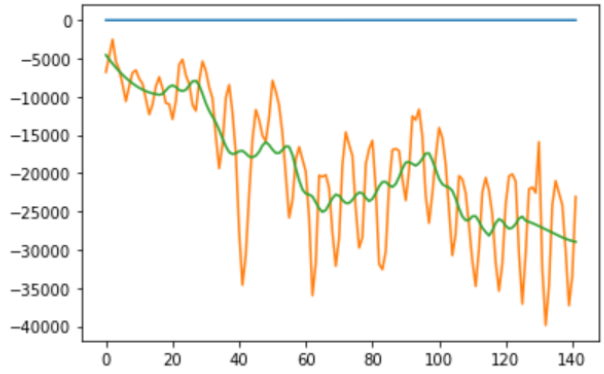

Figure 2: Slow down state sample (California). Orange curve is the original population change curve and Green curve is smoothed population change curve.

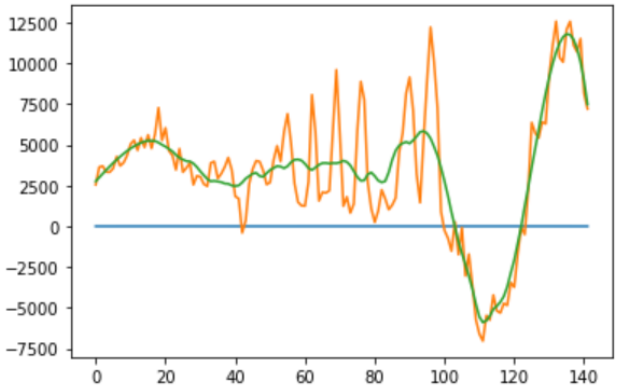

Figure 3: Dramatic change state sample (Iowa). Orange curve is the original population change curve and Green curve is smoothed population change curve.

Furthermore, we observed that the number of confirmed cases is affected by the sign of population change and the speed of the change (slope). We defined the population change as the number of move-in population minus the number of move-out population ($MI - MO$). If the signs of population change of a period prior to a specific day are all positive and its slope is still growing, it is very likely that the number of confirmed cases will grow with a faster speed in the future. In the case, we also increased the slope of our prediction values generated by linear model. As opposite, if the signs of population change of a period prior to a specific day are

all negative and its slope is still decreasing, it is very likely that the number of confirmed cases will grow with a slower speed in the future and we will also slight reduce the slope of the prediction values. Also, if we observe a dramatic population change on a state, it is telling us there will also have a dramatic change on the confirmed cases. If we select the data which matches the dramatic change, the slope of the prediction will be far away from what it should be in the future. In order to tackle this problem, we tried to detect the dramatic change of population by first normalize the data and then looking at the absolute value of the difference between two period prior to the prediction start date. If the value is larger than a threshold, we state that there exists a dramatic change. When we detected a dramatic change, we doubled the size of training set which weaken the effect from the dramatic change.

Before outputting the trained model, we applied a simple result check for it. One certain rule of the value is that, it will never drop down as time goes on. In mathematical term, the slope of the prediction must be equal or greater than 0. We noticed that for states, the slope of prediction is negative which lead to a extremely bad result on score. The reason is that we covered too little data in our training set. For the data with big fluctuation, it is very likely to mislead the model which result in a negative slope of prediction. Our solution to fix this problem was to enlarge the training set so that the model at least knows the rough direction of where the prediction should go.

As we did observe the seasonality from prophet, we also considered applying the seasonality to our prediction generated from linear model as a option. For round 2 data, we gave up the classification of fast-growing states since we observed that it is not the case anymore.

6 EXPERIMENTS DESIGN AND EVALUATION

To evaluate the particular designs of our models proposed, we illustrate some graphical results obtained from training below.

6.1 LSTM

LSTM has two major issues. The first one is that it is hard to stabilize due to the lack of stable features and smooth data categorized by states. As we tested our model independently for each state, we found many states have unstable curves, which are hard to be predicted by our LSTM model. However, we also found that some states have great performance. For example, Alabama and California tend to have very accurate predictions on the confirm and deaths number, as we recognized that the prediction curve is well fitted to the true value both for the training set and the validation set. The second issue is that our LSTM seems to simply regenerate the shape of the previous curve when predicting the future trends, which is not reasonable for the curve of confirm and deaths number. It is probably because our LSTM cannot ideally generalize the high dimensional features. So, we decided to reconsider the implementation of LSTM model and use or incorporate other models as our current priority.

6.2 ARIMA

In order to optimize our prediction results, we tried the most common time series prediction model in statistical models – Arima.

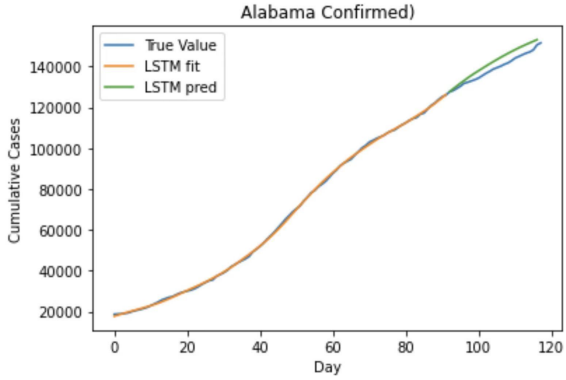

Figure 4: Example of LSTM prediction (good) on Alabama Confirmed cases

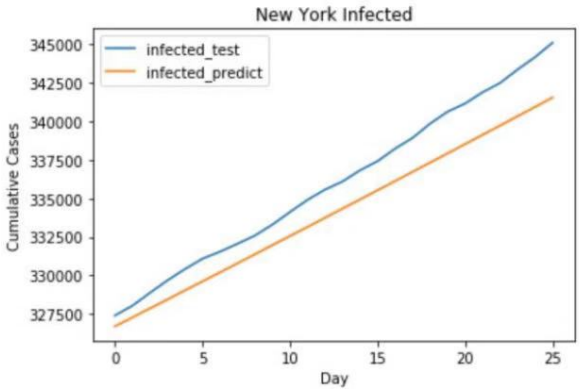

Figure 5: Example of LSTM prediction (bad) on NY Confirmed cases

We implemented it for further reducing the prediction error, especially for those states that we failed to get good prediction results using other models. We tried adjusting parameters for different states and did validation and evaluation on both round 1 and round 2. For round 1, we used data from April 12th to August 31st for modeling and used data of the following 26 days for validation. For round 2, we used data of 225 days before November 23rd as training data and the data of November 24th to December 1st for validation. In this process, we noticed that since Arima model requires the data to be stationary or stable by differential differentiation, the model itself tended to be unstable. For some states, it won't yield results with many specifications and for some small states or states that we needed to do higher differencing on their data, the prediction results are generally not good. When we used the model to predict the deaths in some states which only need first order differencing, the results were good.

For example, the result for predicting Florida deaths from November 24th to December 1st was only around 0.3123 as MAPE. However, when we used Arima to predict Alaska death in the same time window, which required second order differencing to stablize its data, the best result we could attain was 8.172 as MAPE. In general,

ARIMA model did not solve our problems on optimizing our prediction methods and improving our prediction result, so we decided to not use this model to generate our final results and leave it for future exploring.

6.3 Prophet

From training on round1 data (full dataset), we saw that the Prophet model overall performs relatively well, averaging an MAPE of 2.5. We chose some of the results obtained for further analyses below.

For results on round1 and round2, we crawled test data for the actual prediction values from Johns Hopkins University's Github repository of "CSSEGISandData" for visualization purposes.

Specifically, we used Sep 1-16 for round1 and November 24-Dec 6 for round2 as the testing data in plots. We chose to show only predictions made on death values as it is harder to model than confirmed values for illustrating the relative power of our models.

We saw that for states that had smoother changes and followed a nicely predictable trend (rather stable 2nd derivative changes), our Prophet model did exceptionally well to fit for the detailed, granular changes which had clear seasonality.

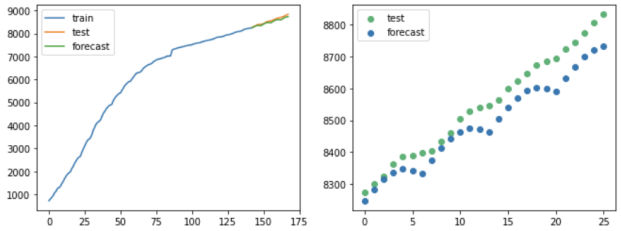

Figure 6: Death prediction for Illinois from Sep1-16

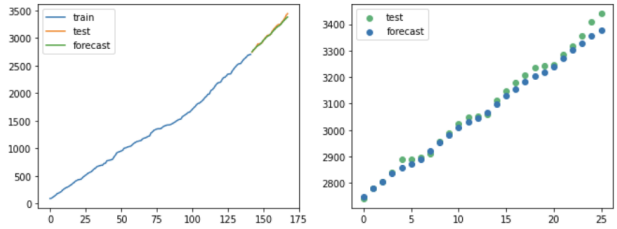

Figure 7: Death prediction for North Dakota from Sep1-16

However, we wished to illustrate cases of which this model was not particularly good at adapting to. First, we observed the case from Texas which shows that the model needs a bias correction as its linear component put too much emphasis on the last part of training data seen.

Also, when the data flattens, shows just a bud of drastic change or exhibits a new seasonal trend, the model does not pick up on it.

Hence, we identified and summarized several key weaknesses of the prophet model here: 1) It still uses linear best-fit for non-satiating parts; 2) It does not identify sudden or flat/no changes; 3) It relies on historical data too much for generating new seasonal patterns.

However, the Prophet package provides us with additional information about the model that can be used as a feature generation tool for identifying and correcting the model behaviors. For example, the seasonality plot and change points plot, which are both adequately stable measures suitable for us to safely exploit on.

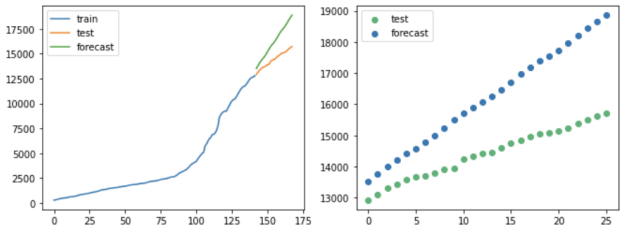

Figure 8: Death prediction for Texas from Sep1-16

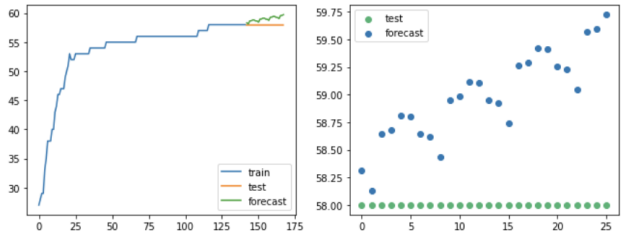

Figure 9: Death prediction for Vermont from Sep1-16

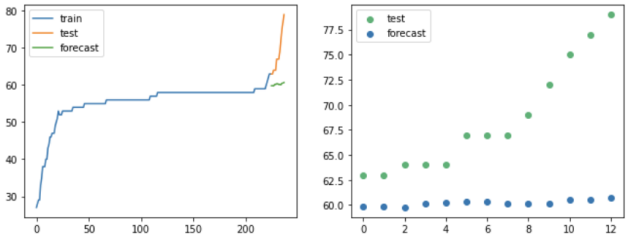

Figure 10: Death prediction for Vermont from Nov24-Dec6

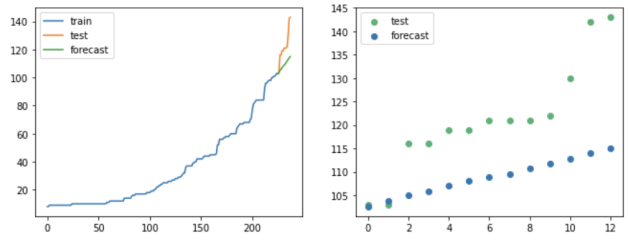

Figure 11: Death prediction for Alaska from Nov24-Dec6

6.4 Linear and Polynomial Models

Since we observed the best result from pure linear models in the preliminary models test, we decided to use it as our based model. Therefore, the most of Kaggle submissions are used for getting the result of linear model experiment. The well tuned linear model with only the L2 regularization achieves an overall 2.19 MAPE score for round 1 data. By applying the fast-growing classification, the score comes down to 2.14 which is not a significant improvement.

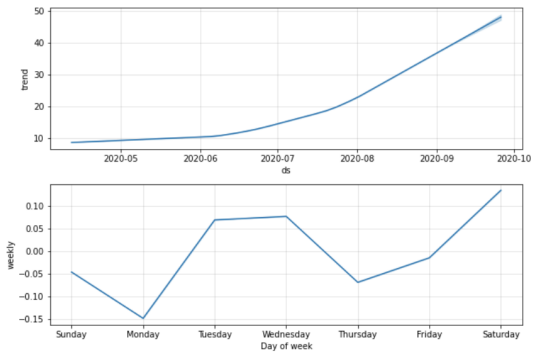

Figure 12: Alaska Deaths Seasonality breakdown on Apr-Sep

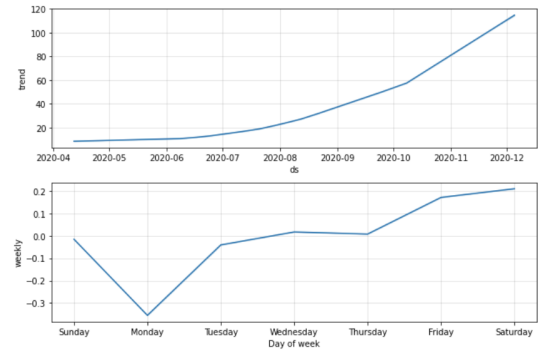

Figure 13: Alaska Deaths Seasonality breakdown on Apr-Nov

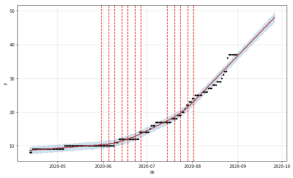

Figure 14: Change points identified for Alaska from Apr-Sep

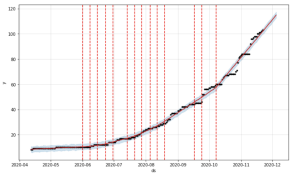

Figure 15: Change points identified for Alaska from Apr-Nov

It makes sense because the only difference is that we ignored the L2 regularization and change the size of training set for the fast-growing states.

By leveraging the mobility data, we knew the population change of states. Based on the change, we adjusted the parameters of the model in terms of slope of prediction and size of training set. This brings us a huge improvement on the score which is 1.9070 for round 1. We also tried to apply the seasonality, which is extracted from Prophet, to our prediction result, for which it further improved our MAPE from 1.9070 to 1.9024, an inconsequential change and we do not know whether it can also improve the round 2 result or not. Therefore, we decided to exclude the seasonality part for round 2.

6.5 Round 1 validation

In order to verify the feasibility of our models as well as avoiding overfitting in round 1, we did validation using part of the training data. We split the data from April 12th to August 31st into two parts. The first part contains data before August 22nd and the second part contains the data of the remaining 10 days in training set. We used the first part as training set and evaluated it through calculating the MAPE using the second part as validation set. The results of validating round1 with our best model are 1.909 for confirmed, 1.713 for deaths and the combined MAPE is 1.811. It was good enough for applying on the whole training data to getting our final result of round 1.

```
print(MAPE(confirmed, real_confirmed), MAPE(deaths, real_deaths))
print('MAPE is: ',(MAPE(confirmed, real_confirmed)+MAPE(deaths, real_deaths))/2)
```

```
1.9090610619834287 1.7132390770694017
MAPE is:  1.8111500695264153
```

Figure 16: MAPE for Round1 validation Results

6.6 Round 1 results

We further illustrate the breakdown of our results of round1 with our best full model, evaluated using MAPE. (Confirmed = 1.621, Death = 2.211, Combined = 1.916)

```
MAPE(submission_df.Confirmed, ds.Confirmed), MAPE(ds.Deaths, submission_df.Deaths)
```

```
(1.6207810836731198, 2.2114146656057123)
```

```
(MAPE(submission_df.Confirmed, ds.Confirmed) + MAPE(ds.Deaths, submission_df.Deaths))/2
```

```
1.9160978746394162
```

Figure 17: Breakdown of MAPE for Round1 Results

7 CONCLUSIONS AND FINDINGS

The best result we have achieved for round 1 is a **test MAPE of 1.916** using a *linear regression with ridge regularization* and varying window sizes (approximately less than 10 days for all) automatically tuned for each state. For round 2, we simply test the functions used in round 1 and remain the functions we think that also work for round 2 data.

For all of the models we have attempted, restricting a tighter window size would always lead to better performance on the test set, which fit our assumption as recent events (e.g. lockdown measures placed/lifted, increased state testing due to new site openings, etc.) might elevate/lower the daily changes of our target data (number of confirmed cases and number of deaths), in which taking account of past information would only prevent the model from learning and adapting quickly for recent predictions. However, our current worry is that this might be overfitting for the short-term predictions (e.g. predicting the next 10 days) and would affect long-term predictions (predicting 10 days of data in the next month).

Moreover, we observed that the distribution of our data varies quite drastically between each state, not only in their slopes but

also second derivatives that marked how their slopes changed and represented a measure of stability of data (the more stable the data, the closer their second derivative to zero). And oftentimes, the drastic changes that occurred in one state would not correspond to that of another while states that had similar trends at first would not often continue to exhibit similar trends, which posed significant challenges for our model to provide excellent predictions across-the-board without careful, extensive hand-tuning for each state.

8 REFERENCES

Helfenstein U. Box-Jenkins modelling of some viral infectious diseases. Stat Med. 1986;5:37–47.

Alok Kumar Sahai. Forecasting of COVID19 per regions using ARIMA models and polynomial functions. 14(5): 1419–1427. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7386367/

Selva Prabhakaran. ARIMA Model – Complete Guide to Time Series Forecasting in Python. https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

Tsega, Tajebe. (2020). COVID-19 Outbreak Data Analysis and Prediction Modeling Using Data Mining Technique. International Journal of Computer (IJC). Volume 38. pp 37-60.

Taylor SJ, Letham B. 2017. Forecasting at scale. *PeerJ Preprints* 5:e3190v2 *https://doi.org/10.7287/peerj.preprints.3190v2*

Data from John Hopkins University: https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports_us

9 TASK DISTRIBUTION FORM

Symbols	Meaning

REFERENCES