CS145 - Team 3 Final Report

Amith Lukkoor - 504769928
amithlukkoor@gmail.com
University of California - Los Angeles
Los Angeles, California, United States

Arnav Garg - 304911796
arnavgrg@ucla.edu
University of California - Los Angeles
Los Angeles, California, United States

Devyan Biswas - 804988161
devyanbiswas@outlook.com
University of California - Los Angeles
Los Angeles, California, United States

Enika Biswas - 004997956
enikabiswas@gmail.com
University of California - Los Angeles
Los Angeles, California, United States

ABSTRACT

As the COVID-19 pandemic continues to rage throughout the United States, many public statistical models have offered predictions of infection trends in order to aid preparations for future virus spread. We participated in a Kaggle competition to create reliable forecasts of cumulative confirmed cases and deaths for each state in the country. Our final submission achieved high performance with respect to our metric, the mean absolute percentage error (MAPE), by utilizing the HoltsWinter Exponential smoothing technique. We achieved training MAPE scores of **0.298** for confirmed and **0.587** for deaths, and we achieved a testing MAPE score of **2.293** on the Kaggle submission, which resulted in 23rd place. We concluded that COVID-19 confirmed cases and deaths are represented well by additive time series, while seasonality has little influence on the data. In the future, we suggest that prediction models implement vector autoregression (VAR) or recurrent neural networks (RNN) to take further advantage of the problem's structure.

KEYWORDS

Covid19, Time Series Forecasting, Exponential Smoothing, Autoregressive Integrated Moving Average, Machine Learning

ACM Reference Format:
Amith Lukkoor - 504769928, Arnav Garg - 304911796, Devyan Biswas - 804988161, and Enika Biswas - 004997956. 2020. CS145 - Team 3 Final Report. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

For much the world at large, the year 2020 has become defined by the COVID-19 pandemic. American life has been fundamentally altered, with concerts and events cancelled, schools and universities closed, and businesses left struggling through the accompanying recession. As of early December 2020, more than 15 million people in the United States have become infected with the coronavirus, and at least 290,000 have died[1]. The ability to model trends in infections and deaths is critical to providing necessary medical infrastructure and resources, as well as shaping government policy in response to viral surges. Our project attempts to utilize several months of data on COVID-19 trends in order to model the pandemic's near future outlook.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Several research groups have made their continually updating forecasting models publicly available. A compilation on the data blog FiveThirtyEight includes and describes analyses from a variety of universities and health institutes[2]. Highlights from this collection include the Johns Hopkins University (JHU) model, which integrates data on stay-at-home orders; the Institute of Health Metrics and Evaluation (IHME) model, which uses anonymized cell phone data to quantify interpersonal contact; and the U.S. Army model, which implements a variation of SEIR, a well-known algorithm for modeling epidemic spread. These models have been following the transmission of the coronavirus since April and have earned trust from media outlets and experts across the country. Overall, the many online forecasts constitute a fascinating diversity of approaches towards understanding public health trends. However, many of the research studies show little insight into their process of error analysis and model improvement. An important motivation behind this project thus becomes to implement reliable prediction models and create accessible accuracy reports.

3 PROBLEM FORMALIZATION

We developed our COVID-19 forecasts with the goal of providing key insights on upcoming pandemic trends to the government and medical institutions. More precisely, we aimed to predict the cumulative totals of confirmed cases and deaths for each state in the United States for up to a month in the future, with specific focus on September 2020 and December 2020.

4 DATA PROCESSING

4.1 Dataset Overview

There are 2 main datasets that can be utilized for the task of COVID-19 Confirmed Cases and Deaths prediction. The first is a training set data containing various features' data from April-August 2020. The second is graph data, which shows the mobility of people in/out of different states during the same time frame. A few of the features (explanations from the GitHub of the project): Confirmed

(aggregate cases for the state), Deaths (aggregate deaths for the state), Recovered (aggregate recovered cases for the state), Active (total cases - total recovered - total death), Incident rate (cases per 100,000), People tested, People hospitalized, Mortality Rate (fatality ratio), Testing rate (tests per 100,000), Hospitalization rate (total hospitalized/total cases). For graph data, each row has a source and target state, where the columns represent each day's amount of people going from the source to the target state.

4.2 Exploratory Data Analysis (EDA)

To analyze the data's general trends, we split the data by state (this step is also done in the data pre-processing). That way, when we plotted the data trends over time, we could more easily observe the overall trends for each feature per state, which is beneficial to our later uni-variate models. Here is a small sub-sampling of the state of Alabama, and the plots of a few of its features over time. Note that NaN values were not explicitly dealt with in this step, as the main purpose of these plots were to gain a general understanding of the features for each state. Examples of the variable trends over time can be found from **Figures 1-5**.

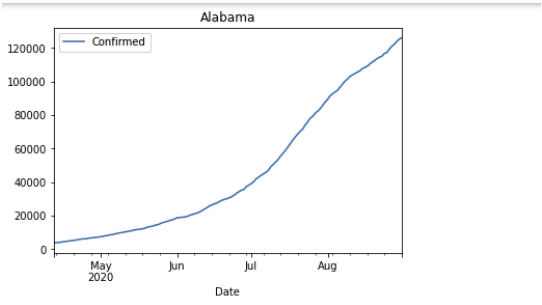

Figure 1: Confirmed vs Time

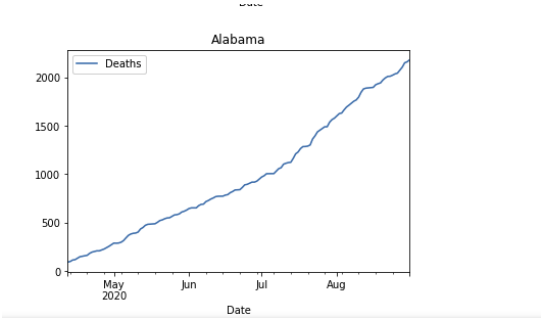

Figure 2: Deaths vs Time

We did the above process for every state across all the dates in the training dataset. The main thing that became evident, especially when looking at the confirmed and deaths vs time graphs, was that they were generally increasing, and could also be described as either exponential or pseudo-linear growth for most states. It was these initial observations on the trends of confirmed and deaths that led us to develop models using linear regression and Decision

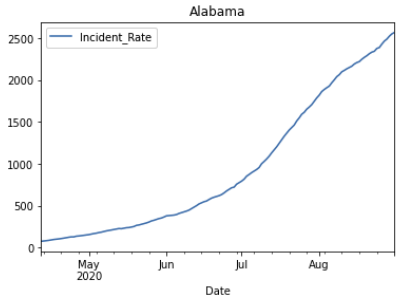

Figure 3: Incident Rate vs Time

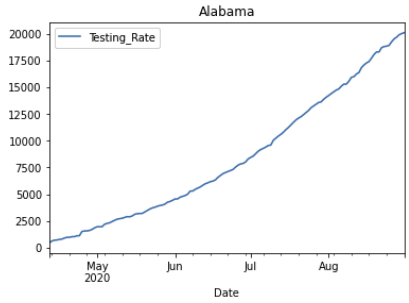

Figure 4: Testing Rate vs Time

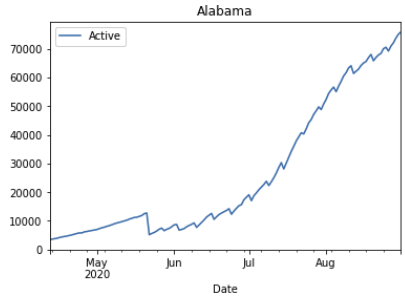

Figure 5: Active vs Time

Trees/Random Forest, while also later reinforcing our attempts at creating univariate models using a variety of different algorithms.

4.3 Feature Selection

We started by implementing a Pearson Correlation matrix for each state to see the relationships among the features.

As can be seen from **Figure 6**, there is high correlation in the top left square from features Confirmed to People_Tested. The other states had very similar correlation matrices as well. Interestingly, People_Hospitalized did not have a significant correlation in either Confirmed or Deaths for most states. We are trying to gain two main insights from the correlation matrix: Correlation of features with Confirmed/Deaths and correlation of features among each other. The second insight is useful because once we find features that are highly correlated with Confirmed/Death, we need to see if those selected features are highly correlated among each other

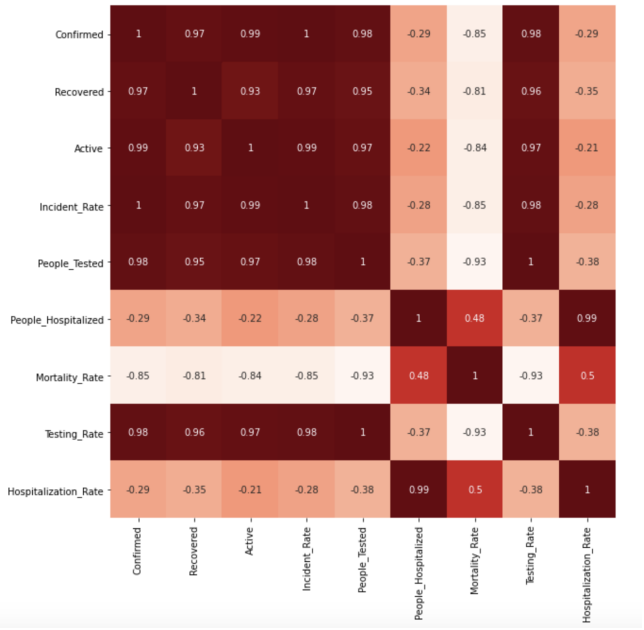

Figure 6: Alaska heat map to show correlation among all features.

as well. This will help us remove redundancy and make it easier to develop our models as there are less features to consider.

Our correlation score threshold was **0.75**, meaning features that have a correlation score of 0.75 or above are considered to be highly correlated. This threshold was selected based on observing the heat maps of each state and noticing that 0.75 was a threshold that made sense for all states.

If the features have a correlation score of 0.75 or above when being compared with Confirmed/Deaths, then the features are considered relevant. If the correlation score among those selected features are greater than 0.75, then only one of those features is considered and the other is discarded. We also performed the same thing to test for negative correlation among features with the threshold being -0.75.

Finally, these filtered relevant feature lists are then uploaded to a text file with the corresponding state as seen in **Figure 7**.

State	Features selected
Alabama	['Recovered', 'Active', 'Incident_Rate', 'People_Hospitalized', 'Testing_Rate']
Alaska	['Recovered', 'Active', 'Incident_Rate', 'Testing_Rate']
Arizona	['Recovered', 'Active', 'Incident_Rate', 'People_Hospitalized', 'Testing_Rate']
Arkansas	['Recovered', 'Active', 'Incident_Rate', 'People_Hospitalized', 'Testing_Rate']

Figure 7: Some of the states and the features that have been selected for each of them.

4.4 Pre-Processing

Our first approach to pre-processing was to start by grouping all the data by state starting with Alabama and ending with Wyoming. After careful inspection of the data, we noticed that three features had NaN values: Recovered, People_Hospitalized, and Hospitalization_Rate. To resolve this issue, we filled the NaN values with zeroes depending on which columns had NaNs for each state. We chose to replace the NaNs with zeroes instead of the average of the particular column because we were not sure what the NaN represented. It could be that there was no data collected for that particular day or it could have been data that didn't make sense compared to the typical values of the feature. Thus, zero is a safe choice to pick to account for this uncertainty and to essentially have the model disregard this feature for that particular day.

Next, we merged the graph data set with the train data set. This was accomplished by adding the source states of a particular target state from the graph data as additional columns to the training data set. This transformed the data set by increasing the dimension of the training set by at most 50 more columns (one new column for each state), an example of which can be seen in **Figure 8**.

People_Hospitalized	...	South Dakota	Tennessee
5971.857143	...	48.428571	7150.000000
0.000000	...	12.428571	55.571429
9166.714286	...	136.571429	403.428571

Figure 8: For target/province state of Alabama, here are all the source states added as columns to the training set.

Next, to smooth out the data over time, we tried implementing a 7 day moving average for each feature. To better see the trends in the data, and for modeling this timeseries task using multivariate models, we also tried Z-score normalization for both the training and testing data set as well as normalizing the day and month. Both day and month are ordinal features, so we converted them into a range of values from 0 to 1, where 0 is the smallest day/month (1,1) and 1 is the highest day/month (31,12).

However, after careful examination of the data, we noticed that many of these pre-processing steps were unnecessary. To reach this conclusion, we graphed Confirmed vs Date and Deaths vs Date for each state. The resultant graphs showed that both Confirmed vs Date and Deaths vs Date for each state followed either a linear or exponential trend. Thus, we focused our attention on creating models that worked for linear and exponential type data, and as

a result, the other features besides Date were not considered for creating our models.

5 MODEL EVALUATION AND SELECTION

We decided to train each model variant on data for $t = 1..(t-20)$. This allowed us to use the last 20 days of August 2020 as our validation set. We chose this because the test set (September 2020) had $t = 27$ days of required forecast predictions and $t = 20$ served as a good enough heuristic to judge the model's performance capability, i.e., the same model always won for $t = 20..27$. Furthermore, we made the assumption that a model that could generalize well and make good predictions (by giving us the lowest error) for the last 20 days of August would likely also have accurate predictions for September 2020 when the model was retrained using the entire training set. This is based on a second assumption - that the underlying direction of the trend line during these 20 days was unlikely to vary too much during September 2020. Unlike the first assumption, the second assumption is largely supported by the data for most of the states, where deaths and confirmed Covid19 cases seem to have a linearly growing trend with minimal variance, an example of which is shown below for the state of California in **Figures 9 and 10**.

California

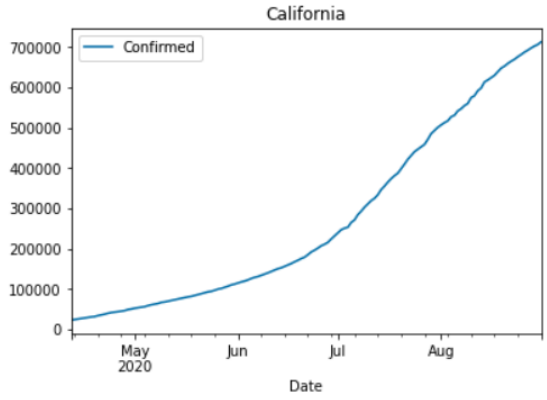

Figure 9: California Confirmed Cases

For evaluation, we used Mean Absolute Percentage Error (MAPE). For model selection, we performed a grid search across both multivariate and univariate regression model hyper-parameters and picked the model that had the lowest MAPE. This is explained in more detail in the sub-sections below. We pick the model with the lowest MAPE, and use that to forecast values for the test set.

5.1 Mean Absolute Percentage Error (MAPE)

To evaluate our models, we decided to use Mean Absolute Percentage Error (MAPE) as our metric. We chose MAPE because it is a strong metric to evaluate the accuracy of forecasting models, and also allowed us to correctly estimate our model's performance since it matched Kaggle's evaluation metric on the public leaderboard. We calculate MAPE using [3]:

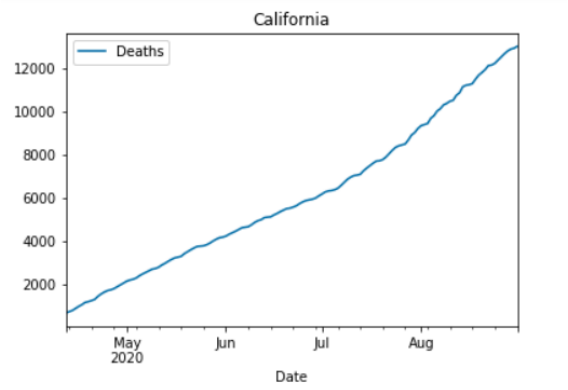

Figure 10: California Deaths

$$MAPE = \frac{1}{n}\sum_{t=1}^{N}\left|\frac{A_t - F_t}{A_t}\right| * 100$$

where

- n: total number of data points (actual or predicted)
- A_t: actual ground truth value
- F_t: forecasted value

This gave us two strategies to evaluate the performance of our models -

- `Strategy 1 - Regular Validation`

  In this method, the model is trained only once on data from time steps 1 to t. Then, this model is used to forecast values from t+1 to t+k, where $k = 1..n$ is the total number of days that we need to forecast values for using the trained model. This forecast over k days is compared to the ground truth values for the same k day period, and the MAPE value is computed.

- `Strategy 2 - One-Step Walk-Forward Validation`:

  In walk-forward validation, the model is updated each time step new data is received. As can be seen in **Figure 11**, a model is trained on data from time step 1..k. Then, the model is made to forecast the value for time step k+1. This value is compared to the ground truth value at time step k+1, and the MAPE value is calculated and saved. Then, the ground truth value k+1 is added to the training data, and the model is trained again. This loop repeats for all time-steps from t+1 to t+k, where $k = 1..n$ is the total number of days that we want to forecast values for. Finally, this forecast over k days is compared to the ground truth values for the same k day period, and the MAPE value is computed.

There were three main reasons we picked one-step walk-forward validation over regular validation methods. Firstly, in time series modelling, the predictions become less and less accurate over time and hence it is a more realistic approach to re-train the model with actual data as it gets available for further predictions. This was also

corroborated by results we observed, where the model started to take largely divergent paths from the actual data as the size of the forecast duration increases, largely skewing the MAPE values with larger time steps. The second reason is that most of the papers we read on time-series forecasting always seemed to use Strategy 2 as the de facto evaluation method. The third reason is that when we tried using Strategy 1 for our ARIMA model (described below), we got very large MAPE values that prevented us from tuning the ARIMA model correctly and resulted in poor results. We saw much better performance and scores from ARIMA after swapping to Strategy 2.

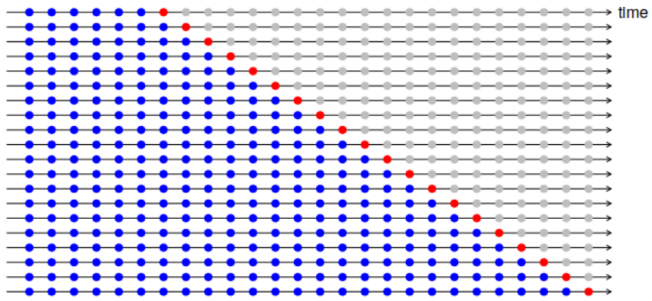

Figure 11: Walk-Forward Validation

5.2 Grid Search

We ran a grid search over multiple models from the scikit-learn library and several parameters to determine the best classical regression models for Confirmed and Deaths. We tested LinearRegression, SGDRegressor, DecisionTreeRegressor, and RandomForestRegressor. For the SGD algorithm we varied the regularization term and its constant multiplier α. For both of the tree algorithms, we tested different maximum depths.

We reported the training MAPE score for each model to determine whether any of the models could fit the data successfully. We found that the best model for Confirmed was the linear regression, which had a MAPE score of **3.6338**. Our best model for Deaths was the random forest regressor with maximum depth 25, which had a MAPE score of **24.9654**. Although we had achieved a decent baseline for the Confirmed dataset, it was clear that these algorithms were failing to fit the Deaths dataset.

6 MULTIVARIATE MODELS

6.1 Linear Regression

The LinearRegression model performs Ordinary Least Squares on the data. It fits the data by minimizing its error sum of squares. The prediction made is the optimal linear combination of the features. The model reported MAPE scores of **3.6338** for Confirmed and **97.2914** for Deaths.

The SGDRegressor fits a linear model to the data by using Stochastic Gradient Descent to minimize the cost function. It allows the user to use the squared Euclidean norm L_2 or the absolute norm L_1 as the regularizer, and takes input for the constant α used to scale this term. Grid search resulted in a model with MAPE scores of **3.6338** for Confirmed and **82.4146** for Deaths. We note that the Confirmed score is identical to that of the LinearRegression model, while the Deaths score is an improvement.

While the Confirmed dataset allows for reasonably accurate linear modeling, it is clear that linear regression is not fitting the Deaths dataset well.

6.2 Decision Tree

The DecisionTreeRegressor uses if-then-else decision rules to create a regression model for the dataset. We used the default criterion of minimizing MSE and set various maximum depth hyperparameters to prevent overfitting. The best decision tree model had a maximum depth of 15 and resulted in MAPE scores of **3.6338** for Confirmed and **27.1710** for Deaths. It was interesting to see that the score for Confirmed exactly matched that of the linear models, while the score for Deaths was significantly improved.

6.3 Random Forest

The RandomForestRegressor is an ensemble method consisting of several decision trees. The default algorithm fits 100 trees to the data and averages their output to both boost accuracy and limit overfitting. We again implemented various maximum depths as an additional check on overfitting. We found that a maximum depth of 25 resulted in the best random forest model, which had MAPE scores of **3.6338** for Confirmed and **24.9654** for Deaths. Again, we note that the score for Confirmed was a match with all of the other models, while the score for Deaths is the best overall from the previous models.

6.4 Summary

All of the previous models achieved a reasonable baseline of performance for the Confirmed dataset, but failed to fit at desired levels. Moreover, each algorithm failed to achieve even a baseline model for the Deaths dataset. We realized that these classical regression methods were not well optimized for this data. One reason for this issue is that the data seems to follow exponential curves, which are not predicted well by linear or decsion tree models. Another explanation for our lackluster results is that for this time series data, the more recent examples are more important for prediction than older examples, another fact that the previous models do not take into account. Thus, we turned to experimentation with univariate models.

7 UNIVARIATE MODELS

Univariate models are used to model a time series that consists of single (scalar) observations recorded sequentially over equal time increments. This means that we try to find a relationship between time and a single variable, which in our case is confirmed and deaths respectively. Our goal was to build two univariate models for each state, one to predict the number of confirmed cases and the other to predict the confirmed deaths for that given state. We explored four different univerative time-series forecasting models, the results of which are described in the following subsections.

7.1 Simple Exponential Smoothing

Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight.

Simple Exponential Smoothing (SES) is time series forecasting technique for data without trend or seasonality. The basic mathematical principle of SES is as follows [5]:

$$SES = \alpha \times series[n] + (1 - \alpha) \times result[n - 1]$$

In this equation series[n] represents the data point at the current time we are observing, the alpha refers to the smoothing parameter between 0 and 1, and result[n-1] refers to the previous SES value from the last iteration. Essentially, this model builds off of the last value calculated to improve its accuracy.

We experimented with smoothing parameters of 0.5 and 0.7. Ultimately, we picked 0.5 because 0.7 caused over fitting of the data. Below is an example graph of Alabama to test our SES on our training data:

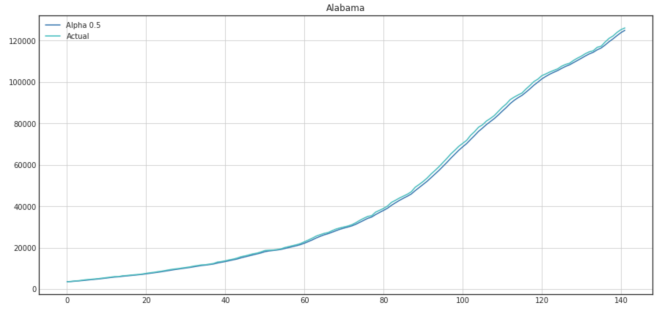

Figure 12: This shows SES being trained on Alabama training set.

Next we take data from before August 15 as the confirmed training data, and take the data from after August 15 as the confirmed test data. Finally, we apply Python's SimpleExpSmoothing(..) function with 0.5 as the alpha. We then calculated the MAPE accordingly.

State	Validation MAPE for Confirmed
Alabama	8.512
Alaska	14.276
Arizona	3.270
Arkansas	7.661
California	8.867

Figure 13: This is a table for a few states showing the Validation MAPE values for Confirmed cases.

Ultimately, we did not fully implement SES because upon further research it seemed that HoltsWinter was a better choice due to its flexibility of not having to choose a particular smoothing parameter, and its accuracy being significantly better. Additionally, SES is usually not the best option when considering trend and seasonality in a data set. Since HoltsWinter ended up performing very well, this would tell us that there is a trend in our data set and it needs to be accounted for.

7.2 HoltsWinter/Exponential Smoothing

The Holt-Winters seasonal method essentially builds on Simple Exponential Smoothing and Trend methods. It comprises the forecast equation and three smoothing equations — one for the level l_t, one for the trend b_t and one for the seasonal component s_t, with corresponding smoothing parameters α, $\beta*$ and γ. We use m to denote the frequency of the seasonality [4].

The trend equation extends the level equation from Simple Exponential Smoothing to allow the forecasting of data with a trend. The seasonality equation extends the trend and level equations to account for seasonality trends that may influence forecasts.

Largely, there are two variations of this method - the addition method and the multiplicative method, that differ only in the nature of the seasonal components. We only discuss the additive method below since it is what we used for our final submission.

Additive Method [4]

$$Y_{t+h|l} = l_t + hb_t + s_{t+h-m(k+1)}$$
$$l_t = \alpha(y_t - s_{t-m}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$
$$b_t = \beta * (l_t - l_{t-1}) + (1 - \beta*)b_{t-1}$$
$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m}$$

The first thing to notice is the interdependence between the three equations. Here, l_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, and $\beta*$ is the smoothing parameter for the trend. As with simple exponential smoothing, the level equation here shows that l_t is a weighted average of observation y_t and the one-step-ahead training forecast for time t, here given by $l_{t-1} + b_{t-1}$. The trend equation shows that b_t is a weighted average of the estimated trend at time t based on $l_t - l_{t-1}$ and b_{t-1}, the previous estimate of the trend. The h-step-ahead forecast is equal to the last estimated level plus h times the last estimated trend value. Hence the forecasts are a **linear** function of h [4]. The seasonal equation shows a weighted average between the current seasonal index, $(y_t - l_{t-1} - b_{t-1})$, and the seasonal index of the same season last year (i.e., m time periods ago).

For this model, we first tried performing a grid search over all 6 parameters offered by the `statsmodel` Python package [5]. These are whether to use an additive or multiplicative trend, additive or multiplicative seasonality, the seasonality period, whether to use dampening, whether to enforce that the average residual is equal to zero and whether to apply a boxcox transformation. This resulted in significant overfitting, where the one-step walk-forward validation MAPE on confirmed was **0.24** and the and that on deaths was **0.54**. On our Kaggle submission, this produced a MAPE of **4.40**, which was much higher than the baseline score.

Some analysis indicated that the trend lines were far too complex and were overfitting, so we decided to switch to just using an additive trend with no dampening, and forced the average residual to be set to zero. We also enabled boxcox transformations, and

avoided seasonality trends. This resulted in linear trendlines that looked to be much more generalized. The removing of dampening also had a significant effect since the curves became completely linear in most cases. **Figures 14-17** show some of the graphs where the model did well and did poorly.

With these parameters, we were able to prevent overfitting/underfitting for most states. This resulted in a MAPE of approximately **0.298** for confirmed and **0.587** for deaths. This resulted in a MAPE of **2.293** on Kaggle, which was the best score that we achieved.

State	Confirmed (MAPE)	Deaths (MAPE)
New Jersey	0.076	0.118
South Dakota	0.6876	0.447
New Hampshire	0.132	0.165
Hawaii	0.999	3.050
California	0.346	0.489

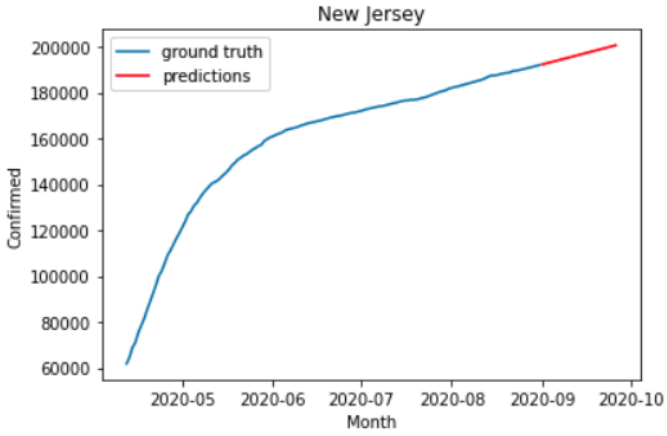

Figure 14: ETS - New Jersey Confirmed Prediction

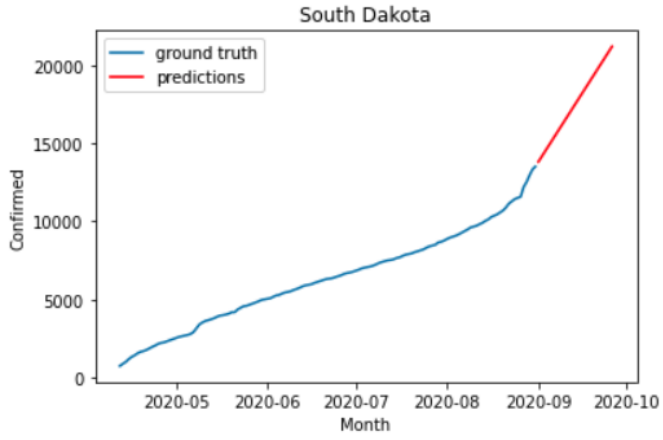

Figure 15: ETS - South Dakota Confirmed Prediction

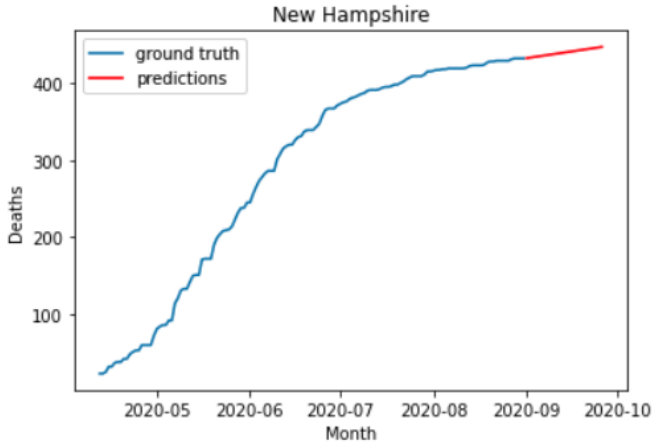

Figure 16: ETS - New Hampshire Deaths Prediction

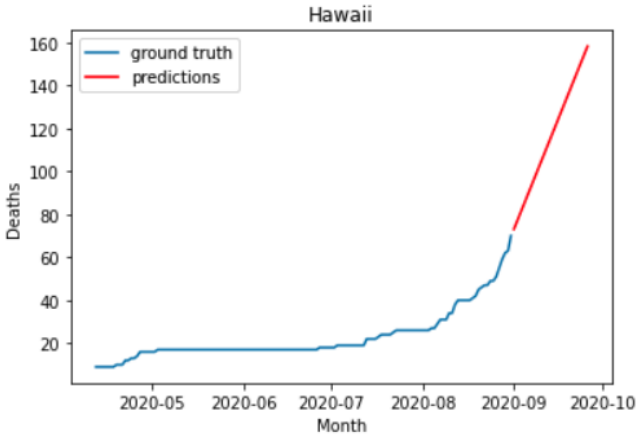

Figure 17: ETS - Hawaii Deaths Prediction

One possible explanation for HoltsWinter's success is that our data has an underlying trend pattern that cannot be modeled using Simple Exponential Smoothing, and in particular, the data showcases an additive trend with no seasonality component. This does make sense the data is growing linearly. Furthermore, since the direction of the trend is very sensitive to the values observed in the most recent days, being able to use a weighted average of past observations also provides a boost since as observations get older (in time), the importance of these values get exponentially smaller. Therefore, more importance is placed on the last few days in the dataset.

7.3 Autoregression Integrated Moving Average (ARIMA)

While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data. ARIMA is a univariate

forecasting technique that projects the future values of a series based entirely on its own intertia.

There are three main aspects to this model:

- **AR**: This stands for *Autoregression*. This model uses the dependent relationship between an observation and some number of lagged observations
- **I**: This stands for *Integrated*. This model uses differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA**: This stands for *Moving Average*. This model uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

A "lag" is a fixed amount of passing time. One set of observations in a time series is plotted (lagged) against a second, later set of data. The kth lag is the time period that happened "k" time points before time i. [6] A stationary time series is one whose properties do not depend on the time at which the series is observed [7]. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times.

From some of the plots shown in **Figures 9 and 10**, it can be seen that both Confirmed and Death datasets for each state has a clear trend. This suggests that the time series is not stationary and will require differencing to make it stationary, at least a difference order of 1. We created Autocorrelation (ACF) and Partial Autocorrelation plots (PACF) - ACF helps us determine whether we need to add MA terms. PACF helps determine whether we need to add AR terms. With a differencing factor of 2, it seemed like the data was almost stationary. Below are the plots for Confirmed cases using a differencing of 2 **(Figure 18)**.

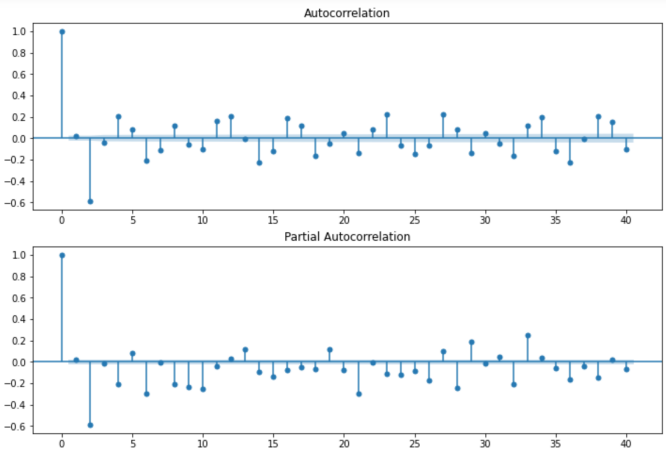

Figure 18: Autocorrelation and Partial Autocorrelation Plots

To determine the level of lag required, we created autocorrelation plots for each state. We noticed that for a majority of states, there is a positive correlation till the first 45-50 lags, but is most significant for the first 40 lags. This was fairly consistent across all of the states, as can be seen in the plot for California (Figure 19).

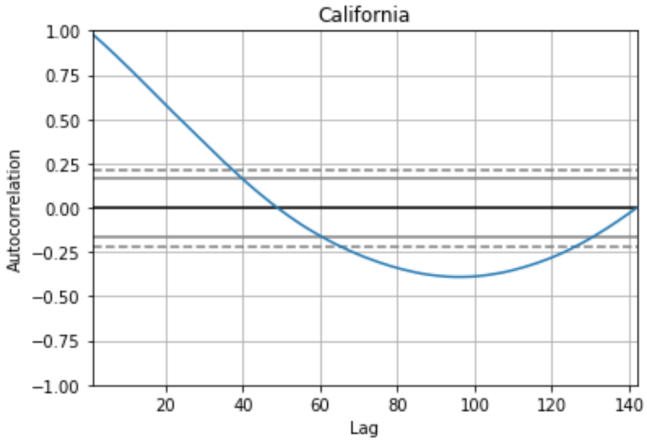

Figure 19: Autocorrelation vs Lag for California Confirmed cases

We then performed regular validation (rather than walk-forward validation) where the last 20 days of August were used for validation. The model was trained using a lag order of 35 (changed from 40 since 40 looked like it was causing overfitting), differencing of 1 and moving average order of 0. Below are some of the results we observed for Confirmed cases **(Figure 20 and 21)**.

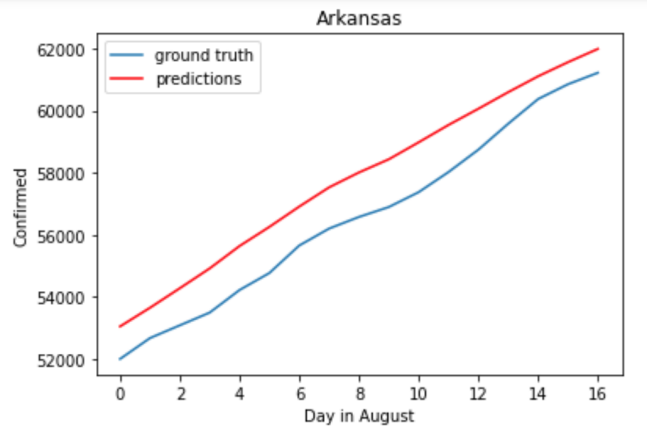

Figure 20: ARIMA: Arkansas Prediction on Validation (overfitting) [MAPE: 2.178]

In general, we observed that the model usually resulted in significant overfitting or underfitting for about 75% of the states. This was true for both confirmed cases and deaths.

State	Confirmed (MAPE)	Deaths (MAPE)
Arkansas	2.178	4.558
Maine	1.479	0.485
New York	0.610	0.082
Hawaii	2.164	9.785

The average MAPE for confirmed cases was **1.818** and for deaths was **2.5367** across the states, which was significantly higher than

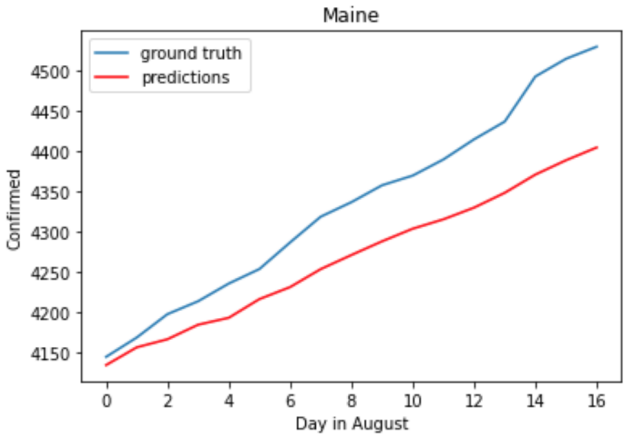

Figure 21: ARIMA: Maine Prediction on Validation (underfitting) [MAPE: 1.479]

the verage MAPE scores seen for Exponential Smoothing. Although ARIMA is typically more general than Exponential Smoothing, we believe our model didn't perform as well because this is a tricky model to tune the hyperparameters for since there are many of them, and performing grid search across these parameters for each state would have taken very long. In addition, despite spending a lot of time trying to understand how this model worked, it still seemed to be very black-box like which made it hard to evaluate and understand its predictions. Lastly, the issue of hyperparameter tuning could have been avoided by training ARIMA using R, where the `auto.arima()` package automatically tunes all three of these parameters. However, none of us had prior expertise with R so we decided to skip trying to implement this model using a different programming language.

7.4 Vector Auto Regression

While we did not manage to create a submission for this model, we determined that this type of model would be the logical next step based on our previous work and research. At its root, VAR uses the past values in order to to forecast its future values. Each feature/variable is modeled as a time series equation, which is a combination of both its own past values and the past values of the other features/variables in the system. The number of previous time periods is called the order, which determines how far back we look for relevant values, and can be considered a hyperparameter that can be optimized.

Second Order (VAR(2)) Example[8]

$$Y_{1,t} = \beta_{11,1} * Y_{1,t-1} + \beta_{12,1} * Y_{2,t-1} + \beta_{11,1} * Y_{1,t-2} + \beta_{12,1} * Y_{2,t-2}$$

$$Y_{2,t} = \beta_{21,1} * Y_{1,t-1} + \beta_{22,1} * Y_{2,t-1} + \beta_{21,1} * Y_{1,t-2} + \beta_{22,1} * Y_{2,t-2}$$

This overall model fits well with the available data and general intuition that comes from this particular task of COVID-19 prediction. We can also look to called the Granger Causality test using a max value threshold like 0.05 to determine which features are correlated to each other (any correlation < 0.05 is correlated). This method will be particularly useful in determining if the particular problem is a good fit for VAR in general. It could also be used to reduce the number of lag variables upon which we define our system of equations; by determining what variables are not relevant or correlated to the confirmed cases and deaths, we can potentially remove them from further calculations and possibly from the entire system of equations, given our interest solely in confirmed and deaths.

Alabama

	Confirmed_x	Deaths_x	Recovered_x	Active_x
Confirmed_y	1.0000	0.0017	0.0000	0.0000
Deaths_y	0.0000	1.0000	0.0121	0.0000
Recovered_y	0.0273	0.0004	1.0000	0.0246
Active_y	0.0000	0.0014	0.0000	1.0000
Incident_Rate_y	0.0004	0.0020	0.0000	0.0000
People_Tested_y	0.0193	0.1674	0.0021	0.0222
People_Hospitalized_y	0.0000	0.2076	0.2732	0.0463
Mortality_Rate_y	0.0449	0.0003	0.0000	0.5377
Testing_Rate_y	0.0211	0.2713	0.0010	0.0263
Hospitalization_Rate_y	0.0004	0.0827	0.0999	0.0516

From there, we could also perform a co-integration test in order to further confirm statistically-significant correlation between two particular features. This would leverage the idea of order of integration to determine statistical connections between two time series. Then, we can iteratively perform the Augmented Dicky-Fuller test on each state-model, differentiating when non-stationary, until each state-model's feature time-series is transformed into stationary (meaning independent of seasonality and trends, and with a constant variance and mean over time). At this point, we can also tune our lag hyperparameters, and perform training to fit the set of equations like the one provided in the above example.

8 CONCLUSION

8.1 Main Takeaways

After performing deep EDA and trying a large range of popular multivariate and univariate models, we had two main takeaways:

1. In general, predicting deaths accurately is much harder than predicting confirmed cases. Firstly, the number of confirmed cases is several orders of magnitude larger than the number of deaths, which means that the trends are more predictable. Secondly, the large errors aren't as obvious for confirmed case predictions because of the magnitude of confirmed case numbers, however, they become very prominent for deaths because the scale is within a few hundreds if not lesser. This is also empirically supported from the results of all of our models, where the average MAPE value for deaths across states is always much higher than that of confirmed cases for each of the models we tried.
2. States such as Hawaii, North Dakota and Wyoming make using one type of model architecture difficult, since the underlying assumptions made during modeling (such as while using Exponential Smoothing) don't necessarily support the underlying trends seen in the data for these states. This is particularly true for the deaths, where the values are change very slowly and create staggered lines in the graph. One way to counteract this could be to apply some sort of smoothing to the data before passing it into the model for prediction, however, this would likely cause an inflation of the ground truth values and deteriorate the quality of the model's predictions overall.

8.2 Results

Given what we found with our multivariate models, which had on average higher MAPE's, we opted to try out univariate models to determine if we could more accurately represent the series of confirmed and deaths. Our first attempt, using simple Exponential Smoothing, resulted in MAPE values for confirmed cases as showin in the Exponential smoothing section. We then extended this with our first attempt at Holts-Winter Exponential Smoothing, attempting to optimize over various parameters involving things like seasonality and trend-type. This gave us an overfitted model, with MAPEs for confirmed and deaths at 0.24 and 0.54 respectively during testing, but a Kaggle MAPE score of 4.40. We then analyzed the trends further and reduced the number of parameters, which allowed us to create more linear trend lines. This approach resulted in our best training MAPE scores, **0.298** for confirmed and **0.587** for deaths, as well as a Kaggle MAPE score **2.293** - our best score!

This result tells us quite a bit about COVID-19 confirmed cases and deaths; namely, that they seem to be represented by additive, not multiplicative, time series. Moreover, seasonality is not particularly present in the data, which is a good indicator that basic Vector Autoregression may be a good way to approximate these variables' trends. Our ending Kaggle place is 23rd, with a score of **2.29327** as mentioned before.

8.3 Future Steps

In a previous section, the initial steps of Vector Autoregression were mentioned, but the full implementation was not completed. The next step to take would be to finish implementing state-level models for each state using this model, and then tuning parameters such as the order of and the number of equations per model. Additionally, the time-series nature of this data set makes it an ideal candidate for Recurrent Neural Networks (RNN's). The "memory" aspect of RNN's would make them ideal for this situation, and we could potentially mimic the idea of weighting more recent values higher than older ones from the Exponential Smoothing models with N's. One other factor that may improve the predictive ability of our current and future models is inclusion of more types of data, such as stay at home/quarantine efficacy/levels on a day-day basis, or even population density data.

9 TASK DISTRIBUTION

Task	Teammates Involved
Exploratory Data Analysis	Arnav, Enika
Data Augmentation	Amith
Multivariate Models	Amith, Arnav, Devyan
Grid Search	Amith
Simple Exponential Smoothing	Enika
ARIMA	Arnav
Holts-Winter	Arnav
Vector Auto Regression	Devyan
Midterm Report	Devyan, Enika
Final Report	Amith, Arnav, Devyan, Enika

10 CITATIONS AND BIBLIOGRAPHIES

1 : "Coronavirus in the U.S.: Latest Map and Case Count." The New York Times, 3 Mar. 2020, www.nytimes.com/interactive/2020/us/coronavirus-us-cases.html?name=styln-coronavirus.

2 : Best, Ryan, and Jay Boice. "Where The Latest COVID-19 Models Think We're Headed - And Why They Disagree." FiveThirtyEight, 9 Dec. 2020, projects.fivethirtyeight.com/covid-forecasts/.

3 : Stephanie, S. (2020, September 05). Mean absolute percentage error (MAPE). Retrieved December 07, 2020, from https://www.statisticshowto.com/mean-absolute-percentage-error-mape/

4 : Hyndman, R.J., Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2.

5 : Statsmodels.tsa.holtwinters.ExponentialSmoothing.(2009). Retrieved December 11, 2020, from https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html

6 : Stephanie, S. (2016, November 13). Lag Plot: Definition, Examples. Retrieved December 12, 2020, from https://www.statisticshowto.com/lag-plot/

7 : Forecasting: Principles Practice. (n.d.). Retrieved December 12, 2020, from https://otexts.com/fpp2/stationarity.html

8 : Prabhakaran, S. (2019). Vector Autoregression (VAR) - Comprehensive Guide with Examples in Python - ML+. ML+. Retrieved 12 December 2020, from https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/?fbclid=IwAR1DAzUcq7wQ4VexhnWDdnTVGTzexBP-0JDvYPnRce5I-KVB0bY-f-SbGdg.