CS 145 Final Report Team 25

Keshav Chakrapani
keshavc217@g.ucla.edu
UID: 405176266

Harsh Chobisa
harshc4@g.ucla.edu
UID: 505103854

Sultan Madkhali
sultanm@g.ucla.edu
UID: 705175982

Prithvi Kannan
prithvi.kannan@gmail.com
UID: 405110096

Tejas Bhat
tejasgbhat@gmail.com
UID: 705199445

ACM Reference Format:
Keshav Chakrapani, Harsh Chobisa, Sultan Madkhali, Prithvi Kannan, and Tejas Bhat. 2022. CS 145 Final Report Team 25. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Our goal for this project was to to create a model that could predict COVID-19 cases and deaths in the 50 US States. The implementations was done using different Machine Learning, Data Science and Data Mining techniques. Through extensive research and testing, we have been able to build successful models that have predicted these cases and deaths to a certain degree. As this project heavily related to the current day COVID-19 crisis, much public research has been available for us to absorb and learn from.

This project involves multiple different models that we attempted over the courseof he quarter. In the beginning weeks of the project we have attempted models such Linear Regression, Polynomial Regression, Decision Tree Regressor, Random Forest Generator and Multi-Layer Perceptron Regressor(MLP). From all these initial models, we saw the most potential in Linear Regression. We then advanced that model to combine it with Support Vector Regression, which is a continuous predictive version of Support Vector Machine. By the later end of the quarter, we decided to switch towards time series analysis. Here, we got great results from utilizing the ARIMA and SARIMAX time series models and got bad results from the Prophet model. Our best model, SARIMAX, produced results with a MAPE score of 2.30, which was very good for our use case.

2 INTRODUCTION

COVID-19 is one of the most tragic and historic events to have taken place in our lifetime. The virus first made news headlines in early March, 2020, and has dominated and dictated our lives ever since. Since then, many researchers, virologists, and computer scientists have worked together to create models that attempt to accurately predict the two most important components of this virus: how many people are affected totally, and how many people have died totally. Over the course of the year, it has been relatively easy to see that while some models have predicted the virus' growth semi accurately, none have been able to do so perfectly. This search for a very applicable model has deep ties to our current class, Data Mining.

In today's world, data is abundant. Our class aims to teach us the best ways to harness this data, and this project is an implementation of these methods. Through ever-improving machine learning algorithms, we can leverage this data to guide us through the future of this virus. Our goal with this project is to create an accurate and extensible model that is capable of predicting total covid cases and deaths within 2.3 MAPE. We intend for this model to not only be extremely accurate for the predictions of September, but also want it to be equally as accurate when used on the future December data. Since the only feature available to us is the Date, we formalize the problem as follows: given only one feature on which to predict, can an accurate and honest (non-overfit) model to predict COVID cases and deaths be derived? We believe that we have done so, and illustrate this in the following subsections.

3 RELATED WORK

There has been a lot of prior research with regards to using machine learning for forecasting the spread of viruses. Since COVID-19 outbreak occurred nearly a year ago, there have alreadu been several expensive papers from distinguished sources that showcase how various machine learning models can be used in order to forecast the spread of the virus in various regions of the world. We will focus our discussion on three of the most prominent methods, which have all been successful in different ways in the task of forecasting COVID-19 spread.

3.1 Deep Learning

One popular method that seems to be quite widespread is the use of deep learning in order to forecast COVID-19. Sun et al., a group of researchers from China, applied deep learning methods in order to predict the spread of COVID-19 within mainland China[1]. Their model was not like most existing deep learning epidemic predictors as it utilized some dynamic parameter changing based on government response and restrictions for the virus. This allowed

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

[1] Sun, Jichao, et al. "Forecasting the Long-Term Trend of COVID-19 Epidemic Using a Dynamic Model." Nature News, Nature Publishing Group, 3 Dec. 2020, www.nature.com/articles/s41598-020-78084-w. Sun et al.

their model to perform very well, "predicting the long trend up to 40 days and the exact date of the outbreak peak." In fact, their results were only 3.8% different from the real results over every day of this 40 day period, which is very accurate. Another group of researchers from the Indian Institute of Technology Patna also utilized deep learning in a different way [2]. Their goal was to create a very highly interpretable neural network based on a fine-grained attention mechanism, with a focus on hidden state dimensions. This framework worked well for this group, as they were able to predict COVID-19 cases well in Canada, Italy, and France with very low RMSE error.

3.2 Variations of Linear Regression

Another popular methodology that has been attempted in many prominent research journals is linear regression, which is especially suitable for COVID-19 given its growth rate in various areas. Rath et al. used a simple linear regression model to predict cases in India, and had a lot of success with this method [3]. They also found a lot of success by combining multiple linear regression models together by solving for several different variables and putting them together to get a final regression output. However, this combination of linear regression models isn't always necessary. Argawu from Ambo University utilized a single linear regression model in order to predict cases and deaths in Ethiopia, and was able to create results with a very low RMSE [4]. As such, linear regression seems to be a viable approach to take for this problem.

3.3 Time Series

Since COVID-19 cases fluctuate a lot with relation to time, various time series models have been used in order to predict future cases. Chaurasia and Pal attempted using several different time series models and eventually found that a simple ARIMA model was best to predict case counts in several different countries [5]. However, other researchers find more success with different time series models. A research group from the University of New South Wales finds that a simple ARIMA model did not work well for predicting case count in Australia, and instead opted for an ARIMA model with much more transformation and dynamic model estimation with special consideration of the timeline of government intervention measures [6]. Regardless of complexity, though, it seems like an ARIMA model will be a very good approach in order to model this problem, as seen in both of these prior publications.

[2]Neeraj, et al. "A Deep Learning Framework for COVID Outbreak Prediction." ArXiv.org, 7 Oct. 2020, arxiv.org/abs/2010.00382.

[3]Rath, Smita et al. "Prediction of new active cases of coronavirus disease (COVID-19) pandemic using multiple linear regression model." 1 Aug. 2020, Diabetes metabolic syndrome vol. 14,5 (2020): 1467-1474. doi:10.1016/j.dsx.2020.07.045.

[4]Argawu, Alemayehu Siffir. "Regression Models for Predictions of COVID-19 New Cases and New Deaths Based on May/June Data in Ethiopia." MedRxiv, Cold Spring Harbor Laboratory Press, 1 Jan. 2020, www.medrxiv.org/content/10.1101/2020.09.04.20188094v1.

[5]Chaurasia, Vikas and Pal, Saurabh. "Application of Machine Learning Time Series Analysis for Prediction COVID-19 Pandemic." Research on Biomedical Engineering, Springer International Publishing, 1 Jan. 1970, link.springer.com/article/10.1007/s42600-020-00105-4.

[6]Rashidi, Taha Hossein, et al. "Real-Time Time-Series Modelling for Prediction of COVID-19 Spread and Intervention Assessment.", www.medrxiv.org/content/10.1101/2020.04.24.20078923v1.

4 PROBLEM DEFINITION AND FORMALIZATION

We formally define our problem statement to be: the collection and maximal utilization of data relevant to the total number of COVID-19 cases and the total number of COVID-19 deaths. This includes the creation of a machine learning model that generates relevant predictions for both the immediate and distant future.

Mathematically, we followed a similar structure to the NIH. We defined the relevant data to be the following:

- P: The population of the state
- P_0: The population of the state vulnerable, set at 0.9P
- R: The rate of spread of the virus in the Unites States
- L: The mean latent period (how long before a case turns symptomatic), set in accordance with NIH data to be 3 days
- I: The mean infectious period, set in accordance with NIH data to be 5 days.
- γ: The transition rate
- t: The time frame

The NIH paper found that the rate of change of intensity of the virus was related to the following quantity:

$$\frac{\delta D}{\delta d}(t) = \gamma H$$

Here, H is the state data, which is in accordance with the quantities above.

We intend to minimize the Mean Average Percentage Error, formulated by: $\sum_i \frac{|P_i - G_i|}{G_i}$,where P is the predicted data value, and G is the ground truth value.

Finally, we arrive at the overarching formalization of our problem statement:

$$P = (P_C, P_D) = (F(t, H, \gamma), F'(t, H, I))$$

Here, P is the prediction data, a tuple of the prediction for the number of cases P_C, and the prediction for the number of deaths, P_D. F is defined as a prediction function for confirmed cases that utilizes state data, time frame, and the transition rate, while F' is defined as a prediction function for confirmed deaths that utilizes state data, time frame, and the infectious period (deaths more likely to climb I days after a surge, via NIH).

Without using the ground truth data, we want to fine tune parameters such that:

$$M = min(P_i, \forall i \in N)$$

This states that our final model will be the one that minimizes MAPE (via cross validation, not ground truth data).

5 METHODS DESCRIPTION

5.1 Data Preprocessing and Transformation

5.1.1 Imputation. Imputation is the process by which data columns that don't have values assigned to them are automatically given one based on the method chosen. In the training data, there is some data missing, such as entries in the hospitalization rate, etc. Since our model cannot deal with NaN entries, we used a SimpleImputer to fill those values in, in a manner comprehensible to our model. For

most data columns, we adopted an "average" strategy, or replacing these NaN entries with an average of the data points around it. This allowed us to make use of the holes in the data instead of just dropping the columns entirely.

5.1.2 Data Cleaning. The Date column is something that models generally don't like. Since it comes in the form of a string, most models will classify each date as its own separate class, and have no way of telling that 08/24/20 is a date that is chronologically smaller than 09/24/20. Because of this, we adopted an approach of converting the date to a single number. The process was as follows: given a day and month from a date, create a new column dno, which is equal to $dno = month * 100 + day * 3$. This provided an almost linear way of representing the dates, one after another. As an example, 8/24/20 would be converted to 872, and 09/01/20 would be 903. Now the model can tell that 8/24 is a date before 9/01. This allowed the model to take in the date as an index-like variable.

5.1.3 Fitting with Features. We attempted fitting our data to different features. By fit transforming our data with polynomial features, we were able to make a linear regression return a polynomial curve. Such fitting allows us to anticipate what degree our data might follow and enables us to use simpler models to characterize complicated data inflections.

5.2 Initial Modeling Attempts

These initial attempts were models we experimented with in order to get a better feel for the problem, which was using only a single feature to predict multiple features. We split this into two sub problems: finding a correlation between the date and the number of cases (an obvious increasing correlation), and finding a correlation between the date and the number of deaths (again, an increasing correlation). Since the only thing we know about our testing data is the date for which we need to generate predictions, this seemed like a good place to start. We primarily looked at regressors, as they generally are well suited for numerical prediction tasks. Once we narrowed down that we only wanted to predict date vs. confirmed and date vs. death, model selection was a simple process. Our basic candidates are outlined below:

5.2.1 Linear Regression. This is the most basic model, the benchmark upon which many complicated ML algorithms are built. The basis of a Linear Regression is a line of the form $y = mx+b$. This line is plotted via a Least Square Regression, which essentially ensures that the line minimizes the average of the variation between it and its corresponding data point. As one might guess, covid's trends cannot be captured by a single straight line. Due to this, our Linear Regression of date vs. confirmed did not give us accurate results.

5.2.2 Polynomial Regression. From the trials with Linear regression, and by plotting the data, it was clear that a model capable of predicting inflections (ie. curves) was required. To adjust our linear model for this, I employed polynomial fitting. This equips our data with polynomial features, and allows the Linear model to now fit curves (ie. instead of a simple y = mx + b, a function of any degree could be graphed). However, this model wasn't sensitive enough, and either ended up massively underfitting or overfitting the data, based on the degree specified.

5.2.3 Decision Tree Regressor. Decision tree's work best when there are ways to classify the data, repeatedly categorizing data until it is confident each one follows a pattern. In theory, this type of classifier should be able to predict turbulent data, but since we only have one feature in the x category, the decision tree ends up massively overfitting, as it makes each data point its own class. This can definitely be improved upon when more data is used, but currently, it is unusable.

5.2.4 Random Forest Regressor. A random forest regressor's prediction is built off of the average of a host of decision tree regressors' outputs. The random forest regressor subsamples the features it is given to create a host of decision trees with slightly different outputs, then averages over the outputs to create a prediction. Once again however, we run into the problem of overfitting, and we unable to get good results from this model.

5.2.5 Multi-Layer Perceptron Regressor. The MLP regressor is one of scikit-learn's many Neural Network models. After hearing the professor elaborate on its popularity and effectiveness, we decided to implement one and test it against some of our other already complete models. Since the training data had more features than the testing data, we adopted the following approach:

(1) Utilize a linear, polynomial, or SVR to fit and predict extra features, like hospitalization rate, against the date.
(2) Once extra relevant data features had been predicted for the range of Sept 1-26, we'd combine them all into one big csv, and use that as the new testing data. The idea was to leverage these artificially created / predicted values to feed the Neural Network more data to improve accuracy.

While the results were certainly more accurate than any of the previously mentioned models, it did not hold up to the standards set by the SVR and ARIMAX models. We found that the data was being underfit in some cases, and overfit in others. This is illustrated via the below plots (confirmed cases as y axis vs date as an integer as x axis):

Figure 1: Date (as integer) vs Confirmed Cases

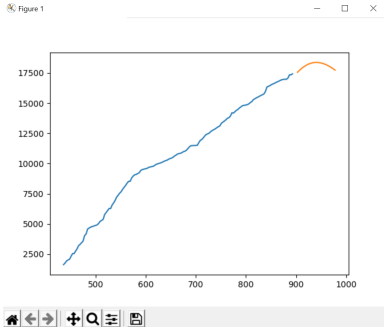
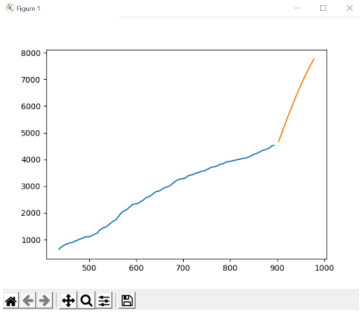


Figure 2: Date (as integer) vs Confirmed Cases



There are many possible reasons for these results:

(1) Error propagation is multiplicative: All the errors that enter the model when trying to predict the extra features multiply with each other, leading to a much higher error right off the bat.
(2) Not enough data: MLP regressors and NN models in general shine when there is plenty of data to crunch. Unfortunately, we didn't have a large amount of data points per state, and hence did not get entirely accurate predictions for the MLP.
(3) Lack of parameter fine tuning: Due to an absence of promising results, the team decided that it was not worth pursuing this model further as we already had significantly more accurate results from other models.

5.3 Support Vector Regression and Linear Regression

The first model that we implemented and tested was one that combined Support Vector Regression(SVR) and Linear Regression. An SVR is a regressor that draws its roots from the Support Vector Machine. It is a supervised machine learning model, suited mainly for tasks involving either a classification or a regression analysis. The model works by mapping example points in space, in such a way that points of different classifications can be clearly separated by a hyperplane. This differs from Linear Regression in that it attempts to find the hyperplane that maximizes the mean distance between all points and the hyperplane [7]. SVM's are further classified into two categories: hard SVM's, where accuracy is generally preferred over usability, and soft SVM's, where margins of error can help generate better trend lines. The SVR is based off of a soft SVM [8]. The SVR works by introducing slack variables so as to minimize the least-square function. Since predicting a number is far more complicated than predicting a classification, the SVR sets a margin of tolerance (that can be set via a parameter), which allows it to approximate an answer to a problem instead of sifting through infinite possibilities for an exact, likely overfitted, prediction.

SVM Pros:

(1) Works very well when features are clearly discernible
(2) Effective when higher dimensions are required
(3) Effective when number of dimensions required is greater than the number of features (as in this case)
(4) Memory efficient.

SVM Cons:

(1) Not suitable for large datasets without the RBF kernel
(2) Does not perform well when data is noisy and inconsistent (not the case except for some states death data)
(3) Not ideal when the number of features exceeds number of data points (not the case here)

Since the disadvantages are not relevant to the case of our current problem, we thought that an SVM would be one of the ideal models for COVID prediction. In fact, a lot of the pros (such as the model being good for discernable features) work very well for our situation here. As such, we developed an extensive SVM model as our first major approach to forecasting COVID cases and deaths.

The reason we decided to use the SVR was mainly due to the power of the RBF kernel. The RBF (radial basis function) kernel is capable of mapping data points to an infinitely higher dimension, in order to find the best separating plane for the data. The RBF kernel draws its roots from the Gaussian Mixed Model, whose power we saw observed in depth in one of our homework assignments. The reason it is used is that it works especially well with sets that have a large amount of data (like the COVID data, which contains over 7,000 rows). 'C' is another parameter to the SVR which defines the relation between the euclidean distance between two points and the euclidean distance between the two points after the kernel's mapping. The larger the value of C, the more likely the model is to overfit. During initial testing, we found that the SVR captured the non-linear trends of COVID extremely well, and decided it was worth exploring further. r

One key idea that dramatically improved the performance of this model was changing the prediction category for confirmed cases. I found that with a lot of states where, at the time, the COVID curve was flattening, the SVR would take that as a sign that it was on a full downward trend, and would predict a curve with negative slope for the month of September. While that might make sense algorithmically, we know that the total number of confirmed cases is a value that can never decrease. Once one is diagnosed with COVID, no matter what happens, their tally is added to the total number of cases and is never removed, even after recovery. Because the SVR was so sensitive to downward trends, we decided to instead make it predict the number of new cases each day. We transformed the training data as follows: given total cases on day x and day x+1, new cases = day(x+1) - day(x). This operation was performed on every single data point for confirmed cases. Once the SVR predicted on this new data, we added them progressively, until it was back in the form of total cases a day. To summarize the procedure, given training data y for confirmed cases: $a = y[0]$, then set $y[0] = 0$ and say $y[i+1] -= y[i]$ for all i. Run the SVR on this new y, and then restore it by setting $y[0] = a$, and then $y[i+1] += y[i]$ for all i. This eliminated all downward trends and helped the model break the baseline accuracy.

[7] Vapnik, Vladimir. "Support-Vector Networks." CiteSeerX, 1995, citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.9362.
[8] Smola, Alex J. "A Tutorial on Support Vector Regression." CiteSeerX, 1998 citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1452.

Before vs After Modifying Prediction Criteria:

Figure 3: Results before modifying criteria

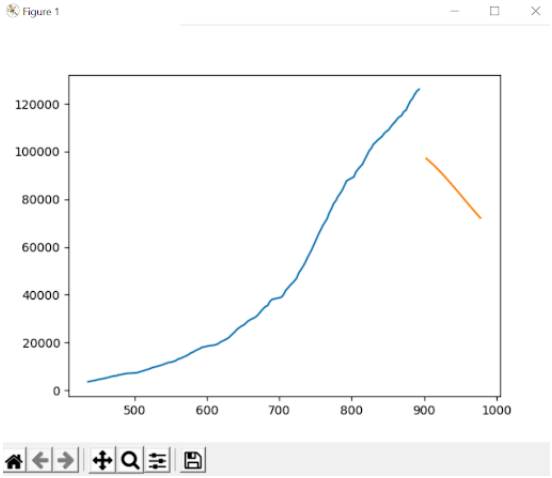
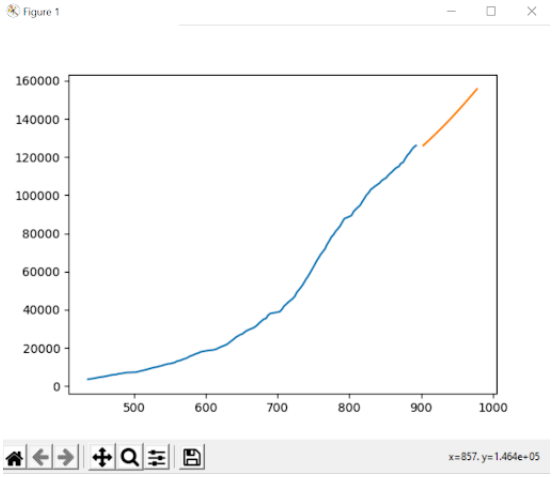


Figure 4: Results after modifying criteria



This prediction post data wrangling worked great for confirmed cases, but death counts before September were still very low and fragmented, so the SVR would still gravely overfit predicted total deaths. We knew that a key part of data mining was observing data and understanding data before using models to predict them. Upon observation, we noticed that deaths are a largely linear trend, and spike at intervals, averaging out over time [9]. Due to this, we decided to make the following adjustment: if the SVR predicts deaths in such a way that the September 26th prediction is less than the September 18th prediction (implying that the SVR overfit and projected a downward curve for total deaths), then switch to a Linear Regression model and repeat the prediction.

[9]Hopkins. (2020). Mortality Analyses. Johns Hopkins Coronavirus Resource Center, https://coronavirus.jhu.edu/data/mortality.

Before vs After Modifying Prediction Criteria:

Figure 5: Results before modifying criteria

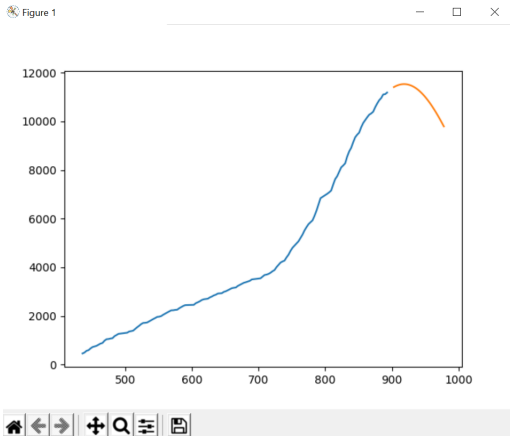
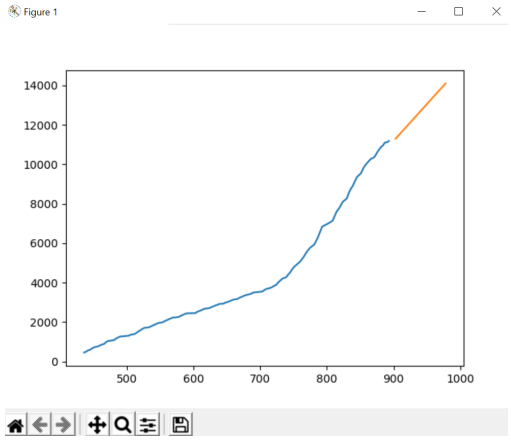


Figure 6: Results after modifying criteria



The Linear regression was not based on the whole training data, but specifically on the training data from 14 days ago to present. The reason for this is that, according to the JHU (3) mortality analysis, deaths generally trend based on measurements from two weeks prior. This has to do with COVID incubation periods, among other circumstances. Upon changing the death predictor to choose between an SVR and a Linear Regression, the final MAPE for this model as shown by Kaggle was 2.415. While this is undoubtedly an accurate model, we wanted to see whether there was another model that could predict more accurately than this model. The following subsections illustrate these efforts.

5.4 Prophet

One popular time series model we were interested in using is called Prophet. Prophet was developed by Facebook's Core Data Science Team. According to its documentation, it is "a procedure for forecasting time series data based on an additive model where nonlinear

trends are fit with yearly, weekly, and daily seasonality, plus holiday effects." We thought this would have a fantastic application towards our problem here. This is because we saw clear non-linear trends in COVID that oftentimes increased day by day, indicating that an additive model may be a good application. We also wanted a model that was robust to outliers. Although there are clear trends in the COVID data, there are also a lot of outliers in the data because of testing and reporting spikes. Luckily, Prophet meets this criteria as well, because it's known to be robust to outliers. Prophet also has the ability to tune itself given certain hyperparameters inputted by the user, allowing us to make tweaks to the model to try and increase its performance. Finally, Prophet can incorporate the effects of holidays into its model. Holidays often cause outliers in trends, and we assumed they would be causing outliers in our COVID data because travel (and therefore the spread of COVID) increases during the holidays as well. Thus, as a whole, we thought that Prophet's strengths aligned with the problem we have, and would be a good choice for a potential solution.

Prophet's model works by using a piecewise linear or logistic (the user specifies whether linear or logistic should be used) growth curve trend, and then automatically identifying inflection points in the data where the trends change. It also incorporates a yearly seasonality component using Fourier variables as well as a weekly seasonality component using dummy variables. It uses all these factors to fit an additive model to the data that is provided. Prophet also gives the user the ability to tune it using some hyperparameters, such as seasonality prior scale, holidays prior scale, seasonality mode, and others. We tuned a variety of these hyperparameters to try and get the best performance out of the model.

Despite this, that Prophet didn't work as well as we expected it to. It gave us a MAPE value of about 3.5, which was not as low as some of the other models we came up with. We have a few ideas for why this may have occurred. The first is that Prophet relies heavily on this idea of identifying trends by yearly seasonality. While this may work great for predicting sales of a product (sales tend to spike during the holiday season and drop off in the spring), it is not as useful for a problem like COVID-19 case prediction because we have not been through enough seasons for any seasonality trends to be noticed. Prophet is also optimized for weekly seasonalities (for example, football engagement is highest on Sundays), but there are not many strong week on week trends for COVID-19. The second is data. Prophet's documentation indicates that it performs best when it is supplied with a full year's worth of data. In our case, we did not have a full year of data, leading to worse performance. Furthermore, we thought that Prophet's ability to handle holidays would be a strength, but it turned out to be a weakness. This is likely due to COVID-19's incubation period, meaning we see case spikes a week or two after a big holiday, and not the day of.

5.5 ARIMA

One of the most powerful time series models is the ARIMA model, which stands for Auto Regressive Integrated Moving Average. While previous models like SVR use the training dataset to predict each day in the test set, ARIMA is autoregressive, so it predicts each day based on all previously observed days, including previously predicted ones. This concept is formally known as using its own lags in prediction. An ARIMA model is characterized by 3 terms: p, d, q. The p term is the order of the autoregressor, in other words number of previous lags to be used in prediction. The d term represents the degree of differencing across the raw observations. The q term is the order of the moving average, which represents the number of lagged errors to bring into the model. ARIMA is a general family of models, and setting p, d, or q to 0 can produce more simple versions. ARIMA's flexibility to be tuned as complex or simple as we desired made it very useful to the team.

Before training an ARIMA model, we had to consider the model architecture from the perspective of inputs and outputs. The Kaggle task required us to predict confirmed cases and deaths across each of the 50 states. We assume that each state's "curve" is independent of one another, since stay at home orders limit the amount of travel. While the relationship between confirmed cases and deaths clearly exists, we opted to model each separately, since death rate can vary from state to state (based on factors such as average age and access to healthcare). Thus we design 2 ARIMA models for each state's cases and deaths, totalling 100 models. This technique is known as ensemble modeling, and proved very effective for the team.

While we train separate ARIMA models for each state, we opt to standardize the hyperparameters to maintain generality within the system. To guess a starting p, d, q we used various quantitative indicators, such as standard plots and autocorrelation plots. Based on our initial guesses, we conducted a grid search of each parameter from 0-5. The optimal hyperparameters for confirmed cases are $p = 3, d = 2, q = 1$ and for deaths $p = 4, d = 2, q = 3$. The lag and moving average parameters vary between the cases and deaths, likely representing a different shape of the graph, while the same degree of differencing represents the same overall family of the curve.

5.6 SARIMAX

SARIMAX is a variant of the ARIMA model family, but introduces seasonality. In addition to the existing p, d, q terms from ARIMA, SARIMAX also has special seasonal hyperparameters P, D, Q, m. P, D, Q are used to model the impact of seasonality on the autoregressive, integral, and moving average terms respectively. The m is used to describe the duration of the season. SARIMAX has been highly successful when modeling outcomes such as retail sales, where there is increased demand during the holiday seasons.

SARIMAX as an algorithm has certain advantages over ARIMA, but the statsmodels library also has better support for SARIMAX. When training ARIMA models, we occasionally ran into "LinAlgError: SVD did not converge" or "LinAlgError: LU decomposition error". Upon searching into stackoverflow forums, we learned that the statsmodels community has focused updates on SARIMAX, since it does everything that ARIMA does and more. The open source community at statsmodels has added boolean arguments enforce stationarity and enforce invertibility, which can be set to false and prevent the above errors.

Since we are working just over 7 months of COVID data, we could not conclusively identify seasonal patterns, although certain regions experienced a second or third wave of cases. Therefore, we chose not to tune any of the additional SARIMAX parameters.

Our SARIMAX model worked incredibly well for predicting both death count and case count. This can be seen in the following two figures (which have time as the x-axis).

Figure 7: Time vs Confirmed Accurate Cases

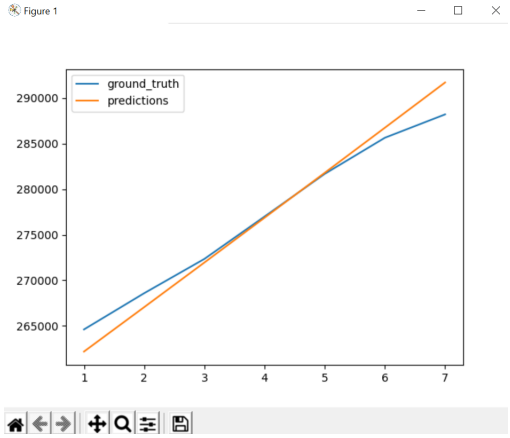
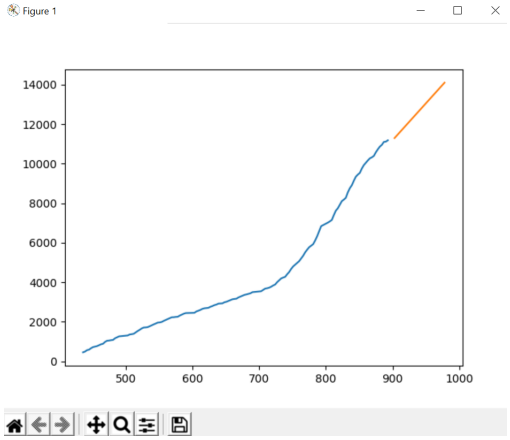


Figure 8: Time vs Confirmed Accurate Deaths



6 EXPERIMENT EVALUATION

6.1 Model MAPE Scores

In order to evaluate our different models, we used a metric called mean absolute percentage error, or MAPE. MAPE is calculated by finding the average absolute percent error for each time period minus actual values divided by actual values. MAPE is one of the most common metrics used to evaluate forecasting models, and is thus a perfect choice for the problem we have at hand.

SVR and Linear Regression	2.415
Prophet	3.501
ARIMA	2.37
SARIMAX	2.30

Table 1: Best Models and MAPE Scores

6.2 Comparison of Models

Since we have implemented and extensively tested four different models, it is important to compare them in order to see which one is the best for our use case.

6.2.1 ARIMA. The ARIMA model is a powerful model that uses time series to build a solid predictive model. It uses previous days' data to make a prediction, meaning it would fair well with a bigger data set. Using an independent model for each state's cases and deaths may have been a disadvantage, but the model pulled off a 2.37 MAPE with some fine tuning of the three hyperparameters it has.

6.2.2 SARIMAX. Due to ARIMA's potential, we have then decided to use the ARIMA variant, SARIMAX. With a little more fine tuning of the new special parameters that came with SARIMAX, we were able to get our best MAPE score of 2.30 due to how advance and compatible the model was. Given more time, SARIMAX's seasonality would have done much better with a longer data span that would have assisted with the model's seasonality parameters because we were only given 7 months of data.

6.2.3 SVR and Linear Regression. Since we had a combination of SVR and Linear Regression in one model, we will treat is as such. We used the Linear Regression model to predict the death cases if a certain conditional passed. The SVR model we worked on relied heavily over the RBF Kernel which assisted in its success with the project's relatively big data set. Since it was out first model and it had fared well with predicting the non-linearity of the data, we decided to use it given the 2.4 MAPE score.

6.2.4 Prophet. Following the promising results of an additive model like ARIMA, we chose to explore with a lesser known model called Prophet. It is a time-series model that checks inflections of where trends change and is based on seasonality. This was our worst model giving us a 3.5, and has suffered by the drawbacks of the nature of the project more than the others, being that we didn't have enough data to detect seasonality yet.

6.2.5 Final Thoughts. After looking at all of these various models and the outputs that they produced, it is clear that the SARIMAX was the best model for our case. It consistently produced the best MAPE results, and fit the training dataset very well. It seems that both ARIMA and SARIMAX, the main time series models that we tried, worked very well for this set, which makes sense since the main dictating variable for both COVID cases and deaths is simply time. Although the Prophet model seemed promising at first, its

model based on inflection checking didn't work well for our use case. Finally, while the SVR and Linear Regression worked well for a while, it is clear that this model capped out early and may have been overfitting to the public Kaggle set. It was incredibly important that we were able to dive deep into each of these models so that we could truly find out the best model for our case, SARIMAX.

6.3 Final Model Evaluation

Our model ended up performing very well on the public Kaggle leaderboard. Our team finished 25th on the leaderboard with a MAPE score of 2.29580. Throughout the process of improving our model, we made a total of 33 submissions. We found that we could've improved our MAPE score further, but decided against this as we were very aware of the threat of overfitting to the public Kaggle data set. In our training, we ensured no test set leakage and abided by all rules set in the competition.

7 CONCLUSION

Our goal for this project was to to create a model that would predict both COVID-19 cases and deaths in the 50 US States. This model would be able to predict our test data for September as well as future December data without deceivingly overfitting.

Upon exploring, implementing, and researching different models, we have found successes and failures. Even though we would have done better with more data, our seasonal model—SARIMAX—has done an excellent job giving us our top MAPE score of 2.30. It would have helped extensively if we were able to focus on time series from the start of the quarter, but this wasn't possible since we found the model late in the process. Hypertuning our parameters helped us quite a few to get from an imported model to a model that was specialized towards this project.

Our initial model attempts didn't provide good results for us until we used the SVR model, which was the first to show real potential. As the quarter went by, we gained information from the teaching assistants and the professor as to which model we should use. Private research of models also helped as we were able to look at published models with different implementations in order to improve our own. A great resource was the research done by the many universities around the world trying to attempt to create a model. This was definitely a universal challenge that we were extremely excited to be working on.

As a final note, our group members feel that this project has been a great advancement to our data science and machine learning careers. We have learned a lot about different models and picking the right models for certain tasks. This gave us a great insight into the top-down approach that's often used to solve data mining projects. From this work, we have gained valuable insight on how to convert raw data into a professional, usable current day model that is useful for predicting COVID-19.

8 REFERENCES

(1) Sun, Jichao, et al. "Forecasting the Long-Term Trend of COVID-19 Epidemic Using a Dynamic Model." Nature News, Nature Publishing Group, 3 Dec. 2020, www.nature.com/articles/s41598-020-78084-w. Sun et al.
(2) Neeraj, et al. "A Deep Learning Framework for COVID Outbreak Prediction." ArXiv.org, 7 Oct. 2020, arxiv.org/abs/2010.00382.
(3) Rath, Smita et al. "Prediction of new active cases of coronavirus disease (COVID-19) pandemic using multiple linear regression model." 1 Aug. 2020, Diabetes metabolic syndrome vol. 14,5 (2020): 1467-1474. doi:10.1016/j.dsx.2020.07.045.
(4) Argawu, Alemayehu Siffir. "Regression Models for Predictions of COVID-19 New Cases and New Deaths Based on May/June Data in Ethiopia." MedRxiv, Cold Spring Harbor Laboratory Press, 1 Jan. 2020, www.medrxiv.org/content/10.1101/2020.09.04.20188094v1.
(5) Chaurasia, Vikas and Pal, Saurabh. "Application of Machine Learning Time Series Analysis for Prediction COVID-19 Pandemic." Research on Biomedical Engineering, Springer International Publishing, 1 Jan. 1970, link.springer.com/article/10.1007/s42600-020-00105-4.
(6) Rashidi, Taha Hossein, et al. "Real-Time Time-Series Modelling for Prediction of COVID-19 Spread and Intervention Assessment.", www.medrxiv.org/content/10.1101/2020.04.24.20078923v1.
(7) Vapnik, Vladimir. "Support-Vector Networks." CiteSeerX, 1995, citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.9362.
(8) Smola, Alex J. "A Tutorial on Support Vector Regression." CiteSeerX, 1998 citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1452.
(9) Hopkins. (2020). Mortality Analyses. Johns Hopkins Coronavirus Resource Center, https://coronavirus.jhu.edu/data/mortality.

9 TASK DISTRIBUTION FORM

Task	Assigned Member(s)
Data Preprocessing	Keshav, Harsh
Adding local MAPE tester	Keshav
Scraping extra data from JHU repository	Keshav
Experimenting with Random Forest regression	Harsh
Experimenting with Decision Tree regression	Harsh
Experimenting with MLP regression	Keshav
Experimenting with linear regression	Keshav, Sultan
Implementing SVR model	Keshav
Implementing ARIMA model	Prithvi
Implementing SARIMAX model	Prithvi, Tejas
Implementing Prophet model	Harsh
Tuning time series prediction models	Tejas, Prithvi, Harsh
Researching related data mining work	Tejas, Prithvi
Formatting final report with LaTeX	Tejas
Formatting midterm report with LaTeX	Sultan
Writing report sections	All Members

Table 2: Task division among members.

All of our group members contributed to the project and were available to help one other out at all times. Our communication and teamwork were core reasons as to why we were able to succeed throughout this project. This allowed us to try out several different models and thoroughly observe our results throughout the process in order to find the best possible solution. We're grateful for this opportunity to work as a team during these times where there is so little interpersonal interaction.