Team 18: CS145 Kaggle Competition Midterm Report

Vaishnavi Tipireddy
tvaishu9@gmail.com
Student ID: 705143552

Steven Luong
trungluong93@gmail.com
Student ID: 605361550

Khoa Le
rubikvn2100@gmail.com
Student ID: 813981162

Danning Yu
danningyu@ucla.edu
Student ID: 305087992

ABSTRACT

This report describes the work that has been done and future plans for Team 18 of the CS 145 Fall 2020 Kaggle competition. The goal of this competition is to predict COVID-19 confirmed cases and deaths for 50 states in America using data from the Johns Hopkins University COVID-19 dataset. Training data will be given for a particular time range (such as April to August) and then the model will predict COVID-19 cases and deaths for a future time range (such as September). This task boils down to the problem of time series prediction: given data about the past, how accurately can we predict the future. This report will cover the data processing work that was done and then move on to discuss the models that have tried, along with their performance. The report will conclude with a discussion of team plans for the rest of the competition.

ACM Reference Format:
Vaishnavi Tipireddy, Steven Luong, Khoa Le, and Danning Yu. 2020. Team 18: CS145 Kaggle Competition Midterm Report. In *Proceedings of CS 145*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1111/1111111.1111111

1 DATA PROCESSING

Since the training data is provided as CSV files with column headers, there is not much data processing that needs to be done to load it into a Pandas dataframe in a Python program. One can simply use the `pandas.read_csv()` function, and from there it is easy to extract the needed data using standard Pandas dataframe methods. The only issue is that some data values are missing from the CSV and thus show up as `NaN` in the Pandas dataframe. To remove these values for a particular column, the function `df.dropna` can be used. Finally, since we need to predict COVID-19 cases for the 50 states individually, it is helpful to get the data for a particular state. To do this, one can use `df.loc[filter]`, where the filter matches values in the "Province_State" column with the desired state.

A transformation was applied to the data to convert dates given in the MM-DD-YYYY format to days since 4/12/2020. This is because for all of our models, we will be using date as the input variable, and so it is easier for the models to handle a numerical input rather than a date input. To do this, each entry in the "Date" column was recalculated to show the number of days since 4/12/2020 by "subtracting" 4/12/2020 from its current value. This new date column that has values ranging from 0 to 141. When predictions are made for September, it will be made for inputs in the range [142, 167] rather than 9/1/2020 to 9/26/2020. This simplifies the data, essentially turning it into a timeseries that starts at time 0. This makes it easier to train models and make forecasts.

2 DATA VISUALIZATION

The first step for any project that aims to create prediction models is to visualize the data so that we have a better sense of what trends are present. Plots of confirmed cases versus time and deaths versus time were plotted for each state, and these plots for California and Maine are shown below in Figure 1. Through these plots, it was found that every state has different rates of increase in their confirmed cases and deaths. The implication of this is that we will probably have to predict data for each state individually. Also, we will probably need to use different hyperparameters or even entirely different models for different states and confirmed cases versus deaths.

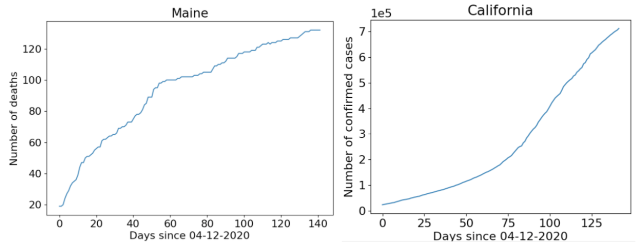

Figure 1: COVID-19 deaths in Maine and confirmed cases in California, showing how different states can have very different case curves. In Maine, there was an initial sharp rise followed by a flattening curve, while the opposite is true for California. Also, the data for Maine is not very smooth, which will also pose a challenge for models.

3 PREDICTION MODELS

3.1 Kernel Regression

This was one of our first ideas, as we covered this model in class through a combination of nonlinear SVMs and regression. As seen in Figure 1, the COVID-19 trends are nonlinear, and so we thought a kernel regression model would be a good way to handle this nonlinear data. After implementing the model using the `scikit-learn` package with a polynomial kernel of degree 3, we realized that the model was overfitting. This is because after it was trained using the first 131 days of data (that is, all data points in the training set except for the last 10), the predictions made for the next 10 days were not very accurate. Even worse, as predictions were made farther out for September, they actually showed a decreasing trend, which makes no sense for cumulative confirmed cases or deaths. This is most likely due to the fact that a polynomial model fits the training data really well, but then has no guarantees of behavior for predictions outside the domain of the training data. Furthermore, polynomials are not guaranteed to be monotonically increasing. The result was an overall MAPE of 18.444% on the test data set for September, which is very high.

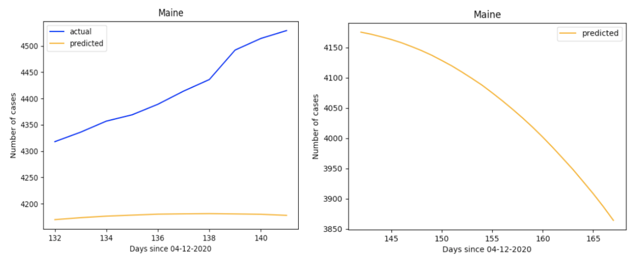

Figure 2: Kernel regression for COVID-19 confirmed cases in Maine with a polynomial kernel of degree 3. The graph on the left shows predictions (orange) versus actual (blue) for 8/22/2020 to 8/31/2020, while the graph on the right shows predictions for 9/1/2020 to 9/26/2020. The further into the future a prediction is made, the lower the result is, which is a nonsensical trend.

After looking at these results, we realized fitting the kernel regression model to the entire set of data from April to August does not make sense, as case numbers in April have little bearing on what the number of cases in September. Thus, we decided to only use the last 5 days of the training data. This resulted in better results for the predictions, as shown in the left graph of Figure 3, as well as slightly better accuracy for predictions farther out, with the curve decreasing later in September than before. With this adjustment, the MAPE error was decreased to 9.423%. The fact that only using the last 5 days of data resulted in more accurate predictions hints at the fact that a good model will probably use a weighted average of the past couple days of data to predict the future, rather than the entire set of training data. Also, since COVID-19 total cases and deaths will only ever go up, we should explore ways to add an increasing monotonicity constraint to our model.

3.2 Autoregression

Autoregression (AR) is the idea that we can predict outputs for a particular day based on a linear combination of outputs from previous days. That is, for an AR model of order p, which means it uses data from the past p days, the output y on day t is determined by the equation

$$y_t = c + \epsilon + \sum_{i=1}^{p} p_i y_{t-i}$$

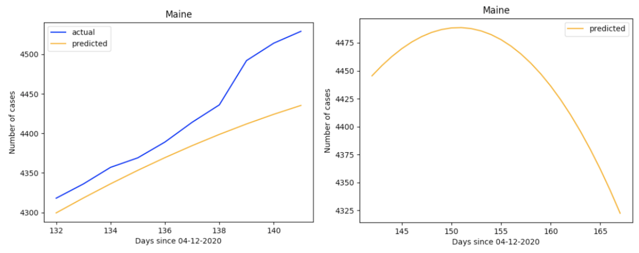

Figure 3: Kernel regression for COVID-19 confirmed cases in Maine with a polynomial kernel of degree 3 trained only on the last 5 days of training data. The graph on the left shows the predictions versus expected results for the last 10 days, while the graph on the right shows predictions for September. Ultimately, the predictions still take on a downwards trend.

where each p_i is a weight that needs to be tuned, c is a constant, and ϵ is some normally distributed noise. Using the `statsmodels` package, I created an AR model with order 4, as using data from 4-5 days ago seemed to work well for the kernel regression model. The model was trained on the first 131 days of data (same as kernel regression) and then used to make predictions on the remaining 10 days of training data. From the graphs below, we can see that for some states like Alabama, the predictions are quite close to the actual data, while for other states such as New Jersey, there is still the issue of the model predictions creating a decreasing trend.

When this model was applied for all 50 states for both confirmed cases and deaths, it resulted in an MAPE error of 8.872%, which is slightly better than kernel regression%. This high number is probably due to the mispredictions for states like New Jersey. There are also states that have not had deaths for a couple days, and so for those states, the AR model predicted no deaths for the future, which is also quite inaccurate. To improve this model, we should do some research into how to create an AR model that is monotonically increasing. Also, the hyperparameter p, which determines how many past days of data the model uses, was not tuned, and so tuning it should also result in slightly better performance. The AR model is also the simplest model in a family of autoregression models, and so we should explore the more complex ones.

3.3 Exponential Smoothing

Exponential smoothing is a time series prediction technique similar to AR, except that the weight of each observation exponentially decreases as the observation gets older. This makes intuitive sense for COVID-19, as the number of confirmed COVID-19 cases from 10 days ago should have a lot smaller effect than confirmed cases from the previous day. Predictions are made using the following base equation (from which more elaborate versions can be created):

$$y_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \ldots$$

In this equation, we are predicting the value of y at time $t+1$ by using past observations. The variable α is called the smoothing parameter and can vary from 0 to 1. If $\alpha = 1$, then this is simply a naive predictor since the equation reduces to $y_{t+1} = y_t$. As the value

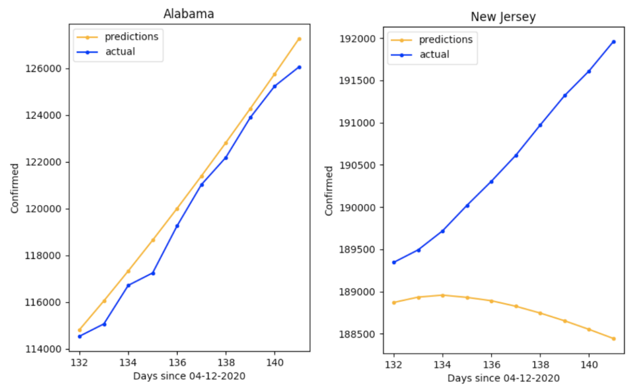

Figure 4: AR model of order 4 for COVID-19 confirmed cases in Alabama and New Jersey. This model works great for Alabama, but has the same issue as kernel regression for New Jersey, where it starts predicting a decrease in total cumulative COVID-19 cases. Some tuning needs to be done; a model of order 4 may not be appropriate for every state

of α decreases, so does the weight of the more recent observations. Thus, α here is an important hyperparameter that needs to be tuned.

I used the `statsmodels` package to create a Holt's model, which is a particular variant of the exponential smoothing model. Holt's model elaborates upon the simple exponential smoothing model above by adding a trend component, which has its own hyperparameter β, which is called the smoothing trend. By adding a trend component, this ensures that the predictions will be monotonic, thus addressing some of the difficulties observed with the autoregression and kernel regression models. This model can be described using the following recursive equations, where y is the prediction, h is how many days into the future we want to predict, l is called the level component, and b is called the trend component. The level component determines the magnitude of the prediction, while the trend component determines the tend of the predictions. The level component has a hyperparameter α associated with it and the trend component has a hyperparameter β.

$$y_{t+h} = l_t + hb_t$$
$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

When I created the model, I set the smoothing level to 0.4 and smoothing trend to 0.2. When trained on the first 131 days of data, this model produced predictions for the next 10 days that were quite accurate, as shown by the graph below for Maine. Most importantly, when predictions were made for September, for all 50 states, the data had a positive slope for both confirmed cases and deaths. Thus, this avoids the issue of the prediction curve going down or staying flat that has been observed with autoregression and kernel regression. Using this model score produced an excellent MAPE of 2.309%, which is the team's current submission on Kaggle. Thus, this model is one of the most promising ones, and so there are plans to spend more time studying this model and its variations in subsequent weeks.

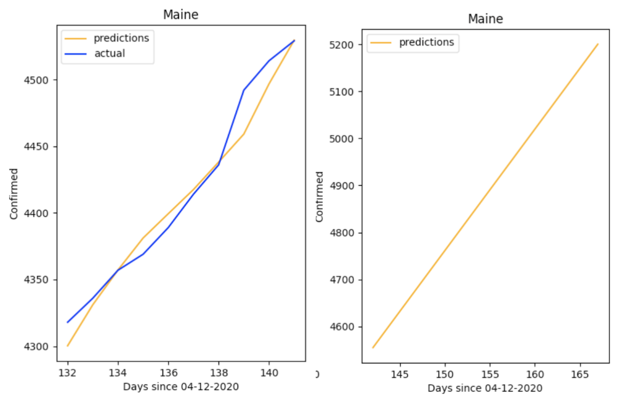

Figure 5: Predicted COVID-19 cases for Maine for the last 10 days of September using a Holt's exponential smoothing model with smoothing level = 0.4 and smoothing trend = 0.2. We can see that the predictions match the actual case counts quite closely, making this a good model.

3.4 MLP Regression

MLP regressor (Multi-layer perceptron regressor) is a type of feed-forward artificial neural network that consists at least three layer: input, hidden and output layer. Between those layers, there are some activation functions to handle back-propagation. As a result, MLP could work better than linear prediction models in handling nonlinear data.

When applying the MLP regressor with only column "Date" as texttttX_train to predict the numbers of confirmed cases and deaths, an MAPE of 3.36% was obtained. The MLP regressor works based on the idea of neural network, so it needs to be fed more than one columns in training set and more data in general to achieve better accuracy. In the early stage of development, only the date column is fed to prediction model, hence it's accuracy is lower than some other models.

For the future optimization, we should include more data columns to training set other than date column. For example, when predicting for number of deaths, some other features such as number of people tested, number of people hospitalized, and mortality rate would be useful to increase the accuracy of this MLP model.

3.5 Linear Regression

We wanted to see the strength of the linear correlation between the days passed and the number of confirmed cases and deaths and evaluate the performance of a simple naive model. We implemented a simple linear regression model using the `scikit-learn` package, and the resulting predictions had an MAPE of around 14.57%. As expected, the error for this model is much higher than for the other models because the predictions for this model are based solely on the number of days that passed since the first day data was available (4/12/2020). Also, unlike some other models used, such as autoregression, linear regression tries to fit all the data at once instead of using data from the last couple days to make predictions. As noted before, the data from months previous is intuitively not

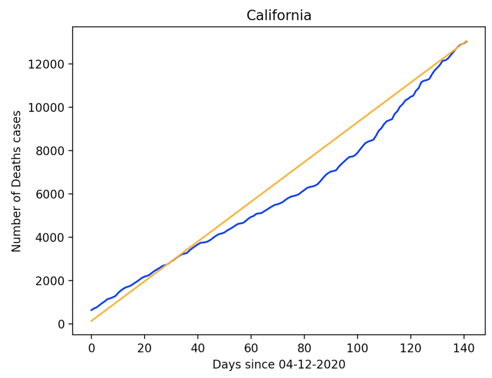

Figure 6: Deaths due to COVID-19 using MLP regressor model. The prediction (yellow line) closes matches the actual cases (blue line)

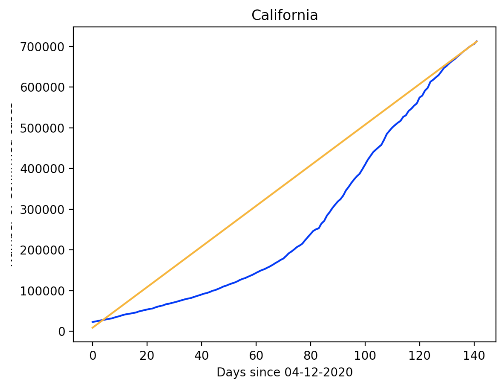

Figure 7: Confirmed COVID-19 cases using MLP regressor model. The prediction does not work well because the actual number of cases is not perfectly linear. Hence, a non-linear prediction model is needed for better accuracy.

as indicative of cases and deaths the next day. An optimal model would mainly only consider data from the previous few days.

3.6 Recurrent Neural Network (vanilla version)

This model is another approach to using a simple neural network to learn patterns of the time series data. This vanilla version of the recurrent neural network (RNN) is similar to a feed forward two layer fully connected neural network that has seven input nodes, one output node, and fifty nodes in a hidden layer. The training data is segmented into chunks of eight contiguous data points, resulting in 135 chunks of training data for each state. For each chunk of eight contiguous data points, the first seven are used as input and the last one is the actual value that will be used to calculate the error that will generate the backward update to the weight.

This vanilla version of the RNN is a simple approach to time series data analysis. But it easily overfits and leads to extreme predictions when the data has irregularities, such as a sudden increase in cases.

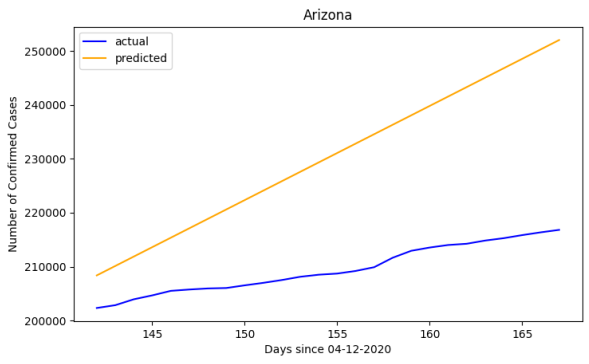

Figure 8: Number of confirmed COVID-19 cases in Arizona in September predicted using linear regression. The prediction values are much higher than the actual number of cases.

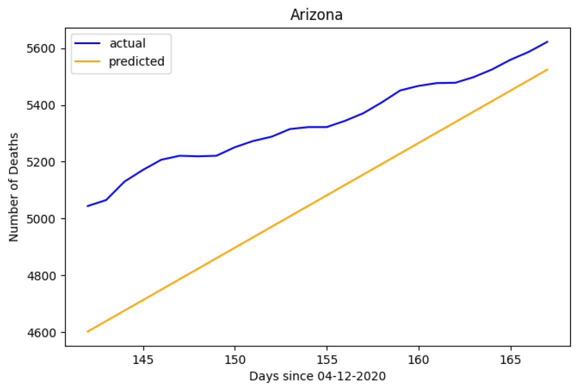

Figure 9: Number of deaths due to COVID-19 in Arizona in September predicted using linear regression. The prediction consistently underestimates the number of deaths.

3.7 Model Summary

To summarize, Table 1 shows all the models that were explored and their accuracies, as well as the baseline model's accuracy as a point of reference.

Table 1: Mean Absolute Percentage Errors (MAPE)

Model Name	MAPE Score (%)
Baseline	2.93
MLP Regressor	3.36
Kernel Regression on all data	18.44
Kernel Regression on last 5 days of data	9.42
Autoregression of order 4	8.87
Exponential Smoothing	2.31
Linear Regression	14.57
RNN (vanilla version)	16.89

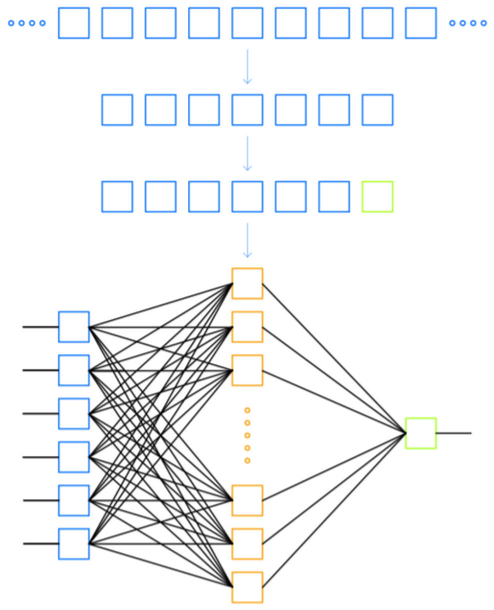

Figure 10: The structure of the vanilla RNN which is a fully connected neural network with 50 hidden nodes. The input and output of the network is a subsequence of the time series data.

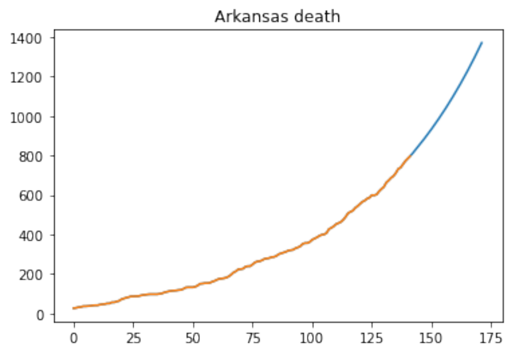

Figure 11: The prediction of vanilla RNN fit well in this case for deaths in Arkansas where the series is smooth.

4 FUTURE WORK PLANS

Our first step moving forward is to narrow down the list of models we're working with and build on the most promising ones. For all of the results shown in this report, we only used date, confirmed cases, and deaths information from `train.csv`. Thus, the main goal over the next couple weeks is to incorporate data from `graph.csv` and the other features from `train.csv` into our model. For example, we plan use a vector autoregression model (autoregression but with multivariable inputs) and compare it to the single-variable autogression model. We also plan to identify which features have the strongest correlation with our outputs (confirmed cases and deaths). This can be done with either correlation plots or principal component analysis. Finally, once we identify which inputs are most important and narrow down our models to 1 or 2, we will shift focus to fine-tuning our model through optimizing their hyperparameters.

Ultimately the purpose of all these steps is to decrease our error rate; we will have to constantly evaluate how important any changes are to yielding more accurate predictions. We also understand that, even with a very robust model, each state may follow different patterns, so an ensemble method may be appropriate to handle this variability. Alternatively, we could also train multiple models on the states and then pick the one that works best to make predictions. This way, the final predictor for each state uses a model that has been optimized for that particular state and whether it is predicting confirmed cases or deaths.

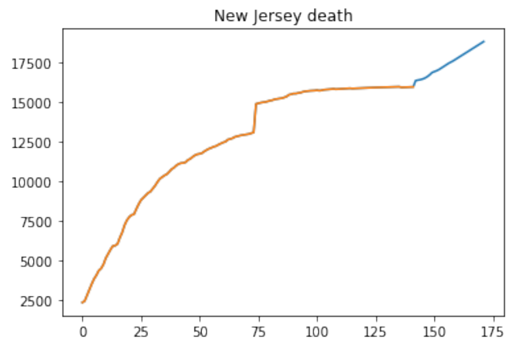

Figure 12: The vanilla RNN overfitted in this case for New Jersey deaths due to the step in the training data.

Table 2: Future Timeline

Week	Goal
6	Research and incorporate multivariate data (other columns + graph.csv)
7	Better understand models, narrow down which models to use
8	Better understand models, narrow down which models to use
9	Hyperparameter tuning, ensemble methods
10	Submit final model to Kaggle, work on report
Finals	Submit final report