Long Supervised Learning Project: COVID-19

Cody Black (5033) and Reza Niazi (5033)

Abstract

The COVID-19 pandemic has and will continue to be the subject of advances in many areas of science and technology. Whether it is finding a vaccine or creating models that can predict and help mitigate the spread of the virus, there are already vast amounts of literature on this topic. In this project, we use Elastic Net Regression in order to to predict the number of deaths per day on a state and national level. In addition, a Neural Network approach to binary classification is used to determine whether the 7-day average number of new cases in the US would increase or decrease in the next week based on the current week's daily new case numbers.

Project Domain

In this project, a subset of the common SL techniques are used for prediction, in the form of regression and classification. The original datasets that are used come from Kaggle [5], but these were modified in order to create the datasets in our analysis. One method used to do prediction will be Elastic Net Regression, which is used to get the benefits of both the Ridge and Lasso regression techniques. Specifically, the Elastic Net Regression is quite proficient at preventing overfitting and detecting multicollinearity. The regression is performed on two different datasets, one with information about the number of deaths per day in California and the other with the deaths per day in the United States. For classification, a Neural Network and a simple Perceptron will be used. The classification will use a dataset containing the number of cumulative COVID-19 cases reported in each county in the US on each day.

Literature Review

Our use of regression is similar to the work in [1]-[4]. However, the data from the California and US Deaths dataset has more variation than the ones in those studies, in the sense that the datasets from [1]-[4] are for a more truncated period of time and usually exhibit monotonic behavior, whereas ours do not. In addition, the articles [1]-[4] focus less on prediction of deaths per day, but rather the total deaths and total cases, which of course are monotonic. We wanted to see how well we could model a more varying/challenging dataset like the number of deaths per day instead. The main tool for regression in [1] is Support Vector Regression, Neural Network in [2], and Ordinary Least Squares in [3] and [4].

Experiment Review

Since our experiments mainly deal with regression/prediction and classification, we will split the experiment review into these two sections.

Regression

Hypotheses:
One set of hypotheses that we have is about the Elastic Net Regression. For the California and US deaths per day datasets, we predict our model will outperform a polynomial regression model. As each dataset has its own model, these will form two different hypotheses. Both sets of these regressions will be performed on the Elastic Net code. Finally, we also hypothesize that our models created with the Elastic Net code will be able to outperform the standard scikitlearn Support Vector (SVR) regression.

Implementation:
To implement the Elastic Net Regression, we performed stochastic gradient descent in order to calculate the regression coefficients of our model. The cost function that was to be minimized is

$$C(X, Y, \beta) = \frac{1}{n}\left(\sum_{i=1}^{n} ||Y_i - \hat{Y}_i||^2 + \lambda_1 \sum_{j=1}^{k} |\beta_j| + \lambda_2 \sum_{j=1}^{k} \beta_j^2\right)$$

where there are n data points and $k+1$ parameters $(\beta_0, \beta_1, ..., \beta_k)$ in the model. We can therfore workout the components of the gradient to be

$$\frac{\partial C}{\partial \beta_0} = -\frac{2}{n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)$$

$$\frac{\partial C}{\partial \beta_j} = \frac{1}{n}\left(-2 \sum_{i=1}^{n} X_{ij}(Y_i - \hat{Y}_i) \pm \lambda_1 + 2\lambda_2 \beta_j\right) \qquad j \in \{1, 2, ..., k\}$$

In order to perform cross validation, we used the first 75 percent of the data to train our model, and the last 25 percent was used to test/validate. Since the data is time dependent, it was essential to choose data points that are sequential and not random. With this split in data, we determined to have $\eta = .01, \lambda_1 = 0, \lambda_2 = 0,$ iterations $= 100000$. For the Support Vector Regression, the same training and testing data set was used, and the default parameters were used when making the regressor object. Now, we can look at the California and US datasets, to justify our choice of model on each dataset.

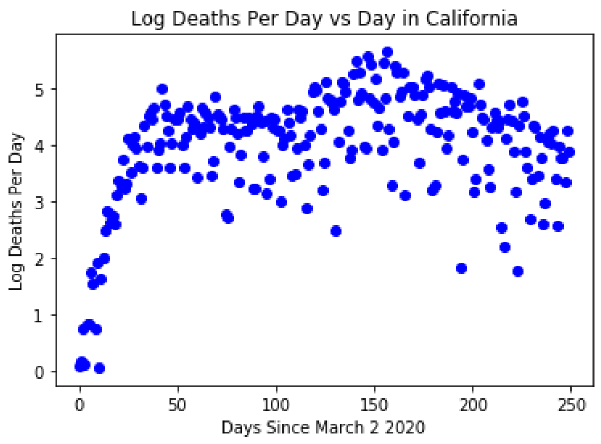

Figure 1

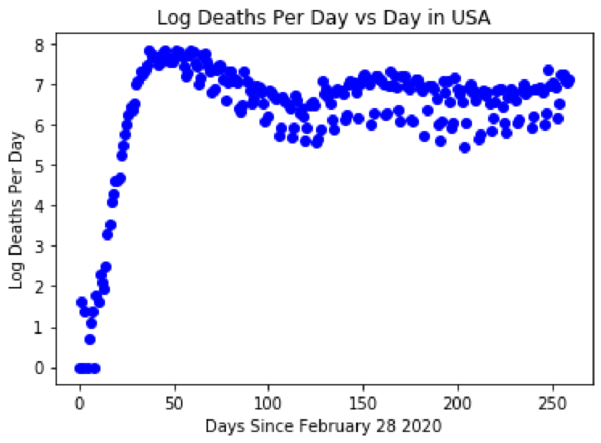

Figure 2

The general idea in both models is that the number of deaths will oscillate, with non-constant amplitude, but eventually the deaths should decay to 0. Therefore, in both models we have a trigonometric expression along with an exponential damping term. Noting that the oscillatory trend holds except for the transient behavior in the beginning when the cases spiked drastically, we also added logarithmic term to account for this. Finally, we also performed the log transform of the response in an attempt to linearize the psuedoexponential data as much as possible. For the California data in Figure 1, we proposed the model

$$\ln(\text{Deaths Per Day}) = \beta_0 + e^{-\text{day}/10}\bigg(\beta_1 \sin(\pi\text{day}) + \beta_2\text{day}\bigg) + \beta_3 \ln(\text{day} + .2) \tag{1}$$

For the US data, we noticed that the data looks very similar to the curve generated by the Fresnel Integral $C(x) = \int_0^x \cos(t^2)\ dt$. As this again has an oscillatory nature that is desired, we add damping to this and the logarithmic expression in order to make it viable for the future. Therefore, the model for US data in Figure 2 is

$$\ln(\text{Deaths Per Day}) = \beta_0 + \beta_1 e^{-\text{day}/10} C(1.1\text{day}) + \beta_2 e^{-\text{day}/2} \ln(2\text{day} + .1) \tag{2}$$

Evaluation Metrics:
The evaluation metrics that are used are the R^2,R_a^2, and Mean Squared Error (MSE) values of the model calculated on the entire dataset. The R^2 value explains what percentage of the variability in the data can be explained by our model, and the R_a^2 value close to R^2 indicates that our model is not performing well as consequence of blindly adding many predictors. The MSE tells us,in some sense, how close our fitted values are to the actual values. Finally, we look at the regression curve and how well it fits the data, especially in the 194 days and on range. This is because this is the range of days in the testing data.

Results:
After fitting both datasets with the models (1) and (2) respectively and then fitting a 4th order polynomial model to both datasets, we get the following results. The 4th order model was better than the cubic in both cases, and so this was chosen. Higher order terms would have just produced similar results, and this is why we chose the order to be 4. Note that the R^2, R_a^2, and MSE values are in the caption of each respective image. However, the SVR models do not have R_a^2 values.

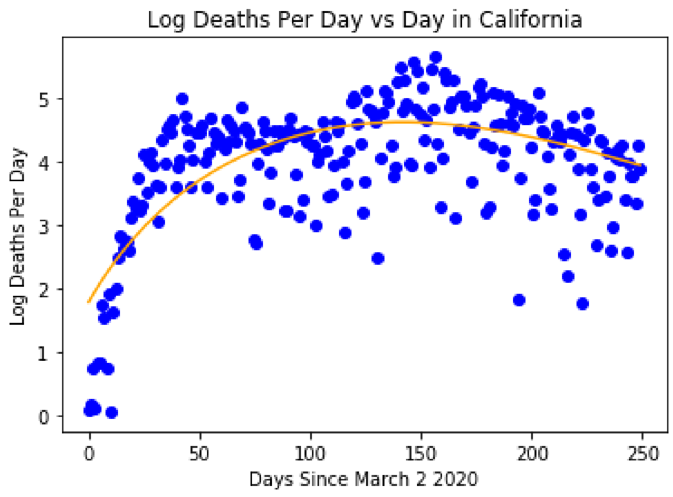

Figure 3: Elastic Net Regression used to fit the model in equation (1). $R^2 = .4592, R_a^2 = .4503, MSE = 1.1689$

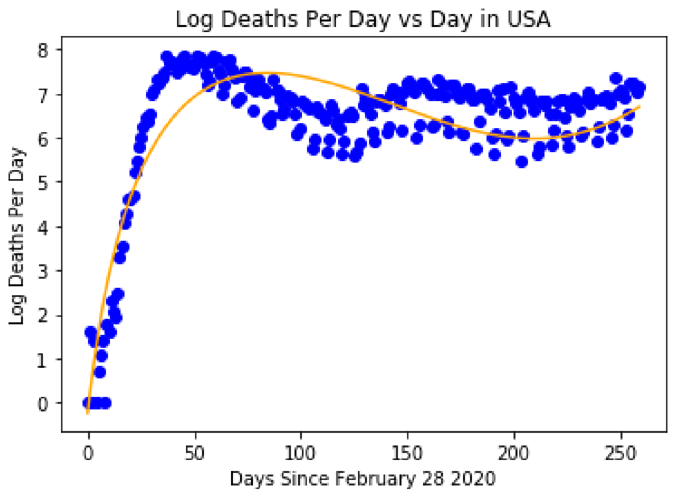

Figure 4: Elastic Net Regression used to fit the model in equation (2). $R^2 = .7417, R_a^2 = .7390, MSE = 3.6360$

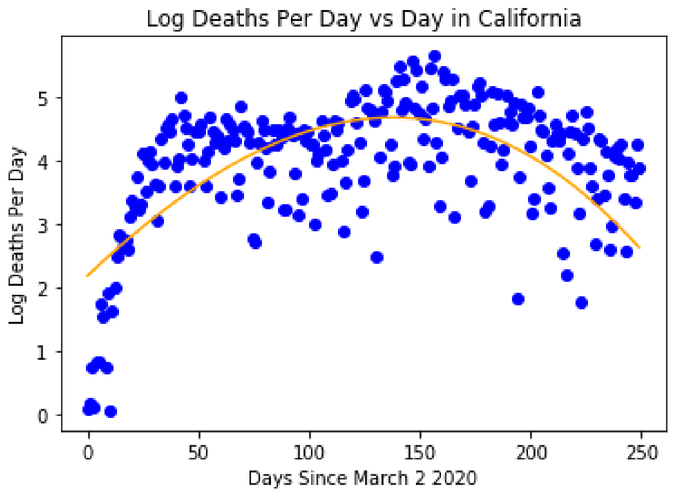

Figure 5: Elastic Net Regression used to fit a fourth order polynomial model. $R^2 = .3746, R_a^2 = .3608, MSE = 2.1064$

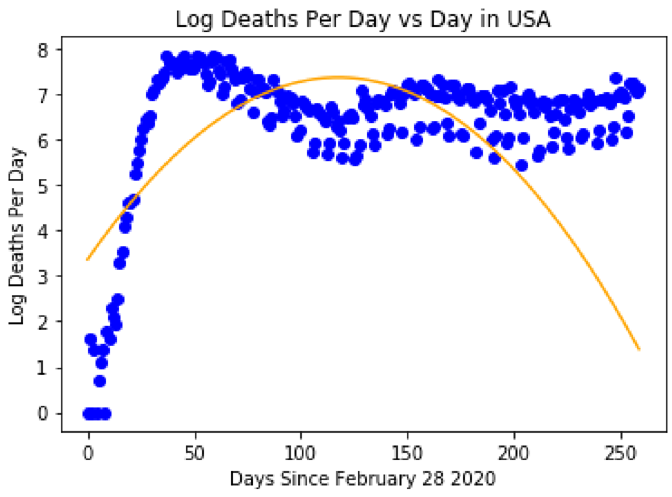

Figure 6: Elastic Net Regression used to fit a fourth order polynomial model. $R^2 = -0.7564, R_a^2 = -0.7936, MSE = 139.2926$

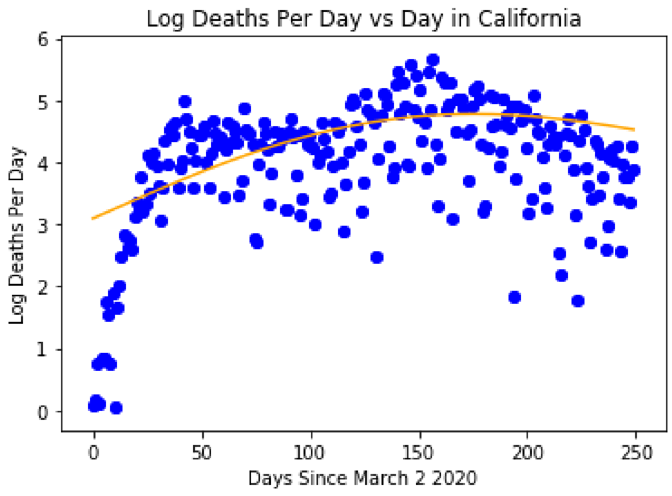

Figure 7: Support Vector Regression. $R^2 = .1921, MSE = 29.070603549616422$

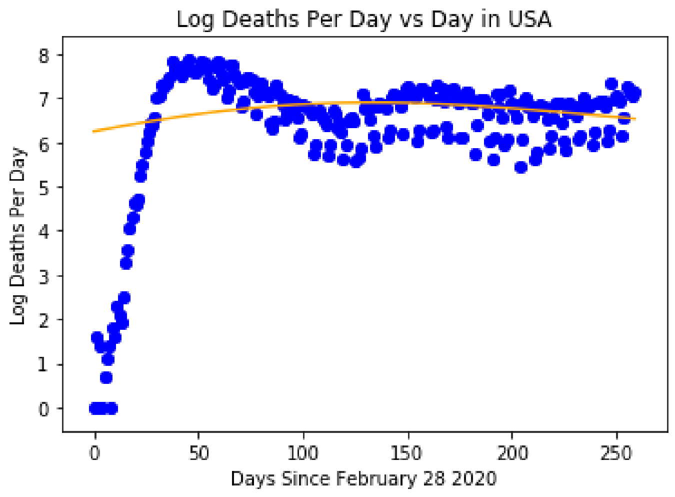

Figure 8: Support Vector Regression. $R^2 = .0730, MSE = 25.532075391692374$

Analysis:
Looking at Figure 3 and 5, we can see that the R^2 and MSE values of our proposed model are better than the fourth order polynomial one. Though, the results are closer than one might expect, since the data does have a quadratic/quartic form to it. However, it is unlikely that this trend would continue in the future, and it is for this reason our model is a much better option. Finally, as our R_a^2 value is close to R^2, we know that our fit was not a consequence of just adding multiple terms. Next, if we look at Figure 3 and 7, we can see that our Elastic Net model again is superior to the Support Vector model. Though, the only major difference can be seen at how it models the early explosion in cases.

Now, we can move onto the USA data. Looking at Figures 4 and 6, it is quite clear that our model is much better than the fourth order model. This is corroborated by the R^2 and MSE values. Again, our Elastic Net model is better than the Support Vector one, which can be seen by looking at Figures 4 and 8, respectively. As in the California data, the two models become quite good in the later days, but the Support Vector model does not capture the quick rise in cases in the beginning.

Classification

Hypotheses:
We also wanted to experiment with making predictions using classification. The idea was to give a model recent data and have it make general predictions about the future. In this case, we wanted to see if a model could predict if the average number of new cases would increase or decrease in the next week based on the current week's daily new case numbers. Since all of our inputs are continuous values, we decided to use a neural network, as it would not require us to discretize our inputs.

Our first classification experiment involved varying the hidden layer size of a neural network with 1 hidden layer. Our goal was to find the best hidden layer size to use for the remaining experiments. In the second experiment, we used the neural network that performed the best in the first experiment and compared its performance to a perceptron. For the first and second classification experiments, we used a relatively small dataset that contained data about the first two weeks of April. However, for the third classification experiment, we created a much larger dataset that included every grouping of 7 consecutive days from late January to early December. We trained a new neural network using the same hidden layer size as the second experiment and compared its performance to the neural network from the second experiment. The goal of this experiment is to determine if the neural network can learn a set of weights that applies to the entire year instead of just a couple of weeks.

Implementation:
Using a dataset that contained the cumulative number of COVID-19 cases for each county in the US and another dataset that gave the population for each county in the US, we calculated the 7-day average of new cases per 100,000 people for each day in each county. For our training dataset for the first and second classification experiments, we used the first week of April and a value of 0 or 1, which indicated whether the second week of April had more cases than the first week for that county. For our validation and testing datasets, we used the second week of April and an additional value of 0 or 1, which indicated whether the third week of April had more cases than the second week. The validation dataset contained a random selection of 80% of the counties, and the testing dataset contained the remaining 20% of the counties.

For the third classification experiment, we created a dataset that included every grouping of 7 consecutive days. For example, instead of a dataset containing the weeks 4/1/20 to 4/7/20, 4/8/20 to 4/14/20, etc. this dataset contains the weeks 4/1/20 to 4/7/20, 4/2/20 to 4/8/20, 4/3/20 to 4/9/20, and so on. For each week, the dataset contains a 0 or 1 which indicates whether the following week had an increase or decrease in new cases. This is done for each of the 3,000+ counties in the US, which results in more than 700,000 total examples. The training dataset consisted of 80% of the examples selected randomly, and the validation dataset consisted of the remaining 20%.

Our neural network implementation is based on the backpropagation algorithm from Chapter 4 of the Mitchell Machine Learning book [6]. The neural network is made up of several neurons, and in our Python code, this is implemented as a dictionary (the neural network) that contains two lists of dictionaries (the hidden layer and output layer neurons). Each neuron dictionary contains a list of weights and a bias weight. We used a sigmoid activation function, a learning rate of 0.01, and iterated over the datasets 200 times. For the first experiment, where we vary the size of the hidden layer, we found that a large hidden layer did not lead to much, if any, improvement. Additionally, as the number of neurons increased, the runtime of our code increased substantially. As a result, we limited ourselves to testing neural networks with hidden layers containing 1-10 neurons.

Evaluation Metrics:
For each model, the confusion matrix was calculated. Using the values from the confusion matrix, we were able to calculate many metrics such as true positive rate (TPR), false positive rate (FPR), precision, etc. However, in order to choose which neural network performed the best in the first experiment and to be able to compare performance between experiments, we had to choose a small subset of these metrics to focus on. As a result, we focused mainly on the TPR, the FPR, and the Youden index, which is the difference between the TPR and FPR. We also plotted the ROC curve for the neural network in the second and third experiments, which allows you to visually compare

the performance of the neural network to that of a random classifier.

Results:
For the first experiment, after fitting the neural networks with different hidden layer sizes using the training dataset and predicting over the validation dataset, we get the results shown in Table 1 below.

Hidden layer size	TPR	FPR	Youden index
1	0.6302	0.2643	0.3659
2	0.6366	0.2726	0.3640
3	0.6405	0.2759	0.3646
4	0.6418	0.2749	0.3669
5	0.6224	0.2520	0.3704
6	0.6430	0.2716	0.3714
7	0.6366	0.2659	0.3707
8	0.6443	0.2708	0.3735
9	0.6237	0.2529	0.3709
10	0.6224	0.2512	0.3712

Table 1: Results from the first classification experiment, which compared neural networks with different hidden layer sizes.

According to the Youden index, the neural network that used 8 neurons in the hidden layer seems to have the best performance. For the next two experiments, a neural network of the same hidden layer size is used. The results of using the perceptron and the neural network with 8 neurons in the hidden layer on the test dataset are shown in Figure 9 and Table 2 below.

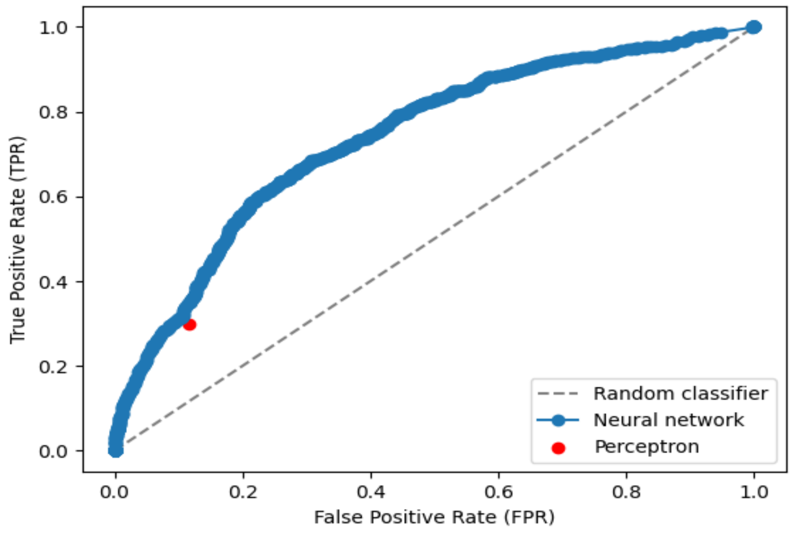

Figure 9: ROC curve for neural network from the second classification experiment.

Model	TPR	FPR	Youden index
Neural network	0.6684	0.2566	0.4118
Perceptron	0.2979	0.1154	0.1825

Table 2: Metrics from the second classification experiment.

The neural network clearly performed better than the perceptron on the test dataset. For the final experiment, a neural network with the same hidden layer size was used on the much larger dataset, and the results are shown in Figure 10 and Table 3 below.

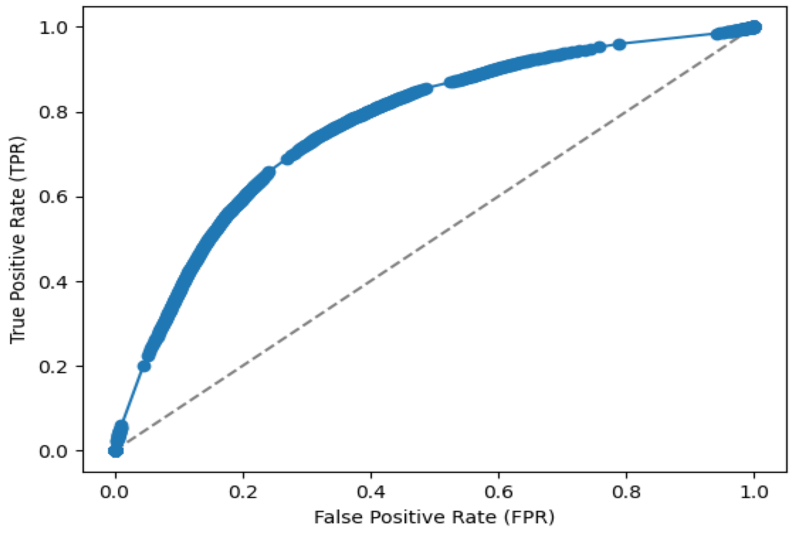

Figure 10: ROC curve for neural network from the third classification experiment.

TPR	FPR	Youden index
0.7128	0.2900	0.4228

Table 3: Metrics for third classification experiment.

Despite being fitted to and making predictions for a much larger dataset that covered a longer period of time, the neural network is able to achieve slightly better performance than in the other two classification experiments.

Analysis:
Overall, the neural network's performance was not amazing, but that was somewhat expected based on the limited data that it was given to make a prediction. Perhaps if it was given more data that related to the number of new COVID-19 cases, such as what restrictions were in effect in an area or the mobility of the local population based on cell phone data, it could make correct predictions at a higher rate.

In the first experiment, all of the neural networks have very similar performance despite having different numbers of neurons in their hidden layer, which was not an expected result. This may further support the idea that the neural network is being limited by the data being given to it, since changing the structure of the network doesn't change the result in a meaningful way. The neural network performed better than a perceptron in the second experiment, which is not surprising given the relatively simple way in which the perceptron makes a binary classification. However, it at least indicates that the neural network was able to find some sort of pattern that allowed it to make correct predictions in some cases. In the third experiment, the neural network achieved a slightly higher level of performance compared to the first and second experiments. This seems to indicate that there is some sort of correlation for new cases between all weeks of the year and not just the first two weeks of April.

Comparison with Previous Work

For regression, the articles [1]-[4] had better results than those of our models. This is to be expected as most of the data that was being used had much less variation over time and variance between data points. Therefore, it would be difficult to draw too many conclusions by just looking at the results of the papers. Another difference is that our chosen models take into account the oscillating nature of the deaths/day and has a damping term to account for the realistic decay in deaths over time. For these reasons our experiments were essentially chosen to compare our Elastic Net Regression model to some of the prevalent models in the papers. The first set of comparisons come between our model and polynomial regression. As can be seen, our chosen models perform much better than their polynomial counterpart.

As [1] uses Support Vector Regression with great efficacy, we chose to compare our model to this form of regression. Just looking at R^2 and MSE values, our model seems to work better, but in terms of making predictions about the testing data, they both perform quite well. However, it is unlikely that the Support Vector Regression models will have better long term efficacy, as they are only fitted on the given data and don't have the purposeful decay and oscillations added to them. Likewise, they do not model the early blowup of cases well.

Conclusion

Despite the presence of biological models, in the form of ODEs or PDEs, used to predict the course of epidemics, machine learning techniques have shown to be quite effective in modeling various aspects of the COVID-19 pandemic. The findings in our project only corroborate this. The use of regression for prediction and a neural network or perceptron for classification have shown to be somewhat effective in making predictions about complex trends like the deaths per day or 7-day average number of cases in the US. As the pandemic continues, more data will become available, and this would allow us to better train our models.

One way to improve our regression approach would be to possibly try other regression techniques, or have our model not only be dependent on the day. For example, we could also use the total number of positive cases, hospitalizations, or even the weather as predictors. For classification, adding more predictors could also help improve the performance of the neural network. Of course, making the neural network more sophisticated would also be helpful in the future.

References

[1] Milind Yadav, Murukessan Perumel, and M. Srinivas *Analysis on novel coronavirus (COVID-19) using machine learning methods.* Chaos, Solitons & Fractals Volume 139

[2] Sina F. Ardabili, Amir Mosavi, Pedram Ghamisi,Filip Ferdinand,Annamaria R. Varkonyi-Koczy, Uwe Reuter, Timon Rabzuk, Peter M. Atkinson *COVID-19 Outbreak Prediction and Machine Learning* . SSRN 3580188

[3] Gaurav Pandey, Poonam Chaudhary, Rajan Gupta, Saibal Pal *SEIR and Regression Model based COVID-19 outbreak predictions in India* . arXiv preprint arXiv:2004.00958, 2020 - arxiv.org

[4] Samit Ghosal, Sumit Sengupta, Milan Majumder, Binayak Sinha *Linear Regression Analysis to Predict the Number of Deaths in India due to SARS-CoV-2 at 6 Weeks from day 0(100 cases-March 14th 2020)* . Diabetes & metabolic syndrome vol. 14,4 (2020): 311-315

[5] Sudalairaj Kumar *https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset* Kaggle

[6] Tom M. Mitchell *Machine Learning.* McGraw-Hill, 1997