Multivariable Linear Regression and Neural Network for COVID-19 Factor Prediction

Andrew Duffy, Mahin Master

CS4100 Artificial Intelligence, Khoury College of Computer Sciences, Northeastern University, Boston, United States of America
duffy.a@northeastern.edu, master.mah@northeastern.edu

Abstract

Regression analysis is a machine learning application in which a set of algorithms are used to predict an outcome or target variable based on the value of input predictor variables. There are many algorithms that can be used for the purpose of supervised machine learning in the regression algorithm space. These algorithms are used to construct a model of the data for prediction purposes. We are studying the use of Multivariable Linear Regression (MLR) as well as Neural Network Regression (NNR) for COVID-19 data. By training these regression algorithms with historical COVID-19 data, future values can be predicted given a set of values for input variables. After applying both regression algorithms to a dataset of daily U.S. COVID-19 statistics, different degrees of accuracy were found in each state.

Introduction

The COVID-19 pandemic's impact is difficult to predict due to volatile and inconsistent data accumulation. This can be attributed, in part, to fluctuating testing availability, changes in our understanding of symptoms, and varying administrative responses to the crisis around the world. Using machine learning, we can provide clear numerical predictions for crucial COVID-19 impact factors (mortality rate, expected case number, etc.) by running regressions on previously recorded statistics. Supervised machine learning looks at past data to make future conclusions. We can accomplish this by setting a series of input variables which combine through some function *f* such that *f([independent variables]) = D* where *D* is some dependent (target) variable.

The data values to be predicted are quantitative and non-binary, so regression is more applicable than classification. The two regression algorithms chosen, Multivariable Linear Regression (MLR) and Neural Network Regression (NNR), are both well suited for this purpose. MLR uses a straightforward linear combination of the given independent variables to calculate the target variable's value but assumes linearity. The complexity of our input variables makes this linearity assumption unverifiable, therefore NNR is used to assess the quality of inputs and output in a non-linear way. By using both methods, the user can get a range of results that can be statistically useful in interpreting/predicting COVID-19 factors.

Background

We can concretely define the COVID-19 factor prediction problem by breaking it up into various components. First, we can apply the generic factor prediction problem to our COVID-19 dataset with two subproblems defined below. We also provide data descriptions for our input dataset and our output vector.

Problem 1

Given a set of COVID statistics in an area for each time step t over the range t = 0 to t = n, predict the (number of cases | mortality rate) at time t = n+1.

Problem 2

Given a set of COVID statistics in an area for each time step t over the range t = 0 to t = n, calculate the impact of each COVID statistic with respect to the (number of cases | mortality rate).

Inputs

A list of states and their daily COVID attributes over a time interval. The statistics include the following:

- *Date:* The date in the form of MM/DD/YYYY that the following data corresponds to

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

- *Province_State:* Represents the name of the state/province
- *Confirmed:* Number of confirmed cases in the state/province
- *Deaths:* Number of deaths in the state/province
- *Recovered:* Number of recovered cases in the state/province
- *Active:* The active number of cases (Active cases = total cases - total recovered - total deaths)
- *Incident_Rate:* Cases per 100,000 persons
- *Case_Fatality_Ratio (%):* Number recorded deaths / Number cases.

The programmatic input is in the form of one csv file for each date. The rows correspond to each of the states/provinces. The data is obtained from the following link. Our input is a set of independent variables that would predict a dependent variable. We are designing the program to use any set of independent variables for any dependent variable. Therefore, any group of these variables can be used as the input.

Outputs

A set of feature-weight pairings that can be used to estimate the number of cases on a particular day given a set of feature values. *f(W1*F1 , W2*F2 , … , Wn*Fn) = Vt*

Related Work

We designed the codebase and regression algorithms to be used with any dataset and with mutable input/output variable assignments. Moreover, any dataset can be used, and any columns can be used as the independent variables to predict any other column as the dependent variable. We used data pertaining to COVID-19 because COVID-19 is extremely impactful and relevant at the time of this project's production. Due to the pertinence of the pandemic in everyone's daily life, it is a significant problem that people are hoping to understand and create solutions for.

With this dataset, we explored the option of using classification rather than regression. By assigning a Boolean value to the quantitative values of a target variable and training and testing the data using the Boolean values, a classification approach could have been used. There are several models and algorithms that could be used for the classification approach; however, we determined that predicting a specific value is more relevant than having a Boolean value of a prediction greater than or less than a given numeric.

There are also several different regression algorithms that could have been used. We chose these two specific algorithms because they range well in complexity and style. Linear regression, the most simplistic regression algorithm, models the data in the most simplest of forms. However, neural network regression allows for the data to be interpreted in layers.

Other researches have also explored regression analysis for COVID-19 data. For example, on September 29th 2020, several researchers from the Boston Medical Center released the study, *Using machine learning of clinical data to diagnose COVID-19: a systematic review and meta-analysis.* During this study, COVID-19 patient symptom data was analyzed to create a model to classify COVID-19 patients and influenza patients. This study was exclusively for classification purposes; however, it was very prominent in helping make conclusions about the virus during the early stages. An alternative machine learning classification study was conducted on a dataset that consisted of images of chest x-rays. In *COVID-Classifier: An automated machine learning model to assist in the diagnosis of COVID-19 infection in chest x-ray images*, the researchers used image files to classify patients as either containing the virus, or not containing the virus. This study also used a multilayer neural network with two hidden layers and one output classifier to classify the images. The following studies have shown promising results for classification within the realm of COVID-19.

We chose to implement regression in order to take advantage of the quantitative nature of our dataset. There are also several other regression studies that have been conducted on COVID-19. Generally, regression analysis has been heavily used by the medical community when analyzing historical data to predict the future spread of viral diseases.

Project Description

The first step in implementing our program was accumulating and cleaning daily COVID-19 statistics for each U.S. territory. Our codebase was designed abstractly so any array of independent variables and dependent variables could be used as input.

After cleaning our data, we selected a subset of columns to act as independent variables and a single column to act as our target variable. We formatted these selections into the shapes:

X: [[P1_0, P2_0, …, Pn_0]
[P1_1, P2_0, …, Pn_0]
...
[P1_n, P2_0, …, Pn_0]]
y: [D_0, D_1, … , D_n]

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Each row in *X* is a day of COVID-19 statistics for a U.S. territory, with each *Px* value being a separate independent variable. Each value *D* in *y* is the corresponding target variable's value. A function that maps the inputs in *X* to an output in *y* would take the form *f(P1_0, P2_0, ..., Pn_0) = D_0*

These matrices were then inserted into each respective regression algorithm.

Multivariable Linear Regression

As mentioned in previous sections, we implemented two regression algorithms. The first of which is Multivariable Linear Regression. *Linear regression* is a regression analysis algorithm that interprets an input variable as a linear combination of the target variable. *Multivariable Linear Regression* is similar except instead of accepting one input variable, it can accept any number of input variables. Formally:

> *Given a set of parameters [P1 , ... , Pn] and a target variable [D] that is dependent on those parameters, a linear combination of input parameters [P1 , ... , Pn] can be expressed as:*
> *D = C1*P1 + C2*P2 + ... + Cn*Pn + b*
> *Where b is the bias/intercept of the linear function and [C1 , ... , Cn] are weighted coefficients for the corresponding input parameters.*

To implement this algorithm, we first added a column to the input matrix that kept track of the intercept *b*. Every value in this column was initialized to 1. Then, we found the correct weights vector via the following formula:

> *weights = inverse(X.transpose().dot(X)).dot(X.transpose()).dot(y)*

From this point, given some example input vector (an array of *[P1 , ... , Pn]*), we can cascade our *weights* vector over the input vector, add the intercept value *b*, and return a predicted target value. See section 6. for the full results of this process.

Neural Network

The second algorithm we used is a Neural Network, which is a more complex application of regression that inserts some number of hidden layers between the input vectors and the predicted output. In our case, one hidden layer was used. During the training phase, this hidden layer was used to update the "prediction formula". It does this by running through an activation function which "activates" the update procedure if the activation function returns a value past a certain threshold given the hidden-layer weighted inputs. The activation function used in this project was Sigmoid:

$F(x) = 1 / (1 + e^{x})$

Sigmoid was used because it returns a non-binary value within the range the range (0, 1). This non-binary allowed us to update our weight vectors every training iteration using Sigmoid's derivative (explained below), which slowly increased our weight vector's accuracy on a gradient. This process is known as Gradient Descent. The Sigmoid's return value can also be read as a binary value for classification purposes (using 0.5 as the threshold), but this functionality isn't necessary for predicting quantitative values such as mortality rate.

Due to Sigmoid's return being on a (0,1) scale, we needed to include an extra normalization and standardization step when cleaning our data. First, we scaled each of our training inputs and outputs using sklearn's MinMaxScaler. This function takes in an array of values and scales them to a given range, with the maximum value of the set being assigned to 1 and the minimum value of the set being assigned to 0. This normalization process not only helped with Sigmoid, it also standardized the input variables. For example, if our input parameters were "Number of Cases" and "Hospitalization Rate" (% of people in the hospital due to COVID), these values are hard to compare due to their different metrics. Once weighted, all parameters were on the range (0,1).

To calculate the weight "updates" during each training iteration, we took the dot product of the input values X and an array P where:

> *P = (predicted_values - actual_values) * (predicted_values * (1 - predicted_values))*

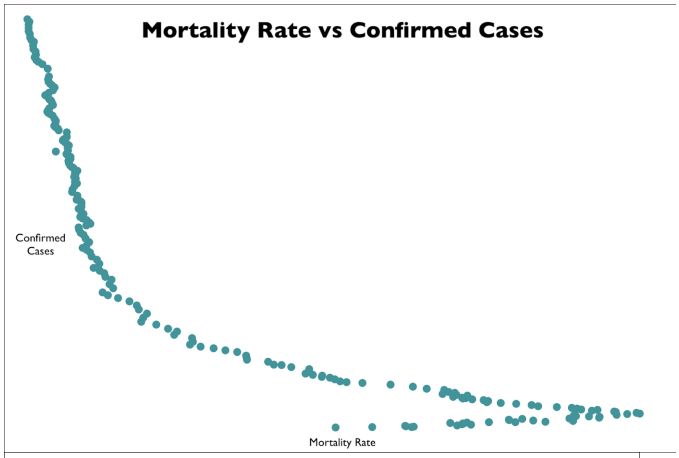
Note: Python's numpy library allows for array operations (-, +, *, /) to take place in one line.

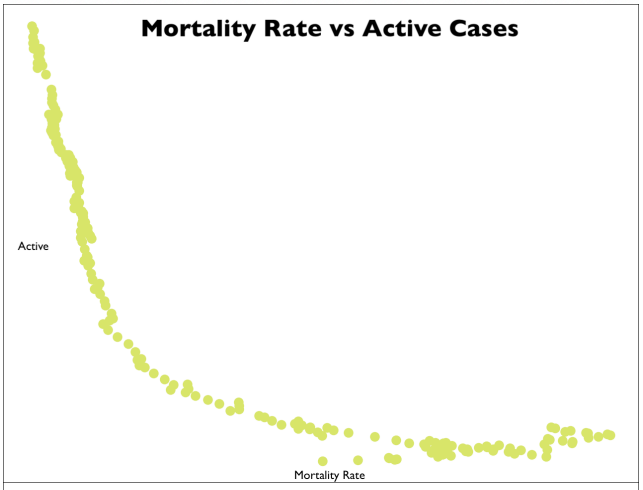
The first component, *(predicted_values - actual_values),* is the difference between the current training iteration's predictions of the target variables and the actual target variable values.
The second component of P is the derivative of Sigmoid.

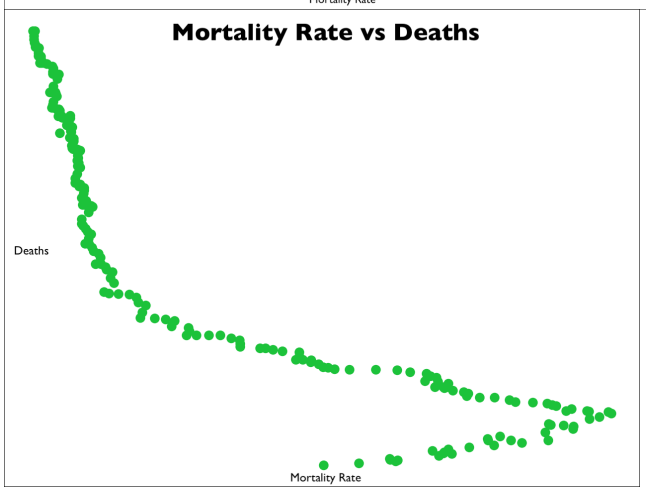
After training our weight vector, we could not just take in a set of variable values and cascade our weight vector like we did in the unscaled Multivariable Linear Regression algorithm. First, we had to scale the input vector by the same variable scales used in training. This puts the new input vector onto the range (0,1), which is then cascaded with the trained weight vector, and finally the output is scaled back into the y vector's original scale. This returned value is our prediction for the target variable D.

Experiments

The first experiment conducted was an analysis of the independent and dependent variables that we will be using for the machine learning predictions. We arbitrarily identified our dependent variable as the mortality rate; however, the program is designed to use any of the variables as the dependent variable. The purpose of the study was to interpret the correlation between each of the independent variables and the dependent variables prior to conducting a regression. Using matplotlib, we graphed the following independent variables against the dependent variable of mortality rate. The following figures were returned:







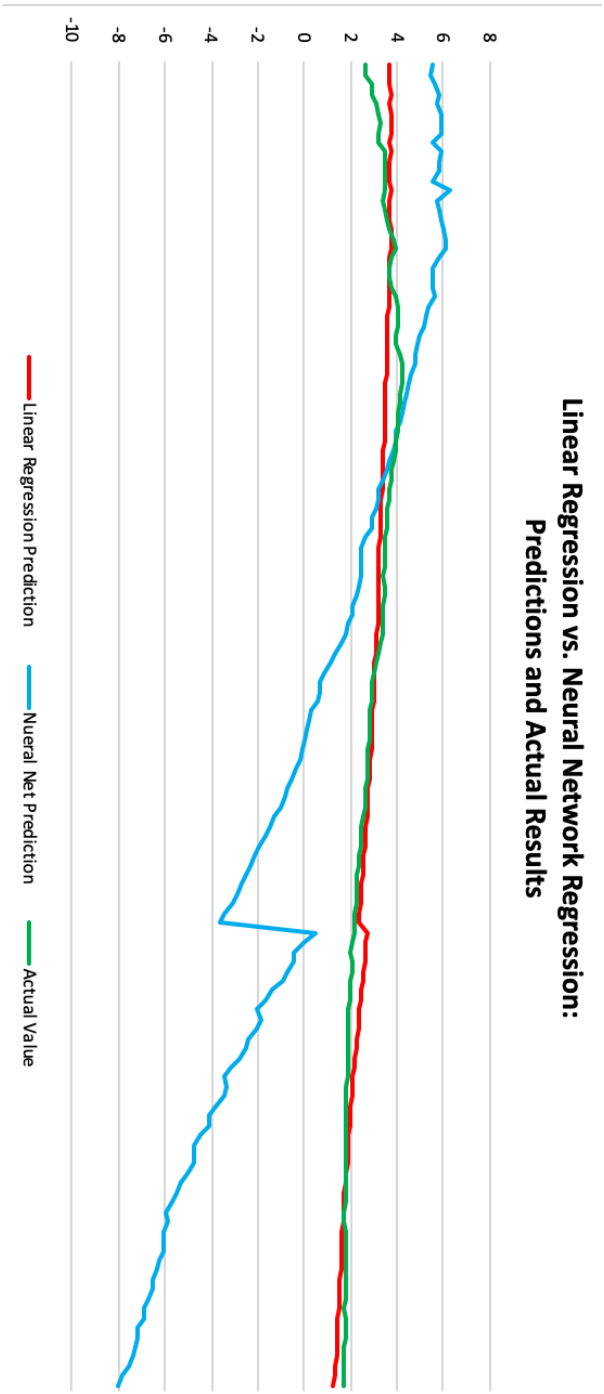


Figure 1. Alabama's predicted Mortality Rates from each regression algorithm after training on Number of Cases and Hospitalization Rates.

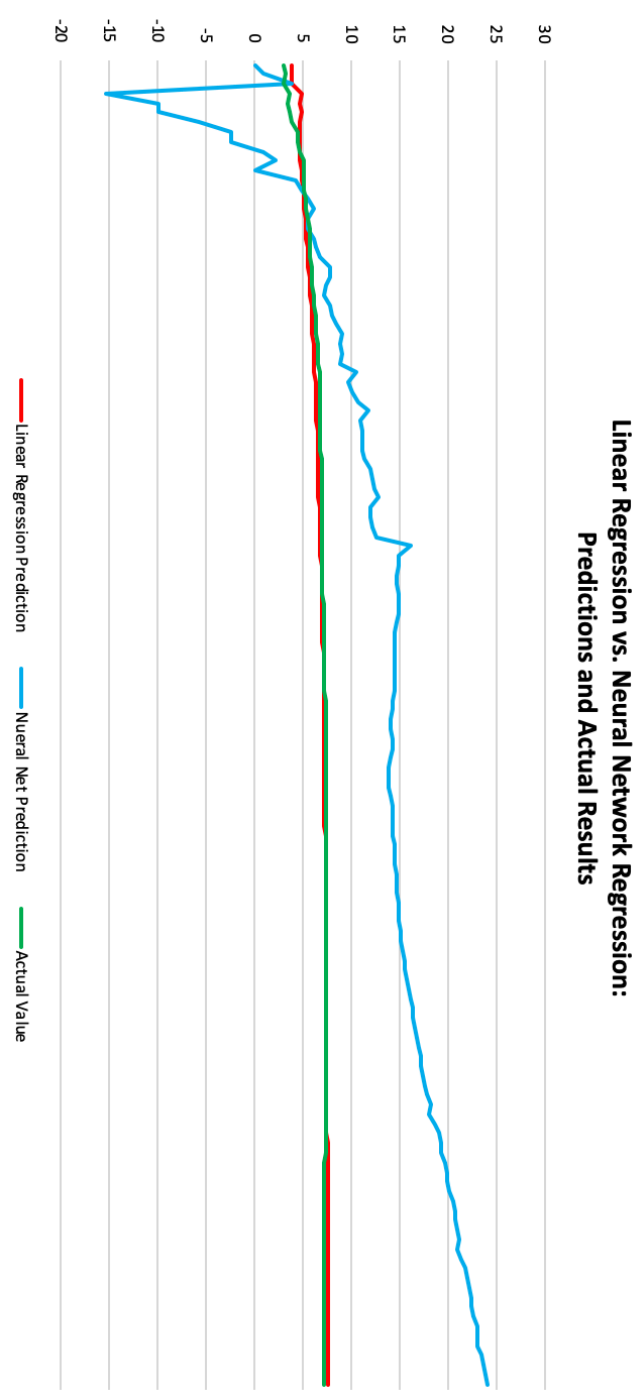


Figure 2. Massachusetts' predicted Mortality Rates from each regression algorithm after training on Number of Cases and Hospitalization Rates.

During experimentation, we wanted to conduct an analysis for which model best represented and predicted this specific data set. To do this, we split the dataset into the corresponding states and territories and ran the regression models for the following dates: 4/12/2020 - 11/08/2020. The independent variables tested were the confirmed case count and hospitalization rate. The dependent variable identified was the mortality rate. The model was trained with a subset of the data and then ran through each of our regression algorithms. After training, we passed every set of independent variables from the training set (re-using them for the test set) into each model. Both models then returned a prediction for the mortality rate for each day on the U.S. territory that we ran the test on. We plotted the regression predictions against the recorded mortality rates for Alabama in Figure 1 and Massachusetts in Figure 2.

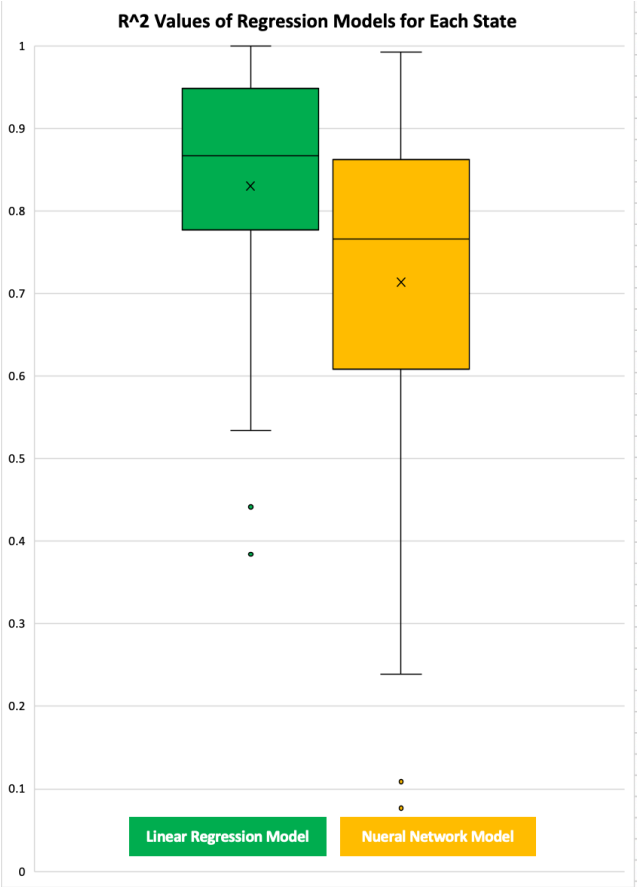


Figure 3. R^2 values for each Regression algorithm's predictions versus the actual Mortality Rate over the time interval 4/12/2020-11/08/2020, each point is the R^2 value for predictions concerning one U.S. territory

To determine the accuracy of each model, we found the R^2 value of the predicted results with respect to the actual results for each Regression algorithm and each state over the time interval (Figure 3). The R^2 value is a good measure for accuracy of the models because it determines the correlation between the expected and predicted values. The R^2 value ranges from 0-1. The closer the value is to 1, the more correlated the predicted results are to the actual results. As shown in the box-and-whisker plot above, the linear regression plot performed better than the neural network model. Some of the outliers are caused by messy data that was received from some of the states. The R^2 formula we used is:

R^2 = 1 – (Sum of squares of residuals) / (Total sum of squares)

Conclusion

Algorithms

The Multivariable Linear Regression Algorithm we implemented returned accurate values based on empirical examples. None of the variable relationships we tested were exactly linear, in fact some were closer to exponential, so the potential for our MLR implementation capped out at a certain level of accuracy. We implemented a third algorithm using Python's sklearn library which takes a similar approach to our MLR implementation. We used this sklearn implementation to test our "homebrewed" results and found that our algorithm's output matched sklearn's output exactly.

Our Neural Network implementation on the other hand struggled to match the level of consistency that our MLR implementation reached, having a larger variance of returns and a smaller R squared. Through testing different variable configurations, we found that the predictions our Neural Network returned were less uniform and we came to expect a higher level of error. This higher error level could be caused in a few places:

1. The use of Sigmoid as an activation function and some in-precision in the scaling mechanism.
2. Not enough training iterations over the dataset.

With that in mind, the R squared value can be increased by plugging in different activation functions (Relu, Linear, Log), allowing the training to run for longer, or adding a more precise bias vector to each layer during training.

COVID-19 Applications

The results we received were astonishing. Throughout this project, we didn't just create a prediction model for a specific set of variables that spits out a prediction of some

other variable, we created a tool that given any set of variables returns a model with which you can predict any other variable in the dataset. The flexibility of our application allows for easy analysis of any COVID-19 factors and how they relate to each other. We focused the graphs included in this report on a specific relationship (namely "number of confirmed cases", "hospitalization rate", and "mortality rate") for congruency, but the code provided allows for any parameters the user wants.

Another analytical strength of this program is the ability to specify which U.S. territories or subset of U.S. territories to calculate variable relationships for. For example, running the regression algorithms on ["Number of Cases", "Hospitalization Rate"] with ["Mortality Rate"] as the target variable for two different states will allow users to directly compare the impact of certain COVID-19 factors across states. This additional level of analysis is extremely useful when analyzing which states were most effective at reducing COVID-19's impact and could be used to compare each state's containment policy's effectiveness when planning for future pandemics.

What We Learned

This project allowed us to explore a plethora of machine learning algorithms, not just the two we ended up using for the final codebase. We both got to participate in high level functional analysis while implementing different iterations of our regression algorithms. Implementing these algorithms at first seemed daunting, but after doing the research and thoroughly discussing the purpose of our project and what we hoped to accomplish we were able to nail down precisely what equations were necessary.

In addition, the depth and scale of what we implemented taught us a lot about project management as well; we had to navigate two fully loaded schedules to create something we are both genuinely proud of. Developer coordination is never easy, but this project gave us countless opportunities to work with each other and grow as professionals.

References

"COVID-19 Map." *Johns Hopkins Coronavirus Resource Center*, coronavirus.jhu.edu/map.html.

Khuzani, Abolfazl Zargari, et al. "COVID-Classifier: An Automated Machine Learning Model to Assist in the Diagnosis of COVID-19 Infection in Chest x-Ray Images." *MedRxiv : the Preprint Server for Health Sciences*, Cold Spring Harbor Laboratory, 18 May 2020, www.ncbi.nlm.nih.gov/pmc/articles/PMC7273278/.

MQ. Zhang, XH. Wang, et al. "Using Machine Learning of Clinical Data to Diagnose COVID-19: a Systematic Review and Meta-Analysis." *BMC Medical Informatics and Decision Making*, BioMed Central, 1 Jan. 1970, bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-01266-z.