# Modelling and Predicting Temporal Spread of COVID-19 in Ontario, Canada

1st Fauzia Jabeen
*Mathematics Department*
*Ryerson University*
Toronto, Canada
fauzia.jabeen@ryerson.ca

***Abstract*—This project aims at predicting the number of cases in the nearest future using various machine learning techniques. The quantitative data is taken from the published data by government of Ontario. This data is used for training, testing and validation purposes. The techniques of linear regression, kernel based non-linear regression and multilayer neural networks were studied and implemented. The project has given an opportunity to learn and implement various machine learning techniques for a real life problem of forecasting COVID-19 cases.**

***Index Terms*—COVID-19, forecasting, prediction, modelling, Ontario COVID-19 data, gradient descent, support vector machine, neural networks, forward- and back-propagation.**

## I. Introduction

### A. Problem Description

With its first case reported in the city of Wuhan, China in December, 2019 [1] , and first death reported on January 10, 2020, **CO**rona **VI**rus **D**isease - 2019 (abbreviated and named as COVID-19, by the World Health Organization (WHO) [2]), became a global pandemic [3] over a span of few months. As of December 15, 2020, John Hopkins University [4] has reported more than 72 million confirmed cases and 1.64 million deaths due to COVID-19. Almost all countries of the world have implemented measures like travel restrictions, social distancing, quarantines, event cancellations, improved testing and lock-downs to stop or slow down the spread of this diseases. While virologists have been working and are introducing vaccines for COVID-19, and medical experts are developing protocols for clinical handling of such cases, data scientists and machine learning experts have been working on developing models to analyse and predict the spread of the disease. The aim of this project is to study various machine learning (ML) techniques and their use in predicting the number of COVID-19 confirmed cases. The study has been focussed on the data obtained for Ontario.

### B. Significance and Challenges

One major challenge is using machine learning (ML) and artificial intelligence (AI) techniques to model COVID-19 cases is the size of the dataset. Machine learning algorithms usually require large datasets for training. The dataset for COVID-19 is less than a year long. There are other challenging factors as well. For example, the virus is new and the knowledge about the parameters required to predict its spread are unknown.

### C. Prior Work

In the last few years, ML techniques have been successfully applied to various predictive tasks including stock exchange values [5], company sales [6], weather prediction [7] and the spread of epidemics [8]. A neural network based ML technique was proposed by Jia et al. [9] for predicting the outbreak of hand-foot-mouth diseases. A number of studies have been made in the recent past to model, analyse and predict the COVID-19 cases. ML algorithms were used by Hamer et al. [10] to predict spatio-temporal epidemic spread of pathological diseases. Fanelli and Piazza [12] have performed forcasting and analysis of COVID-19 in Italy, France and China, based on ventilation systems. A regression based model was developed by Li et al. [11] to determine the exponential growth of COVID-19 cases. The number of such studies has been on the increase ever since the onset of the pandemic.

### D. Report Overview

The rest of the report is organised as: Section II gives details about the problem statement, data gathering, data cleaning and preparation for the ML algorithms. In Section III, the description of various models and methods used in this project is given. Results of the applications of ML techniques along with observations and discussion are given in Section IV. The use of various python packages, pieces of code from GitHub and other sources in the implementation of the solution is given in Section V.

## II. Problem Statement and Dataset

### A. Problem Statement

In this project, I intend to perform a comparative study of various machine learning techniques for the prediction / analysis of COVID-19 positive cases.

### B. Data Source

I have used the data gathered by the Ontario Government and made available for public use on their website *Status of COVID-19 Cases in Ontario*. The link to the website in given in [13]. The first confirmed case of COVID-19 in Ontario was reported on January 26, 2020. I have used the data from this date to December 13, 2020.

### C. Data Cleaning / Preparation

Initially confirmed cases were not reported every day. Therefore, there are no entries on some dates. Also on some days testing was not carried out and there are missing entries for those days. Also the original raw data includes information other than the number of confirmed cases. To prepare the dataset for this project, I wrote down a python function to clean the data from missing entries and to remove the irrelevant information. Further, the data is in the form of a time series of the form $x_0, x_1, \cdots, x_i, \cdots, x_N$ , where $x_i$ is the number of confirmed cases detected on dates indexed as $i = 0, 1, \cdots, n$. If we assume that the series is such that the next value in the series depends only on the just previous value (a Markov chain) then the prediction of $x_{i+1}$ depends only on the value of $x_i$. Therefore, the unknown function to be learnt is $x_{i+1} = f(x_i)$. Thus, our data can be set up as pairs of values $(x_1, x_0), (x_2, x_1), \cdots, (x_N - 1, x_N)$. What we intend to see that the next value for the confirmed cases could be predicted on the basis of the values obtained for last few days. For example, if we want to predict the number of confirmed cases based on the last three values, the regression function takes the form $x_i = f(x_{i-1}, x_{i-2}, x_{i-3})$. The resulting dataset is shown in Table (I).

The size of the input vector (also called step-size) is 3 in this case. In this project, we have experimented with step-sizes of 3, 5 and 7. The resulting data is then split into training, testing and validation datasets. For every ten days, 7 values are used for training, 2 for testing and one for validation purposes, all picked at random. The python function written performs all of the above data cleaning and preparation steps to produce the training, testing and validation datasets.

TABLE I
Model of the output (prediction) depending on the last three day's data

| Day Index | Inputs | | | Output |
|---|---|---|---|---|
| 0 | $x_0$ | $x_1$ | $x_2$ | $x_3$ |
| 1 | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
| ... | ... | ... | ... | ... |
| $N-3$ | $x_{N-3}$ | $x_{N-2}$ | $x_{N-1}$ | $x_N$ |

## III. Methods and Models

I have used three possible ways in which this problem can be modelled: (i) Linear Regression (ii) Support Vector Regression (SVR) and (iii) Neural Networks.

### A. Linear Regression

Based on the material studies in the class, the hypothesis function $h(x)$ for 3-step data can be written in intercept form as,

$$\begin{aligned} h(x) &= \theta_0 + x_1\theta_1 + x_2\theta_2 + x_3\theta_3 \qquad (1) \\ &= \begin{bmatrix} 1 & x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \\ &= \sum_{j=0}^{3} x_j \theta_j \\ &= \mathbf{x^T}\boldsymbol{\theta} \qquad (2) \end{aligned}$$

Now, with $n = N - 3$, the input matrix $\hat{\mathbf{X}}$ can be written as

$$\hat{\mathbf{X}} = \begin{bmatrix} 1 & x_1^{(0)} & x_2^{(0)} & x_3^{(0)} \\ 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(2)} & x_2^{(n)} & x_3^{(n)} \end{bmatrix} = \begin{bmatrix} -\left(x^{(0)}\right)^T - \\ -\left(x^{(1)}\right)^T - \\ \vdots \\ -\left(x^{(n)}\right)^T - \end{bmatrix} \qquad (3)$$

Also the target vector becomes

$$\mathbf{y} = \left[y^{(0)} y^{(1)} \cdots y^{(n)}\right]^{\mathbf{T}} \qquad (4)$$

I have considered mean squared error as the objective (cost/loss) function $J(\theta)$ and can be written as

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left( h(\hat{x}^{(i)}) - y^{(i)} \right)^2 \qquad (5)$$

In vector-matrix form, it can be written as,

$$J(\theta) \;=\; \frac{1}{2}\left(\hat{\mathbf{X}}\boldsymbol{\theta}-\mathbf{y}\right)^T\left(\hat{\mathbf{X}}\boldsymbol{\theta}-\mathbf{y}\right) \tag{6}$$

In regression techniques, a hypothesis function that minimizes the cost (loss) function is determined. Differentiating with respect to $\boldsymbol{\theta}$, the gradient of the cost function is given by:

$$\nabla_\theta J(\theta) \;=\; \hat{\mathbf{X}}^{\mathbf{T}}\left(\hat{\mathbf{X}}\boldsymbol{\theta}-\mathbf{y}\right) \tag{7}$$

In gradient descent approaches, starting with some initial values, the vector $\boldsymbol{\theta}$, is updated by some learning rate ($\alpha$) times the negative of the gradient vector $\nabla_\theta J(\theta)$, i.e.,

$$\begin{aligned}\boldsymbol{\theta} \;&:=\; \boldsymbol{\theta}-\alpha\nabla_\theta J(\theta) \\ &=\; \boldsymbol{\theta}+\alpha\;\hat{\mathbf{X}}^{\mathbf{T}}\left(\mathbf{y}-\hat{\mathbf{X}}\boldsymbol{\theta}\right)\end{aligned} \tag{8}$$

In the method **Gradient Descent (SGD)**, the gradient vector is approximated taking into account all (the whole batch) of the training input vectors, and the updated $\boldsymbol{\theta}$ is given by the above equation. The method is used iteratively to find out the converged set of values for $\boldsymbol{\theta}$. In contract, in the **Stochastic Gradient Descent (SGD)**, the above equation is used for each input vector turn by turn to get the converged set of values for $\boldsymbol{\theta}$.

Another way of solving the linear regression problem is to use the **normal equations**. These equations are obtained by setting the above gradient equal to zero, leading to a system of equations of the form:

$$\mathbf{A}\boldsymbol{\theta}=\mathbf{b} \tag{9}$$

where,

$$\begin{aligned}\mathbf{A} \;&=\; \hat{\mathbf{X}}^{\mathbf{T}}\hat{\mathbf{X}} \\ \mathbf{b} \;&=\; \hat{\mathbf{X}}^{\mathbf{T}}\mathbf{y}\end{aligned} \tag{10}$$

The above equation can be solved using python's NUMPY `na.linalg.solve(A,b)` solver. I have implemented linear regression with **normal equations** using the feature-map of degree-3 polynomial (using the starter code provided). The scatter plot of the training data and the plot of the learnt hypothesis as smooth curve are shown in Figure

### B. Support Vector Regression - SVR

Support Vector Regression (SVR) is an extension of the idea of Support Vector Machine (SVM). For linear classification problems, an SVM finds out that line (decision boundary) that has the greatest distance to the points closest to it. The closest points that identify this line are known as *support vectors* and the region between these points around the decision boundary is known as the *margin*. A parameter (usually labelled as 'C') determines the margin (the margin decreases as C is increased). For non-linear separable data, the data is first transformed in a way so that a hyperplane could be determined as a decision boundary. This transformation is accomplished via the use of kernel functions. Some of the popular kernels are: Radial Basis Function (RBF), Polynomial and Sigmoid. The kernel functions give a non-linear capability to SVM.

SVR can be considered as an adapted version of SVM where the dependent variable is numerical rather than categorical (binary) in nature. SVR is a non-parametric technique, meaning that it does not depend on the Markov assumption and depends on the kernel functions. This in turn, means that we can use the (date-number of confirmed cases) data as is, without converting it to multi-step time series.

### C. Neural Networks - NN

Neural networks are the backbone of various state-of-the-art deep learning algorithms today. I have used a multi-layered neural network with mini-batch back-propagation algorithm is this project. A typical neural network (based on 3-step training/test data) is shown in Figure(1). A mean squared error function has been used as the cost (loss) function. Various activation functions (Sigmoid, Relu, linear etc.) can be used at the hidden and output layers of the network.

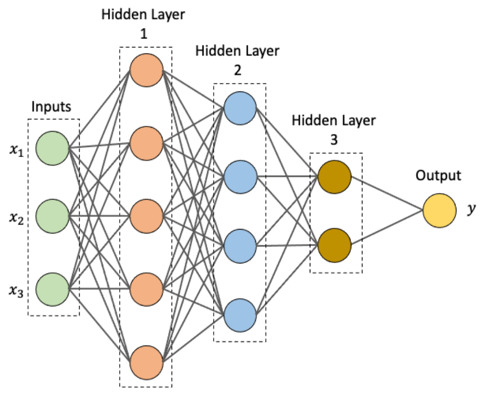

Fig. 1. Architecture of a typical 3-step neural network.

## IV. Results and Discussion

### A. Linear Regression

For this part of the project the python code developed during the execution of the course were modified and extended to take into account the time series dataset created in this project. The number of unknowns ($\theta$'s) for the three cases were 4, 6 and 8. For the GD and SGD cases, we used a learning parameter $\alpha = 2 \times 10^{-14}$ after experimenting and gradually decreasing its value starting from $0.01$. It was found that $10,000$ iterations were required for both GD and SGD for the training data to match with predicted results. The predicted number of confirmed cases for the test data for 3, 5 and 7 previous day's numbers were determined and compared with the original numbers. These results compare very well with those obtained from applying the stochastic gradient descent regression and are shown in Figures (2), (3) and (4). It was found that by solving the normal equations, the predicted results almost have an exact match with the original test data as shown in Figures (2), (3) and (4).

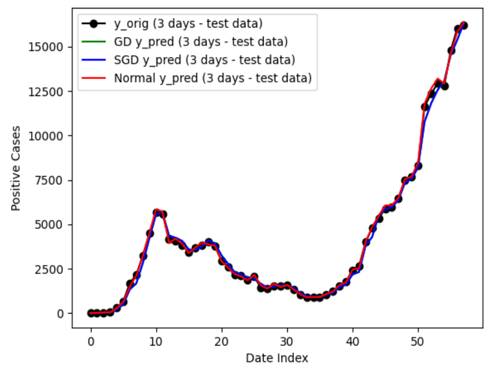

Fig. 2. Prediction of confirmed cases for test data using the GD, SGD and Normal equations trained on the training data.

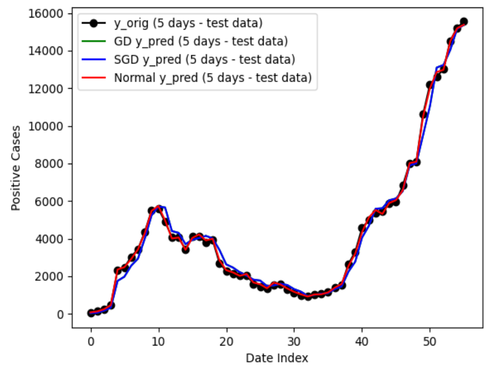

Fig. 3. Prediction of confirmed cases for test data using the GD, SGD and Normal equations trained on the training data.

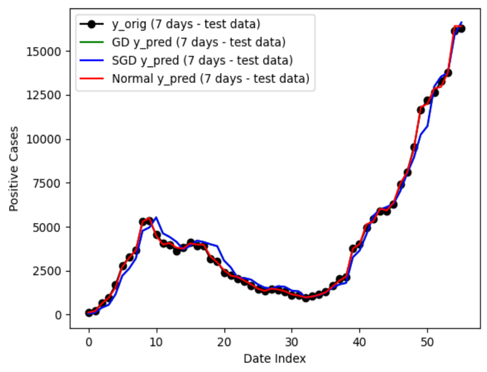

Fig. 4. Prediction of confirmed cases for test data using the GD, SGD and Normal equations trained on the training data.

## B. Support Vector Regression

In case of support vector regression, the original time series data was used for modelling. A radial basis function (RBF) kernel was used for training purposes with $C = 1 \times 10^4$ and `gamma = 0.1`. The SVR was trained using the training data and the results of the prediction for the test data are shown in Figure (5). The performance of the trained SVR for the test data is very good for majority of data points except at the ends.

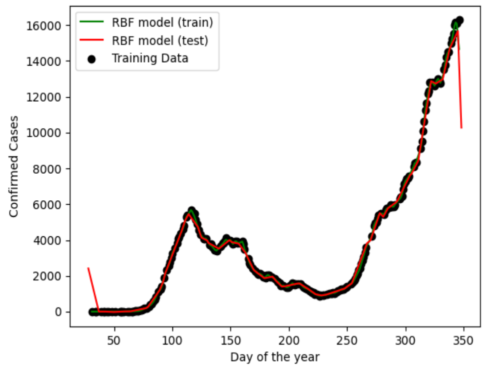

Fig. 5. Prediction of confirmed cases for test data using RBF kernel based SVR trained on the training data.

## C. Neural Networks

After experimenting with several neural network architectures (by gradually increasing the number of layers and number of neurons in each layer) and various combinations of activation functions, I selected a neural network model that was giving good convergence of the mean square error and good results. The neural network thus designed has one input layer, two hidden layers (with 32 and 8 neurons) and an output layer with one neuron. It was found that `sigmoid` activation functions for both the hidden layers were giving smooth convergence in the mean square error. A linear activation function was used for the output neuron. The NN was trained with a learning rate of 0.0005 using gradient descent to minimize the mean square error cost function. Three neural networks with different number of inputs (depending on the three datasets for previous days counts) with the two hidden layers of 32 and 8 neurons and one output neuron were used in the experiments. The number of epoch for each experiment was 1000. The mean square errors attained for the training and testing data are shown in Table (II). It can be noticed that the first neural network for 3-point datasets, gives the least mean square errors for both the training and testing data. The other two networks also give good results, but can be further fine tuned. The original and predicted results are shown in Figures (7), (9) and (11), while the convergence of mean square errors are plotted against epoch numbers in Figures (6), (8) and (10).

TABLE II
MEAN SQUARE ERROR (MSE) FOR TRAINING AND TESTING DATA FOR THE LAST 3, 5 AND 7 DAY'S DATA

| Dataset | Mean Squared Error | | |
|---|---|---|---|
| | 3-point | 5-point | 7-point |
| Training Dataset | 0.0003135 | 0.0004869 | 0.0008302 |
| Testing Dataset | 0.0003181 | 0.0005716 | 0.0010293 |

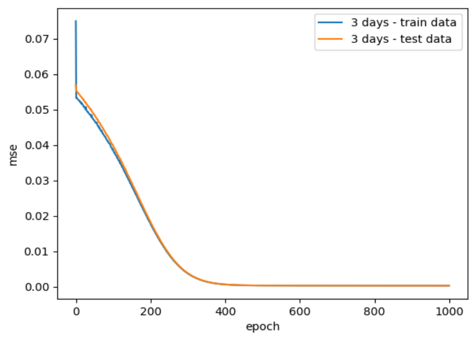

Fig. 6. Mean squared error (MSE) as the neural network learns the training data (3-previous day case) over 1000 epochs.

## V. Implementation and Code

Apart from developing my own code in python, I studied implementation of various python packages and their use from a number of Github pages [14], blogs [15] [16] and research papers. I used `pandas` [17] along with `numpy` [18] in the `dataGeneration` function of my project. For SVR, I used `sklearn` [19] python package. For implementing neural network regression, I used the python package `keras` [20]. Some parts of the code presented on Github pages [14] was used in this project

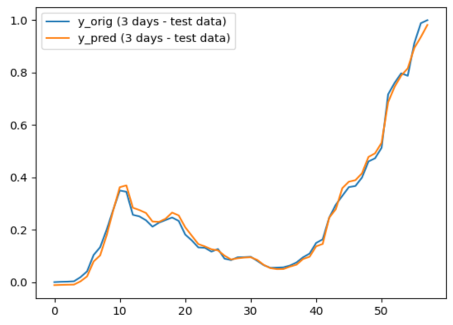

Fig. 7. Prediction of confirmed cases for test data using the neural network trained on the training data (3-previous day case).

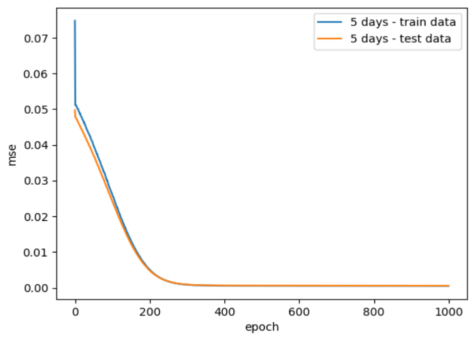

Fig. 8. Mean squared error (MSE) as the neural network learns the training data (5-previous day case) over 1000 epochs.

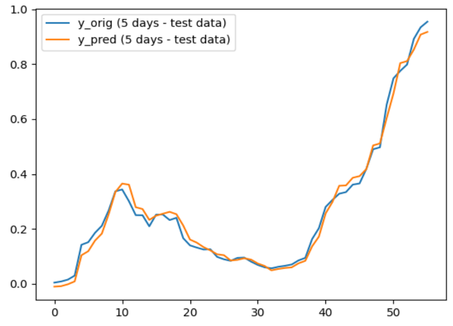

Fig. 9. Prediction of confirmed cases for test data using the neural network trained on the training data (5-previous day case).

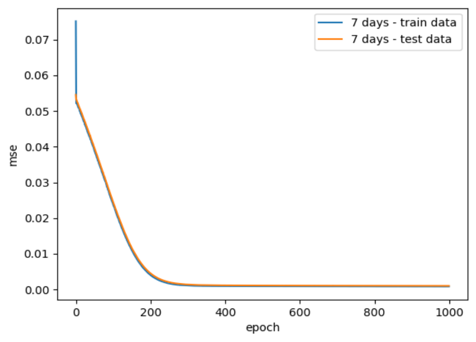

Fig. 10. Mean squared error (MSE) as the neural network learns the training data (7-previous day case) over 1000 epochs.

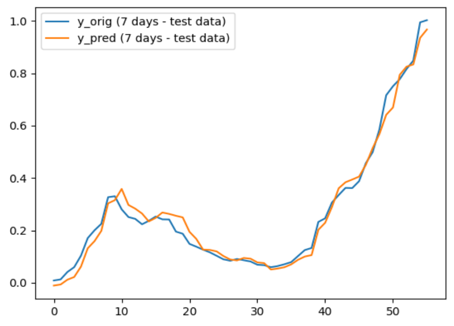

Fig. 11. Prediction of confirmed cases for test data using the neural network trained on the training data (7-previous day case).

with modifications according to the requirements of the project.

## References

[1] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao, Y. Hu, "Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China," *Lancet*, vol. 295(10223), pp. 497–506, 2020.

[2] W.H.O, "Naming the coronavirus disease (COVID-19) and the virus that causes it," [Online] Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it [Accessed: 13- Dec-2020].

[3] Catrin Sohrabi, Zaid Alsafi, Niamh O'Neill, Mehdi Khan, Ahmed Kerwan, Ahmed Al-Jabir, Christos Iosifidis, Riaz Agha, "World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19)", *International Journal of Surgery*, vol. 76, pp. 71–76, 2020.

[4] Johns Hopkins coronavirus resource center [Online]. Available: https://coronavirus.jhu.edu [Accessed: 13- Dec- 2020].

[5] R. Akita, A. Yoshihara, T. Matsubara, K. Uehara, "Deep learning for stock prediction using numerical and textual information". *In Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Okayama, Japan, 26–29 June 2016; pp. 1–6.

[6] M. Ali, Y. Lee, "CRM Sales Prediction Using Continuous Time-Evolving Classification." *In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,* New Orleans, LA, USA, 2–7 February 2018.

[7] F. Xiao, Q. Ai, "Data-driven multi-hidden Markov model-based power quality disturbance prediction that incorporates weather conditions." *IEEE Trans. Power Syst.* vol. 34, pp. 402–412, 2018.

[8] Y. Lu, S. Wang, J. Wang, G. Zhou, Q. Zhang, X. Zhou, B. Niu, Q. Chen, K. C. Chou, "An epidemic avian influenza prediction model based on google trends." *Lett. Org. Chem.* vol. 16, pp. 303–310, 2019.

[9] W. Jia, X. Li, K. Tan, G. Xie, "Predicting the outbreak of the hand-foot-mouth diseases in China using recurrent neural network." *In Proceedings of the 2019 IEEE International Conference on Healthcare Informatics (ICHI),* Xi'an, China, 10–13 June 2019; pp. 1–4.

[10] W. B. Hamer, T. Birr, K. A. Verreet, R. Duttmann, H. Klink, "Spatio-Temporal Prediction of the Epidemic Spread of Dangerous Pathogens Using Machine Learning Methods." *ISPRS Int. J. Geo-Inf.* vol. 9, no. 44., 2020.

[11] Q. Li, W. Feng, Y-H Quan, "Trend and forecasting of the COVID-10 outbreak in CHina." *J. Infection*, vol. 80(4), pp. 469–496, 2020.

[12] D. Fanelli, F. Piazza, "Analysis and forecast of COVID-19 spreading in China, Italy and France." *Chaos Solitons Fractals,* vol. 134(109761), 2020.

[13] Status of COVID-19 cases in Ontario. [Online]. Available: https://data.ontario.ca/dataset/status-of-covid-19-cases-in-ontario [Accessed: 13-Dec-2020]

[14] Source code: Keras loss function, metrics and learning curves. [Online] Available: https://github.com/keras-team/keras/issues/7947#issuecomment-413369777 [Accessed: 12-Dec-2020]

[15] A. Ghose, "'Support vector machine (SVM) tutorial." [Online] Available: https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93 [Accessed: 29-Nov-2020]

[16] J. Bownlee, "Regression tutorial with the Keras deep learning library in python", Available: https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/ [Accessed: 29-Nov-2020]

[17] Pandas: An open source data analysis and manipulation tool. https://pandas.pydata.org

[18] NumPy: The fundamental package for scientific computing with Python. https://numpy.org

[19] scikit-learn: Simple, efficient and open source tools for predictive data analysis and machine learning in Python. https://scikit-learn.org/stable/

[20] Keras: A deep learning API written in Python, running on top of the machine learning platform, Tensorflow. https://keras.io