Utilitizing Neural Architecture Search and AutoKeras in Forecasting COVID-19 Cases in the National Capital Region of the Philippines using Incidence and Risk Factors in a Time Series

A THESIS PRESENTED TO THE
DEPARTMENT OF INFORMATION SYSTEMS
AND COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

BY

KEITH ADRIAN SANTOS

RYAN CARLO BAUTISTA

ANTON MIGUEL CASTILLO

QUEZON CITY, PHILIPPINES

2022

ABSTRACT

The daily number of COVID-19 cases in the National Capital Region (NCR) of the Philippines have continued to rise amidst existing restrictions and safety measures. Researchers have utilized machine learning to forecast these cases to provide more effective measures in reducing the spread of COVID-19. However, their studies primarily focused on the infected cases over a duration of time without including other potential risk factors in the spread of the virus.

This study attempts to utilize Neural Architecture Search (NAS) through AutoKeras to create a neural network model capable of forecasting the number of confirmed COVID-19 cases in NCR two weeks, one month, and two months in advance through the use of risk factors. It's performance was compared against Random Forest (RF) Regression models.

Two types of models were created: a base model, incorporating only COVID case counts, and a risk model, incorporating both COVID case counts and risk factors. The risk factors chosen were susceptibility rates, incidence rates, death rates, recovery rates, quarantine data, community mobility rates, and hospital equipment amounts. Autokeras was configured to test 10 Keras architectures and each were trained over 10 epochs.

It was found that both the base and risk RF models performed better than the NAS models for the 14-day forecasts. However, the risk NAS models performed significantly better than both base and risk RF models for the 30-day and 60-day forecasts, both in accuracy metrics and in following the trend of the actual cases. The results also indicate that integrating multivariable inputs in the training of forecasting neural network models could help improve model performance and provide more accurate forecasts of COVID-19 spread in the future

ACKNOWLEDGMENTS

The authors would like to thank Mr. Jann Railey E. Montalan for their continuous support, guidance, and advisement throughout the research. The authors would also like to thank Dr. John Noel P. Victorino and Dr. John Paul C. Vergara for their feedback and suggestions with regards to the improvement of the research.

The authors would also like to thank the Department of Information Systems and Computer Science (DISCS) and the Ateneo Social Computing and Science Laboratory (ASCSL) of the Ateneo de Manila University (ADMU).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

In December of 2019, the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), or what is now known as the Coronavirus disease (COVID-19), was first discovered in Wuhan, China [26, 23]. As stated by the World Health Organization (WHO), the virus is transmitted from person to person primarily through respiratory droplets and aerosols [25]. The World Health Organization declared it a pandemic on 11 March 2020 [26].

Following the outbreak, the need to monitor and predict COVID-19 was evident. Countries in Europe, Asia, and the American continents have employed the use of mathematical and statistical models to forecast COVID-19 cases, such as the Susceptible-Exposed-Infectious-Recovered (SEIR) [14] and Autoregressive Integrated Moving Average (ARIMA) models [13].

Many previous studies have explored alternate forecasting models to better predict and model the spread of the COVID-19 disease. A study conducted in China utilized machine learning to produce a model that could reliably forecast COVID-19 activity. The proposed model was able to predict two (2) days ahead and outperformed the persistence model in 27 out of 32 provinces [22].

The University of the Philippines utilized the Susceptible-Infectious--Recovered (SIR) model in forecasting COVID-19 during the initial months after the outbreak. The model's predictions were raised to the Philippine government in order to develop a plan to reduce the virus' transmission rate. As a result, the implementation of an Enhanced Community Quarantine (ECQ) had generated positive results in reducing the spread of COVID-19 in the Philippines [8]. However, the Department of Health's (DOH) COVID Tracker shows that the rate of infections still continues to rise despite the preventative measures taken.

Another method used was a recurrent neural network, utilizing a Nonlinear AutoRegressive model with eXogenous inputs (NARX) using case data, airline history, socioeconomic factors, and calculated risk variables, to forecast high or low-risk regions of United States territories [2].

After examining all the aforementioned models, the researchers would like to utilize an advanced technique such as Autokeras' Neural Architecture Search (NAS) to be able to determine the best performing model without needing to implement a specific architecture such as NARX. Because NAS is unhindered by the constraints that limit specific architectures, it can look through a number of neural networks to identify the best possible model for the data given.

Therefore, this study proposes utilizing NAS to model and forecast COVID-19 cases in the Philippines' National Capital Region (NCR) by using case data, incidence rate, susceptibility rate, and other relevant risk factors. The neural network model is expected to output a forecast of positive COVID-19 cases in NCR and its performance will be measured against a standard machine learning regression model. The proposed model would be able to assist the local government units (LGU's) in understanding the risk factors associated with the spread of COVID-19.

1.1 Research Questions

The overarching question this study aims to answer is: is it possible to utilize neural architecture search to find a neural network architecture utilizing various COVID-19 infection risk factors and have it perform better than classical modelling approaches?

Specifically, it seeks to answer the following:

1. How can data relevant to risk calculations be collected, feature engineered, and integrated into the model for forecasting COVID-19 cases in NCR?

2. How can the spread of COVID-19 in NCR be forecast using available risk-factor data and current COVID-19 case data?

3. How does the performance, in terms of accuracy of forecasts, of the Neural Architecture Search model incorporating calculated travel risk variables compare to classical modeling approaches?

1.2 Objectives of the Study

This study aims to model the spread of the COVID-19 disease in NCR using neural architecture search and infection risk factors.

More specifically, it aims to do the following:

1. Collect and feature engineer data on the various risk factors associated with COVID-19 into the model to accurately forecast COVID-19 cases in NCR.

2. Identify if the available data for COVID-19 cases and available risk-factors are sufficient to forecast the spread of COVID-19 in NCR.

3. Identify if the model architecture designed by neural architecture search produces more accurate forecasts when compared to classical modeling approaches used in the forecasting of COVID-19 cases in NCR given the same data.

1.3 Scope and Limitations

This study focuses exclusively on COVID-19 data in the Philippines, particularly from the Philippine Department of Health's (DOH) COVID-19 dataset. The DOH's data is only limited to confirmed COVID-positive cases in hospitals and clinics. Thus the data may not be fully representative of all COVID cases that remain unreported. Another drawback of the DOH dataset is that many COVID-positive patients are lacking location data. Moreover, this study assumes that the local COVID-19 data, is reliable and obtained from verified sources and credible organizations.

Furthermore, the model only considered the spread of COVID-19 in NCR given the priority of testing different parameters to find the best possible forecasting model rather than focusing on multiple subsequent models due to time constraints. The model forecasts have been limited to 14, 30 and 60 days into the future.

1.4 Significance of the Study

This study contributes to a better understanding of the possible tools for use in forecasting COVID-19 cases in the Philippines. The Department of Health provides the most up-to-date data on individuals infected with COVID-19. This study aims to make use of the DOH data for the earlier detection of COVID-19 in the Philippine context. The neural architecture search tool, as well as the use of risk factors in forecasting, could assist the Local Government Units (LGUs) and the Inter-Agency Task Force on Emerging Infectious Diseases (IATF) in better forecasting the spread of COVID-19. This, in turn, may allow the government to take better precautions to reduce the COVID-19 infection rate in the Philippines.

CHAPTER II

REVIEW OF RELATED LITERATURE

This research is concerned with two major components, machine learning in disease modelling and COVID risk factors. These will better help in understanding how to model COVID-19 cases in Luzon.

2.1 Field of Epidemiology

Epidemiology is defined as the study of the distribution and determinants of disease [24]. The field of epidemiology is useful in bringing to light diseases and health events and their underlying factors [10]. These factors are not only scientific but also political and social [18]. There is a rising importance in understanding how a virus is spreading and what kind of countermeasures can be put in place to further prevent its spread.

2.1.1 COVID-19

COVID-19 is an infectious disease that affects the respiratory system, making it difficult for a person to breathe [25]. It is primarily transmitted through droplets of saliva or discharge from the nose when an infected person coughs or sneezes [25]. The World Health Organization named the virus COVID-19 on February

11, 2020 and they characterized it as a pandemic on March 11, 2020 [26].

In almost two years of the virus' progression in the Philippines, the frequency of cases had risen, slowed down, surged, and slowed down again. The government had been imposing various preventive measures in the form of community quarantines and lockdowns in order to lessen the spread of the virus. The movements of people and operations of businesses had been either greatly or minimally affected based on what type of lockdown was present in an area [3]. The differences between the quarantines are presented in Table 2.1 based on the Inter-Agency Task Force for Emerging Infectious Diseases' (IATF) given descriptions [19].

The country was placed under an Enhanced Community Quarantine (ECQ) during the first few months of the pandemic, but the Greater Metro Manila (GMM) area eased into a lighter quarantine, known as the Modified Enhanced Community Quarantine (MECQ), in June 2020. It was only during the transition to a General Community Quarantine (GCQ) in July 2020 that the cases rose in highly populated areas. By August 2020, majority of the country was placed in a more relaxed Modified General Community Quarantine (MGCQ) except the GMM area, which had reverted back to MECQ. Afterwards, most areas remained in GCQ throughout the next two years. Even when vaccinations started rolling-out in March 2021, the cases continued to rise at an exponential

Quarantine Type	Description
Enhanced Community Quarantine (**ECQ**)	**Stringent** limitations on movement and transportation of people, **strict** regulation of operating industries, provision of food and essential services, and heightened presence of uniformed personnel to enforce community quarantine protocols.
Modified Enhanced Community Quarantine (**MECQ**)	The transition phase between ECQ to GCQ, when the ECQ temporary measures are relaxed and become less necessary.
General Community Quarantine (**GCQ**)	Limitations on movement and transportation, regulation of operating industries, and presence of uniformed personnel to enforce community quarantine protocols.
Modified General Community Quarantine (**MGCQ**)	The transition phase between GCQ to the New Normal, when the GCQ temporary measures are relaxed and become less necessary.

Table 2.1. Quarantine Types in the Philippines

rate, spiking in April 2021 and reaching the highest number of recorded cases in September 2021. In November 2021, however, the number of cases have strongly declined, most likely due to the quantity of vaccines that have been administered [3].

2.1.2 Possible Risk Factors for COVID-19 Infectivity

The researchers gathered different socioeconomic risk factors that could potentially influence the rate of infection for COVID-19. These factors can narrow down the necessary features for building the desired machine learning model. The factors that will be used in this study are the number of COVID-19 cases per province [21], the number of hospital beds and medical staff per province [21], community mobility [12], population size [5], incidence rates [4], and lockdown restrictions [3].

2.1.2.1 Community Mobility Reports

Human mobility has been used in various epidemiological studies to estimate the potential number of infections in different locations by noting the interactions between certain areas. This was performed using ideal flow of transportation networks to model the transmission patterns of measles in Western Visayas during the 2019 Philippine measles outbreak. All the simple paths between the 133 municipalities were connected in a network and an ideal flow network (IFN) model was used to resemble the movements of actual commuters [17]. The results indicated that the IFN model projections closely resembled the historical measles incidence compared to other methods used. However, it was also mentioned that the assumption of homogeneous mixing of populations between municipalities may not be the most realistic, as individuals are heterogeneously distributed in space and their unique movement patterns are better representative of the inter-neighborhood interactions [17].

Bloomberg has also been releasing monthly rankings of 53 economies based on their success at containing the virus with the least social and economic disruptions. Their COVID Resiliency Rating scores are calculate using 12 data indicators, equally weighted [7]. One of their notable indicators are the Google COVID-19 Community Mobility Reports, which chart movement trends over time using data from users who turn on their location history. With the data protected and anonymized, Google is able to determine the change of mobility in certain certain areas (e.g. residential, transit, groceries) compared to a baseline of January 2020 to February 2020, a period before the pandemic [12]. The data could be used to correlate the movement of individuals during certain time periods such as spikes in cases, a potential risk factor that could be beneficial in understanding human movement patterns in the Philippine context.

2.2 Disease Modeling

A useful tool in the field of epidemiology are mathematical models. A mathematical model is a conceptual model that uses mathematical concepts to better understand or predict how a system of objects will act [16]. In the context of epidemiology, these models are used to predict epidemic dynamics in populations using relevant epidemiological factors and human behavioral patterns. Models can also be used to infer large-scale infection patterns using smaller-scale sample patterns [16].

A good model requires a balance between three essential elements: accuracy, transparency, and flexibility [16]. Accuracy is exhibited in the model's reliability in predicting future outcomes and reproducing observed data. This aspect is vital in finding insights; however, improving accuracy requires more complex models and specific details to be included at the cost of computing power and simplicity. That is where transparency lies, in the capability to understand how a model's components influence one another. Finding the simplest model implementation without sacrificing relevant components then becomes the challenge. Finally, flexibility is the measure of how adaptive the model is to change, which is vital in the context of epidemics [16].

For the purpose of this study, the mathematical models being used are machine learning models.

2.2.1 Machine Learning

Machine learning is a subset of AI that focuses on systems that are able to emulate human learning to solve problems as opposed to explicit programming [9]. Its methods have been widely used in modelling and understanding human diseases. These techniques have been utilized in several different fields of study, including epidemiology [24, 9]. In the modeling and forecasting of diseases, machine learning methods are often used together with traditional biostatistical approaches [24].

2.2.1.1 Machine Learning Models for Disease Modelling

A previously conducted study designed several Multilayer Perceptrons (MLP), a type of Neural Network, to create a regression model of the global spread of the COVID-19 infection. The main goal of the model was to observe the global infection data as a whole, as opposed to separate areas or localities. A Multilayer perceptron is a neural network architecture made up of at least three layers, input, output and hidden. Each layer has a set of neurons, with the output layer having one, the input having the same number as the number of data sets, and the hidden layer having an arbitrary number. The study utilized MLPs due to their ease of implementation while simultaneously providing accurate models [6].

In order to determine the proper hyperparameter values for the MLP the researchers used the grid search algorithm. The MLP was trained using every possible combination of hyperparameter values. The study utilized a 75% - 25% split for training and testing data respectively. Each trained MLP was evaluated through K-fold cross-validation, it was determined that the models that performed the best were those that utilized 4 hidden layers with 4 neurons each

[6].

Another study detailed the use of a Neural Network model, specifically a Nonlinear Autoregressive model with eXogeneous inputs (NARX), to model and predict Zika outbreaks for the 2015-2016 Zika epidemic in the Americas [2]. A NARX model relates the current value of a time series to the past values of the same series. and it also factors in the current and past values of an external series that influences the time series. The researchers utilized socioeconomic, population, epidemiological, travel, and mosquito suitability data. Specifically, the data consisted of the following:

- Weekly Zika cases taken from Pan American Health Organization
- Passenger volume, Origin, Destination, and Stopovers from 240 airlines and 3400 airports.
- Habitat suitability for the mosquitoes responsible for Zika spread
- GDP
- The ratio of physicians and hospital beds per 10,000 persons
- Population density

In determining the incoming and outgoing travel volumes per country and territory on a monthly interval, the researchers curve fitted the data using the smoothing spline method. The researchers also took into account a calculated risk variable that was calculated per week per territory, and attempted to determine the risk posed by a potentially infected passenger traveling to a specific place at a specific time [2].

Data was normalized to a range of [0,1] prior to being fed into the NARX model. This was done as the model's desired output was a binary risk classifier for each region. The risk designation was taken from a ranking of the regions where an arbitrary number of countries with the highest comparative risk were labelled as high risk. The model proved to be better when predicting smaller time windows, and was more accurate when modelling isolated areas mainly connected through air travel. However, a limit of this binary classification method is that it may not accurately represent transmission patterns between regions [2].

2.2.1.2 Feature Engineering in Machine Learning

Feature engineering is an important facet when it comes to machine learning [24]. It is a process that creates features from available data to improve a model's accuracy and capture latent effects, or effects that exist but have not yet been manifested [24]. Additionally, having data outside of the primary data set is important to have when training a model. This is because it can improve a model's accuracy and its performance [24]. Studies that have used machine learning to model diseases have undergone challenges in training accurate models, with one problem being the ”curse of dimensionality” that comes with having too many features. Thus, feature engineering steps such as dimensionaltiy reduction and feature selection are necessary processes for these studies [1].

2.2.1.3 Random Forest

In a study that dealt with forecasting COVID-19 cases at a U.S. county level, researchers developed a non-parametric Random Forest (RF) model [11]. It mainly used the following data for forecasting cases 1 to 4 weeks into the future: regional COVID-19 case counts, population mobility data, demographics, and population health data [11]. The evaluation metrics used were mean absolute error (MAE) and R-squared [11]. Unlike other compartmental models such as Susceptible-Exposed-Infectious-Recovered (SEIR) that rely on limited quality assumptions in their design about disease spread dynamics and their interpretability and usability, RF models avoid assumptions about the distribution of input data and generate forecasts based on observed empirical trends in data [11]. The results showed that the Random Forest model was able to outperform other compartmental models [11].

2.2.1.4 Auto-Keras Model Selection

Selection of an appropriate machine learning architecture is an important part of getting accurate results [24]. A method of doing so is through the use of a Neural Architecture Search (NAS), an algorithm designed to automatically design neural networks [15]. However, common NAS algorithms, such as Sequential Model-Based Algorithm Configuration (SMAC) and Tree-based Pipeline Optimization Tool (TPOT), that are available on large cloud computing services are difficult to get access to [15]. Cloud services are not free to use and come with their own charges [15]. Cloud-based AutoML also usually requires compicated software that the average user may not understand [15]. They also lack the needed security and privacy for the data [15]. Auto-Keras (AK), an open source package for Keras that can run locally, is an efficient NAS that utilizes network morphism, which keeps the functionality of a neural network while changing its neural architecture.. Through the use of Bayesian optimization, it is able to guide the design of an appropriate neural network much quicker than previously mentioned NAS algorithms.

2.3 Synthesis

In order to properly forecast COVID-19 cases, there first needs to be proper collection of data. Risk factors of a disease such as COVID-19 are varied, ranging from community mobility to the availability of hospital beds and equipment. After collecting the risk data, it should be properly preprocessed. Finally, in model design and creation, it is important to utilize the correct model. This can be done through the use of Auto-Keras in order to automatically search for a neural network given a dataset. A Random Forest model will then be implemented as a regression comparison to the Auto-Keras neural network's performance.

CHAPTER III

METHODOLOGY

3.1 Research Process

The research process began with data collection in order to find as much relevant information to use in modelling. The data was then cleaned and preprocessed. Afterwards, relevant features from the dataset were selected to train the machine learning model. The data was fed into NAS and after a model was selected and trained, the outputs for given sets of inputs were compiled for further review. Afterwards, evaluating and comparing the results took place and it involved comparison of the root mean squared error (RMSE), mean absolute percentage error (MAPE), and mean absolute error (MAE) values. In the case that the forecasts were unsatisfactory (i.e. had unreasonably large RMSE or MAPE values), the model was retrained using a different combination of parameters to see if there were improvements.

If the poor results persisted, the assumption was that the data format is not ideal, thus it would be preprocessed again and fed back into NAS. The cycle of selecting different sets of features from the dataset and retraining the NAS model constitutes a significant part of the study as shown in Figure 3.1.

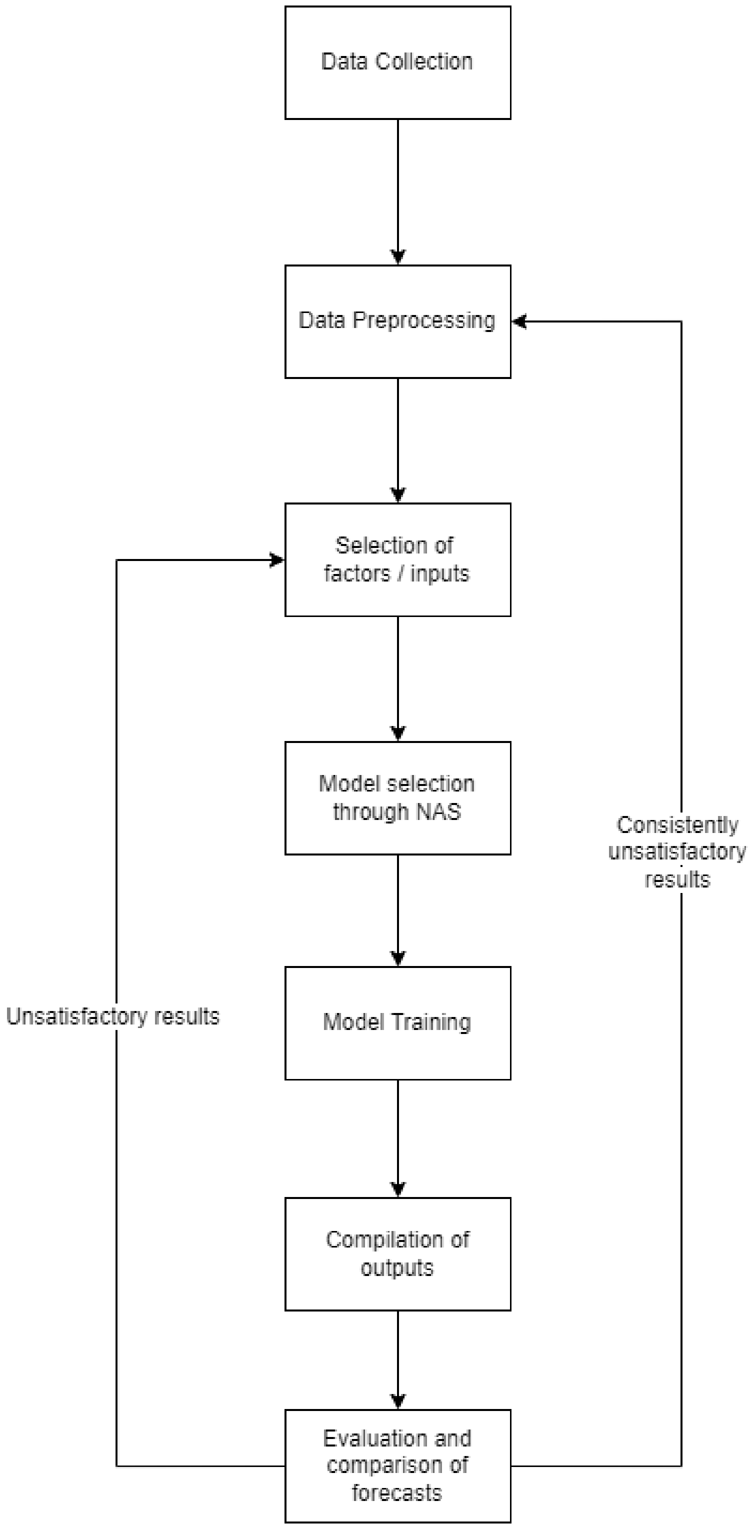
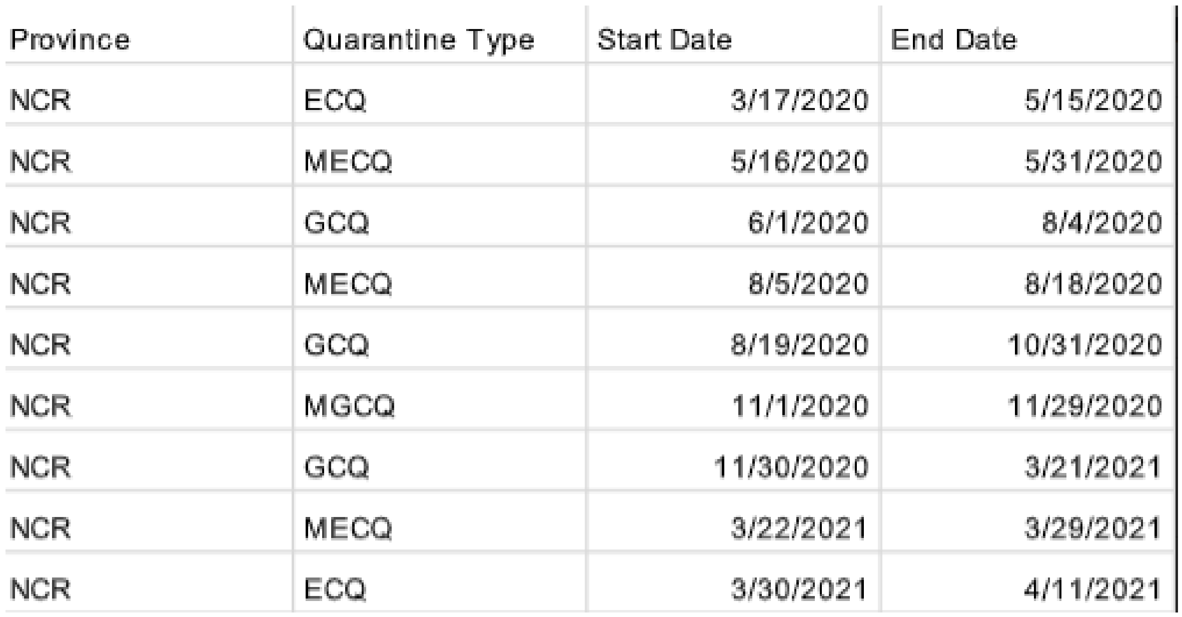

Data Collection
Data Preprocessing
Selection of factors / inputs
Model selection through NAS
Model Training
Compilation of outputs
Evaluation and comparison of forecasts
Unsatisfactory results
Consistently unsatisfactory results

Figure 3.1. Research Process Flowchart

3.2 Data Collection

The researchers collected the epidemiological data for NCR COVID-19 cases from March 2020 to September 2021 from the DOH COVID-19 Data Drop [21]. The data included information on each confirmed case of COVID-19 such as the municipality, city, date where the case was confirmed, and date when the individual either recovered or died. The DOH datasets also include hospital data and the respective quantities of available resources and staff at specific periods. They collected the NCR population data from the Philippines Statistics Authority's (PSA) 2020 population census [5].

The researchers compiled the lockdown restriction data using information from various national news sources and government announcements. The lockdown dataset is comprised of 4 columns namely: the location, the quarantine type, the date when the quarantine was implemented, and the date the quarantine ended. It includes the four types of quarantines found in the Philippines, in decreasing order of strictness of restrictions: ECQ, MECQ, GCQ, and MGCQ [19]. Table 2.1 on page 9 presents descriptions of what each quarantine type entails. Figure 3.2 shows the first few entries of the dataset.



Province	Quarantine Type	Start Date	End Date
NCR	ECQ	3/17/2020	5/15/2020
NCR	MECQ	5/16/2020	5/31/2020
NCR	GCQ	6/1/2020	8/4/2020
NCR	MECQ	8/5/2020	8/18/2020
NCR	GCQ	8/19/2020	10/31/2020
NCR	MGCQ	11/1/2020	11/29/2020
NCR	GCQ	11/30/2020	3/21/2021
NCR	MECQ	3/22/2021	3/29/2021
NCR	ECQ	3/30/2021	4/11/2021

Figure 3.2. NCR Lockdown Types Dataset

The researchers then obtained Philippine community mobility data from Google's COVID-19 community mobility reports [12]. The dataset presents a time series of the increase or decrease in human movement in certain area types, with the values ranging from 100 to -100 as compared to the baseline movement during January 2020 to February 2020 pre-pandemic. The data is presented in a time series and the timeframe available is from February 2020 until present date. Only the dates from March 2020 to September 2021 was utilized in the modelling process. The data mainly featured metropolitan areas and regions as a whole, thus the data for NCR was easily available.

3.3 Data Preprocessing

The researchers used Pandas, a software library written for Python that's used for data manipulation and analysis, to preprocess all their datasets. They performed general preprocessing steps to all their datasets, such as: dropping unused columns, formatting dates with string formats into datetime formats, resampling missing dates with zero as their value, and sorting the datasets by province and dates respectively. Certain columns were also renamed to be more intuitive and "NCR" in all datasets was made uniform (e.g. all National Capital Region fields to NCR). The datasets were joined using the date column as the common value.

3.3.1 COVID Cases and Hospital Data

The researchers aggregated COVID case data from the DOH data drop by date using pandas. Following this, they calculated the case counts in NCR for each day in a time series format, ranging from March 17, 2020 until September 13, 2021. The dates not included in the datasets were compensated through populating the dataset with the missing dates, and assigning them values of 0 cases/day.

For the hospital data, the data included the number of various hospital beds, ventilators, infected hospital staff, confined hospital staff, confined individuals, and severity of cases. Only columns containing hospital beds, specifically ward beds and isolation beds, were utilized as the other columns were mostly comprised of missing values. The columns for each type of bed were initially separated into 2 columns each, one for COVID patients and another for non-COVID patients, but were merged to represent how the hospitals utilized most, if not all, of their resources to accommodate COVID patients. The data retained was comprised of: Isolation Beds and Ward Beds.

The researchers aggregated the hospital data per date, thus the number of beds in all hospitals within NCR were summed up together per date. Similar to how missing dates were handled in the case data, the same was done with the hospital data by using 0 as the default filler value. An important matter to note is that the hospital dataset's first date is on July 2, 2020, thus almost all the hospital data values from March 2020 until July 2, 2020 are zeroes.

3.3.2 Incidence Rate, and Susceptible Counts Data

Because the Philippine population data for 2021 was unavailable, the researchers used the 2020 Philippine population data to calculate for susceptible counts. For the susceptible counts, C represents the number of new cases of the disease per day while S represents the number of the population at risk of contracting the disease during the same period of time as C. Deaths as D and Recoveries as R also follow the time frame of C. To obtain S per day, the researchers simply subtracted the number of cases, deaths, and recoveries from the population of NCR, as shown below in Equation 3.1.

$$S = P - (C + D + R) \tag{3.1}$$

Following this, the PSA [4] provides a formula that utilizes the susceptible counts to calculate the incidence rate of a disease as shown in Equation 3.2:

$$I = \frac{C}{S} \times 100,000 \tag{3.2}$$

The incidence rate I is the rate at which new cases of a certain disease occur in the population. C and S still represent the number of daily cases and susceptible individuals respectively. This rate is usually expressed as the number of cases per 100,000 persons hence the multiplier.

In their final dataset, the researchers appended the incidence rate and susceptible counts columns then populated them per date. The population size remained static for all dates while both the incidence rates and susceptible cases were dynamic per day, causing the researchers to drop the population size column after using it to obtain the latter two as a static column would be provide no value in the context of a time series forecast.

3.3.3 Quarantines in the NCR Dataset

The researchers created their own dataset detailing quarantines in NCR as a compilation of relevant news sources and government announcements. Since the data was in the form of categorical data, they integer encoded the strings and assigned different values based on how restrictive the lockdowns were in terms of movement. Table 3.1 contains each lockdown type with its corresponding integer value.

Lockdown Type	Integer Value
ECQ	4
MECQ	3
GCQ	2
MGCQ	1

Table 3.1. Integer Encoded Values of Each Lockdown

The quarantine types column in the dataset is also in the form of a time series, with each date in NCR having an assigned integer based on which lockdown they were under at that time.

3.3.4 Compiled Dataset

After preprocessing all the datasets, the researchers compiled them into a singular dataset that was fed into the machine learning and neural network models. The dataset's rows each signify one date in NCR while the columns contain the various risk factors as show in Table 3.2. The compiled dataset has around 546 rows and is sorted by date, representing 546 days in NCR where each day had a different number of COVID cases.

Data	Timescale
COVID-19 Cases	Daily
Incidence Rates	Daily
Susceptible Counts	Daily
Recovered Counts	Daily
Death Counts	Daily
Lockdown Data	Daily
Community Mobility Rates	Daily
Hospital Equipment	Daily

Table 3.2. Possible Forecasting Model Inputs.

3.4 Machine Learning Model

The researchers implemented a Random Forest (RF) model using Skforecast, a Python library that eases using Scikit-Learn regressors as multi-step forecasters [20], to be used as a comparison model to Neural Architecture Search (NAS) model's performance. The researchers implemented two types of RF models for forecasting; a "base" model, which used only the number of COVID-19 cases , and a "risk" model, which used every possible risk factor the researchers have gathered in addition to the COVID-19 case counts. These are to determine if the addition of risk factors have a significant effect on the the forecasts of the RF. Table 3.3 shows a breakdown of the Base and Risk models and their features for both RF and NAS.

The researchers obtained the average performance of the models using a ten-fold cross-validation. Optimal hyperparameters for Base and Risk RF models were also obtained using Skforecast's `grid_search_forecaster` with half (50%) the NCR dataset being used as the validation set. The function implements the validation using time series backtesting, which assess the mean squared error (MSE) of a forecast using the existing data and aims to find hyperparameters that minimize the MSE. Table 4.1 presents the optimal hyperparameters used in the RF models.

Model type	**Features**
Base model	Case counts
Risk model	Case counts Incidence rates Susceptible counts Recovered counts Hospital equipment data Lockdown restriction data Community mobility data

Table 3.3. Feature inputs for Base and Risk models

Each RF model in the cross-validation set was trained on the full NCR dataset (March 17, 2020 to September 13, 2021) and is made to generate a 14-day forecast for September 14–27, 2021, a 30-day forecast for September 14 - October 13, 2021, and a 60-day forecast for September 14 - November 12, 2021. Their forecasts were compared against the actual recorded case counts for each respective time span as the testing set. The RMSE, MAPE, and MAE of each model's results, as well as the average and standard deviation of the cross validation set, were taken note of.

3.5 Neural Network Model

Similar to the RF models, the researchers implemented a “base” NAS model and “risk” NAS model with their respective feature inputs seen in Table 3.3. The input data of the model was arranged in the form (Samples, Timesteps, Features). The input of the Neural Network Models were scaled to [0,1] using a MinMaxScaler. Input was split 80/20 for train-validation data. The models were made using the `TimeSeriesForecaster` class of AutoKeras as it utilized NAS to search for the optimal model structure that could create the best forecast based on the time series data it was given [15]. Both NAS models were configured to test up to a maximum of 10 Keras architectures and to train each proposed architecture over 10 epochs, with the objective of minimizing validation loss per epoch. Tenfold cross-validation was also applied to the models to determine their average performance. Each NAS model iteration had also generated a 14-day forecast for September 14–27, 2021. This process was repeated for the 30-day forecast and the 60-day forecast, instead forecasting the dates September 14 – October 13 of 2021, and September 14 – November 12 of 2021, respectively. For the 30-day and 60-day forecasts the lookback of the model was set to same number of days as the forecast length. The forecasts were compared against the actual case counts and the RMSE, MAPE, and MAE of each model were obtained, The average and standard deviation of the cross validation set were also recorded.

3.6 Model Sensitivity Analysis and Feature Selection

Due to the number of potential risk factors, it is important to determine which of the possible risk factors have the highest effect on the model's accuracy. For the RF model, Skforecast's `get_feature_importance` was able to determine which risk factors were the most significant in the forecasting. It showed that only two risk factors were contributing to the forecasting as they had around 98% of the overall feature importance. These risk factors were the *incidence rate* and *susceptible counts* and it is assumed that they were seen as the most significant risk factors due to the former being the direct proportion of people infected per day while the latter is inversely proportional to the daily infected counts per day.

However, tests utilizing only the COVID-19 cases and the two risk factors yielded forecasts that were extremely underfit. Similarly, RF Risk models that utilized all the risk factors also resulted in extremely underfit forecasts. It is assumed that regression models struggle to perform well when utilizing too many variables. Thus, feature dimensionality reduction steps were implemented on the RF model's risk dataset, specifically principal component analysis (PCA) then t-distributed stochastic neighbor embedding (t-SNE).

PCA was first used to reduce the 14 risk factors into 3 principal components to retain around 95% of the risk factors' variance. t-SNE was then used on the principal components to produce 2 components, further reducing the dimensionality of the risk factors. A ten-fold cross-validation was implemented to obtain the best TSNE seed. Table 3.4 presents the RF models' updated features after the implementation of the feature dimensionality reduction.

RF Model type	**Features**
Base model	Case counts
Risk model	Case counts
	t-SNE component 1
	t-SNE component 2

Table 3.4. Feature inputs for the RF models after dimensionality reduction

CHAPTER IV

Results and Discussion

4.1 Results

The models for the Random Forest (RF) and the Neural Architecture Search (NAS) made use of only the National Capital Region of the Philippines for their inputs. Tables and graphs containing the best model outcomes are shown in this section while the more specific ten-fold cross-validation tables and larger graphs can be found in Appendices B and C respectively.

For the RF models, table 4.1 presents the optimal hyperparameter, RMSE, MAPE, and MAE values for the best Base and Risk RF models in the tenfold cross validation with their respective forecast date ranges. Figures 4.1 and 4.2 present the plots of the Base and Risk RF models, respectively. The best performing RF model was a 14-day Risk RF model with an RMSE of 794 and a MAPE of 13.76% as seen in Table 4.1. It can be noted that the further into the future the RF model forecasts, the worse its performance becomes. The RF Risk 60-day forecast seems to be an outlier, as it's RMSE is much lower than the ten-fold cross-validation average RMSE of 3955 as seen in table B.3

Model Name	lags	depth	estimators	RMSE	MAPE (%)	MAE
RF Base 14-day	30	15	20	826	16.09	564
RF Base 30-day	7	10	150	3,313.83	130.17	2,928.47
RF Base 60-day	7	15	20	4,668.03	556.84	3,535.45
RF Risk 14-day	30	10	20	794	13.76	546
RF Risk 30-day	7	5	150	3,070.75	120.11	2,704.51
RF Risk 60-day	60	5	20	1,105.45	82.15	925.00

Table 4.1. Best RF Models

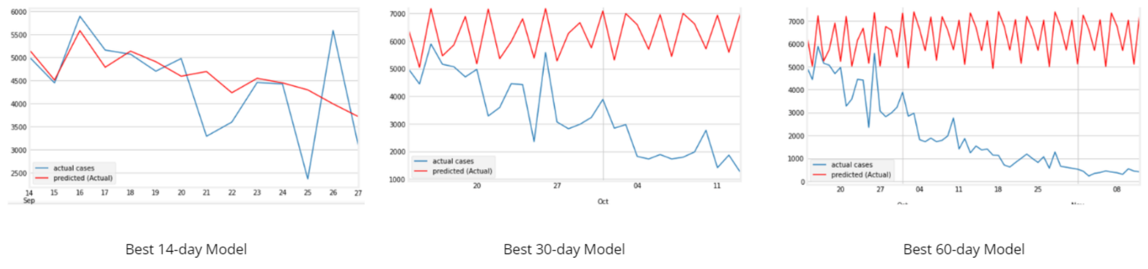


Figure 4.1. Best 14-Day, 30-Day, and 60-Day Base RF Models

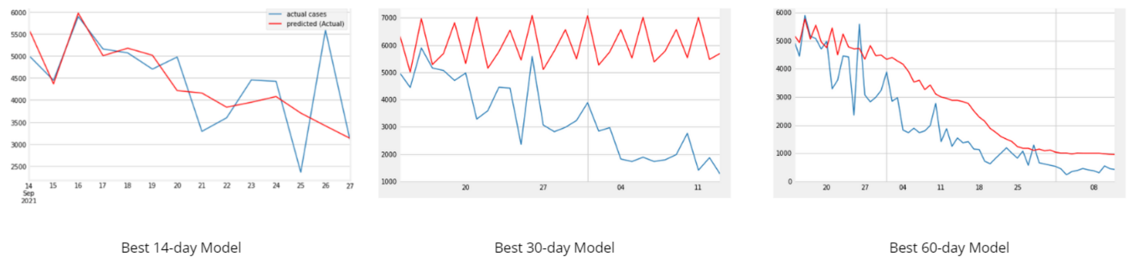


Figure 4.2. Best 14-Day, 30-Day, and 60-Day Risk RF Models

The best performing NAS model was a 60-day Risk model with an RMSE of 819.39 and MAPE of 62.97% as seen in Table 4.2. The graphs of the best performing models are shown in Figures 4.3 and 4.4.

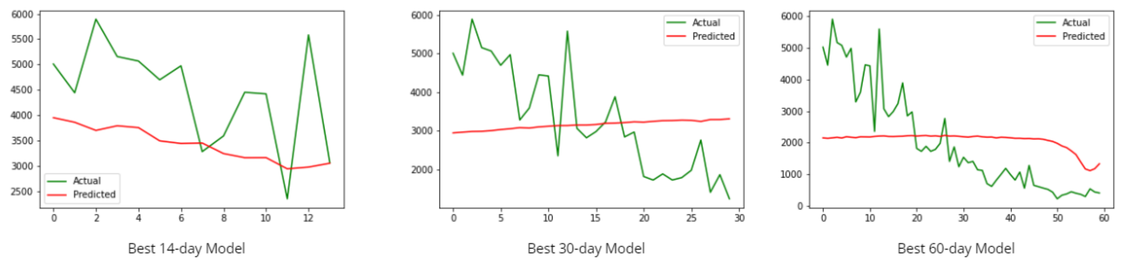


Figure 4.3. Best 14-Day, 30-Day, and 60-Day Base NAS Models

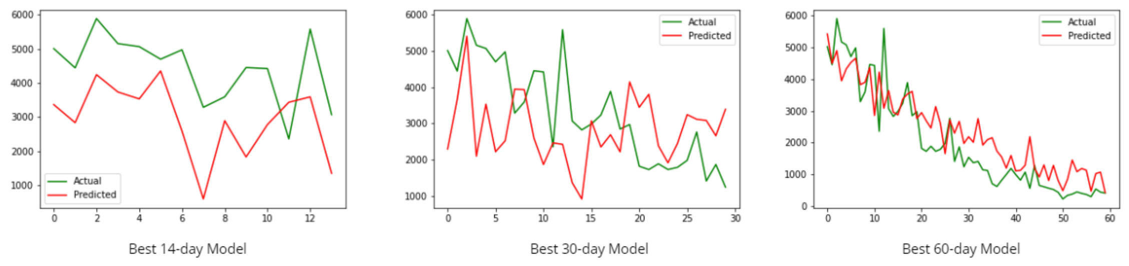


Figure 4.4. Best 14-Day, 30-Day, and 60-Day Risk NAS Models

Model Name	**Lookback**	**RMSE**	**MAPE (%)**	**MAE**
NAS Base 14-day	14 days	1312.50	23.14	1106.86
NAS Base 30-day	30 days	1460.69	45.55	1242.49
NAS Base 60-day	60 days	1523.56	126.30	1297.30
NAS Risk 14-day	14 days	1765.81	38.95	1646.37
NAS Risk 30-day	30 days	1638.25	46.65	1367.56
NAS Risk 60-day	60 days	819.39	62.97	649.45

Table 4.2. Best NAS Models

When comparing with the metric values of the best RF and NAS models alone, it was observed that both the risk and base RF models performed better than the NAS models for 14-day forecasts. On the other hand, NAS models performed significantly better than the RF models when forecasting 30 days and 60 days into the future.

When comparing risk models to base models, the risk factors have shown to benefit both the RF and NAS models, but benefit NAS models more significantly. The improvement in RF models can be observed the most in the 60-day forecasts, with the 14-day and 30-day forecast only minimally changing. Risk NAS models, on the other hand, were more accurate at forecasting and following the actual trend when compared to base NAS models, particularly in the 60-day forecast models. Base NAS models exhibited a more conservative, almost horizontal forecast trend. All NAS base models were unable to forecast the peaks with just case counts alone. Risk NAS models also improved in accuracy the further into the future they forecasted, though this may also be caused by the increase in lookback days. Inversely, the forecast accuracy of both base and risk RF models suffer the further into the future the forecast gets.

This led to the observation that both RF and NAS models create better forecasts when utilizing risk factors, as the risk factors have shown to directly benefit the forecasting accuracy and trends of the models.

CHAPTER V

Conclusions and Future Work

5.1 Conclusions

The researchers were able to collect and feature engineer COVID-19 case and risk factor data to successfully implement a neural network model that is able to forecast COVID-19 cases in NCR. They were also to compare the performance of classical modeling approaches to state-of-the-art neural architecture search given epidemiological risk factors that affected the greater transmission of the virus.

The researchers were able to implement the neural network model, NAS, and the comparison machine learning model, RF, in forecasting COVID-19 cases in NCR. They can conclude that when forecasting using both case data only and case data with risk factors, the RF models performed better than NAS models for 14-day forecasts while the NAS models performed significantly better than the RF models for 30-day and 60-day forecasts. The results show that while the Random Forest regressor performs best with short-term forecasts, the Neural Architecture Search models perform better in long-term forecasts, which is more significant.

The best overall model in terms of RMSE, MAPE, and MAE was a 14-day Base RF model while the model that had the best fit in terms of the prediction trend and still had great performance metrics was a 60-day Risk NAS model. It could be seen that the presence of risk factors produces more consistent and accurate model forecasts on average while also allowing the forecasts to follow the actual case' trend.

Neural Architecture Search provides a good baseline model to begin predictions. However, it does requires further tuning of the model to more accurately predict the spread of COVID-19.

Finally, integrating risk factors or other multivariable inputs in the training of time-series forecasting neural network models could help improve model performance and provide more accurate forecasts of COVID-19 spread in the future.

5.2 Recommendations for Future Work

The Neural Architecture Search tool AutoKeras provides a good starting point for time series forecasting. However, it may be further improved through getting the best architecture and further improving upon it outside of AutoKeras. Due to the study's limitation of 10-fold cross-validation, it is recommended to increase the number of seeds tested and to increase the sample size of tests in order to provide a better idea of how to effectively utilize NAS' in disease forecasting. Experimenting with further forecast dates and ranges may also add more insights on the model's limitations.

Furthermore, it is also recommended to explore NAS's performance given more data points, as this may increase the accuracy of the models chosen. A potential alternative could be to forecast cases from various locations in the Philippines with a similar population size as NCR then implement an ensemble learning technique. This method would address the problem of lacking data points for training, as the other models' weights would contribute to the main model's overall training.

Finally. performing a feature sensitivity analysis on NAS is also highly recommended, as it can assist in determining which risk factors are the most significant to use and which are not.

BIBLIOGRAPHY

[1] AHMAD, A., GARHWAL, S., RAY, S. K., KUMAR, G., MALEBARY, S. J., AND BARUKAB, O. M. The number of confirmed cases of covid-19 by using machine learning: Methods and challenges. *Archives of Computational Methods in Engineering* (2020).

[2] AKHTAR, M., KRAEMER, M. U., AND GARDNER, L. M. A dynamic neural network model for predicting risk of zika in real time. *BMC medicine 17*, 1 (2019), 1–16.

[3] ARGOSINO, F. Covid-19 response: A timeline of community quarantine, lockdowns, alert levels. Manila Bulletin. Retrieved from https://mb.com.ph/2021/11/09/covid-19-response-a-timeline-of-community-quarantine-lockdowns-alert-levels/.

[4] AUTHORITY, P. S. Philippine statistics authority incidence rate (of a disease). *Philippine Statistics Authority.[Google Scholar]* (2017).

[5] AUTHORITY, P. S. 2020 census of population and housing (2020 cph) population counts declared official by the president. *Philippine Statistics Authority* (2021).

[6] Car, Z., Baressi Segota, S., Andelic, N., Lorencin, I., and Mrzljak, V. Modeling the spread of covid-19 infection using a multilayer perceptron. *Computational and mathematical methods in medicine 2020* (2020).

[7] Chang, R., and Tam, F. Methodology: Inside bloomberg's covid resilience ranking. Retrieved from https://www.bloomberg.com/news/articles/2020-11-24/inside-bloomberg-s-covid-resilience-ranking.

[8] David, G., Rye, R. S., and Agbulos, M. P. Covid-19 forecasts in the philippines: Insights for policy-making, Apr 2020.

[9] El Naqa, I., and Murphy, M. J. What is machine learning? *Machine Learning in Radiation Oncology* (2015), 3–11.

[10] Frérot, M., Lefebvre, A., Aho, S., Callier, P., Astruc, K., and Aho Glélé, L. S. What is epidemiology? changing definitions of epidemiology 1978-2017. *PLOS ONE 13*, 12 (2018).

[11] Galasso, J., Cao, D. M., and Hochberg, R. A random forest model for forecasting regional covid-19 cases utilizing reproduction number estimates and demographic data. *Chaos, Solitons & Fractals* (2022), 111779.

[12] GOOGLE. Google covid-19 community mobility reports. Retrieved from https://www.google.com/covid19/mobility/.

[13] HE, S., PENG, Y., AND SUN, K. Seir modeling of the covid-19 and its dynamics. *Nonlinear Dynamics 101*, 3 (2020), 1667–1680.

[14] ILIE, O.-D., COJOCARIU, R.-O., CIOBICA, A., TIMOFTE, S.-I., MAVROUDIS, I., AND DOROFTEI, B. Forecasting the spreading of covid-19 across nine countries from europe, asia, and the american continents using the arima models. *Microorganisms 8*, 8 (2020), 1158.

[15] JIN, H., SONG, Q., AND HU, X. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 1946–1956.

[16] KEELING, M. J., AND ROHANI, P. *Modeling infectious diseases in humans and animals*. Princeton University Press, 2007.

[17] MONTALAN, J. R., ESTUAR, M. R. J., TEKNOMO, K., AND GARDON, R. W. Measles metapopulation modeling using ideal flow of transportation networks. In *Proceedings of the 2nd International Conference on Software Engineering and Information Management* (2019), pp. 147–151.

[18] Nature Communications. Epidemiology is a science of high importance. *Nature Communications 9*, 1 (2018).

[19] of the Philippines Inter-Agency Task Force (IATF) for the Management of Emerging Infectious Diseases, R. Omnibus guidelines on the implementation of community quarantine in the philippines. Retrieved from https://doh.gov.ph/sites/default/files/health-update/20210506-OMNIBUS-RRD.pdf.

[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[21] Philippines Department of Health. Doh covid data drop. Retrieved from https://doh.gov.ph/2019-nCoV.

[22] Poirier, C., Liu, D., Clemente, L., Ding, X., Chinazzi, M., Davis, J., Vespignani, A., and Santillana, M. Real-time forecasting of the covid-19 outbreak in chinese provinces: machine learning approach using novel

digital data and estimates from mechanistic models. *Journal of medical Internet research 22*, 8 (2020), e20285.

[23] THE GUARDIAN. Coronavirus timeline: from wuhan to washington state, Jan 2020.

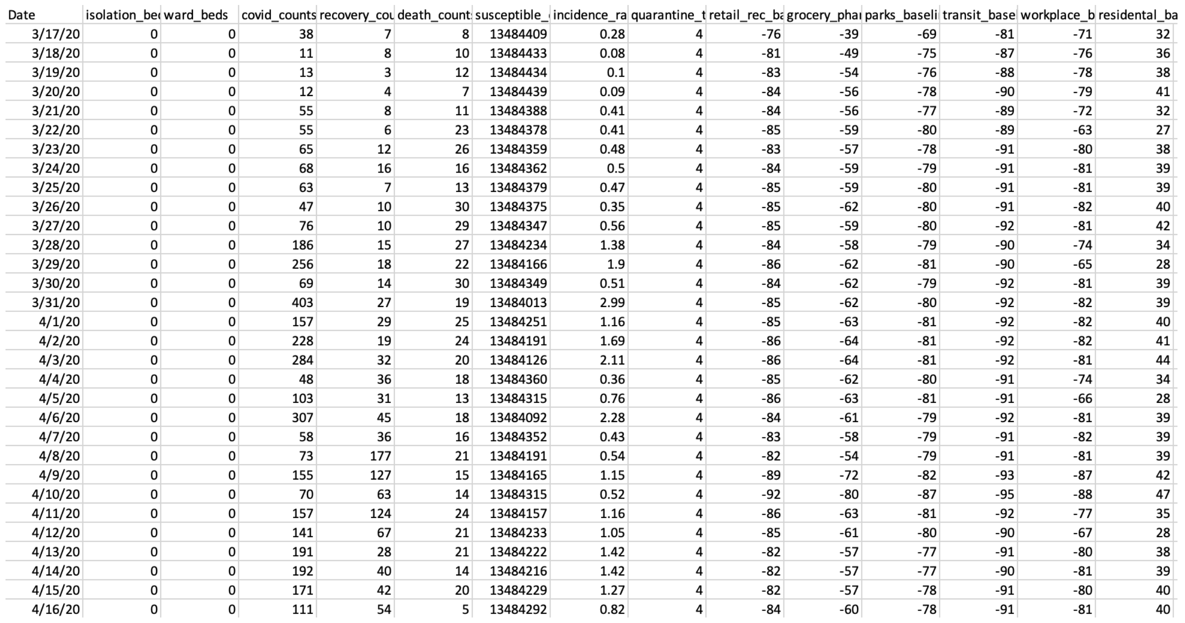
[24] WIEMKEN, T. L., AND KELLEY, R. R. Machine learning in epidemiology and health outcomes research. *Annual review of public health 41* (2020), 21–36.

[25] WORLD HEALTH ORGANIZATION. Modes of transmission of virus causing covid-19: Implications for ipc precaution recommendations.

[26] WORLD HEALTH ORGANIZATION. Naming the coronavirus disease (covid-19) and the virus that causes it.

APPENDIX A

Data



Date	isolation_bec	ward_beds	covid_counts	recovery_cou	death_count:	susceptible_	incidence_ra	quarantine_t	retail_rec_ba	grocery_phai	parks_baseli	transit_basel	workplace_b	residental_ba
3/17/20	0	0	38	7	8	13484409	0.28	4	-76	-39	-69	-81	-71	32
3/18/20	0	0	11	8	10	13484433	0.08	4	-81	-49	-75	-87	-76	36
3/19/20	0	0	13	3	12	13484434	0.1	4	-83	-54	-76	-88	-78	38
3/20/20	0	0	12	4	7	13484439	0.09	4	-84	-56	-78	-90	-79	41
3/21/20	0	0	55	8	11	13484388	0.41	4	-84	-56	-77	-89	-72	32
3/22/20	0	0	55	6	23	13484378	0.41	4	-85	-59	-80	-89	-63	27
3/23/20	0	0	65	12	26	13484359	0.48	4	-83	-57	-78	-91	-80	38
3/24/20	0	0	68	16	16	13484362	0.5	4	-84	-59	-79	-91	-81	39
3/25/20	0	0	63	7	13	13484379	0.47	4	-85	-59	-80	-91	-81	39
3/26/20	0	0	47	10	30	13484375	0.35	4	-85	-62	-80	-91	-82	40
3/27/20	0	0	76	10	29	13484347	0.56	4	-85	-59	-80	-92	-81	42
3/28/20	0	0	186	15	27	13484234	1.38	4	-84	-58	-79	-90	-74	34
3/29/20	0	0	256	18	22	13484166	1.9	4	-86	-62	-81	-90	-65	28
3/30/20	0	0	69	14	30	13484349	0.51	4	-84	-62	-79	-92	-81	39
3/31/20	0	0	403	27	19	13484013	2.99	4	-85	-62	-80	-92	-82	39
4/1/20	0	0	157	29	25	13484251	1.16	4	-85	-63	-81	-92	-82	40
4/2/20	0	0	228	19	24	13484191	1.69	4	-86	-64	-81	-92	-82	41
4/3/20	0	0	284	32	20	13484126	2.11	4	-86	-64	-81	-92	-81	44
4/4/20	0	0	48	36	18	13484360	0.36	4	-85	-62	-80	-91	-74	34
4/5/20	0	0	103	31	13	13484315	0.76	4	-86	-63	-81	-91	-66	28
4/6/20	0	0	307	45	18	13484092	2.28	4	-84	-61	-79	-92	-81	39
4/7/20	0	0	58	36	16	13484352	0.43	4	-83	-58	-79	-91	-82	39
4/8/20	0	0	73	177	21	13484191	0.54	4	-82	-54	-79	-91	-81	39
4/9/20	0	0	155	127	15	13484165	1.15	4	-89	-72	-82	-93	-87	42
4/10/20	0	0	70	63	14	13484315	0.52	4	-92	-80	-87	-95	-88	47
4/11/20	0	0	157	124	24	13484157	1.16	4	-86	-63	-81	-92	-77	35
4/12/20	0	0	141	67	21	13484233	1.05	4	-85	-61	-80	-90	-67	28
4/13/20	0	0	191	28	21	13484222	1.42	4	-82	-57	-77	-91	-80	38
4/14/20	0	0	192	40	14	13484216	1.42	4	-82	-57	-77	-90	-81	39
4/15/20	0	0	171	42	20	13484229	1.27	4	-82	-57	-78	-91	-80	40
4/16/20	0	0	111	54	5	13484292	0.82	4	-84	-60	-78	-91	-81	40

Figure A.1. A snippet of the final compiled dataset used in training the NAS and RF models

All datasets used in the research can be found here. This includes the raw input datasets and the final datasets generated.

APPENDIX B

Model Metric Results

B.1 Random Forest

Iterations	RMSE		MAPE (%)		MAE	
	Base RF	**Risk RF**	**Base RF**	**Risk RF**	**Base RF**	**Risk RF**
1	1451.00	1370.18	32.66	30.10	1154.00	1077.82
2	1521.00	1217.67	32.60	34.05	1190.00	901.36
3	953.00	1516.60	20.71	34.31	712.00	1258.67
4	1488.00	1236.15	32.15	26.81	1170.00	910.42
5	1619.00	**794.06**	35.98	**13.76**	1350.00	**545.87**
6	**826.00**	1542.07	**16.09**	34.85	**564.00**	1201.59
7	1327.00	1711.43	28.87	37.85	1003.00	1375.47
8	1449.00	1477.85	31.49	32.65	1114.00	1164.40
9	1621.00	1127.75	35.69	23.67	1351.00	794.99
10	2055.00	1361.15	47.23	30.57	1743.00	1066.87
Average	1431.00	1335.79	31.35	29.11	1135.00	1029.75
Std Dev	346.04	258.22	8.49	6.91	331.65	245.69

Table B.1. RF Model Metric Results (14 days)

Iterations	RMSE		MAPE (%)		MAE	
	Base RF	**Risk RF**	**Base RF**	**Risk RF**	**Base RF**	**Risk RF**
1	3563.00	3403.00	139.00	134.00	3159.00	3006.00
2	3788.00	3836.00	149.00	151.00	3379.00	3404.00
3	3883.00	**3071.00**	153.00	**120.00**	3461.00	**2705.00**
4	**3314.00**	3502.00	**130.00**	138.00	**2928.00**	3069.00
5	3544.00	3130.00	138.00	123.00	3135.00	2750.00
6	3406.00	3489.00	133.00	137.00	2985.00	3081.00
7	3486.00	3731.00	137.00	147.00	3105.00	3307.00
8	3485.00	3521.00	137.00	139.00	3101.00	3115.00
9	3898.00	3491.00	153.00	137.00	3460.00	3092.00
10	3974.00	3182.00	157.00	125.00	3535.00	2815.00
Average	3634.00	3435.00	143.00	135.00	3225.00	3035.00
Std Dev	220.00	236.00	9.00	10.00	205.00	215.00

Table B.2. RF Model Metric Results (30 days)

Iterations	RMSE		MAPE (%)		MAE	
	Base RF	**Risk RF**	**Base RF**	**Risk RF**	**Base RF**	**Risk RF**
1	5157.00	5237.00	630.00	627.00	3159.00	4946.00
2	5457.00	4827.00	656.00	592.00	3379.00	4427.00
3	5223.00	4185.00	635.00	509.00	3461.00	3849.00
4	5242.00	4740.00	632.00	574.00	2928.00	4389.00
5	5337.00	3945.00	645.00	482.00	3135.00	3604.00
6	5106.00	3520.00	621.00	434.00	2985.00	3197.00
7	4768.00	3594.00	572.00	407.00	3105.00	3371.00
8	4957.00	4702.00	604.00	568.00	3101.00	4376.00
9	5321.00	3695.00	646.00	455.00	3460.00	3345.00
10	**4668.00**	**1105.00**	**557.00**	**82.00**	**3535.00**	**925.00**
Average	5124.00	3955.00	620.00	473.00	3225.00	3643.00
Std Dev	241.00	1102.00	31.00	147.00	205.00	1058.00

Table B.3. RF Model Metric Results (60 days)

B.2 Neural Architechture Search

Iterations	RMSE		MAPE (%)		MAE	
	Base NAS	**Risk NAS**	**Base NAS**	**Risk NAS**	**Base NAS**	**Risk NAS**
1	2113.21	2253.75	39.77	46.40	1900.36	2122.75
2	2585.91	**1765.81**	48.44	38.95	2298.29	**1646.37**
3	2983.65	2216.01	63.18	41.53	2852.49	1988.45
4	**1312.50**	2477.42	**23.14**	49.67	**1106.86**	2289.32
5	2414.70	2551.79	46.54	50.25	2199.90	2321.43
6	2795.81	2106.31	57.31	39.47	2609.08	1890.40
7	2142.52	1989.01	40.40	**37.47**	1928.06	1783.68
8	3109.82	2097.81	64.02	39.63	2942.22	1860.78
9	2125.98	2589.31	40.01	49.47	1911.93	2341.70
10	2828.81	2519.08	52.15	47.13	2497.22	2232.61
Average	2441.29	2256.63	47.50	44.00	2224.64	2047.75
Std Dev	538.42	274.35	12.46	5.06	545.44	248.00

Table B.4. NAS Model Metric Results (14 days)

Iterations	RMSE		MAPE (%)		MAE	
	Base NAS	**Risk NAS**	**Base NAS**	**Risk NAS**	**Base NAS**	**Risk NAS**
1	2299.67	1990.64	48.47	56.89	1875.43	1693.42
2	1521.23	2193.99	55.88	44.56	1353.30	1752.34
3	**1460.69**	2074.49	45.55	**42.65**	1242.49	1621.47
4	2321.02	2288.30	49.15	52.72	1897.84	1814.30
5	2330.16	2097.43	48.30	69.13	1870.60	1766.91
6	2254.15	**1638.25**	49.05	46.65	1865.81	**1367.56**
7	1502.62	2378.90	55.46	73.31	1315.44	2088.60
8	1474.03	2563.83	43.77	60.94	1235.65	2174.06
9	1482.05	1801.25	**34.50**	47.80	**1186.05**	1534.98
10	2192.03	2018.82	43.81	65.96	1741.46	1744.14
Average	1883.76	2104.59	47.39	56.06	1558.41	1755.78
Std Dev	419.07	270.53	6.15	10.91	313.57	238.24

Table B.5. NAS Model Metric Results (30 days)

Iterations	RMSE		MAPE (%)		MAE	
	Base NAS	**Risk NAS**	**Base NAS**	**Risk NAS**	**Base NAS**	**Risk NAS**
1	1811.64	2038.76	83.06	87.27	1291.01	1508.37
2	1544.65	2137.33	134.95	134.61	1339.19	1606.62
3	1857.76	2258.21	73.87	103.11	1311.58	1711.67
4	1894.77	890.73	73.23	68.97	1341.12	650.09
5	1858.45	**819.39**	80.33	**62.97**	1339.49	**649.45**
6	1878.12	1745.77	74.91	116.81	1324.80	1332.72
7	1779.24	1652.26	68.80	149.77	**1261.61**	1499.81
8	**1523.56**	1078.40	126.30	98.20	1297.30	893.51
9	1879.52	901.40	73.36	71.30	1322.90	762.20
10	2140.20	1904.76	**65.81**	88.83	1537.09	1431.05
Average	1816.79	1542.70	85.46	98.18	1336.61	1204.55
Std Dev	177.41	564.80	24.39	28.64	74.86	418.20

Table B.6. NAS Model Metric Results (60 days)

APPENDIX C

Model Plots

C.1 Random Forest

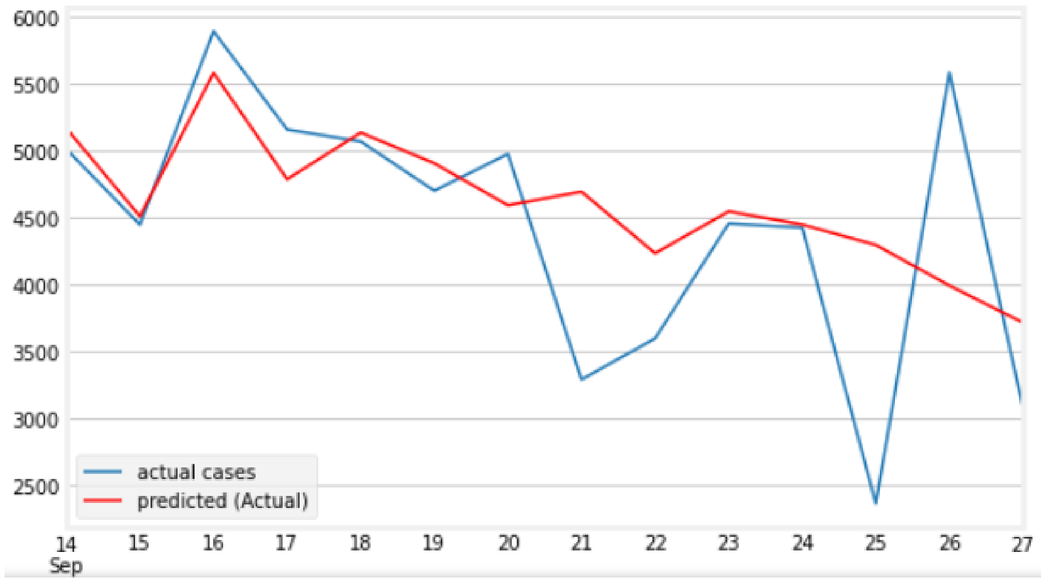


Figure C.1. Base RF model plot (14 days)

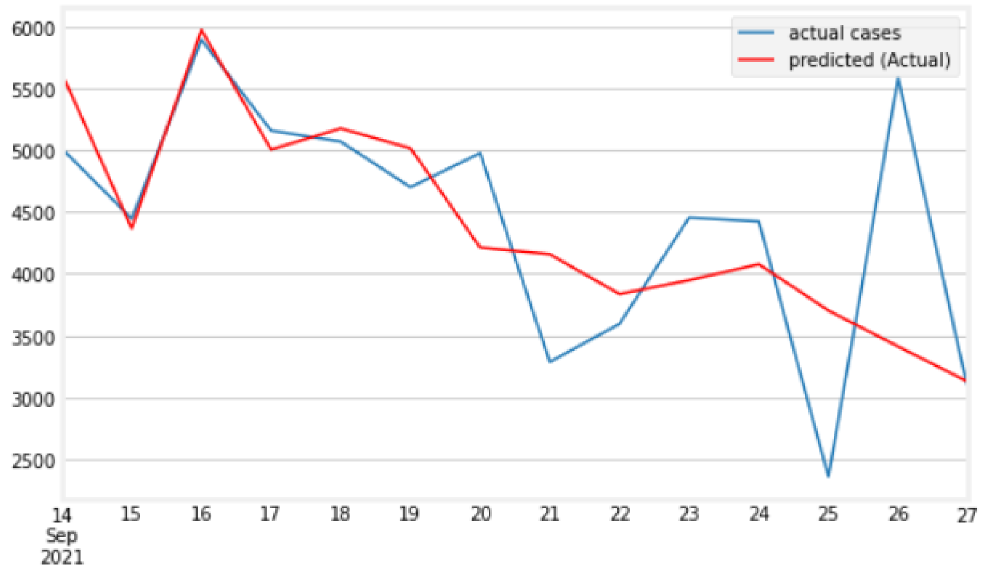


Figure C.2. Risk RF model plot (14 days)

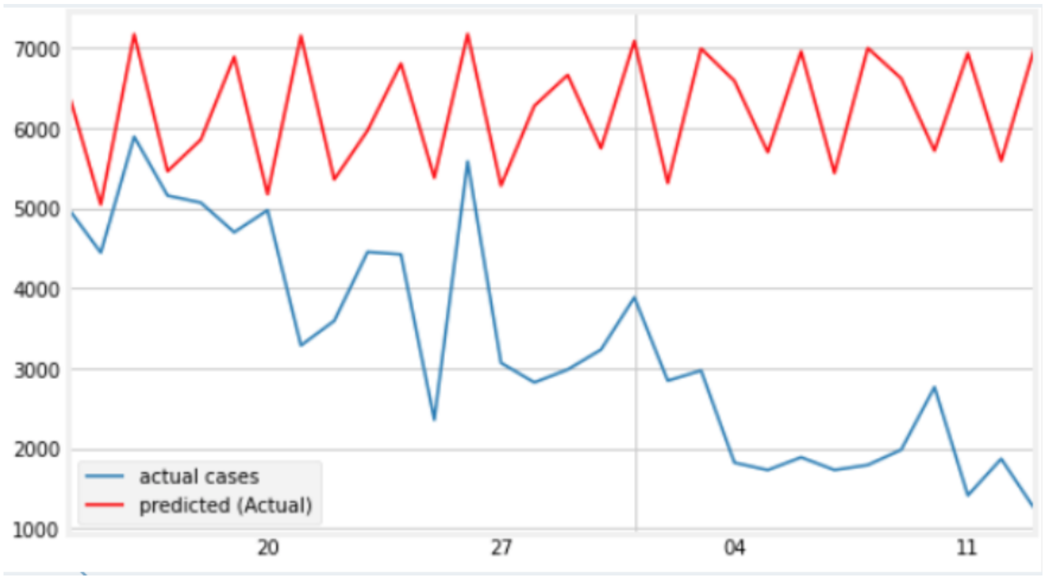


Figure C.3. Base RF model plot (30 days)

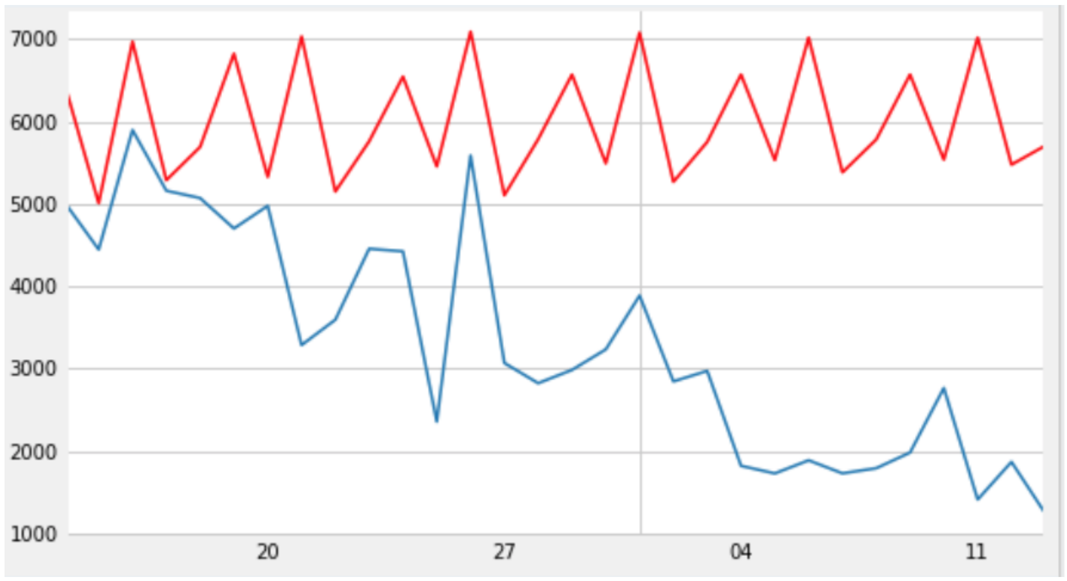


Figure C.4. Risk RF model plot (30 days)

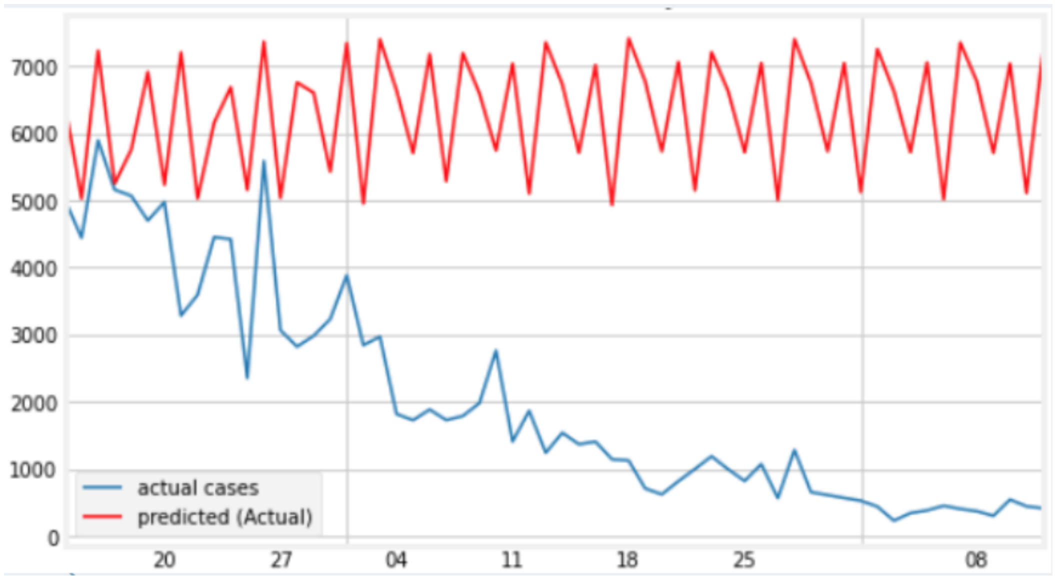


Figure C.5. Base RF model plot (60 days)

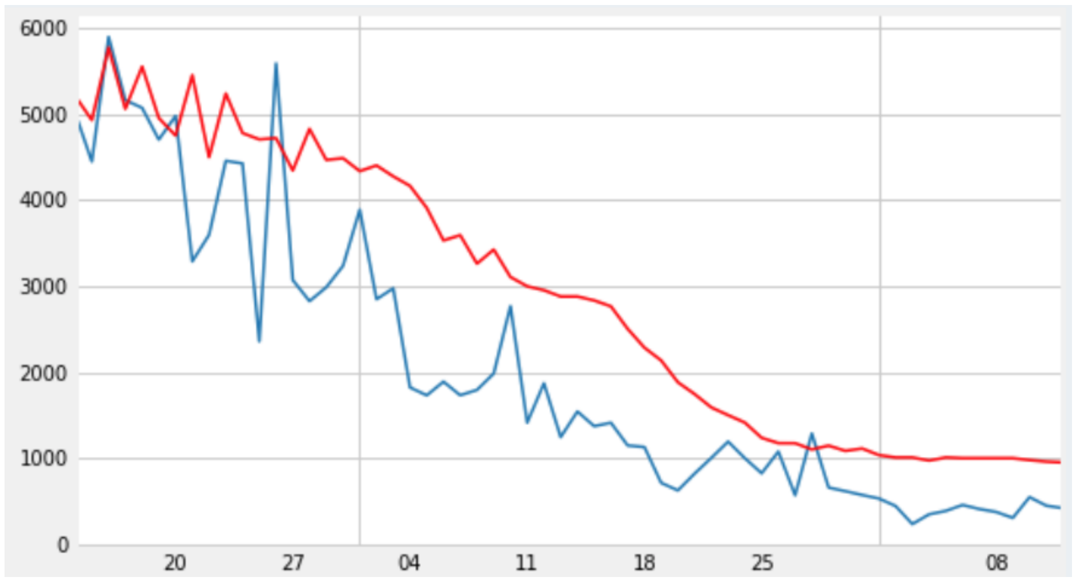


Figure C.6. Risk RF model plot (60 days)

C.2 Neural Architecture Search

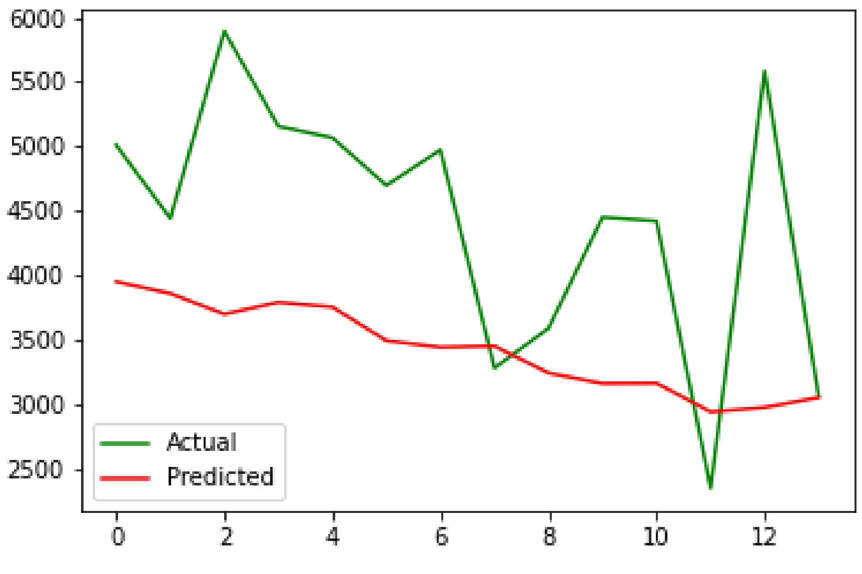


Figure C.7. Base NAS model plot (14 days)

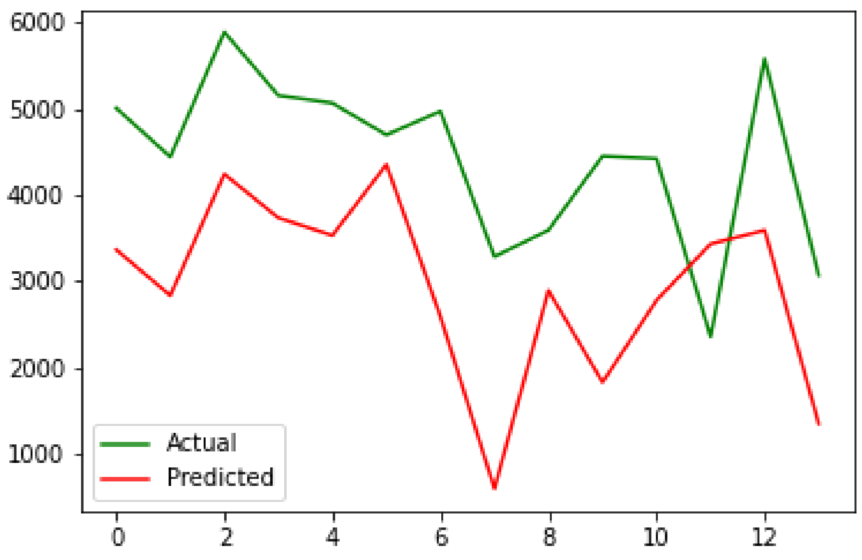


Figure C.8. Risk NAS model plot (14 days)

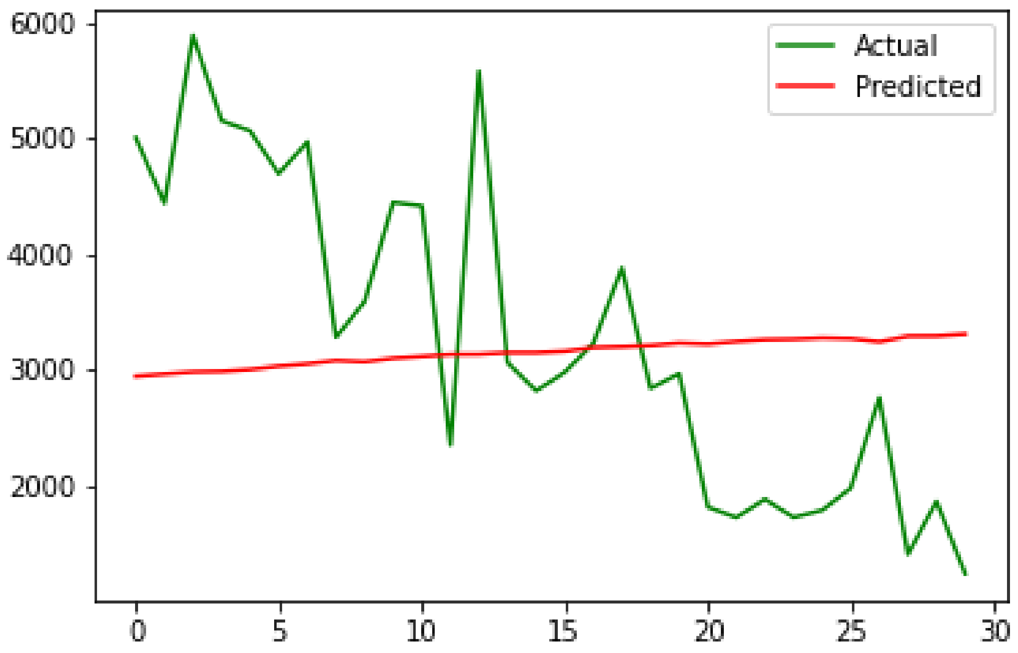


Figure C.9. Base NAS model plot (30 days)

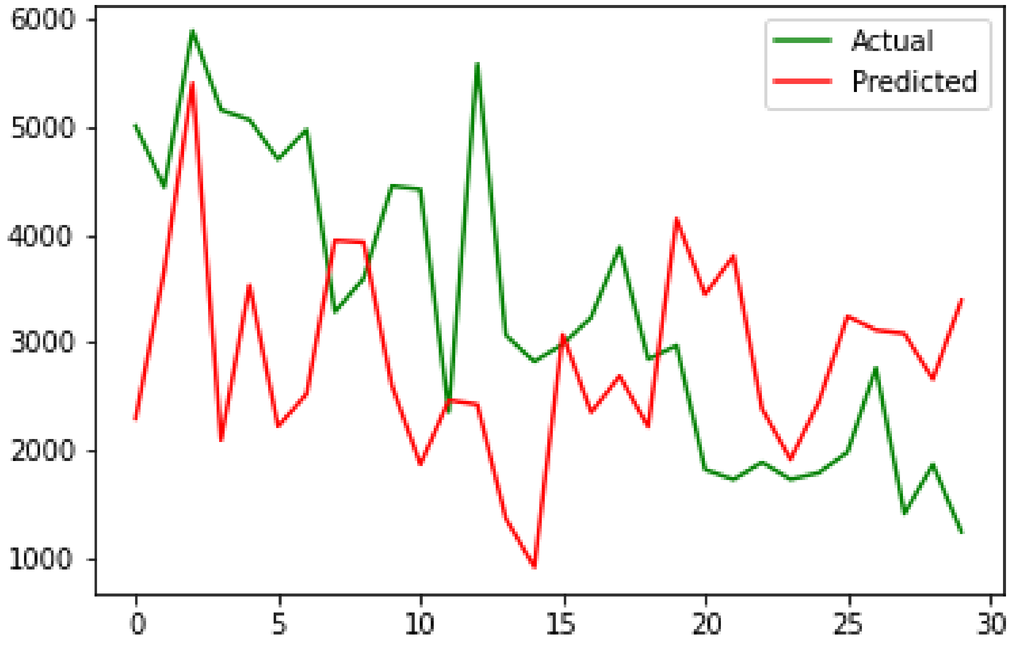


Figure C.10. Risk NAS model plot (30 days)

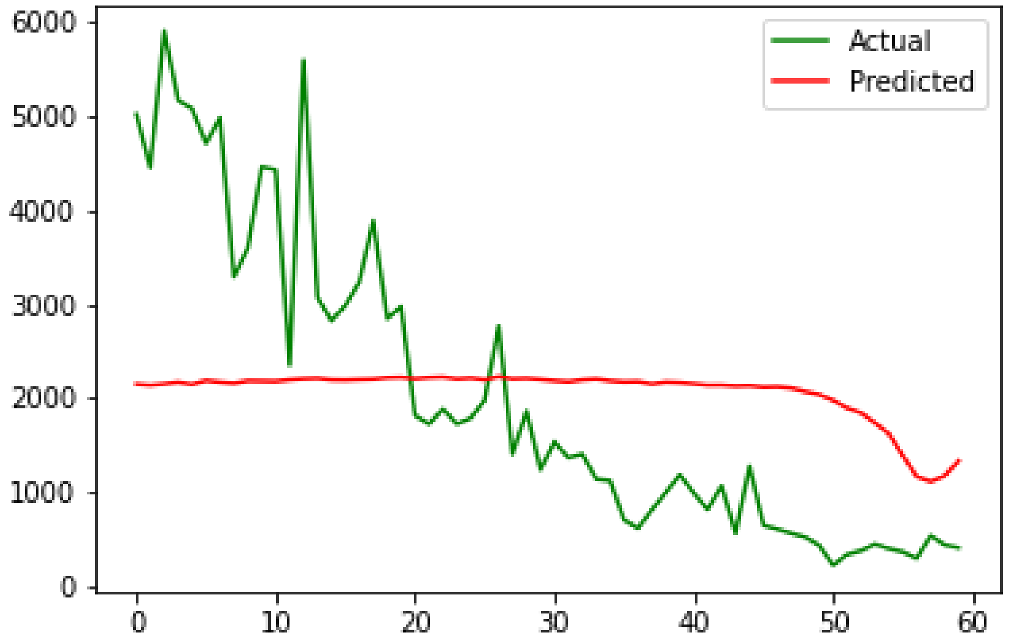


Figure C.11. Base NAS model plot (60 days)

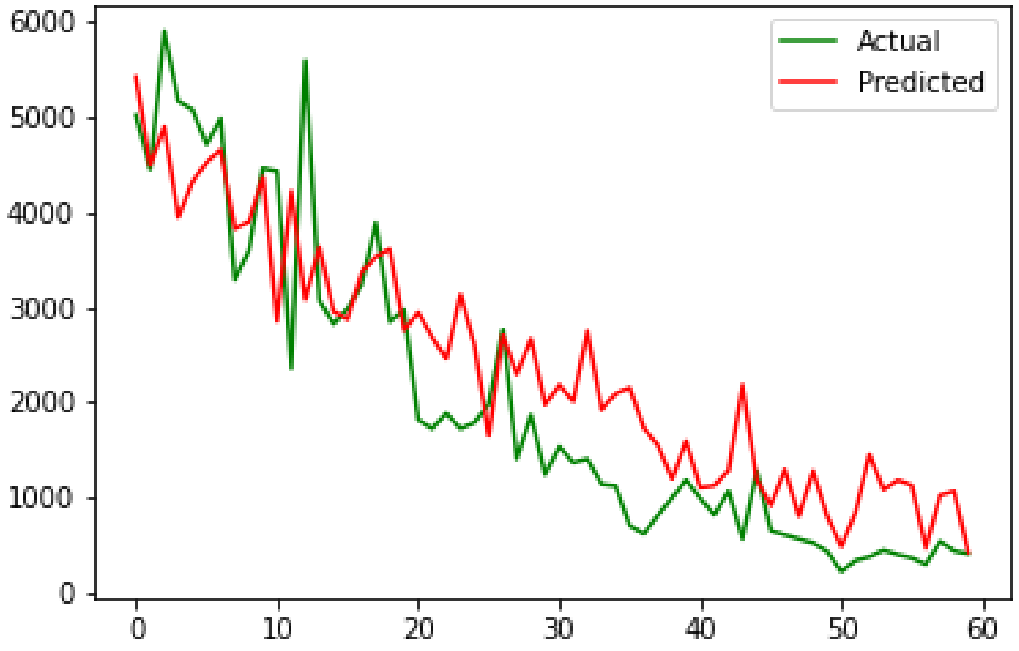


Figure C.12. Risk NAS model plot (60 days)

APPENDIX D

PCSC Manuscript and Poster

D.1 Philippine Computing Science Congress (PCSC) 2022 paper

Comparing Machine Learning approaches in Forecasting COVID-19 using Confirmed Cases and Risk Factors

Forecasting for the National Capital Region of the Philippines

Ryan Bautista
Ateneo de Manila University
Philippines
ryan.bautista@obf.ateneo.edu

Anton Miguel Castillo
Ateneo de Manila University
Philippines
anton.castillo@obf.ateneo.edu

Keith Adrian Santos
Ateneo de Manila University
Philippines
keith.santos@obf.ateneo.edu

Jann Railey E. Montalan
Ateneo de Manila University
Philippines
jmontalan@ateneo.edu

ABSTRACT

The daily number of COVID-19 cases in the National Capital Region (NCR) have continued to rise amidst existing restrictions and safety measures. Researchers have utilized machine learning to forecast these cases to provide more effective measures in reducing the spread of COVID-19. However, their studies primarily focused on the infected cases over a duration of time without including other potential risk factors in virus spread.

This study proposes comparing univariate and multivariate machine learning approaches for accurately modeling and forecasting the spread of COVID-19 in NCR. Data from the Department of Health, COVID-19 Community Mobility Reports from Google, and socioeconomic risk factors from the Philippine Statistics Authority were used to train Random Forest regression models and Neural Architecture Search (NAS) regression models. Results show that the neural networks model trained with COVID-19 confirmed case counts and risk factors performed better than the model trained with only case counts. For the Random Forest models, the opposite was observed, likely due to the variances and noise introduced by the additional training features.

For future work, the researchers would look into augmenting the methodology by conducting more rigorous feature engineering and measuring performance against other relevant forecasting metrics.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; **Neural networks**; **Cross-validation**;

KEYWORDS

Machine Learning, COVID-19 forecasting, Neural Architecture Search, Random Forest model, Risk factors

ACM Reference Format:
Ryan Bautista, Anton Miguel Castillo, Keith Adrian Santos, and Jann Railey E. Montalan. 2022. Comparing Machine Learning approaches in Forecasting COVID-19 using Confirmed Cases and Risk Factors: Forecasting for the National Capital Region of the Philippines. In *Proceedings of Philippine Computing Science Congress (PCSC2022).* City of Manila, Philippines, 5 pages.

1 INTRODUCTION

Since the beginning of the COVID-19 outbreak in early 2020, there has been a need for efficient and accurate approaches to forecast COVID-19 cases in a given area [15]. Before the introduction of the COVID-19 vaccines, governments were heavily reliant on social distancing guidelines and lockdowns to mitigate the spread of the virus. These solutions gave people an understanding of where there was a high risk of infection, which then lead to the formation of proper COVID-19 prevention guidelines.

Previous studies approached the situation using machine learning (ML) algorithms that forecast cases in a given municipality or country. One study made use of a Susceptible-Infectious-Recovered (SIR) model in the early months of the disease's outbreak; their findings were used by the Philippine government to implement Enhanced Community Quarantine (ECQ) restrictions to limit human movement and the spread of the virus [7]. However, the number of cases still continued to rise and more people were being hospitalized.

As such, this study utilized classical and state-of-the-art ML approaches such as Random Forest regression and Neural Architecture Search (NAS) to create models that forecast COVID-19 cases in the National Capital Region (NCR) of the Philippines. The researchers tested univariate and multivariate models to determine if the addition of potential risk factors would have a significant impact on the forecasts. Utilizing two different ML approaches provided additional insights on whether neural networks can perform better than classical ML approaches that mainly deal with regression and classification.

2 RELATED LITERATURE

2.1 Risk Factors for COVID-19 Infection

The study made use of socioeconomic risk factors that could potentially influence an individual's chances of being infected with

COVID-19 and further transmitting the virus. The researchers gathered different risk factors from various government resources and studies, such as: the daily number of COVID-19 cases [14], population density [3], community mobility [9], lockdown restrictions [2], incidence rate [4], hospital equipment [14], and population size [5].

Data	Timescale
COVID-19 Cases	Daily
Population Size	Yearly
Population Density	Yearly
Incidence Rates	Daily
Susceptible Counts	Daily
Recovered Counts	Daily
Death Counts	Daily
Lockdown Data	Daily
Community Mobility Rates	Daily
Hospital Equipment	Yearly

Table 1: Possible risk factors for COVID-19 infection.

2.2 Machine Learning Approaches for Disease Modeling

Previous researches have made use of machine learning models to forecast the spread of infectious diseases throughout the globe, a given country, or a select number of regions.

A study utilized a non-parametric Random Forest model to forecast COVID-19 cases at a U.S. county level. The model primarily used regional COVID case counts, demographics, population mobility, and population health to forecast cases 1, 2, 3, and 4 weeks into the future, with mean absolute error (MAE) and R-squared as the evaluation metrics. It was determined that the model was able to outperform other standard compartmental models [8].

One study used different configurations of a multilayer perceptron (MLP) to forecast the spread of COVID-19 globally. The time series data used in the model contained information on coronavirus patients and the number of patients per location from January 22, 2020 until March 12, 2020. It was determined that the best performing model configurations with the given data had 4 hidden layers with each layer having 4 neurons [6].

Another study made use of a Nonlinear Autoregressive model with eXogeneous inputs (NARX) to forecast outbreaks of the Zika virus in the Americas from 2015 to 2016. The model was given socioeconomic, population, epidemiological, travel, and mosquito suitability data. The output of the study was a binary risk classifier that would determine whether or not a region would be labelled as high risk for infection of the Zika virus [1].

For this study, a Neural Architecture Search (NAS) by AutoKeras was utilized [10]. It makes use of Bayesian optimization to guide network morphism, a paradigm which preserves the functionality of a neural network while the architecture changes.

3 METHODOLOGY

3.1 Data Collection

The researchers collected epidemiological data for COVID-19 cases from the Department of Health's (DOH) COVID-19 data drop [14]. The dataset contained information on each infected individual, such as: when and where they were tested positive with COVID-19, when they recovered, and if applicable, when they died. The dataset, updated on a daily basis, also contained the number of hospital equipment and staff in each hospital in the Philippines.

Population data for NCR was collected from the Philippines Statistics Authority's (PSA) 2020 population census [5] while the land area data was collected from the PSA's 2015 census [3]. The researchers compiled information from various national news sources and government announcements to create a lockdown restriction dataset. The data includes the 4 main quarantine restrictions that have been implemented in the NCR region, namely: Enhanced Community Quarantine (ECQ), Modified Enhanced Community Quarantine (MECQ), General Community Quarantine (GCQ), and Modified General Community Quarantine (MGCQ) and the dates they were implemented [12].

Finally, the researchers obtained community mobility data from Google's COVID-19 community mobility reports. The dataset showed the increase or decrease in human movement, compared to pre-COVID movement patterns in January 2020 to February 2020, in various area types per region in the Philippines. The values begin from February 2020 and are updated on a daily basis [9].

3.2 Data Preprocessing

The researchers preprocessed the data using Pandas, a Python library widely used to manipulate data in various formats [11]. General preprocessing steps were implemented for all datasets such as: dropping unnecessary columns, correcting date formats, resampling values for missing dates, and sorting.

COVID-19 case counts and hospital equipment were both aggregated by date to create the respective datasets. The final dataset has variables from March 17, 2020 until September 13, 2021. Certain factors in the hospital dataset were derived from multiple columns in the original data, such as how the total hospital beds include beds from COVID and non-COVID patients.

Due to the unavailability of the 2021 Philippine population data, the 2020 census was used instead. Population density was then obtained by dividing the 2020 population size by the given land area in NCR. The values associated with the COVID-19 cases, such as recoveries and deaths, were aggregated per day with the case counts. The number for susceptible individuals or the number of individuals at risk of contracting the disease (S) was derived by taking the total NCR population (P) and subtracting the number of new cases of the disease per day (C), the number of COVID-19-related deaths (D), and recoveries from COVID-19 (R) as shown in Equation 1 below.

$$S = P - (C + D + R) \tag{1}$$

Using the susceptible counts, the researchers were able to calculate the incidence rate of a disease (I) using the PSA's incidence rate formula [4]. The incidence rate is the rate at which new cases of a certain diseases occur in a population and is usually expressed as the number of cases per 100,000 persons, as shown in Equation 2.

$$I = \frac{C}{S} \times 100,000 \tag{2}$$

Finally, the researchers integer-encoded the categorical quarantine levels and assigned different numerical values based on how restrictive the quarantines were in terms of movement. Table 2 contains each lockdown type with its corresponding integer value.

Lockdown Type	Integer Value
ECQ	4
MECQ	3
GCQ	2
MGCQ	1

Table 2: Integer-encoded values of each lockdown restriction

3.3 Machine Learning Model

The machine learning models were implemented using Skforecast, a Python library that eases using Scikit-Learn regressors as multi-step forecasters [13]. For their comparison model, the researchers chose to use an extended version of a Random Forest (RF) model using Scikit-Learn's RandomForestRegressor. Two sets of models were created, a set of "base" models trained only on COVID-19 confirmed case counts on Table 3, and a set of "risk" models trained using multivariate data in the same table. Ten-fold cross-validation was applied to determine the average performance of the models.

Model type	Features
Base models	Case counts
Risk models	Case counts Population size and density Incidence rates Susceptible counts Recovered counts Hospital equipment data Lockdown restriction data Community mobility data

Table 3: Feature inputs for Base and Risk models

To obtain the optimal hyperparameters for Base and Risk RF models, Skforecast's `grid_search_forecaster` was used. The function implements the validation using time series backtesting, which assess the mean squared error (MSE) of a forecast using the existing data. Table 4 presents the hyperparamaters used in the RF models.

	Base RF model	Risk RF model
Lags	7	7
Max Depth	3	10
n estimators	100	100

Table 4: Random Forest model hyperparameters

Each RF model in the cross-validation was made to generate a 14-day forecast for September 14–27, 2021, and plotted against the actual cases for that time span. The root mean squared error (RMSE) of each model was taken, and the average and standard deviation of each iteration was noted.

3.4 Neural Network Model

The creation of the neural network models was done using the Neural Architecture Search (NAS) tool, AutoKeras [10]. The researchers utilized the `TimeSeriesForecaster` of AutoKeras to create the Base and Risk NAS model sets. The NAS was configured to test up to a maximum of 10 Keras architectures, and to train each proposed architecture over 10 epochs, with the objective of minimizing validation loss per epoch.

Similar to the RF models, ten-fold cross-validation was applied to determine the average performance of the NAS models. Each NAS model iteration generated a 14-day forecast for September 14–27, 2021. The forecasts were checked against the actual case counts, and the RMSE for each model forecast was calculated.

4 RESULTS AND DISCUSSION

The Base RF model's average RMSE greatly outperforms the Risk RF model's RMSE; however, it was evident that the Risk RF models were greatly underfit to the training data. It could be noted that the addition of risk factors likely introduced variance and noise to the model training rather than improving the forecasts of the model. Table 5 presents the RMSE values for the Base and Risk RF models. The best performing models have been highlighted. Figures 1 and 2 present the plots of the Base and Risk RF models, respectively.

Iterations	RMSE	
	Base RF model	Risk RF model
1	1089.53	**943.45**
2	1072.29	1020.34
3	1077.51	1294.87
4	1062.57	1189.91
5	1060.46	976.16
6	1043.76	1115.44
7	**940.52**	1007.79
8	1043.48	1012.81
9	1084.77	1031.88
10	1074.60	1139.52
Average	1054.95	1073.22
Standard Deviation	43.10	109.35

Table 5: RF RMSE Results

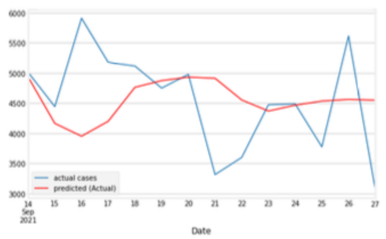


Figure 1: Base RF model plot

In terms of average RMSE, the Risk NAS model performed better than the Base NAS model. Overall, the best performing model was a Risk NAS model with an RMSE of 760.37 as seen in Table 6.

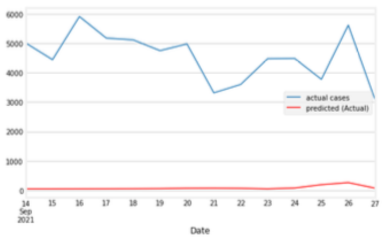


Figure 2: Risk RF model plot

Iterations	RMSE	
	Base NAS model	**Risk NAS model**
1	3331.49	2804.50
2	2770.08	2717.69
3	3437.61	3693.92
4	2693.09	2677.63
5	3044.04	2478.57
6	2921.51	2692.26
7	**1526.51**	2221.06
8	2823.96	2802.28
9	3170.72	**925.75**
10	2944.63	3248.23
Average	2867.26	2626.19
Standard Deviation	528.27	721.31

Table 6: NAS RMSE Results

When comparing the average RMSE's of the RF and NAS models, it was observed that the Base RF models performed better than the Base NAS models, while Risk NAS models performed better than the Risk RF models. It should be noted that despite the Base RF models having more consistent RMSE's, the risk NAS models' best iteration had the best overall RMSE and followed the actual COVID-19 cases better. This can be seen in Figure 4 compared to the Base RF model's best iteration in Figure 1.

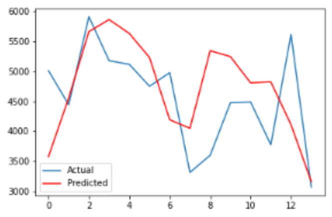


Figure 3: Base NAS model plot

Similarly, the NAS models had much higher standard deviations compared to the RF models. However, the researchers observed that the Risk NAS models' standard deviation is much less than the Base NAS models', leading to the observation that the NAS models perform better with multivariate inputs while the RF models perform better with univariate inputs.

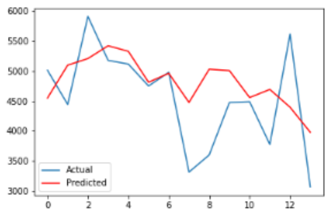


Figure 4: Risk NAS model plot

5 CONCLUSION AND FUTURE WORK

The research aimed to compare univariate and multivariate machine learning approaches in forecasting COVID-19 cases. It also aimed to compare the performance of classical modeling approaches and state-of-the-art neural networks given socioeconomic risk factors that can affect the greater transmission of the virus.

The researchers can conclude that when forecasting using only case data, RF models performed better than NAS models. However, when using both the case data and the risk factor data, NAS models performed better than RF models. Overall, considering risk factors in the training of forecasting models can improve performance and provide better forecasts of COVID-19 spread in the future.

For future work, the researchers would like to use feature selection methods to determine which risk factors have the greatest positive contributions to the models' predictions. They also aim to evaluate models on other metrics, such as Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), to better understand which model performs the best. Finally, different NAS configurations could be tested, with different epochs, objectives, and maximum Keras architectures.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Jann Railey E. Montalan for their continuous support, guidance, and advisement throughout the research. The authors would also like to thank Mr. John Noel P. Victorino and Dr. John Paul C. Vergara for their feedback and suggestions with regards to the improvement of the research.

The authors would also like to thank the Department of Information Systems and Computer Science (DISCS) and the Ateneo Social Computing and Science Laboratory (ASCSL) of the Ateneo de Manila University (ADMU).

REFERENCES

[1] Mahmood Akhtar, Moritz UG Kraemer, and Lauren M Gardner. 2019. A dynamic neural network model for predicting risk of Zika in real time. *BMC medicine* 17, 1 (2019), 1–16.
[2] Faith Argosino. [n.d.]. *COVID-19 response: A timeline of community quarantine, lockdowns, alert levels.* https://mb.com.ph/2021/11/09/covid-19-response-a-timeline-of-community-quarantine-lockdowns-alert-levels/ Manila Bulletin. Retrieved from https://mb.com.ph/2021/11/09/covid-19-response-a-timeline-of-community-quarantine-lockdowns-alert-levels/.
[3] Philippine Statistics Authority. 2016. Philippine population density (based on the 2015 census of population). *Philippine Statistics Authority.[Google Scholar]* (2016). https://psa.gov.ph/content/philippine-population-density-based-2015-census-population
[4] Philippine Statistics Authority. 2017. Philippine Statistics Authority Incidence Rate (of a disease). *Philippine Statistics Authority.[Google Scholar]* (2017). https://psa.gov.ph/content/incidence-rate-disease-2

[5] Philippine Statistics Authority. 2021. 2020 Census of Population and Housing (2020 CPH) Population Counts Declared Official by the President. *Philippine Statistics Authority* (2021). https://psa.gov.ph/content/2020-census-population-and-housing-2020-cph-population-counts-declared-official-president

[6] Zlatan Car, Sandi Baressi Segota, Nikola Andelic, Ivan Lorencin, and Vedran Mrzljak. 2020. Modeling the spread of COVID-19 infection using a multilayer perceptron. *Computational and mathematical methods in medicine* 2020 (2020).

[7] Guido David, Ranjit Singh Rye, and Ma. Patricia Agbulos. 2020. COVID-19 forecasts in the Philippines: Insights for policy-making. https://up.edu.ph/covid-19-forecasts-in-the-philippines-insights-for-policy-making/

[8] Joseph Galasso, Duy M Cao, and Robert Hochberg. 2022. A random forest model for forecasting regional COVID-19 cases utilizing reproduction number estimates and demographic data. *Chaos, Solitons & Fractals* (2022), 111779.

[9] Google. [n.d.]. *Google COVID-19 Community Mobility Reports.* https://www.google.com/covid19/mobility/ Retrieved from https://www.google.com/covid19/mobility/.

[10] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1946–1956.

[11] Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, Vol. 445. Austin, TX, 51–56.

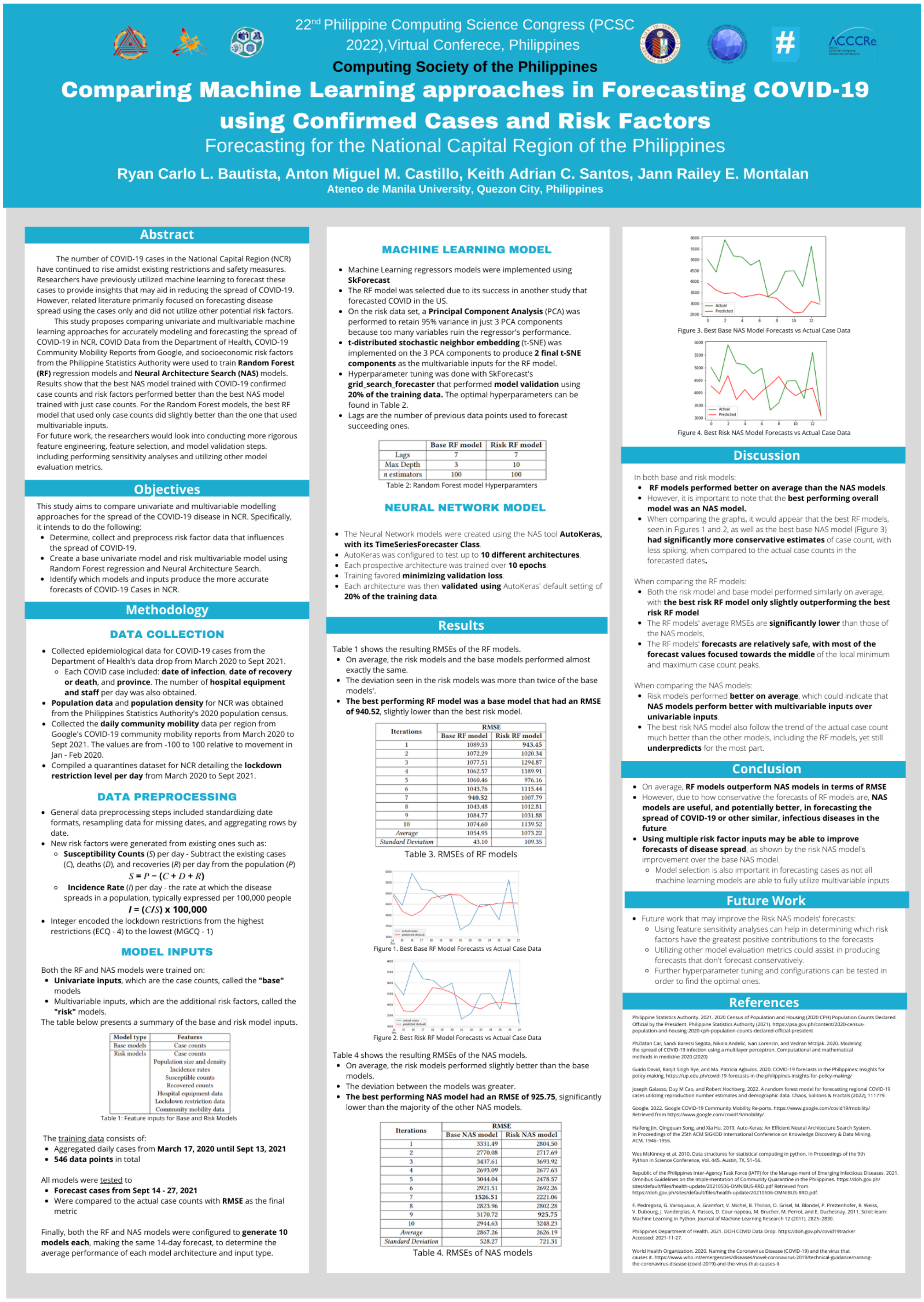
[12] Republic of the Philippines Inter-Agency Task Force (IATF) for the Management of Emerging Infectious Diseases. [n.d.]. *Omnibus Guidelines on the Implementation of Community Quarantine in the Philippines.* https://doh.gov.ph/sites/default/files/health-update/20210506-OMNIBUS-RRD.pdf Retrieved from https://doh.gov.ph/sites/default/files/health-update/20210506-OMNIBUS-RRD.pdf.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[14] Philippines Department of Health. 2021. DOH COVID Data Drop. https://doh.gov.ph/covid19tracker Accessed: 2021-11-27.

[15] World Health Organization. [n.d.]. Naming the Coronavirus Disease (COVID-19) and the virus that causes it. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it

D.2 Philippine Computing Science Congress (PCSC) 2022 poster



22nd Philippine Computing Science Congress (PCSC 2022),Virtual Conferece, Philippines

Computing Society of the Philippines

Comparing Machine Learning approaches in Forecasting COVID-19 using Confirmed Cases and Risk Factors

Forecasting for the National Capital Region of the Philippines

Ryan Carlo L. Bautista, Anton Miguel M. Castillo, Keith Adrian C. Santos, Jann Railey E. Montalan
Ateneo de Manila University, Quezon City, Philippines

Abstract

The number of COVID-19 cases in the National Capital Region (NCR) have continued to rise amidst existing restrictions and safety measures. Researchers have previously utilized machine learning to forecast these cases to provide insights that may aid in reducing the spread of COVID-19. However, related literature primarily focused on forecasting disease spread using the cases only and did not utilize other potential risk factors.

This study proposes comparing univariate and multivariable machine learning approaches for accurately modeling and forecasting the spread of COVID-19 in NCR. COVID Data from the Department of Health, COVID-19 Community Mobility Reports from Google, and socioeconomic risk factors from the Philippine Statistics Authority were used to train **Random Forest (RF)** regression models and **Neural Architecture Search (NAS)** models. Results show that the best NAS model trained with COVID-19 confirmed case counts and risk factors performed better than the best NAS model trained with just case counts. For the Random Forest models, the best RF model that used only case counts did slightly better than the one that used multivariable inputs.

For future work, the researchers would look into conducting more rigorous feature engineering, feature selection, and model validation steps. including performing sensitivity analyses and utilizing other model evaluation metrics.

Objectives

This study aims to compare univariate and multivariable modelling approaches for the spread of the COVID-19 disease in NCR. Specifically, it intends to do the following:

- Determine, collect and preprocess risk factor data that influences the spread of COVID-19.
- Create a base univariate model and risk multivariable model using Random Forest regression and Neural Architecture Search.
- Identify which models and inputs produce the more accurate forecasts of COVID-19 Cases in NCR.

Methodology

DATA COLLECTION

- Collected epidemiological data for COVID-19 cases from the Department of Health's data drop from March 2020 to Sept 2021.
  - Each COVID case included: **date of infection**, **date of recovery or death**, and **province**. The number of **hospital equipment and staff** per day was also obtained.
- **Population data** and **population density** for NCR was obtained from the Philippines Statistics Authority's 2020 population census.
- Collected the **daily community mobility** data per region from Google's COVID-19 community mobility reports from March 2020 to Sept 2021. The values are from -100 to 100 relative to movement in Jan - Feb 2020.
- Compiled a quarantines dataset for NCR detailing the **lockdown restriction level per day** from March 2020 to Sept 2021.

DATA PREPROCESSING

- General data preprocessing steps included standardizing date formats, resampling data for missing dates, and aggregating rows by date.
- New risk factors were generated from existing ones such as:
  - **Susceptibility Counts** (S) per day - Subtract the existing cases (C), deaths (D), and recoveries (R) per day from the population (P)

$$S = P - (C + D + R)$$

  - **Incidence Rate** (I) per day - the rate at which the disease spreads in a population, typically expressed per 100,000 people

$$I = (C/S) \times 100{,}000$$

- Integer encoded the lockdown restrictions from the highest restrictions (ECQ - 4) to the lowest (MGCQ - 1)

MODEL INPUTS

Both the RF and NAS models were trained on:

- **Univariate inputs**, which are the case counts, called the **"base"** models
- Multivariable inputs, which are the additional risk factors, called the **"risk"** models.

The table below presents a summary of the base and risk model inputs.

Model type	Features
Base models	Case counts
Risk models	Case counts
	Population size and density
	Incidence rates
	Susceptible counts
	Recovered counts
	Hospital equipment data
	Lockdown restriction data
	Community mobility data

Table 1: Feature inputs for Base and Risk Models

The training data consists of:

- Aggregated daily cases from **March 17, 2020 until Sept 13, 2021**
- **546 data points** in total

All models were tested to

- **Forecast cases from Sept 14 - 27, 2021**
- Were compared to the actual case counts with **RMSE** as the final metric

Finally, both the RF and NAS models were configured to **generate 10 models each**, making the same 14-day forecast, to determine the average performance of each model architecture and input type.

MACHINE LEARNING MODEL

- Machine Learning regressors models were implemented using **SkForecast**
- The RF model was selected due to its success in another study that forecasted COVID in the US.
- On the risk data set, a **Principal Component Analysis** (PCA) was performed to retain 95% variance in just 3 PCA components because too many variables ruin the regressor's performance.
- **t-distributed stochastic neighbor embedding** (t-SNE) was implemented on the 3 PCA components to produce **2 final t-SNE components** as the multivariable inputs for the RF model.
- Hyperparameter tuning was done with SkForecast's **grid_search_forecaster** that performed **model validation** using **20% of the training data.** The optimal hyperparameters can be found in Table 2.
- Lags are the number of previous data points used to forecast succeeding ones.

	Base RF model	Risk RF model
Lags	7	7
Max Depth	3	10
n estimators	100	100

Table 2: Random Forest model Hyperparamters

NEURAL NETWORK MODEL

- The Neural Network models were created using the NAS tool **AutoKeras, with its TimeSeriesForecaster Class**.
- AutoKeras was configured to test up to **10 different architectures**.
- Each prospective architecture was trained over **10 epochs**.
- Training favored **minimizing validation loss**.
- Each architecture was then **validated using** AutoKeras' default setting of **20% of the training data**.

Results

Table 1 shows the resulting RMSEs of the RF models.

- On average, the risk models and the base models performed almost exactly the same.
- The deviation seen in the risk models was more than twice of the base models'.
- **The best performing RF model was a base model that had an RMSE of 940.52**, slightly lower than the best risk model.

Iterations	RMSE	
	Base RF model	Risk RF model
1	1089.53	943.45
2	1072.29	1020.34
3	1077.51	1294.87
4	1062.57	1189.91
5	1060.46	976.16
6	1043.76	1115.44
7	**940.52**	1007.79
8	1043.48	1012.81
9	1084.77	1031.88
10	1074.60	1139.52
Average	1054.95	1073.22
Standard Deviation	43.10	109.35

Table 3. RMSEs of RF models

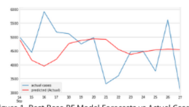


Figure 1. Best Base RF Model Forecasts vs Actual Case Data

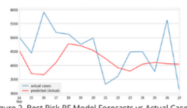


Figure 2. Best Risk RF Model Forecasts vs Actual Case Data

Table 4 shows the resulting RMSEs of the NAS models.

- On average, the risk models performed slightly better than the base models.
- The deviation between the models was greater.
- **The best performing NAS model had an RMSE of 925.75**, significantly lower than the majority of the other NAS models.

Iterations	RMSE	
	Base NAS model	Risk NAS model
1	3331.49	2804.50
2	2770.08	2717.69
3	3437.61	3693.92
4	2693.09	2677.63
5	3044.04	2478.57
6	2921.51	2692.26
7	**1526.51**	2221.06
8	2823.96	2802.28
9	3170.72	**925.75**
10	2944.63	3248.23
Average	2867.26	2626.19
Standard Deviation	528.27	721.31

Table 4. RMSEs of NAS models

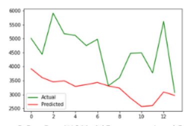


Figure 3. Best Base NAS Model Forecasts vs Actual Case Data

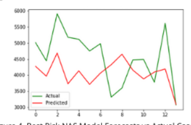


Figure 4. Best Risk NAS Model Forecasts vs Actual Case Data

Discussion

In both base and risk models:

- **RF models performed better on average than the NAS models**.
- However, it is important to note that the **best performing overall model was an NAS model.**
- When comparing the graphs, it would appear that the best RF models, seen in Figures 1 and 2, as well as the best base NAS model (Figure 3) **had significantly more conservative estimates** of case count, with less spiking, when compared to the actual case counts in the forecasted dates**.**

When comparing the RF models:

- Both the risk model and base model performed similarly on average, with **the best risk RF model only slightly outperforming the best risk RF model**
- The RF models' average RMSEs are **significantly lower** than those of the NAS models,
- The RF models' **forecasts are relatively safe, with most of the forecast values focused towards the middle** of the local minimum and maximum case count peaks.

When comparing the NAS models:

- Risk models performed **better on average**, which could indicate that **NAS models perform better with multivariable inputs over univariable inputs**.
- The best risk NAS model also follow the trend of the actual case count much better than the other models, including the RF models, yet still **underpredicts** for the most part.

Conclusion

- On average, **RF models outperform NAS models in terms of RMSE**
- However, due to how conservative the forecasts of RF models are, **NAS models are useful, and potentially better, in forecasting the spread of COVID-19 or other similar, infectious diseases in the future**.
- **Using multiple risk factor inputs may be able to improve forecasts of disease spread**, as shown by the risk NAS model's improvement over the base NAS model.
  - Model selection is also important in forecasting cases as not all machine learning models are able to fully utilize multivariable inputs

Future Work

- Future work that may improve the Risk NAS models' forecasts:
  - Using feature sensitivity analyses can help in determining which risk factors have the greatest positive contributions to the forecasts
  - Utilizing other model evaluation metrics could assist in producing forecasts that don't forecast conservatively.
  - Further hyperparameter tuning and configurations can be tested in order to find the optimal ones.

References

Philippine Statistics Authority. 2021. 2020 Census of Population and Housing (2020 CPH) Population Counts Declared Official by the President. Philippine Statistics Authority (2021). https://psa.gov.ph/content/2020-census-population-and-housing-2020-cph-population-counts-declared-official-president

PhZlatan Car, Sandi Baressi Segota, Nikola Andelic, Ivan Lorencin, and Vedran Mrzljak. 2020. Modeling the spread of COVID-19 infection using a multilayer perceptron. Computational and mathematical methods in medicine 2020 (2020)

Guido David, Ranjit Singh Rye, and Ma. Patricia Agbulos. 2020. COVID-19 forecasts in the Philippines: Insights for policy-making. https://up.edu.ph/covid-19-forecasts-in-the-philippines-insights-for-policy-making/

Joseph Galasso, Duy M Cao, and Robert Hochberg. 2022. A random forest model for forecasting regional COVID-19 cases utilizing reproduction number estimates and demographic data. Chaos, Solitons & Fractals (2022), 111779.

Google. 2022. Google COVID-19 Community Mobility Re-ports. https://www.google.com/covid19/mobility/ Retrieved from https://www.google.com/covid19/mobility/.

Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 1946–1956.

Wes McKinney et al. 2010. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, Vol. 445. Austin, TX, 51–56.

Republic of the Philippines Inter-Agency Task Force (IATF) for the Manage-ment of Emerging Infectious Diseases. 2021. Omnibus Guidelines on the Imple-mentation of Community Quarantine in the Philippines. https://doh.gov.ph/sites/default/files/health-update/20210506-OMNIBUS-RRD.pdf Retrieved from https://doh.gov.ph/sites/default/files/health-update/20210506-OMNIBUS-RRD.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cour-napeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

Philippines Department of Health. 2021. DOH COVID Data Drop. https://doh.gov.ph/covid19tracker Accessed: 2021-11-27.

World Health Organization. 2020. Naming the Coronavirus Disease (COVID-19) and the virus that causes it. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it

APPENDIX E

Code

E.1 Creating the Dataset

Disclaimer: the code provided generates a dataset that includes all Luzon provinces, the NCR dataset is only obtained after the final Luzon dataset is generated.

E.1.1 Imports and Initialization

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
pd.set_option('display.max_rows', None)
```

E.1.2 COVID-19 Cases, Deaths, and Recoveries

```python
df = pd.read_csv("May72022_COVID_Data.csv")
df = df.dropna(subset=['DateRepConf'])
df["DateRepConf"] = pd.to_datetime(df["DateRepConf"], errors='coerce')

# Actual COVID Cases dataframe
df_new = pd.concat([df["CaseCode"], df["DateRepConf"],df["ProvRes"]],
    axis=1)
```

```
df_new = df_new[df_new["ProvRes"].str.contains("NCR|ABRA|ALBAY|APAYAO|
    AURORA|BATAAN|BATANGAS|BENGUET|BULACAN|CAGAYAN|CAMARINES NORTE|
    CAMARINES SUR|CAVITE|IFUGAO|ILOCOS NORTE|ILOCOS SUR|ISABELA|KALINGA|LA
     UNION|LAGUNA|MOUNTAIN PROVINCE|NUEVA ECIJA|NUEVA VIZCAYA|PAMPANGA|
    PANGASINAN|QUEZON|QUIRINO|RIZAL|SORSOGON|TARLAC|ZAMBALES")==True]
cases_agg_df = df_new.groupby(["ProvRes", "DateRepConf"]).agg(['count']).
    sort_values(["ProvRes", "DateRepConf"], ascending = (True, True))
cases_agg_df = cases_agg_df.reset_index()

# Recovery and Death dataframes
df_recov = pd.concat([df["CaseCode"], df["DateRecover"],df["ProvRes"]],
    axis=1)
df_died = pd.concat([df["CaseCode"], df["DateDied"],df["ProvRes"]], axis
    =1)
df_recov.shape + df_died.shape

conditional1 = (df_recov.DateRecover.isnull() != True)
conditional2 = (df_died.DateDied.isnull() != True)
df_recov = df_recov[conditional1]
df_died = df_died[conditional2]

df_recov = df_recov[df_recov["ProvRes"].str.contains("NCR|ABRA|ALBAY|
    APAYAO|AURORA|BATAAN|BATANGAS|BENGUET|BULACAN|CAGAYAN|CAMARINES NORTE|
    CAMARINES SUR|CAVITE|IFUGAO|ILOCOS NORTE|ILOCOS SUR|ISABELA|KALINGA|LA
     UNION|LAGUNA|MOUNTAIN PROVINCE|NUEVA ECIJA|NUEVA VIZCAYA|PAMPANGA|
    PANGASINAN|QUEZON|QUIRINO|RIZAL|SORSOGON|TARLAC|ZAMBALES")==True]
df_died = df_died[df_died["ProvRes"].str.contains("NCR|ABRA|ALBAY|APAYAO|
    AURORA|BATAAN|BATANGAS|BENGUET|BULACAN|CAGAYAN|CAMARINES NORTE|
    CAMARINES SUR|CAVITE|IFUGAO|ILOCOS NORTE|ILOCOS SUR|ISABELA|KALINGA|LA
     UNION|LAGUNA|MOUNTAIN PROVINCE|NUEVA ECIJA|NUEVA VIZCAYA|PAMPANGA|
    PANGASINAN|QUEZON|QUIRINO|RIZAL|SORSOGON|TARLAC|ZAMBALES")==True]

df_recov["DateRecover"] = pd.to_datetime(df_recov["DateRecover"])
df_died["DateDied"] = pd.to_datetime(df_died["DateDied"])

# Aggregating each the number of deaths and recoveries per province then
    per date
df_aggrecov = df_recov.groupby(["ProvRes", "DateRecover"]).agg(['count'])
    .sort_values(["ProvRes", "DateRecover"], ascending = (True, True))
df_aggrecov = df_aggrecov.reset_index()

df_aggredied = df_died.groupby(["ProvRes", "DateDied"]).agg(['count']).
    sort_values(["ProvRes", "DateDied"], ascending = (True, True))
df_aggredied = df_aggredied.reset_index()
```

E.1.3 Community Mobility

```
df_commob = pd.read_csv("ph_commobility.csv")
df_commob = df_commob.drop(columns=['country_region_code', '
    country_region', 'iso_3166_2_code', 'census_fips_code', 'place_id'])
df_commob["date"] = pd.to_datetime(df_commob["date"], dayfirst=True,
    errors='coerce')

# renaming for shorter column names
```

```
df_commob.rename(columns={'retail_and_recreation_percent_change_from_baseline': 'retail_rec_baseline',
                          'grocery_and_pharmacy_percent_change_from_baseline': 'grocery_pharma_baseline',
                          'parks_percent_change_from_baseline': 'parks_baseline',
                          'transit_stations_percent_change_from_baseline': 'transit_baseline',
                          'workplaces_percent_change_from_baseline': 'workplace_baseline',
                          'residential_percent_change_from_baseline': 'residental_baseline'}, inplace=True)

# changing to NCR to make it uniform
df_commob.loc[df_commob['sub_region_1'].str.contains('National Capital Region', case=False, na=False), 'sub_region_1'] = 'NCR'
df_commob = df_commob[(df_commob['date'] > "2020-3-16") & (df_commob['date'] < "2022-5-8")].reset_index(drop=True)
df_commob = df_commob.sort_values(['sub_region_1', 'date'], ascending=(True, True)).reset_index(drop=True)
```

E.1.4 Hospital Equipment

```
hospitals_df = pd.read_csv("DOH_Hospitals.csv")
hospitals_df["reportdate"] = pd.to_datetime(hospitals_df["reportdate"],
    dayfirst=True, errors='coerce')
hospitals_df["isolation_beds"] = hospitals_df["isolbed_v"] + hospitals_df
    ["isolbed_o"]
hospitals_df["ward_beds"] = hospitals_df["beds_ward_v"] + hospitals_df["
    beds_ward_o"]

new_hosp_df = pd.concat([hospitals_df["reportdate"], hospitals_df["
    province"], hospitals_df["isolation_beds"], hospitals_df["ward_beds"
    ]],  axis=1)
new_hosp_df = new_hosp_df[new_hosp_df["province"].str.contains("NCR|ABRA|
    ALBAY|APAYAO|AURORA|BATAAN|BATANGAS|BENGUET|BULACAN|CAGAYAN|CAMARINES
    NORTE|CAMARINES SUR|CAVITE|IFUGAO|ILOCOS NORTE|ILOCOS SUR|ISABELA|
    KALINGA|LA UNION|LAGUNA|MOUNTAIN PROVINCE|NUEVA ECIJA|NUEVA VIZCAYA|
    PAMPANGA|PANGASINAN|QUEZON|QUIRINO|RIZAL|SORSOGON|TARLAC|ZAMBALES")==
    True]
# rename to NCR for uniformity
new_hosp_df.loc[new_hosp_df['province'].str.contains('NCR', case=False),
    'province'] = 'NCR'

new_hosp_df = new_hosp_df[(new_hosp_df['reportdate'] > "2020-03-16") & (
    new_hosp_df['reportdate'] < "2022-05-08")].reset_index(drop=True)
new_hosp_df = new_hosp_df.groupby(["province", "reportdate"]).agg(['count
    ']).sort_values(["province", "reportdate"], ascending = (True, True))
new_hosp_df = new_hosp_df.reset_index()
```

E.1.5 PH Lockdowns

```
lockdowns_df = pd.read_csv("PH_lockdowns_cleanest.csv")
lockdowns_df["Start Date"] = pd.to_datetime(lockdowns_df["Start Date"])
lockdowns_df["End Date"] = pd.to_datetime(lockdowns_df["End Date"])
lockdowns_df.rename(columns={'Start Date': 'Start_Date', 'End Date': '
    End_Date', 'Quarantine Type': 'Quarantine_Type'}, inplace=True)
#populate the data given 2 date ranges
compiled_df = pd.DataFrame(columns = ["Province", "Quarantine_Type"])

for row in lockdowns_df.itertuples():
  holder_df = pd.DataFrame(columns = ["Province", "Quarantine_Type", "
    Start_Date", "End_Date"])
  holder_df.loc[holder_df.shape[0]] = list(row)[1:]

  idx = pd.date_range(holder_df.iloc[0]["Start_Date"], holder_df.iloc[0][
    "End_Date"])
  holder_df.set_index(holder_df.Start_Date, inplace=True)
  holder_df = holder_df.resample('D').sum().reindex(idx)
  holder_df['Date'] = holder_df.index
  holder_df = holder_df.reset_index(drop=True)

  holder_df["Province"] = holder_df.iloc[0]["Province"]
  holder_df["Quarantine_Type"] = holder_df.iloc[0]["Quarantine_Type"]

  if row.Index == 0:
    compiled_df = holder_df
  else:
    compiled_df = compiled_df.append(holder_df, ignore_index=True)
```

```

# integer encoding the quarantine values from highest movement
    restrictions to lowest
compiled_df = compiled_df.replace({'Quarantine_Type': {"ECQ": 4, "MECQ":
    3, "GCQ": 2, "MGCQ": 1}})
compiled_df["Quarantine_Type"] = pd.to_numeric(compiled_df["
    Quarantine_Type"])
```

E.1.6 Incidence Rates

```
incidences_df = pd.read_csv("Luzon_Incidence_Rates_PopDens_2020.csv")
incidences_df["Start_Date"] = pd.to_datetime(incidences_df["Start_Date"])
incidences_df["End_Date"] = pd.to_datetime(incidences_df["End_Date"])

populated_incidences_df = pd.DataFrame(columns = ["Province", "
    Population_Size"])
#populate the data given 2 date ranges
for row in incidences_df.itertuples():
  holder_df = pd.DataFrame(columns = ["Province", "Population_Size", "
    Population_Density", "Start_Date", "End_Date"])
  holder_df.loc[holder_df.shape[0]] = list(row)[1:]

  idx = pd.date_range(holder_df.iloc[0]["Start_Date"], holder_df.iloc[0][
    "End_Date"])
  holder_df.set_index(holder_df.Start_Date, inplace=True)
  holder_df = holder_df.resample('D').sum().reindex(idx)
  holder_df['Date'] = holder_df.index
  holder_df = holder_df.reset_index(drop=True)

  holder_df["Province"] = holder_df.iloc[0]["Province"]
  holder_df["Population_Size"] = holder_df.iloc[0]["Population_Size"]
  holder_df["Population_Density"] = holder_df.iloc[0]["Population_Density"]

  if row.Index == 0:
    populated_incidences_df = holder_df
  else:
    populated_incidences_df = populated_incidences_df.append(holder_df, ignore_index=True)
```

E.1.7 Compiling the Final Dataset

```
small_provs = ["ABRA", "ALBAY", "APAYAO", "AURORA", "BATAAN", "BATANGAS", "BENGUET", "BULACAN", "CAGAYAN", "CAMARINES NORTE", "CAMARINES SUR", "CAVITE", "IFUGAO", "ILOCOS NORTE", "ILOCOS SUR",
              "ISABELA", "KALINGA", "LA UNION", "LAGUNA", "MOUNTAIN PROVINCE", "NCR", "NUEVA ECIJA", "NUEVA VIZCAYA", "PAMPANGA", "PANGASINAN", "QUEZON", "QUIRINO", "RIZAL", "SORSOGON", "TARLAC", "ZAMBALES"]

complete_provinces_data_df = pd.DataFrame(columns = ["Province"])

for i in small_provs:
  prov_name = i
```

```
prov_idx = pd.date_range('03-17-2020', '05-07-2022')

#cases section
cases_holder_df = cases_agg_df.loc[cases_agg_df['ProvRes'].str.contains
  (prov_name)].copy().reset_index(drop=True)
cases_holder_df.set_index(cases_holder_df.DateRepConf, inplace=True)
cases_holder_df = cases_holder_df.resample('D').sum().reindex(prov_idx)
  .fillna(0)

#recoveries section
recover_holder_df = df_aggrecov.loc[df_aggrecov['ProvRes'].str.contains
  (prov_name)].copy().reset_index(drop=True)
recover_holder_df.set_index(recover_holder_df.DateRecover, inplace=True
  )
recover_holder_df = recover_holder_df.resample('D').sum().reindex(
  prov_idx).fillna(0)

#deaths section
death_holder_df = df_aggredied.loc[df_aggredied['ProvRes'].str.contains
  (prov_name)].copy().reset_index(drop=True)
death_holder_df.set_index(death_holder_df.DateDied, inplace=True)
death_holder_df = death_holder_df.resample('D').sum().reindex(prov_idx)
  .fillna(0)

#hospital section
hosp_holder_df = new_hosp_df.loc[new_hosp_df['province'].str.contains(
  prov_name)].copy()
```

```
hosp_holder_df.set_index(hosp_holder_df.reportdate, inplace=True)
hosp_holder_df = hosp_holder_df.resample('D').sum().reindex(prov_idx).
    fillna(0)
hosp_holder_df.insert(loc=0, column='Province', value=prov_name)

#lockdown section
lock_holder_df = compiled_df.loc[compiled_df['Province'].str.contains(
    prov_name)].copy()
lock_holder_df.set_index(lock_holder_df.Date, inplace=True)

#incidence rates section
incs_holder_df = populated_incidences_df.loc[populated_incidences_df['
    Province'].str.contains(prov_name)].copy()
incs_holder_df.set_index(incs_holder_df.Date, inplace=True)

#community mobility section
if prov_name in "ALBAY|CAMARINES NORTE|CAMARINES SUR|SORSOGON":
    com_mob_df = df_commob.loc[df_commob['sub_region_1'].str.contains('
    Bicol', na=False)].copy()
elif prov_name in "CAGAYAN|ISABELA|NUEVA VIZCAYA|QUIRINO":
    com_mob_df = df_commob.loc[df_commob['sub_region_1'].str.contains('
    Cagayan Valley', na=False)].copy()
elif prov_name in "BATANGAS|CAVITE|LAGUNA|QUEZON|RIZAL":
    com_mob_df = df_commob.loc[df_commob['sub_region_1'].str.contains('
    Calabarzon', na=False)].copy()
elif prov_name in "AURORA|BATAAN|BULACAN|NUEVA ECIJA|PAMPANGA|TARLAC|
    ZAMBALES":
```

```
    com_mob_df = df_commob.loc[df_commob[’sub_region_1’].str.contains(’Central Luzon’, na=False)].copy()
elif prov_name in "ABRA|APAYAO|BENGUET|IFUGAO|KALINGA|MOUNTAIN PROVINCE":
    com_mob_df = df_commob.loc[df_commob[’sub_region_1’].str.contains(’Cordillera Administrative Region’, na=False)].copy()
elif prov_name in "ILOCOS NORTE|ILOCOS SUR|LA UNION|PANGASINAN":
    com_mob_df = df_commob.loc[df_commob[’sub_region_1’].str.contains(’Ilocos Region’, na=False)].copy()
elif prov_name in "NCR":
    com_mob_df = df_commob.loc[df_commob[’sub_region_1’].str.contains(’NCR’, na=False)].copy()

com_mob_df.set_index(com_mob_df.date, inplace=True)
com_mob_df = com_mob_df.resample(’D’).sum().reindex(prov_idx).fillna(0)

# Hosp Holder df is the dataframe everything else is appended to
hosp_holder_df["covid_counts"] = cases_holder_df["CaseCode"]
hosp_holder_df["recovery_counts"] = recover_holder_df["CaseCode"]
hosp_holder_df["death_counts"] = death_holder_df["CaseCode"]
hosp_holder_df["population_size"] = incs_holder_df["Population_Size"]
hosp_holder_df["incidence_rate"] = ((hosp_holder_df["covid_counts"]/incs_holder_df["Population_Size"]) * 100000).round(2)
hosp_holder_df["quarantine_type_int"] = lock_holder_df["Quarantine_Type"]

hosp_holder_df["retail_rec_baseline"] = com_mob_df["retail_rec_baseline
```

```
    "]
  hosp_holder_df["grocery_pharma_baseline"] = com_mob_df["
    grocery_pharma_baseline"]
  hosp_holder_df["parks_baseline"] = com_mob_df["parks_baseline"]
  hosp_holder_df["transit_baseline"] = com_mob_df["transit_baseline"]
  hosp_holder_df["workplace_baseline"] = com_mob_df["workplace_baseline"]
  hosp_holder_df["residental_baseline"] = com_mob_df["residental_baseline
    "]

  hosp_holder_df.insert(loc=0, column='Date', value = hosp_holder_df.
    index)

  if i == "ABRA":
    complete_provinces_data_df = hosp_holder_df
  else:
    complete_provinces_data_df = complete_provinces_data_df.append(
    hosp_holder_df, ignore_index=True)

# insert susceptible counts after the DF is made bec computations can't
    be made while it's being modified
complete_provinces_data_df.insert(
    loc = 8,
    column = 'susceptible_counts',
    value = complete_provinces_data_df["population_size"] - (
    complete_provinces_data_df["covid_counts"] +
    complete_provinces_data_df["recovery_counts"] +
    complete_provinces_data_df["death_counts"]))
```

```

# generate the NCR only dataset
ncr_may72022 = complete_provinces_data_df[complete_provinces_data_df["
    Province"] == "NCR"].copy()
ncr_may72022.drop(['Province', 'population_size'], axis=1, inplace = True
    )
ncr_may72022.to_csv('ncr_may72022_slice.csv', index=False)
```

E.2 Random Forest Models

E.2.1 Imports and Initialization

```
## Data manipulation
import numpy as np
import pandas as pd
# Plots
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.rcParams['lines.linewidth'] = 1.5
%matplotlib inline
get_ipython().run_line_magic('matplotlib', 'inline')
# Warnings configuration
import warnings
warnings.filterwarnings('ignore')
import sklearn
# Modeling and Forecasting
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn import preprocessing
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from skforecast.ForecasterAutoregCustom import ForecasterAutoregCustom
from skforecast.ForecasterAutoregMultiOutput import
    ForecasterAutoregMultiOutput
from skforecast.model_selection import grid_search_forecaster
from skforecast.model_selection import backtesting_forecaster
from joblib import dump, load
# Evaluation
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_absolute_error
#Comparison Functions

def measure_model_success(realVals, predictedVals):
    rmse = round(mean_squared_error(realVals, predictedVals, squared=
    False), 2)
    print ("RMSE: {0}".format(rmse))
    mape = round(mean_absolute_percentage_error(realVals, predictedVals)
    * 100, 2)
    print ("MAPE: {0}".format(mape))
    mae = round(mean_absolute_error(realVals, predictedVals), 2)
    print ("MAE: {0}".format(mae))

def measure_model_RMSE(realVals, predictedVals):
    rmse = round(mean_squared_error(realVals, predictedVals, squared=
   False), 2)
    return rmse
    print ("RMSE: {0}".format(rmse))
#random generation of seeds
import random as rand

def randomSeed(how_many):
    seeds = []
    for i in range(0,how_many):
        n = rand.randint(0,9999)
        seeds.append(int(n))
    return seeds
```

E.2.2 Data Imports and PCA / TSNE

```
data = pd.read_csv("ncr_may72022_slice.csv")
data['Date'] = pd.to_datetime(data['Date'], format='%m/%d/%Y')
# get all dates for training before Sept 14, 2021
data = data[data['Date'] < "2021-9-14"].reset_index(drop=True)
data = data.set_index('Date')
data = data.asfreq('D')

# removing the covid_counts for PCA / TSNE in a new dataframe = data 2
```

```
data2 = data.copy()
data2.drop(['covid_counts'], axis=1, inplace=True)
data2

# this is the TEST dataset containing the ACTUAL cases from Sept 14-27,
    2021
actual_cases_df = pd.read_csv("sept1427_caseonly.csv")
actual_cases_df['Date'] = pd.to_datetime(actual_cases_df['Date'], format=
    '%m/%d/%Y')
actual_cases_df = actual_cases_df.set_index('Date')
actual_cases_df = actual_cases_df.asfreq('D')

#PCA Functions
scaler = preprocessing.MinMaxScaler()
names = data2.columns
d = scaler.fit_transform(data2)
normed_df = pd.DataFrame(d, columns=names)

from sklearn.decomposition import PCA
pca = PCA(n_components=3, random_state=3058)
principalComponents = pca.fit_transform(normed_df)

#TSNE Functions
from sklearn.manifold import TSNE
tsne = TSNE(random_state=3058)
X_tsne = tsne.fit_transform(principalComponents)
tsne_df = pd.DataFrame(data = X_tsne, columns = ['tsne_1', 'tsne_2'])
```

```
tsne_df.index = data.index
tsne_df["covid_counts"] = data["covid_counts"]

# comment out based on the use, data = univariate , tsne_df =
    multivariable
#data_train = data
data_train = tsne_df
```

E.2.3 RF Initial Modelling and Hyperparameter Tuning

```
forecaster = ForecasterAutoreg(
                regressor = RandomForestRegressor(random_state=6935),
                lags      = 14
# This value will be replaced in the grid search
             )
# hyperparameter tuning function, these lists are the possible
    configurations
param_grid = {'n_estimators': [20, 50, 100, 150],
              'max_depth': [3, 5, 10, 15]}
lags_grid = [7, 14, 30, 60]
results_grid = grid_search_forecaster(
                        forecaster  = forecaster,
                        y           = data_train['covid_counts'],
# comment the exog out if doing univariate but leave it if multivariate
                        exog        = data_train[['tsne_1', 'tsne_2']],
                        param_grid  = param_grid,
                        lags_grid   = lags_grid,
                        steps        = 14,
                        refit        = True,
                        metric       = 'mean_absolute_percentage_error',
                        initial_train_size = int(len(data_train)*0.50),
                        return_best = True,
                        verbose      = False
                    )
# get the feature importance
forecaster.get_feature_importance()

# Predictions
# multivariable
predictions = forecaster.predict(steps=14, exog=data_train[['tsne_1', '
    tsne_2']])
#Univariate
#predictions = forecaster.predict(steps=14)

# Ploting the model predictions vs the actual predictions
fig, ax=plt.subplots(figsize=(8, 5))
actual_cases_df['covid_counts'].plot(ax=ax, label='actual cases', color='
    tab:blue')
predictions.plot(ax=ax, label='predicted (Actual)', color = 'red')
ax.set_facecolor('xkcd:white')
plt.title("RF TSNE Seed 3058, RF Seed 5996, 14 Day Forecast")
ax.legend();
measure_model_success(actual_cases_df['covid_counts'], predictions)
```

E.2.4 10 Fold Cross Validation of RF models

```python
# generates 10 random seeds
seed1 = randomSeed(10)

def generate_models_with_seeds(seed_list):

    seed_numbers = []
    rmse_list = []
    mape_list = []
    mae_list = []

    for chosen_seed in seed_list:

        # comment this out until the data_train part if univariate
        from sklearn.manifold import TSNE
        tsne = TSNE(random_state=3058)
        X_tsne = tsne.fit_transform(principalComponents)

        tsne_df = pd.DataFrame(data = X_tsne, columns = ['tsne_1', '
    tsne_2'])
        tsne_df.index = data.index
        tsne_df["covid_counts"] = data["covid_counts"]

        data_train = tsne_df

        #forecaster
        forecaster = ForecasterAutoreg(
                 regressor = RandomForestRegressor(random_state=chosen_seed),
                 lags      = 14 # This value will be replaced in the grid search
              )

    #hyperparameter inputs - manually input the optimal parameters obtained from the previous tuner to save compute time
    param_grid = {'n_estimators': [20],
                  'max_depth': [15]}

    lags_grid = [30]

    results_grid = grid_search_forecaster(
                              forecaster   = forecaster,
                              y            = data_train['covid_counts'],
                              # comment the exog out if doing univariate but leave it if multivariate
                              exog         = data_train[['tsne_1', 'tsne_2']],
                              param_grid   = param_grid,
                              lags_grid    = lags_grid,
                              steps        = 14,
                              refit        = True,
                              metric       = 'mean_squared_error',
                              initial_train_size = int(len(data_train)*0.50),
                                return_best = True,
                                verbose     = False
                            )

        #Multivariable
        predictions = forecaster.predict(steps=14, exog=data_train[['tsne_1', 'tsne_2']])

        #Univariate
#         predictions = forecaster.predict(steps=14)

        # plotting
        fig, ax=plt.subplots(figsize=(8, 5))
        actual_cases_df['covid_counts'].plot(ax=ax, label='actual cases', color='tab:blue')
        predictions.plot(ax=ax, label='predicted (Actual)', color = 'red')
        ax.set_facecolor('xkcd:white')
        plt.title("RF Risk, TSNE Seed 3058, Seed {}, 14 Day Forecast".format(chosen_seed))
        plt.savefig("results/{0}_RF30_Risk_Seed {1}".format(seed_list.index(chosen_seed), chosen_seed))
        ax.legend();

        #add to the list
        seed_numbers.append(chosen_seed)
        rmse_list.append(measure_model_RMSE(actual_cases_df['covid_counts
'], predictions))
        mape_list.append(round(mean_absolute_percentage_error(actual_cases_df['covid_counts'], predictions) * 100, 2))
        mae_list.append(round(mean_absolute_error(actual_cases_df['covid_counts'], predictions), 2))


    list_joiner = []
    list_joiner.extend([seed_numbers, rmse_list, mape_list, mae_list])

    return list_joiner

# calls the function and the output are the 10 metrics of the 10 random models
first_attempt = generate_models_with_seeds(seed1)

# save the Seed number and evaluation metrics in a CSV
one = first_attempt[0]
two = first_attempt[1]
three = first_attempt[2]
four = first_attempt[3]

dictionary_TSNE = {'Seed': one, 'RMSE':two, 'MAPE': three, 'MAE':four}
df_dict = pd.DataFrame(dictionary_TSNE)
df_dict.to_csv("results/RF_Risk_30Days_2.csv", index = False)
```

E.3 Neural Architecture Search Models

E.3.1 Imports and Initialization

```python
#running on Python 3.8.12
import tensorflow as tf
import numpy as np
import pandas as pd
import autokeras as ak
import statistics as stat
import random as rand
import keras_tuner as kt

from numpy import asarray, hstack, array

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential, clone_model, load_model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.metrics import MeanAbsoluteError
from tensorflow.keras.metrics import MeanAbsolutePercentageError

from matplotlib import pyplot as plt

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error

from pandas import Series
```

```
src = r'.\data\ncr_may72022.csv'
dataset = pd.read_csv(src)
dataset = dataset.drop(0)
actual = pd.read_csv(r'.\data\ncr_may72022_casesonly.csv')
actual['Date'] = pd.to_datetime(actual['Date'],format = '%m/%d/%Y')
```

E.3.2 Function Definition

```
def reshape(series):
    series = series.reshape((len(series),1))
    return series
```

```
def scale(series):
    scaler = MinMaxScaler(feature_range = (0,1))
    series = scaler.fit_transform(series)
    return series
```

```
def split_sequences(sequences, n_steps_in, n_steps_out):
    X, y = list(), list()
    for i in range(len(sequences)):
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out-1
        if out_end_ix > len(sequences):
            break
        seq_x, seq_y = sequences[i:end_ix, :-1],sequences[end_ix-1:
    out_end_ix, -1]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
```

```
def split_y(sequences, n_steps_in, n_steps_out):
    y = list()
    for i in range(len(sequences)):
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out-1
        if out_end_ix > len(sequences):
            break
        seq_y = sequences[end_ix-1:out_end_ix, -1]
        y.append(seq_y)
    return array(y)
```

```
def akTSFModel(seed, lookback):
    clf = ak.TimeseriesForecaster(
        lookback=lookback,
        max_trials=10,
        metrics=[tf.keras.metrics.RootMeanSquaredError(), tf.keras.
    metrics.MeanAbsoluteError(), tf.keras.metrics.
    MeanAbsolutePercentageError()],
        overwrite = True,
        loss = 'mse',
        seed = seed,
    )
    return clf
```

```
def plot(predictions, actual):
    plt.plot(actual, color = 'green')
    plt.plot(predictions, color = 'red')
    plt.title('Risk Model vs Actual')
    plt.legend(['Actual','Predicted'])
    plt.savefig('data/risk/' +str(i)+'.png')
    plt.clf()
```

E.3.3 Data Reshaping

```
#WHEN USING THIS FOR BASE MODEL, COMMENT OUT EVERYTHING EXCEPT covscale,
    AND covid_counts.
#Split dataset
covscale = dataset['covid_counts'].astype('float64')
covid_counts = dataset['covid_counts'].astype('float64')
isolation_beds = dataset['isolation_beds'].astype('float64')
susceptible = dataset['susceptible_counts'].astype('float64')
recovery = dataset['recovery_counts'].astype('float64')
death = dataset['death_counts'].astype('float64')
incidence_rate = dataset['incidence_rate'].astype('float64')
quarantine_type_int = dataset['quarantine_type_int'].astype('float64')
retail_rec = dataset['retail_rec_baseline'].astype('float64')
grocery_pharma = dataset['grocery_pharma_baseline'].astype('float64')
parks = dataset['parks_baseline'].astype('float64')
transit = dataset['transit_baseline'].astype('float64')
```

```
workplace = dataset['workplace_baseline'].astype('float64')
residential = dataset['residential_baseline'].astype('float64')
```

```
#Get values of each column
covscale = covscale.values
covid_counts = covid_counts.values
susceptible = susceptible.values
recovery = recovery.values
death = death.values
incidence_rate = incidence_rate.values
quarantine_type_int = quarantine_type_int.values
retail_rec = retail_rec.values
grocery_pharma = grocery_pharma.values
parks = parks.values
transit = transit.values
workplace = workplace.values
residential = residential.values
isolation_beds = isolation_beds.values
```

```
#reshape all features.
covscale = reshape(covscale)
covid_counts = reshape(covid_counts)
susceptible = reshape(susceptible)
recovery = reshape(recovery)
death = reshape(death)
incidence_rate = reshape(incidence_rate)
quarantine_type_int = reshape(quarantine_type_int)
retail_rec = reshape(retail_rec)
```

```
grocery_pharma = reshape(grocery_pharma)
parks = reshape(parks)
transit = reshape(transit)
workplace = reshape(workplace)
residential = reshape(residential)
isolation_beds = reshape(isolation_beds)
```

```
#normalize
covid_counts = scale(covid_counts)
susceptible = scale(susceptible)
recovery = scale(recovery)
death = scale(death)
incidence_rate = scale(incidence_rate)
quarantine_type_int = scale(quarantine_type_int)
retail_rec = scale(retail_rec)
grocery_pharma = scale(grocery_pharma)
parks = scale(parks)
transit = scale(transit)
workplace = scale(workplace)
residential = scale(residential)
isolation_beds = scale(isolation_beds)
```

```
#prepare inverse scaler
scaler = MinMaxScaler(feature_range = (0,1))
scaler.fit(covscale)
```

```
#stack columns horizontally
#use basedata if running base model.
#basedata = np.expand_dims(dataset_stacked, 1)
```

```
dataset_stackedcov = hstack((covid_counts, susceptible, recovery, death,
    incidence_rate, quarantine_type_int, retail_rec, grocery_pharma, parks
    , transit, workplace, residential, isolation_beds, covid_counts))
dataset_stacked = hstack((covid_counts, susceptible, recovery, death,
    incidence_rate, quarantine_type_int, retail_rec, grocery_pharma, parks
    , transit, workplace, residential, isolation_beds))
```

```
#arrange x to (samples, features), arrange y to (samples, timesteps)
#FOR BASE MODEL, REPLACE dataset_stackedcov WITH basedata.
lookback, predict = 14,14
X, throw = split_sequences(dataset_stackedcov, lookback, predict)
y = split_y(covid_counts, lookback, predict)
```

E.3.4 Modelling Proper

```
#546 used here, because that is index of September 14, 2021. End of the
    training and start of testing.

seeds = rand.sample(range(100000),10)
print(seeds)
RMSE = {}
MAE = {}
MAPE = {}
for i in seeds:
    try:
        clf = akTSFModel(i, lookback)
        clf.fit(x=train_X[:546,:], y=train_y[:546], batch_size = lookback
    , epochs=10)
```

```python
    except Exception as e:
        print(e)
    model = clf.export_model()
    try:
        model.save("final models/autokeras_risk" + str(i), save_format="tf")
    except Exception:
        model.save("final models/autokeras_risk" + str(i) + ".h5")

    #prediction on last data value
    X_test2 = (X[546:547,:])
    predictions = model.predict(X_test2)
    #inverse scaling
    unscaled_predictions = scaler.inverse_transform(predictions)
    unscaled_predictions = unscaled_predictions[0]

    RMSE[str(i)] = [str(mean_squared_error(actual['covid_counts'][546:(546+predict)], unscaled_predictions, squared = False))]
    MAE[str(i)] = [str(mean_absolute_error(actual['covid_counts'][546:(546+predict)], unscaled_predictions))]
    MAPE[str(i)] = [str(mean_absolute_percentage_error(actual['covid_counts'][546:(546+predict)], unscaled_predictions))]

    #plotting
    to_plot = actual['covid_counts'][546:(546+predict)]
    to_plot = to_plot.reset_index(drop = True)
    plot(unscaled_predictions, to_plot)

    #metrics into dataframe to csv
    metricsRMSE = pd.DataFrame(RMSE)
    metricsMAE = pd.DataFrame(MAE)
    metricsMAPE = pd.DataFrame(MAPE)

metricsRMSE.to_csv('data/risk/Risk RMSEs.csv')
metricsMAE.to_csv('data/risk/Risk MAEs.csv')
metricsMAPE.to_csv('data/risk/Risk MAPEs.csv')
```