Forecasting of Coronavirus (COVID-19) Cases Through the Process of Mathematical Simulation

Zalven Lemuel Dayao, Dean Michezson Espinosa, Benja Carlo Soliman, Maico Angelo Gutierrez, Christian Alvin Gomez, and Adrian Parco

Faculty, Department of Computer Science, College of Computer Science, New Era University
Quezon City, Philippines
jtcarpio@neu.edu.ph

College of Computer Science, New Era University
Quezon City, Philippines
zalven.dayao@neu.edu.ph
dean.espinosa@neu.edu.ph
benja.soliman@neu.edu.ph
cafgomez@neu.edu.ph
maico.gutierrez@neu.edu.ph
adrian.parco@neu.edu.ph

Abstract

The purpose of this study is to forecast the future cases of Coronavirus (COVID-19). To accomplish the objective of the study, the researchers used the Time Series model to track the movement of the chosen data points over a specified period of time with data points recorded at regular intervals. Python Programming is the main tool to show the results based on the data gathered, mainly regarding the number of people in the Philippines who got infected by the said sickness. The data used on this study came from the World Health Organization where daily updates are available such as the number of infected, cured, deceased and other information related to the virus. After the process of data cleaning and using only the information needed, the graph shows not only the current number of infected, but also the trend of the graph that shows the increase of coronavirus cases on the near future.

Keywords

COVID-19, Time Series, Forecasting, Prophet, pandemic

1. Introduction

COVID-19, also known as severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), is an infectious illness caused by the Coronavirus. The disease was initially discovered in China's Hubei Province's capital, Wuhan district, in December 2019, and has since spread throughout the world, becoming a worldwide pandemic. COVID-19 instances have been increasing since it was discovered in the Philippines. Acute respiratory distress syndrome (ARDS) causes respiratory failure in patients, which is the leading cause of mortality. During this pandemic, healthcare workers are one of the most impacted groups.

The first confirmed COVID-19 case in the Philippines was identified last January 30, 2020. Since this kind of disease is recent upon human discovery, thousands of people died due to unknown cure at the early days of the pandemic. Through the use of data sets that shows the recent cases and other related information pertaining to COVID-19 in the Philippines, the researchers plan to use different kinds of algorithm to proceed with the simulation. Based on the patterns through virtual simulation, we will then be knowledgeable whether the cases of coronavirus will further increase or decrease on the near future. This simulation project is also flexible because not only it can be used to see the future cases of COVID-19, it may also be used to predict the cases for different sicknesses and other important matters such as banking to reduce fraud, measure market risk, and identify opportunities. Overall, the goal is to create a predictive system with high accuracy to ensure the validity of testing certain topic.

2. METHODOLOGY

To start, the researchers needed to collect data that shows the number of people who got infected and who died daily. The researchers got their data from the World Health Organization. Cleaning of data was done to remove information that wasn't needed such as cases from other countries and other data which isn't needed to predict the future cases regarding the Coronavirus Cases.

2.1 Time Series

The algorithm used by the researchers is Time Series. A time series is a sequence of data points that occur in successive order over some period of time. This can be contrasted with cross-sectional data, which captures a point-in-time. In investing, a time series tracks the movement of the chosen data points, such as a security's price, over a specified period of time with data points recorded at regular intervals. There is no minimum or maximum amount of time that must be included, allowing the data to be gathered in a way that provides the information being sought by the investor or analyst examining the activity.

We use a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

g(t): piecewise linear or logistic growth curve for modelling non-periodic changes in time series
s(t): periodic changes (e.g., weekly/yearly seasonality)
h(t): effects of holidays (user provided) with irregular schedules
ε_t: error term accounts for any unusual changes not accommodated by the model

A time series can be taken on any variable that changes over time. In investing, it is common to use a time series to track the price of a security over time. This can be tracked over the short term, such as the price of a security on the hour over the course of a business day, or the long term, such as the price of a security at close on the last day of every month over the course of five years. Time series analysis can be useful to see how a given asset, security, or economic variable changes over time. It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period. Time series is also used in several non-financial contexts, such as measuring the change in population over time. The figure below depicts such a time series for the growth of the U.S. population over the century from 1900-2000.

2.2 Auto-regressive Integrated Moving Average (ARIMA)

In every time series analysis, the forecast is solely based on the past values of the series called lags. A simple overview of a model that depends on one lag or one variable in the series is depicted in the equation below.

$$Y_t = \omega + \varphi \times Y_{t-1} + e_t$$

Here, the predicted value, Y_t, depends on the previous prediction Y_{t-1} and the error e_t calculated as the difference between the predicted and actual outcome. φ is the slope coefficient, and ω is the nonzero mean. The ARIMA model is one of the models which directly doesn't depend on the lags or variables of the times series but depend on the error lags which is estimated by subtracting the actual outcome from the forecasted outcome. ARIMA models assume a linear correlation between the time series values and attempt to exploit these linear dependencies in observations, in order to extract local patterns, while removing high-frequency noise from the data.

2.3 Long short-term memory

Time series prediction problems are a difficult type of predictive modeling problem. Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables. A powerful type of neural network designed to handle sequence dependence is called recurrent neural networks. The Long Short-Term Memory network or LSTM network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained.

The researchers use long short-term memory or LSTM because it can be used to create large recurrent networks the in turn can be used to address difficult sequence problems in machine learning and achieve state of the art results. Instead of neurons, LSTM networks have memory blocks that are connected through layers. A block has components that it smarter than a classical neuron and a memory for recent sequences. A block contains gates that manage the block’s state and output. A block operates upon an input sequence and each gate within a block uses the sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block conditional.

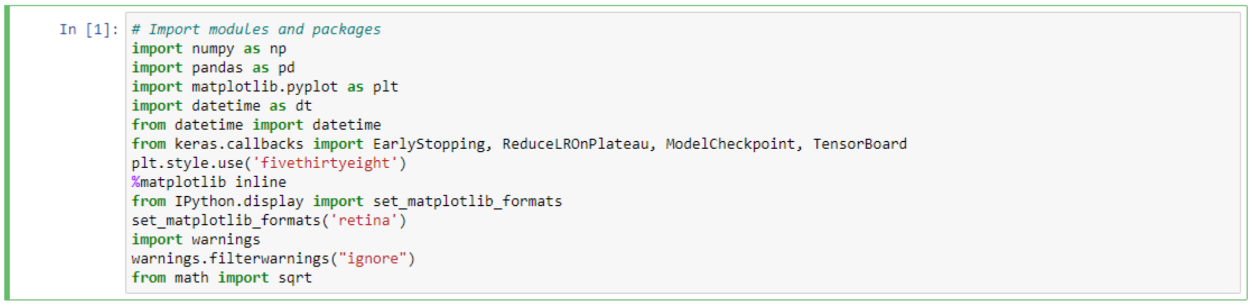
2.4 Prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

3. RESULT DISCUSSION

This part of the research consists of graph of data. The data graph was based on the data set provided by the World Health Organization. And by the use of ARIMA LSTM and Prophet the researchers were able to forecast the time series data.

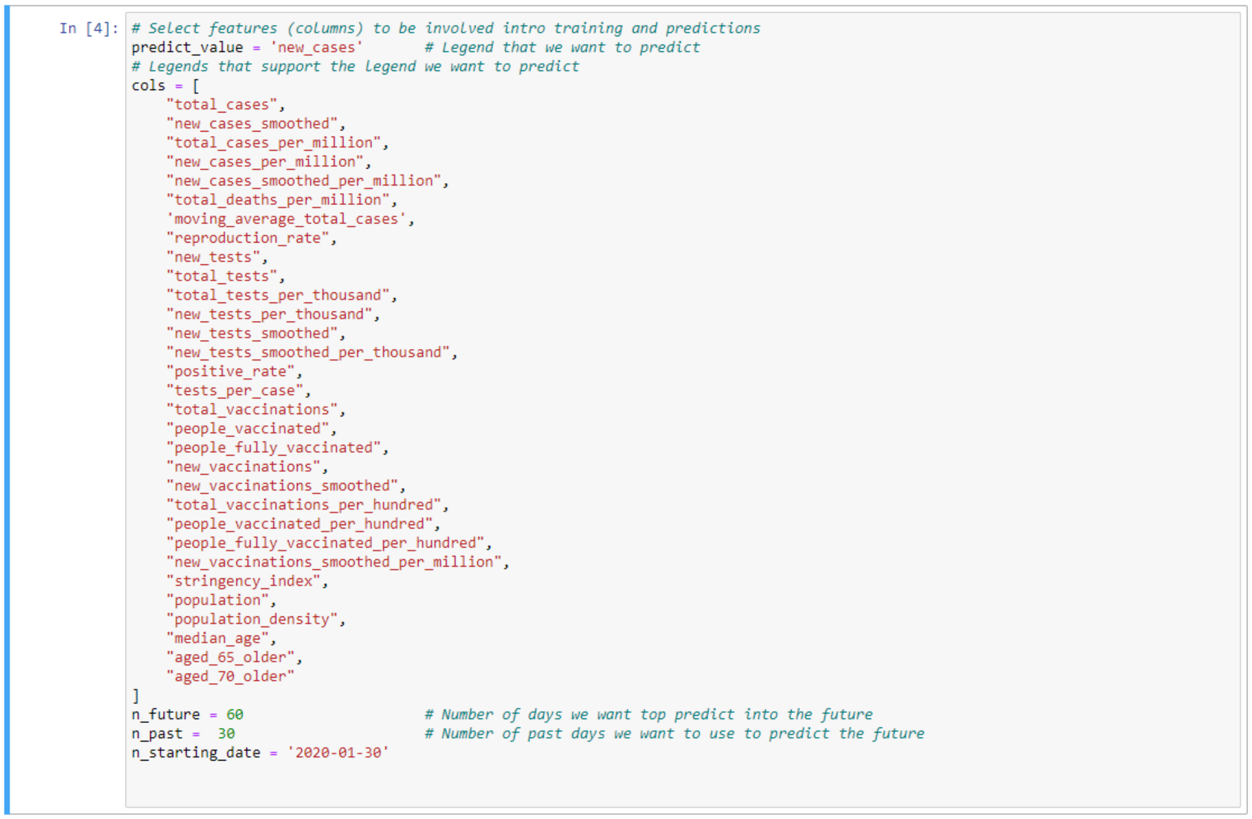
3.1 LSTM



```
In [1]: # Import modules and packages
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import datetime as dt
        from datetime import datetime
        from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, TensorBoard
        plt.style.use('fivethirtyeight')
        %matplotlib inline
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('retina')
        import warnings
        warnings.filterwarnings("ignore")
        from math import sqrt
```

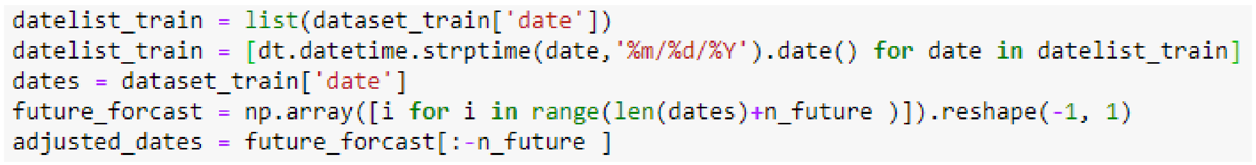
Figure 1. Import Libraries

Before we get started, here in figure 1 we import all of the functions and classes we intend to use. This assumes a working SciPy environment with the Keras deep learning library installed.



```
In [4]: # Select features (columns) to be involved intro training and predictions
        predict_value = 'new_cases'       # Legend that we want to predict
        # Legends that support the legend we want to predict
        cols = [
            "total_cases",
            "new_cases_smoothed",
            "total_cases_per_million",
            "new_cases_per_million",
            "new_cases_smoothed_per_million",
            "total_deaths_per_million",
            'moving_average_total_cases',
            "reproduction_rate",
            "new_tests",
            "total_tests",
            "total_tests_per_thousand",
            "new_tests_per_thousand",
            "new_tests_smoothed",
            "new_tests_smoothed_per_thousand",
            "positive_rate",
            "tests_per_case",
            "total_vaccinations",
            "people_vaccinated",
            "people_fully_vaccinated",
            "new_vaccinations",
            "new_vaccinations_smoothed",
            "total_vaccinations_per_hundred",
            "people_vaccinated_per_hundred",
            "people_fully_vaccinated_per_hundred",
            "new_vaccinations_smoothed_per_million",
            "stringency_index",
            "population",
            "population_density",
            "median_age",
            "aged_65_older",
            "aged_70_older"
        ]
        n_future = 60                     # Number of days we want top predict into the future
        n_past =  30                      # Number of past days we want to use to predict the future
        n_starting_date = '2020-01-30'
```

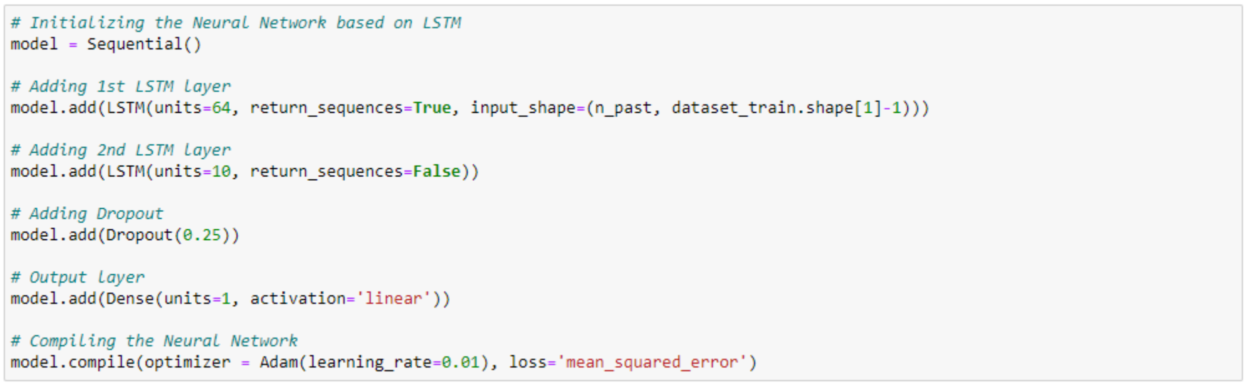
Figure 2. All necessary data from the csv file.



```
datelist_train = list(dataset_train['date'])
datelist_train = [dt.datetime.strptime(date,'%m/%d/%Y').date() for date in datelist_train]
dates = dataset_train['date']
future_forcast = np.array([i for i in range(len(dates)+n_future )]).reshape(-1, 1)
adjusted_dates = future_forcast[:-n_future ]
```

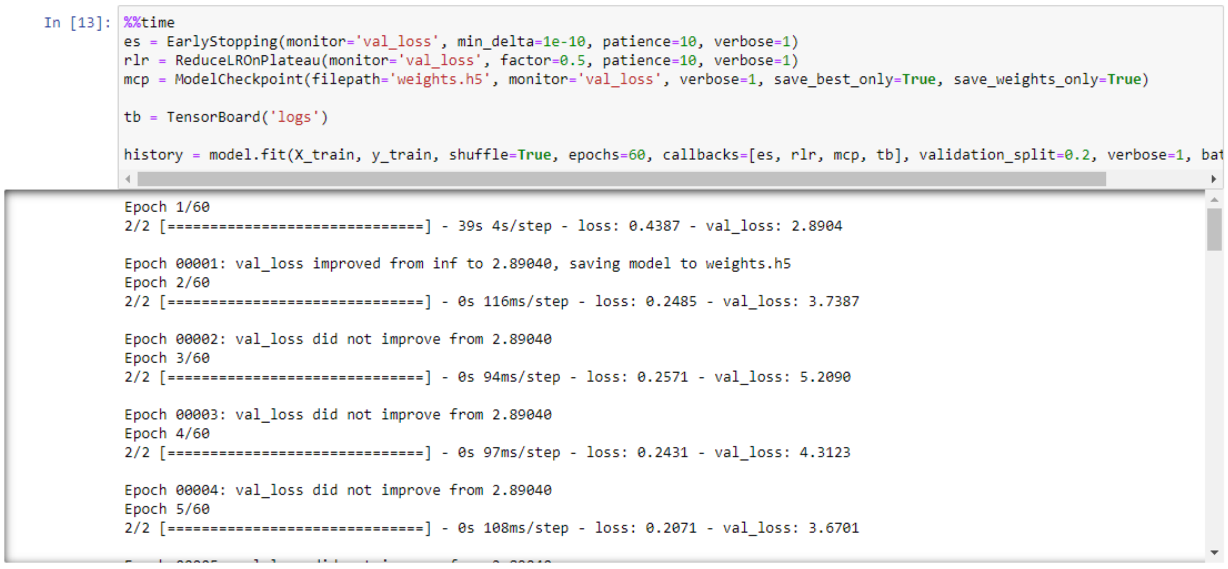
Figure 3. Extract dates

In figure 2 we import all the required libraries to the model and read the original data csv file. load all necessary data and select columns to be involved into the training and predictions. In Figure 3 we extract dates and time stamps that will be use in visualization



```
# Initializing the Neural Network based on LSTM
model = Sequential()

# Adding 1st LSTM layer
model.add(LSTM(units=64, return_sequences=True, input_shape=(n_past, dataset_train.shape[1]-1)))

# Adding 2nd LSTM layer
model.add(LSTM(units=10, return_sequences=False))

# Adding Dropout
model.add(Dropout(0.25))

# Output layer
model.add(Dense(units=1, activation='linear'))

# Compiling the Neural Network
model.compile(optimizer = Adam(learning_rate=0.01), loss='mean_squared_error')
```

Figure 4. LSTM Training Model



```
In [13]: %%time
         es = EarlyStopping(monitor='val_loss', min_delta=1e-10, patience=10, verbose=1)
         rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10, verbose=1)
         mcp = ModelCheckpoint(filepath='weights.h5', monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=True)

         tb = TensorBoard('logs')

         history = model.fit(X_train, y_train, shuffle=True, epochs=60, callbacks=[es, rlr, mcp, tb], validation_split=0.2, verbose=1, bat
```

```
Epoch 1/60
2/2 [==============================] - 39s 4s/step - loss: 0.4387 - val_loss: 2.8904

Epoch 00001: val_loss improved from inf to 2.89040, saving model to weights.h5
Epoch 2/60
2/2 [==============================] - 0s 116ms/step - loss: 0.2485 - val_loss: 3.7387

Epoch 00002: val_loss did not improve from 2.89040
Epoch 3/60
2/2 [==============================] - 0s 94ms/step - loss: 0.2571 - val_loss: 5.2090

Epoch 00003: val_loss did not improve from 2.89040
Epoch 4/60
2/2 [==============================] - 0s 97ms/step - loss: 0.2431 - val_loss: 4.3123

Epoch 00004: val_loss did not improve from 2.89040
Epoch 5/60
2/2 [==============================] - 0s 108ms/step - loss: 0.2071 - val_loss: 3.6701
```

Figure 5. Start training

In figure 4 we setup LSTM training model environment using Keras deep learning libraries we divide the data set into (D) into train (Dtrain) and test (Dtest) sets and figure 5 start training the data combining both Dtrain¹ and Dtest¹, we get a new dataset D¹. Train a classifier on Dtrain¹ and get the accuracy on test data Dtest¹. If the accuracy we get on test data is low, we can conclude that both train and test distributions are similar. Let me elaborate in concise way, when will accuracy be low in this case? If the predicted class label for test data (actual label is 0) is 1. That means if the model is predicting the test data to be label 1, they are actually similar to train data. If the accuracy we get on test data is medium, we can conclude that both train and test distributions are not very similar. If the accuracy we get on test data is high, we can conclude that both train and test distributions are not similar.

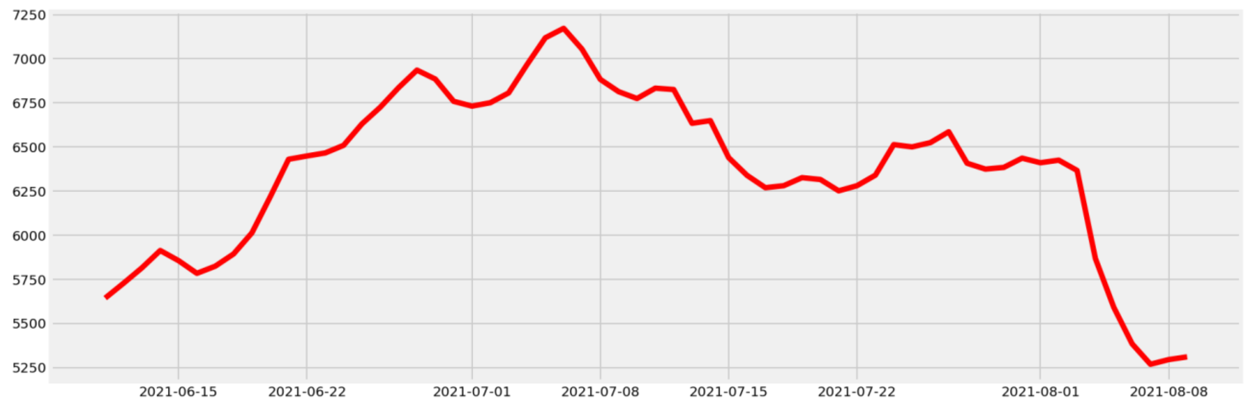
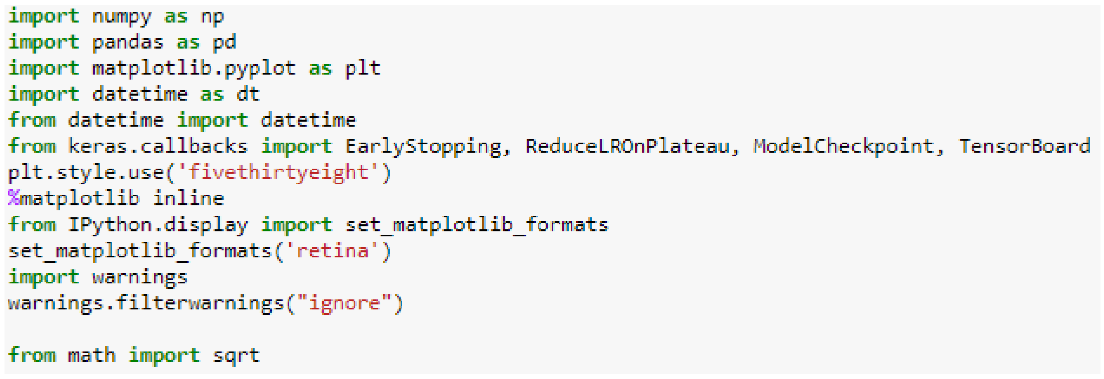


Figure 6. Prediction Result Graph

Figure 6 is the prediction result of covid-19 new cases from 06/15/2021 up to 08/06/2021.

3.2 ARIMA

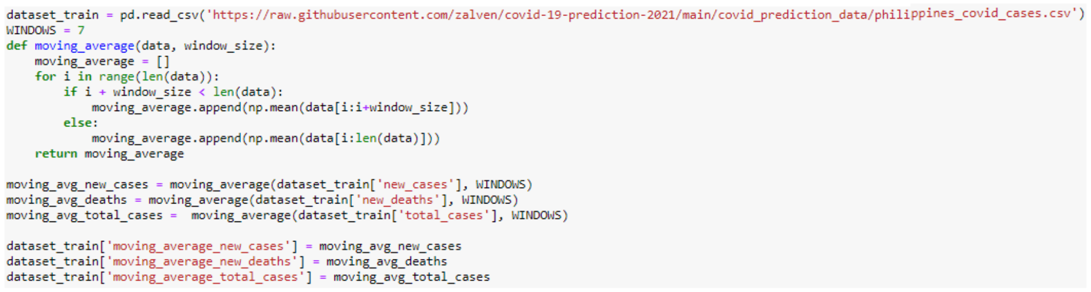
This part of the simulation will show the results after using ARIMA (Autoregressive Integrated Moving Average) to forecast the future coronavirus cases for the next sixty (60) days.



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
from datetime import datetime
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, TensorBoard
plt.style.use('fivethirtyeight')
%matplotlib inline
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('retina')
import warnings
warnings.filterwarnings("ignore")

from math import sqrt
```

Figure 7. Importing of Libraries

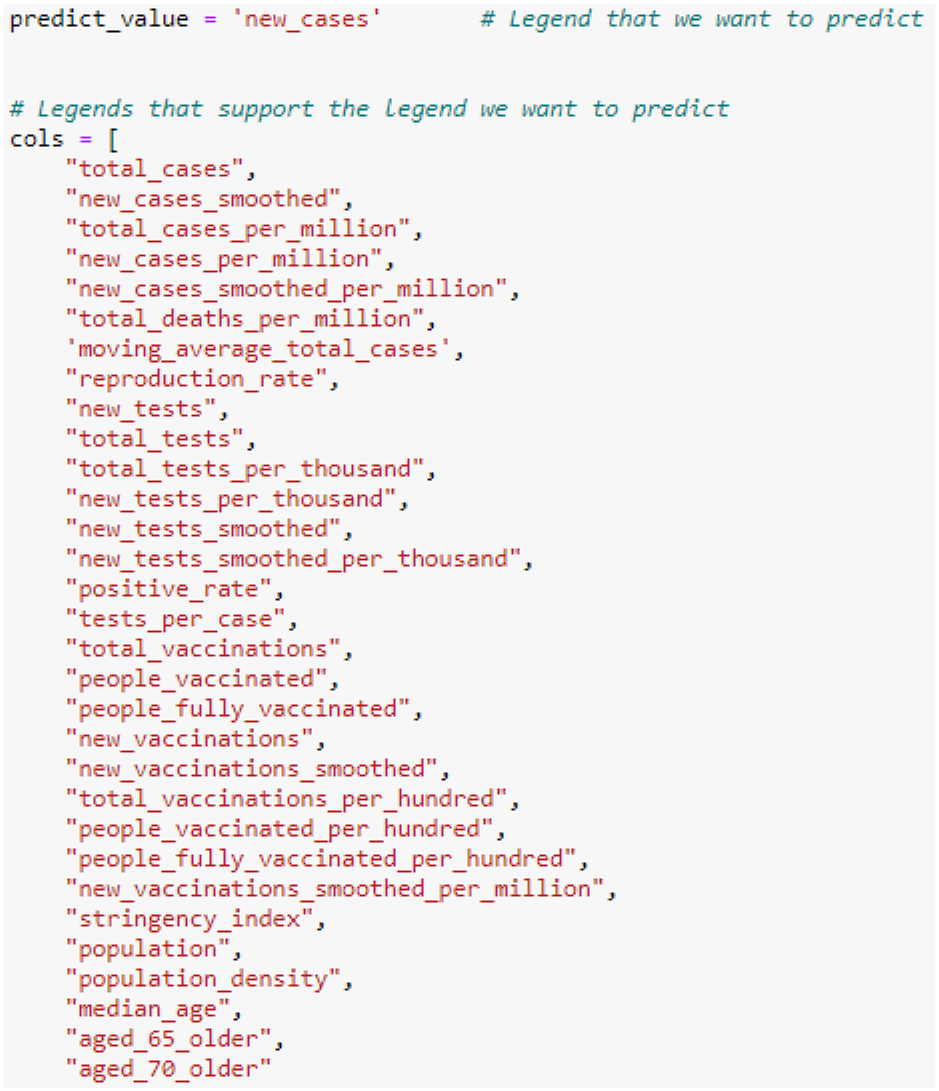
Figure 7 shows the start of data pre-processing. This was done by importing libraries to fulfill the needs of the study.



```python
dataset_train = pd.read_csv('https://raw.githubusercontent.com/zalven/covid-19-prediction-2021/main/covid_prediction_data/philippines_covid_cases.csv')
WINDOWS = 7
def moving_average(data, window_size):
    moving_average = []
    for i in range(len(data)):
        if i + window_size < len(data):
            moving_average.append(np.mean(data[i:i+window_size]))
        else:
            moving_average.append(np.mean(data[i:len(data)]))
    return moving_average

moving_avg_new_cases = moving_average(dataset_train['new_cases'], WINDOWS)
moving_avg_deaths = moving_average(dataset_train['new_deaths'], WINDOWS)
moving_avg_total_cases =  moving_average(dataset_train['total_cases'], WINDOWS)

dataset_train['moving_average_new_cases'] = moving_avg_new_cases
dataset_train['moving_average_new_deaths'] = moving_avg_deaths
dataset_train['moving_average_total_cases'] = moving_avg_total_cases
```

Figure 8. Data reading

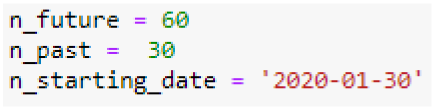
The next figure shows the reading of data related to coronavirus, which is provided by the World Health Organization (WHO).



```python
predict_value = 'new_cases'       # Legend that we want to predict


# Legends that support the legend we want to predict
cols = [
    "total_cases",
    "new_cases_smoothed",
    "total_cases_per_million",
    "new_cases_per_million",
    "new_cases_smoothed_per_million",
    "total_deaths_per_million",
    'moving_average_total_cases',
    "reproduction_rate",
    "new_tests",
    "total_tests",
    "total_tests_per_thousand",
    "new_tests_per_thousand",
    "new_tests_smoothed",
    "new_tests_smoothed_per_thousand",
    "positive_rate",
    "tests_per_case",
    "total_vaccinations",
    "people_vaccinated",
    "people_fully_vaccinated",
    "new_vaccinations",
    "new_vaccinations_smoothed",
    "total_vaccinations_per_hundred",
    "people_vaccinated_per_hundred",
    "people_fully_vaccinated_per_hundred",
    "new_vaccinations_smoothed_per_million",
    "stringency_index",
    "population",
    "population_density",
    "median_age",
    "aged_65_older",
    "aged_70_older"
```

Figure 9. Data cleaning

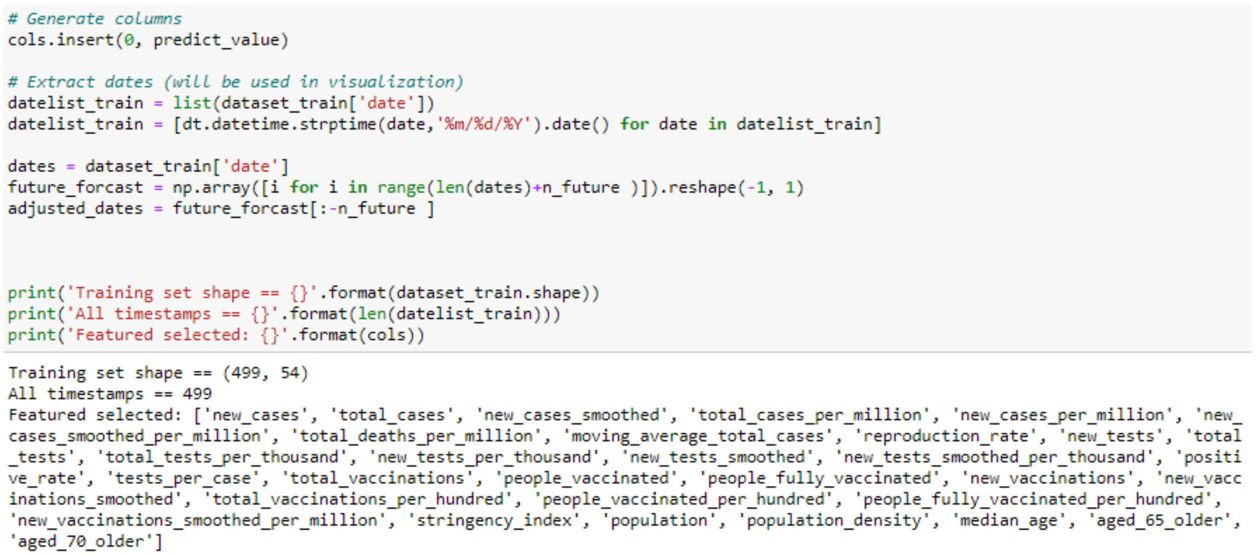
Data cleaning was done to collect the only information needed to be involved in data training and predictions, such as the cases specifically in the Philippines. And the figure above shows the columns that are to be included upon simulation.



```python
n_future = 60
n_past =  30
n_starting_date = '2020-01-30'
```

Figure 10. Days to Predict

Since ARIMA predicts future values based on past values, the researchers used data from the past thirty (30) days in order to forecast the future cases for the next sixty (60) days.

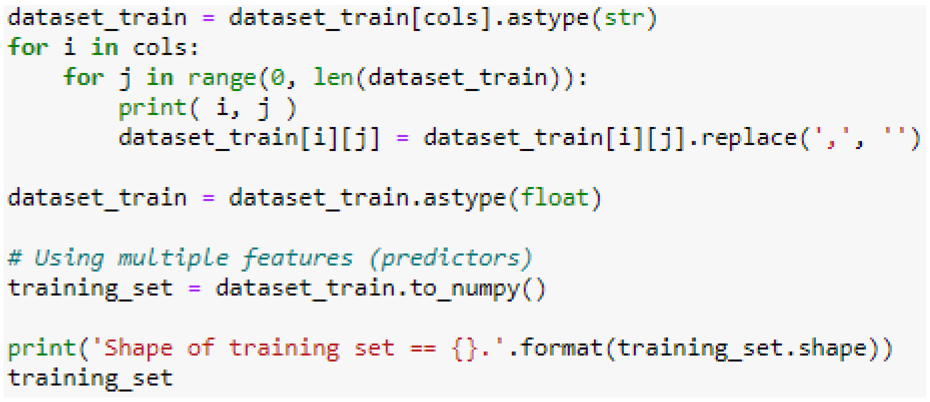


```
# Generate columns
cols.insert(0, predict_value)

# Extract dates (will be used in visualization)
datelist_train = list(dataset_train['date'])
datelist_train = [dt.datetime.strptime(date,'%m/%d/%Y').date() for date in datelist_train]

dates = dataset_train['date']
future_forcast = np.array([i for i in range(len(dates)+n_future )]).reshape(-1, 1)
adjusted_dates = future_forcast[:-n_future ]



print('Training set shape == {}'.format(dataset_train.shape))
print('All timestamps == {}'.format(len(datelist_train)))
print('Featured selected: {}'.format(cols))
```

```
Training set shape == (499, 54)
All timestamps == 499
Featured selected: ['new_cases', 'total_cases', 'new_cases_smoothed', 'total_cases_per_million', 'new_cases_per_million', 'new_
cases_smoothed_per_million', 'total_deaths_per_million', 'moving_average_total_cases', 'reproduction_rate', 'new_tests', 'total
_tests', 'total_tests_per_thousand', 'new_tests_per_thousand', 'new_tests_smoothed', 'new_tests_smoothed_per_thousand', 'positi
ve_rate', 'tests_per_case', 'total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated', 'new_vaccinations', 'new_vacc
inations_smoothed', 'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred',
'new_vaccinations_smoothed_per_million', 'stringency_index', 'population', 'population_density', 'median_age', 'aged_65_older',
'aged_70_older']
```

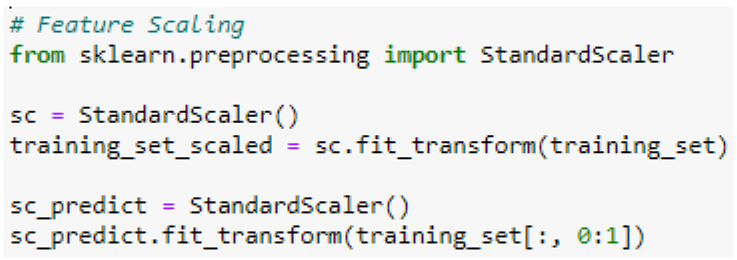
Figure 11. Important variables and dates

Generating the columns and extracting of dates is shown on figure above.



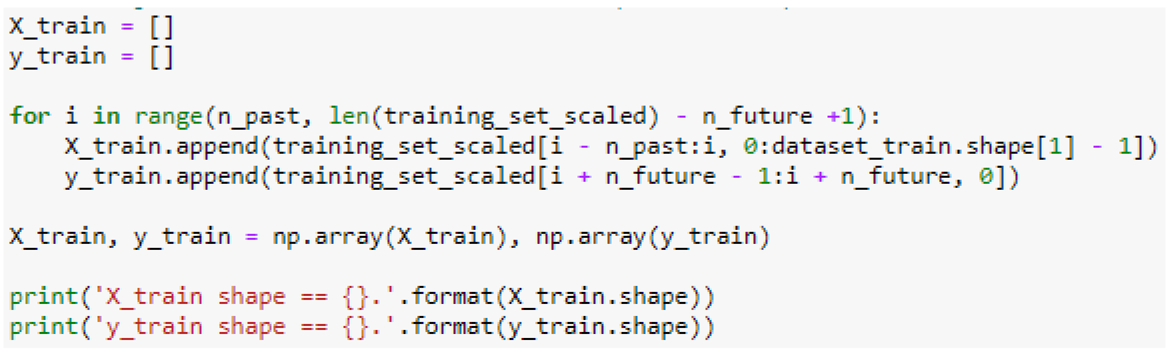
```
dataset_train = dataset_train[cols].astype(str)
for i in cols:
    for j in range(0, len(dataset_train)):
        print( i, j )
        dataset_train[i][j] = dataset_train[i][j].replace(',', '')

dataset_train = dataset_train.astype(float)

# Using multiple features (predictors)
training_set = dataset_train.to_numpy()

print('Shape of training set == {}.'.format(training_set.shape))
training_set
```

Figure 12. Data training



```
# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
training_set_scaled = sc.fit_transform(training_set)

sc_predict = StandardScaler()
sc_predict.fit_transform(training_set[:, 0:1])
```

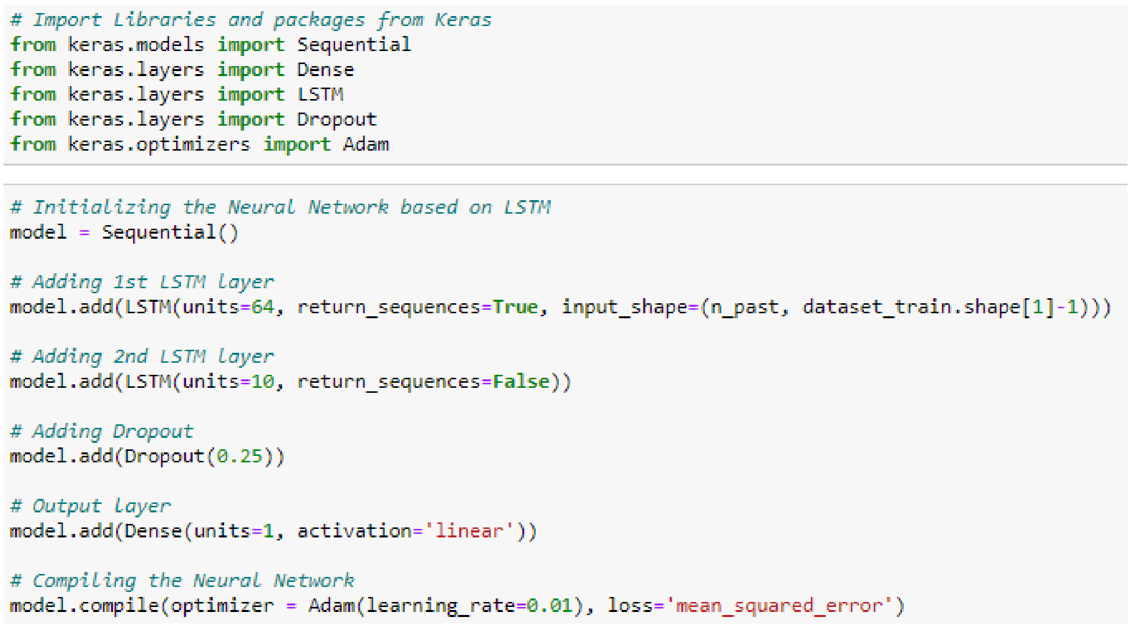
Figure 13. Feature scaling



```
X_train = []
y_train = []

for i in range(n_past, len(training_set_scaled) - n_future +1):
    X_train.append(training_set_scaled[i - n_past:i, 0:dataset_train.shape[1] - 1])
    y_train.append(training_set_scaled[i + n_future - 1:i + n_future, 0])

X_train, y_train = np.array(X_train), np.array(y_train)

print('X_train shape == {}.'.format(X_train.shape))
print('y_train shape == {}.'.format(y_train.shape))
```

Figure 14. Use of past data for training

Figures 12 , 13 and 14 shows the start of data training. This will help to present a more accurate forecast of the future COVID-19 cases.

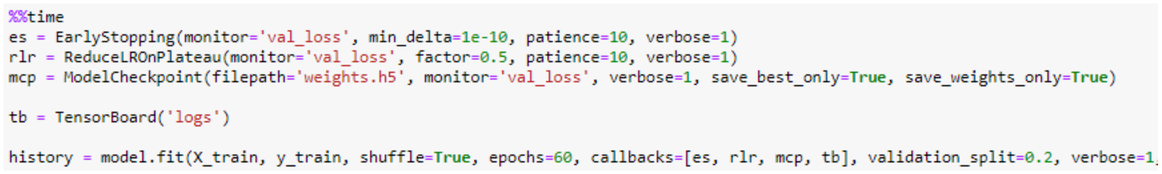


```
# Import Libraries and packages from Keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.optimizers import Adam
```

```
# Initializing the Neural Network based on LSTM
model = Sequential()

# Adding 1st LSTM layer
model.add(LSTM(units=64, return_sequences=True, input_shape=(n_past, dataset_train.shape[1]-1)))

# Adding 2nd LSTM layer
model.add(LSTM(units=10, return_sequences=False))

# Adding Dropout
model.add(Dropout(0.25))

# Output layer
model.add(Dense(units=1, activation='linear'))

# Compiling the Neural Network
model.compile(optimizer = Adam(learning_rate=0.01), loss='mean_squared_error')
```

Figure 15. Model creation

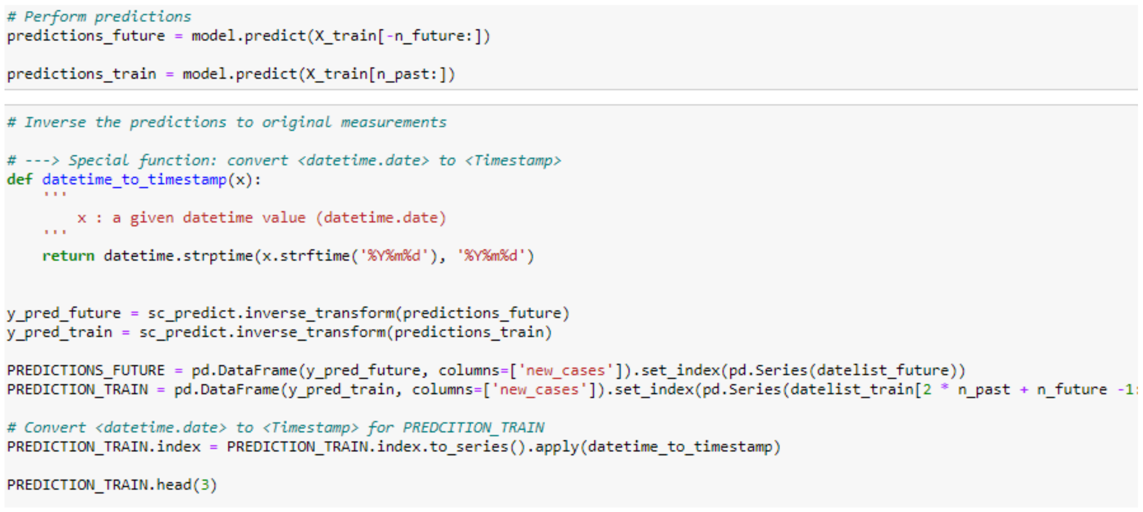
Figure #shows the second part of this simulation using ARIMA, which is the creation of the model.



```
%%time
es = EarlyStopping(monitor='val_loss', min_delta=1e-10, patience=10, verbose=1)
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10, verbose=1)
mcp = ModelCheckpoint(filepath='weights.h5', monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=True)

tb = TensorBoard('logs')

history = model.fit(X_train, y_train, shuffle=True, epochs=60, callbacks=[es, rlr, mcp, tb], validation_split=0.2, verbose=1,
```

Figure 16. With or without improvement

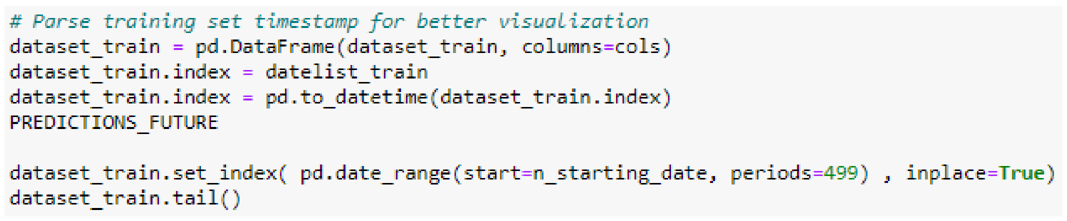
And as the last part of creating the model, the figure above shows the start of training of data. The result then will show if improvements were done.



```
# Perform predictions
predictions_future = model.predict(X_train[-n_future:])

predictions_train = model.predict(X_train[n_past:])
```

```
# Inverse the predictions to original measurements

# ---> Special function: convert <datetime.date> to <Timestamp>
def datetime_to_timestamp(x):
    '''
        x : a given datetime value (datetime.date)
    '''
    return datetime.strptime(x.strftime('%Y%m%d'), '%Y%m%d')


y_pred_future = sc_predict.inverse_transform(predictions_future)
y_pred_train = sc_predict.inverse_transform(predictions_train)

PREDICTIONS_FUTURE = pd.DataFrame(y_pred_future, columns=['new_cases']).set_index(pd.Series(datelist_future))
PREDICTION_TRAIN = pd.DataFrame(y_pred_train, columns=['new_cases']).set_index(pd.Series(datelist_train[2 * n_past + n_future -1:

# Convert <datetime.date> to <Timestamp> for PREDCITION_TRAIN
PREDICTION_TRAIN.index = PREDICTION_TRAIN.index.to_series().apply(datetime_to_timestamp)

PREDICTION_TRAIN.head(3)
```

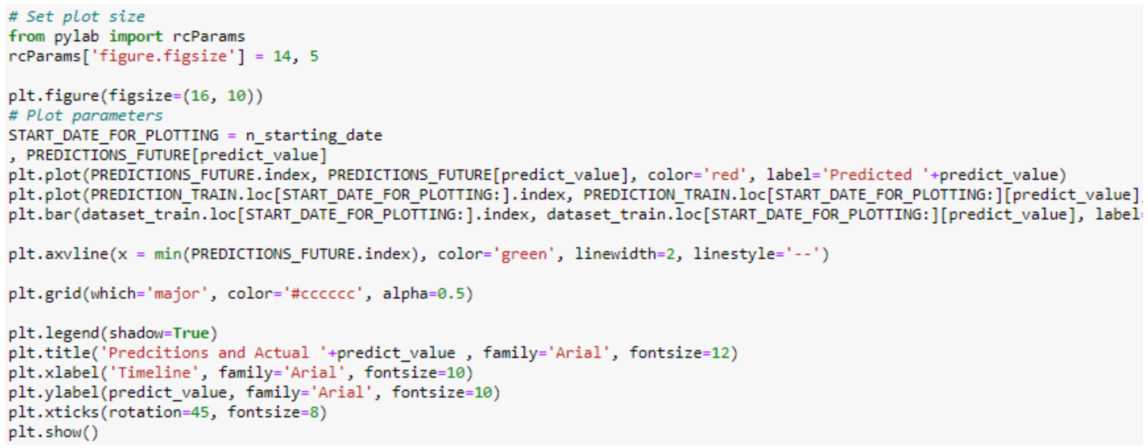
Figure 17. Model for future prediction



```
# Parse training set timestamp for better visualization
dataset_train = pd.DataFrame(dataset_train, columns=cols)
dataset_train.index = datelist_train
dataset_train.index = pd.to_datetime(dataset_train.index)
PREDICTIONS_FUTURE

dataset_train.set_index( pd.date_range(start=n_starting_date, periods=499) , inplace=True)
dataset_train.tail()
```

Figure 18. Parse training

Figures # and # shows the code needed to forecast the future cases of the coronavirus depending on the data sets chosen.



```
# Set plot size
from pylab import rcParams
rcParams['figure.figsize'] = 14, 5

plt.figure(figsize=(16, 10))
# Plot parameters
START_DATE_FOR_PLOTTING = n_starting_date
, PREDICTIONS_FUTURE[predict_value]
plt.plot(PREDICTIONS_FUTURE.index, PREDICTIONS_FUTURE[predict_value], color='red', label='Predicted '+predict_value)
plt.plot(PREDICTION_TRAIN.loc[START_DATE_FOR_PLOTTING:].index, PREDICTION_TRAIN.loc[START_DATE_FOR_PLOTTING:][predict_value]
plt.bar(dataset_train.loc[START_DATE_FOR_PLOTTING:].index, dataset_train.loc[START_DATE_FOR_PLOTTING:][predict_value], label

plt.axvline(x = min(PREDICTIONS_FUTURE.index), color='green', linewidth=2, linestyle='--')

plt.grid(which='major', color='#cccccc', alpha=0.5)

plt.legend(shadow=True)
plt.title('Predcitions and Actual '+predict_value , family='Arial', fontsize=12)
plt.xlabel('Timeline', family='Arial', fontsize=10)
plt.ylabel(predict_value, family='Arial', fontsize=10)
plt.xticks(rotation=45, fontsize=8)
plt.show()
```

Figure 19. Visualization

Lastly, the figure above presents the code that will visualize the results of the prediction.

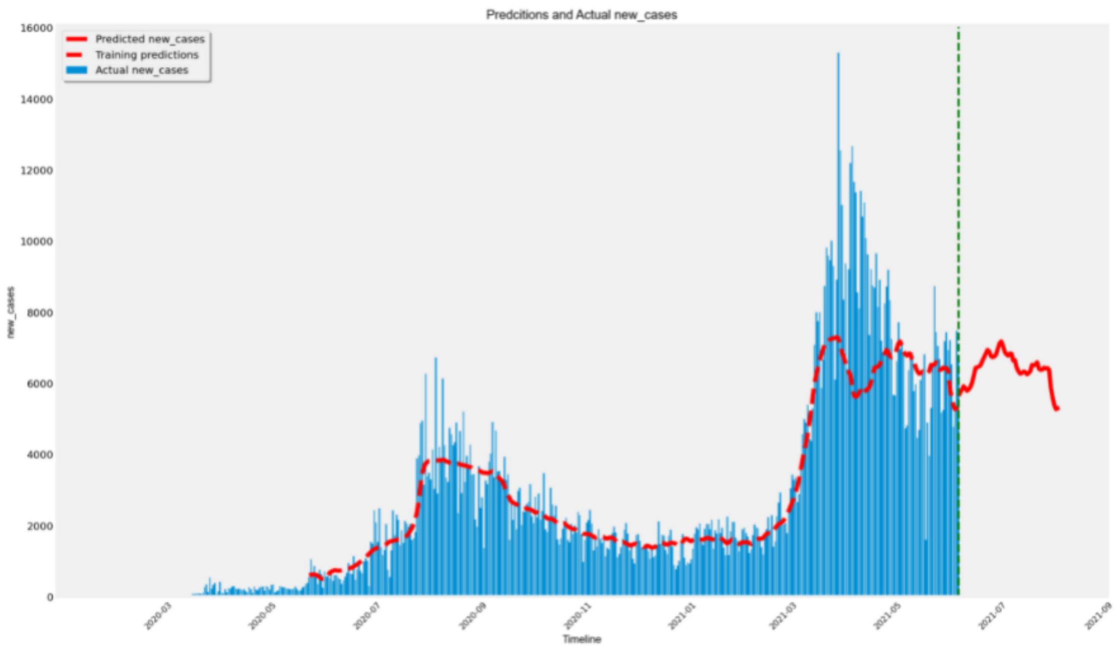


Figure 20. ARIMA Results

Figure # shows the graph with the dates included, starting from the month of March, year 2020 up to June 2021. The actual number of cases is represented by the blue vertical lines. The red discontinued lines show the training predictions, and lastly, the red line on the right side of the graph shows the predicted value of the coronavirus cases.

3.3 Prophet

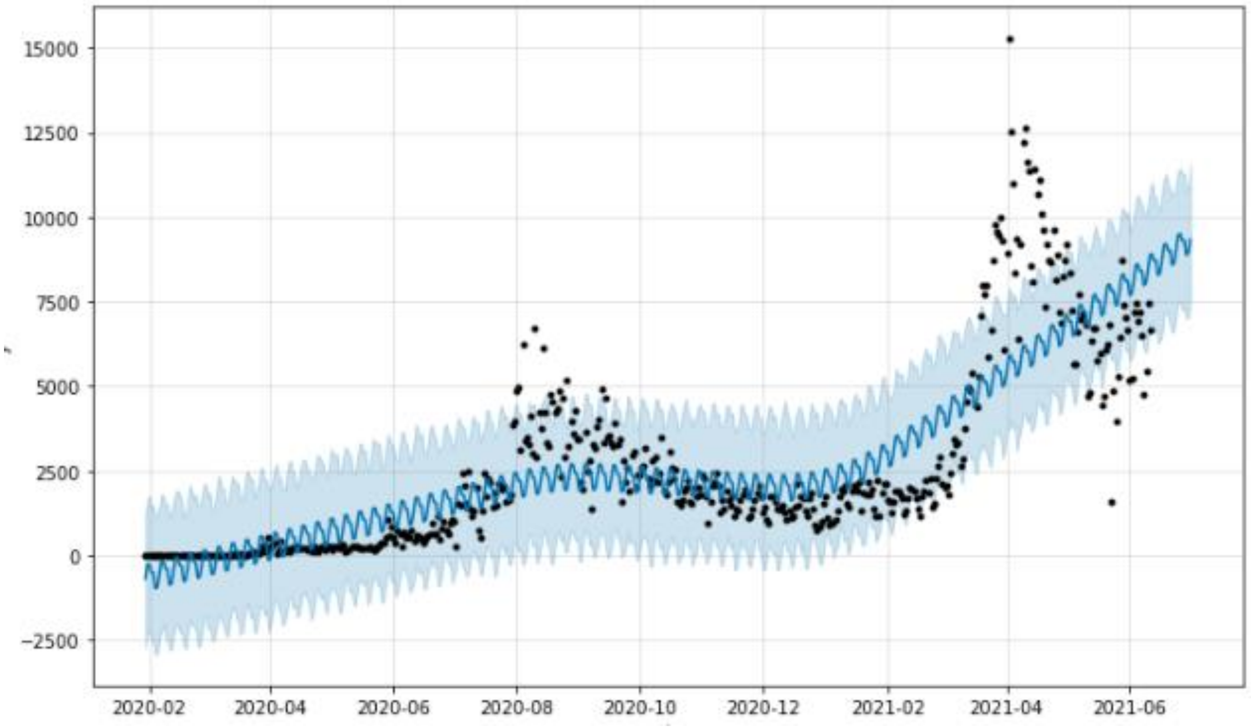


Figure 21. Prophet model of COVID-19 cases

Figure 21 shows the timeline (horizontally) starting from February 2020, up to the month of June, year 2021. Vertically, it shows the number of cases recorded up to 15000s. The black dots on the graph shows the actual number of cases on the given timeline. As an example, the month that shows the highest confirmed cases was last April 2021 in which it went up to 15000s and above. The blue line however shows the forecast of the future updates 20 days ahead of the given timespan. The light blue aura surrounding the blue line shows the possible number of cases on a specific timeline. To conclude, the forecasting is somewhat accurate because it follows the trend of the actual number of coronavirus cases.

ACKNOWLEDGEMENTS

The researchers would like to express their warmest gratitude and sincere thanks to the significant people who have extended their assistance for the success of the study:

To **Dr. Jennifer T. Carpio**, the researcher's research adviser, for her guidance, patience and encouragement. She patiently helped the researchers to improve and correct the study. The researchers would not be able to finish the research without her guidance.

To the **World Health Organization** for supplying daily updates of data regarding the number of cases of the coronavirus disease.

To the **researcher's parents**, for allowing their sons to work during non-school days and for the continuous support they show.

Lastly, to the **Almighty God**, for the divine assistance and strength throughout the research study.

The researchers are very thankful and blessed to have finished their study. They are hoping that this study can help students and other individuals on their study to develop proper understanding about forecasting.

REFERENCES

Saswat S., Chandreyee C., Ayan K.P.& Sarmistha N. (2021). *Journal of The Institution of Engineers (India)*: Series B

Kumar, N. and Susan, S. (2020). "COVID-19 Pandemic Prediction using Time Series Forecasting Models," 2020 *11th International Conference on Computing, Communication and Networking Technologies* (ICCCNT), pp. 1-7

Petropoulos F., and Makridakis S. (2020). *Forecasting the novel coronavirus* COVID-19. PLoS ONE 15(3): e0231236

Adam Hayes (2021). *Advance Technical Analysis (Auto-regressive Integrated Moving Average)* Investopedia

Republic of the Philippines Department of Health. (2020). *2019-NCOV CASE TRACKER*. Available from: https://www.doh.gov.ph/node/19197.

Author

Zalven Dayao Is a Software engineer and A computer science student specializing Machine Learning and Data Science. He is working as a Full-stack website Developer while pursuing Bachelor of Computer science

Dean Michezson Espinosa is a Computer Science student at New Era University. He studied a computer related course, mainly to improve his technological knowledge and problem-solving skills. He plans to take another course after graduating.

Benja Carlo Soliman is 2nd year computer science student at New Era University and aspiring software engineer. He like to play mobile games and have a huge interest in crypto currencies. He also like digital arts and competetive programming.

Maico Angelo Gutierrez is a computer science student at New Era University. He mostly spend his free time developing his skills in programming. He is also an aspiring data scientist.

Christian Alvin Gomez is a Computer Science student at New Era University. He always watching computer studies, mainly to improve his skills. He plans to take another course after graduating.

Adrian Parco is a Web Developer and a Computer Science student at New Era University. The course has helped him to become more efficient in coding, problem-solving, and be professional in web development. He plans to take another course after graduating.