Predicting Dengue Spread based on Historical Data

Mohd Zairi Abd Ghani, Anisah Kamsin, Jeya Jassvine a/p Jeyabala Sundram, Charae Eh Sin
[1,2,3,4]Department of Industrial Computing, Faculty of Information and Technology, Universiti Teknikal Malaysia Melaka, Malaysia.
Email: [1]mohamadzairi658@gmail.com [2]anisahkamsin@gmail.com [3]jassvine@gmail.com
[4]charae.ehsin@gmail.com

Abstract— Dengue fever or also known as break bone fever is caused by mosquito infections. A dengue fever may lead to severe flu-like illness and may even be fatal in some cases. The mosquito involved in the spread of Dengue fever is known as the Aedes mosquito. According to a study, majority of infections are asymptomatic in only a brief systemic viral illness, a small proportion of patients develop potentially fatal complications. These severe manifestations occur relatively late in the disease course to identify the group of patients likely to progress to these complications. However, the cases seen each day during outbreaks remain challenging, simple and inexpensive strategies are urgently needed in order to improve case management and to facilitate appropriate use of limited resources. They focus particularly on the clinical features of the disease and on conventional investigations that are usually accessible in mid-level healthcare facilities in endemic areas, and then discuss a variety of viral, immunological and vascular biomarkers that have the potential to improve risk prediction. In this article we do the Exploratory Data Analysis (EDA) to investigate the dengue dataset. EDA is important because it is widely used technique to ensure the results are valid and applicable to any desired business outcome. We also used 4 types of models which are Naïve Bayes, Simple Exponential Smoothing, Holt's Winter Seasonal Model and LSTM to choose the best model for this project by comparing the RMSE. The smallest the RMSE, the better the model. The better the results are obtained.

Index Terms— Dengue, Naïve Bayes, Simple Exponential Smoothing, Holt's Winter Seasonal Model, LSTM, EDA

I. INTRODUCTION

A. Business Understanding

A dengue fever can bring about varying levels of disease symptoms from mild to severe or worst still, even fatal conditions. Mild symptoms of dengue are such as muscle and joint aches, nausea, vomiting, fever and so on. However, severe cases may result in shock, internal bleeding and even death. It is also discovered that about 1 in every 20 people who are down with a dengue fever will develop severe dengue. It is also reported by WHO that the number of deaths caused by dengue fever has reached 5.2 million in the year of 2019.

A dengue fever is induced by a simple mosquito bite from a specific species of female mosquito namely the Aedes mosquito. There are 4 viruses present in this mosquito that will be injected into the victim's bloodstream and this will cause the victim to fall sick. Despite the years of research, there isn't much of a solution for dengue fever. There is no proper vaccination rolled out and this disease is of no cure.

Besides, a majority of infections are asymptomatic in only a brief systemic viral illness, a small proportion of patients develop potentially fatal complications [1]. These severe manifestations, including a unique plasma leakage syndrome, a coagulopathy sometimes accompanied by bleeding, and organ impairment, occur relatively late in the disease course to identify the group of patients likely to progress to these complications. However, the cases seen each day during outbreaks remain challenging, and simple and inexpensive strategies are urgently needed in order to improve case management and to facilitate appropriate use of limited resources. They focus particularly on the clinical features of the disease and on conventional investigations that are usually accessible in mid-level healthcare facilities in endemic areas, and then discuss a variety of viral, immunological and vascular biomarkers that have the potential to improve risk prediction.

The objective of this project whereby our task will be to predict the number of dengue cases based on week of year per location. These dengue cases are predicted based on environmental factors that contribute to the outbreak of this dengue disease. For example, the changes in temperature, precipitation, humidity, and more. The purpose of these prediction analysis is for the people of the affected area to be able to prepare themselves and take precautions. Not only that, an understanding relationship between this disease and the contributing environmental factors can help further improvements of research initiatives to curb this life-threatening pandemic that has been going on for years now. With the help of data driven technology, we hope to fight this dengue disease. The next objective would be to apply the knowledge of data cleaning, analysis, as well as web scraping to obtain the prediction of dengue cases. Besides that, we also aim to contribute to the SDG 3 - Good Health & Well-Being by carrying out prediction analysis.

B. Sustainable Development Goal

The Sustainable Development Goals which was launched by WHO in January 2016 is a continuation of the Millennium Development Goals (MDG). Based on this project, we can contribute to SDG 3 which has been incorporated into MOH's strategic plan and also the Health Service Transformation Plan.

Sustainable Development Goal (SDG) for our project is predicting disease spread where it is to find a relationship understanding between environmental factors encouraging outbreak of dengue fever. It is also to predict dengue cases for the state based on the week of year and climate factors based on historical data. Next SGD is to derive useful insights from the datasets from various sources for prediction of dengue case by applying understanding, wrangling and exploring the dataset as well as applying web scraping. Lastly to give contribution to SDG 3 where it discusses Good Health & Well-Being in which we would like to curb a life-threatening disease by predicting how seriously the environment factors affect which can help achieve the goal of obtaining good health and well-being amongst the citizens.

C. Goals

1. Predicting Disease Spread
   a. Find a relationship understanding between environmental factors encouraging outbreak of Dengue fever.
   b. Predict dengue cases for the state based on the week of year and climate factors based on historical data.
2. Derived useful insights from the datasets from various sources for prediction of dengue case
   a. By applying understanding, wrangling and exploring the dataset as well as applying web scraping.
3. Contribute to SDG 3
   a. SDG 3 discusses Good Health & Well-Being in which we would like to curb a life-threatening disease by predicting how seriously the environmental factors affect which can help achieve the goal of obtaining good health and well-being amongst the citizens.

D. Questions

1. How can we predict the areas with the highest number of dengue cases?
2. Does temperature, humidity, and precipitation correlate with dengue spread?
3. Is historical data can be used to predict future dengue cases?
4. Can we predict the dengue based on week of year only? Or can we also predict dengue based on the weather condition? Or do we need both for prediction?
5. What can we learn from this historical data and what can we do other than predicting future dengue cases?
6. Which weather data does not correlate much with dengue cases?
7. What are the causes of dengue cases in the area?
8. Can real-time monitoring be done to automate the process of detecting future dengue outbreak based on weather data and accumulative cases.

E. Success and Its Measurement

TABLE I
TABLE OF SUCCESS AND MEASUREMENTS

Success	**Measurements**
1. Dengue fever can be predicted based on the historical data.	1. By giving the week of year only, or with the weather condition, we can predict the case of dengue for that particular location.
2. Predicting when the future outbreak will occur based on the real time weather data.	2. High accuracy in detecting the potential for dengue outbreak based on data and alerting the user based on real-time data.
3. Improve public health officials' preparation and standardization of dengue case reporting.	3. Early warnings of epidemics, which can help to minimize fatality rates by using the predicted case of the dengue from our model so any preventative method can be applied.

F. Data Sources

The data is obtained from two sources. First, the weekly dengue case data is the data from Ministry of Health which is available at Portal Terbuka Malaysia, which contains weekly cases for all state from 2010-2017. The second source is the weather data that is downloaded from Visual Crossing website, which contains daily weather data from 2010 until 2017. The weather data reading is obtained from Sultan Abdul Aziz Shah International Airport in Subang. The weather data contains temperature, humidity, and precipitation, which is regarded as the factors that affect dengue transmission [2][3].

BILANGAN KES PENYAKIT DENGGI MINGGUAN MENGIKUT NEGERI, TAHUN 2010

MINGGU EPID	JOHOR	KEDAH	KELANTAN	MELAKA	NEGERI SEMBILAN	PAHANG	PERAK	PERLIS	PULAU PINANG	SABAH	SARAWAK	SELANGOR	TERENGGANU	WP KUALA LUMPUR	WP LABUAN	MALAYSIA
1	46	7	17	8	20	17	23	0	37	55	255	400	18	62	0	965
2	49	5	16	7	25	21	38	1	28	35	207	468	15	83	2	1000
3	71	11	23	14	29	20	24	2	27	45	183	502	12	101	1	1065
4	79	11	13	16	70	31	18	6	20	73	133	525	24	80	2	1101
5	75	15	18	9	112	33	19	4	20	63	121	461	43	109	1	1103
6	54	7	20	7	92	29	34	8	27	49	123	474	19	116	0	1059
7	59	9	18	6	62	25	27	2	22	48	82	451	30	93	1	935
8	50	4	18	7	50	31	36	2	34	48	114	583	28	116	0	1121
9	45	14	28	4	31	38	65	4	23	49	100	496	23	154	0	1074
10	59	9	27	10	32	33	46	0	24	39	106	467	18	169	0	1039
11	53	12	23	11	25	21	31	1	18	29	72	447	15	140	0	898
12	51	15	35	6	20	28	27	1	14	13	87	381	12	107	0	797
13	66	12	21	11	23	19	40	1	18	25	55	425	13	94	0	823
14	55	24	21	12	24	24	49	5	14	26	51	460	15	106	0	886
15	79	21	38	11	25	17	29	1	21	19	66	459	18	109	1	914
16	50	18	29	18	37	13	38	4	18	22	42	441	17	138	0	885
17	62	18	31	16	51	20	40	2	17	18	55	329	12	119	0	790
18	88	23	35	10	42	15	38	4	17	37	47	301	11	92	0	760
19	53	21	32	15	51	17	33	4	18	35	44	337	15	91	0	766
20	61	19	32	18	56	14	37	6	27	53	48	370	22	71	0	834
21	90	33	45	17	40	34	38	3	21	39	46	287	15	81	0	789
22	96	25	33	14	50	27	54	2	21	43	44	311	11	83	0	814
23	91	17	32	13	70	57	78	5	23	91	54	286	7	66	1	891
24	113	14	40	13	65	44	66	3	31	77	57	325	16	76	2	942

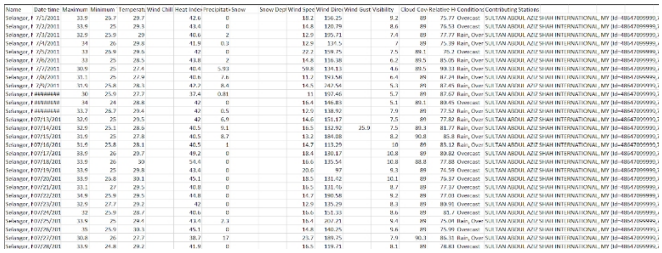


Fig. 1. The dengue dataset (top) and historical weather dataset (bottom)

The attributes for the historical dataset are:

a. Name
b. Date time
c. Maximum Temperature
d. Minimum Temperature
e. Average Temperature
f. Average Precipitation
g. Wind Chill
h. Heat Index
i. Snow
j. Snow Depth
k. Wind Speed
l. Wind Direction
m. Wind Gust
n. Visibility
o. Cloud Cover
p. Relative Humidity
q. Condition
r. Contributing Stations

II. PROBLEM FORMULATION

This section presents the problem formulation on the dataset obtained, which includes data wrangling, Exploratory Data Analysis (EDA), data management, and data preparation.

A. Data Cleaning

The weekly dengue cases data does not need to be cleaned as there is no missing value in this dataset. However, as the scope of this study is for Selangor state only, the weekly dengue case data for other state other than Selangor are being removed. In addition, as the data for each year are being stored in a separate Excel file, all this data are merged later into one data frame.

Meanwhile, for historical weather dataset, there are several missing values present in the data, notably in the Wind Gust attributes. In addition, all row in Snow and Snow Depth attributes are missing values due to there is no snow in Malaysia. Hence, both attributes are removed.

Furthermore, As for Wind Gust attributes, despite there is a few strategies that can be utilized for dealing with the missing values, we decide to drop this column too because it does not play a major role in contributing to dengue case. Besides, the number of missing values in this column are huge, so a simple strategy such as replace missing value with the mean cannot be used.

Moreover, attributes Names, Conditions, and Contributing Stations are also removed. Conditions contains value such as 'Rain' and 'Overcast' and it is harder to convert this into numerical value for analysis and modeling. Meanwhile, the other two attributes contain the same value for all row, which is 'Selangor, Malaysia' for Name and 'Sultan Abdul Aziz International Airport' for Contributing Stations.

B. Data Reshaping

For data reshaping, the daily weather data is converted into weekly data to reflect the weekly cases data. This is achieved by summing seven rows – which means a week – and then the average is calculated. Meanwhile, to find the Min and Max Temperature, instead of average, the lowest and highest value in a week is obtained. Next, this value is stored in a data frame and finally written in a .csv file once the whole weather data are converted into weekly data. Figure 2 shows a code snippet of the Python code used for this.

```
count = 0
maxTempArr = []
weeklyData = []

for i in df['Maximum Temperature']:
    if (count % 7 != 0):
        weeklyData.append(i)
        count = count + 1
    elif count % 7 == 0:
        weeklyData.append(i)
        count = count + 1

    if count == 7:
        maxTempArr.append(max(weeklyData))
        weeklyData = []
        count = 0 #reset count

print(maxTempArr)
```

Fig. 2. The code snippet for converting daily weather data into weekly data

Note that based on the code in Figure 2, which is for converting daily Max Temperature into weekly data, the maxTempArr array that is used to store the overall data for this particular year while the weeklyData is used to store maximum value for the current week. For-loop is used to iterate data every seven row (which means a week). The results of the conversion is shown in Figure 3.

	Year	Week	Week Start Date	Max Temperature	Min Temperature	Average Temperature	Average Precipitation	Wind Speed	Wind Direction	Visibility	Cloud Cover	Average Relative Humidity
0	2011	1	01/01/2011	34.0	25.1	27.400000	11.068571	16.200000	186.085714	9.228571	89.814286	84.811429
1	2011	2	01/08/2011	33.9	25.0	27.342857	3.130000	14.471429	197.910000	8.500000	89.357143	83.650000
2	2011	3	01/15/2011	34.9	25.0	27.785714	0.571429	13.642857	185.270000	8.071429	89.328571	79.125714
3	2011	4	01/22/2011	33.0	25.7	27.571429	7.061286	14.642857	181.302857	8.114286	89.257143	82.911429
4	2011	5	01/29/2011	33.9	25.0	26.800000	8.284286	12.885714	195.960000	8.728571	90.414286	85.298571
5	2011	6	02/05/2011	34.9	25.9	28.428571	2.542857	16.700000	187.840000	7.400000	89.200000	80.888571
6	2011	7	02/12/2011	34.9	26.0	28.371429	3.654286	17.428571	186.300000	7.371429	89.457143	82.614286
7	2011	8	02/19/2011	35.9	26.0	28.485714	2.485714	18.300000	176.685714	8.871429	89.042857	80.344286
8	2011	9	02/26/2011	34.0	26.7	28.457143	15.301429	15.914286	186.228571	8.185714	89.328571	81.721429
9	2011	10	03/05/2011	35.8	26.0	27.957143	1.380000	15.371429	186.430000	8.600000	90.328571	84.022857
10	2011	11	03/12/2011	34.1	25.9	27.557143	15.973000	14.857143	186.922857	7.785714	89.928571	85.817143
11	2011	12	03/19/2011	33.0	25.0	27.285714	23.111286	17.257143	218.464286	8.642857	90.228571	85.785714
12	2011	13	03/26/2011	34.9	27.7	29.642857	0.410000	17.000000	237.887429	10.185714	89.285714	73.916714
13	2011	14	04/02/2011	35.9	26.9	29.228571	4.291429	16.714286	210.134286	9.985714	89.257143	79.485714
14	2011	15	04/09/2011	34.0	26.0	28.328571	14.567143	15.785714	186.482857	8.985714	89.757143	84.185714

Fig. 3. The data after being converted

C. Data Management

The weekly cases and historical weather data that has been cleaned are merged together into a single .csv file. This ensure that the EDA and modeling process are easier as the data has already been prepared and the data only needs to be imported for usage. This is performed manually via Excel.

Finally, the attributes that are used during Data Analysis and modeling later are as follows:

a. Total Case
b. Max Temperature
c. Min Temperature
d. Average Temperature
e. Average Precipitation
f. Wind Speed
g. Wind Direction
h. Visibility
i. Cloud Cover
j. Average Relative Humidity

Total Case is the target attribute that will be predicted while the others are the input variables. Figure below shows the data before and after being reshaped.

	Name	Date time	Maximum Temperature	Minimum Temperature	Temperature	Wind Chill	Heat Index	Precipitation	Snow	Snow Depth	Wind Speed	Wind Direction	Wind Gust	Visibility	Cloud Cover	Relative Humidity	Conditions	Contributing Stations
0	Selangor, Malaysia	01/01/2011	33.9	25.1	27.8	NaN	41.5	19.10	NaN	NaN	20.3	192.00	NaN	9.8	89.1	82.40	Rain, Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...
1	Selangor, Malaysia	01/02/2011	32.9	25.0	28.5	NaN	40.6	0.00	NaN	NaN	16.4	191.63	29.5	6.9	89.1	81.27	Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...
2	Selangor, Malaysia	01/03/2011	31.0	25.1	27.8	NaN	37.7	6.60	NaN	NaN	14.8	198.63	NaN	9.9	89.0	83.54	Rain, Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...
3	Selangor, Malaysia	01/04/2011	34.0	24.9	27.6	NaN	43.0	27.82	NaN	NaN	13.0	182.21	44.3	8.0	89.4	85.70	Rain, Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...
4	Selangor, Malaysia	01/05/2011	33.7	24.3	27.7	NaN	42.0	12.90	NaN	NaN	16.4	175.58	NaN	9.7	89.1	84.40	Rain, Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
179	Selangor, Malaysia	12/27/2011	34.9	25.7	28.4	NaN	43.6	6.97	NaN	NaN	11.2	184.25	NaN	8.5	89.6	78.51	Rain, Overcast	SULTAN ABDUL AZIZ SHAH INTERNATIONAL, MY (id=4...

	Year	Week	Week Start Date	Max Temperature	Min Temperature	Average Temperature	Average Precipitation	Wind Speed	Wind Direction	Visibility	Cloud Cover	Average Relative Humidity
0	2011	1	01/01/2011	34.0	25.1	27.400000	11.068571	16.200000	186.005714	9.228571	89.614286	84.811429
1	2011	2	01/08/2011	33.9	25.0	27.342857	3.130000	14.471429	197.910000	8.500000	89.357143	83.650000
2	2011	3	01/15/2011	34.9	25.0	27.785714	0.571429	13.642857	185.270000	8.071429	89.028571	79.125714
3	2011	4	01/22/2011	33.0	25.7	27.571429	7.064286	14.642857	181.302857	8.114286	89.257143	82.911429
4	2011	5	01/29/2011	33.9	25.0	26.800000	8.284286	12.885714	195.960000	8.728571	90.414286	85.298571
5	2011	6	02/05/2011	34.9	25.9	28.428571	2.542857	16.700000	187.840000	7.400000	89.200000	80.888571
6	2011	7	02/12/2011	34.9	26.0	28.371429	3.654286	17.428571	186.300000	7.371429	89.457143	82.614286
7	2011	8	02/19/2011	35.9	26.0	28.485714	2.485714	18.300000	176.695714	8.871429	89.042857	80.344286
8	2011	9	02/26/2011	34.0	26.7	28.457143	15.301429	15.914286	188.225714	8.185714	89.028571	81.721429
9	2011	10	03/05/2011	35.8	26.0	27.957143	1.380000	15.371429	186.430000	8.600000	90.028571	84.022857
10	2011	11	03/12/2011	34.1	25.9	27.557143	15.970000	14.857143	166.922857	7.785714	89.928571	85.817143
11	2011	12	03/19/2011	33.0	25.0	27.285714	23.111286	17.257143	218.464286	8.642857	90.228571	89.785714
12	2011	13	03/26/2011	34.9	27.7	29.642857	0.410000	17.000000	237.881429	10.185714	89.285714	73.915714
13	2011	14	04/02/2011	35.9	26.9	29.228571	4.291429	16.714286	210.134286	9.085714	89.257143	79.485714
14	2011	15	04/09/2011	34.0	26.0	28.028571	14.567143	15.785714	188.482857	8.085714	89.757143	84.185714

Fig. 4. Data before and after reshaping

D. Exploratory Data Analysis

Exploratory data analysis (EDA) is used in this project to investigate the dengue dataset and to summarize the important and main characteristics as well as applying data visualization methods. This helps us to better understand the dengue dataset and the factors that are affecting the target attribute which is total cases of dengue. EDA helps us discover patterns, spot anomalies, and check for assumptions. EDA is primarily used to see how and what the data can reveal beyond the formal modelling and hypothesis testing which eventually leads to a better understanding of the dataset variables and correlations of the attributes between each other.

EDA is a widely used technique to ensure the results are valid and applicable to any desired business outcome and goals which in our case is to predict the Dengue disease spread as well as to contribute to SDG 3 : Ensure healthy lives and promote well-being for all at all ages.

In our EDA, we analyzed the dataset that has undergone cleaning, management and reshaping. We read the dataset and proceeded with simple generation of the preview of statistics in our dataset. The result of this preview is shown in the diagram below.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 416 entries, 0 to 415
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Unnamed: 0                  416 non-null    int64
 1   Year                        416 non-null    int64
 2   Week                        416 non-null    int64
 3   Week Start Date             416 non-null    object
 4   Total Case                  416 non-null    int64
 5   Max Temperature             416 non-null    float64
 6   Min Temperature             416 non-null    float64
 7   Average Temperature         416 non-null    float64
 8   Average Precipitation       416 non-null    float64
 9   Wind Speed                  416 non-null    float64
 10  Wind Direction              416 non-null    float64
 11  Visibility                  416 non-null    float64
 12  Cloud Cover                 416 non-null    float64
 13  Average Relative Humidity   416 non-null    float64
dtypes: float64(9), int64(4), object(1)
memory usage: 45.6+ KB
```

Fig. 5. Type of data of attributes

	Unnamed: 0	Year	Week	Total Case	Max Temperature	Min Temperature	Average Temperature	Average Precipitation
count	416.000000	416.000000	416.000000	416.000000	416.000000	416.000000	416.000000	416.000000
mean	25.500000	2013.500000	26.500000	649.435096	34.345673	26.026683	28.235062	8.525251
std	15.026402	2.294047	15.026402	465.110678	0.934367	0.816301	0.866290	6.858117
min	0.000000	2010.000000	1.000000	70.000000	31.000000	23.000000	25.785714	0.000000
25%	12.750000	2011.750000	13.750000	198.750000	33.900000	25.700000	27.571429	3.028571
50%	25.500000	2013.500000	26.500000	581.500000	34.100000	26.000000	28.228571	7.035000
75%	38.250000	2015.250000	39.250000	1023.000000	34.900000	26.800000	28.803571	13.275357
max	51.000000	2017.000000	52.000000	2033.000000	37.900000	28.000000	30.628571	32.898571

Wind Speed	Wind Direction	Visibility	Cloud Cover	Average Relative Humidity
416.000000	416.000000	416.000000	416.000000	416.000000
16.265316	184.020192	8.565659	85.415385	80.229444
1.803595	32.589727	1.179123	10.210317	4.677670
10.642857	101.347143	1.857143	58.114286	66.545714
15.153571	163.508571	8.207143	89.028571	77.472143
16.242857	184.175000	8.792857	89.385714	80.724286
17.360714	202.671786	9.232143	89.757143	83.436071
21.228571	288.417143	10.800000	91.414286	90.204286

Fig. 6. Description of dataset

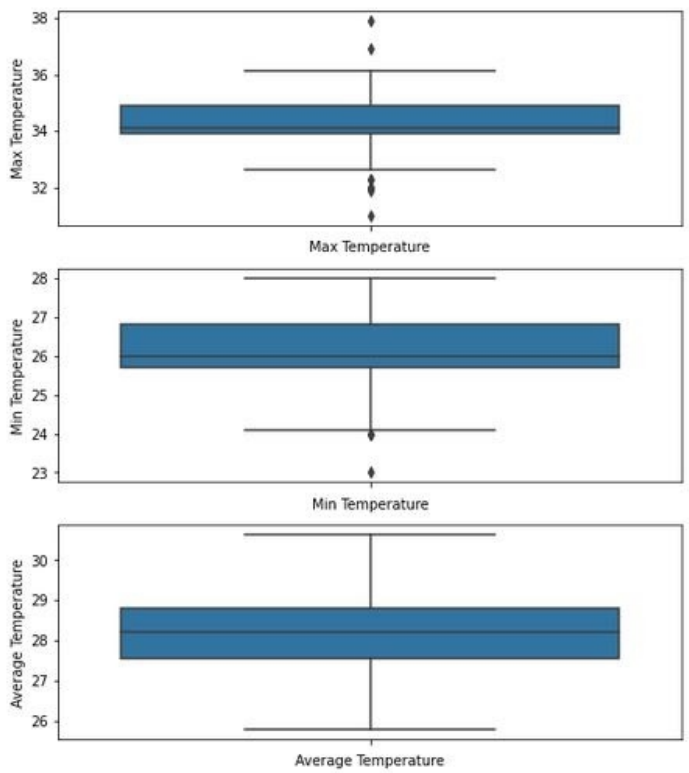
Based off the statistical preview, we can deduce a few critical summaries and measures. Firstly, the distribution of

summary can be deduced. Based on the description of dataset, we can identify that the maximum number of weeks is 52, the highest total cases that has occurred is 2033, the least value of cloud cover occurring in an area is 58.11 and the highest value of relative humidity that occurs is 90.20.

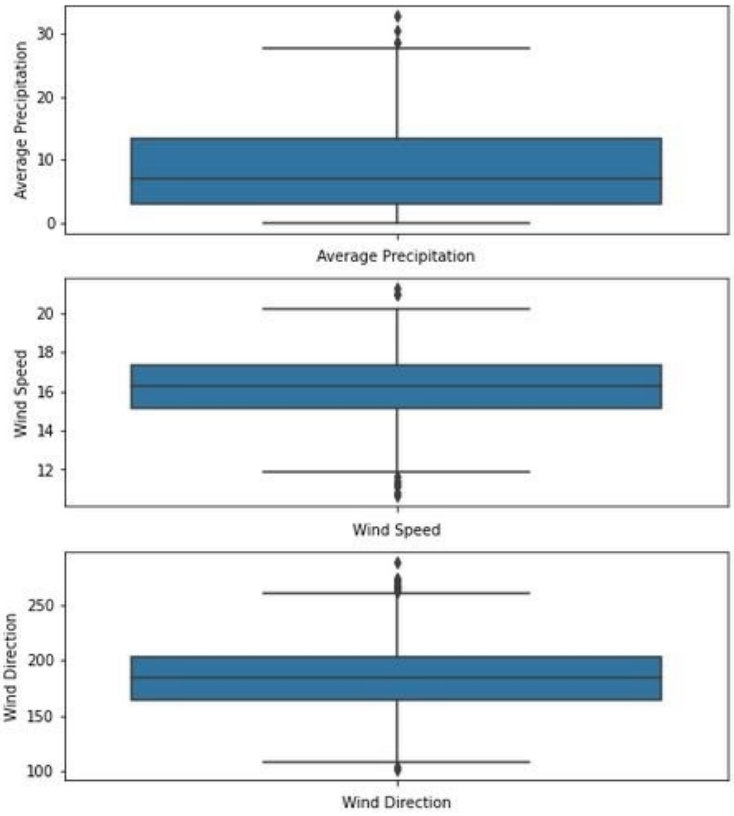
Distributions may also be described in values of central tendency. Central tendency is basically an estimate of the "center" of a distribution amongst the data. There are 3 major estimates which are mean, median and mode. Based on the dengue dataset, the appropriate measure that we used is the mean. Mean is also better known as average and is the most common use of method when it comes to describing central tendency. To compute and obtain a mean value, the values in that attribute is all added up and then divided by the number of values. Based on our dataset, it can be deduced that the mean of average temperature is 28.235062, the minimum value of the average temperature is 25.785714 whilst the maximum value of the average temperature is 30.628571, the mean of average precipitation is 8.525251 and the mean of average relative humidity is 80.229444.

Next, distribution is the dispersion value which refers to the spread of values in correlation to the central tendency itself. There are two common measures of dispersion which are the range and standard deviation. Here we calculated the standard deviation since it is a more accurate and detailed estimate of dispersion. This is because if there are outliers it will greatly affect the standard deviation value. The standard deviation that can be deduced from this dataset is standard deviation of average temperature calculated is 0.866290, standard deviation of average precipitation is 6.858117 and the average relative humidity is 4.677670.

The next step in the EDA process is to handle outlier data. Outlier analysis is a process that involves anomalous observation wi our dataset. Before analyzing the outlier, it is important to understand what an outlier is on its own. An outlier data is an extre value that deviates greatly from other datapoints. Outliers can be caused by incorrect entry, computational errors or sampling errors. Outliers can cause data to be inaccurate or cause it to be statistically insignificant. There are several methods used to detect and handle outliers. Here we have used the box plot method. The box plot is a method for graphically depicting groups of numerical data throughout their quartiles. The box of the box plot data usually extends from the first to the third quartile (Q1 – Q3) and a line will be plotted within the box at the value of the median (Q2). Then there are whiskers that extend from the boxes to show the range of data. If there are any outliers, they will be out of range and beyond the whiskers. Boxplots not only show outlier values but also shows the information regarding symmetry. Our observation of outlier analysis is recorded as per below.



a.



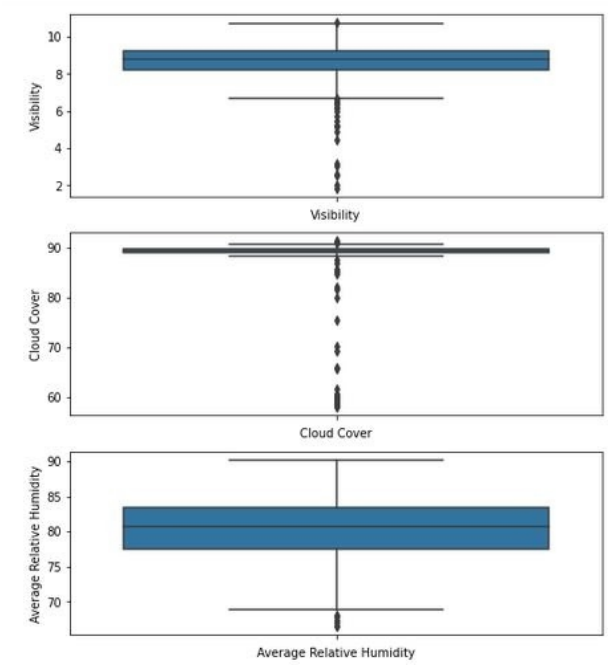
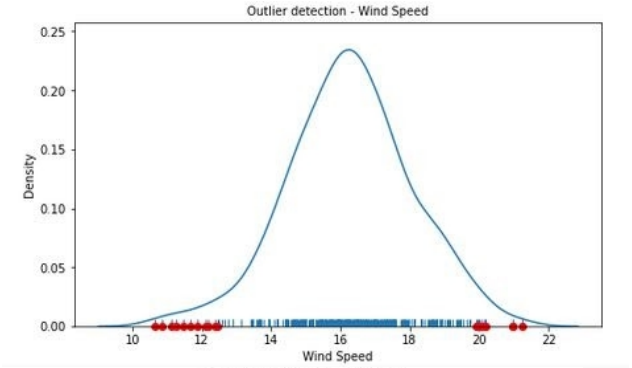
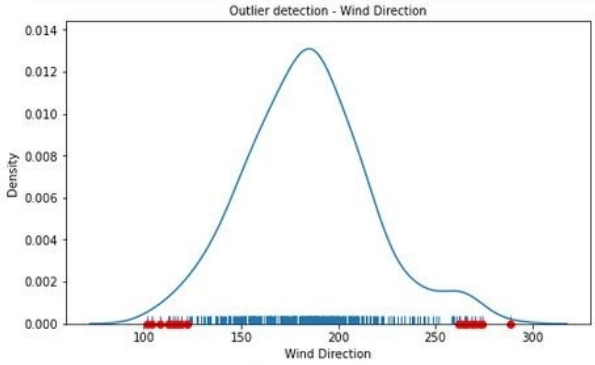
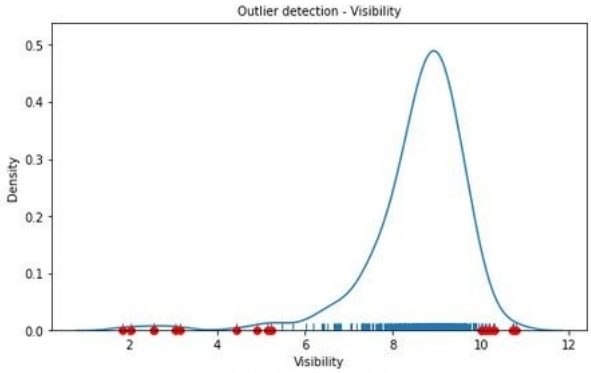


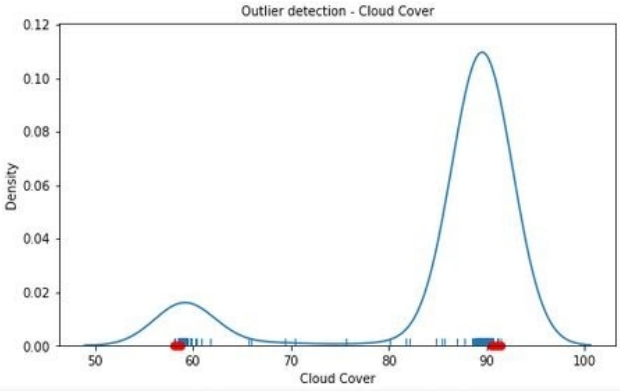
Fig. 7. Shows outlier analysis in dengue dataset.

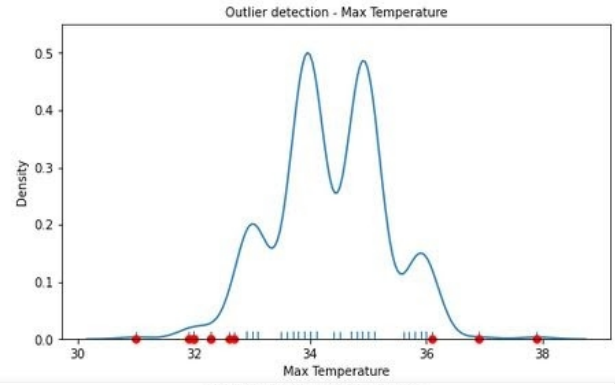
After the removal of outliers, the data is visualized again. The result is recorded as below.

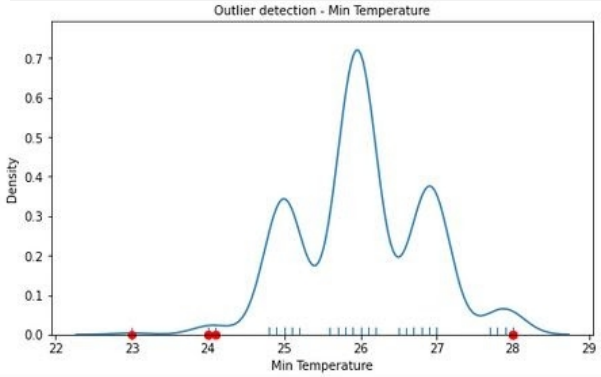












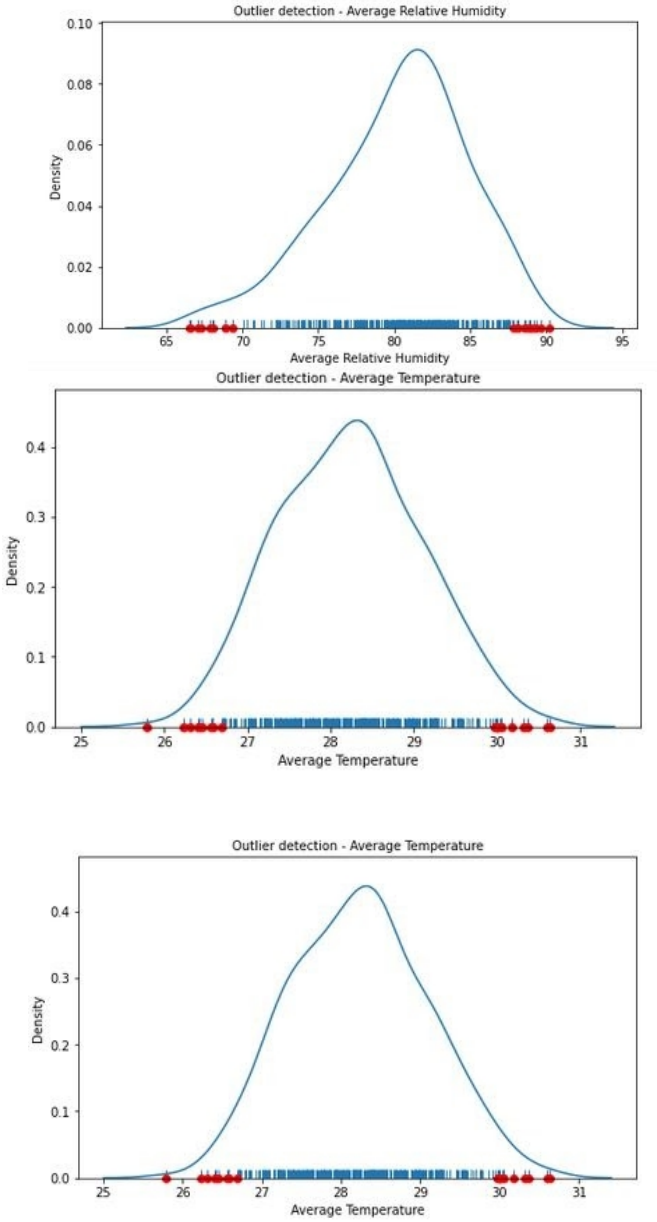


Fig. 9 . Shows the visualization after outlier removal

The values marked with a dot in the x-axis of the graphs are the outlier values that has been removed from the column based on the threshold settings. The threshold is set by calculating interquartile range(Q3-Q1). The limits are calculated to lie between the calculated interquartile range.

Next, we analyze the correlations of the variables present in the dataset by using the correlation matrix. A correlation matrix is a table that shows the correlation coefficients between variables. This matrix table is used to summarize data as to put input into a more advanced analysis to see patterns, for example how strongly they correlate with each other. The correlation matrix obtained by using our dengue dataset is represented in the diagram below.

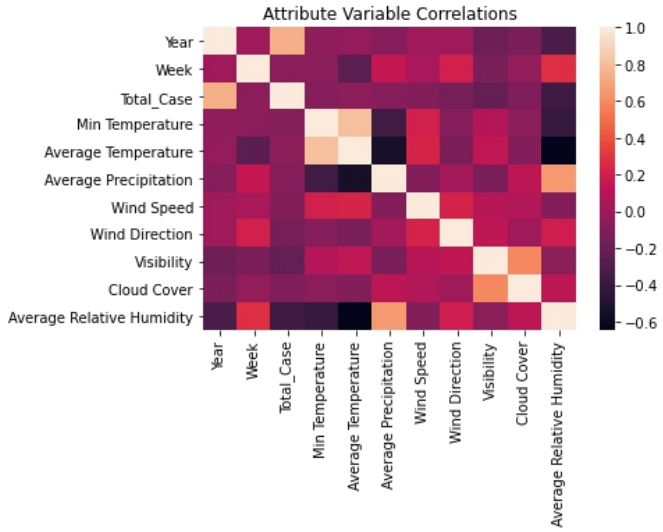
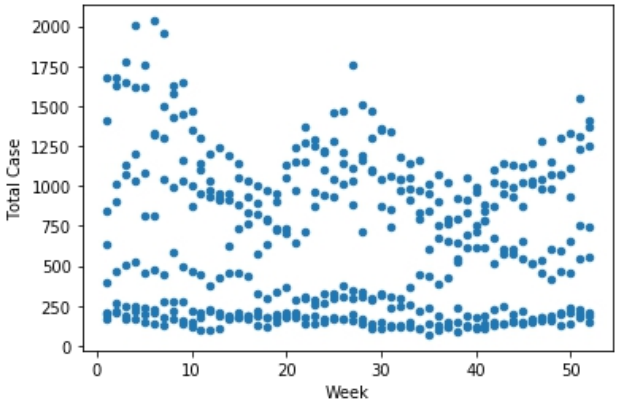


Fig. 10 . Correlation matrix of dengue dataset

To read and understand this matrix, we have to match the color of the boxes to the colored scale on the right side. The value 1.0 which is represented by the lighter shade means that the variables show a positive linear correlation between the two variables. It is important to note that the diagonal elements of the correlation matrix is always 1.o because they are the correlation of the column with itself. As the color scale goes down, it shows a medium dark purple hue that ranges around 0 values. 0 simply indicates there is no linear correlation between the two variables. And as the scale goes lower, it enters a negative scale which indicates a negative correlation between two variables. -1 indicates a perfect negative correlation. From this correlation derived from our dengue dataset, it can be deduced that the temperature data are mostly strongly correlated with each other. However, the overall total case variable does not show obvious strong correlations.

And finally, scatterplot analysis. This analysis is done two datasets against each other to see a relationship. The relationship will be concluded from the characteristic of the scatter in the plot. The scatterplot analysis within this dataset is shown as below.



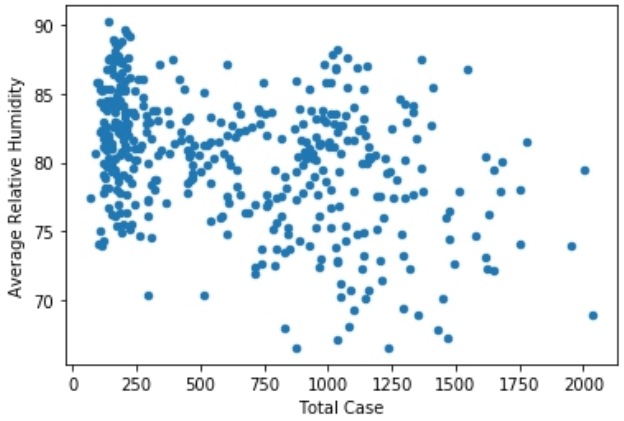


Fig. 11. Scatterplot analysis

In the first scatterplot, the comparison of data between the Week and Total Case is observed. However, there is no obvious correlation between these data to deduce a relationship.

Whereas, in the second diagram scatterplot which plots the data between Total Case and Average Relative Humidity, it shows an obvious relationship of the humidity affecting the number of total cases. As the average relative humidity increases, the total cases also increase. A hypothesis that can be made is that, when the humidity increases it provides a conducive environment for the mosquitoes to breed. This then increases the number of mosquitoes which will directly influence the probability of humans being affected by the mosquitoes and being diagnoses with dengue.

III. MODELING AND EVALUATION

A. *Naïve Bayes*

Naïve Bayes classification algorithm can be applied fast and easy because it is not as complex as most algorithms. Although its classification accuracy is not as precise as the one of more complex algorithms, you might get similar results in a fraction of the computation time. To calculate the conditional class probability of the attribute values, the *Naïve Bayes* classifier algorithm uses the Bayes' theorem. This kind of probability is called the posterior class probability.

```
from sklearn.metrics import mean_squared_error
from math import sqrt

dd= np.asarray(train.Total_Case)
y_hat = test.copy()
y_hat['naive'] = dd[len(dd)-1]

plt.figure(figsize=(12,8))
plt.plot(train.index, train['Total_Case'], label='Train')
plt.plot(test.index,test['Total_Case'], label='Test')
plt.plot(y_hat.index,y_hat['naive'], label='Naive Forecast')
plt.legend(loc='best')
plt.title("Naive Forecast")
plt.show()
```

Fig. 12 . Code for Naïve Bayes

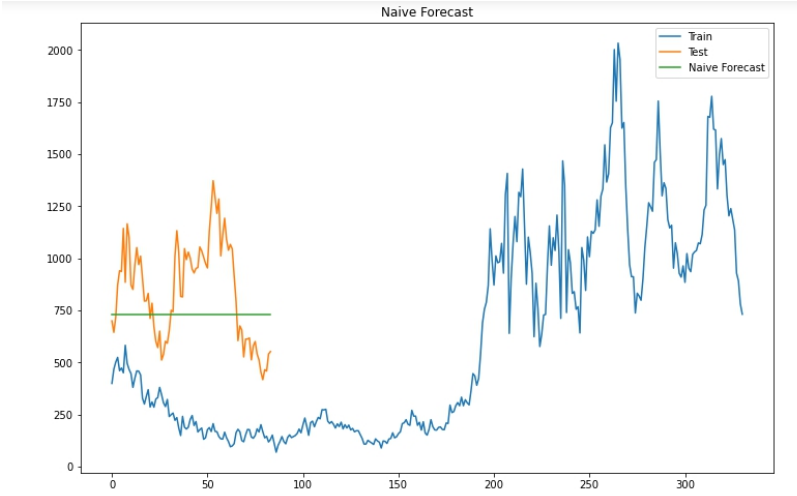


Fig. 13. Graph of Naïve Bayes

Based on the above graph output, the green line show that the prediction total case for next week. Now, the value of RMSE is calculated to determine the accuracy.

```
In [9]: rms = sqrt(mean_squared_error(test.Total_Case, y_hat.naive))
        print("Mean squared error: %.2f" %rms)

        Mean squared error: 260.32
```

Fig. 14. RMSE coding for Naïve Bayes

B. *Simple Exponential Smoothing*

Simple exponential smoothing may be sensible to attach larger weights to more recent observations than to observations from the distant past. Forecasts are calculated using weighted averages where the weights decrease exponentially as observations come from further in the past, the smallest weights are associated with the oldest observations.

$$\hat{y}_{t+1|t} = \alpha * y_t + (1-\alpha) * \hat{y}_{t|t-1}$$

So essentially, we've got a weighted moving average with two weights: α and $1-\alpha$, where $0 \leq \alpha \leq 1$ is the **smoothing** parameter.

The one-step-ahead forecast for time T+1 is a weighted average of all the observations in the series y1,…,yT. The rate at which the weights decrease is controlled by the parameter α.

As we can see, $1-\alpha$ is multiplied by the previous expected value $\hat{y}$ x−1 which makes the expression recursive. And this is why this method is called Exponential. The forecast at time t+1 is equal to a weighted average between the most recent observation yt and the most recent forecast $\hat{y}$ t|t−1.

Below is the coding for Simple Exponential Smoothing (SES). The smoothing level used is 0.6 because it will generate a better model for SES.

```
In [8]: from statsmodels.tsa.api import ExponentialSmoothing

In [9]: from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing
        y_hat_avg = testing.copy()
        fit2 = SimpleExpSmoothing(np.asarray(training['Total_Case'])).fit(smoothing_level=0.6,optimized=False)
        y_hat_avg['SES'] = fit2.forecast(len(testing))
        plt.figure(figsize=(18,8))
        plt.plot(training['Total_Case'], label='Train')
        plt.plot(testing['Total_Case'], label='Test')
        plt.plot(y_hat_avg['SES'], label='SES')
        plt.legend(loc='best')
        plt.title("Simple Exponential Smoothing")
        plt.show()
```

Fig. 15. Coding for Simple Exponential Smoothing

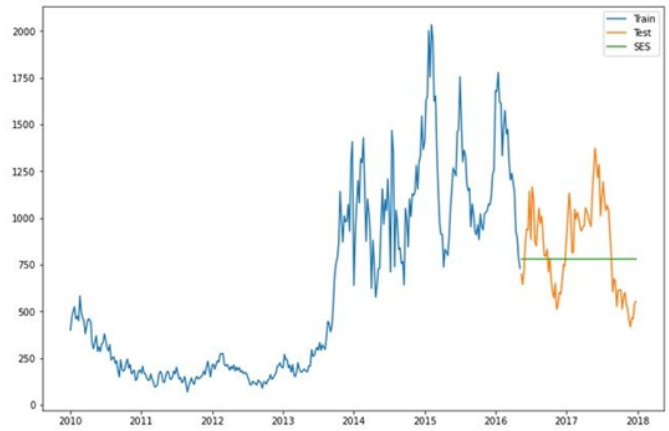


Fig. 16. Graph of Smooth Exponential Smoothing

Based on the output graph, it can be observed that the green line in Graph of SES indicates weighted average. Now we will calculate RSME to determine accuracy.

```
In [10]: from sklearn.metrics import mean_squared_error
         from math import sqrt
         rms = sqrt(mean_squared_error(testing.Total_Case, y_hat_avg.SES))
         print(rms)

         243.90173853417286
```

Fig. 17. RMSE coding for Simple Exponential Smoothing

C. *Holt's Winter Seasonal Method*

Holt-Winters is a considerable option when there is seasonality in factor. Seasonality basically means there is variations throughout the year. In our case it is weekly. In this Holt's method, it comprises the forecast equation and 3 smoothing equations. one for the level ℓt, one for trend bt and one for the seasonal component denoted by st, with smoothing parameters α, β and γ. s is the length of the seasonal cycle, for $0 \le \alpha \le 1$, $0 \le \beta \le 1$ and $0 \le \gamma \le 1$.

$$\begin{aligned}
\text{level} \quad L_t &= \alpha(y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}); \\
\text{trend} \quad b_t &= \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}, \\
\text{seasonal} \quad S_t &= \gamma(y_t - L_t) + (1-\gamma)S_{t-s} \\
\text{forecast} \; F_{t+k} &= L_t + kb_t + S_{t+k-s},
\end{aligned}$$

Fig. 17. Formula of Holt-Winters Seasonal Method

The first equation is called level. This calculates the weighted average between the adjusted observations seasonally and the non-seasonal forecast for time t. The second equation is called trend and it's the Holt's linear model method. The Holt's linear model and the third equation is called seasonal. here it shows a weighted average between current seasonal index and seasonal index of the same season but from a different period, in this case in different weeks. And finally, there is the forecast equation that is used to forecasting of the dengue prediction.

The coding for the Holt's Winter Seasonal Method is very basic as it starts with the import of libraries and data and a direct coding involving the four mathematical equations presented earlier. Below is the coding to model the dataset.

```
In [19]: # Importing the libraries
         import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd

In [20]: # Importing the dataset
         training = pd.read_csv('TRAINING.csv',index_col='Week Start Date',parse_dates=True)
         testing = pd.read_csv('FEATURE.csv',index_col='Week Start Date',parse_dates=True)

In [22]: from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt

In [23]: import statsmodels.api as sm

In [24]: y_hat_avg1 = testing.copy()
         fit11 = ExponentialSmoothing(np.asarray(training['Total_Case']) ,seasonal_periods=7 ,trend='add', seasonal='add',).fit()
         y_hat_avg1['Holt_Winter'] = fit11.forecast(len(testing))
         plt.figure(figsize=(16,8))
         plt.plot( training['Total_Case'], label='Train')
         plt.plot(testing['Total_Case'], label='Test')
         plt.plot(y_hat_avg1['Holt_Winter'], label='Holt_Winter')
         plt.legend(loc='best')
         plt.show()
```

Fig. 18. Coding of Holt's Seasonal Winter Method

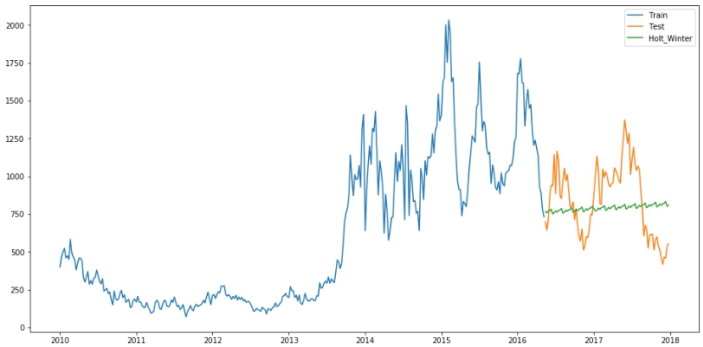


Fig. 19. Graph of Holt's Seasonal Winter Method

Based on the graph output, the green line is the forecast model to predict dengue predictions. Now we calculate the RMSE so we can determine the accuracy.

```
In [25]: from sklearn.metrics import mean_squared_error
         from math import sqrt
         rms = sqrt(mean_squared_error(testing.Total_Case, y_hat_avg1.Holt_Winter))
         print(rms)

         245.32695939960783
```

Fig. 20. RMSE coding for Holt's Seasonal Winter Method

D. LSTM

The first step is to load the dataset and then perform data preprocessing. As a thorough data preprocessing has been done during Data Management part, the only thing needed to do now is remove an irrelevant attribute, which is the Week Start Date, as it will not be used for the LSTM later. Next, we rename the columns to remove space in the column names (such as "Total Case" to "total_case") as some dependencies in later parts of the code won't accept variable names with space.

```
from math import sqrt
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

Fig. 21. The code to import the packages and dependencies

```
#read datset
dataset = pd.read_csv('processedData.csv', index_col=[0,1,2])



#drop columns
dataset.drop('Week Start Date', axis=1, inplace=True)

#rename columns
dataset = dataset.rename(columns={"Total Case": "total_cases",
                    "Max Temperature": "max_temperature",
                    "Min Temperature": "min_temperature",
                    "Average Temperature": "avg_temperature",
                    "Average Precipitation": "avg_precipitation",
                    "Wind Speed": "wind_speed"})
```

Fig. 22. The code to read the dataset and rename the columns

Furthermore, the next step is to define a function that will convert the time series data into a supervised learning dataset [6]. This means a new column is created which contains the Total Cases data for the next time step. The code for this part is as followed:

```
[ ] #prepare data for LSTM
    #convert seriest to supervised learning
    def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
        n_vars = 1 if type(data) is list else data.shape[1]
        df = pd.DataFrame(data)
        cols, names = list(), list()

        #input sequence (t-n, ...t-1)
        for i in range(n_in, 0, -1):
            cols.append(df.shift(i))
            names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

        #forecast sequence (t, t+1, ..., t+n)
        for i in range(0, n_out):
            cols.append(df.shift(-i))
            if i == 0:
                names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
            else:
                names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]

        #combine all together
        agg = pd.concat(cols, axis=1)
        agg.columns = names

        #drop rows with NaN values
        if dropnan:
            agg.dropna(inplace=True)
        return agg
```

Fig. 23. The code to define the series_to_supervised function to prepare data for LSTM

Next, we need to normalize the values in our dataset to a range of 0 to 1. This is done using the LabelEncoder to normalize labels into values between 0 and n-classes and the MinMaxScaler package to normalize the values in data frame. We also must ensure that all the values are float. After that, we call the series_to_supervised function that we declared earlier to convert the normalized dataset into supervised learning dataset. However, once done, we must remove some extra columns that we do not want to predict in the LSTM (the columns are generated by the series_to_supervised function).

```
[ ] values = dataset.values

    # integer encode direction
    encoder = LabelEncoder()
    values[:, 4] = encoder.fit_transform(values[:, 4])

    #ensure all data is float
    values = values.astype('float32')

    #normalize features
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled = scaler.fit_transform(values)

    #frame as supervised learning
    reframed = series_to_supervised(scaled, 1, 1)

    #drop columns that we don't wanna predict
    reframed.drop(reframed.columns[[11, 12, 13, 14, 15, 16, 17, 18, 19]], axis=1, inplace=True)
    #reframed.drop(reframed.columns[[7, 8, 9, 10, 11]], axis=1, inplace=True)

    reframed.head()
```

Fig. 24. The code to normalize the data and invoking the series_to_supervised function.

And the results of this step are shown below. Note that the last columns are the total cases for the next time step (var1(t)).

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var7(t-1)	var8(t-1)	var9(t-1)	var10(t-1)	var1(t)
1	0.168110	0.405797	0.56	0.365781	0.763636	0.492578	0.315833	0.750799	0.952810	0.629491	0.202751
2	0.202751	0.275362	0.36	0.244837	0.472727	0.377868	0.225049	0.758786	0.946804	0.561862	0.220071
3	0.220071	0.275362	0.40	0.427729	0.046753	0.201080	0.354993	0.774760	0.937366	0.539037	0.231788
4	0.231788	0.434783	0.38	0.365781	0.514286	0.348178	0.381064	0.741214	0.936937	0.622909	0.199185
5	0.199185	0.565217	0.54	0.451327	0.522078	0.565452	0.361454	0.720447	0.942943	0.597790	0.205807

Fig. 25. The dataset after being normalized and returned from the series_to_supervised function

After that, we define the model and fit the data into the LSTM model. For the dataset, we use 332 as the values of the train data (which is roughly 80% of total data) and we assign this into variable n_train_hours. Once the train and test set are defined, we split both sets into input and output. Again, note that there are 10 columns for input and one for output (which is obtained from the series_to_supervised function). Finally, the input is reshaped into 3-dimensional shape consisting of sample, timesteps, and feature

```
#fit the model

values = reframed.values
n_train_hours = 332
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]

#split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]

#reshape the input into 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))

print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

Fig. 26. The code to split the data into test and train set

The following is the code to fit the model and plotting the loss of train and test set. For the LSTM, 50 neurons are used in the first hidden layer and 1 neuron is used for the output layer, to replicate the experiment done by Brownlee [5]. Moreover, 50 epochs with 72 batch sizes are used for training. Figure 28 shows the graph of the train and test loss over 50 epochs.

```
#fit the model
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

#fit the network
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=(test_X, test_y), verbose=2, shuffle=False)

#plot history
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show(0)
```

Fig. 27 The code to define the LSTM model and its hyperparameter for training and testing

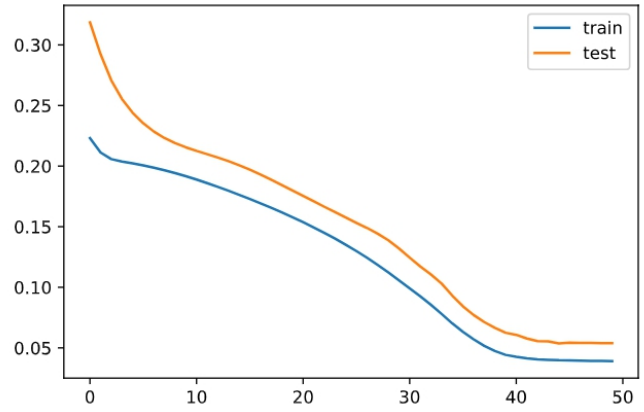


Fig. 28. The graph showing the train and test loss over 50 epochs.

In addition, the next step is to calculate the RMSE of this model. First, the model scale must be inverted as the current value is normalized in values between 0 and 1. Once done, RMSE is calculated. The RMSE for this model is 131.457.

Finally, the graph for the prediction for the test set is plotted. The results show the model can predict the dengue case. Although the prediction is still off the actual value, as most predictions are either lower or higher than the actual value, it still manages to predict the jump or fall in the total case, as shown in the graph.

```
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))

#invert scaling for forecast
inv_yhat = np.concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:, 0]

#invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = np.concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:, 0]

#calculate the RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
```

Fig. 29. The code to obtain the RMSE of this model

```
plt.plot(inv_y, label='actual')
plt.plot(inv_yhat, label='prediction')
plt.legend()
plt.show()
```

Fig. 30. The code to plot the graph of the actual and predicted cases of the test set

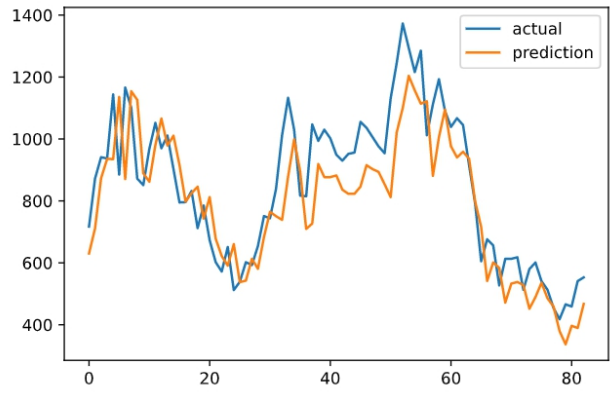


Fig. 31. The graph of the actual and predicted cases of the test set.

E. Model Evaluation

TABLE 2
TABLE OF MODEL AND RMSE

MODEL	RMSE
Naive Bayes	217.21
Simple Exponential Smoothing	243.90
LSTM	131.457
Holt's Winter Seasonal Method	245.33

According to the table above, in terms of RMSE, the best model from the four models that we test are LSTM, with RMSE of 131.457 compared to the second highest which is 217.21 (Naive Bayes). One possible explanation of why LSTM performs the best is due to its capability to store long-term memory which is used to predict the next point in the prediction. Besides, it is a neural network, so it can continuously tune its result during the training in the 50 epochs.

Moreover, according to the plot graph of this mode on the test set, it shows that it manages to closely resemble the actual values of the graph. This is important, as even though the value is off and not accurate, it can predict that the outbreak (significant rise of cases) will occur so relevant can use this data to take necessary actions to curb the dengue cases.

However, it must be noted that deep learning does suffer from overfitting problem, especially as we do not know what is happening in the 'black box'. Besides, its performance significantly different from another model. Although the train and test loss graph indicate overfitting might not occur (as the test lost is higher than train loss), we cannot verify whether the model is overfitting or not. For future works, we can try do another deep learning model and compare the two results to see whether the performance differ much or not.

IV. CONCLUSION

This study has several limitations. First, the weather data is only obtained from a single weather station, despite Selangor contains many other weather stations. Therefore, the current weather data might not reflect the whole state as the weather at different stations might be different. Second, our method for converting data from daily readings to weekly might also not be the same with the actual date that is being used by the Ministry of Health (MOH) when publishing this data. For instance, for each year, we calculate from the first day (1st January) but the MOH might use the Monday (for example, this could be on 3rd January) as the beginning of the week. Hence, inaccuracies might occur due to this. Besides, there is nothing specified by MOH in the dataset description that we can use as a guide for conversion.

For future works, first, we will obtain weather data for more weather stations for each state and averaged that data to get a more accurate data. In addition, we can expand this study to not only dengue cases in Selangor, but also to other states. Moreover, we can also try to get the data for cases by districts instead of by state. Besides, feature selection can be applied to select the best set of features that are used for the modeling. This will improve the results of our algorithms as only the significant features are being used.

APPENDIX

A. What Went Wrong

First, we planned to do web scraping to get the weather dataset. However, we found out late in our project that the websites that we have identified does not allow web scraping as they render data using JavaScript hence, we cannot get the data on the website by using Python package such as BeautifulSoup. As a result, we have to resort to downloading data manually from Visual Crossing website. This is also the reason why we only get the data from one weather station as the website limits how many rows of data that can be downloaded in a day for free user.

B. What We Might Do Differently Next Time

The first thing that we can do differently is try using the API to access the weather data, such as using the Malaysia Meteorology Department API. Currently we do not use that API due to lack of documentation on how to use it but in the future, we can try to explore that API, or find another API for the historical weather dataset.

Next, we can expand this study to other state. In addition, we can also try to get the data by district as the relevant party can take better action to curb this disease and know which district the hotspot of the dengue is to take any necessary actions. However, from our knowledge, the by district data are not available anywhere in a structured format online, and mostly available in a form of Facebook posts in form of image, which is handled by the health authority in that district. Hence, more steps are required to get and extract those data and Computer Vision knowledge is needed to automate the data extraction process.

Moreover, we can also add other data such as population and vegetation, as both of this variable affect dengue cases, as high population and high dense area usually means more cases compared to low dense area. As a result, a more accurate result might be achieved due to more factors that are used for the modeling.

In addition, a dashboard can be developed to illustrate the results and do the modeling in real time, which can be used by the health authority to continually monitor the prediction. Also,

feature selection can be performed to remove unrelated features and improve the results of our models.

Additionally, the visualization of our results can be improved by putting all the graphs of our four models into a single graph, which will make it easier for the reader to see how each model perform compared to each other. Currently, we only plot the result for each model, so it is harder to do the comparison.

ACKNOWLEDGMENT

The authors would like to thank our mentor, PROFESSOR DR SHARIFAH SAKINAH BINTI SYED AHMAD for the constant supervision as well as mentoring us throughout the progress of this project. Besides that, PROFESSOR DR SHARIFAH SAKINAH BINTI SYED AHMAD also provided us as a team with necessary information and guidance. We also would like to extend our gratitude given by our mentor in our project presentation as we have greatly improved thanks to her comments and advises.

Moreover, many thanks and appreciations go out to the co-authors that worked together in the making of this project. As a team, we worked together and gave full cooperation into developing this project. We would also like to extend many thanks to classmates and members of the BITI course for their kind co-operation and encouragement which helped us in the completion of this project.

REFERENCES

[1] Yacoub, S., & Wills, B. (2014). Predicting outcome from dengue. BMC medicine, 12, 147. https://doi.org/10.1186/s12916-014-0147-9

[2] Yuan, HY., Liang, J., Lin, PS. et al. The effects of seasonal climate variability on dengue annual incidence in Hong Kong: A modelling study. Sci Rep 10, 4297 (2020). https://doi.org/10.1038/s41598-020-60309-7

[3] Tran, Bao-Linh; Tseng, Wei-Chun; Chen, Chi-Chung; Liao, Shu-Yi. 2020. "Estimating the Threshold Effects of Climate on Dengue: A Case Study of Taiwan" Int. J. Environ. Res. Public Health 17, no. 4: 1392. https://doi.org/10.3390/ijerph17041392

[4] Olah, C. (2015). Understanding LSTM Network. Colah's Blog [Blog]. Retrieved from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[5] Brownlee, J. (2017). Multivariate Time Series Forecasting With LSTMs in Keras. Machine Learning Mastery. Retrieved from: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

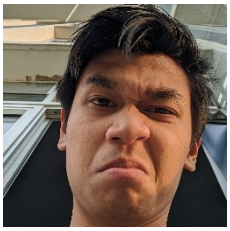
[6] Forecasting: Principles and Practice (2nd ed). 7.3 Holt-Winters' seasonal method. (n.d.). https://otexts.com/fpp2/holt-winters.html.

[7] Brownlee, J. (2017). How to Convert a Time Series to a Supervised Learning Problem in Python. Machine Learning Mastery. Retrieved from: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Mohamad Zairi Bin Abddd Ghani

Anisah Binti Kamsin

Jeya Jassvine A/P Jeyabala Sundram



Charae A/L Eh Sin