Analysis of Association between Dengue, Aedes Mosquito Specimens, and Meteorological Conditions

Thejomayi Malempati[1], Alden Jettpace[1], Saitejaswi Cherukupalli[1], Sunaina Kakollu[1], Sai Varshitha Thurpu[1], Manaswini Dasari[1], and Balakumar Rahul[1]
tmalempa@iu.edu, ajettpac@iu.edu, saicheru@iu.edu, skakollu@iu.edu, sthurpu@iu.edu, dasarim@iu.edu, brahul@iu.edu

[1]Indiana University-Purdue University, Indianapolis, IN 46202, USA

Abstract: This project analyzes the relationship between Dengue, Aedes-Genus Mosquitoes, and Meteorological conditions. The goal is to identify whether Meteorological conditions can predict either Mosquito populations or monthly numbers of Dengue cases, as well as establish whether or not the relationship between these conditions and Dengue is causal. We wish to show which conditions put communities at risk for Dengue so that they may take protective measures in advance, and thus fight the spread of Dengue at-risk communities.

Keywords: Dengue, Mosquitoes, Aedes, Weather, Humidity, Meteorological, Conditions, Predictors, Causation, Health, Data Analysis, Data Visualization, SQL, Python, GLM, Time-Series

Table of Contents

1. Project Scope

1.1 Introduction

Parasitic insects such as ticks or mosquitoes are often the chief spreaders of some of the most prolific diseases in the world. While the most often cited examples are the tick-borne Bubonic Plague or the famous mosquito-carried disease of Malaria, a growing epidemic of bloodborne illness over the last decades has been Dengue Fever. In Colombia, the situation is serious, whereby in week 36 of 2018, the dengue incidence rate was 96 cases per 100,000 people, with a total of 111 deaths (PubMed,2018).

According to the WHO (World Health Organization), dengue is now endemic in over 100 countries, with severity ranging from non-symptomatic to life-threatening. In hyperendemic nations like Colombia, where dengue is widespread, understanding epidemiological trends is crucial for the prevention of endemic disease (TropicalMedicine,2022).

1.2 Aim

We aim to Establish whether or not there is a correlation and thus, predictive ability between Dengue, Mosquitoes, and Meteorological Conditions.

Thus we have two research questions:

1. Do meteorological conditions have any correlation or significant predictive power concerning Aedes Female Mosquitoes?
2. Do meteorological conditions have any correlation or significant predictive power regarding the Monthly Dengue Case Count?

Based on these questions, our two possible hypotheses are

- Null Hypothesis: There is no association between Mosquitoes and Meteorological Conditions, nor Dengue and Meteorological Conditions.
- Alternate Hypothesis: There is a statistically significant association between Mosquitoes and Meteorological Conditions and/or Dengue and Meteorological Conditions, and such conditions can be used as predictors for Specimen Counts or Monthly Case counts.

1.3 Purpose

The purpose of this study is to predict the monthly number of Dengue cases based on Meteorological conditions from the immediate time of cause, 2-weeks prior, and 4-weeks prior, and compare the performance of the models to determine not just predictive power and correlation, but also whether or not the relationship between Dengue and Meteorological Conditions is causal.

The changes within these conditions will then be modeled to visualize monthly changes in Dengue Counts, and Meteorological 2 weeks before to see if they mirror the performance of Model Coefficients.

With these tests, we hope to prove or disprove an association between Meteorological conditions and Dengue, to prove or disprove such condition's viability as a predictor of monthly Dengue cases. Many locations where Dengue is prevalent are also terribly impoverished, with few means to treat Dengue cases as they arise, and hence can only truly adopt preventative measures by planning for abundant mosquito populations, and the associated conditions.

The logic is thus: Since mosquitoes have a two-weeks hatching period which could be affected by temperatures during egg laying, along with Dengue having a 2-week gestation period. Taking this logic into account, a strong correlation between 2-week prior conditions and Mosquitoes, and 4-week prior conditions and Dengue, would give probable evidence for causation.

Such evidence of correlation along with predictive models would allow communities to be forewarned of at-risk periods of Dengue such as during 'Mosquito season, and help communities prepare in advance for such seasons and hopefully lessen the spread of Dengue.

2. Methodology

2.1 Steps of the Project

Our project focuses on examining the correlation between Dengue, female Aedes Mosquitoes, and the Meteorological Conditions of Maximum Temperature, Minimum Temperature, Average Temperature, and Percentage-Humidity using database technology of SQL and data analysis and visualization tools of Python. Specifically, we used phpMyAdmin for data extraction and relational data table creation, and Python Jupyter Notebook for Correlation Analysis and Visualization.

2.2 Team Member Responsibilities

Our team held only three members with background knowledge of the tools and methods used, and responsibilities were divided based on this fact.

Team Member Name	Background	Responsibilities
Thejomayi Malempati	Bachelor's in Health Information Management, basic knowledge in SQL and Python	Project Manager, Data Collection, Data Analysis, Regression Model Integration
Alden Jettpace	Bachelor's in Math-Economics, intermediate experience in Python	Data Cleaning, SQL Joining, Data Extraction, Data Analysis, Regression Model Integration
Saitejaswi Cherukupalli	Bachelor's in Dental Surgery, slight experience in SQL and Python	Project Presentation, Finding and Citing Sources, Regression Model Integration
Sunaina Kakollu	Bachelor's in Pharmaceutical Sciences	Project Report
Sai Varshitha Thurpu	Bachelor's in Dental Surgery	Project Report
Manaswini Dasari	Bachelor's in Dental Surgery	Finding and Citing Sources, Project Presentation
Balakumar Rahul	Bachelor's in Computer Science, intermediate experience in SQL	Regression Model Integration, Visualization

3. Data Collection

The dataset was collected from Kaggle from the author David Restrepo, (https://www.kaggle.com/datasets/davidrestrepo/stratification-of-dengue-in-cauca-colombia?select=Dengue+Dataset.pdf) who collected the data from several outside sources, with Dengue data and Mosquito Specimen Data coming from the University of Cauca, and the Meteorological Data coming from GloH20, an online source of climate and weather data products. The dataset concerned the distribution of Mosquitoes and Dengue cases within the Cauca Region of Columbia, specifically the sub-regions of Patia, Piamonte, and Miranda.
Not all of the files in the dataset were used, with the files of interest being:

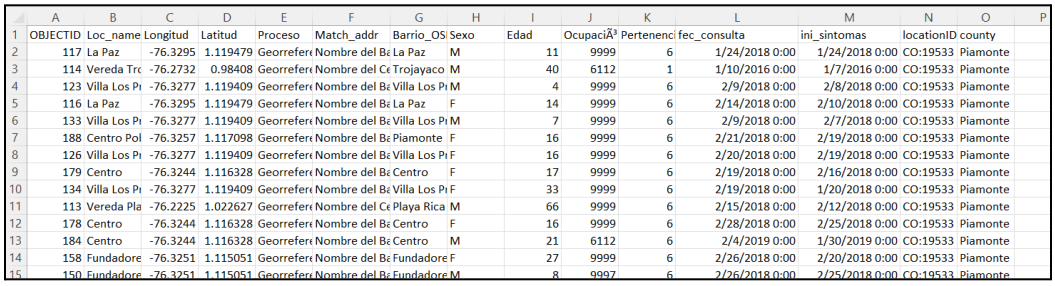
1. Cases_clean.csv: Documented Dengue cases from the years 2015 to the end of 2021
2. Entomologico_clean.csv: Measurements of Mosquito specimens over 16 unique dates in the region
3. H_mswx_2015_2022.nc: Geospatial Data of measurements of humidity in Columbia
4. Tmax_mswx_2015_2022.nc: Geospatial Data of measurements of max temperature in Columbia
5. Tmin_mswx_2015_2022.nc: Geospatial Data of measurements of min temperature in Columbia
6. Tprom_mswx_2015_2022.nc: Geospatial Data of measurements of average temperature in Columbia

The dataset also included Shapefiles for the three subregions, as well as an additional .nc file which was to be the rainfall in inches but was instead a duplicate of the H_mswx_2015_2022 data and thus could not be used.

4. Data Extraction and Storage

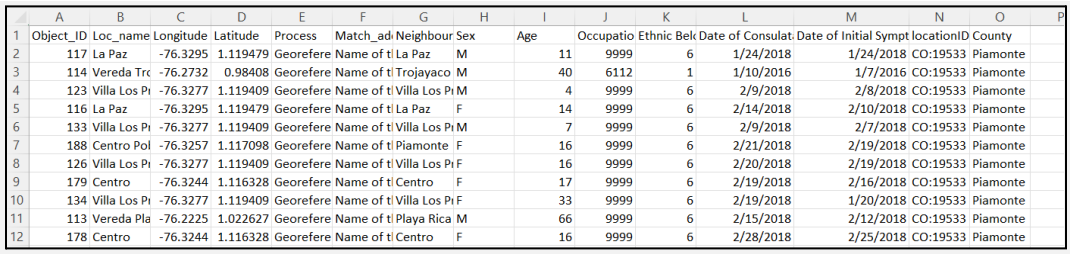
4.1 Data Cleaning

The first step before any data analysis could be done was to translate the data, as the data in most of the files were collected and stored in Spanish. The data had to also be reformatted in terms of dates, as sometimes it fell from the mm/dd/yyyy format to dd/mm/yyyy and had to be corrected. As well, it had to be text-file examined to see the true divisors which might have been fooled by Excel imaging (ie: skipped over commas, etc.).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	OBJECTID	Loc_name	Longitud	Latitud	Proceso	Match_addr	Barrio_OS	Sexo	Edad	OcupaciÃ³	Pertenenc	fec_consulta	ini_sintomas	locationID	county	
2	117	La Paz	-76.3295	1.119479	Georefere	Nombre del Ba	La Paz	M	11	9999	6	1/24/2018 0:00	1/24/2018 0:00	CO:19533	Piamonte	
3	114	Vereda Tro	-76.2732	0.98408	Georefere	Nombre del Ce	Trojayaco	M	40	6112	1	1/10/2016 0:00	1/7/2016 0:00	CO:19533	Piamonte	
4	123	Villa Los Pi	-76.3277	1.119409	Georefere	Nombre del Ba	Villa Los Pi	M	4	9999	6	2/9/2018 0:00	2/8/2018 0:00	CO:19533	Piamonte	
5	116	La Paz	-76.3295	1.119479	Georefere	Nombre del Ba	La Paz	F	14	9999	6	2/14/2018 0:00	2/10/2018 0:00	CO:19533	Piamonte	
6	133	Villa Los Pi	-76.3277	1.119409	Georefere	Nombre del Ba	Villa Los Pi	M	7	9999	6	2/9/2018 0:00	2/7/2018 0:00	CO:19533	Piamonte	
7	188	Centro Pol	-76.3257	1.117098	Georefere	Nombre del Ba	Piamonte	F	16	9999	6	2/21/2018 0:00	2/19/2018 0:00	CO:19533	Piamonte	
8	126	Villa Los Pi	-76.3277	1.119409	Georefere	Nombre del Ba	Villa Los Pi	F	16	9999	6	2/20/2018 0:00	2/19/2018 0:00	CO:19533	Piamonte	
9	179	Centro	-76.3244	1.116328	Georefere	Nombre del Ba	Centro	F	17	9999	6	2/19/2018 0:00	2/16/2018 0:00	CO:19533	Piamonte	
10	134	Villa Los Pi	-76.3277	1.119409	Georefere	Nombre del Ba	Villa Los Pi	F	33	9999	6	2/19/2018 0:00	1/20/2018 0:00	CO:19533	Piamonte	
11	113	Vereda Pla	-76.2225	1.022627	Georefere	Nombre del Ce	Playa Rica	M	66	9999	6	2/15/2018 0:00	2/12/2018 0:00	CO:19533	Piamonte	
12	178	Centro	-76.3244	1.116328	Georefere	Nombre del Ba	Centro	F	16	9999	6	2/28/2018 0:00	2/25/2018 0:00	CO:19533	Piamonte	
13	184	Centro	-76.3244	1.116328	Georefere	Nombre del Ba	Centro	M	21	6112	6	2/4/2019 0:00	1/30/2019 0:00	CO:19533	Piamonte	
14	158	Fundadore	-76.3251	1.115051	Georefere	Nombre del Ba	Fundadore	F	27	9999	6	2/26/2018 0:00	2/20/2018 0:00	CO:19533	Piamonte	
15	150	Fundadore	-76.3251	1.115051	Georefere	Nombre del Ba	Fundadore	M	8	9997	6	2/26/2018 0:00	2/25/2018 0:00	CO:19533	Piamonte	

Fig. 1. Original cases_clean.csv file



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Object_ID	Loc_name	Longitude	Latitude	Process	Match_ad	Neighbour	Sex	Age	Occupatio	Ethnic Bel	Date of Consulat	Date of Initial Sympt	locationID	County	
2	117	La Paz	-76.3295	1.119479	Georefere	Name of t	La Paz	M	11	9999	6	1/24/2018	1/24/2018	CO:19533	Piamonte	
3	114	Vereda Tro	-76.2732	0.98408	Georefere	Name of t	Trojayaco	M	40	6112	1	1/10/2016	1/7/2016	CO:19533	Piamonte	
4	123	Villa Los Pi	-76.3277	1.119409	Georefere	Name of t	Villa Los Pi	M	4	9999	6	2/9/2018	2/8/2018	CO:19533	Piamonte	
5	116	La Paz	-76.3295	1.119479	Georefere	Name of t	La Paz	F	14	9999	6	2/14/2018	2/10/2018	CO:19533	Piamonte	
6	133	Villa Los Pi	-76.3277	1.119409	Georefere	Name of t	Villa Los Pi	M	7	9999	6	2/9/2018	2/7/2018	CO:19533	Piamonte	
7	188	Centro Pol	-76.3257	1.117098	Georefere	Name of t	Piamonte	F	16	9999	6	2/21/2018	2/19/2018	CO:19533	Piamonte	
8	126	Villa Los Pi	-76.3277	1.119409	Georefere	Name of t	Villa Los Pi	F	16	9999	6	2/20/2018	2/19/2018	CO:19533	Piamonte	
9	179	Centro	-76.3244	1.116328	Georefere	Name of t	Centro	F	17	9999	6	2/19/2018	2/16/2018	CO:19533	Piamonte	
10	134	Villa Los Pi	-76.3277	1.119409	Georefere	Name of t	Villa Los Pi	F	33	9999	6	2/19/2018	1/20/2018	CO:19533	Piamonte	
11	113	Vereda Pla	-76.2225	1.022627	Georefere	Name of t	Playa Rica	M	66	9999	6	2/15/2018	2/12/2018	CO:19533	Piamonte	
12	178	Centro	-76.3244	1.116328	Georefere	Name of t	Centro	F	16	9999	6	2/28/2018	2/25/2018	CO:19533	Piamonte	

Fig. 2. Cleaned Data with formatting for SQL import.

The datasets, after being cleaned, then had to be imported into PHPMyAdmin by first creating a table for the files to be selected into, due to category-length errors that came from the titles always being shorter than the length of the string data beneath it.

To import the NC-Files into PHPMyAdmin, they had to be changed from NC-Files into acceptable CSV files, which vastly increased their size. This data can be found in Appendix A.1 as well as within the included data files [file-name].

4.2 Data Extraction

After cleaning our data, we then found the attributes of interest, namely two dependent variables and four independent variables for analysis:

- Dependent Variables
  - Month_Count: An integer representing the monthly number of Dengue cases that occurred in the same month as the current Dengue instance. A derived attribute created in Data Importing of cases_clean
  - Female_Count: An integer representing the number of female mosquitoes counted during any instance of mosquito specimen collection. As a derived variable created in PHPMyAdmin during data importing of entomologico_clean
- Independent Variables:

- Max_Temp: A float representing the max temperature in celsius (C) measured at a certain latitude-longitude combination on a certain date.
- Min_Temp: A float representing the min temperature in celsius (C) measured at a certain latitude-longitude combination on a certain date.
- Aver_Temp: A float representing the average temperature in celsius (C) measured at a certain latitude-longitude combination on a certain date.
- Humidity: A float representing the percentage humidity measured at a certain latitude-longitude combination on a certain date.

4.3 Data Importing

After cleaning the data for cases_clean and entomologico_clean, and copying the .nc files into .csv files, we then began the process of importing data into our shared database, I501saptpurkFall22grp_09_db, set up for our use by our professor. During this, we encountered problems with uploading the meteorological-data files. The sheer size of the files containing our meteorological conditions meant they could not be uploaded into PHPMyAdmin directly, instead being done through a python connection. We then indexed these files by their Date, Latitude, and Longitude to SELECT INTO a single table via NATURAL JOIN. The details of these queries can be found in Appendix A.1.

After creating the table NC_Data, we then attempted to import our Dengue Data and Mosquito data into the database but found it would error due to the data-length limitations established by the column headers being violated by the run-on data below it in certain cells. To get around this error, we created tables with the appropriate data types and acceptable lengths. While the mosquito data was uploaded with no new attributes, a unique auto-incrementing integer value was added to the Dengue Cases to uniquely identify each instance, the details of which can be found in Appendix A.3.

Our mosquito data only contained the total integer count of specimens of both male and female in the attribute ‘Individual’ and then contained the male-female distribution in the string attribute ‘Sex’. Because only female mosquitoes drink blood and thus act as a vector for dengue, we wanted to focus our tests only on the dependent variable of female Aedes mosquitoes. Thus, a function, EXTRACT_FEMALE, was created to be applied within an SQL query to extract the female counts based on the Individual-value and the structure of the Sex string. This function can be found in Appendix A.2

Finally, we then had to join the Dengue cases and the Mosquito specimens to the appropriate meteorological conditions. Since the latitude and longitude did not exactly match any of those in the NC_Data table, instead the closest by absolute distance, or the minimum sum of the absolute difference between latitude and longitude of the Dengue/Mosquito data and the meteorological NC_Data. The Mosquito data was contained in the table MOS_MET which contained the meteorological conditions, the specimen genus, and the latitude/longitude, as well as the female count extracted from the function EXTRACT_FEMALE.

The Dengue data was then joined the appropriate conditions, but into several tables based on the conditions immediately at the time of initial symptoms, 2 weeks prior, and 4 weeks prior, to see if the predictors (meteorological conditions) precede the Dengue cases to prove or disprove causal relation between the two. Three other tables of the same condition were created from intermediate tables but had less granularity by having the total scores and the average conditions per month rather than just the individual cases and their unique conditions. The specifics of this can be found in Appendix A.4

By the end, we had a total of 17-tables, seven of which were of direct use:

- DENGUE_METEOR_COUNT
- DENGUE_METEOR_COUNT_TWO
- DENGUE_METEOR_COUNT_FOUR
- DENGUE_AVER
- DENGUE_TWO_AVER
- DENGUE_FOUR_AVER
- MOS_MET

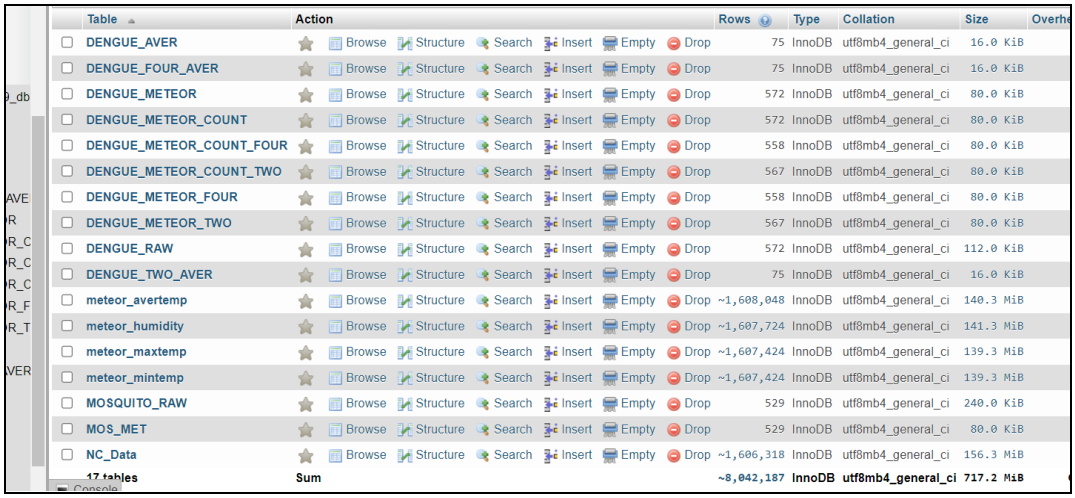
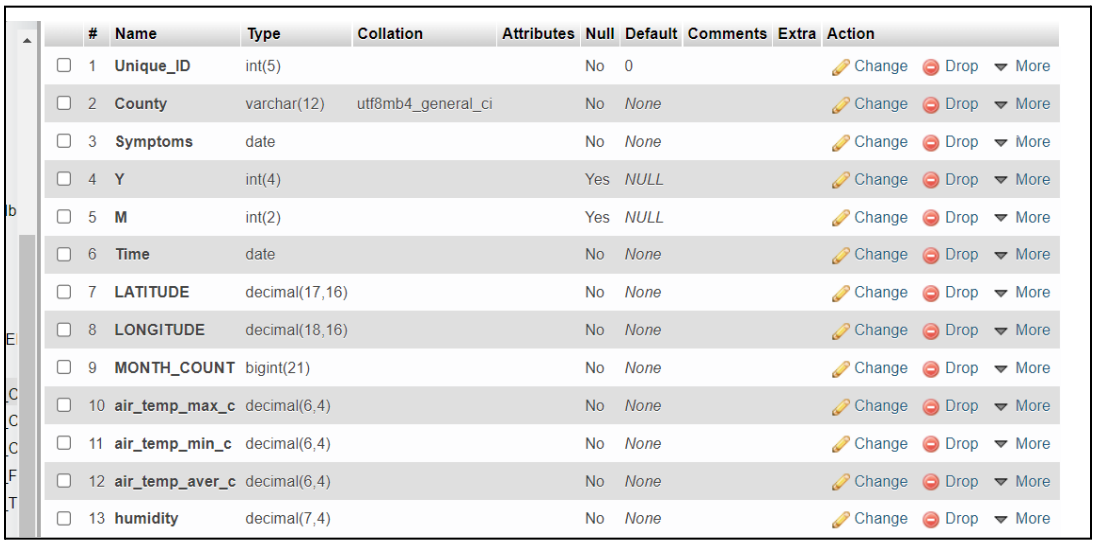


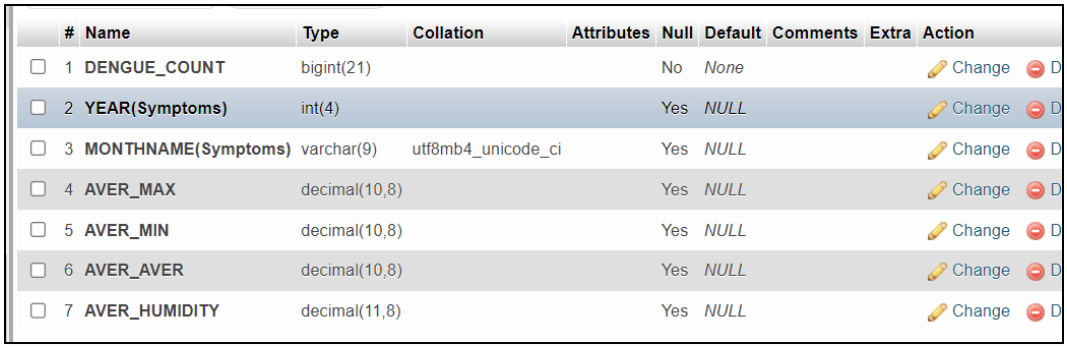
Table	Action	Rows	Type	Collation	Size	Overhe
DENGUE_AVER	Browse Structure Search Insert Empty Drop	75	InnoDB	utf8mb4_general_ci	16.0 KiB	
DENGUE_FOUR_AVER	Browse Structure Search Insert Empty Drop	75	InnoDB	utf8mb4_general_ci	16.0 KiB	
DENGUE_METEOR	Browse Structure Search Insert Empty Drop	572	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_METEOR_COUNT	Browse Structure Search Insert Empty Drop	572	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_METEOR_COUNT_FOUR	Browse Structure Search Insert Empty Drop	558	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_METEOR_COUNT_TWO	Browse Structure Search Insert Empty Drop	567	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_METEOR_FOUR	Browse Structure Search Insert Empty Drop	558	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_METEOR_TWO	Browse Structure Search Insert Empty Drop	567	InnoDB	utf8mb4_general_ci	80.0 KiB	
DENGUE_RAW	Browse Structure Search Insert Empty Drop	572	InnoDB	utf8mb4_general_ci	112.0 KiB	
DENGUE_TWO_AVER	Browse Structure Search Insert Empty Drop	75	InnoDB	utf8mb4_general_ci	16.0 KiB	
meteor_avertemp	Browse Structure Search Insert Empty Drop	~1,608,048	InnoDB	utf8mb4_general_ci	140.3 MiB	
meteor_humidity	Browse Structure Search Insert Empty Drop	~1,607,724	InnoDB	utf8mb4_general_ci	141.3 MiB	
meteor_maxtemp	Browse Structure Search Insert Empty Drop	~1,607,424	InnoDB	utf8mb4_general_ci	139.3 MiB	
meteor_mintemp	Browse Structure Search Insert Empty Drop	~1,607,424	InnoDB	utf8mb4_general_ci	139.3 MiB	
MOSQUITO_RAW	Browse Structure Search Insert Empty Drop	529	InnoDB	utf8mb4_general_ci	240.0 KiB	
MOS_MET	Browse Structure Search Insert Empty Drop	529	InnoDB	utf8mb4_general_ci	80.0 KiB	
NC_Data	Browse Structure Search Insert Empty Drop	~1,606,318	InnoDB	utf8mb4_general_ci	156.3 MiB	
17 tables	Sum	~8,042,187	InnoDB	utf8mb4_general_ci	717.2 MiB	

Fig. 3. All tables within Database



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Unique_ID	int(5)			No	0			Change Drop More
2	County	varchar(12)	utf8mb4_general_ci		No	None			Change Drop More
3	Symptoms	date			No	None			Change Drop More
4	Y	int(4)			Yes	NULL			Change Drop More
5	M	int(2)			Yes	NULL			Change Drop More
6	Time	date			No	None			Change Drop More
7	LATITUDE	decimal(17,16)			No	None			Change Drop More
8	LONGITUDE	decimal(18,16)			No	None			Change Drop More
9	MONTH_COUNT	bigint(21)			No	None			Change Drop More
10	air_temp_max_c	decimal(6,4)			No	None			Change Drop More
11	air_temp_min_c	decimal(6,4)			No	None			Change Drop More
12	air_temp_aver_c	decimal(6,4)			No	None			Change Drop More
13	humidity	decimal(7,4)			No	None			Change Drop More

Fig. 4. Structure of DENGUE_METEOR_COUNT in PHPMyAdmin



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	DENGUE_COUNT	bigint(21)			No	None			Change D
2	YEAR(Symptoms)	int(4)			Yes	NULL			Change D
3	MONTHNAME(Symptoms)	varchar(9)	utf8mb4_unicode_ci		Yes	NULL			Change D
4	AVER_MAX	decimal(10,8)			Yes	NULL			Change D
5	AVER_MIN	decimal(10,8)			Yes	NULL			Change D
6	AVER_AVER	decimal(10,8)			Yes	NULL			Change D
7	AVER_HUMIDITY	decimal(11,8)			Yes	NULL			Change D

Fig. 5. Structure of DENGUE_AVER

5. Data Analysis

Python was used for data analysis, first by extrapolating Mosquito Female Aedes count numbers for each Dengue case, then analyzed both as dependent variables using Linear Regression and Poisson Generalized Linear Regression.

5.1 Steps to Test Distribution of Data

Step 1:

For analysis, we connected to phpMyAdmin SQL through Python Jupyter Notebook, and imported the tables MOS_MET, DENGUE_METEOR_COUNT, DENGUE_METEOR_COUNT_TWO, DENGUE_METEOR_COUNT_FOUR

```
import MySQLdb
import pandas as pd
import numpy as np

groupvars = {}
with open("InformaticsGroup9_AL_Login.txt") as myfile:
    for line in myfile:
        name, var = line.partition(":")[::2]
        groupvars[name.strip()] = var.strip()

conn = MySQLdb.connect(host="localhost", user=groupvars['DB username'],
                       passwd=groupvars['DB password'], db=groupvars['DB databasename'])
cursor = conn.cursor()
```

```
cursor.execute('SELECT * FROM DENGUE_METEOR_COUNT;')
field_names = [i[0] for i in cursor.description]
dengue_month = cursor.fetchall()
DENGUE_METEOR = pd.DataFrame(dengue_month, columns = field_names)
#print(DENGUE_METEOR)
DENGUE_METEOR.head(5)
```

Fig. 6. Python Code using pandas to extract and store DENGUE_METEOR_COUNT

	Unique_ID	County	Symptoms	Y	M	Time	LATITUDE	LONGITUDE	MONTH_COUNT	air_temp_max_c	air_temp_min_c	air_temp_av
0	1	Piamonte	2018-01-24	2018	1	2018-01-24	1.1194790000000000	-76.3295370000000000	5	26.8125	17.0625	22.3
1	2	Piamonte	2016-01-07	2016	1	2016-01-07	0.9840800000000000	-76.2731660000000000	30	32.1250	23.1250	26.9
2	3	Piamonte	2018-02-08	2018	2	2018-02-08	1.1194090000000000	-76.3277190000000000	14	29.0625	23.0000	25.6
3	4	Piamonte	2018-02-10	2018	2	2018-02-10	1.1194790000000000	-76.3295370000000000	14	32.7500	20.8750	26.2
4	5	Piamonte	2018-02-07	2018	2	2018-02-07	1.1194090000000000	-76.3277190000000000	14	30.8125	21.6250	25.4

Fig. 7. Head of DENGUE_METEOR_COUNT

```
cursor.execute('SELECT * FROM DENGUE_METEOR_COUNT_TWO;')
field_names = [i[0] for i in cursor.description]
dengue_month = cursor.fetchall()
DENGUE_METEOR_TWO = pd.DataFrame(dengue_month, columns = field_names)
```

```
#Dengue with 4-week prior meteorlogical conditions
cursor.execute('SELECT * FROM DENGUE_METEOR_COUNT_FOUR;')
field_names = [i[0] for i in cursor.description]
dengue_month = cursor.fetchall()
DENGUE_METEOR_FOUR = pd.DataFrame(dengue_month, columns = field_names)
```

```
#Mosqutito Data
cursor.execute('SELECT * FROM MOS_MET;')
field_names = [i[0] for i in cursor.description]
mos = cursor.fetchall()
MOS_MET = pd.DataFrame(mos, columns= field_names)
```

Fig. 8. Extraction of Tables MOS_MET, DENGUE_METEOR_COUNT_TWO, DENGUE_METEOR_COUNT_FOUR

```
MOS_MET = MOS_MET[MOS_MET['GENUS'] == 'Aedes']

x_std = np.std(MOS_MET['FEMALE_COUNT'])
x_mean = np.mean(MOS_MET['FEMALE_COUNT'])

MOS_MET = MOS_MET[MOS_MET['FEMALE_COUNT'] <= x_mean + 4*x_std]
MOS_MET = MOS_MET[MOS_MET['FEMALE_COUNT'] >= x_mean - 4*x_std]
```

Fig. 9. Extracting only Aedes Genus Mosquitoes from MOS_MET data frame, and removing outliers.

Step 2:

After successfully extracting the data, we must prove whether or not the data is normally distributed in order to decide which model to use for prediction. Since Dengue and Mosquito are both integer counts which can be zero but not negative, we have reason to assume that if their distributions are not normal, then they are Poisson Distributions.

To check normality, we plotted the shape of the data via histograms. The shapes indicated that the distributions were not normal, being highly skewed and following a Poisson distribution shape, especially in the MOS_MET table which most closely resembled a Poisson distribution.

```
import matplotlib.pyplot as plt

plt.hist(MOS_MET['FEMALE_COUNT'], bins = [0,1,2,3,4,5,6,7,8])
```

Fig.10. Code to create Mosquito histogram using Matplotlib

```
Out[11]: (array([ 73., 128.,  28.,   8.,   4.,   4.,   3.,   0.]),
          array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
          <BarContainer object of 8 artists>)
```

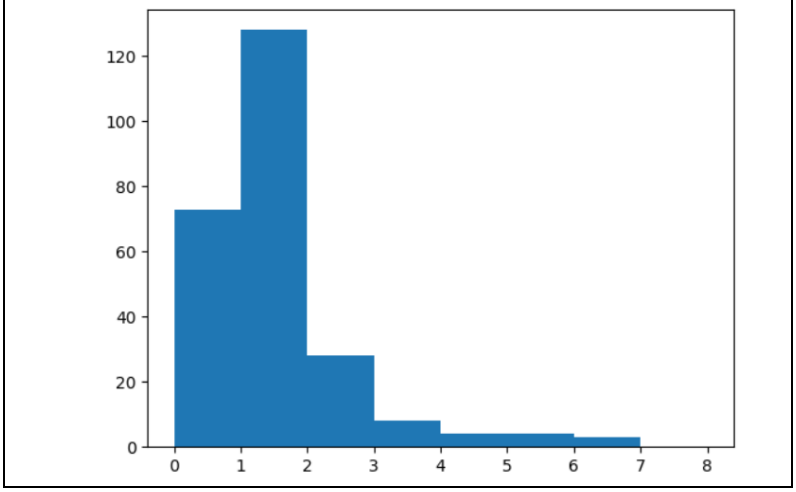


Fig.11. Histogram visualizes if Mosquito Data is normal.

```
plt.hist(DENGUE_METEOR['MONTH_COUNT'])

(array([78., 92., 88., 74., 89., 19., 20., 23., 27., 62.]),
 array([ 1. ,  4.1,  7.2, 10.3, 13.4, 16.5, 19.6, 22.7, 25.8, 28.9, 32. ]),
 <BarContainer object of 10 artists>)
```

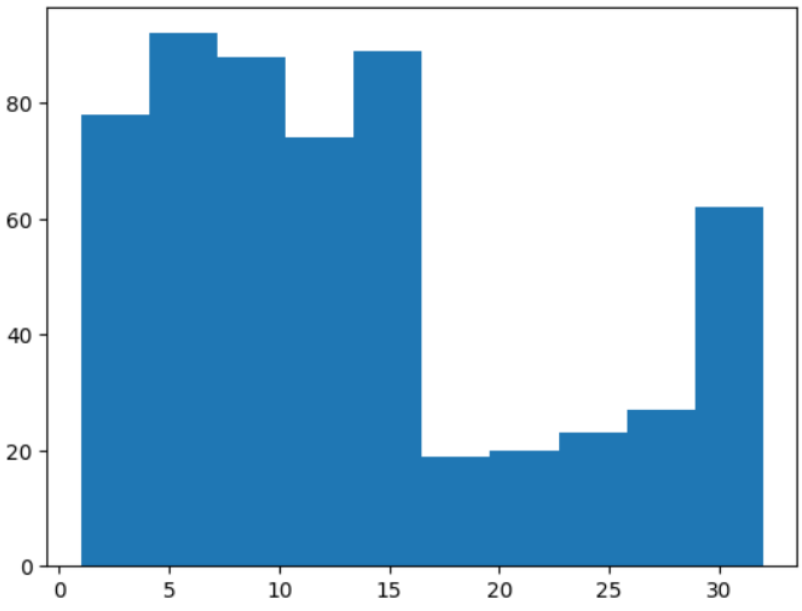


Fig.12. Code to create histogram and histogram itself, visualizing whether Monthly Dengue Count is Normally Distributed

5.2 Project Challenges

Working together as a team proved difficult, due to none of us having any experience in research projects or team projects, and only three of us have any background in statistics and data science. So establishing responsibilities, choosing data, and keeping in regular contact with each other proved to be an uphill task. Working together was further complicated by differences in schedules including in labs. Thejomayi Malempati and Alden Jettpace stepped up to schedule group meetings, with Thejomayi also taking on the extra responsibility of being the main line of communication between the group and outside sources of advice.

Cleaning and importing the data proved to be especially challenging. The data in question was divided into several sub-files, with only six of them being of any real interest. Most of the files were in Spanish and had an inconsistent format, so we had a hard time translating and then appropriately formatting, most commonly in data-time factors. The meteorological conditions were contained in NC files and as such had to be converted into CSV files.

The nature of the data also affected our method of importing the data to PHPMyAdmin. The meteorological-condition files were far too large to import directly, so instead tables had to be made within and then selected from a python-SQL connection. The appropriate meteorological conditions then had to be applied to each Dengue case and mosquito specimen count based on matching by dates and the closest latitude-longitude combination since there were no exact matches in regards to geographic location.

Alongside these challenges, our ability to work with the data proved limited thanks to certain factors unforeseen. One portion of the NC-Files, which was to be rainfall data, could not be used as it was instead a copy of the humidity data. And our mosquito data proved to have little use since it was taken only throughout 16 unique dates and the ranges of the latitude-longitude of the meteorological data proved to be wider than those of the Mosquito data, so there was too little variation to model mosquito data based on Meteorological Conditions, leading to extrapolating the specimen counts on the only remaining shared factors: Latitude and Longitude.

5.3 Descriptive Statistics

5.3.1 Normality Testing

The entire data set had the data type as an integer.

```
dengue_count_df = dengue_count_df.astype('category')
wd_column = dengue_count_df.select_dtypes(['category']).columns
dengue_count_df[wd_column] = dengue_count_df[wd_column].apply(lambda x: x.cat.codes)
print(dengue_count_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75 entries, 0 to 74
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DENGUE_COUNT    75 non-null     int8
 1   AVER_MAX        75 non-null     int8
 2   AVER_MIN        75 non-null     int8
 3   AVER_AVER       75 non-null     int8
 4   AVER_HUMIDITY   75 non-null     int8
 5   MONTH           75 non-null     int8
dtypes: int8(6)
memory usage: 578.0 bytes
None
```

Fig.13. Category of dataset

The distribution shows that our dengue count data is not normally distributed, and has a right skewness to it.

```
dengue_count_df.hist(column='DENGUE_COUNT')
```

```
]: array([[<AxesSubplot:title={'center':'DENGUE_COUNT'}>]], dtype=object)
```

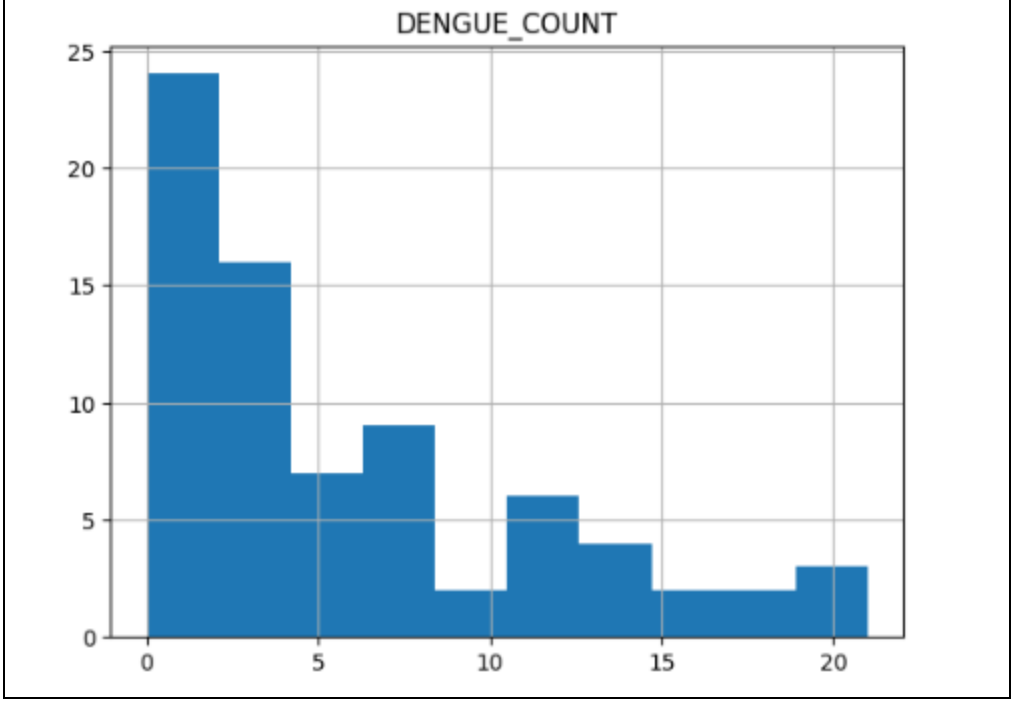


Fig.14. Dengue count distribution

```
#Normaliy Testing
from scipy.stats import normaltest

stats, p = normaltest(dengue_count_df)
print('Statistics = %.3f, p = %.3f' % (stats.all(), p.all()))
# interpret results
alpha = 0.05
if p.all() > alpha:
    print('Dengue data looks Gaussian ')
else:
    print('Dengue data does not look Gaussian ')
```

```
Statistics = 1.000, p = 1.000
```

Fig.15. Normality Testing

5.3.2 Pearson Correlation

We used Pearson Correlation to find if there was a relationship between the dengue cases and the meteorological factors, namely, temperature and humidity. We observed that the correlation between the dengue counts and the meteorological factors was not that strong with the dengue count and the maximum temperature having the closest correlation with a p-value of 0.119.

However, the value was not sufficient in determining conclusively a strong relationship.

```
import scipy
#PEARSON CORRELATION
a=dengue_count_df['DENGUE_COUNT']
x=dengue_count_df['AVER_MAX']
y=dengue_count_df['AVER_MIN']
z=dengue_count_df['AVER_AVER']
b=dengue_count_df['AVER_HUMIDITY']
```

```
maxtemp_dengue=scipy.stats.pearsonr(a,x)
maxtemp_dengue
```

```
PearsonRResult(statistic=0.1811041345114239, pvalue=0.11995713787898706)
```

```
mintemp_dengue=scipy.stats.pearsonr(a,y)
mintemp_dengue
```

```
PearsonRResult(statistic=-0.036851429352620334, pvalue=0.7536052865895588)
```

```
avgtemp_dengue=scipy.stats.pearsonr(a,z)
avgtemp_dengue
```

```
PearsonRResult(statistic=0.12918810206421652, pvalue=0.2693125302100708)
```

```
humidity_dengue=scipy.stats.pearsonr(a,b)
humidity_dengue
```

```
PearsonRResult(statistic=0.04366675998758727, pvalue=0.7098994656772776)
```

Fig.16. Results of Pearson's Correlation

Based on the Pearson Correlation results we can say that we fail to reject the Null Hypothesis mentioned below as most of the p-value is statistically significant.

- Null Hypothesis: There is no association between Mosquitoes and Meteorological Conditions, nor Dengue and Meteorological Conditions.
- Alternate Hypothesis: There is a statistically significant association between Mosquitoes and Meteorological Conditions and/or Dengue and Meteorological Conditions, and such conditions can be used as predictors for Specimen Counts or Monthly Case counts.

5.4 Test for Better Mosquito Extrapolation Model

To test our hypotheses, we need to extrapolate mosquito data to have several female mosquitoes associated with each Dengue Case, and thus each Dengue Count.

Originally we planned to directly train a model based on the Mosquito Data Based on the meteorological conditions, but an unforeseen problem involving the lack of variance in days and by extension a lack of variation in the meteorological conditions for the Mosquito cases meant instead we trained on the only remaining common attribute: Latitude and Longitude.

However, the latitude and longitude had very little correlation or covariance with the Mosquito Count, which meant the extrapolated results would always be suspect.

Since we had several options for training a non-normal integer count distribution model, we compared the results of cross-validation between Poisson GLM and SVR and compared their R^2 value as well as their mean-squared error. From the results, we determined that a Poisson GLM would be a more fitting model for extrapolation.

```
from sklearn.model_selection import KFold, cross_validate
from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR
from sklearn.linear_model import PoissonRegressor
from sklearn import metrics

x = MOS_MET.iloc[:,2:4].values
y = MOS_MET.iloc[:, 0].values

sc_x = StandardScaler()
sc_y = StandardScaler()

sc_x = sc_x.fit_transform(x)
sc_y = np.squeeze(sc_y.fit_transform(y.reshape(-1, 1)))


clf = PoissonRegressor()
kf = KFold(n_splits = 10, random_state = 1, shuffle = True)
scores = cross_validate(clf, x, y, scoring = ('r2','neg_mean_squared_error'), cv = kf, n_jobs = 1)
#print(scores)
print('Poisson Average 10-fold r^2 model value:', np.mean(scores['test_r2']))
print('Poisson Average 10-fold mean_squared_error:', abs(np.mean(scores['test_neg_mean_squared_error'])), '\n')


regressor = SVR(kernel = 'rbf')
kf = KFold(n_splits = 10, random_state = 1, shuffle = True)

scores = cross_validate(regressor, sc_x, sc_y, scoring = ('r2' ,'neg_mean_squared_error'), cv = kf, n_jobs = 1)
#print(scores)
print('SVR Average 10-fold r^2 model value:', np.mean(scores['test_r2']))
print('SVR Average 10-fold mean_squared_error:', abs(np.mean(scores['test_neg_mean_squared_error'])), '\n')

clf = PoissonRegressor()
regressor = SVR(kernel = 'rbf')
clf.fit(x,y)
regressor.fit(sc_x, sc_y)
print(clf.score(x,y))
print(metrics.mean_squared_error(y, clf.predict(x)))
print(regressor.score(sc_x, sc_y))
print(metrics.mean_squared_error(sc_y, regressor.predict(sc_x)))
```

Fig.17. Python Code for Created GLM and SVR models and comparing the average of their cross-validation results trained on Latitude and Longitude, using sklearn

```
Poisson Average 10-fold r^2 model value: -0.041179542338838124
Poisson Average 10-fold mean_squared_error: 1.2486077635077266

SVR Average 10-fold r^2 model value: -0.05104209839064557
SVR Average 10-fold mean_squared_error: 1.0262949384695481

0.005073173973573986
1.232859316712488
-0.022651436133509062
1.0226514361335093
```

Fig. 18. Average Metrics of Cross-Validated Models and fully-trained models.

```
import statsmodels.api as sm
from statsmodels.formula.api import glm
from statsmodels.graphics.api import abline_plot
x = MOS_MET.iloc[:, 2:4]
y = MOS_MET.iloc[:, 0]
x = sm.add_constant(x, prepend = False)

glm_pois = sm.GLM(y,x, family = sm.families.Poisson())
res = glm_pois.fit()
print(res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:           FEMALE_COUNT   No. Observations:                  248
Model:                            GLM   Df Residuals:                      245
Model Family:                 Poisson   Df Model:                            2
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -331.74
Date:                Fri, 02 Dec 2022   Deviance:                       272.41
Time:                        10:04:50   Pearson chi2:                     290.
No. Iterations:                     5   Pseudo R-squ. (CS):           0.009731
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
LATITUDE       0.1460      0.103      1.414      0.157      -0.056       0.348
LONGITUDE     -0.0659      0.206     -0.320      0.749      -0.470       0.338
const         -5.3557     15.920     -0.336      0.737     -36.559      25.847
==============================================================================
```

Fig.19. Statsmodel.api Poisson GLM regression results and Extrapolating Model

```
DENGUE_METEOR['mos'] = res.predict(DENGUE_METEOR.iloc[:, 6:8])
DENGUE_METEOR_TWO['mos'] = res.predict(DENGUE_METEOR_TWO.iloc[:, 6:8])
DENGUE_METEOR_FOUR['mos'] = res.predict(DENGUE_METEOR_FOUR.iloc[:, 6:8])
```

Fig. 20. Creating Extrapolated Mosquito Data for Latitude/Longitude of each Dengue case, for each table.

5.5 Correlation Among the Attributes

To determine if meteorological conditions have an association with Dengue, as well as if preceding conditions have a better association than immediate to examine if the relationship between the two is causal, we plotted heatmaps for the three tables.

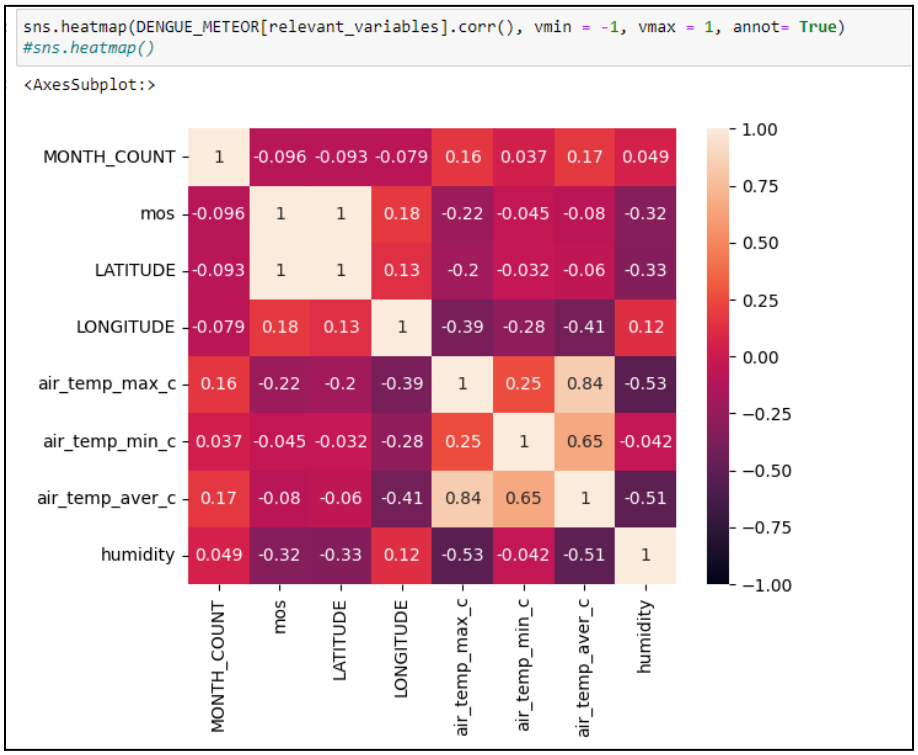


Fig. 21. Heatmap showing the correlation between Dengue and Immediate Meteorlogiocal Conditions

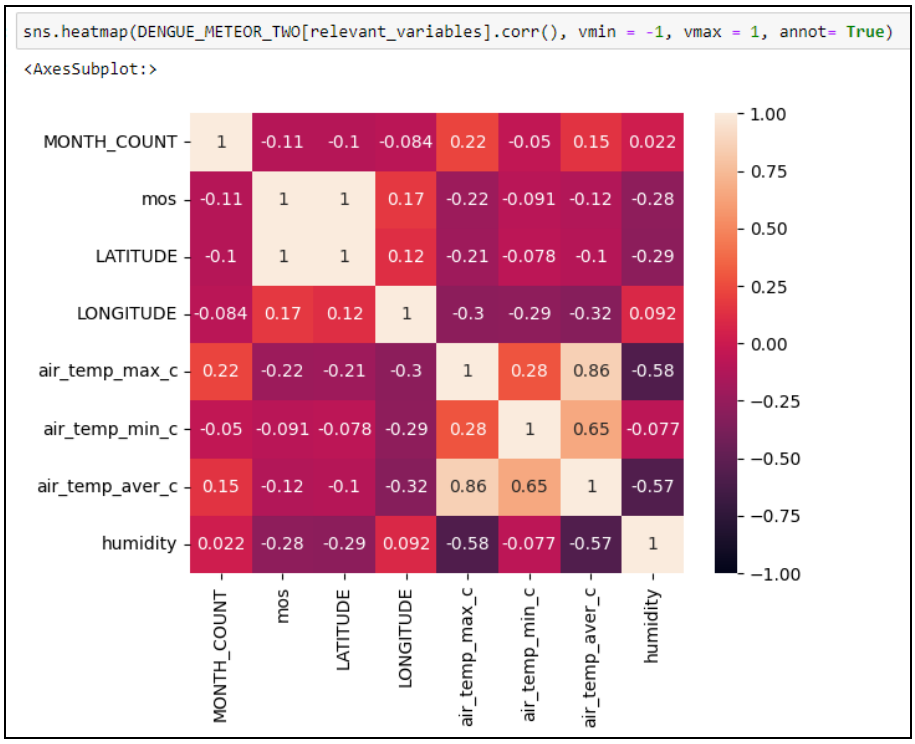


Fig. 22. Heatmap showing the correlation between Dengue and Meteorological Conditions from 2 weeks prior

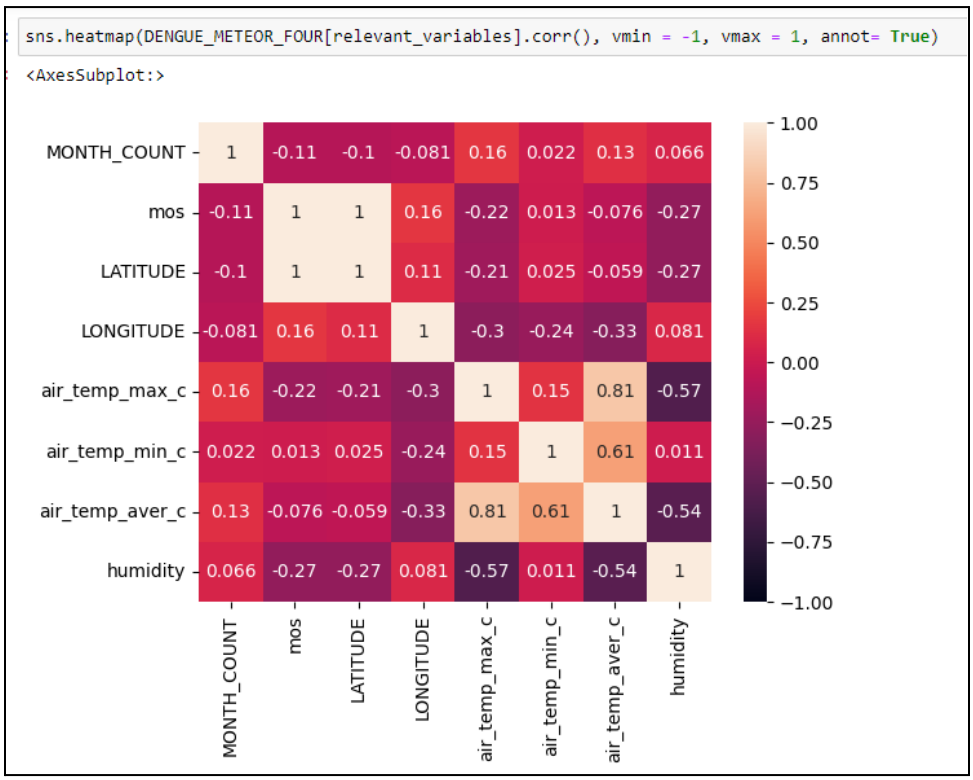


Fig. 23. Heatmap showing the correlation between Dengue and Meteorological Conditions from 4 weeks prior

Interpreting the heatmap, we found that meteorological conditions consistently had a positive correlation, with at least one condition always correlating at least (0.15), while mosquitoes consistently had a negative correlation with meteorological conditions, with at least one condition always correlating (-0.2) or lower, but this correlation is questionable thanks to the poor extrapolation quality of the mosquito values. Overall, the correlations at 2 weeks prior were determined to be slightly stronger compared to immediate and 4 weeks prior.

5.6 Linear Regression Models

Because the results of the GLM Extrapolated Mosquito values were float values ranging between 1.2 and 0.8, we decided to test the association between meteorological conditions and Aedes mosquitoes using linear regression. We measured the quality of the prediction and the association with the R^2 value, the mean absolute error of prediction, and the mean absolute percentage error of prediction to put the mean absolute error into perspective thanks to the scale of the Independent variable.

In each model, we found that Maximum temperature and Humidity consistently maintained a significant negative slope across all times, but that regression based on immediate conditions had more explained variance (0.33) and a lower mean absolute error(0.075) and mean absolute percentage error(0.074) compared to 2-weeks prior and 4-weeks prior. However, the poor extrapolation of the Mosquito values makes the overall results inconclusive.

```
from sklearn.linear_model import LinearRegression
import sklearn.metrics as skm

x = DENGUE_METEOR.iloc[:,-5:-1]
y = DENGUE_METEOR['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)

print('Coefficients for the IV of immediate: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:\n', lm_mos.coef_)
print('Intercept for Linear Model is:', lm_mos.intercept_)
print('R^2 value of linear model for Meteorlogical conditions is:', lm_mos.score(x,y))
print('Mean absolute error of prediction is:', skm.mean_absolute_error(y, predicted_mos))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(y, predicted_mos))

sns.pairplot(DENGUE_METEOR, x_vars = ['air_temp_max_c','air_temp_min_c','air_temp_aver_c','humidity'],\
             y_vars='mos', kind = 'reg')
```

```
Coefficients for the IV of immediate: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:
 [-0.04501409 -0.00856639  0.03383897 -0.01007197]
Intercept for Linear Model is: 2.4619806092827297
R^2 value of linear model for Meteorlogical conditions is: 0.33480486274487586
Mean absolute error of prediction is: 0.07470048924564004
Mean absolute percentage error is: 0.07427119011723855

<seaborn.axisgrid.PairGrid at 0x7f89492573d0>
```

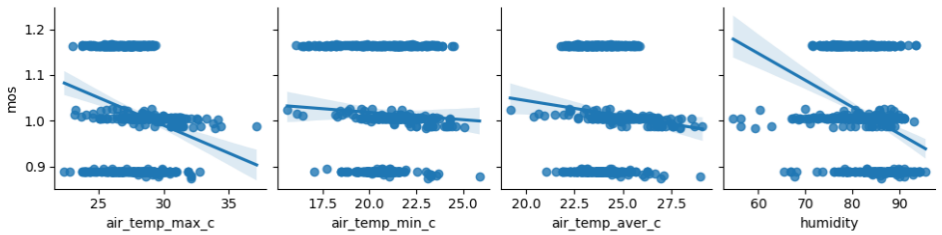


Fig. 24. Coefficients, Metrics, and Plotting of Linear Regression of Mosquitoes based on immediate meteorological conditions

```
x = DENGUE_METEOR_TWO.iloc[:,-5:-1]
y = DENGUE_METEOR_TWO['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)

print('Coefficients for the IV of 2 weeks: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:\n', lm_mos.coef_)
print('Intercept for Linear Model is:', lm_mos.intercept_)
print('R^2 value of linear model for Meteorlogical conditions is:', lm_mos.score(x,y))
print('Mean absolute error of prediction is:', skm.mean_absolute_error(y, predicted_mos))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(y, predicted_mos))

sns.pairplot(DENGUE_METEOR_TWO, x_vars = ['air_temp_max_c','air_temp_min_c','air_temp_aver_c','humidity'],\
             y_vars='mos', kind = 'reg')
```

```
Coefficients for the IV of 2 weeks: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:
 [-0.03881013 -0.00374344  0.01484335 -0.01009269]
Intercept for Linear Model is: 2.651825226665342
R^2 value of linear model for Meteorlogical conditions is: 0.31310347187146936
Mean absolute error of prediction is: 0.07648064025241524
Mean absolute percentage error is: 0.07603705247175534

<seaborn.axisgrid.PairGrid at 0x7f894ac59510>
```

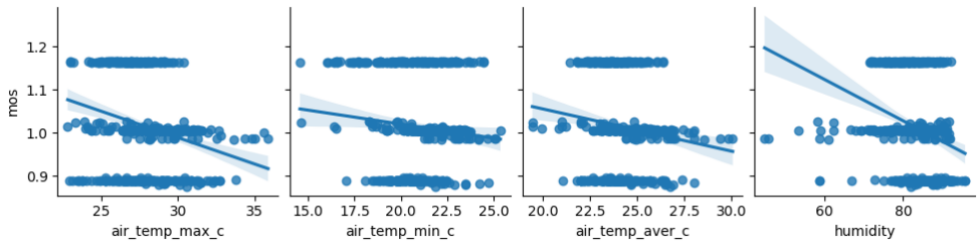


Fig. 25. Coefficients, Metrics, and Plotting of Linear Regression of Mosquitoes based on meteorological conditions 2 weeks prior

```
x = DENGUE_METEOR_FOUR.iloc[:,-5:-1]
y = DENGUE_METEOR_FOUR['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)

print('Coefficients for the IV of 4 weeks: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:\n', lm_mos.coef_)
print('Intercept for Linear Model is:', lm_mos.intercept_)
print('R^2 value of linear model for Meteorlogical conditions is:', lm_mos.score(x,y))
print('Mean absolute error of prediction is:', skm.mean_absolute_error(y, predicted_mos))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(y, predicted_mos))

sns.pairplot(DENGUE_METEOR_FOUR, x_vars = ['air_temp_max_c','air_temp_min_c','air_temp_aver_c','humidity'],\
             y_vars='mos', kind = 'reg')
```

```
Coefficients for the IV of 4 weeks: [air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity] are:
 [-0.03527703  0.00422593  0.00713434 -0.01004641]
Intercept for Linear Model is: 2.568545128476853
R^2 value of linear model for Meteorlogical conditions is: 0.28721082084092864
Mean absolute error of prediction is: 0.07740570697435908
Mean absolute percentage error is: 0.07692401702649065

<seaborn.axisgrid.PairGrid at 0x7f8949256e00>
```

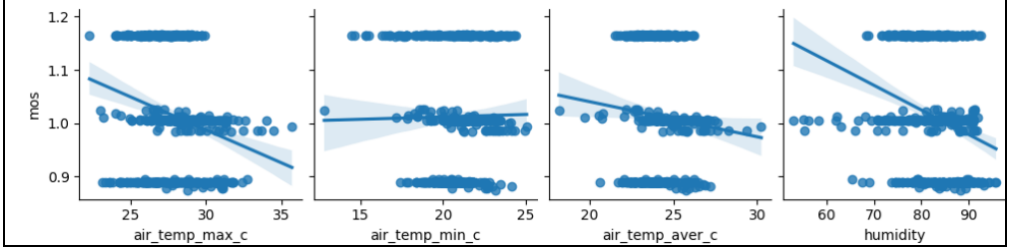


Fig. 26. Coefficients, Metrics, and Plotting of Linear Regression of Mosquitoes based on meteorological conditions 4 weeks prior

5.7 Poisson GLM

Since Dengue also was not normal and had the characteristics of a Poisson distribution, we decided to test the relationship between Dengue, Mosquitoes, and Meteorological Conditions. Rather than use [sklearn] module models, we instead used GLM models with the Poisson transformation from the [statsmodels. api] module, and compared results of Regression based on Mosquitoes and on Meteorological conditions of the three time periods (Immediate, 2 weeks prior, 4 weeks prior) by the metrics of Pseudo-R^2 (a GLM Poisson adjusted R^2 value), the mean-absolute-error, mean-absolute-percentage-error, and Regression p-values.

Step 1:Dengue Regressed on Extrapolated Mosquitoes

To analyze the association between Dengue and Meteorological Conditions, we examined the results of Poisson GLM regression of Mosquitoes to the monthly count of Dengue by each case instance. With just the extrapolated Mosquito data, we found the following results:

- Mean Absolute Error: 7.0607
- Mean Absolute Percentage Error: 1.037
- Pseudo R^2: 0.05171
- P-value: 0.000

Overall, extrapolated mosquitoes proved to be poor predictors of monthly Dengue Counts, even if the p-value is statistically significant. This association is ultimately suspect thanks to the poor quality extrapolation.

```
import statsmodels.api as sm
from statsmodels.formula.api import glm
from statsmodels.graphics.api import abline_plot

x = DENGUE_METEOR['mos'].values
y = DENGUE_METEOR['MONTH_COUNT'].values
x = sm.add_constant(x, prepend = False)

glm_pois = sm.GLM(y,x, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(x)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print(res.summary())
```

```
Mean absolute error of prediction is: 7.060756557935433
Mean absolute percentage error is: 1.0370769312260892
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  572
Model:                            GLM   Df Residuals:                      570
Model Family:                 Poisson   Df Model:                            1
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2793.2
Date:                Tue, 29 Nov 2022   Deviance:                       3173.5
Time:                        09:12:11   Pearson chi2:                 3.23e+03
No. Iterations:                     4   Pseudo R-squ. (CS):            0.05171
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.5715      0.104     -5.492      0.000      -0.776      -0.368
const          3.1824      0.106     30.156      0.000       2.976       3.389
==============================================================================
```

Fig. 27. Python Code and Output for Dengue-Mosquito Regression

Step 2. Dengue Regressed on Linear-Regressed Mosquitoes

If we assume that mosquitoes have an association with Meteorological conditions, then if Dengue also has such an association, then Mosquito specimen counts regressed from these conditions should have a high correlation with Dengue. Since Dengue takes two weeks on average to develop symptoms, then we predict that the mosquitoes predicted from conditions two weeks prior to symptoms will prove to have less error and a larger ratio of explained variance compared to other time periods.

```
x = DENGUE_METEOR.iloc[:,-5:-1]
y = DENGUE_METEOR['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)
predicted_mos = sm.add_constant(predicted_mos, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR['MONTH_COUNT'], predicted_mos, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(predicted_mos)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
##Remeber the square count correction
```

Fig. 28. Code to predict Dengue based on Mosquito Specimens regressed on immediate conditions

```
Mean absolute error of prediction is: 7.002447425423353
Mean absolute percentage error is: 1.0263792694668017
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                  572
Model:                           GLM   Df Residuals:                      570
Model Family:                Poisson   Df Model:                            1
Link Function:                   Log   Scale:                          1.0000
Method:                         IRLS   Log-Likelihood:                -2762.9
Date:                Tue, 29 Nov 2022   Deviance:                       3112.9
Time:                        09:11:06   Pearson chi2:                 3.17e+03
No. Iterations:                     4   Pseudo R-squ. (CS):             0.1471
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -1.7140      0.180     -9.500      0.000      -2.068      -1.360
const          4.3382      0.182     23.802      0.000       3.981       4.695
==============================================================================

<matplotlib.legend.Legend at 0x7f89490c32e0>
```

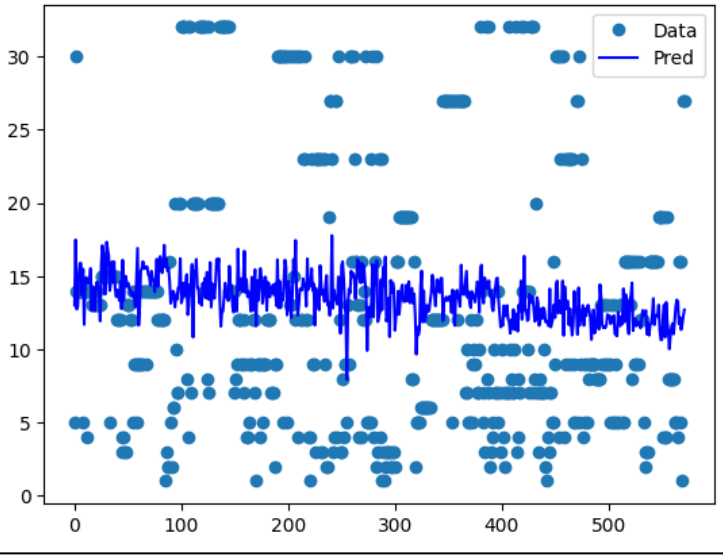


Fig. 29. Regression details and plotting of predicted values to actual values

```
x = DENGUE_METEOR_TWO.iloc[:,-5:-1]
y = DENGUE_METEOR_TWO['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)
predicted_mos = sm.add_constant(predicted_mos, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR_TWO['MONTH_COUNT'], predicted_mos, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(predicted_mos)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR_TWO['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR_TWO['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR_TWO['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
```

Fig. 30. Code to predict Dengue based on Mosquito Specimens regressed on conditions 2 weeks prior

```
Mean absolute error of prediction is: 6.786435540810117
Mean absolute percentage error is: 0.9905918773992103
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                  567
Model:                           GLM   Df Residuals:                      565
Model Family:                Poisson   Df Model:                            1
Link Function:                   Log   Scale:                          1.0000
Method:                         IRLS   Log-Likelihood:                -2683.9
Date:                Tue, 29 Nov 2022   Deviance:                       2982.0
Time:                        09:11:16   Pearson chi2:                 3.08e+03
No. Iterations:                     4   Pseudo R-squ. (CS):             0.3215
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -2.8139      0.192    -14.694      0.000      -3.189      -2.439
const          5.4349      0.193     28.219      0.000       5.057       5.812
==============================================================================

<matplotlib.legend.Legend at 0x7f8948e5e350>
```

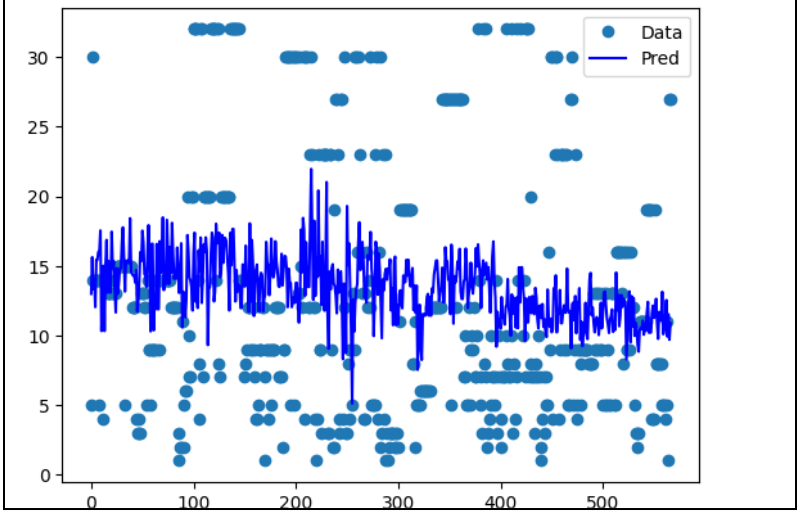


Fig. 31. Regression details and plotting of predicted values to actual values

```
x = DENGUE_METEOR_FOUR.iloc[:,-5:-1]
y = DENGUE_METEOR_FOUR['mos']

lm_mos = LinearRegression()
lm_mos.fit(x,y)
predicted_mos = lm_mos.predict(x)
predicted_mos = sm.add_constant(predicted_mos, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR_FOUR['MONTH_COUNT'], predicted_mos, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(predicted_mos)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR_FOUR['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR_FOUR['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR_FOUR['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
```

Fig. 32. Code to predict Dengue based on Mosquito Specimens regressed on conditions 4 weeks prior

```
Mean absolute percentage error is: 1.0242066781253867
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                  558
Model:                           GLM   Df Residuals:                      556
Model Family:                Poisson   Df Model:                            1
Link Function:                   Log   Scale:                          1.0000
Method:                         IRLS   Log-Likelihood:                -2695.5
Date:               Tue, 29 Nov 2022   Deviance:                       3046.8
Time:                       09:11:31   Pearson chi2:                 3.14e+03
No. Iterations:                    4   Pseudo R-squ. (CS):             0.2763
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -2.6765      0.201    -13.345      0.000      -3.070      -2.283
const          5.2941      0.202     26.257      0.000       4.899       5.689
==============================================================================

<matplotlib.legend.Legend at 0x7f8948e12740>
```

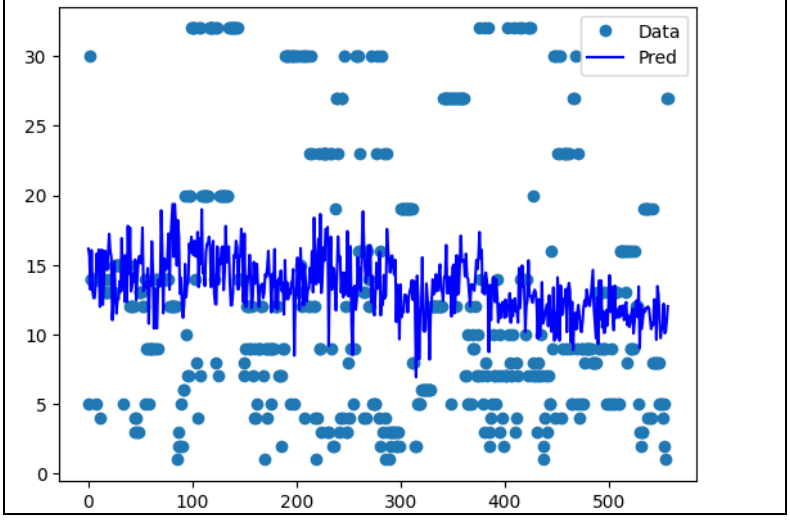


Fig.33. Regression details and plotting of predicted values to actual values

Running Poisson GLM regression upon Linear Regression predicted Mosquito values, we see that our Pseudo R^2 value is considerably larger compared to that of the initially extrapolated Mosquito values, as well as a slightly smaller mean absolute error and mean absolute percentage error. Of the regressed values, those predicted from conditions 2-weeks prior proved to have the better metrics compared to the other two:

- Mean Absolute Error: 6.78
- Mean Absolute Percentage Error: 0.991
- Pseudo R^2: 0.3215
- P-value: 0.000

Considering that Dengue on average takes 2-weeks from the initial bite to develop visible symptoms, and Linear Regression of Mosquitoes proved most accurate when based on immediate conditions, then the conditions two-weeks prior coincide with the immediate conditions of the mosquitoes at the time of the bite, which gives evidence to reject the null hypothesis.

Step 3. Dengue Regressed on Meteorological Conditions

To then test our hypothesis directly, we tested the association between Meteorological Conditions and Dengue, specifically in seeing if it has a better association in the time periods of 2-weeks prior and 4-weeks prior, as having a strong preceding association is evidence of the possible causal relationship between the two variables.

```
x = DENGUE_METEOR.iloc[:,-5:-1]
x = sm.add_constant(x, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR['MONTH_COUNT'], x, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(x)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
```

```
Mean absolute error of prediction is: 6.673026165988001
Mean absolute percentage error is: 0.9990509943001215
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                572
Model:                            GLM   Df Residuals:                    567
Model Family:                 Poisson   Df Model:                        4
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2643.6
Date:                Mon, 21 Nov 2022   Deviance:                       2874.3
Time:                        15:38:38   Pearson chi2:                 2.86e+03
No. Iterations:                     5   Pseudo R-squ. (CS):             0.4380
Covariance Type:            nonrobust
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
air_temp_max_c  -0.0795      0.015     -5.298      0.000      -0.109      -0.050
air_temp_min_c  -0.1785      0.015    -11.730      0.000      -0.208      -0.149
air_temp_aver_c  0.4134      0.033     12.609      0.000       0.349       0.478
humidity         0.0356      0.003     13.762      0.000       0.030       0.041
const           -4.4132      0.417    -10.571      0.000      -5.231      -3.595
================================================================================
```

Fig. 34. Code to predict Dengue based on Immediate Meteorological Conditions and Results

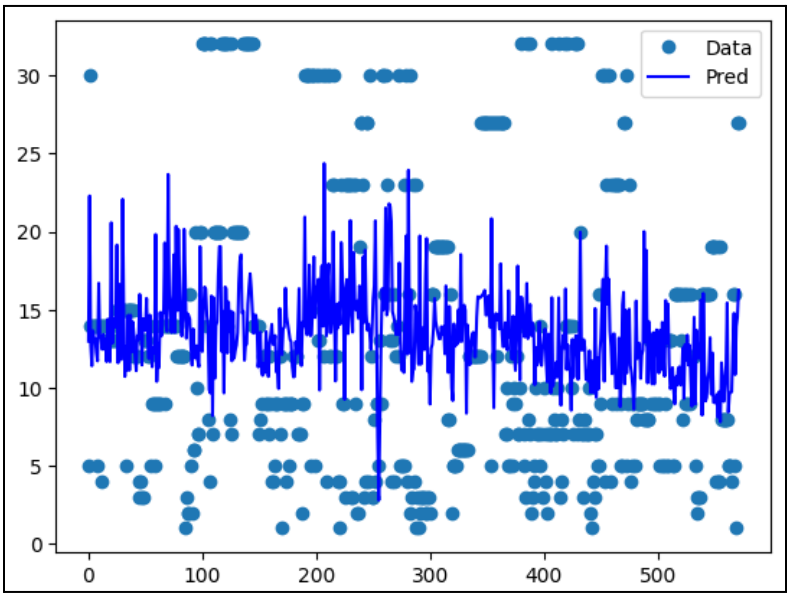


Fig. 35. Plotting of GLM predicted values against actual values

```
x = DENGUE_METEOR_TWO.iloc[:,-5:-1]
x = sm.add_constant(x, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR_TWO['MONTH_COUNT'], x, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(x)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR_TWO['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR_TWO['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR_TWO['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
```

```
Mean absolute error of prediction is: 6.577754171165761
Mean absolute percentage error is: 0.9681247358108724
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                  567
Model:                            GLM   Df Residuals:                      562
Model Family:                 Poisson   Df Model:                            4
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2592.2
Date:                Sat, 19 Nov 2022   Deviance:                       2798.6
Time:                        16:41:36   Pearson chi2:                 2.81e+03
No. Iterations:                     4   Pseudo R-squ. (CS):             0.5090
Covariance Type:            nonrobust
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
air_temp_max_c   0.0267      0.015      1.737      0.082      -0.003       0.057
air_temp_min_c  -0.1666      0.015    -10.899      0.000      -0.197      -0.137
air_temp_aver_c  0.2638      0.034      7.806      0.000       0.198       0.330
humidity         0.0365      0.003     13.494      0.000       0.031       0.042
const           -4.0996      0.445     -9.211      0.000      -4.972      -3.227
================================================================================
```

Fig. 36. Code to predict Dengue based on Conditions 2 weeks prior

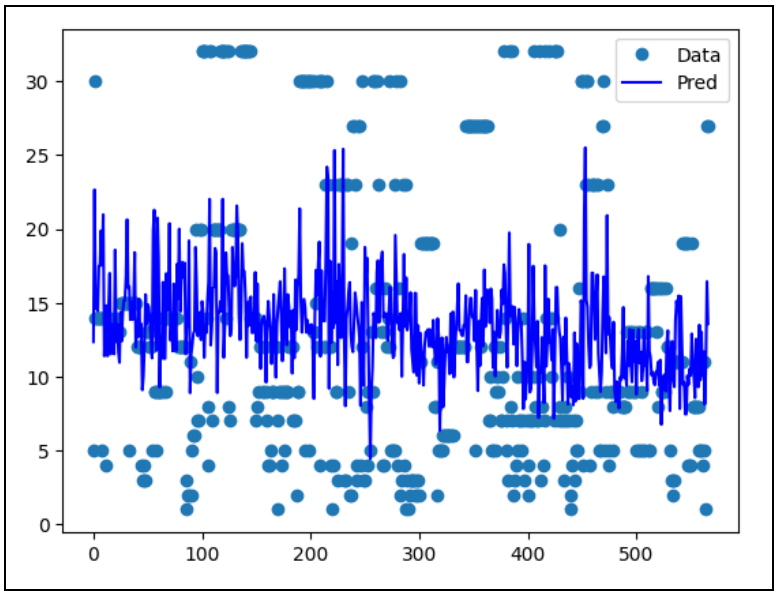


Fig. 37. Plotting of GLM predicted values against actual values

```
x = DENGUE_METEOR_FOUR.iloc[:,-5:-1]
x = sm.add_constant(x, prepend = False)

glm_pois = sm.GLM(DENGUE_METEOR_FOUR['MONTH_COUNT'], x, family = sm.families.Poisson())
res = glm_pois.fit()
y_pred = res.predict(x)
print('Mean absolute error of prediction is:', skm.mean_absolute_error(DENGUE_METEOR_FOUR['MONTH_COUNT'], y_pred))
print('Mean absolute percentage error is:', skm.mean_absolute_percentage_error(DENGUE_METEOR_FOUR['MONTH_COUNT'], y_pred))
print(res.summary())

fig, ax = plt.subplots()
ax.plot(DENGUE_METEOR_FOUR['MONTH_COUNT'], "o", label="Data")
ax.plot(y_pred, "b-", label="Pred")
ax.legend(loc="best")
```

```
Mean absolute error of prediction is: 6.8631194529262745
Mean absolute percentage error is: 1.0173703575020063
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:            MONTH_COUNT   No. Observations:                  558
Model:                           GLM   Df Residuals:                      553
Model Family:                Poisson   Df Model:                            4
Link Function:                   Log   Scale:                          1.0000
Method:                         IRLS   Log-Likelihood:                -2662.3
Date:                Sat, 19 Nov 2022   Deviance:                       2980.3
Time:                        16:41:53   Pearson chi2:                 3.05e+03
No. Iterations:                     4   Pseudo R-squ. (CS):             0.3576
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
air_temp_max_c  0.0077      0.015      0.502      0.616      -0.022       0.038
air_temp_min_c -0.1061      0.016     -6.666      0.000      -0.137      -0.075
air_temp_aver_c  0.2295      0.034      6.792      0.000       0.163       0.296
humidity        0.0336      0.003     12.643      0.000       0.028       0.039
const          -3.7566      0.439     -8.558      0.000      -4.617      -2.896
==============================================================================
```

Fig. 38. Code to predict Dengue based on Conditions 4 weeks prior

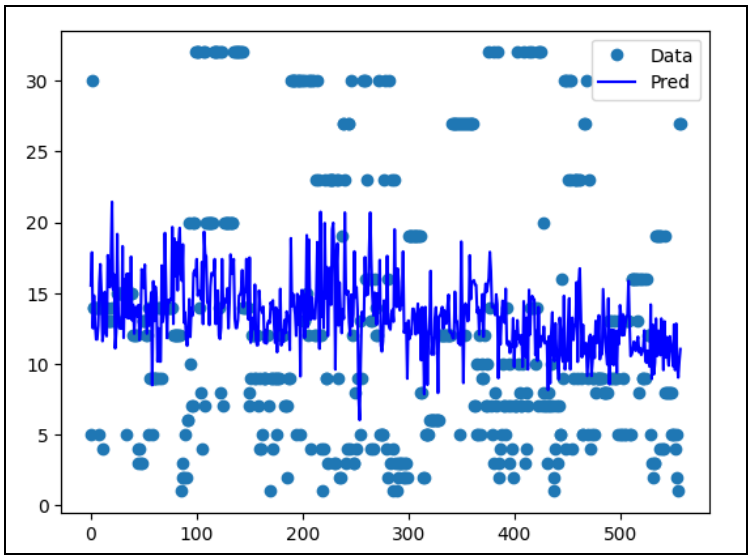


Fig. 39. Plotting of GLM predicted values against actual values

Running our GLM Regression, we see that the behavior is similar to our regression based on linear regression-derived mosquito data, and is consistent with the symptom development period of Dengue, giving further support that Meteorological conditions, specifically those 2-weeks prior, are significantly associated with Dengue.

5.8 Time Series Analysis

For Time Series Analysis we used Holt-Winters Time Series and Auto Regressive model. A method of examining a set of data points gathered over some time is called a "time series analysis." Instead of merely capturing the data points sporadically or arbitrarily, we capture the data points at regular intervals over a predetermined length of time. The method used to model the data throughout a specific period is the shared characteristic.

```
In [14]: ## Visualize the time series data
         dengue_count_df.DENGUE_COUNT.plot(figsize=(12,6))
Out[14]: <AxesSubplot:>
```

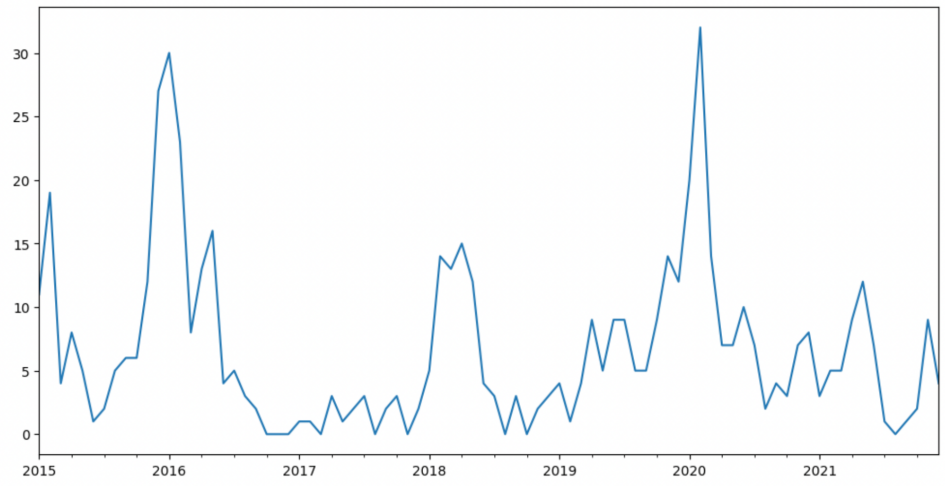


Fig.40. Actual Dengue data.

We have conducted a statistical test called ADFuller test to know if our data is stationary or not. Since the p-value is less than 0.05 we are rejecting the null hypothesis and thus saying that the data is stationary.

```
test_result=adfuller(dengue_count_df)
```

```
#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(cases):
    result=adfuller(cases)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root an
    else:
        print("Weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary "
```

```
adfuller_test(dengue_count_df)
```

```
ADF Test Statistic : -3.8350132652263382
p-value : 0.002568726590228407
#Lags Used : 0
Number of Observations Used : 83
Strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationar
y
```

Fig.41. Code for ADFuller test.

5.8.1 Holt-Winter Time Series Analysis

In Holt-Winter Time Series Analysis, first we decomposed the entire dengue count. And for any time series model, the data sheet has to be in a time stamp.

Hence, we have converted the entire month column into a timestamp model. After decomposing the data we have actual data, trend, seasonality, and residual data.

```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
dengue_count_df.interpolate(inplace = True)
dengue_count_df.index=dengue_count_df.index.to_timestamp()

#dengue_count_df
results=seasonal_decompose(dengue_count_df.DENGUE_COUNT,model='additive').plot();
```

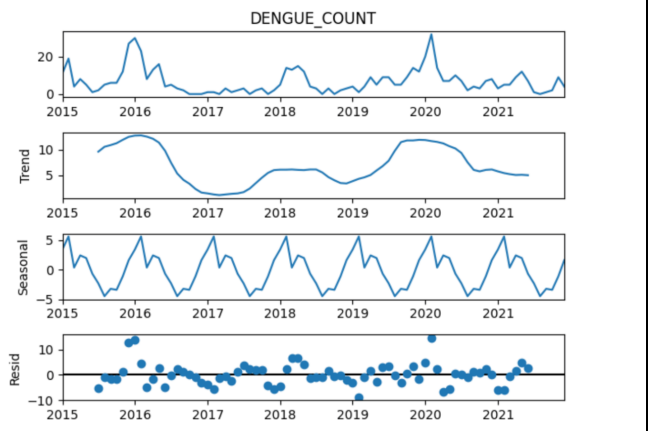


Fig.42. Code for decomposing dengue count and figure showing time stamp model.

We did the test and train split for the data to be in sequential order. Since we had 84 rows in total, we took 64 rows for train data and 20 rows for the test data, which is 75% and 25% respectively.

```
len(dengue_count_df)
```

```
84
```

```
DCtrain=dengue_count_df.DENGUE_COUNT[:64]
DCtest=dengue_count_df.DENGUE_COUNT[64:]
#DC for dengue count
```

```
Ttrain = dengue_count_df.AVER_MAX[:64]
Ttest = dengue_count_df.AVER_MAX[64:]
#T for temperatue
```

```
Htrain = dengue_count_df.AVER_HUMIDITY[:64]
Htest = dengue_count_df.AVER_HUMIDITY[64:]
#H for humidity
```

```
len(DCtest),len(Ttest),len(Htest)
```

```
(20, 20, 20)
```

Fig.43. Code for splitting the data into test and train.

We made a prediction for the number of future dengue cases. The orange one in the image shows the number of future cases.

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

DCholt_model = ExponentialSmoothing(DCtrain, trend='add', seasonal='add', seasonal_periods=20)
DChwmodel = DCholt_model.fit()
#DC model for Dengue count
```

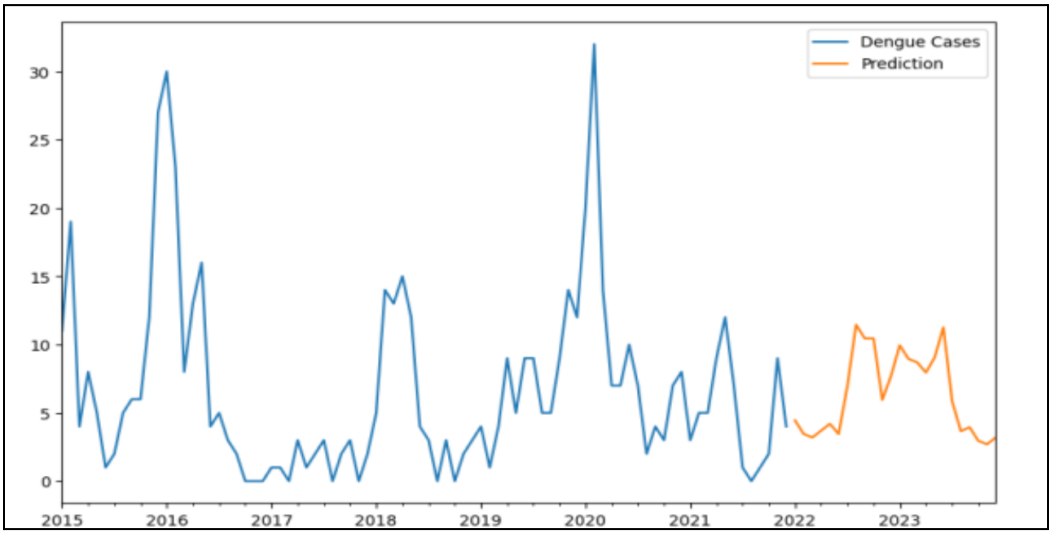


Fig.44. Code for predicting the number of future dengue cases

To predict the no. of dengue cases we divided the entire dataset into a test and trained for temperature, humidity, and dengue count. We have also noticed from the plots that predicting the dengue count was much better than temperature or precipitation.

```
DCtrain.plot(legend=True, label='Train Dengue count', figsize=(10,6))
DCtest.plot(legend=True, label='Test Dengue count')
DCtest_pred.plot(legend=True, label='predicted_test Dengue count')
```

```
<AxesSubplot:>
```

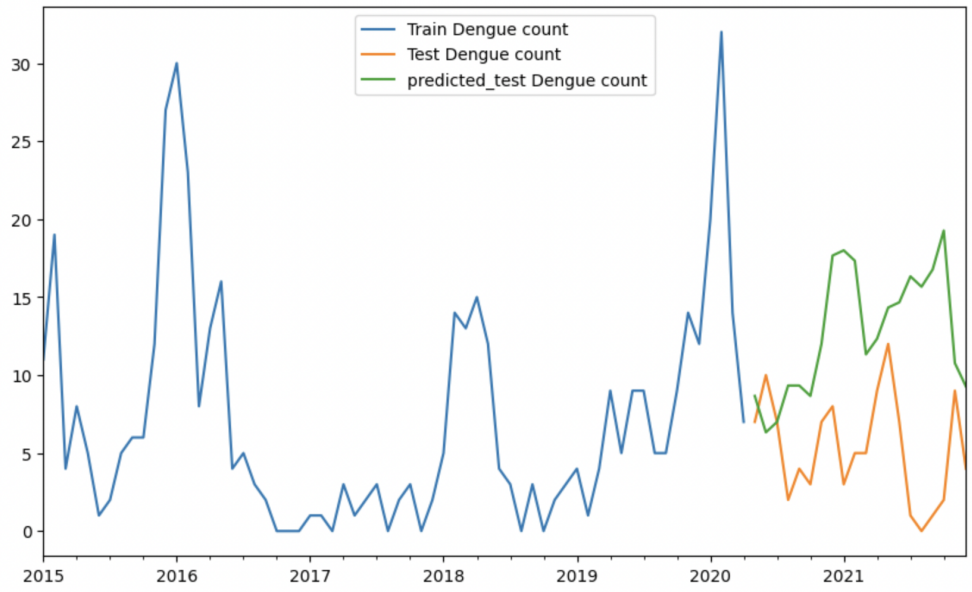


Fig.45. Code and plot for the split dengue count data.

```
Ttrain.plot(legend=True, label='Train Temperature', figsize=(10,6))
Ttest.plot(legend=True, label='Test Temperature')
Ttest_pred.plot(legend=True, label='predicted_test Temperature')
```

```
<AxesSubplot:>
```

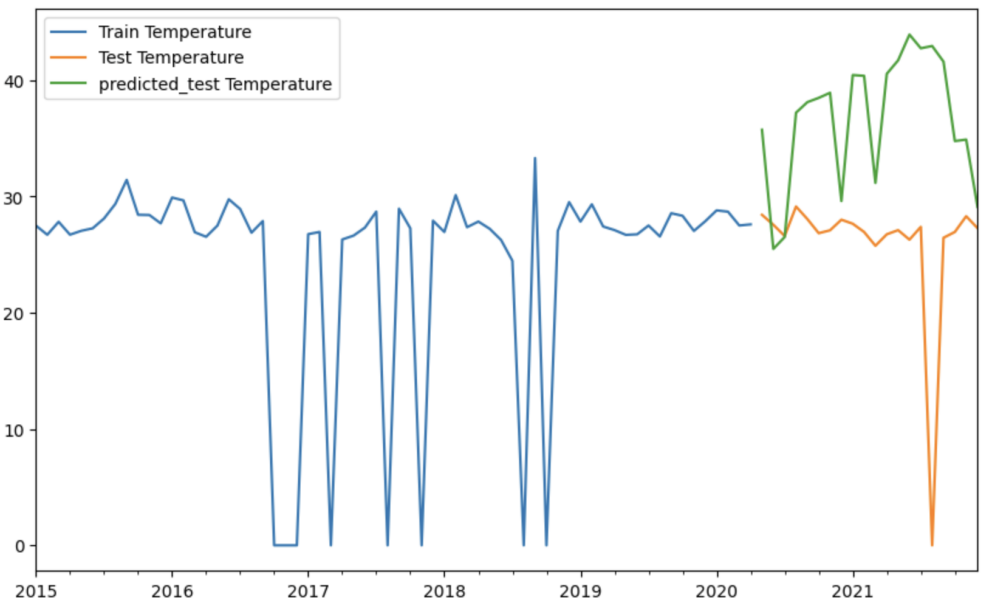


Fig.46. Code and plot for the split temperature data

```
Htrain.plot(legend=True, label='Train Humidity', figsize=(10,6))
Htest.plot(legend=True, label='Test Humidity')
Htest_pred.plot(legend=True, label='predicted_test Humidity')
```

```
<AxesSubplot:>
```

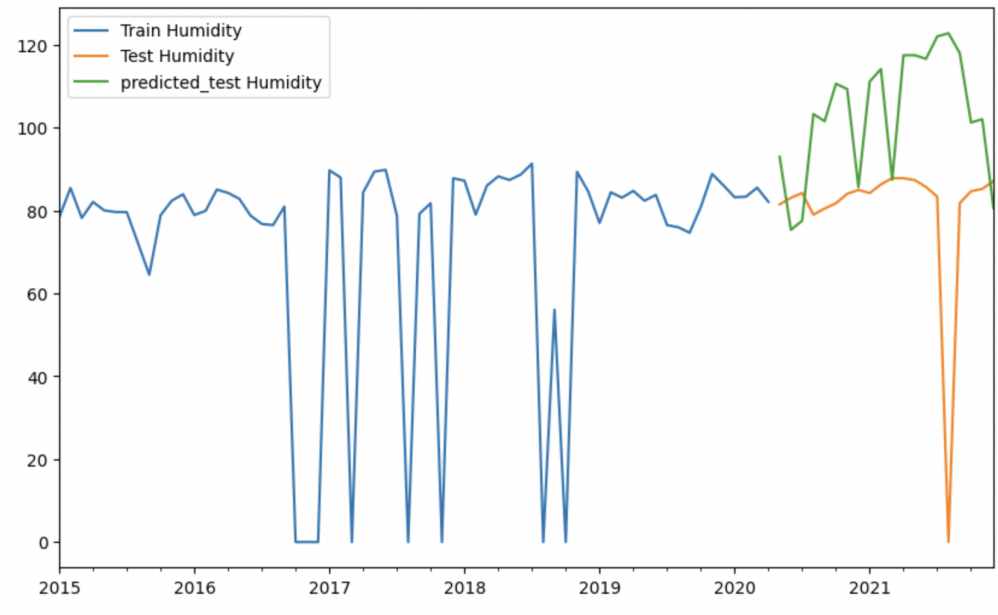


Fig.47. Code and plot for the split humidity data.

We then calculated the Root Mean Square Error for the predicted dengue cases. From the results we can see that the RMSE value is very high for dengue cases, hence we decided to go for the Auto-Regressive Model.

```
import math
MSE = np.square(np.subtract(DCtest,DCtest_pred)).mean()

RMSE = math.sqrt(MSE)
print("Root Mean Square Error:\n")
print(RMSE) # for DC
```

```
Root Mean Square Error:

9.484634444510322
```

Fig.48. Code for Root Mean Square Error Calculation

5.8.2 Auto-Regressive Model

5.8.2.1 Partial-Autocorrelation Plot

We plotted the autocorrelation plot to see if there is any variation in the dengue count variable or to check for any lags in the data that we have. It helps us in giving the lags for the AR Model which we need to calculate the entire formula. So, those lags can be found through the Autocorrelation plot.

```
#AutoCorelation plot
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(dengue_count_df)
plt.show()
```

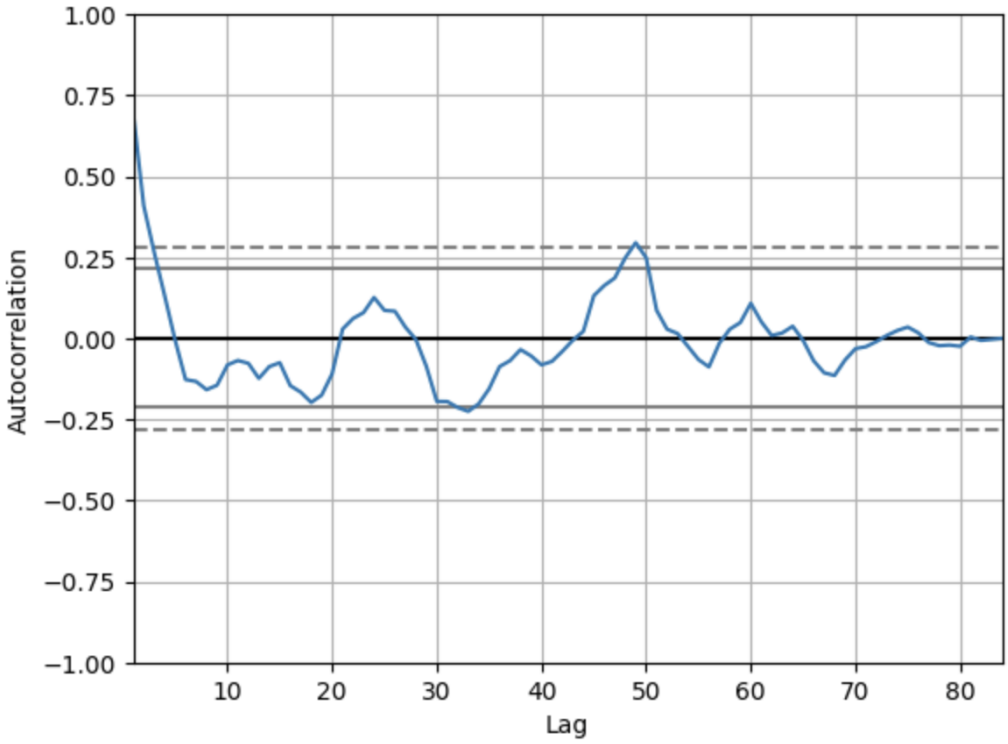


Fig.49. Code and plot for Autocorrelation.

Then we plotted the partial autocorrelation plot to know the number of lags exactly. The autocorrelation plot is more like a wave format and the partial autocorrelation plot helps us in knowing exact lag values. We have taken 3 as a lag period for our AR Model since there is a drastic drop in values from 2 to 3.

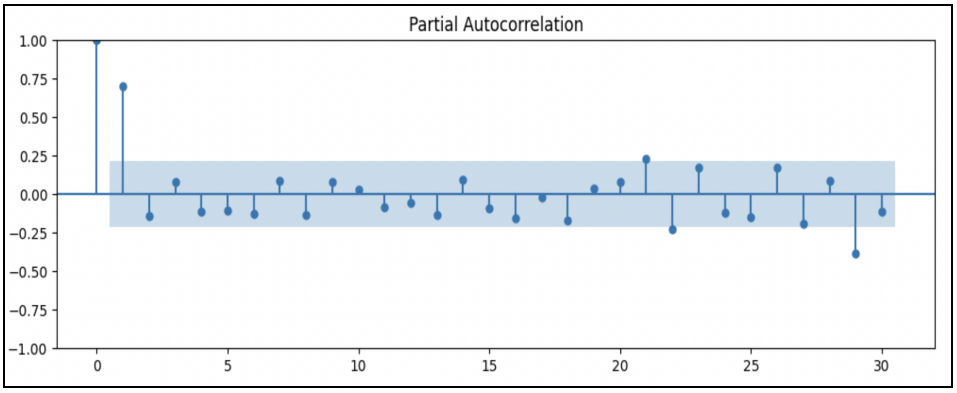


Fig.50. Plot for Partial Autocorrelation

We plotted train data vs predicted data,

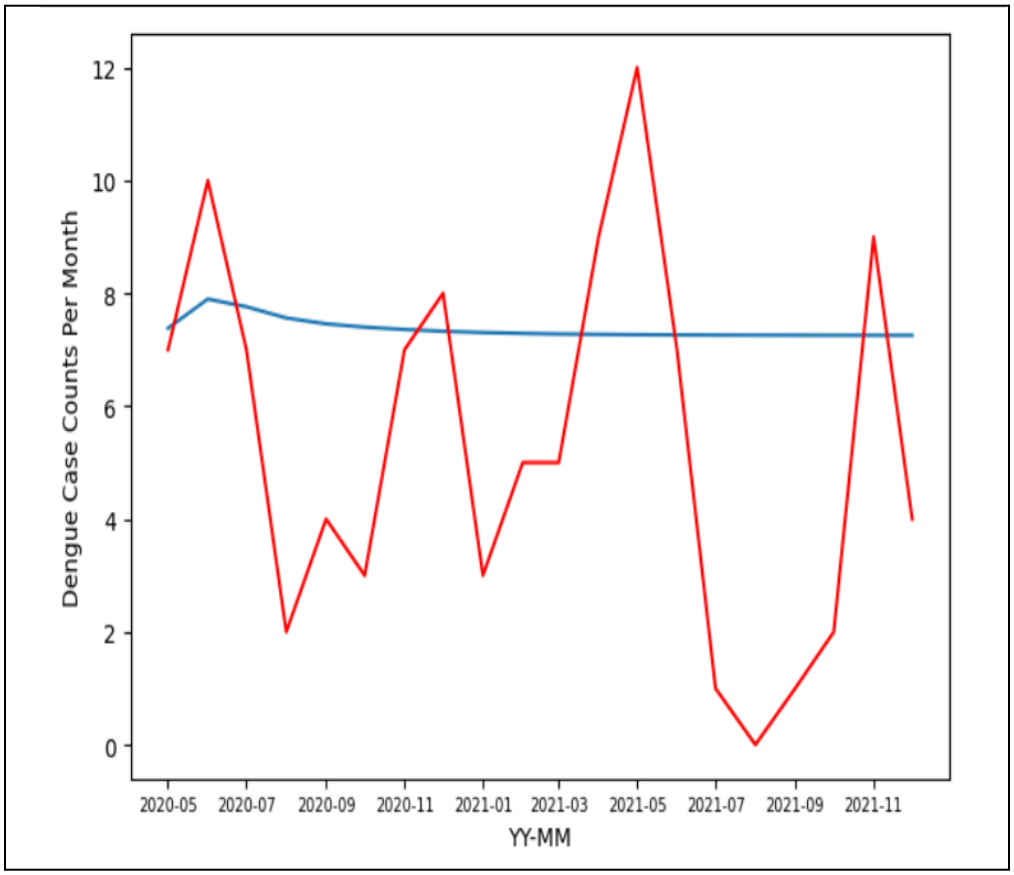


Fig.51. Train data vs Predicted data.

We have calculated the Root Mean Square Error for this model. Although it wasn't a good value, it did very much better compared to the Holt-Winters Time Series Model. Hence, we proceeded with correlation based on the AR Model.

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

```
import math
MSE = np.square(np.subtract(test,pred)).mean()
RMSE = math.sqrt(MSE)
print("Root Mean Square Error:\n")
print(RMSE)
```

```
Root Mean Square Error:

3.8388388260067603
```

```
test.mean(),np.sqrt(test.var())
```

```
(5.3, 3.357630743623592)
```

Fig.52. Code for calculating the Root Mean Square Error value for AR Model.

For calculating the correlation we predicted only the last 20 cells and took their variables to conduct the correlation. Since predicted values were a series data set, we had to squeeze it back into the data frame form.

```
pred_future=model.predict(start=len(train),end=(len(dengue_count_df)-1))
print("The future predictions for the next month")
print(pred_future)
print("The no. of predictions made: \t",len(pred_future))
```

```
The future predictions for the next month
2020-05-01    7.381962
2020-06-01    7.898427
2020-07-01    7.761967
2020-08-01    7.563454
2020-09-01    7.458644
2020-10-01    7.400729
2020-11-01    7.358917
2020-12-01    7.327743
2021-01-01    7.305656
2021-02-01    7.290277
2021-03-01    7.279503
2021-04-01    7.271916
2021-05-01    7.266575
2021-06-01    7.262820
2021-07-01    7.260179
2021-08-01    7.258322
2021-09-01    7.257016
2021-10-01    7.256097
2021-11-01    7.255451
2021-12-01    7.254997
Freq: MS, dtype: float64
The no. of predictions made:     20
```

Fig. 53. Predictions of Dengue cases

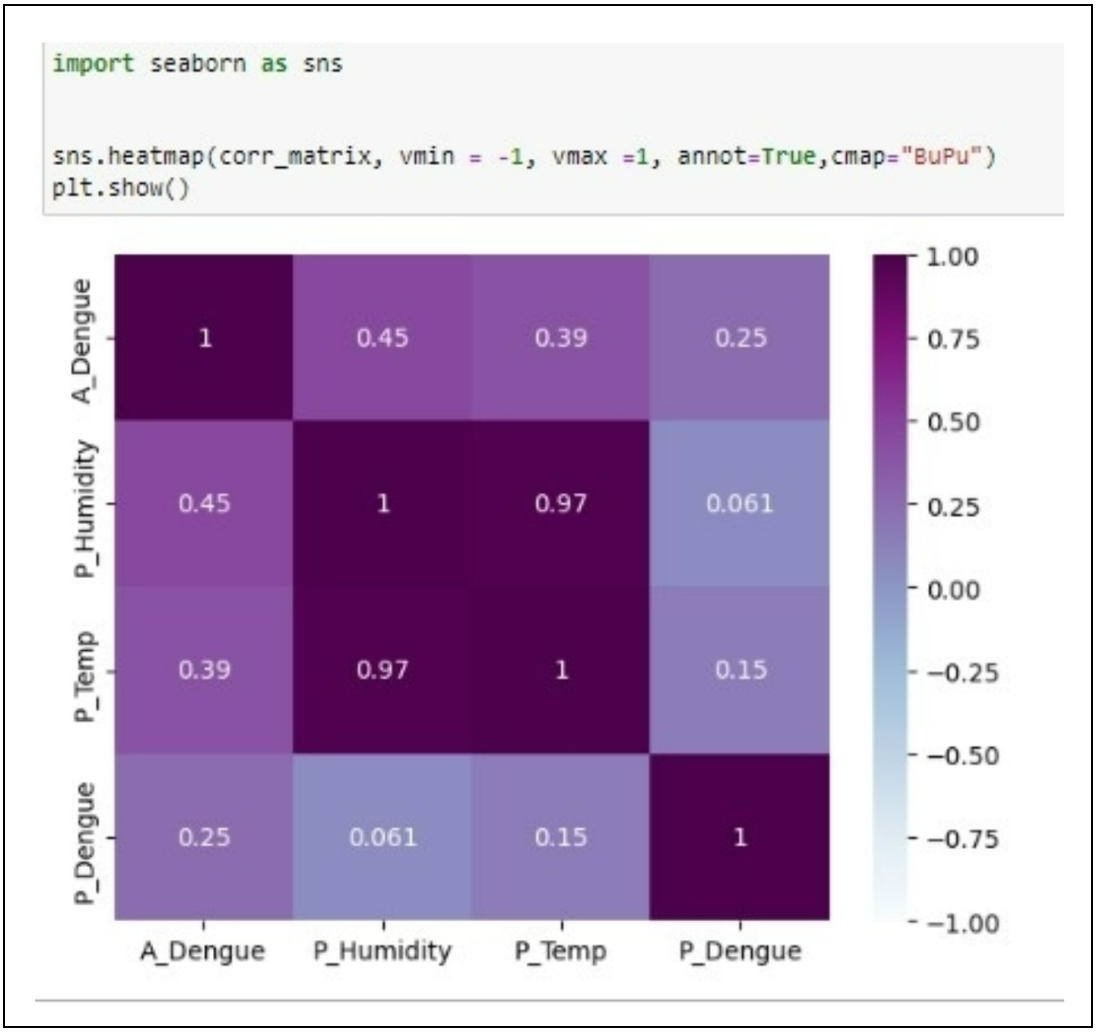


Fig.54. Correlation between Predicted Dengue count, Temperature, Humidity, and Actual dengue count.

The heat map depicts the correlation between the actual values and predicted values, with a value of 1 indicating a very strong positive correlation, and values below 0 indicating a negative study. In our dataset, we observed that the predicted humidity and predicted temperature has the strongest positive correlation at 0.97, whereas, predicted humidity and predicted dengue has the weakest correlation at 0.061.

5.9 Other Regression Models

5.9.1 Random Forest Regression

RF Regression is a supervised algorithm that uses an ensemble learning method to predict a continuous value. We used RF Regression as we had to predict the monthly dengue cases and as it is a sequential process.

```
#Train data
X_train = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][:64]
y_train = np.array(dengue_count_df_full[['DENGUE_COUNT']][:64])

#Test data
X_test = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][64:]
y_test = np.array(dengue_count_df_full[['DENGUE_COUNT']][64:])

rf_d = RandomForestRegressor()
model=rf_d.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('The Random Forest Regressor model R^2 value is:', model.score(X_train, y_train))
print('-----------------------------------------------------------------')
train_acc = model.score(X_train, y_train)
print("The Training Accuracy is {}".format(train_acc*100))
print('-----------------------------------------------------------------')
print('Mean absolute error of prediction is:',mean_absolute_error(y_test, y_pred))
print('-----------------------------------------------------------------')
print('Mean absolute percentage error is:', mean_absolute_percentage_error(y_test, y_pred))
print('-----------------------------------------------------------------')
print('Explained Variance by Regression is:', explained_variance_score(y_test, y_pred))
```

```
The Random Forest Regressor model R^2 value is: 0.8610457013007209
-----------------------------------------------------------------
The Training Accuracy is 86.10457013007209
-----------------------------------------------------------------
Mean absolute error of prediction is: 5.297272727272728
-----------------------------------------------------------------
Mean absolute percentage error is: 2.1903520923520925
-----------------------------------------------------------------
Explained Variance by Regression is: -0.604665642458101
```

Fig. 55. Random Forest Regressor

5.9.2 Gradient Boost Regression

Gradient Boosting Classification is used to predict classes like whether a patient has a certain disease or not, Gradient Boosting Regression is used to predict continuous variables.

Gradient Boost Regressor

```
#Train data
X_train = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][:64]
y_train = np.array(dengue_count_df_full[['DENGUE_COUNT']][:64])

#Test data
X_test = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][64:]
y_test = np.array(dengue_count_df_full[['DENGUE_COUNT']][64:])

gb_d = GradientBoostingRegressor()
model=gb_d.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('The GB Regressor model R^2 value is:', model.score(X_train, y_train))
print('------------------------------------------------------------------')
train_acc = model.score(X_train, y_train)
print("The Training Accuracy is {}".format(train_acc*100))
print('------------------------------------------------------------------')
print('Mean absolute error of prediction is:',mean_absolute_error(y_test, y_pred))
print('------------------------------------------------------------------')
print('Mean absolute percentage error is:', mean_absolute_percentage_error(y_test, y_pred))
print('------------------------------------------------------------------')
print('Explained Variance by Regression is:', explained_variance_score(y_test, y_pred))
```

```
The GB Regressor model R^2 value is: 0.9890262120486008
------------------------------------------------------------------
The Training Accuracy is 98.90262120486007
------------------------------------------------------------------
Mean absolute error of prediction is: 5.487338157125395
------------------------------------------------------------------
Mean absolute percentage error is: 2.364548547492539
------------------------------------------------------------------
Explained Variance by Regression is: -1.2373003319378002
```

Fig. 56. Gradient Boost Regressor

5.9.3 XGB Regressor

The model performance of the XGB Regressor is better than both the RF Regressor and GB Regression hence we decided to stick with this model.

XG Boost Regressor

```
#!pip install --user xgboost
import xgboost as xg
```

```
#Train data
X_train = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][:64]
y_train = np.array(dengue_count_df_full[['DENGUE_COUNT']][:64])

#Test data
X_test = dengue_count_df_full[['AVER_MAX',  'AVER_MIN',  'AVER_AVER',  'AVER_HUMIDITY']][64:]
y_test = np.array(dengue_count_df_full[['DENGUE_COUNT']][64:])

xg_d = xg.XGBRegressor()
model=xg_d.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('The XG Regressor model R^2 value is:', model.score(X_train, y_train))
print('-----------------------------------------------------------------')
train_acc = model.score(X_train, y_train)
print("The Training Accuracy is {}".format(train_acc*100))
print('-----------------------------------------------------------------')
print('Mean absolute error of prediction is:',mean_absolute_error(y_test, y_pred))
print('-----------------------------------------------------------------')
print('Mean absolute percentage error is:', mean_absolute_percentage_error(y_test, y_pred))
print('-----------------------------------------------------------------')
print('Explained Variance by Regression is:', explained_variance_score(y_test, y_pred))
```

```
The XG Regressor model R^2 value is: 0.999999983882541
-----------------------------------------------------------------
The Training Accuracy is 99.9999983882541
-----------------------------------------------------------------
Mean absolute error of prediction is: 4.545642310922796
-----------------------------------------------------------------
Mean absolute percentage error is: 1.900262668466499
-----------------------------------------------------------------
Explained Variance by Regression is: -0.8700055657695183
```

Fig. 57. XG Boost Regressor

6. Summary of Findings

- Simple Linear Regression: The regression model was done for the mosquito count which yielded a different R^2 value for immediate, 2 weeks, and 4 weeks which was seen to be decreasing as the time period increased starting from 0.33 to 0.28. But the mean absolute error and mean absolute percentage errors started at 0.0747 and 0.0742 respectively. They are observed to increase to 0.0774 and 0.0769 respectively.

- Poisson GLM: In Poisson's GLM, the model was carried out for the same three different time periods again. This time, the pseudo R^2 value started at 0.43 for immediate time and increased at 2 weeks, and again decreased at 4 weeks from 0.509 to 0.357. The mean absolute error and mean absolute percentage error started at 6.673 and 0.999 respectively. In the second week, it was seen to decrease to 6.577 and 0.968 and in the 4th week, it again increased to 6.863 and 1.01.

From these readings, it can be said that the Poisson GLM was better than linear regression.

- Random Forest Regression: This regression was done for the dengue count at the immediate time period. The R^2/ Pseudo R^2 value was found to be 0.86 which shows an increase when compared with the R^2 results from the Poissons GLM. The mean absolute error and mean absolute percentage error were observed to be 5.29 and 2.19 respectively.

- Gradient Boost Regressor: In GB Regression, when compared with the RF regression, there is an increase in the R^2, mean absolute error and mean absolute percentage error values at the immediate time period with 0.98, 5.48, and 2.39 respectively.

- XGBoost Regressor: This XG Boost regression was done for the dengue count at the immediate time period which yielded an R^2 value of 0.99 is the highest among all the other models. The value of mean absolute error was found to be the least with 4.54 and the mean absolute percentage error is observed to be decreased to 1.90 from GB regression.

MODEL	R^2/Pseudo R^2	MAE	MAPE
Linear Regression	0.33	0.07	0.07
Poisson GLM	0.32	6.7	0.99
RF Regression	0.86	5.29	2.19
GB Regression	0.98	5.48	2.36
XGB Regression	0.99	4.54	1.90

Fig.58. Model Comparison

From the above table, we see that the XGB regressor happens to be the best performing model amongst all the linear regression models used for this study based on the available metrics. The R^2 value for this happens to be the best among all, with a relatively lower Mean Absolute Error(MAE) and Mean Absolute Percentage Error(MAPE).

From all of these results, it is evident that the XGBoost Regressor was able to predict the dengue count much more accurately than all the other regression models.
Additionally, Holt Winter time series analysis was performed and the Root Mean Square Error for the anticipated dengue cases was then determined to be 9.484. We used the auto-regressive model since the findings show that the RMSE value for dengue cases is extremely high. For this model, the Root Mean Square Error was found to be 3.838. Even though it did not do well, it performed substantially better than the Holt-Winters Time Series Model. As a result, using the AR Model, we moved forward with the correlation.

7. Limitations

- We should have categorized the entire data set as a weekly data set instead of monthly categorization, which could have resulted in more rows.
- Bigger data set would have given better results.
- The data obtained from the metrics was not that distinguishable to come to a proper conclusion.
- We could not find any correlation between the temperature, humidity, and dengue cases.

A. Appendix

A.1 Python Code to Import NC-File Meteorological Data

Necessary Modules Imported

```
#pip install xarray
#This will be necessary if you wish to convert the NC files

#pip install netcdf4
#also necessary to have xarray work to convert NC Files

import xarray as xr
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

Fig.59. Necessary modules for SQL Data importing: xarray, netcdf4, numpy, pandas, warnings

Converting NC Files to CSV

```
humidity = xr.open_dataset("NCFiles/H_mswx_2015_2022.nc")
humidity_df = humidity.to_dataframe()
temp_max = xr.open_dataset("NCFiles/Tmax_mswx_2015_2022.nc")
temp_max_df = temp_max.to_dataframe()
temp_min = xr.open_dataset("NCFiles/Tmin_mswx_2015_2022.nc")
temp_min_df = temp_min.to_dataframe()
temp_aver = xr.open_dataset("NCFiles/Tprom_mswx_2015_2022.nc")
temp_aver_df = temp_aver.to_dataframe()
#Creates the dataframes to then save as csv files, said csv files (
#creates from this python code.
#Have already completed this step. Can be repeated as you like on o

humidity_df.to_csv("NCFiles/humidity.csv", sep = ",")
temp_max_df.to_csv("NCFiles/temp_max.csv", sep = ",")
temp_min_df.to_csv("NCFiles/temp_min.csv", sep = ",")
temp_aver_df.to_csv("NCFiles/temp_aver.csv", sep = ",")
```

Fig.60. Converting NC-Files into DataFrames, and these DataFrames into CSV-Files

Creating Meteorological tables to import Data into

```
import MySQLdb
conn = MySQLdb.connect(host="localhost", user[redacted]
                       passwd[redacted], db=[redacted])
cursor = conn.cursor()
#for group database
```

Fig.61. Opening SQL and establishing a connection to import large datasets

```
make_table_1 = """CREATE TABLE meteor_maxtemp (
time DATE NOT NULL,
lon DECIMAL(18,16) NOT NULL,
lat DECIMAL(17,16) NOT NULL,
air_temp_max_c DECIMAL(6,4) NOT NULL
)"""
cursor.execute(make_table_1)

make_table_2 = """CREATE TABLE meteor_mintemp (
time DATE NOT NULL,
lon DECIMAL(18,16) NOT NULL,
lat DECIMAL(17,16) NOT NULL,
air_temp_min_c DECIMAL(6,4) NOT NULL
)"""
cursor.execute(make_table_2)

make_table_3 = """CREATE TABLE meteor_avertemp (
time DATE NOT NULL,
lon DECIMAL(18,16) NOT NULL,
lat DECIMAL(17,16) NOT NULL,
air_temp_aver_c DECIMAL(6,4) NOT NULL
)"""
cursor.execute(make_table_3)

make_table_4 = """CREATE TABLE meteor_humidity (
time DATE NOT NULL,
lon DECIMAL(18,16) NOT NULL,
lat DECIMAL(17,16) NOT NULL,
humidity DECIMAL(7,4) NOT NULL
)"""
cursor.execute(make_table_4)

conn.commit()
```

Fig.62. Creating Tables in SQL to upload Meteorological Data into

```
load_table_1 = """LOAD DATA LOCAL INFILE 'NCFiles/temp_max.csv'
INTO TABLE meteor_maxtemp
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (time, lon, lat, air_temp_max_c)"""
cursor.execute(load_table_1)
#result: Success!

load_table_2 = """LOAD DATA LOCAL INFILE 'NCFiles/temp_min.csv'
INTO TABLE meteor_mintemp
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (time, lon, lat, air_temp_min_c)"""
cursor.execute(load_table_2)
#result: Success!

load_table_3 = """LOAD DATA LOCAL INFILE 'NCFiles/temp_aver.csv'
INTO TABLE meteor_avertemp
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (time, lon, lat, air_temp_aver_c)"""
cursor.execute(load_table_3)
#result: Success!

load_table_4 = """LOAD DATA LOCAL INFILE 'NCFiles/humidity.csv'
INTO TABLE meteor_humidity
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (time, lon, lat, humidity)"""
cursor.execute(load_table_4)
#result: Success!
```

```
1612200
```

Fig.63. Loading Meteorological CSV files into SQL data tables

Creating Indexes for Natural Joins

```
index_1 = """CREATE INDEX hum_index ON meteor_humidity (time, lon, lat);"""
index_2 = """CREATE INDEX aver_index ON meteor_avertemp (time, lon, lat);"""
index_3 = """CREATE INDEX max_index ON meteor_maxtemp (time, lon, lat);"""
index_4 = """CREATE INDEX min_index ON meteor_mintemp (time, lon, lat);"""
cursor.execute(index_1 + index_2)
cursor.execute(index_3 + index_4)
```

```
0
```

Fig. 64. Indexing Datatables to speed up natural joins for creating NC_Data table

Creating NC_Data Table

```
make_table_5 = """CREATE TABLE NC_Data (
time DATE NOT NULL,
lon DECIMAL(18,16) NOT NULL,
lat DECIMAL(17,16) NOT NULL,
air_temp_max_c DECIMAL(6,4) NOT NULL,
air_temp_min_c DECIMAL(6,4) NOT NULL,
air_temp_aver_c DECIMAL(6,4) NOT NULL,
humidity DECIMAL(7,4) NOT NULL
)"""
cursor.execute(make_table_5)
```

Fig.65. Creating NC_Data Table in SQL to upload joined meteorological data tables into.

Importing Joined Meteorological Data Tables into NC_Data Table.

```
load_table_5 = """INSERT INTO NC_Data SELECT meteor_mintemp.time, meteor_mintemp.lon, meteor_mintemp.lat,
air_temp_max_c, air_temp_min_c, air_temp_aver_c, humidity  FROM meteor_mintemp NATURAL JOIN meteor_maxtemp
NATURAL JOIN meteor_avertemp NATURAL JOIN meteor_humidity"""
cursor.execute(load_table_5)
```

```
1612200
```

Fig.66. Loading Joined Meteorological data tables into NC_Data table

Indexing NC_Data Table

```
index_5 = """CREATE INDEX hum_index ON NC_Data (time, lon, lat);"""
cursor.execute(index_5)
```

```
0
```

Fig.67. Indexing NC_Data Table to speed up joins with Dengue Data Tables and Mosquito Data Table.

A.2 Python Code for Time Series

```
print(model.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:           DENGUE_COUNT   No. Observations:                   64
Model:                     AutoReg(3)   Log Likelihood                -183.562
Method:               Conditional MLE   S.D. of innovations              4.905
Date:                Tue, 13 Dec 2022   AIC                            377.125
Time:                        20:22:57   BIC                            387.679
Sample:                    04-01-2015   HQIC                           381.261
                         - 04-01-2020
==============================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             2.0375      0.940      2.168      0.030       0.196       3.879
DENGUE_COUNT.L1   0.8582      0.122      7.034      0.000       0.619       1.097
DENGUE_COUNT.L2  -0.2104      0.155     -1.353      0.176      -0.515       0.094
DENGUE_COUNT.L3   0.0713      0.137      0.519      0.604      -0.198       0.341
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.4220           -0.0000j            1.4220           -0.0000
AR.2            0.7637           -3.0456j            3.1399           -0.2109
AR.3            0.7637           +3.0456j            3.1399            0.2109
-----------------------------------------------------------------------------
```

Fig.68. Summary of AR Model

```
#df1=predicted dengue cases
df1 = pred_future.reset_index()
# df1 =df1.drop(columns=['index'])
# df1
df1.rename(columns = {'index':'MONTH'}, inplace = True)
df1.rename(columns = {0:'P_Dengue'}, inplace = True)
df1
```

	MONTH	P_Dengue
0	2020-05-01	7.381962
1	2020-06-01	7.898427
2	2020-07-01	7.761967
3	2020-08-01	7.563454
4	2020-09-01	7.458644
5	2020-10-01	7.400729
6	2020-11-01	7.358917
7	2020-12-01	7.327743
8	2021-01-01	7.305656
9	2021-02-01	7.290277
10	2021-03-01	7.279503
11	2021-04-01	7.271916
12	2021-05-01	7.266575
13	2021-06-01	7.262820
14	2021-07-01	7.260179
15	2021-08-01	7.258322
16	2021-09-01	7.257016
17	2021-10-01	7.256097
18	2021-11-01	7.255451
19	2021-12-01	7.254997

Fig.69. Squeezing series to data frame

```
cor_df = pd.merge(cor_df,df1, how="outer", on=["MONTH"])
#cor_df=cor_df.drop('P_Temp_y')
cor_df
```

	MONTH	A_Dengue	P_Humidity	P_Temp	P_Dengue
0	2020-05-01	7	82.274471	28.747183	7.381962
1	2020-06-01	10	80.859788	26.862787	7.898427
2	2020-07-01	7	80.164776	25.339134	7.761967
3	2020-08-01	2	78.077973	28.800164	7.563454
4	2020-09-01	4	79.040332	27.602726	7.458644
5	2020-10-01	3	77.592562	25.531501	7.400729
6	2020-11-01	7	79.765974	25.737112	7.358917
7	2020-12-01	8	85.467107	28.094887	7.327743
8	2021-01-01	3	86.289981	28.303510	7.305656
9	2021-02-01	5	89.380470	27.838706	7.290277
10	2021-03-01	5	85.128861	24.934021	7.279503
11	2021-04-01	9	86.063402	26.059515	7.271916
12	2021-05-01	12	85.964018	26.542229	7.266575
13	2021-06-01	7	82.363388	25.254514	7.262820
14	2021-07-01	1	81.255069	26.587319	7.260179
15	2021-08-01	0	0.774560	0.262111	7.258322
16	2021-09-01	1	73.375737	23.867009	7.257016
17	2021-10-01	2	81.511497	25.937936	7.256097
18	2021-11-01	9	89.440800	29.658161	7.255451
19	2021-12-01	4	88.526608	27.628668	7.254997

Fig.70. Squeezing all the required variables

A.3 SQL Function to Extract Female Mosquito Specimens

```
DELIMITER //
CREATE FUNCTION `EXTRACT_FEMALE` (sx_string VARCHAR(50), amount INTEGER)
RETURNS INTEGER
BEGIN
DECLARE F_A INTEGER;
IF (sx_string LIKE '%|%') THEN SET F_A = CONVERT(SUBSTRING_INDEX(SUBSTRING_INDEX(sx_string, "|", -1), "Female", 1), INTEGER);
ELSEIF (sx_string LIKE "%Female%") THEN SET F_A = amount;
```

```
ELSE SET F_A = 0;
END IF;
RETURN (F_A);
END//
DELIMITER ;
```

A.4 SQL Queries to create Dengue and Mosquito Upload tables to upload into

Creating DENGUE_RAW table

```
CREATE TABLE DENGUE_RAW
(
   Object_ID INTEGER NOT NULL,
   Loc_Name VARCHAR(255) NOT NULL,
   Lon DECIMAL(18,16) NOT NULL,
   Lat DECIMAL (17,16) NOT NULL,
   Process VARCHAR(50) NOT NULL,
   Match_addr VARCHAR(155) NOT NULL,
   Neighborhood VARCHAR(155) NOT NULL,
   Sex CHAR(1) NOT NULL,
   Age INTEGER NOT NULL,
   Occupation INTEGER(4) NOT NULL,
   Ethnicity INTEGER(1) NOT NULL,
   Consultation DATE NOT NULL,
   Symptoms DATE NOT NULL,
   Loc_ID VARCHAR(12) NOT NULL,
   County VARCHAR(12) NOT NULL
   );
```

Adding Unique Auto incrementing Identifier to DENGUE_RAW after importing from CC_Translated

```
ALTER TABLE DENGUE ADD Unique_ID int(5) NOT NULL PRIMARY KEY
AUTO_INCREMENT;
```

Creating MOSQUITO_RAW table

```
CREATE TABLE MOSQUITO_RAW
(
   ID    VARCHAR(155) NOT NULL,
   Individual INTEGER NOT NULL,
```

```
    Sex VARCHAR(50) NOT NULL,
    Remarks VARCHAR(50),
    Event_ID VARCHAR(50),
    Event_Date DATE NOT NULL,
    Location_Num VARCHAR(15) NOT NULL,
    County VARCHAR(12) NOT NULL,
    Locality VARCHAR(155) NOT NULL,
    Elevation VARCHAR(6) NOT NULL,
    Lat DECIMAL(17,16) NOT NULL,
    Lon DECIMAL(18,16) NOT NULL,
    Scientific_Name VARCHAR(50) NOT NULL,
    Higher_Class VARCHAR(155) NOT NULL,
    Genus VARCHAR(50) NOT NULL,
    Subgenus VARCHAR(50) NOT NULL,
    SpecificEpi VARCHAR(50) NOT NULL,
    Vernacular VARCHAR(50) NOT NULL
);
```

A.5 SQL Queries to Create Final Count-Condition Tables

Creating MOS_MET table, meeting with appropriate meteorological conditions

```
CREATE TABLE MOS_MET AS
SELECT EXTRACT_FEMALE(Mos.Sex, Mos.Individual) as FEMALE_COUNT, Mos.Genus as GENUS, Mos.Lat as LATITUDE, Mos.Lon as LONGITUDE, Mos.County as COUNTY,
NC.air_temp_max_c as TEMP_MAX, NC.air_temp_min_c as TEMP_MIN,
NC.air_temp_aver_c as TEMP_AVER, NC.humidity as HUMIDITY
FROM MOSQUITO_RAW as Mos INNER JOIN NC_Data as NC ON Mos.Event_Date = NC.Time
INNER JOIN
(SELECT Mos.ID, MIN(ABS(Mos.Lat - NC.lat) + ABS(Mos.Lon - NC.lon)) AS min_abs_value
FROM MOSQUITO_RAW as Mos INNER JOIN NC_Data as NC ON Mos.Event_Date = NC.Time
GROUP BY Mos.ID) as t ON Mos.ID = t.ID AND ABS(Mos.Lat - NC.lat) + ABS(Mos.Lon - NC.lon) = t.min_abs_value;
```

Creating Intermediate Table DENGUE_METEOR, DENGUE_METEOR_TWO, DENGUE_METEOR_FOUR

```
CREATE TABLE DENGUE_METEOR AS
SELECT Den.Unique_ID, Den.County, Den.Symptoms, Den.Lat as LATITUDE, Den.Lon as LONGITUDE, NC.Time, NC.air_temp_max_c, NC.air_temp_min_c, NC.air_temp_aver_c, NC.humidity FROM DENGUE_RAW as Den INNER JOIN NC_Data as NC ON Den.Symptoms = NC.Time INNER JOIN
(SELECT Den.Unique_ID, MIN(ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon)) AS min_abs_value FROM DENGUE_RAW as Den INNER JOIN NC_Data as NC ON Den.Symptoms = NC.Time
GROUP BY Den.Unique_ID) as t ON Den.Unique_ID = t.Unique_ID AND
ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon) = t.min_abs_value;

CREATE TABLE DENGUE_METEOR_TWO AS SELECT Den.Unique_ID, Den.County, Den.Symptoms, Den.Lat as LATITUDE, Den.Lon as LONGITUDE, NC.Time, NC.air_temp_max_c,
NC.air_temp_min_c, NC.air_temp_aver_c, NC.humidity FROM DENGUE_RAW as Den
INNER JOIN NC_Data as NC ON (Den.Symptoms - INTERVAL 2 WEEK) = NC.Time INNER JOIN
(SELECT Den.Unique_ID, MIN(ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon)) AS min_abs_value FROM DENGUE_RAW as Den INNER JOIN NC_Data as NC ON (Den.Symptoms - INTERVAL 2 WEEK) = NC.Time GROUP BY Den.Unique_ID) as t ON Den.Unique_ID = t.Unique_ID AND ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon) = t.min_abs_value;

CREATE TABLE DENGUE_METEOR_FOUR AS SELECT Den.Unique_ID, Den.County, Den.Symptoms, Den.Lat as LATITUDE, Den.Lon as LONGITUDE, NC.Time, NC.air_temp_max_c, NC.air_temp_min_c, NC.air_temp_aver_c, NC.humidity FROM DENGUE_RAW as Den
INNER JOIN NC_Data as NC ON (Den.Symptoms - INTERVAL 4 WEEK) = NC.Time INNER JOIN
(SELECT Den.Unique_ID, MIN(ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon)) AS min_abs_value FROM DENGUE_RAW as Den INNER JOIN NC_Data as NC ON (Den.Symptoms - INTERVAL 4 WEEK) = NC.Time GROUP BY Den.Unique_ID) as t ON Den.Unique_ID = t.Unique_ID AND
ABS(Den.Lat - NC.lat) + ABS(Den.Lon - NC.lon) = t.min_abs_value;
```

Create count with average conditions tables DENGUE_AVER, DENGUE_TWO_AVER, DENGUE_FOUR_AVER

```
CREATE TABLE DENGUE_AVER AS SELECT COUNT(*) as DENGUE_COUNT, YEAR(Symptoms), MONTHNAME(Symptoms), AVG(air_temp_max_c) as AVER_MAX,
AVG(air_temp_min_c) as AVER_MIN, AVG(air_temp_aver_c) as AVER_AVER, AVG(humidity) as AVER_HUMIDITY FROM `DENGUE_METEOR`
GROUP BY YEAR(Symptoms), MONTH(Symptoms);

CREATE TABLE DENGUE_TWO_AVER AS SELECT COUNT(*) as DENGUE_COUNT, YEAR(Symptoms), MONTHNAME(Symptoms), AVG(air_temp_max_c) as AVER_MAX,
AVG(air_temp_min_c) as AVER_MIN, AVG(air_temp_aver_c) as AVER_AVER, AVG(humidity) as AVER_HUMIDITY FROM `DENGUE_METEOR_TWO`
GROUP BY YEAR(Symptoms), MONTH(Symptoms);
CREATE TABLE DENGUE_FOUR_AVER AS SELECT COUNT(*) as DENGUE_COUNT, YEAR(Symptoms), MONTHNAME(Symptoms), AVG(air_temp_max_c) as AVER_MAX, AVG(air_temp_min_c) as AVER_MIN, AVG(air_temp_aver_c) as AVER_AVER,
AVG(humidity) as AVER_HUMIDITY FROM `DENGUE_METEOR_FOUR`
GROUP BY YEAR(Symptoms), MONTH(Symptoms);
```

Create Granulated Monthly Dengue Count tables of DENGUE_METEOR_COUNT, DENGUE_METEOR_COUNT_TWO, DENGUE_METEOR_COUNT_FOUR

```
CREATE TABLE DENGUE_METEOR_COUNT AS SELECT Den.Unique_ID, Den.County, Den.Symptoms, Year(Den.Symptoms) as Y, Month(Den.Symptoms) as M, Den.Time, Den.LATITUDE, Den.LONGITUDE, DENGUE_AVER.DENGUE_COUNT as MONTH_COUNT,
Den.air_temp_max_c, Den.air_temp_min_c, Den.air_temp_aver_c, Den.humidity FROM DENGUE_METEOR as Den INNER JOIN DENGUE_AVER on YEAR(Den.Symptoms) = DENGUE_AVER.`YEAR(Symptoms)` and MONTHNAME(Den.Symptoms) = DENGUE_AVER.`MONTHNAME(Symptoms)`;
```

```
CREATE TABLE DENGUE_METEOR_COUNT_TWO AS SELECT Den.Unique_ID, Den.County, Den.Symptoms, Year(Den.Symptoms) as Y, Month(Den.Symptoms) as M, Den.Time, Den.LATITUDE, Den.LONGITUDE, DENGUE_TWO_AVER.DENGUE_COUNT as MONTH_COUNT, Den.air_temp_max_c, Den.air_temp_min_c, Den.air_temp_aver_c, Den.humidity FROM DENGUE_METEOR_TWO as Den INNER JOIN DENGUE_TWO_AVER on YEAR(Den.Symptoms) = DENGUE_TWO_AVER.`YEAR(Symptoms)` and MONTHNAME(Den.Symptoms) = DENGUE_TWO_AVER.`MONTHNAME(Symptoms)`;
```

```
CREATE TABLE DENGUE_METEOR_COUNT_FOUR AS SELECT Den.Unique_ID, Den.County, Den.Symptoms, Year(Den.Symptoms) as Y, Month(Den.Symptoms) as M, Den.Time, Den.LATITUDE, Den.LONGITUDE, DENGUE_FOUR_AVER.DENGUE_COUNT as MONTH_COUNT, Den.air_temp_max_c, Den.air_temp_min_c, Den.air_temp_aver_c, Den.humidity FROM DENGUE_METEOR_FOUR as Den INNER JOIN DENGUE_FOUR_AVER on YEAR(Den.Symptoms) = DENGUE_FOUR_AVER.`YEAR(Symptoms)` and MONTHNAME(Den.Symptoms) = DENGUE_FOUR_AVER.`MONTHNAME(Symptoms)`;
```

References

1. WHO : https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue

2. López-Montenegro, L. E., Pulecio-Montoya, A. M., & Marcillo-Hernández, G. A. (2019). Dengue Cases in Colombia: Mathematical Forecasts for 2018-2022. *MEDICC review*, *21*(2-3), 38–45. https://doi.org/10.37757/MR2019.V21.N2-3.8

3. Gutierrez-Barbosa H, Medina-Moreno S, Zapata JC, Chua JV. Dengue Infections in Colombia: Epidemiological Trends of a Hyperendemic Country. Tropical Medicine and Infectious Disease. 2020 Oct;5(4):E156. DOI: 10.3390/tropicalmed5040156. PMID: 33022908; PMCID: PMC7709707. https://europepmc.org/article/PMC/PMC7709707