A Mini Project Report

On

**TIME SERIES ANALYSIS-BASED PREDICTION OF DENGUE SPREAD USING CLIMATE DATA**

*Submitted to JNTU HYDERABAD*

*In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted By

**B.N.Gayathri (208R1A0511)**

**N.Swapnil (208R1A0542)**

**R.Eeshwar (208R1A0549)**

**B.Sai Rohit (208R1A0507)**

Under the Esteemed guidance of

**Mr. E. Suresh Babu**

Associate Professor, Department of CSE



**Department of Computer Science & Engineering**

**CMR ENGINEERING COLLEGE**

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**2022-2023**

**CMR ENGINEERING COLLEGE**

*(Accredited by NBA,Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501 401*

**Department of Computer Science & Engineering**



**CERTIFICATE**

This is to certify that the project entitled **“TIME SERIES ANALYSIS-BASED PREDICTION OF DENGUE SPREAD USING CLIMATE DATA”** is a bonafide work carried out by

**B. N. Gayathri (208R1A0511)**

**N. Swapnil (208R1A0542)**

**R. Eeshwar (208R1A0549)**

**B. Sai Rohit (208R1A0507)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

|  |  |  |  |
| --- | --- | --- | --- |
| **Internal Guide** | **Mini Project Coordinator** | **Head of the Department** | **External Examiner** |
| **Mr. E. Suresh Babu** | **Mr. S. Kiran Kumar** | **Dr. Sheo Kumar** |  |
| Associate Professor | Assistant Professor | Professor & H.O.D |  |
| CSE, CMREC | CSE, CMREC | CSE, CMREC |  |

**DECLARATION**

This is to certify that the work reported in the present project entitled " **TIME SERIES ANALYSIS-BASED PREDICTION OF DENGUE SPREAD USING CLIMATE DATA”** is a record of bonafide work done by us in the Department of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

**B .N. Gayathri (208R1A0511)**

**N. Swapnil (208R1A0542)**

**R.Eeshwar (208R1A0549)**

**B.Sai Rohit (208R1A0507)**

**ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr.Sheo Kumar**, HOD, **Department of CSE, CMR Engineering College** fortheir constant support**.**

I am extremely thankful to **Mr. E. Suresh Babu,** Associate Professor, Internal Guide, Department of CSE, for his/ her constant guidance, encouragement and moral support throughout the project.

I will be failing in duty if I do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

I thank **S Kiran Kumar** Mini Project Coordinator for his constant support in carrying out the project activities and reviews.

I express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, I am very much thankful to my parents who guided me for every step.

**B .N. Gayathri (208R1A0511)**

**N. Swapnil (208R1A0542)**

**R.Eeshwar (208R1A0549)**

**B.Sai Rohit (208R1A0507)**

**CONTENTS**

**TOPIC PAGENO**

**ABSTRACT** i

**List of Figures ii**

**List of Tables iii**

1. **INTRODUCTION**

1.1. Introduction & Objectives

1.2. Purpose of the Project

1.3 Existing system & Disadvantages

1.4 Proposed system with Features

**2. LITERATURE SURVEY**

**3. SOFTWARE REQUIREMENT ANALYSIS**

3.1. Problem Specification

* 1. 3.2. Modules and their Functionalities

3.3. Functional Requirements

3.4. Non Functional Requirements

3.5. Feasibility Study

4. **SOFTWARE & HARDWARE REQUIREMENTS**

4.1. Software requirements

4.2. Hardware requirements

**5. SOFTWARE DESIGN**

5.1 Data Flow Diagrams

5.2. Control Flow Diagrams

5.3. ER Diagrams

5.4. UML Diagrams

**6. CODING AND IMPLEMENTATION**

6.1 .Sample code

6.2. Data Dictionary

**7. SYSTEM TESTING**

7.1 Testing Strategies

**8. OUTPUT SCREENS**

**9. CONCLUSION**

**10. FUTURE ENHANCEMENTS**

**11. BIBLIOGRAPHY AND REFERENCES**

**12. APPENDICES**

**ABSTRACT**

Dengue is a human arbovirus disease transmitted by the female mosquito of the genus Aedes, mainly Aedes aegypti and Ae. albopictus. Dengue, the most frequent arthropod-borne viral disease, is prevalent in tropical and subtropical regions. Two major clinical forms of dengue illness involve the mild form of dengue fever and severe form mostly characterized by plasma leakage with or without haemorrhage. Two-fifths of the world population (about 2.5 billion people) is at risk of dengue infection. The prevalence of this disease has grown dramatically in the recent decades. Between 50 and 100 million people are infected each year worldwide and more than 500,000 are hospitalized. The average annual incidence was multiplied by thirty in the last fifty years. Incidence of dengue haemorrhagic fever (DHF) is increasing in many tropical regions inducing 20,000 deaths per year, mostly among children under 15 years. Dengue is endemic in all surrounding countries with the four serotypes circulating in the region within a period of ten years. Countries or territories with the highest number of reported dengue cases were Puerto Rico, the Dominican Republic, Martinique, Trinidad and Tobago and French Guiana. Population movement is an important factor in the virus dissemination. It contributes to carry new virus strains, but it also participates to introduce nonimmune subjects in an endemic area. This proposed system is built to predict the spread of dengue fever with climate data using the concept of time series analysis. In addition, this project also performs the exploratory data analytics on the dengue dataset over a period of time. Finally, prediction analysis also performed with the usage of advancement rendered by machine learning algorithms.

**LIST OF FIGURES**

**S.NO DESCRIPTION PAGENO**

3.1 Functional Requirements 4

3.2 Non Functional Requirements 5

5.2 Context Flow Diagram 11

5.2 Data Flow diagram For Admin 12

5.2 Data Flow Diagram For Staff 13

5.2 Data Flow Diagram for Customer 14

5.3 ER-Diagram 16

5.4 Class Diagram 18

5.4 Use case Diagram Of Admin 19

5.4 Use case Diagram Of Staff 19

5.4 Use case Diagram Of Customer 20

5.4 Sequence Diagram 21

5.4 Activity Diagram Of Admin 23

5.4 Activity Diagram of Staff 23

5.4 Activity Diagram of Customer 24

5.4 Component Diagram 25

5.4 Deployment Diagram 26

8.1 Home Page 51

8.2 Projects Page 51

8.4 Customer Registration Page 52

8.5 Login Page 53

8.6 Admin Home Page 53

8.7 Admin Add Staff 54

8.8 Customer Registraion View By Admin 54

8.9 Reports View 55

8.10 Complaints View By Admin 55

8.11 Staff Home page 56

8.12 Complaints View By Staff 56

8.13 Reports Generated By Staff 57

8.14 Customer Home Page 57

8.15 Complaint Submission Form 58

8.16 Feed Back Submission Form 58

**LIST OF TABLES**

**S.NO DESCRIPTION PAGENO**

1 Feed back 37

2 Contactus 37

3 Customer registration 38

4 Add Complaints 39

5 Complaint Status 39

6 Assign Complaints 40

7 Add Report 40

8 Add Staff 41

**1. INTRODUCTION**

**1.1. Introduction to Project:**

Dengue is a potentially life-threatening arboviral disease transmitted by female Aedes mosquitoes, especially A. aegypti, A. albopictus, and A. vitattus. These vectors are common tropical hematophagous ectoparasites. This zoonotic disease spread from African or Asian non-human primates 500 to 1000 years ago, but within the last 60 years it has spread from just 9 countries experiencing severe epidemics to become endemic in over 100 countries worldwide, even affecting non-tropical or subtropical areas. Moreover, approximately one hundred million people yearly suffer from the symptomatic disease caused by its four serotypes. Given the significant impact of environmental changes on disease transmission, the One Health approach is urgently needed to implement the integration between human, animal, and ecological health.

The objective of this paper is to provide an insight into techniques that can be used for future predictive models based on the One Health perspective, particularly in respect to Latin America but also elsewhere.

One Health is a multidisciplinary approach that acknowledges the synergy between human and animal health and their shared environment. This idea is not new; the noted nineteenth-century pathologist (and originator of the term zoonosis) Rudolph Virchow famously asserted in 1858 that “between animal and human medicine, there are no dividing lines—nor should there be”.

This approach has become increasingly important in the 21st Century with the convergence of the pressures of changing climate, migration of human and animal populations, and the growing human population that increases the proximity between wildlife and humans. Indeed, the term One Health was only coined in the early 2000s with the appearance of the zoonotic SARS and H5N1 influenza diseases.

Whilst the One Health perspective is widely seen as necessary and increasingly used for better disease control, epidemiological approaches have not kept up with this change. Conventional epidemiological perspectives tend to view disease broadly from a human-only perspective, focusing on human demographic conditions with often only climatic/environmental factors accommodating the disease vector health. In contrast, One Health requires the health and lifecycle of the zoonotic disease vectors to be explicitly considered alongside the human environment, demographics, and interaction with the zoonotic host vectors.

For example, whilst environmental and sociological considerations often take a back seat in One Health, they frequently occupy the centre stage in epidemiology. Factors such as mean temperatures and rainfall used in predicting dengue, with a very vague consideration of how they affect the mosquito vectors, are an emergent challenge to be considered. High rainfall, for instance, is beneficial to mosquitoes because it provides water-filled locations for eggs and larvae, whilst the mosquitoes are primarily impervious to strikes by raindrops that might otherwise kill them. In addition, temperature and rain generally affect many other infectious and tropical diseases.Top of Form

* 1. **Purpose of the Project**

The purpose of this system is to predict the spread of dengue fever with climate data using the concept of time series analysis. In addition, this project also performs the exploratory data analytics on the dengue dataset over a period of time. Finally, prediction analysis also performed with the usage of advancement rendered by machine learning algorithms.

**1.2 Proposed system**

This research work combines time series analysis techniques, feature engineering, and machine learning using the XGBoost algorithm to predict dengue spread based on climate data. It contributes to early warning systems for dengue outbreaks and supports public health efforts to mitigate the impact of the disease.

1. **LITERATURE SURVEY**

**Majeed, M.A.; Shafri, H.Z.M.; [1]** dengue fever cases in Malaysia using machine learning techniques. A dataset consisting of weekly dengue cases at the state level in Malaysia from 2010 to 2016 was obtained from the Malaysia Open Data website and includes variables such as climate, geography, and demographics. Six different long short-term memory (LSTM) models were developed and compared for dengue prediction in Malaysia: LSTM, stacked LSTM (S-LSTM), LSTM with temporal attention (TA-LSTM), S-LSTM with temporal attention (STA-LSTM), LSTM with spatial attention (SA-LSTM), and S-LSTM with spatial attention (SSA-LSTM).

**Cabrera, M.; Leake, J.; [2]** epidemiological prediction of dengue fever using the One Health perspective, including an analysis of how Machine Learning techniques have been applied to it and focuses on the risk factors for dengue in Latin America to put the broader environmental considerations into a detailed understanding of the small-scale processes as they affect disease incidence. Determining that many factors can act as predictors for dengue outbreaks, a large-scale comparison of different predictors over larger geographic areas than those currently studied is lacking to determine which predictors are the most effective.

**Dey, Samrat Kumar, et al. [3]** develop a machine learning model that can use relevant information about the factors that cause Dengue outbreaks within a geographic region. To predict dengue cases in 11 different districts of Bangladesh, we created a DengueBD dataset and employed two machine learning algorithms, Multiple Linear Regression (MLR) and Support Vector Regression (SVR). This research also explores the correlation among environmental factors like temperature, rainfall, and humidity with the rise and decline trend of Dengue cases in different cities of Bangladesh. The entire dataset was divided into an 80:20 ratio, with 80 percent used for training and 20% used for testing. The research findings imply that, for both the MLR with 67% accuracy along with Mean Absolute Error (MAE) of 4.57 and SVR models with 75% accuracy along with Mean Absolute Error (MAE) of 4.95, the number of dengue cases reduces throughout the winter season in the country and increases mainly during the rainy season in the next ten months, from August 2021 to May 2022.

**Kakarla, S.G., Kondeti, P.K., et al. [4]** applied vector auto regression, generalized boosted models, support vector regression, and long short-term memory (LSTM) to predict the dengue prevalence in Kerala state of the Indian subcontinent. Consider the number of dengue cases as the target variable and weather variables viz., relative humidity, soil moisture, mean temperature, precipitation, and NINO3.4 as independent variables. Various analytical models have been applied on both datasets and predicted the dengue cases. Among all the models, the LSTM model was outperformed with superior prediction capability (RMSE: 0.345 and R2:0.86) than the other models.

**Roster, Kirstin, et al. [5]** developed a model for predicting monthly dengue cases in Brazilian cities 1 month ahead, using data from 2007–2019. We compared different machine learning algorithms and feature selection methods using epidemiologic and meteorological variables. They found that different models worked best in different cities, and a random forests model trained on monthly dengue cases performed best overall. It produced lower errors than a seasonal naive baseline model, gradient boosting regression, a feed-forward neural network, or support vector regression.

**Sarder, Faysal, et al. [6]** predict the accuracy of dengue outbreak from climate data. A dengue dataset, containing information of climate variables, dengue cases during 2019 to 2021 from Meteorology Department and Directorate General of Health Services (DGHS), Bangladesh. We split the whole dataset into 70:30 ratios were 70% considered as training and 30% for testing purposes. Such, prediction of accuracy we apply various supervised machine learning (ML) algorithms like Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), Naïve Bayes (NB), AdaBoostClassifier (AdaBoost), XGBRegressor, GradientBoostingClassifier and Random Forest (RF). Finally, from these algorithms, SVM provide the highest accuracy of 96.73%.

**Ochida, N., Mangeas, M., et al. [7]** proposed statistical estimation of the effective reproduction number (Rt) based on case counts to create a categorical target variable: epidemic week/non-epidemic week. A machine learning classifier has been trained using relevant climate indicators in order to estimate the probability for a week to be epidemic under current climate data and this probability was then estimated under climate change scenarios.

**Anuranjan, M. B., et al. [8]** considered three different modelling techniques: interpolation, gradient boosting regression and random forest regression. Parameters were tuned and adjusted for optimal performance. Results are based on prediction accuracy and mean absolute error (MAE). The performance was analysed, and the result points out that the gradient boosting regression performs significantly better than the other models and is therefore considered to be a better approach. Future results can be improved by obtaining large amounts of meaningful data and implementing better models associated with time series predicting.

**Gupta, G.; Khan, S.; et al. [9]** developed dengue predictive models, data from microarrays and RNA-Seq have been used significantly. Bayesian inferences and support vector machine algorithms are two examples of statistical methods that can mine opinions and analyze sentiment from text. In general, these methods are not very strong semantically, and they only work effectively when the text passage inputs are at the level of the page or the paragraph; they are poor miners of sentiment at the level of the sentence or the phrase.

**Rocha, F.P., Giesbrecht, M., et al. [10]** models were trained with data from the municipality of São Luís do Maranhão, state of Maranhão, Brazil. The majority of related works analyze states, countries, or continental levels, with greater availability of data. To apply the approach to such a small region, some oversampling techniques were used. The number of cases per neighborhood from 2014 to and 2020 and climatic, territorial, and environmental data was used as input variables to estimate the probability of dengue occurrence in the municipality. Due to the unbalanced database, we used the SMOTE, ADASYN, and DBSMOTE oversampling techniques. The DBSMOTE-trained Random Forest classifier achieved the best results with a 75.1% AUC, 75.43% sensitivity and a 60.53% specificity.

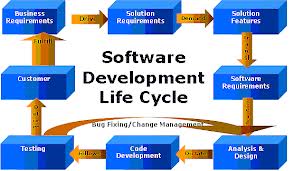
[11] proposed a method that combines these three factors with data of Taiwanese dengue fever and uses the secondary area divided by the population as the granularity. Random Forest (RF) and XGBoost (XGB) are used for prediction model of weekly dengue fever infection area. Experimental results showed that the Receiver Operator Characteristic (ROC)/Area Under the Curve (AUC) of RF and XGB are both higher than 93%, and the Recall rate is higher than 80%. With the result, government can determine which area should do disinfection process to reduce the infection rate of dengue infection. Because of accurate prediction and disinfection process, the personnel cost can be reduced and it can prevent waste of medical recourse.

**Pacheco, Paolo Ramon DC et al. [12]** developed from machine learning algorithms were often used to provide accurate predictions as it can analyze trends from historical dengue data. Currently, the basis for the predictions of machine learning algorithms is unknown, which is why it is termed a “black-box”. Climate-based random forest models were created through two implementations: randomForest package and ranger package. The ranger package slightly outperformed the conventional randomForest package, making it a better alternative in implementing random forest. Furthermore, models yielded more accurate predictions of dengue incidences with a delayed effect on the datasets. For local and global interpretations, most of the best models had relative humidity as the most influential to dengue incidence in Metropolitan Manila at all spatial scales.

**3. SOFTWARE REQUIREMENT ANALYSIS**

**3.1. SDLC:**

The **Systems Development Life Cycle (SDLC)** or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



**Figure 3.1(a):** Software Development Life Cycle

**Requirement Analysis and Design:**

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

**Implementation:**

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, Java, .Net are used for coding. With respect to the type of application, the right programming language is chosen.

**Testing:**

In this phase the system is tested. Normally programs are written as a series of individual modules, these subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

**Maintenance:**

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

**3.2.System Study:**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Classified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**3.3. Modules and their Functionalities:**

**3.4. Present work and process model used with justification:**

**4. Feasibility Study**

**4.1 Feasibility Report:**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are:**

* Technical Feasibility
* Social Feasibility
* Economical Feasibility

**4.2 Technical Feasibility:**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**4.3. Social Feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**4.4. Economical Feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

1. **SYSTEM REQUIREMENTS SPECIFICATION**

**5.1.Requirement Specification:**

A requirement specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of usecases that describe all the interactions the users will have with the software. In addition to usecases, the SRS also contains non-functional requirements. Non-functional requirements which impose constraints on the design or implementation such as performance engineering requirements, quality standards.

System requirement specification is a structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify the business problems and propose solutions. Within the system development life cycle domain, the business analyst typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

* 1. **Hardware Requirements:**

|  |  |
| --- | --- |
| RAM : | Minimum 4GB |
| Processor : | Minimum Intel i3 |
| Hard Disk Size : | Minimum 250GB |

* 1. **Software Requirements:**

|  |  |
| --- | --- |
| Operating System : | Windows, Linux |
| IDE & Tools : | Python IDLE 3.7, Anaconda 3.7, Jupiter, Google colab |
| Technologies : | Python |

* 1. **SELECTED SOFTWARE:**

1. **Introduction to Python:**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

* + Machine Learning
  + GUI Applications (like Kivy, Tkinter, PyQt etc.)
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

**Advantages of Python**

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system.  
Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. "Some changes in Python 7.3:

* Print is now a function.
* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.‍

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 5.4.1:** Python installation site

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

**Figure 5.4.2:** Download Python

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

**Figure 5.4.3:** Select Python 3.7.4 file

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

**Figure 5.4.4:** Select operating system

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

**Figure 5.4.5:** Open Downloaded Python

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Figure 5.4.6:** Install Python

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Figure 5.4.7:** Setup successful

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 5.4.8:** Check the version

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated  
**Figure 5.4.9:** Execution in IDLE

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability" \o "Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



**Figure 5.4.10:** Django

Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



**Figure 5.4.11:** Django Project Model

**6. SOFTWARE DESIGN**

**System Architecture:**

Figure 6.1 shows the proposed system model. The detailed operation illustrated as follows:

**Step** **1. Exploratory Data Analysis (EDA):**

* **Data Collection**: Gather historical climate data and dengue spread records. This data typically includes variables such as temperature, humidity, rainfall, and the number of dengue cases over time.
* **Data Inspection**: Examine the dataset's structure, including its size, data types, and any missing values. Ensure that the data is correctly formatted for time series analysis.
* **Time Series Decomposition**: Decompose the time series data into its constituent components, including trend, seasonality, and residual noise. This helps in understanding the underlying patterns in the data.
* **Data Visualization**: Create visualizations such as line plots, histograms, and seasonal decomposition plots to visualize the time series data. These visualizations can reveal trends, seasonality, and any anomalies.
* **Correlation Analysis**: Perform correlation analysis to identify relationships between climate variables (e.g., temperature, rainfall) and dengue spread. This helps in selecting relevant features for modeling.

**Step 2. Preprocessing of Dataset:**

* **Handling Missing Data**: Address any missing data points in the time series, using techniques like imputation or interpolation to fill gaps.
* **Feature Engineering**: Create additional features, such as lag variables (previous time steps), moving averages, and seasonality indicators, to capture relevant patterns in the data.
* **Normalization/Scaling**: Normalize or scale the data if necessary to ensure that all features have similar scales.
* **Train-Test Split**: Divide the time series data into training and testing sets. Typically, earlier data points are used for training, and more recent data points are reserved for testing.

**Step 3. XGBoost Model Training:**

* **Selecting Features**: Choose the relevant climate variables and engineered features as input features (X) and the number of dengue cases as the target variable (y).
* **Hyperparameter Tuning**: Tune the hyperparameters of the XGBoost model, such as learning rate, maximum depth, and the number of estimators (trees), using techniques like grid search or random search.
* **Model Training**: Train the XGBoost model using the training data. XGBoost is a gradient boosting algorithm known for its effectiveness in time series forecasting.

**Step 4. Prediction:**

* **Model Evaluation**: Use the trained XGBoost model to make predictions on the test dataset. Evaluate the model's performance using appropriate metrics for time series forecasting, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).
* **Visualization of Predictions**: Visualize the model's predictions against the actual dengue cases on a time series plot. This allows you to assess how well the model captures the patterns and trends in the data.

A diagram of a software process

Description automatically generated

**Figure 6.1:** Proposed System model.

**6.1. Dataset description**

The dataset contains various features, including climatic variables, for different cities over multiple years. Here's a description of the columns:

* city: The name of the city where the data was collected (sj for San Juan, iq for Iquitos).
* year: The year in which the data was recorded.
* weekofyear: The week of the year in which the data was recorded.
* week\_start\_date: The start date of the week.
* ndvi\_ne, ndvi\_nw, ndvi\_se, ndvi\_sw: Normalized Difference Vegetation Index (NDVI) values for different quadrants (NE, NW, SE, SW).
* precipitation\_amt\_mm: The amount of precipitation in millimeters.
* reanalysis\_air\_temp\_k: The air temperature in Kelvin.
* reanalysis\_avg\_temp\_k: The average temperature in Kelvin.
* reanalysis\_dew\_point\_temp\_k: The dew point temperature in Kelvin.
* reanalysis\_max\_air\_temp\_k: The maximum air temperature in Kelvin.
* reanalysis\_min\_air\_temp\_k: The minimum air temperature in Kelvin.
* reanalysis\_precip\_amt\_kg\_per\_m2: The amount of precipitation in kilograms per square meter.
* reanalysis\_relative\_humidity\_percent: The relative humidity as a percentage.
* reanalysis\_sat\_precip\_amt\_mm: The saturated amount of precipitation in millimeters.
* reanalysis\_specific\_humidity\_g\_per\_kg: The specific humidity in grams per kilogram.
* reanalysis\_tdtr\_k: The diurnal temperature range in Kelvin.
* station\_avg\_temp\_c: The average temperature recorded by a weather station in degrees Celsius.
* station\_diur\_temp\_rng\_c: The diurnal temperature range recorded by a weather station in degrees Celsius.
* station\_max\_temp\_c: The maximum temperature recorded by a weather station in degrees Celsius.
* station\_min\_temp\_c: The minimum temperature recorded by a weather station in degrees Celsius.
* station\_precip\_mm: The amount of precipitation recorded by a weather station in millimeters.
* total\_cases: The total number of Dengue fever cases recorded.
* The dataset contains a total of 1456 rows (or records) and 25 columns (or features).
* Each row represents data for a specific week in a specific city and year, with corresponding climatic and environmental variables, as well as the total number of Dengue fever cases

recorded during that week.

**6.2. UML Diagrams:**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

* UML stands for **U**nified **M**odeling **L**anguage.
* UML is a pictorial language used to make software blue prints.

# UML Modeling Types:

It is very important to distinguish between the UML model. Different diagrams are used for different type of UML modeling. There are three important type of UML modelings:

**6.2.1 Structural Things:**

Structural things are classified into seven types those are as follows:

|  |
| --- |
|  |

**Class diagram:**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaboration. Class diagrams basically represent the object oriented view of a system which is static in nature. Active class is used in a class diagram to represent the concurrency of the system.

Class diagram represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

A screenshot of a computer

Description automatically generated

**Figure 6.2.1.1:** Class Diagram

**Use Case Diagram:**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

* Functionalities to be represented as an use case
* Actors
* Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

* The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
* Give a suitable name for actors.
* Show relationships and dependencies clearly in the diagram.
* Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
* Use note when ever required to clarify some important points…

A diagram of a person with text

Description automatically generated

**Figure 6.2.1.2:** Use Case Diagram

**6.2.2 Behavioral Things**

Behavioural things are considered as verbs of a model.These are the ‘dynamic' parts which describes how the model carry out its functionality with respect to time and space. Behavioral things are classified into two types:

From the term Interaction, it is clear that the diagram is used to describe some type of interactions among the different elements in the model. This interaction is a part of dynamic behavior of the system.

**Purpose of Interaction Diagrams**

The purpose of interaction diagrams is to visualize the interactive behavior of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction.

Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle.

The purpose of interaction diagram is −

* To capture the dynamic behaviour of a system.
* To describe the message flow in the system.
* To describe the structural organization of the objects.
* To describe the interaction among objects.

## **How to Draw an Interaction Diagram?**

As we have already discussed, the purpose of interaction diagrams is to capture the dynamic aspect of a system. So to capture the dynamic aspect, we need to understand what a dynamic aspect is and how it is visualized. Dynamic aspect can be defined as the snapshot of the running system at a particular moment

We have two types of interaction diagrams in UML. One is the sequence diagram and the other is the collaboration diagram. The sequence diagram captures the time sequence of the message flow from one object to another and the collaboration diagram describes the organization of objects in a system taking part in the message flow.

Following things are to be identified clearly before drawing the interaction diagram

* Objects taking part in the interaction.
* Message flows among the objects.
* The sequence in which the messages are flowing.
* Object organization.

Following are two interaction diagrams modeling the order management system. The first diagram is a sequence diagram and the second is a collaboration diagram

### **The Sequence Diagram**

The sequence diagram has four objects (Customer, Order, SpecialOrder and NormalOrder).

The following diagram shows the message sequence for SpecialOrder object and the same can be used in case of NormalOrder object. It is important to understand the time sequence of message flows.

A diagram of a computer program

Description automatically generated with medium confidence

**Figure 6.2.2.1:** Sequence Diagram

## **Where to Use Interaction Diagrams?**

We have already discussed that interaction diagrams are used to describe the dynamic nature of a system. Now, we will look into the practical scenarios where these diagrams are used. To understand the practical application, we need to understand the basic nature of sequence and collaboration diagram.

The main purpose of both the diagrams are similar as they are used to capture the dynamic behavior of a system. However, the specific purpose is more important to clarify and understand.

Sequence diagrams are used to capture the order of messages flowing from one object to another. Collaboration diagrams are used to describe the structural organization of the objects taking part in the interaction. A single diagram is not sufficient to describe the dynamic aspect of an entire system, so a set of diagrams are used to capture it as a whole.

Interaction diagrams are used when we want to understand the message flow and the structural organization. Message flow means the sequence of control flow from one object to another. Structural organization means the visual organization of the elements in a system.

Interaction diagrams can be used −

* To model the flow of control by time sequence.
* To model the flow of control by structural organizations.
* For forward engineering.
* For reverse engineering.

2.State chart diagram

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in the next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines the states, it is used to model the lifetime of an object.

## **Purpose of Statechart Diagrams**

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams −

* To model the dynamic aspect of a system.
* To model the life time of a reactive system.
* To describe different states of an object during its life time.
* Define a state machine to model the states of an object.

Activity diagram is another important diagram in UML to *describe the dynamic aspects of the system*.

A diagram of a data flow

Description automatically generated

**Figure 6.2.2.2:** Activity Diagram

**Deployment diagram** : The deployment diagram visualizes the physical hardware on which the software will be deployed.

A diagram of a device

Description automatically generated

**Figure 6.2.2.3:** Deployment Diagram

**6.3. XGBoost Model**

XGBoost is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "XGBoost is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the XGBoost takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Diagram

Description automatically generated

**Figure 6.3.1:** XGBoost algorithm.

XGBoost, which stands for "Extreme Gradient Boosting," is a popular and powerful machine learning algorithm used for both classification and regression tasks. It is known for its high predictive accuracy and efficiency, and it has won numerous data science competitions and is widely used in industry and academia. Here are some key characteristics and concepts related to the XGBoost algorithm:

* **Gradient Boosting:** XGBoost is an ensemble learning method based on the gradient boosting framework. It builds a predictive model by combining the predictions of multiple weak learners (typically decision trees) into a single, stronger model.
* **Tree-based Models:** Decision trees are the weak learners used in XGBoost. These are shallow trees, often referred to as "stumps" or "shallow trees," which helps prevent overfitting.
* **Objective Function:** XGBoost uses a specific objective function that needs to be optimized during training. The objective function consists of two parts: a loss function that quantifies the error between predicted and actual values and a regularization term to control model complexity and prevent overfitting. The most common loss functions are for regression (e.g., Mean Squared Error) and classification (e.g., Log Loss).
* **Gradient Descent Optimization:** XGBoost optimizes the objective function using gradient descent. It calculates the gradients of the objective function with respect to the model's predictions and updates the model iteratively to minimize the loss.
* **Regularization:** XGBoost provides several regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, to control overfitting. These regularization terms are added to the objective function.
* **Parallel and Distributed Computing:** XGBoost is designed to be highly efficient. It can take advantage of parallel processing and distributed computing to train models quickly, making it suitable for large datasets.
* **Handling Missing Data:** XGBoost has built-in capabilities to handle missing data without requiring imputation. It does this by finding the optimal split for missing values during tree construction.
* **Feature Importance:** XGBoost provides a way to measure the importance of each feature in the model. This can help in feature selection and understanding which features contribute the most to the predictions.
* **Early Stopping:** To prevent overfitting, XGBoost supports early stopping, which allows training to stop when the model's performance on a validation dataset starts to degrade.
* **Scalability:** XGBoost is versatile and can be applied to a wide range of machine learning tasks, including classification, regression, ranking, and more.
* **Python and R Libraries:** XGBoost is available through libraries in Python (e.g., **xgboost**) and R (e.g., **xgboost**), making it accessible and easy to use for data scientists and machine learning practitioners.

XGBoost, which stands for eXtreme Gradient Boosting, is a popular machine learning algorithm that is particularly effective for structured/tabular data and is often used for tasks like classification, regression, and ranking. It is an ensemble learning technique based on decision trees. Here's how XGBoost operates:

**Ensemble Learning**: XGBoost is an ensemble learning method, which means it combines the predictions from multiple machine learning models to make more accurate predictions than any single model. It uses an ensemble of decision trees, known as "boosted trees."

**Boosting**: Boosting is a sequential technique in which multiple weak learners (usually decision trees with limited depth) are trained one after the other. Each new tree tries to correct the errors made by the previous ones.

**Gradient Boosting**: XGBoost is a gradient boosting algorithm. It minimizes a loss function by adding weak models (trees) that minimize the gradient of the loss function at each stage. This is done by fitting a tree to the residuals (the differences between the predicted and actual values) of the previous model.

**Regularization**: XGBoost includes L1 (Lasso regression) and L2 (Ridge regression) regularization terms in the objective function to prevent overfitting. These regularization terms help control the complexity of individual trees and reduce the risk of overfitting the training data.

**Tree Pruning**: XGBoost uses a technique called "pruning" to remove branches of the trees that do not contribute significantly to the model's predictive power. This reduces the complexity of the trees and helps prevent overfitting.

**Feature Importance**: XGBoost provides a feature importance score, which helps you understand the contribution of each feature (input variable) in making predictions. You can use this information for feature selection and interpretation.

**Parallel and Distributed Computing**: XGBoost is designed for efficiency and can take advantage of parallel and distributed computing to train on large datasets faster.

**Handling Missing Data**: XGBoost can handle missing data by finding an optimal direction for missing values during tree construction.

**Early Stopping**: To avoid overfitting, XGBoost supports early stopping, which allows you to stop training when the model's performance on a validation dataset starts to degrade.

**Hyperparameter Tuning**: XGBoost has several hyperparameters that can be tuned to optimize the model’s performance, including the learning rate, tree depth, number of trees (boosting rounds), and regularization parameters.

**7. CODING AND IMPLEMEMTATION**

**7.1. Sample code:**

# XGBoost for Dengue Prediction

import pandas as pd

import numpy as np

import random

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

import statsmodels.api as sm

import statsmodels.formula.api as smf

from sklearn.dummy import DummyRegressor

import xgboost

from sklearn.metrics import mean\_absolute\_error

from sklearn.model\_selection import train\_test\_split

pd.set\_option('display.max\_rows', 50)

pd.set\_option('display.max\_columns', 50)

pd.set\_option('display.width', 1000)

get\_ipython().run\_line\_magic('matplotlib', 'inline')

features\_train = pd.read\_csv('data/dengue\_features\_train.csv')

labels\_train = pd.read\_csv('data/dengue\_labels\_train.csv')

features\_test = pd.read\_csv('data/dengue\_features\_test.csv')

features\_test

labels\_train

features = list(features\_train)

label = ['total\_cases']

df = features\_train.merge(labels\_train, how='left', left\_on=['city', 'year','weekofyear'], right\_on = ['city', 'year','weekofyear'])

DATA = df.copy()

df

# # Exploration

df.info()

df.describe()

df\_iq = df[df['city']=='iq'].sort\_values(by=['week\_start\_date'])

df\_sj = df[df['city']=='sj'].sort\_values(by=['week\_start\_date'])

plt.figure(figsize=(20, 5))

plt.plot(df\_sj['week\_start\_date'], df\_sj["total\_cases"], label='San Juan')

plt.plot(df\_iq['week\_start\_date'], df\_iq["total\_cases"], label='Iquitos')

plt.xticks(df\_sj['week\_start\_date'], rotation=45)

plt.locator\_params(axis='x', nbins=25)

plt.title("Total Cases", size=20)

plt.legend()

plt.show()

df.hist(bins=50, figsize=(20,15))

plt.show()

week = df[df["weekofyear"]==1][["city","year","weekofyear","week\_start\_date"]]

print(len(week))

week = df[df["weekofyear"]==2][["city","year","weekofyear","week\_start\_date"]]

print(len(week))

week = df[df["weekofyear"]==0][["city","year","weekofyear","week\_start\_date"]]

print(len(week))

df.head()

#We create a dataframe with all the numerical data (without city, or week\_start\_date)

df\_num = df.drop(['city', 'week\_start\_date'], axis=1)

features\_test\_num = features\_test.drop(['city', 'week\_start\_date'], axis=1)

features\_test\_cols = features\_test.columns

features\_test

#We fill the NaNs with a KNN:

from sklearn.impute import KNNImputer

imputer = KNNImputer(n\_neighbors=5)

features\_cols\_num = features\_test\_num.columns

features\_test\_num\_treated = imputer.fit\_transform(features\_test\_num)

features\_test\_num\_treated = pd.DataFrame(features\_test\_num\_treated, columns = features\_cols\_num)

df\_num\_treated = imputer.fit\_transform(df\_num)

df\_num\_treated = pd.DataFrame(df\_num\_treated, columns = df\_num.columns)

df\_num\_treated.head()

#We add the non-numeric columns city and week\_start\_date

dffffff = pd.concat([df\_num\_treated, df[['city', 'week\_start\_date']] ], axis=1)

features\_test = pd.concat([features\_test\_num\_treated, features\_test[['city', 'week\_start\_date']] ], axis=1)

#reorder columns as before

df = dffffff[df.columns]

features\_test = features\_test[features\_test\_cols]

cols = list(df)

cols.insert(3, "NDVI")

df["NDVI"] = df[["ndvi\_ne", "ndvi\_nw", "ndvi\_se", "ndvi\_sw"]].mean(axis=1)

df = df[cols]

df = df.drop(["ndvi\_ne", "ndvi\_nw", "ndvi\_se", "ndvi\_sw"], axis=1)

df.head()

cols = list(features\_test)

cols.insert(3, "NDVI")

features\_test["NDVI"] = features\_test[["ndvi\_ne", "ndvi\_nw", "ndvi\_se", "ndvi\_sw"]].mean(axis=1)

features\_test = features\_test[cols]

features\_test = features\_test.drop(["ndvi\_ne", "ndvi\_nw", "ndvi\_se", "ndvi\_sw"], axis=1)

features\_test.head()

corrMatrix = df.corr()

fig = plt.figure(figsize=(12,10))

#Mask for the uppertriangle

mask = np.zeros\_like(corrMatrix, dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = True

sns.heatmap(corrMatrix, mask = mask, cmap="RdBu\_r", square= True, linewidths=0.5,cbar\_kws={"shrink": 0.5}, center = 0)

plt.title("Correlation Matrix")

plt.show()

df = df.drop(['reanalysis\_dew\_point\_temp\_k'], axis=1)

replace\_dict = {'city': {'iq': 0, 'sj': 1}}

features\_test.replace(replace\_dict, inplace=True)

df.replace(replace\_dict, inplace=True)

df.head()

df.head()

years\_sj = df[df['city']==1.0]['year'].astype('category').cat.categories.tolist()

years\_iq = df[df['city']==0.0]['year'].astype('category').cat.categories.tolist()

n\_iq = 7 #7 out of 12 first years

n\_sj = 14 # first 14 out of 19 years

#Train

train\_iq = df[(df['city']==0.0) & ( df['year'].isin(years\_iq[:n\_iq]))]

train\_sj = df[(df['city']==1.0) & ( df['year'].isin(years\_sj[:n\_sj]))]

X\_train = pd.concat([train\_iq, train\_sj]).drop(['total\_cases'], axis=1)

y\_train = pd.concat([train\_iq, train\_sj])[['total\_cases']]

#Test

test\_iq = df[(df['city']==0.0) & ( df['year'].isin(years\_iq[n\_iq:]))]

test\_sj = df[(df['city']==1.0) & ( df['year'].isin(years\_sj[n\_sj:]))]

X\_test = pd.concat([test\_iq, test\_sj]).drop(['total\_cases'], axis=1)

y\_test = pd.concat([test\_iq, test\_sj])[['total\_cases']]

X\_test\_copy = X\_test.copy() #We keep the week\_start\_date for the visualization of the results

X\_train = X\_train.drop(['week\_start\_date'], axis=1)

X\_test = X\_test.drop(['week\_start\_date'], axis=1)

X\_train = X\_train.drop(['year'], axis=1)

X\_test = X\_test.drop(['year'], axis=1)

X\_test

X\_train

y\_test

xgb = xgboost.XGBRegressor(n\_estimator =120,max\_depth=215,verbosity=1,learning\_rate=1,booster='dart',importance\_type='cover')

xgb.fit(X\_train, y\_train)

y\_pred = xgb.predict(X\_test)

y\_pred = np.rint(y\_pred)

score = mean\_absolute\_error(y\_test, y\_pred)

score

importances = xgb.feature\_importances\_

# Sort feature importances in descending order

indices = np.argsort(importances)[::-1]

# Rearrange features names so they match the sorted feature importances

names = [list(X\_train)[i] for i in indices]

# Create plot

plt.figure()

plt.title("Xtreme Gradient Boosting")

plt.barh(range(X\_train.shape[1]), importances[indices], color="r", align="center")

plt.yticks(range(X\_train.shape[1]), names)

# Show plot

plt.show()

y\_pred = pd.DataFrame(y\_pred, index=X\_test.index, columns=["pred\_cases"])

y\_pred = pd.concat([X\_test\_copy[['city', 'week\_start\_date']], y\_pred], axis=1)

y\_test = pd.concat([X\_test\_copy[['city', 'week\_start\_date']], y\_test], axis=1)

y\_test\_iq = y\_test[y\_test["city"]==0]

y\_pred\_iq = y\_pred[y\_pred["city"]==0]

y\_test\_sj = y\_test[y\_test["city"]==1]

y\_pred\_sj = y\_pred[y\_pred["city"]==1]

plt.figure(figsize=(20, 5))

plt.plot(y\_pred\_iq['week\_start\_date'], y\_pred\_iq["pred\_cases"], label='Pred')

plt.plot(y\_test\_iq['week\_start\_date'], y\_test\_iq["total\_cases"], label='True')

plt.xticks(y\_pred\_iq['week\_start\_date'], rotation=45)

plt.locator\_params(axis='x', nbins=25)

plt.title("Iquitos", size=20)

plt.legend()

plt.show()

plt.figure(figsize=(20, 5))

plt.plot(y\_pred\_sj['week\_start\_date'], y\_pred\_sj["pred\_cases"], label='Pred')

plt.plot(y\_test\_sj['week\_start\_date'], y\_test\_sj["total\_cases"], label='True')

plt.xticks(y\_pred\_sj['week\_start\_date'], rotation=45)

plt.locator\_params(axis='x', nbins=25)

plt.title("San Juan", size=20)

plt.legend()

plt.show()

features\_test\_iq = features\_test[features\_test['city']==0]

features\_test\_sj = features\_test[features\_test['city']==1]

#Prediction to send

submission = xgb.predict(features\_test.drop(['year', 'reanalysis\_dew\_point\_temp\_k', "week\_start\_date"], axis=1))

#Proper formatting

submission = pd.DataFrame(submission, columns=['total\_cases'])

submission = pd.concat([features\_test[['city', 'year', 'weekofyear']], submission], axis=1)

#Cast values to int:

submission['total\_cases'] = np.rint(submission['total\_cases'])#We round up, because the int is an integral part...

submission = submission.astype(int)

#Dictionary that associates a number with each value

replace\_map\_comp = {'city': {0.0: 'iq', 1.0: 'sj'}}

submission.replace(replace\_map\_comp, inplace=True)

submission.head(10)

submission.to\_csv('XGB200\_mae\_knn5\_noyear\_density.csv', index=False)

**8. SYSTEM TESTING**

**8.1 Testing Strategies:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 Types of Testing**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**9 EXPERIMENTAL RESULTS**

**9.1 Implementation Description:**

Building a Dengue fever prediction model using the XGBoost algorithm

* Importing Libraries:

The necessary libraries are imported, including pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, warnings for handling warnings, statsmodels for statistical modeling, sklearn for machine learning tools, and xgboost for the XGBoost algorithm.

* Loading Data:

The code loads the training and test datasets for Dengue fever prediction. The training data includes features (features\_train) and labels (labels\_train), while the test data is stored in features\_test.

* Data Merging:

The features and labels are merged based on the city, year, and week of the year.

* Data Exploration:

This section explores the data. It includes displaying information about the dataset, descriptive statistics, and generating visualizations to understand the patterns in total Dengue cases over time.

* Data Preprocessing:

The code handles missing values using a K-Nearest Neighbors (KNN) imputer. This technique fills in missing values based on the values of their nearest neighbors.

* Feature Engineering:

The code calculates a new feature NDVI (Normalized Difference Vegetation Index) by averaging existing features related to vegetation.

* Correlation Analysis:

A correlation matrix is visualized to understand the relationships between different features.

* Data Transformation:

The data is transformed by encoding the 'city' variable and dropping a feature (reanalysis\_dew\_point\_temp\_k) that is not deemed useful.

* Train-Test Split:

The data is split into training and testing sets for model training and evaluation. The split is done based on the city and year.

* Model Training:

An XGBoost regressor model is instantiated with specified hyperparameters. The model is then trained on the training data.

* Model Prediction:

The trained model is used to make predictions on the test data.

* Model Evaluation:

The Mean Absolute Error (MAE) is calculated to evaluate the performance of the model.

* Feature Importance:

The code computes and visualizes feature importances, which helps identify which features have the most impact on the predictions.

* Result Visualization:

The code creates plots to compare predicted and actual cases for both cities (Iquitos and San Juan).

* Final Prediction for Submission:

The model is used to make predictions on the provided test data (features\_test). The results are formatted in the required submission format.

* Saving the Submission:

The final predictions are saved in a CSV file for submission.

Please note that this code assumes that the necessary data files are present in the specified paths, and it's designed for a specific competition or task related to Dengue fever prediction.

**9.2 Screenshots:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Figure 9.2.1:** Sample dataset used for Dengue spread prediction

Figure 9.2.1 represents a portion of the dataset that was used for predicting the spread of Dengue fever. It includes various features (columns) and corresponding target values (total cases) for a specific period.

A screenshot of a computer program

Description automatically generated

**Figure 9.2.2:** Summary dataset used for Dengue spread prediction

Figure 9.2.2 provides an overview or summary statistics of the dataset used for Dengue spread prediction. It includes information such as mean, median, minimum, maximum, and quartiles for numerical columns, as well as counts for categorical variables.

A graph of cases with blue and orange lines

Description automatically generated

**Figure 9.2.3:** Line plot to visualize the total number of Dengue fever cases over time for two different cities

Figure 9.2.3 is a line plot that displays the total number of Dengue fever cases over time for two different cities: San Juan and Iquitos. The x-axis represents time (likely in weeks or months), while the y-axis represents the total number of cases. There are two lines, one for each city, showing how the cases change over time.

A collage of graphs

Description automatically generated

**Figure 9.2.4:**Histogram for each numerical column in Data Frame

Figure 9.2.4 consists of multiple histograms, each representing the distribution of values for a numerical column in the DataFrame. It provides insights into the frequency of different values within each column.

A close-up of a diagram

Description automatically generated

**Figure 9.2.5:** Heatmap of correlation of columns in a dataset used for dengue spread prediction

Figure 9.2.5 is a heatmap that visualizes the correlation between different columns (features) in the dataset used for predicting Dengue spread. Each cell in the heatmap represents the correlation coefficient between two columns. A warmer color (closer to red) indicates a stronger positive correlation, while a cooler color (closer to blue) indicates a stronger negative correlation.

A screenshot of a computer screen

Description automatically generated

**Figure 9.2.6:** Data frame after preprocessing used for dengue spread

The above figure displays a portion of the dataset after it has undergone preprocessing steps. Preprocessing may include tasks like handling missing values, feature engineering, and encoding categorical variables. It represents the cleaned and transformed data ready for modelling.

A screenshot of a computer

Description automatically generated

**Figure 9.2.7**: Data frame of features column of a dataset after preprocessing

The above figure specifically focuses on the features column(s) of the dataset after preprocessing. It may display the values, statistics, or distribution of the features that will be used for predicting Dengue spread.

A screenshot of a cell phone

Description automatically generated

**Figure 9.2.8:** Target column of a dataset after preprocessing

This figure provides a view of the target column (total cases) of the dataset after preprocessing. It may show the distribution of target values and any transformations or adjustments made during preprocessing.

A graph with orange and blue lines

Description automatically generated

**Figure 9.2.9:** line plot to compare the predicted and actual cases of Dengue fever for the city of Iquitos (iq).

This is a line plot that compares the predicted cases (likely generated by a machine learning model) with the actual cases of Dengue fever for the city of Iquitos. The x-axis represents time, while the y-axis represents the number of cases. The plot helps assess how well the model's predictions align with the actual data.

A graph of a number of people

Description automatically generated with medium confidence

**Figure 9.2.10:** Line plot to compare the predicted and actual cases of Dengue fever for the city of San Juan (S j).

Similar to Figure 9.2.9, this figure compares the predicted cases with the actual cases of Dengue fever, but for the city of San Juan.

A table with numbers and letters

Description automatically generated

**Figure 9.2.11:** Prediction results using XG Boost classifier

This figure displays the overall prediction results obtained using an XGBoost classifier. It may include metrics such as Mean Absolute Error (MAE) or other evaluation measures to assess the performance of the model. All These figures collectively provide a comprehensive view of the data, its preprocessing, and the results of the Dengue spread prediction model

**10. FUTURE ENHANCEMENTS**

The future scope of this research holds several exciting possibilities. Firstly, the incorporation of more advanced machine learning algorithms and deep learning models could enhance the accuracy of Bitcoin price predictions. Additionally, incorporating sentiment analysis of news, social media, and market sentiment could provide valuable insights into market dynamics and help improve forecasting accuracy. Furthermore, expanding the analysis to consider other cryptocurrencies and their interrelationships with Bitcoin could provide a more comprehensive view of the cryptocurrency market.

Another avenue for future exploration is the development of real-time prediction models that adapt to changing market conditions, as cryptocurrency markets are highly influenced by breaking news and events. Moreover, the integration of blockchain analytics and on-chain data into the prediction process could offer unique insights into Bitcoin's price movements.

Lastly, this research can extend its focus to explore the broader economic and regulatory implications of Bitcoin price predictions. As Bitcoin continues to gain prominence in global financial markets, accurate forecasting becomes not only a financial asset but also a tool for policy-makers and investors. In summary, the future scope of this study lies in the refinement of prediction models, the incorporation of additional data sources, and the exploration of the broader implications of Bitcoin price forecasts in the context of the global economy and financial markets.

**11. CONCLUSION**

In conclusion, this study embarked on the challenging task of predicting the price of Bitcoin, a highly volatile and rapidly evolving cryptocurrency, by employing time series analysis with an ARIMA model. Traditional financial prediction methods often fall short when applied to Bitcoin due to its unique characteristics and lack of seasonality. However, through the application of machine learning techniques, we have explored patterns and trends within Bitcoin's price data, striving to gain insights into its future price movements. While the results may not always provide perfect predictions, they underscore the potential of machine learning models in understanding and forecasting cryptocurrency markets.

**12. REFERENCES**

1. Majeed, M.A.; Shafri, H.Z.M.; Zulkafli, Z.; Wayayok, A. A Deep Learning Approach for Dengue Fever Prediction in Malaysia Using LSTM with Spatial Attention. Int. J. Environ. Res. Public Health 2023, 20, 4130. <https://doi.org/10.3390/ijerph20054130>
2. Cabrera, M.; Leake, J.; Naranjo-Torres, J.; Valero, N.; Cabrera, J.C.; Rodríguez-Morales, A.J. Dengue Prediction in Latin America Using Machine Learning and the One Health Perspective: A Literature Review. Trop. Med. Infect. Dis. 2022, 7, 322. <https://doi.org/10.3390/tropicalmed7100322>
3. Dey, Samrat Kumar, et al. "Prediction of dengue incidents using hospitalized patients, metrological and socio-economic data in Bangladesh: A machine learning approach." PLoS One 17.7 (2022): e0270933.
4. Kakarla, S.G., Kondeti, P.K., Vavilala, H.P. et al. Weather integrated multiple machine learning models for prediction of dengue prevalence in India. Int J Biometeorol 67, 285–297 (2023). <https://doi.org/10.1007/s00484-022-02405-z>
5. Roster, Kirstin, Colm Connaughton, and Francisco A. Rodrigues. "Machine-Learning–Based Forecasting of Dengue Fever in Brazilian Cities Using Epidemiologic and Meteorological Variables." American Journal of Epidemiology 191.10 (2022): 1803-1812.
6. Sarder, Faysal, Sorefa Akter, and Sharmin Akter. "Predicting Dengue Outbreak from Climate Data Using Machine Learning Algorithms." 2022 IEEE International Conference on Data Science and Information System (ICDSIS). IEEE, 2022.
7. Ochida, N., Mangeas, M., Dupont-Rouzeyrol, M. et al. Modeling present and future climate risk of dengue outbreak, a case study in New Caledonia. Environ Health 21, 20 (2022). <https://doi.org/10.1186/s12940-022-00829-z>.
8. Anuranjan, M. B., et al. "Machine Learning Techniques for Predicting Dengue Outbreak." Innovations in Information and Communication Technologies: Proceedings of ICIICT 2022. Singapore: Springer Nature Singapore, 2022. 45-56.
9. Gupta, G.; Khan, S.; Guleria, V.; Almjally, A.; Alabduallah, B.I.; Siddiqui, T.; Albahlal, B.M.; Alajlan, S.A.; AL-subaie, M. DDPM: A Dengue Disease Prediction and Diagnosis Model Using Sentiment Analysis and Machine Learning Algorithms. Diagnostics 2023, 13, 1093. <https://doi.org/10.3390/diagnostics13061093>.
10. Rocha, F.P., Giesbrecht, M. Machine learning algorithms for dengue risk assessment: a case study for São Luís do Maranhão. Comp. Appl. Math. 41, 393 (2022).
11. Zheng, Cong-Han, et al. "Predicting Infection Area of Dengue Fever for Next Week Through Multiple Factors." Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings. Cham: Springer International Publishing, 2022.
12. Pacheco, Paolo Ramon DC. "Application of interpretable machine learning in revealing spatial and temporal patterns of dengue disease using climatic factors." (2022).