VIRUS OUTBREAK ANALYSIS

A Summer internship Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING-DATA SCIENCE

Submitted by

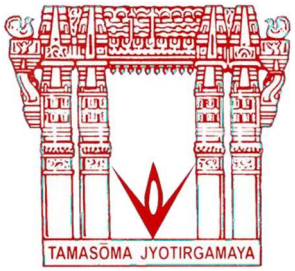
Mr.B.Bhanuprasad (21071A6775)

Mr.Ch.Sadvik Reddy (21071A6782)

Mr.P.Raghavendra (21071A67B1)

Under the guidance of

Mrs.G.Ashalatha
(Assistant Professor, Department of CSE-(CYS, DS) and AI&DS)



DEPARTMENT OF CSE-(CYS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS

CERTIFICATE

This is to certify that the project report entitled **"Virus Outbreak Analysis"** is a bonafide work done under our supervision and is being submitted by **Mr. B. Bhanuprasad(21071A6775), Mr. Ch Sadvik Reddy(21071A6782), Mr. P Raghavendra(21071A67B1)** in partial fulfillment for the award of the degree of Bachelor of Technology in CSE-CYS, DS and AI&DS , of the VNRVJIET, Hyderabad during the academic year 2023-2024. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Mrs.G. Ashalatha
Assistant Professor
Dept. of CSE-(CYS, DS) and AI&DS
VNR VJIET

Dr.M.Raja Sekar
Professor and Head
Dept. of CSE-(CYS, DS) and AI&DS
VNR VJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS

DECLARATION

We declare that the major project work entitled "**Virus Outbreak Analysis**" submitted in the department of **CSE-(CYS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **CSE-(CYS, DS) and AI&DS**is a bonafide record of our own work carried out under the supervision of **Mrs.G. Ashalatha**, Assistant **Professor, Department of CSE-(CYS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad

B.Bhanuprasad	**Ch. Sadvik Reddy**	**P. Raghavendra**
(21071A6775)	(21071A6782)	(21071A67B1)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. M.Raja Sekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, **Mrs.G. Ashalatha,** and the project coordinators **Dr.V. Prashanthi and Mrs.B.Deepika** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering department and to our fellow classmates who directly or indirectly helped us.

MrB.Bhanuprasad (21071A6775)

Mr Ch. Sadvik Reddy (21071A6782)

MrP. Raghavendra (21071A67B1)

ABSTRACT

When a newly emerging infectious disease breaks out in a country, it brings critical damage to both human health conditions and the national economy. For this reason, apprehending which disease will emerge, and preparing countermeasures for that disease, are required. Many different types of infectious diseases are emerging and threatening global human health conditions. For this reason, the detection of emerging infectious disease patterns is critical. However the existing models deal with the mentioned problems. So we tried to take a different approach by using the available datasets.

The "Virus Outbreak Analysis" project focuses on examining past data to predict the number of COVID-19 cases in the next 30 days. By utilizing historical information, the project aims to forecast the potential spread of the virus. The analysis involves the application of data-driven approaches to understand patterns and trends, enabling the projection of future infection rates. The project's goal is to provide valuable insights to help anticipate and plan for the potential impact of COVID-19 virus, contributing to informed decision-making in public health and resource allocation.

By leveraging historical outbreak data, demographic information, and healthcare infrastructure data, we aim to identify patterns and factors that contribute to the spread and impact of viral outbreaks. Through the application of predictive modelling techniques, we seek to identify regions at higher risk and provide insights that can aid in resource allocation, preparedness planning, and timely intervention strategies. The outcomes of this project can assist public health authorities and policymakers in making informed decisions to mitigate the impact of future virus outbreaks and protect vulnerable populations.

INDEX

List of Tables

1. INTRODUCTION

Public health is a critical facet of societal well-being, and amid the ongoing global challenges posed by infectious diseases, forecasting and managing the trajectory of virus outbreaks have become paramount. The 'Virus Outbreak Analysis' project aims to harness the power of historical data to predict the number of COVID-19 cases in the next 30 days. In the face of this unprecedented pandemic, understanding past trends and utilizing data-driven methodologies becomes essential for providing valuable insights. By examining patterns and leveraging predictive modeling, this project seeks to empower decision-makers in public health, aiding in proactive planning, resource allocation, and effective response strategies to mitigate the impact of the ongoing virus outbreak.

In navigating this complex and ever-evolving landscape, the project seeks to transcend the conventional boundaries of infectious disease management. By harnessing the power of predictive modeling and advanced analytics, it aims to empower decision-makers in the realm of public health. The goal extends beyond mere prediction; it aspires to be a proactive tool, offering decision-makers the foresight needed for strategic planning, resource allocation, and the formulation of effective response strategies.

Amid the intricate dance between infectious diseases and societal well-being, the "Virus Outbreak Analysis" project stands as a testament to the resilience of scientific inquiry and its application in real-world scenarios. Through its meticulous examination of historical data, it not only endeavors to predict the future trajectory of COVID-19 cases but also aims to contribute significantly to the adaptive and proactive management of public health resources. In essence, this project emerges as a linchpin in the ongoing battle against the virus outbreak, bridging the gap between past experiences and a future that is shaped by informed, strategic, and data-driven decisions in the realm of public health.

2.LITERATURE SURVEY/EXISTING SYSTEM

2.1 FEASIBILITY STUDY

In the dynamic landscape of public health challenges, the 'Virus Outbreak Analysis' project emerges as a timely initiative, harnessing advancements in data analytics and machine learning. With extensive historical data accessibility and ongoing analytical progress, predicting COVID-19 trajectory becomes increasingly feasible. This project aims to leverage technological strides, offering valuable insights into virus spread for informed decision-making. Feasibility is supported by growing data availability, sophisticated machine learning algorithms, and the imperative for proactive planning amid global health crises.

2.1.1 ORGANIZATION FEASIBILITY

The "Virus Outbreak Analysis" project aligns seamlessly with organizational feasibility objectives. Designed to cater to the unique needs of public health entities and decision-makers, this initiative offers a comprehensive tool for analyzing and predicting the spread of COVID-19. Its utility spans across various organizations involved in healthcare, government, and emergency response. By providing timely insights into potential virus outbreaks, the project aids organizations in proactive planning, resource allocation, and effective response strategies. The simplicity and accessibility of the analysis make it a valuable asset for organizations grappling with the challenges posed by the ongoing global health crisis.

2.1.2 ECONOMIC FEASIBILITY

The "Virus Outbreak Analysis" project showcases strong economic feasibility. Initially, the project is available at no cost, ensuring financial accessibility for users. Furthermore, the pre-trained model eliminates the necessity for constant internet connectivity, adding to cost-effectiveness.

Regarding software, the project utilizes economically feasible tools. Python and Jupyter Notebook are not only user-friendly and accessible but also available for free, minimizing potential costs associated with software licensing. This economic viability enhances the project's accessibility and sustainability, making it a cost-effective solution for analyzing and predicting virus outbreaks.

2.1.3 TECHNICAL FEASIBILITY

The technical feasibility of the "Virus Outbreak Analysis" project relies on the installation of essential tools like Visual Studio Code and the Python programming language. With a minimum of 8GB RAM, the system can smoothly run the required software. Necessary modules and packages must be installed for project development and execution. Internet connectivity is needed for package downloads; however, once the packages are acquired, the project can function independently. The user-friendly nature of this project makes it technically feasible, ensuring ease of use for individuals without extensive technical expertise.

2.1.4 BEHAVIORAL FEASIBILITY

The project is behaviorally feasible as it does not require advanced technical knowledge for operation. Users interact with the system in a straightforward manner, and the absence of complex technical instructions enhances user-friendliness. The behavioral feasibility is grounded in the simplicity of project usage, making it accessible to a broad audience without the need for technical guidance.

2.2 LITERATURE REVIEW

Infectious disease outbreak prediction using media articles with machine learning models [1]:

The experiment is based on the expected accuracy of whether the diseases that had not been reported for (6 or 3) months will break out or not by country. Tables show a comparison of the results with SVM, SSL, and DNN in terms of the accuracy, ROC, and F1 score. For each of the three models, the best performance was selected by searching over the respective model-parameter space.

Early detection and prediction of infectious disease outbreaks
Volume 45-5, May 2, 2019: Climate change and infectious diseases:The solutions [2]:

There is a growing trend towards automation because of the ability to process high volumes of data, and the accuracy of ML and NLP methods to identify events are improving and may one day surpass the ability of human moderators. Risk modelling to understand and predict the health impacts of infectious diseases is commonly performed using statistical inference and compartmental modelling approaches.

COVID-19 Outbreak Prediction with Machine Learning [3]:

This paper presents a comparative analysis of ML and soft computing models to predict the COVID-19 outbreak. The results of two ML models (MLP and ANFIS) reported a high generalization ability for long-term prediction. With respect to the results reported in this paper and due to the highly complex nature of the COVID-19 outbreak and differences among nations, this study suggests ML as an effective tool to model the time series of the outbreak. We should note that this paper provides an initial benchmarking to demonstrate the potential of machine learning for future research.

SEIR and Regression Model based COVID-19 outbreak predictions in India [4]:

In this study, outbreak of this disease has been analysed for India till 30th March 2020 and predictions have been made for the number of cases for the next 2 weeks. SEIR model and Regression model have been used for predictions based on the data collected from John Hopkins University repository in the time period of 30th January 2020 to 30th March 2020. The performance of the models was evaluated using RMSLE and achieved 1.52 for SEIR model and 1.75 for the regression model. The RMSLE error rate between SEIR model and Regression model was found to be 2.01. Also, the value of R0 which is the spread of the disease was calculated to be 2.02.

2.3 EXISTING SYSTEM

- The analysis of virus outbreak data is predominantly conducted manually by designated personnel.

- Currently, the system triggers an investigation only in response to significant virus outbreaks or notable events.

2.4 DRAWBACKS OF THE EXISTING SYSTEM

- Analyzing virus outbreak data manually is a time-consuming process due to the vast amount of data records that need scrutiny.

- The investigation process heavily relies on professional staff, potentially affecting the overall efficiency and resources of the organization.

- In cases where the number of complaints is high, there is a risk of neglecting smaller outbreaks,which could lead to potential losses for the organization.

3. SOFTWARE REQUIRMENT ANALYSIS

3.1 INTRODUCTION

Following elicitation, requirement analysis is an important and necessary process. To create uniform and unambiguous requirements, we examine, improve, and scrutinize the obtained requirements. This exercise goes over all of the requirements and may show a graphical representation of the full system.

3.1.1 DOCUMENT PURPOSE

This document serves as a guiding framework for software developers, outlining the specific requirements and functionalities essential for the creation of an effective "Virus Outbreak Analysis" system tailored to public health and epidemiological research needs.

3.1.2 DEFINITIONS

Virus Data Analysis:
Virus data analysis involves the examination of epidemiological data related to virus outbreaks to identify patterns, trends, and correlations that can inform public health strategies and interventions.

Data Cleansing:
Data cleansing is the process of detecting and correcting errors or inconsistencies in virus outbreak data to improve its quality and reliability for analysis and decision-making purposes.

One Hot Encoding:
One-hot encoding is used in machine learning as a method to quantify categorical data. In short, this method produces a vector with length equal to the number of categories in the data set. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. In this case each value becomes a new column.

Data Aggregation:
Data aggregation involves the combination of individual virus outbreak data points into summary statistics or groups to facilitate analysis and interpretation of trends and patterns.

Feature Extraction:
Feature extraction is the process of selecting and transforming relevant variables or features from virus outbreak data to represent key aspects of the outbreak dynamics and characteristics.

Model Training:
Model training involves the use of historical virus outbreak data to train machine learning algorithms to recognize patterns and make predictions about future outbreak trends and dynamics.

Prediction Modeling:
Prediction modeling utilizes trained machine learning algorithms to forecast the spread and impact of virus outbreaks based on historical trends, environmental factors, and demographic variables.

Evaluation Metrics:
Evaluation metrics are quantitative measures used to assess the performance and accuracy of virus outbreak prediction models, such as precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC).

Geospatial Analysis:
Geospatial analysis involves the examination of virus outbreak data in spatial contexts to identify geographic patterns, clusters, and hotspots of infection transmission for targeted intervention and control measures.

Temporal Analysis:
Temporal analysis focuses on the examination of virus outbreak data over time to identify temporal trends, seasonality, and periodic patterns that may influence the dynamics and spread of infections.

Risk Assessment:
Risk assessment involves the systematic evaluation of virus outbreak data to assess the likelihood and potential impact of future outbreaks on public health, socio-economic factors, and healthcare systems.

3.2 SYSTEM ARCHITECTURE

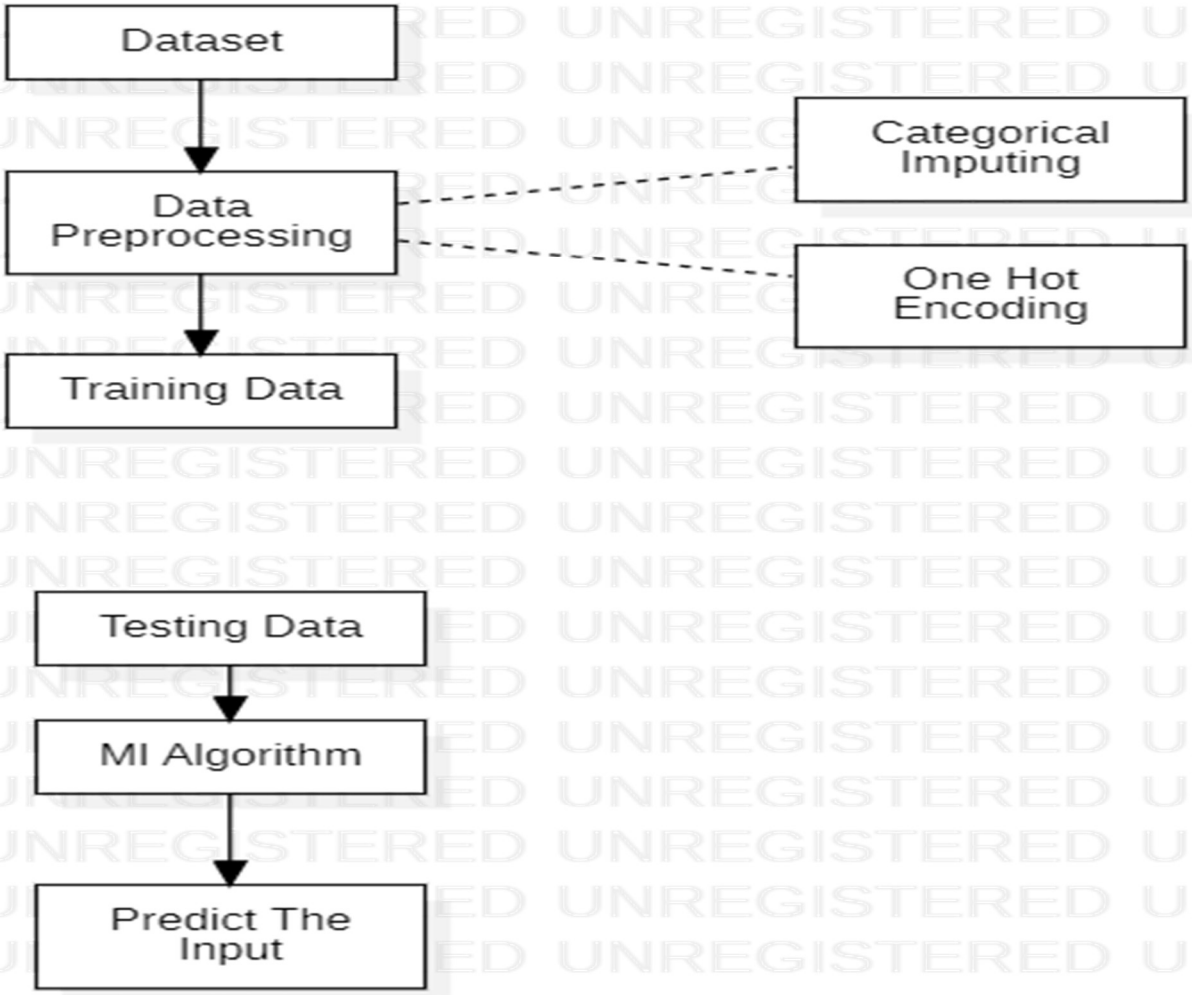


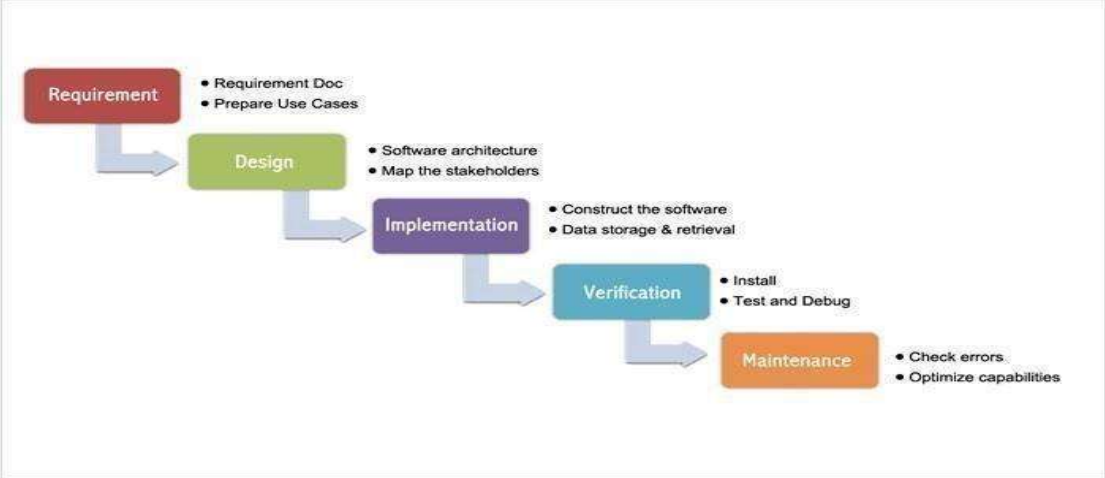
Fig 3.2.1: System Architecture

It is a conceptual model that describes the structure, viewpoints, and behavior of a system. A formal description and representation of a system that is organized in a way that facilitates reasoning about the system's structures and behaviors is known as an architecture description. A system architecture consists of system components and created sub-systems that work together to implement the overall systemThe proposed system for "Virus Outbreak Analysis" integrates data preprocessing, trend detection, and a user-friendly plot area to offer insights into virus dynamics. It involves refining raw data, employing machine learning for trend analysis, and presenting results visually for informed decision-making. The system ensures seamless collaboration between functionalities and prioritizes user-friendly design.

3.3 FUNCTIONAL REQUIREMENTS

- data preprocessing
- detect emerging trends
- The plot area

3.4 SYSTEM ANALYSIS



It was the first Process Model to be introduced. A linear sequential life cycle model is another name for it. It's quite simple to use and understand. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The first SDLC

approach used during software development was the waterfall model.
The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap.
The steps in the waterfall model are explained below.

Requirements: The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented.

Design: This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

Implementation: The design was converted into a machine-readable format in order for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.
Verification: It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply to the previously specified requirements.
Maintenance: Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.
Useful factors: The Waterfall model has its advantages like it is simple to use. Additionally, while using the model all the system requirements can be defined as a whole, explicitly and at the start the product can run without many issues.
It is economic to make changes in the early stages of the project when there are problems with system requirements then when the problems which arise in later stages.

3.5 NON-FUNCTIONAL REQUIREMENTS

Ease of Use

- The system is simple, user friendly.

Extensibility

- The system can be easily extended to incorporate additional functionality.

Security

- The system is secure because it doesn't share any information of the client without his/her consent.

Maintainability:

- The system will be as self-contained as possible to allow for ongoing maintenance.

Reliability:

- Most efficient model is chosen that is fbprophet.

3.6 SOFTWARE REQUIREMENT SPECIFICATION

JUPITER NOTEBOOK

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.
Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use

GOOGLE COLAB

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

3.7 SOFTWARE REQUIREMENTS

- Software : jupiter notebook and google colab
- Operating System : Windows family
- Technology : Machine Learning

3.8 HARDWARE REQUIREMENTS

- Minimum 8GB Ram Laptop and Internet Connection

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

The Device Architecture Manual describes the application requirements, operating state, application and subsystem functionality, documents and repository setup, input locations, yield types, human-machine interfaces, management reasoning, and external interfaces. The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. A UML context is defined as five distinct viewpoints that present the system from a particularly different point of view.

The components are similar to modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

User Model View
The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined in light of the user and what the user wants from it.

Structural model view
This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

Behavioral Model View
It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagrams.

Implementation Model View
The essential and actions as frame pieces are discussed in this when they are to be

manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

Environmental Model View

The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

- Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
- UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

4.1.1 USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams.

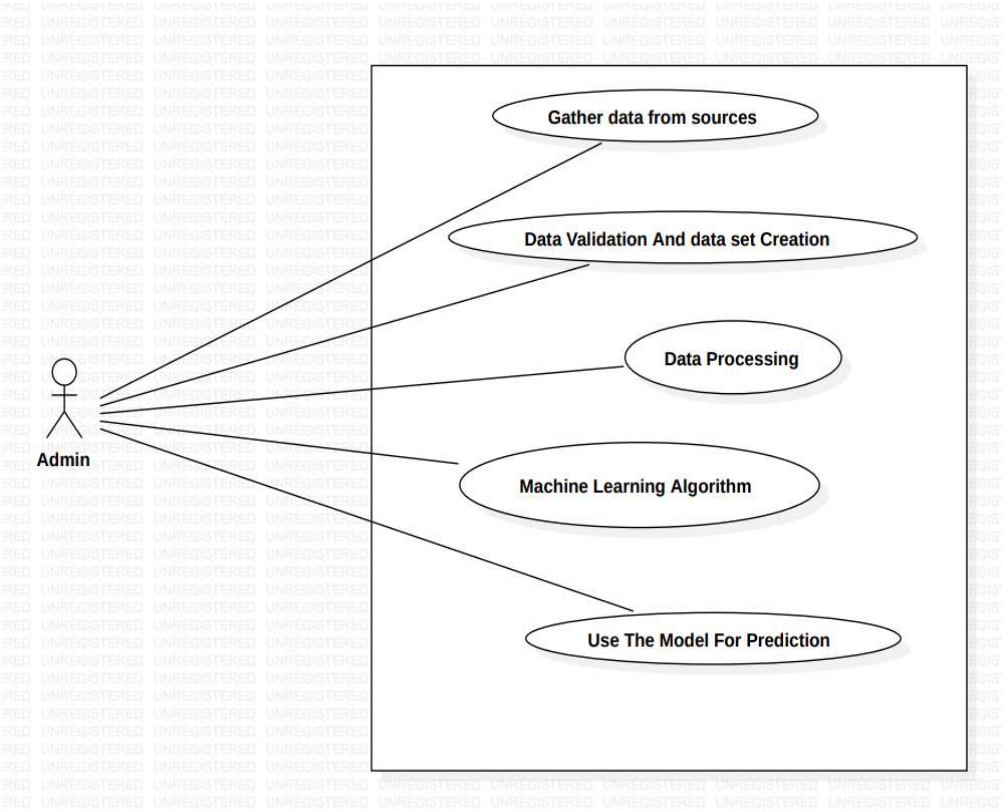


Fig 4.1.1.1: Use Case diagram for the application

Actors:

- Admin

Use Cases:

- Gather Datafromsources
- Data Validation And Data Set Creation
- Data Processing
- Machine Learning Algorithm
- Use The Model For Prediction

Connections:

- Admin must gather data required to choose which algorithm to be used
- Then the admin will be processing the data and and choose the algorithm based on accuracy
- User will be using the efficient model for prediction on the data

4.1.2 SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and business people to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

- Describe the specifics of a UML use case.
- Create a model of the logic of a complex procedure, function, or operation.
- Examine how objects and components interact with one another in order to complete a process.
- Plan and comprehend the specific functionality of a current or future scenario.

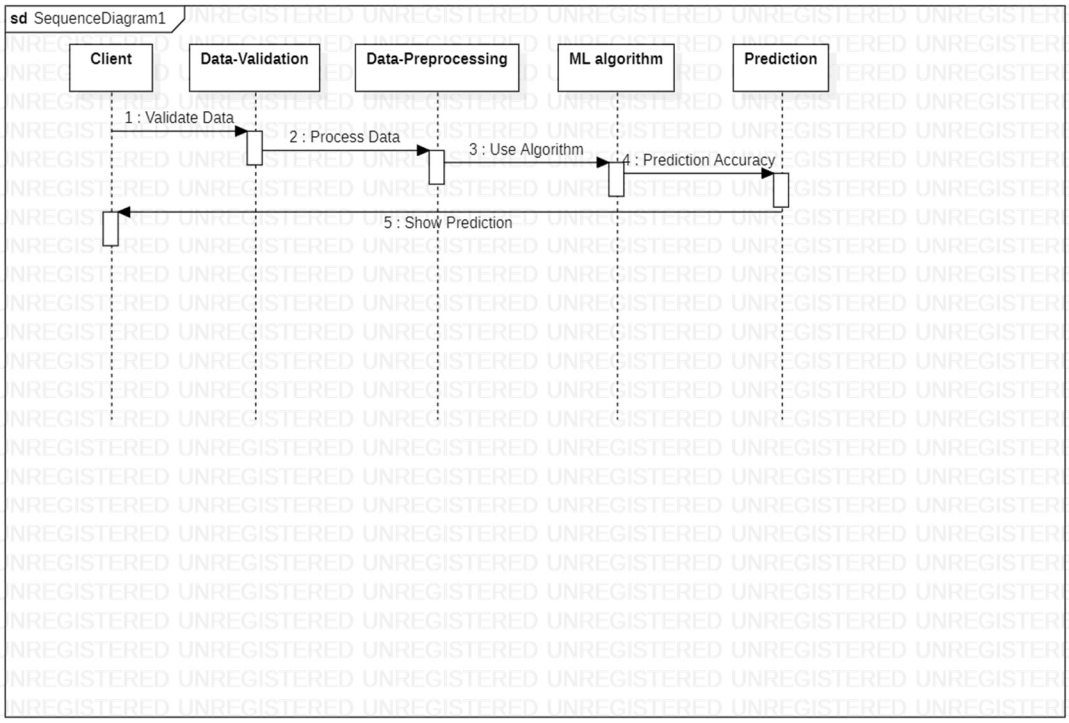


Fig 4.1.2.1: Sequence Diagram for the application

In the above sequence diagram, the lifelines are:

- Client
- Data-Validtion
- Data-Preprocessing
- ML Algorithm
- Prediction

The sequence starts from clientsending the data required for the prediction process that includes data-validation,data-preprocessing.

4.1.3 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity.

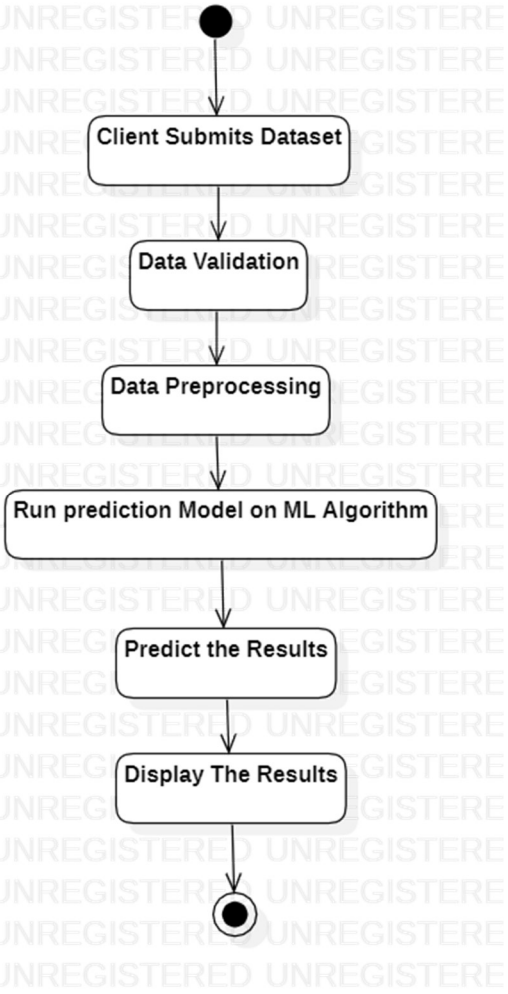


Fig 4.1.3.1: Activity Diagram for the application

This activity diagram shows the whole activity of the system. The Activity starts with client submitting the data required for prediction and data validation,data preprocessing followed by predicting the results.

4.1.4 CLASS DIAGRAM

A static diagram is also referred to as a class diagram. It depicts the static view of an application. A class diagram can be used to visualize, describe, and document various parts of a system, as well as to create executable code for a software programmer.

The traits and activities of a class, as well as the constraints, are described in a class diagram.

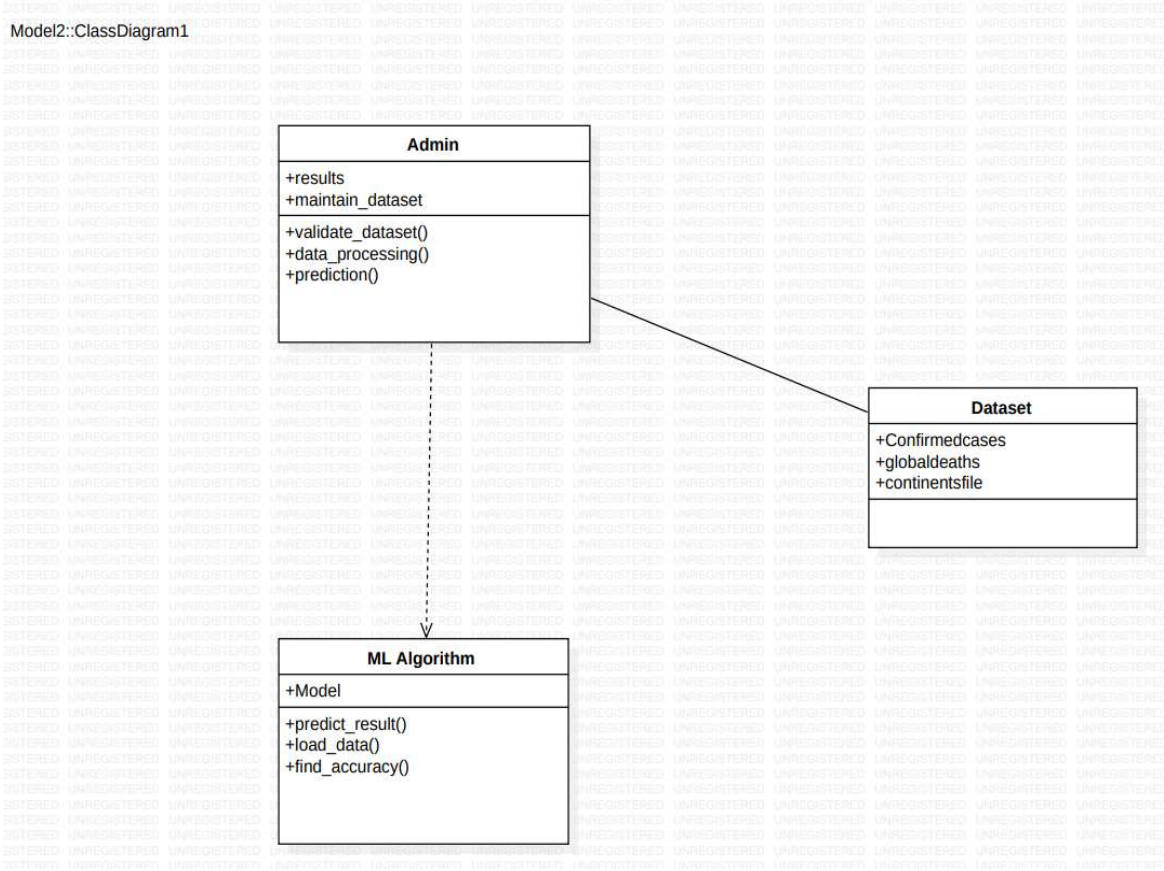
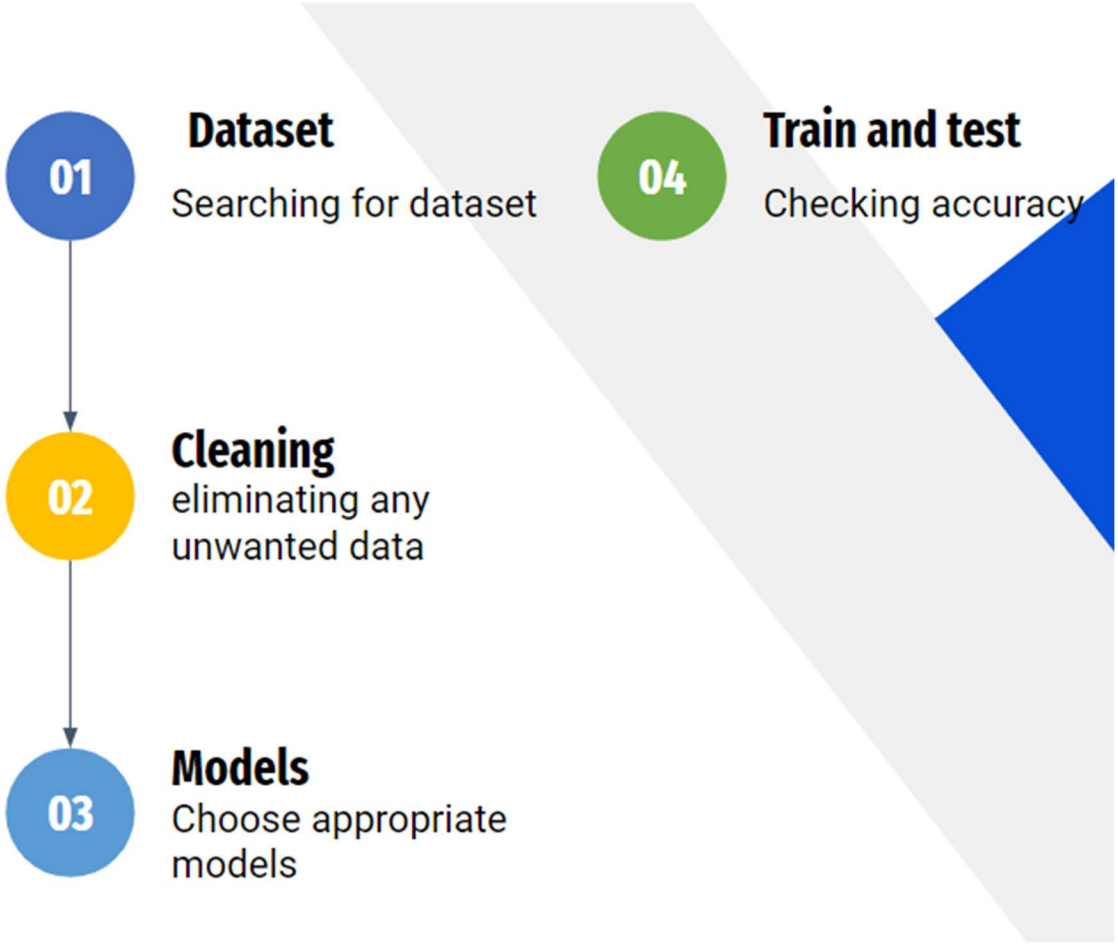


Fig 4.1.4.1: Class Diagram for the application

The main class in this class diagram is the camera that captures and detects the image of the person. Another classes are the train image and track image which saves the saves and recognizes the mask. The track class recognises and the label class labels the result as mask and no mask.

5 PROPOSED SYSTEM

5.1 METHODOLOGY



Data Set:

Data is crucial in finding pattern and use that pattern to predict the outcome in our case finding the fraud cases. For our project we give CSV file to train the model.
Confirmed Dataset consists of 290 columns and 1143rows,GlobaldeathsDataset consists of 290 columns and 1143rows,Continents Dataset consists of 11 columns and 249rows.

Cleaning:

Data cleaning is the process of preparing data for analysis by weeding out information that is irrelevant or incorrect.

This is generally data that can have a negative impact on the model or algorithm it is fed into by reinforcing a wrong notion.

Data cleaning not only refers to removing chunks of unnecessary data, but it's also often associated with fixing incorrect information within the train-validation-test dataset and reducing duplicates.

Here are some key takeaways on the best practices you can employ for data cleaning:

1. Identify and drop duplicates and redundant data
2. Detect and remove inconsistencies in data by validating with known factors
3. Maintain a strict data quality measure while importing new data.
4. Fix typos and fill in missing regions with efficient and accurate algorithms

Models:

A machine learning model is defined as a mathematical representation of the output of th*e training process.* Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

In Our Project we used ARIMA(AutoRegressive Integrated Moving Average) model.
It's a way of modelling time series data for forecasting (i.e., for predicting future points in the series)

Train and Test:

Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

5.2 FUNCTIONALITIES

5.2.1 Importing Data:

- we import CSV file
- file contain data collected from sources

5.2.2 Preprocessing data:

- Removing inconsistencies in data like null values and outliers
- dealing with categorical data
- Modifying the Dataset according to our requirements

5.2.3 Visualizing the Data:

- Getting visual representation of geographical data using choropleth map

5.2.4 Forecast values:

- train the Model and Using the values predict the no of cases in coming 30 days

5.3 ADVANTAGES OF PROPOSED SYSTEM:

- Easy to maintain large datasets

- we can reduce the size of the data set without effecting the data
- Takes less time as compared to traditional methods
- Less human effort

6. CODING AND IMPLEMENTATIONS

6.1 DATA SET

It is used to train the model. The model learns from this data set and gives the outputs based on the learning during the testing.
We have data consisting of 3 files :confirmed cases file, global deaths file, continents file

6.2 Understanding Data

Initially we imported all libraries which are required like pandas ,matplotlib seaborn etc. Data set is in the CSV format. Data consists of both categorical and numerical data .
(i)**confirmed cases file** has 290 columns and 1143rows , columns contain the contries and the rows contain the dates and no.of cases.
(ii)**global deaths** file has 290 columns and 1143rows , columns contain the contries and the rows contain the dates and no.ofdeaths.
(iii)**continents file** has 11 columns and 249rows, contains the geographical data of all the countries.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
```

```
plt.style.use("ggplot")
```

```
from google.colab import files


uploaded = files.upload()
```

Choose Files 2 files

- **CONVENIENT_global_confirmed_cases.csv**(text/csv) - 1635228 bytes, last modified: 12/23/2023 - 100% done
- **CONVENIENT_global_deaths.csv**(text/csv) - 1397176 bytes, last modified: 12/23/2023 - 100% done

```
Saving CONVENIENT_global_confirmed_cases.csv to CONVENIENT_global_confirmed_cases (1).csv
Saving CONVENIENT_global_deaths.csv to CONVENIENT_global_deaths (1).csv
```

```
df0 = pd.read_csv("CONVENIENT_global_confirmed_cases.csv")
df1 = pd.read_csv("CONVENIENT_global_deaths.csv")
```

```
df0.shape
```

```
(1143, 290)
```

+ Code + Text

```
df0.head()
```

	Country/Region	Afghanistan	Albania	Algeria	Andorra	Angola	Antarctica	Antigua and Barbuda	Argentina	Armenia	...	Uruguay	Uzbekistan	Vanuatu	Venezuela	Vietnam	West Bank and Gaza	Winter Olympics 2022	Yemen
0	Province/State	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1/23/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
2	1/24/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1/25/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1/26/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 290 columns

The global deaths dataset and the continents dataset are as follows:

```
df1.shape
```

```
(1143, 290)
```

```
df1.head()
```

	Country/Region	Afghanistan	Albania	Algeria	Andorra	Angola	Antarctica	Antigua and Barbuda	Argentina	Armenia	...	Uruguay	Uzbekistan	Vanuatu	Venezuela	Vietnam	West Bank and Gaza	Winter Olympics 2022	Yemen
0	Province/State	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1/23/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1/24/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1/25/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1/26/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 290 columns

```
from google.colab import files


uploaded = files.upload()
```

Choose Files continents2.csv

- **continents2.csv**(text/csv) - 19700 bytes, last modified: 12/23/2023 - 100% done

```
Saving continents2.csv to continents2 (1).csv
```

```
continent=pd.read_csv("continents2.csv")
continent["name"]=continent["name"].str.upper()
```

```
continent.shape
```

```
(249, 11)
```

```
continent.head()
```

	name	alpha-2	alpha-3	country-code	iso_3166-2	region	sub-region	intermediate-region	region-code	sub-region-code	intermediate-region-code
0	AFGHANISTAN	AF	AFG	4	ISO 3166-2:AF	Asia	Southern Asia	NaN	142.0	34.0	NaN
1	ÅLAND ISLANDS	AX	ALA	248	ISO 3166-2:AX	Europe	Northern Europe	NaN	150.0	154.0	NaN
2	ALBANIA	AL	ALB	8	ISO 3166-2:AL	Europe	Southern Europe	NaN	150.0	39.0	NaN
3	ALGERIA	DZ	DZA	12	ISO 3166-2:DZ	Africa	Northern Africa	NaN	2.0	15.0	NaN
4	AMERICAN SAMOA	AS	ASM	16	ISO 3166-2:AS	Oceania	Polynesia	NaN	9.0	61.0	NaN

6.3 Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Some preprocessing tasks we are going to do are:

1. removing unwanted features
2. replacing a feature with more meaning full feature
3. removing or replacing null(?) values
4. Dealing with Categorical data

```
world = pd.DataFrame({"Country":[],"Cases":[]})
world["Country"] = df0.iloc[:,1:].columns
cases = []
for i in world["Country"]:
    cases.append(pd.to_numeric(df0[i][1:]).sum())
world["Cases"]=cases
```

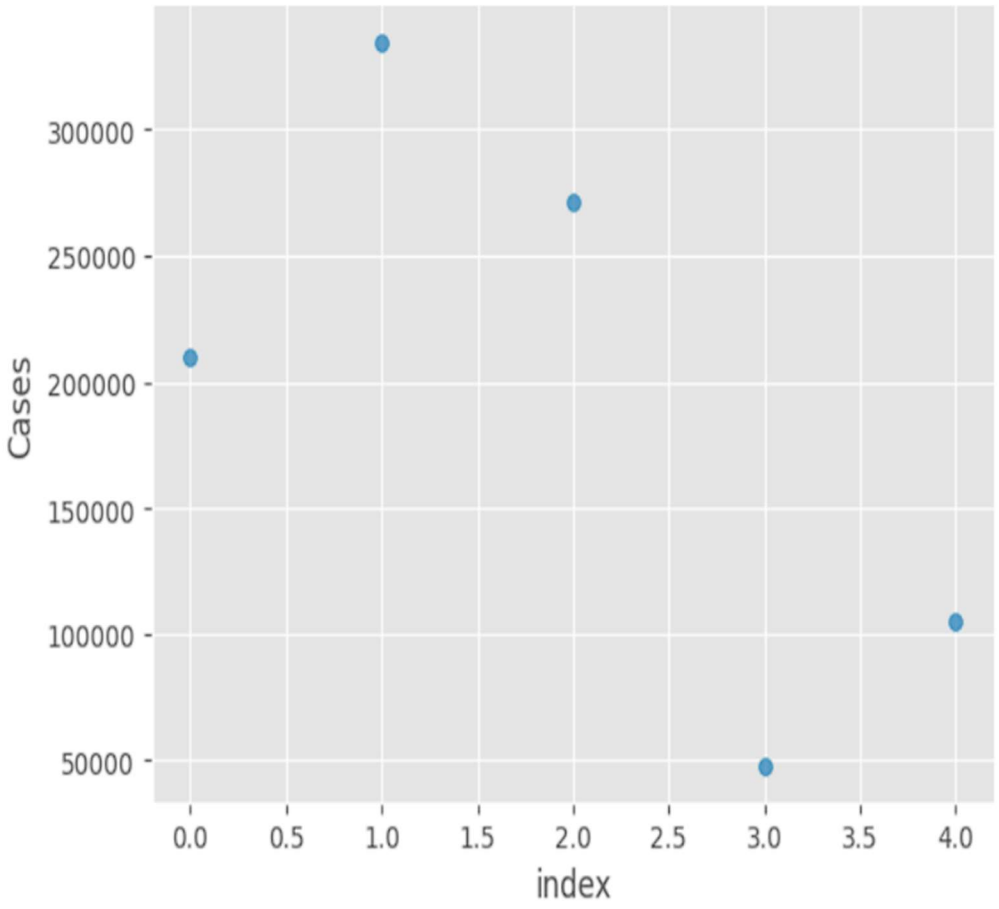
```
country_list=list(world["Country"].values)
idx = 0
for i in country_list:
    sayac = 0
    for j in i:
        if j==".":
            i = i[:sayac]
            country_list[idx]=i
        elif j=="(":
            i = i[:sayac-1]
            country_list[idx]=i
        else:
            sayac += 1
    idx += 1
world["Country"]=country_list
world = world.groupby("Country")["Cases"].sum().reset_index()
world.head()
```

1 to 5 of 5 entries Filter

index	Country	Cases
0	Afghanistan	209451.
1	Albania	334457.
2	Algeria	271496.
3	Andorra	47890.
4	Angola	105288.

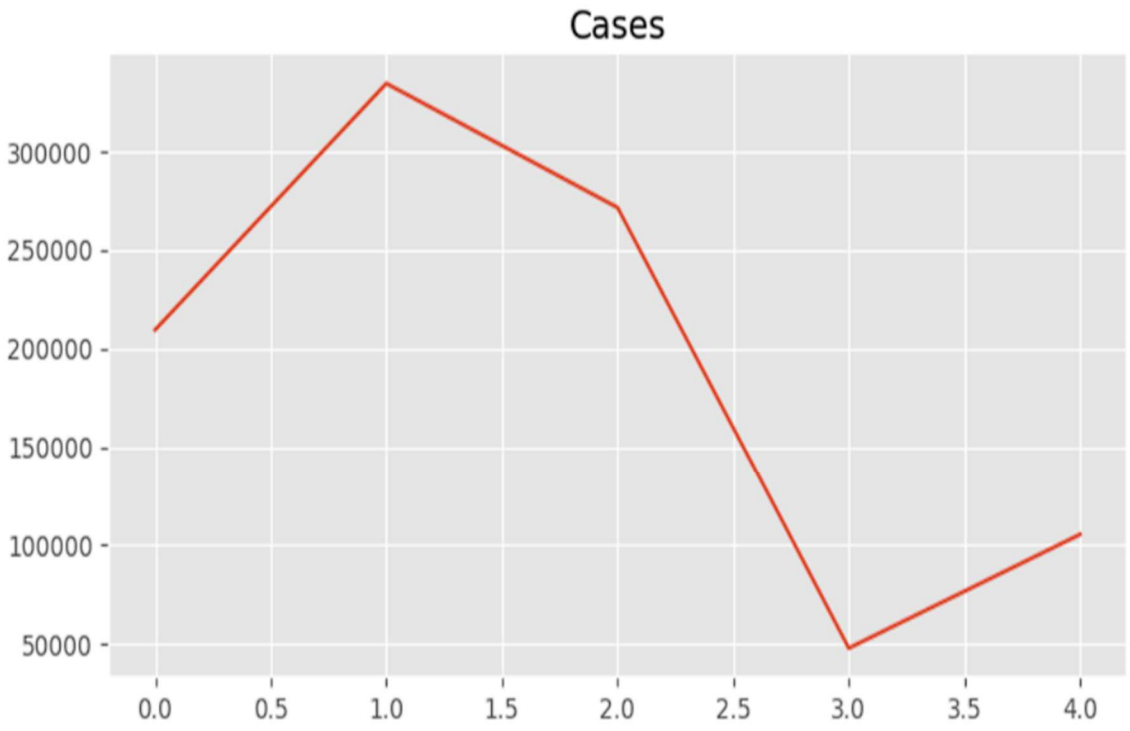
Show 25 per page

```
from matplotlib import pyplot as plt
_df_3.plot(kind='scatter', x='index', y='Cases', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```



The above plot shows the 2d distribution for each country using the index and the number of cases for each country,here each country is given a index according to the alphabetical order.

```
from matplotlib import pyplot as plt
_df_7['Cases'].plot(kind='line', figsize=(8, 4), title='Cases')
plt.gca().spines[['top', 'right']].set_visible(False)
```

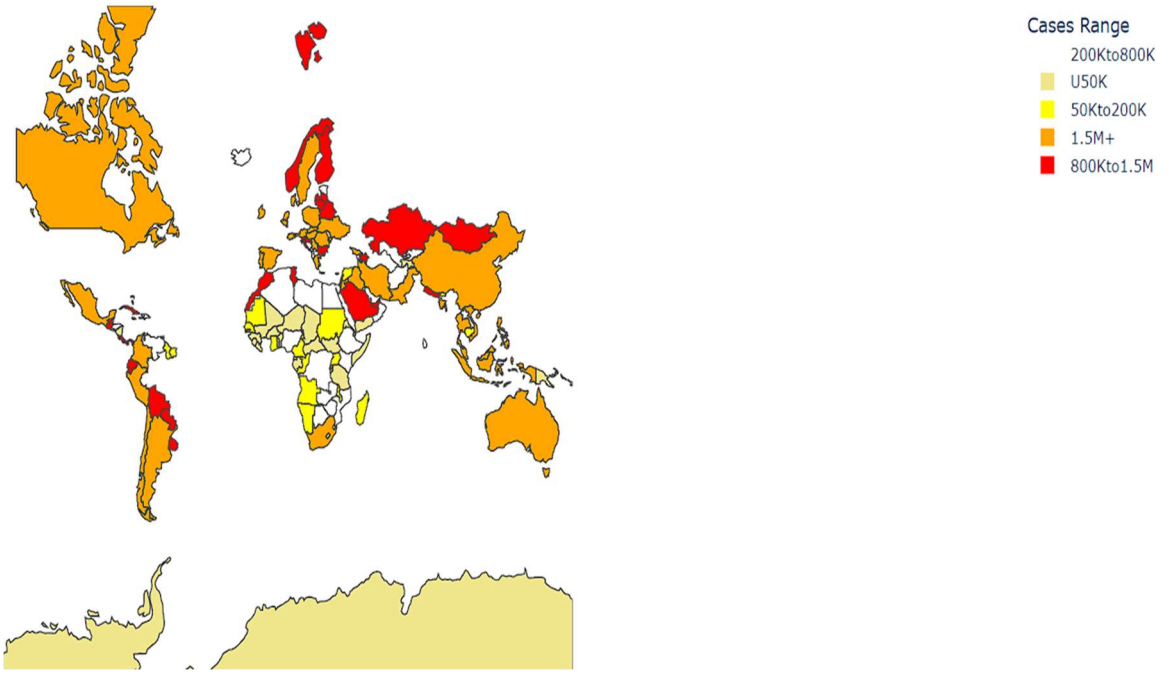


The above plot shows the line plot for each country using the index and the number of cases for each country,here each country is given a index according to the alphabetical order.

6.4 DataVisualisation :

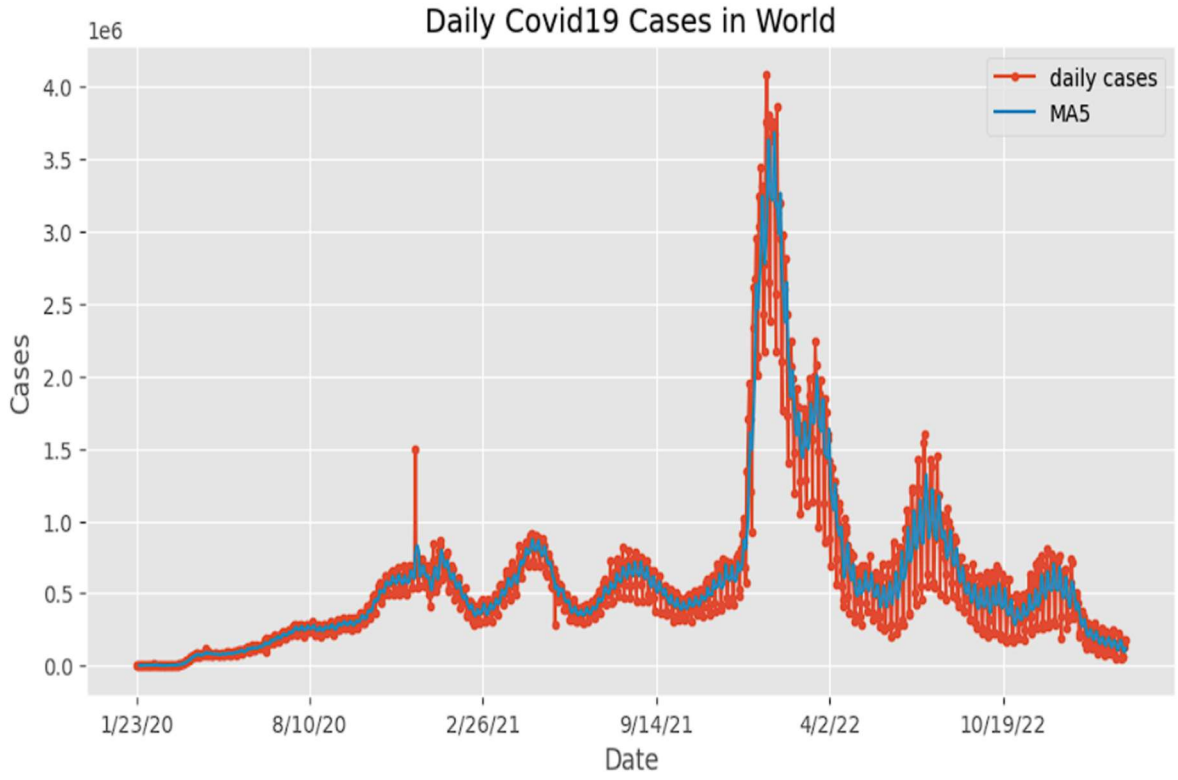
To visualize the data , we used a choropleth map which shows the number of cases across the world visually using different colors.

```
world["Cases Range"]=pd.cut(world["Cases"],[-150000,50000,200000,800000,1500000,15000000],labels=["U50K","50Kto200K","200Kto800K","800Kto1.5M","1.5M+"])
alpha =[]
for i in world["Country"].str.upper().values:
    if i == "BRUNEI":
        i="BRUNEI DARUSSALAM"
    elif  i=="US":
        i="UNITED STATES"
    if len(continent[continent["name"]==i]["alpha-3"].values)==0:
        alpha.append(np.nan)
    else:
        alpha.append(continent[continent["name"]==i]["alpha-3"].values[0])
world["Alpha3"]=alpha

fig = px.choropleth(world.dropna(),
                    locations="Alpha3",
                    color="Cases Range",
                     projection="mercator",
                     color_discrete_sequence=["white","khaki","yellow","orange","red"])
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



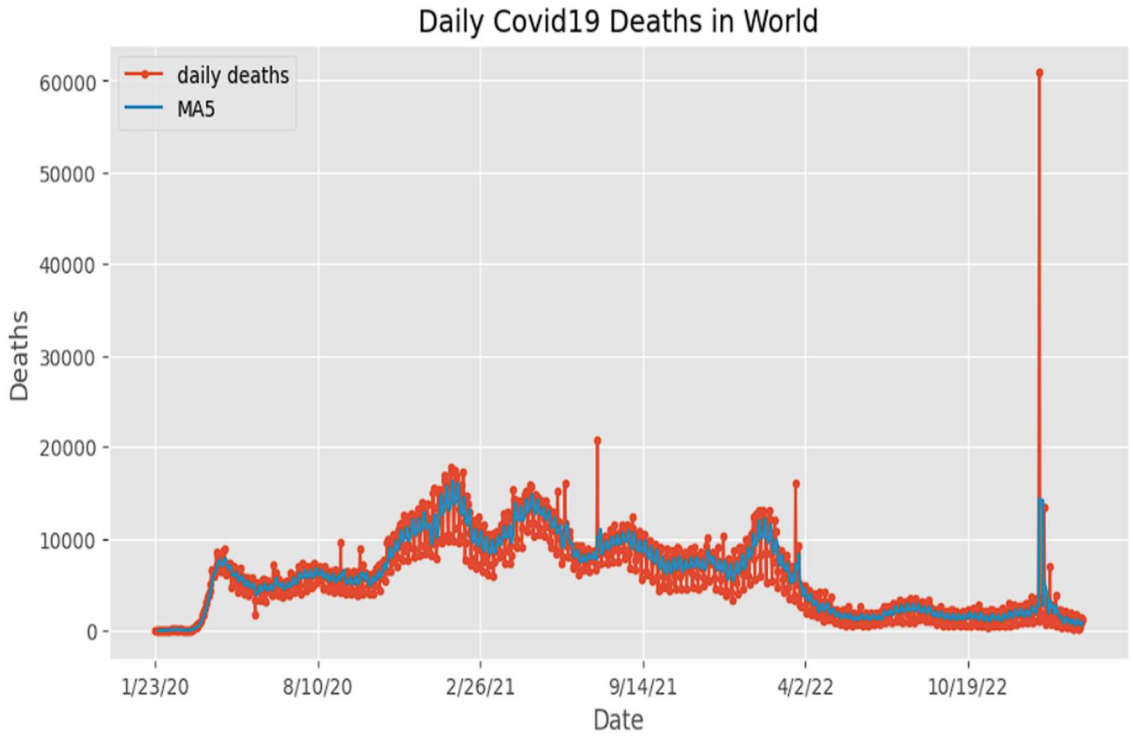
We used time series analysis to represent the number of cases across the world visually by using rolling window technique we also represented the global deaths in the time series graph using the rolling window technique

```
count = []
for i in range(1,len(df0)):
    count.append(sum(pd.to_numeric(df0.iloc[i,1:].values)))

df = pd.DataFrame()
df["Date"] = df0["Country/Region"][1:]
df["Cases"] = count
df=df.set_index("Date")

count = []
for i in range(1,len(df1)):
    count.append(sum(pd.to_numeric(df1.iloc[i,1:].values)))

df["Deaths"] = count

df.Cases.plot(title="Daily Covid19 Cases in World",marker=".",figsize=(100,50),label="daily cases")
df.Cases.rolling(window=5).mean().plot(figsize=(10,5),label="MA5")
plt.ylabel("Cases")
plt.legend()
plt.show()
```



The above plot represents the global confirmed cases over the time. The red line represents the number of confirmed cases on a particular day and the blue line represents the rolling average of the confirmed cases over a particular window size.

```
df.Deaths.plot(title="Daily Covid19 Deaths in World",marker=".",figsize=(10,5),label="daily deaths")
df.Deaths.rolling(window=5).mean().plot(figsize=(10,5),label="MA5")
plt.ylabel("Deaths")
plt.legend()
plt.show()
```



The above plot represents the global deaths over the time. The red line represents the number of deaths occurred on a particular day and the blue line represents the rolling average of the deaths.

6.5 Train ARIMA Model:

The ARIMA (AutoRegressive Integrated Moving Average) model stands as a statistical powerhouse for analyzing and forecasting time series data.

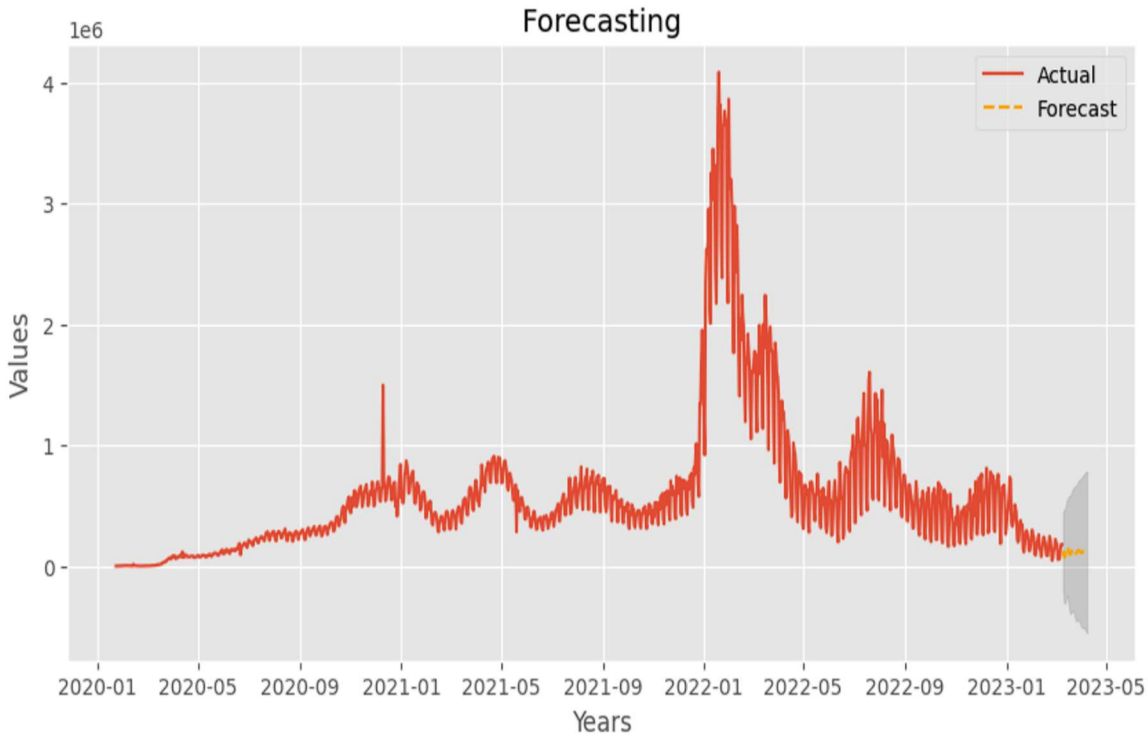
It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration.

```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

class ARIMAModel(object):
    def fit(self, data):
        self.data = data
        self.model = ARIMA(self.data['y'], order=(5,1,0))  # Adjust order based on your data characteristics
        self.results = self.model.fit()

    def forecast(self, periods):
        last_date = self.data['ds'].iloc[-1]
        forecast_index = pd.date_range(start=last_date, periods=periods + 1, freq='D')[1:]
        self.forecast_values = self.results.get_forecast(steps=periods)
        self.df_forecast = pd.DataFrame({'ds': forecast_index, 'yhat': self.forecast_values.predicted_mean})


    def plot(self, xlabel="Years", ylabel="Values"):
        plt.figure(figsize=(10, 5))
        plt.plot(self.data['ds'], self.data['y'], label='Actual')
        plt.plot(self.df_forecast['ds'], self.df_forecast['yhat'], label='Forecast', linestyle='--', color='orange')
        plt.fill_between(x=self.df_forecast['ds'], y1=self.forecast_values.conf_int()['lower y'],
                         y2=self.forecast_values.conf_int()['upper y'], color="gray", alpha=0.3)
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        plt.legend()
        plt.title("Forecasting")
        plt.show()
```

```
df_arima = pd.DataFrame({"ds": [], "y": []})
df_arima["ds"] = pd.to_datetime(df.index)
df_arima["y"] = df.iloc[:, 0].values

arima_model = ARIMAModel()
arima_model.fit(df_arima)
arima_model.forecast(30)
arima_model.plot()
```

In the above code the data is fit into the model and the values are forecasted and are plotted. The functions fit, forecast, plot are used to execute the mentioned operations .



In the above forecast graph the dotted line represents the number of cases that will occur in the next 30 days and the shaded region represents the possible deviation according to the data and the model.

7. TESTING

In machine learning, testing is mainly used to validate raw data and check the ML model's performance. The main objectives of testing machine learning models are:

- Quality Assurance
- Detect bugs and flaws

Once your machine learning model is built (with your training data), you need unseen data test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Testing data has two main criteria. It should:

- Represent the actual dataset
- Be large enough to generate meaningful predictions

7.1 TYPES OF TESTING

7.1.1 MANUAL TESTING

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is "**100% Automation is not possible**". This makes Manual Testing imperative.

7.1.2 AUTOMATED TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

7.2 TESTING LEVELS

7.2.1 NON-FUNCTIONAL TESTING

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

7.2.1.1 PERFORMANCE TESTING

Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measure according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

7.2.1.2 STRESS TESTING

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

7.2.1.3 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

7.2.1.4 PORTABILITY TESTING

Portability Testing is one of Software Testing which is carried out to determine the degree of ease or difficulty to which a software application can be effectively and efficiently transferred from one hardware, software or environment to another one. The results of portability testing are measurements of how easily the software component or application will be integrated into the environment and then these results will be compared to the non-functional requirement of portability of the software system.

7.2.1.5 USABILITY TESTING

Usability Testingalso known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

7.2.2 FUNCTIONAL TESTING

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application. Functional testing also called as black-box testing.

7.3 TEST CASES

Sl.no	Testcase	Expected Result	Actual Result	Pass/ Fail
1.	After Training testing with test sample data	Predicting the number of cases in next 30 days	Gives the number of cases along with the possible deviation	PASS

Table 7.3.1 testcases

8. RESULTS

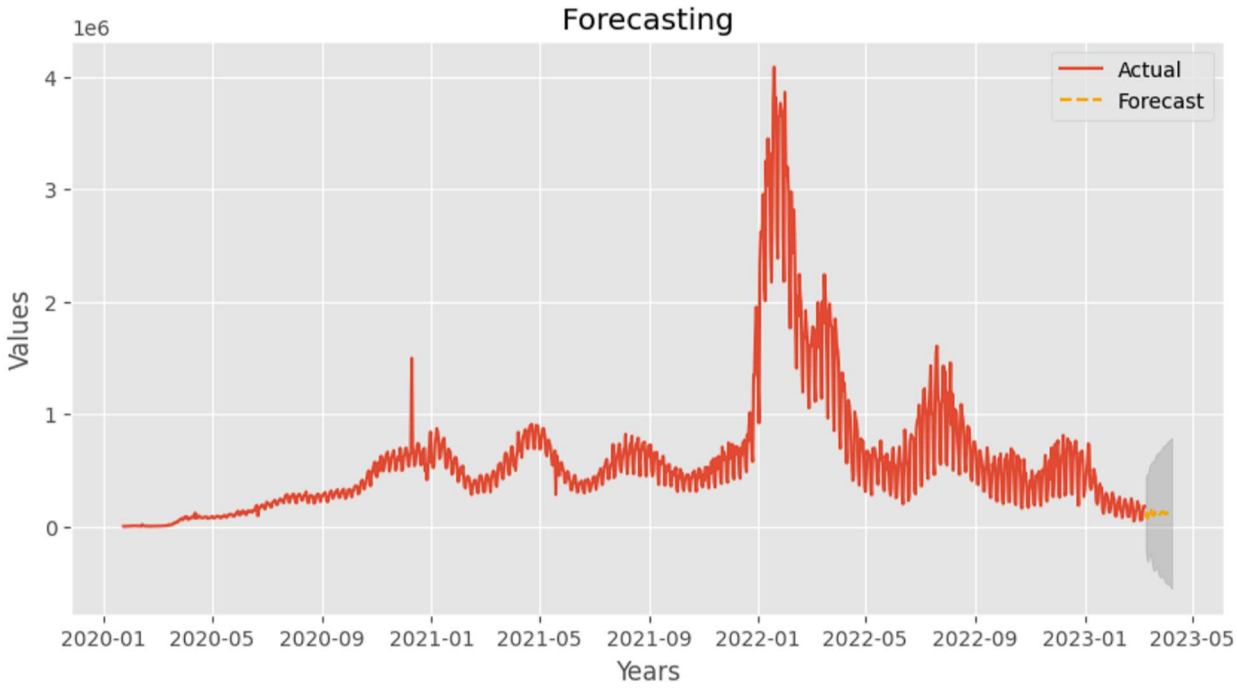


fig 8.1ARIMA model forecasting

The above plot gives the representation of the forecast of the data in the next 30 days using the time series plot . The red line represents the values present in the dataset according to their dates and the dotted line represents the forecasted number of cases in the next 30 days according to the given data and the model.

9. CONCLUSION AND FURTHER WORK

Our project aims to make better predictions about how many people might get sick with COVID-19 using technologies such as Machine Learning and Time Series Analysis. we use python libraries to visualize the data using chloropeth maps and time series analysis and visualization and forecasting the future no of cases using ARIMA model for the next 30 days. This helps to the ongoing global efforts to strengthen preparedness and response mechanisms. By embracing technology and data-driven approaches, we aim to improve our understanding of the evolving dynamics of COVID-19 outbreaks. The insights gained from our predictive models have the potential to empower decision-makers, healthcare professionals, and policymakers to implement targeted strategies for mitigating the impact of future virus outbreaks.In future the project can be further enhanced by producing a UI. Collecting the data regularly and updating the data gives more accurate results . this project can be further enhanced by using the local medical facilities data which helps to alert the authorities in case of a crisis.

10.REFERENCES

- [1] Kim, Juhyeon, and Insung Ahn. "Infectious disease outbreak prediction using media articles with machine learning models." Scientific reports 11.1 (2021): 4413.
- [2] Rees, E., et al. "Early detection and prediction of infectious disease outbreaks." CCDR 45.5 (2019).
- [3] Ardabili, Sina F., et al. "Covid-19 outbreak prediction with machine learning." Algorithms 13.10 (2020): 249.
- [4] Pandey, Gaurav, et al. "SEIR and Regression Model based COVID-19 outbreak predictions in India." arXiv preprint arXiv:2004.00958 (2020).

Datasets:

- https://github.com/CSSEGISandData/COVID-19
- https://www.kaggle.com/datasets/folaraz/world-countries-and-continents-details