

A Project-2 Report on

**A TIME-SERIES PREDICTION MODEL USING
LONG-SHORT TERM MEMORY NETWORKS FOR
PREDICTION OF COVID- 19 DATA**

Submitted to

The Department of Computer Science and Engineering

In partial fulfillment of the academic requirements of

Jawaharlal Nehru Technological University

For

The award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

(2017 – 2021)

By

BADDAM HIMABINDU	17311A05D2
CHEEDU SOWMYA	17311A05D4
MANNEM NAGALAXMI	17311A05F1

Under the Guidance of

Ms. J. NEHA



Sreenidhi Institute of Science and Technology

(An Autonomous Institution)

Yamnapet, Ghatkesar, R.R. District, Hyderabad – 501301

Department of Computer Science and Engineering

Sreenidhi Institute of Science and Technology



CERTIFICATE

This is to certify that this Group Project report on “**A TIME-SERIES PREDICTION MODEL USING LONG-SHORT TERM MEMORY NETWORKS FOR PREDICTION OF COVID-19 DATA**”, submitted by BADDAM HIMABINDU(17311A05D2), CHEEDU SOWMYA(17311A05D4), MANNEM NAGALAXMI(17311A05F1) in the year 2021 in partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as part of their **Project-2 during Fourth Year Second Semester**, under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Internal guide

Ms. J. Neha
Assistant Professor
Department of CSE

Project Co-ordinator

Mrs. P. Vasavi
Assistant Professor
Department of CSE

Head of Department

Dr. Aruna Varanasi
Professor & HOD
Department of CSE

External Examiner

Date:-

DECLARATION

We, **BADDAM HIMABINDU(17311A05D2), CHEEDU SOWMYA(17311A05D4) and MANNEM NAGALAXMI(17311A05F1)**, students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR**, studying IVth year 2nd semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Industry Oriented Mini project work, titled “**A TIME-SERIES PREDICTION MODEL USING LONG-SHORT TERM MEMORY NETWORKS FOR PREDICTION OF COVID-19 DATA**” is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **Dr. T. Ch. Siva Reddy**, who most ably run the institution and has had the major hand in enabling me to do my project.

I profoundly thank Dr. **ARUNA VARANASI**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank my internal guide **Ms.J NEHA** for his/her technical guidance, constant encouragement and support in carrying out my project at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

BADDAM HIMABINDU
CHEEDU SOWMYA
MANNEM NAGALAXMI

17311A05D2
17311A05D4
17311A05F1

Abstract

As of mid-May 2020, the current outbreak of Coronavirus disease 2019 (COVID-19), which is caused by the coronavirus 2 (SARS-CoV-2) that causes severe acute respiratory syndrome (SARS), has killed over 3,00,000 individuals and infected over 4.7 million people over the world. During this time, there have been over 1.8 million recoveries. It is critical for governments to be aware of the situation and to be able to forecast future patient numbers in order to maintain health-care readiness and plan for other essential steps. Using the Long-Short Term Memory (LSTM) network as a driving force, a model for predicting the number of COVID-19 patients was constructed and subsequently used it for forecasting future cases. India's cases are taken into consideration. According to the findings, the LSTM network constructed in this paper outperforms other networks and hence could be a good contender for predicting the number of COVID-19 patients in the future.

LIST OF FIGURES

S.No	Fig No	Title of Figure	Page No
1	4.1	Architecture Diagram	7
2	4.2	Architecture for lstm	8
3	4.2.1	Use Case Diagram	12
4	4.2.2	Class Diagram	13
6	4.2.3	Sequence Diagram	14
7	4.2.4	Activity Diagram	15
8	5.1.1	Dataset	16
9	5.1.1.1	Dataset Image 1	17
10	5.1.1.2	Dataset Image 2	18
11	5.2.2.1	Grouping Data	22
12	5.2.2.2	Get all lists	23
13	5.2.3.1	Code Snippet for Data preparing	24
14	5.2.4.1	Code spinet for splitting of data	25
15	5.2.5.1	Code snippet for loading long short term Memory(lstm)model	25
16	5.2.6.1	Code snippet for Prediction	26
17	5.2.7.1	Code snippet for model accuracy	27
18	5.2.8.1	Code snippet for the graph	27

INDEX	Page Number
List of Figures	i
1. INTRODUCTION	
1.1 Motivation	1
1.2 Problem Definition	1
1.3 Objective of Project	1
2. LITERATURE SURVEY	
2.1 Related Work	2
2.1 Existing System	2
2.2 Proposed System	4
3. SYSTEM ANALYSIS	
3.1 Functional Requirement Specifications	5
3.2 Software Requirements	5
3.3 Hardware Requirements	6
3.2 Performance Requirements	6
4. SYSTEM DESIGN	
4.1 Architecture of Proposed System	7

4.2 Modules	10
4.2 UML Diagrams	11

4.2.1 Use Case Diagram	12
4.2.2 Class Diagram	13
4.2.3 Sequence Diagram	14
4.2.4 Activity Diagram	15
5. IMPLEMENTATION AND RESULTS	
5.1 Methodology	16
5.2 Implementation	22
5.3 Results	28
6. TESTING	
6.1 Types of Testing	31
6.2 List of Test Cases	31
7. CONCLUSION AND FUTURE SCOPE	33
BIBLIOGRAPHY	34

Appendix-A:Python Programming	36
-------------------------------	----

Appendix-B:Unified Modeling Language	39
Plagiarism Report	44
PO and PSO correlation	47

1.INTRODUCTION

1.1 MOTIVATION

In the current scenario there has been no treatment method or vaccination available, and therefore the only safeguards against this pandemic are social distancing and washing hands frequently. Many countries have implemented lock-down to impose the social distancing and reducing crowd spread that has caught up the spread. However, this action has resulted in an enormously big economic slowdown and can't be the future solution for this pandemic. Currently, it's a significant health crisis across the world and it'd not be wrong to mention that it's 'a backdrop to humanity'. During this circumstance, the sole option is preventing the spreading of infection and preparing our healthcare system for the probable up-comings.

1.2 Problem Definition

The primary motive for the project is to develop a system which is useful for covid cases prediction with improved accuracy than the existing methods. It involves predicting the total number of covid cases in future with high accuracy rate from the different classification algorithms implemented. The data pre-processing is performed on the dataset which would involve process like data cleaning, data scaling and data labelling. This project is implemented using the Indian Covid cases Dataset. This is the dataset taken from the WHO website. With the available standard dataset, appropriate data pre-processing techniques and implementing Long Short Term Memory neural networks It helps for the prediction future covid cases which makes the diagnosis process easier and quicker.

1.3 Objectives of Project

The main aim of the project is to predict the covid cases using LSTM model. Due to the accuracy of the model, it becomes more efficient and dependable. We develop a code which helps in the determination of graph of confirmed cases and also accuracy which proves that it is better than other classical algorithms. This system can be used directly using by the end user.

2.LITERATURE SURVEY

2.1 Related Work

Firstly, the dataset for our project was taken from the standard dataset from the WHO repository of world covid cases. The name of the dataset used for our project is “World Covid cases Dataset”. It consists of 285308 records, covid cases of the entire world state wise. Further, we have referred to technical papers available to study the existing system so that we can build a project which is better and more accurate than the existing system. This would be discussed in depth in the following section.

2.1 EXISTING APPROACH:

The recent outbreak of Coronavirus 2019 (COVID-19), which is caused by severe acute respiratory syndrome (SARS) coronavirus 2 (SARS-CoV-2), is answerable for the deaths of over 3,00,000 people and at the identical time has infected more than 4.7 million people within the whole world as of mid- May, 2020. There has been around that 1.8 million people who recoveries during this era too. The number of covid patients are increasing at a high level, especially in India. The existing methods do not have satisfying accuracy rates. There may also be cases where the prediction is false. In this case, it would lead to problem with the healthcare system, patient’s health and even dead. False Prediction will lead to disastrous results.

In the paper titled, “Prediction of covid-19 cases using CNN and X-rays” explains the CNN algorithm used for the prediction of disease. The primary motive of the paper was to provide an analysis on convolutional - neural networks skipping the pre-processing and also checking that reduced layers is capable of detecting COVID-19 in a limited count of, and even in not balanced, chest :-X-ray images. This paper only discusses about algorithms and its performance but if we consider the dataset it requires all the x-ray images of covid effected and non-effected people, it increases the cost as it involves CT scan .It also increases the computational cost because various image proportions, different network architectures, state of the art pre trained networks; and with the help of images and statistical data , machine learning models are evaluated and

implemented .

Secondly, in a paper titled, “Coronavirus Disease (COVID-19) Global Prediction Using Hybrid Artificial Intelligence Method of ANN Trained with Grey Wolf Optimizer”. Its objective was to engross an artificial neural network collaborated with grey wolf optimizer to predicting the covid19 severity using global dataset. The maximum accuracy obtained after analysis is very less. There are chances for better accuracies using other machine learning techniques.

Thirdly, the paper titled “Software- based Prediction of Liver Disease with Feature Selection and Classification Techniques”. The intention of this paper is to use the updated SEIR model to observe the changing of trend in the early stage of the Covid-19 it explains the SEIR model Susceptible-S, Exposed-E, Infector-I and Recovered-R used for prediction of covid cases trend. These methods are utilized to analyze the present data and foresee the pattern of disease in the main cities of China. The main disadvantage of mathematical models are they have high complexity and low reliability. But there are new algorithms emerging which may provide better accuracies with less complexity.

Lastly, in a paper titled, “Quantitative Factors and Mathematical Modeling of CoVID-19 Pandemic Under Human Interventions”. This paper explains the transmission dynamics of the novel coronavirus (CoVID-19) pandemic with the human intervention is studied. A standard model is built based on the features of the pandemic, and the concept of quantitative factors are introduced to quantitatively measure the intensity of prevention and control efforts and also the control efficiency on the event of the pandemic; using the information obtained daily by Hubei Provincial Health Commission, China National Health Commission, World Health Organization (WHO) and John Hopkins CSSE. This paper does not suggest any proved implementation of the technologies. It does not show any proved test cases.

The existing system consists of accuracy rates which are not satisfactory. The use would lead to false prediction which may lead to disastrous results.

2.2 Proposed System

In our paper, a method using the Long-Short Term Memory LSTM network is been proposed for the prediction of the total count of effected patients. Implementing this network, the new future covid cases can be analyzed with even high accuracy. With the help of this prediction model the policy designers and healthcare along with frontline workers can be ready to face the upcoming situations. In this we upload our dataset along with the saved model, the lstm model will be loaded then trained and finally tested. Using testing output, we calculate the accuracy of the model based on the accuracy we decide the saved model. Now we use the model with highest accuracy to predict the future cases. The dataset used for the system is built through a statistical analysis applied to the information obtained from the IMF, WB, WHO and IT. It consists of 285308 rows and 8 columns as we are working on the Indian covid cases we extract the Indian records which are of 459 records.

For the diagnosis of the covid diseases to be more effective, efficient and accurate we have to prevent the false prediction of the covid cases. Misdiagnosis of any disease is not a problem until the covid disease patient is diagnosed as a healthy individual. Developing a system with better performance and precision than the existing system to help in preventing false prediction of the covid cases helps in providing the best and required medication and treatment for the patient by improving the healthcare system and alerting the people in advance.

3. SYSTEM ANALYSIS

3.1 Functional Requirements Specifications

These are the requirements that are to be offered by the system to the end-user. All these functionalities need to be undoubtedly integrated into the system as a part of the contract. These are represented or stated in the form of input to the system, the operations are performed and we get expected result. They are the basic requirements stated by the user which are reflected directly in the final product, far from the non-functional requirements

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modelling
5. Predicting

3.2 Software Requirements

The product perspective and features, operating system and operating environment, graphics requirements, design limitations, and user documentation are all included in the functional requirements or overall description documents. The appropriation of needs and implementation limitations provides a comprehensive picture of the project in terms of its strengths and weaknesses, as well as how to address them.

- Python idle 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

3.3 Hardware Requirements

- Processor : Intel i3 and above
- RAM : 4GB and Higher
- Hard Disk : 500GB: Minimum

3.4 Performance requirements

Performance is calculated in terms of with the application's output, which is mostly based on accuracy. The formulation of requirements is critical in the analysis of a system. It is possible to create a system only when the requirement specifications are properly stated. Any system's need specification can be stated as follows:

1. The system should be precise.
2. The system should be able to interface with the existing system.
3. The system should be better than the previous system.

4. SYSTEM DESIGN

4.1 Architecture of Proposed System

The architecture of the proposed system involves the input data, the trained model and the output. The user provides the feature values as input and these are processed by the trained model. Further the output is shown to the user as a graph. The dataset is divided into Train Dataset and Validation Dataset. Both the datasets are used to build an appropriate Classification of Model Training and further proceed with performance evaluation. Hence, we achieve the resulting in trained model. We can provide the input and then predict the output.

An architecture diagram is a system diagram that abstracts the overall structure of a software system as well as the interactions, constraints, and boundaries between its components. It's an important tool because it gives you a bird's-eye view of the software system's physical deployment as well as its evolution path.

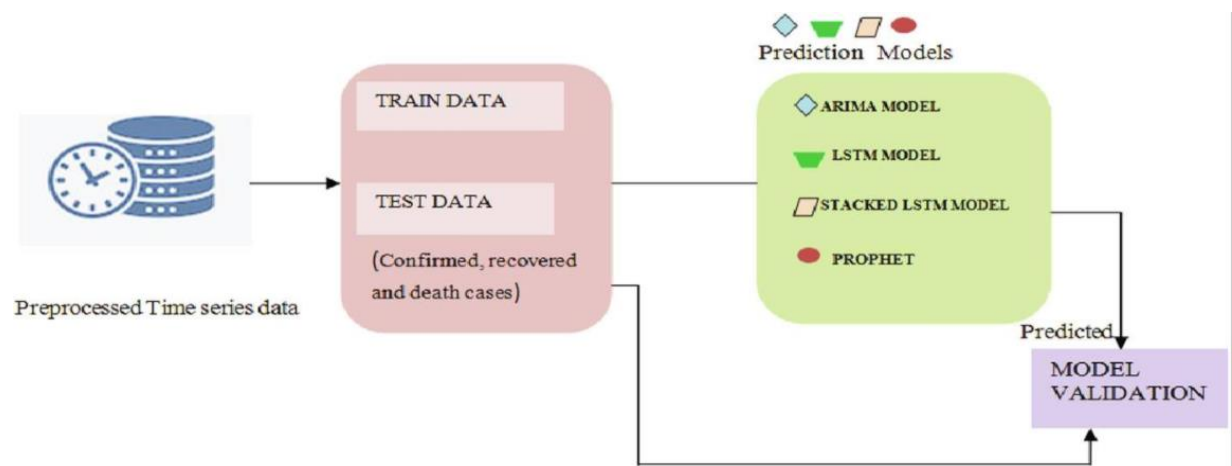


Fig 4.1 : Architecture Diagram

We have made use of a simple Long Short Term Memory (LSTM) model which consist of an input layer, a single hidden layer, and an output layer that is implemented to make

for a prediction.

Before constructing the LSTM model, the following data pre-processing and feature engineering must be completed.

- Create the dataset and make sure that all of the data is float.
- Make the features more consistent.
- Divide the data into training and test sets.
- Create a dataset matrix from an array of values.
- Resize to $X=t+1$ and $Y=t$.
- Change the input to a three-dimensional format (num samples, num timesteps, num features).

Some neurons in the input layer are equal to 5 sequence steps. An LSTM layer with 10 hidden units (neurons) and a rectified linear unit (ReLU) as an activation function makes up the hidden layer. For anticipating the output, the output layer had a dense layer with 1 unit. The testing rate is set to 0.2, with a five-epoch decay. Furthermore, we chose 300 epochs as the number of iterations, Adam as the optimizer, and the loss function is the mean square error. The model was then fitted to the data in order to make a forecast. Because of the stochastic nature of the LSTM model, the generated results may vary; as a result, we ran it numerous times. Finally, we use the output from the previous sequence to predict the next value in the series.

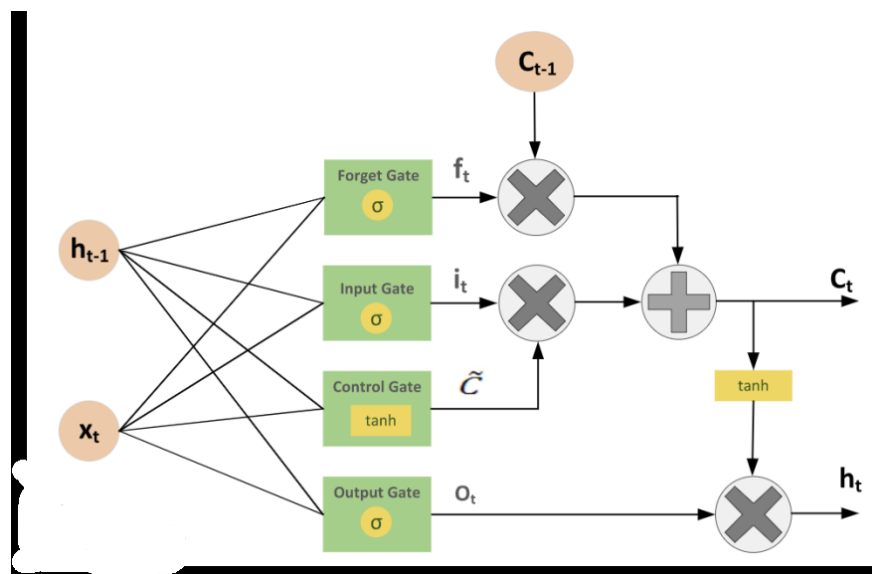


Fig 4.2 : Architecture of lstm

Long short-term memory (LSTM) is a deep learning architecture that uses an artificial recurrent neural network (RNN). LSTM has feedback connections, unlike normal feedforward neural networks. It can process not only single data points (like photos), but also complete data sequences (such as speech or video). For example, activities like unsegmented, connected handwriting identification, speech recognition, and anomaly detection in network traffic or IDSs can all benefit from LSTM (intrusion detection systems).

A cell, an input gate, an output gate, and a forget gate make up a typical LSTM unit. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals.

Because there might be lags of undetermined duration between critical occurrences in a time series, LSTM networks are well-suited to categorizing, processing, and making predictions based on time series data. LSTMs were created to solve the problem of vanishing gradients that can occur when training traditional RNNs. In many cases, LSTM has an advantage over RNNs, hidden Markov models, and other sequence learning approaches due to its relative insensitivity to gap length.

STM networks are a sort of recurrent neural network that may learn order dependence in sequence prediction challenges. LSTMs solve two technical problems: vanishing gradients and exploding gradients, both of which are connected to how the network is taught. One input layer, one hidden layer, and one output layer are used in our networks. Memory cells and gate units are found in the (completely) self-connected hidden layer. An LSTM layer is made up of memory blocks, which are recurrently connected blocks. Each one has one or more recurrently connected memory cells as well as three multiplicative units (input, output, and forget cell).

In the input sequences, RNNs can maintain track of arbitrary long-term dependencies. The issue with vanilla RNNs is computational (or practical) in nature: when employing back-propagation to train a vanilla RNN, the back-propagated gradients can "vanish" (that is, drift to zero) or "explode" (that is, they can tend to infinity), because the process involves computations that use finite-precision numbers. Because LSTM units allow

gradients to flow unchanged, RNNs using LSTM units partially solve the vanishing gradient problem.

4.2 Modules

The modules help to understand the functional area. The project is divided into modules where each discrete unit has a particular functionality. The modules can be used to logically separate the functionality for the project. We can normally have one or more than a single module in any project.

There are three main important modules that can be defined in our project: (a) Developer (b) Server and (3) User

- Developer:** The developer plays the most important role. They are responsible for taking out tasks from the initial pre-processing of dataset to the building the model. The developer is also responsible for making further developments in the later stages as per the requirements of the user. The developer does the data
- Server:** The dataset used for our project is uploaded in the server and read by the algorithm. It is further continued to process. The final model is deployed into the server which the end user can use in for practical application.
- End User:** The end user also plays the important role. The model is built and changed accordingly to the requirements of them. The end user is the one who uses the final and provides feedback regarding any changes to be made.

4.3 UML Diagrams

The UML diagrams helps in better visualization of any project. They help in understanding the project and also describe the purpose with easy visualization.

System requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces are all described in the System Design Document.

Some of the important uses of UML diagrams is described as follows:

- It helps in understanding the behaviour of the end resultant system.
- We can also spot any mistakes or problems in the beginning stages itself.
- It helps in communicating the proposed project effectively with no misconceptions.
- It helps in getting to know the requirements and also helps to ideate from where we can implementation i.e., the beginning step.
- It can be used in different fields like web applications, embedded systems etc.
- One of the most important reasons for UML's widespread adoption is its tools. It comes with a plethora of tools. These tools are useful for a lot more than just generating diagrams.

In the following section, we have discussed about the different UML diagrams for our project.

4.3.1 Use Case Diagram

The use case diagram depicts the situation details regarding the actors and their functionalities. It is basically a graphical representation. Following use case diagram consists of the following actors

- Developer
- End user

Following is the use case diagram which provides information about the actors and their functionality or role:

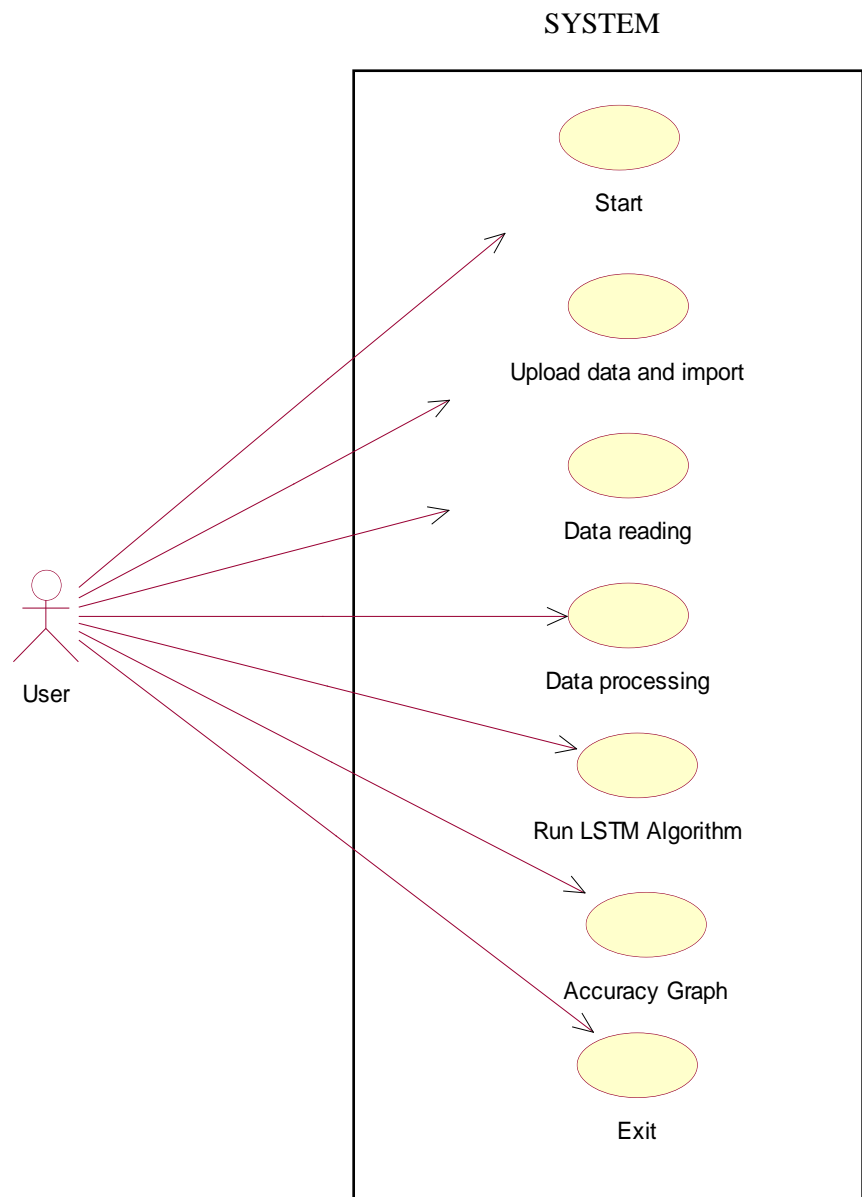


Fig 4.2.1.: Use Case Diagram

4.3.2 Class Diagram

The UML class diagram plays a prominent role in object-oriented programming. The class diagram depicts the static structure which gives information about the classes in the system, the attributes and operations as well as the relationship amongst the various classes. The class diagram provides details about various classes in the system.

Following is the class diagram describing the different classes containing attributes and methods in our system.

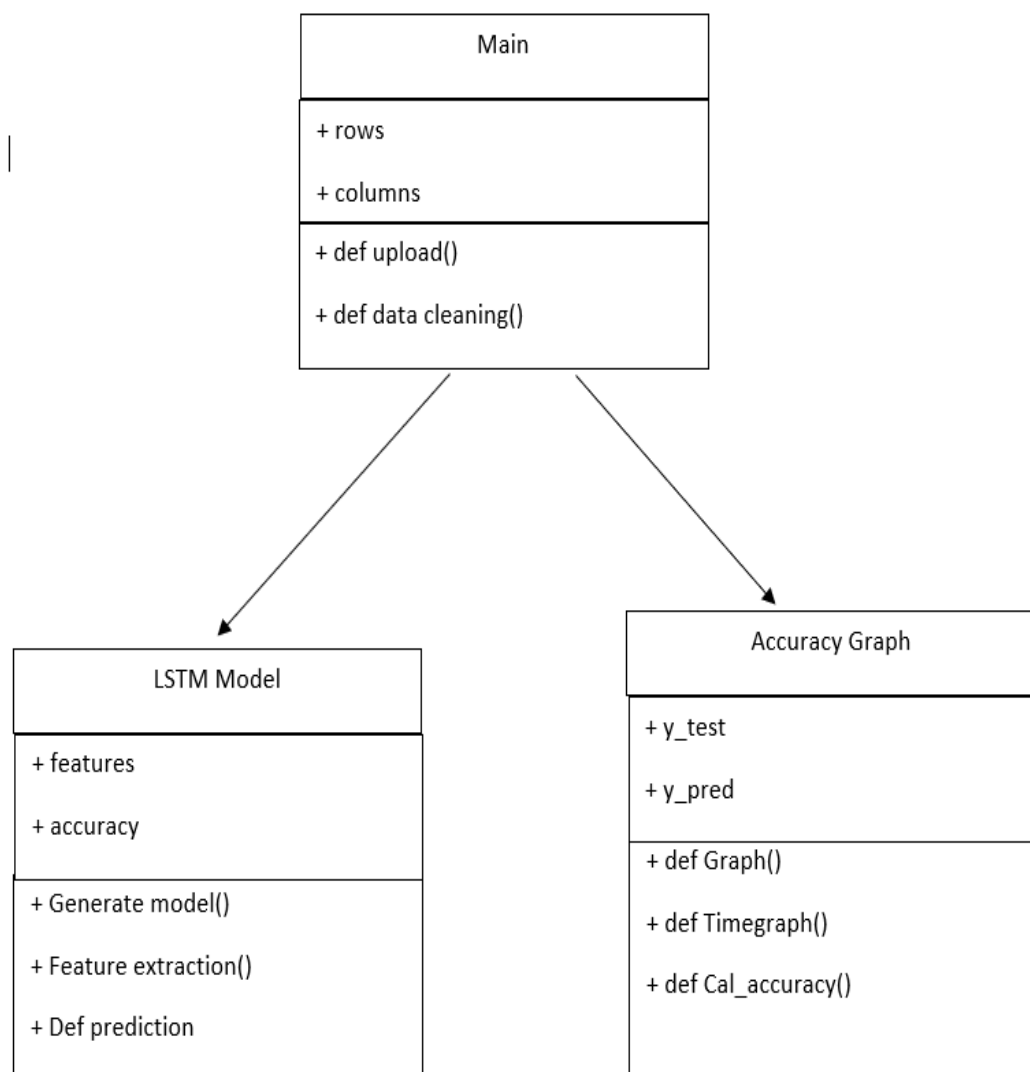


Fig 4.2.2.: Class Diagram

4.3.3 Sequence Diagram

The sequence diagram in UML provides information about how the operations are carried out. In other words, they provide details on what messages are sent and when. It is an interaction diagram where details are organized according to time. It provides dynamic view of the system i.e., the model.

Following is the sequence diagram for our model:

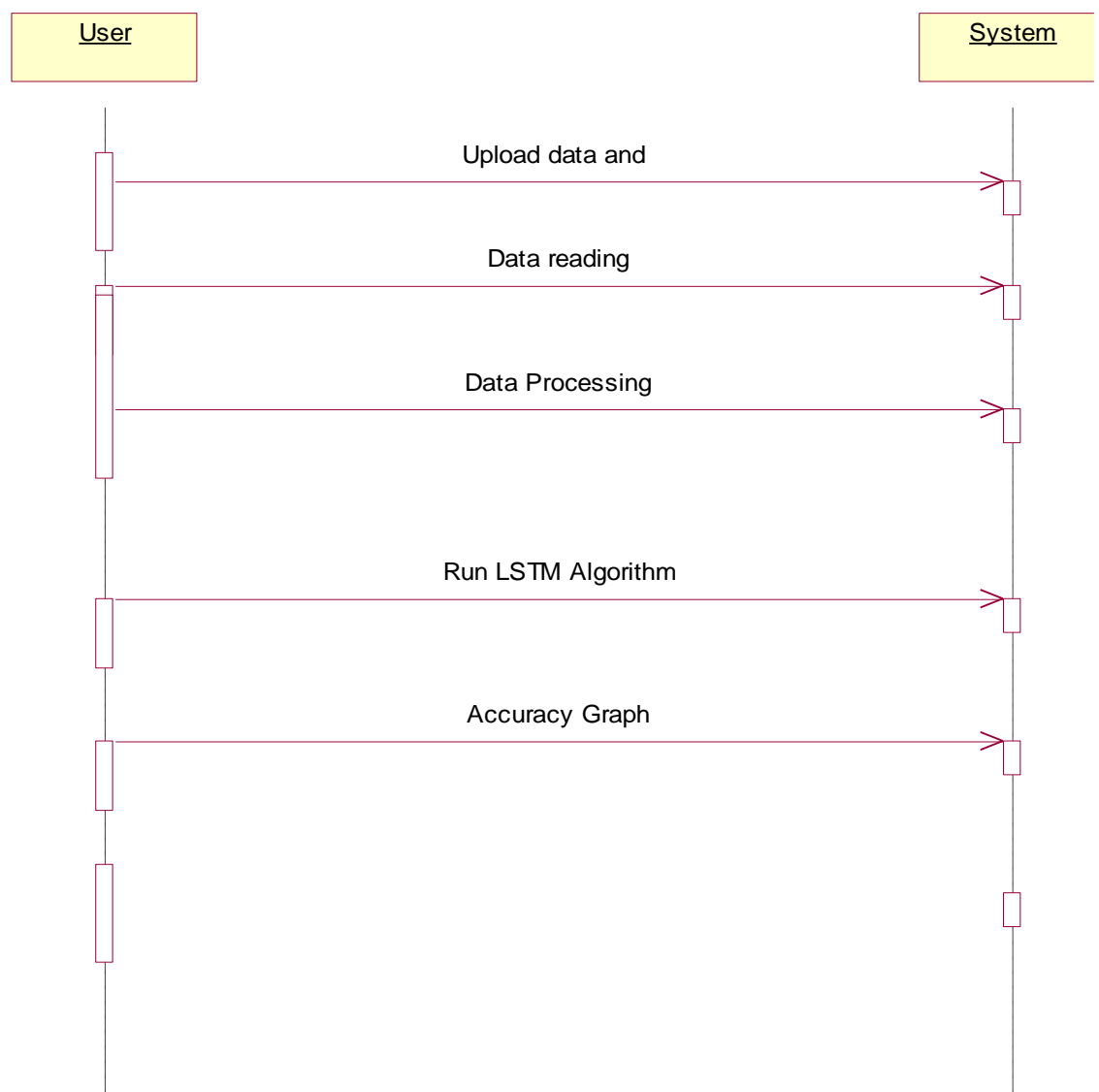


Fig 4.2.3: Sequence diagram

4.3.4 Activity Diagram

The umlactivity diagram is used for representing the work flow of the activities. It helps in showing the stepwise activities for a model. It shows the control flow from the beginning point until the finishing point depicting various decision paths that are present when the activity is executed.

The activity diagram consists of rounded rectangles which are used for representing actions, bars which are used for representing the start or end of the activities and a black circle depicting the start and end of the control flow.

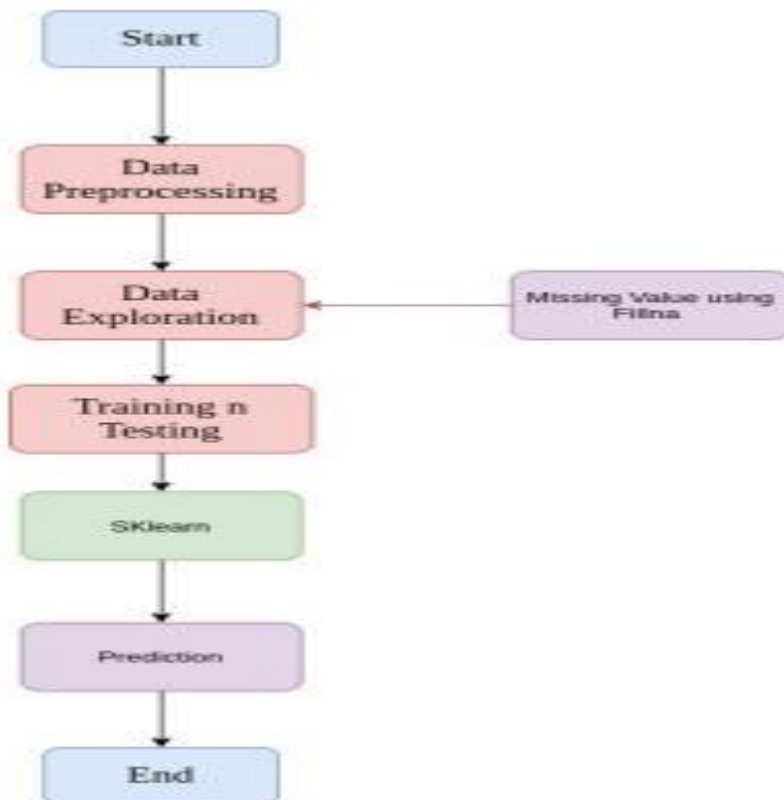


Fig 4.2.4: Activity Diagram

5.IMPLEMENTATION

5.1 Methodology

5.1.1 Dataset

The dataset used for the system is built through a statistical analysis applied to the information obtained from the IMF, WB, WHO and IT. This dataset comprises of 7 attributes having information of all countries covid cases. The data set consists of 5 attributes of numerical type and two attributes of nominal type. We extract the Indian covid cases and group them based on the observation date in the data pre-processing.

Following figure gives the details regarding the various attributes for the dataset:

S.NO	ATTRIBUTES	ATTRIBUTES TYPE
1.	Observation Date	Numerical
2.	State	Nominal
3.	Country	Nominal
4.	Last Update	Numerical
5.	Confirmed cases	Numerical
6.	Recovered cases	Numerical
7.	Death cases	Numerical

Fig 5.1.1 Dataset

The result attribute is the numerical values of future confirmed covid cases

covid_19_data (1) - Excel

sowmya cheedu

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Clipboard Font Alignment Number Styles Cells Editing

Calibri 11 A A

B I U

General

Conditional Formatting Format as Table Cell Styles

Insert Delete Format

AutoSum Fill Sort & Find & Filter Select

P7

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	V
1	SNo	Observat Province/S	Country/R	Last Update		Confirmed Deaths	Recovered																
2	1	01/22/2020	Anhui	Mainland (1/22/2020 17:00		1	0	0															
3	2	01/22/2020	Beijing	Mainland (1/22/2020 17:00		14	0	0															
4	3	01/22/2020	Chongqing	Mainland (1/22/2020 17:00		6	0	0															
5	4	01/22/2020	Fujian	Mainland (1/22/2020 17:00		1	0	0															
6	5	01/22/2020	Gansu	Mainland (1/22/2020 17:00		0	0	0															
7	6	01/22/2020	Guangdong	Mainland (1/22/2020 17:00		26	0	0															
8	7	01/22/2020	Guangxi	Mainland (1/22/2020 17:00		2	0	0															
9	8	01/22/2020	Guizhou	Mainland (1/22/2020 17:00		1	0	0															
10	9	01/22/2020	Hainan	Mainland (1/22/2020 17:00		4	0	0															
11	10	01/22/2020	Hebei	Mainland (1/22/2020 17:00		1	0	0															
12	11	01/22/2020	Heilongjiang	Mainland (1/22/2020 17:00		0	0	0															
13	12	01/22/2020	Henan	Mainland (1/22/2020 17:00		5	0	0															
14	13	01/22/2020	Hong Kong	Hong Kong 1/22/2020 17:00		0	0	0															
15	14	01/22/2020	Hubei	Mainland (1/22/2020 17:00		444	17	28															
16	15	01/22/2020	Hunan	Mainland (1/22/2020 17:00		4	0	0															
17	16	01/22/2020	Inner Mon	Mainland (1/22/2020 17:00		0	0	0															
18	17	01/22/2020	Jiangsu	Mainland (1/22/2020 17:00		1	0	0															
19	18	01/22/2020	Jiangxi	Mainland (1/22/2020 17:00		2	0	0															
20	19	01/22/2020	Jilin	Mainland (1/22/2020 17:00		0	0	0															
21	20	01/22/2020	Liaoning	Mainland (1/22/2020 17:00		2	0	0															
22	21	01/22/2020	Macau	1/22/2020 17:00		1	0	0															
23	22	01/22/2020	Ningxia	Mainland (1/22/2020 17:00		1	0	0															
24	23	01/22/2020	Qinghai	Mainland (1/22/2020 17:00		0	0	0															
25	24	01/22/2020	Shaanxi	Mainland (1/22/2020 17:00		0	0	0															
26	25	01/22/2020	Shandong	Mainland (1/22/2020 17:00		2	0	0															
27	26	01/22/2020	Shanghai	Mainland (1/22/2020 17:00		9	0	0															
28	27	01/22/2020	Shanxi	Mainland (1/22/2020 17:00		1	0	0															
29	28	01/22/2020	Sichuan	Mainland (1/22/2020 17:00		5	0	0															

covid_19_data (1)

Ready

Type here to search

34°C 14:40 28-05-2021

Figure 5.1.1.1 Dataset image 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
205368	205367	01/18/202	Chongqing	Mainland C	02-04-2021 15:13	591	6	584														
205369	205368	01/18/202	Chukotka	Russia	02-04-2021 15:13	582	4	518														
205370	205369	01/18/202	Chuvashia	Russia	02-04-2021 15:13	18181	725	16336														
205371	205370	01/18/202	Ciudad de	Mexico	02-04-2021 15:13	409551	19531	0														
205372	205371	01/18/202	Coahuila	Mexico	02-04-2021 15:13	55256	4705	0														
205373	205372	01/18/202	Colima	Mexico	02-04-2021 15:13	8650	839	0														
205374	205373	01/18/202	Colorado	US	02-04-2021 15:13	376171	5386	0														
205375	205374	01/18/202	Connecticut	US	02-04-2021 15:13	230125	6670	0														
205376	205375	01/18/202	Cochimbo	Chile	02-04-2021 15:13	15491	322	14654														
205377	205376	01/18/202	Cordoba	Colombia	02-04-2021 15:13	31134	1692	28701														
205378	205377	01/18/202	Crimea Re	Ukraine	02-04-2021 15:13	29762	654	23685														
205379	205378	01/18/202	Cundinam	Colombia	02-04-2021 15:13	83596	2077	75657														
205380	205379	01/18/202	Curacao	Netherlan	02-04-2021 15:13	4523	19	4358														
205381	205380	01/18/202	Cusco	Peru	02-04-2021 15:13	26355	563	0														
205382	205381	01/18/202	Dadra and	India	02-04-2021 15:13	3390	2	3377														
205383	205382	01/18/202	Dagestan	Russia	02-04-2021 15:13	26314	1187	22996														
205384	205383	01/18/202	Dalarna	Sweden	02-04-2021 15:13	12357	278	0														
205385	205384	01/18/202	Delaware	US	02-04-2021 15:13	70910	1210	0														
205386	205385	01/18/202	Delhi	India	02-04-2021 15:13	632590	10754	619501														
205387	205386	01/18/202	Diamond F	Canada	02-04-2021 15:13	0	1	0														
205388	205387	01/18/202	Diamond F	US	02-04-2021 15:13	49	0	0														
205389	205388	01/18/202	District of	US	02-04-2021 15:13	34033	857	0														
205390	205389	01/18/202	Distrito Fe	Brazil	02-04-2021 15:13	265274	4427	253569														
205391	205390	01/18/202	Dnipropet	Ukraine	02-04-2021 15:13	64557	1645	49998														
205392	205391	01/18/202	Donetsk O	Ukraine	02-04-2021 15:13	41817	762	35203														
205393	205392	01/18/202	Drenthe	Netherlan	02-04-2021 15:13	16978	212	0														
205394	205393	01/18/202	Durango	Mexico	02-04-2021 15:13	27119	1724	0														
205395	205394	01/18/202	East Fland	Belgium	02-04-2021 15:13	69556	0	0														
205396	205395	01/18/202	Ehime	Japan	02-04-2021 15:13	827	14	555														

Figure 5.1.1.1 Dataset image 2

In the above figures, we showing the dataset which is in the form csv file.

This has to be loaded into the algorithm for performing the training and testing operations. These entries are to be pre-processed which we will be discussing in the next section

5.1.2 Data Pre-processing

The most important part of every machine learning based system is data pre-processing. The dataset obtained maybe in raw format and cannot be fed directly into the machine. Almost all the datasets must be cleaned, processed and transformed in order to make them more precise and usable. This is because any un processed dataset consists of missing values, duplicates, outliers and null values etc. Such dataset when used without processing leads to less accurate results which are disastrous. Generally, every dataset is different and hence we can face various challenges. The data is dynamic in nature because the cases may vary depending on the seasons, population, and other factors. There may be different kinds of data which has to be made homogeneous accordingly with respect to the technologies used. In the case of our model, the missing values, duplicate values and the null values are removed in the data pre-processing. The outliers are also removed also we select only required data in our project as we are working on cases in India, we select the data of only India. Furthermore, we group our data based on the observation dates. Out of 285308 records of whole world we extract 459 records which are of India.

The data pre-processing in our project consists of the following process:

- Data Standardization
- Data Cleaning
 - Handling missing data
 - Handling null values
 - Handling duplicate values
 - Managing the outliers

Data standardization:

Data standardization is also a vital data pre-processing technique as it helps in eliminating or removing the inconsistent data. The process of data standardization helps in making the consistent internally. It is handy in transforming attributes acquiring a Gaussian distribution which differ means standard deviation into a

Gaussian Standard Gaussian Distribution. In our project we standard our data in the dataset using the scikit-learn library.

Data cleaning:

Data cleaning is one in all necessary steps in the attaining good accuracy in machine learning. It is also an essential step in our project. Data cleaning, although having great importance is mostly a hidden step that is not mentioned by most of the professionals. It helps in uncovering and cleaning the missing values, outliers the inconsistent data which may interrupt in successful execution of the model. In general, if we are using a better data which is clean there is a chance of better results even when we use simple algorithm. Our dataset contains different attributes of different types. Hence it would require different methods for cleaning the dataset.

The data cleaning techniques we used in our project are elaborated as follows:

- **Handling missing data:** Missing data could be misleadingly tough issue in the field of machine learning. These may lead to miss interpretation and wrong results. We cannot simply remove the record in order to overcome this issue. We need to identify a method that would helping in filling up these missing values so that the important records could not be missed. We deal this issue in two ways. The former is by removing the entry of the missing values and the latter is by filling it up based on the past observations. The former step can be performed if the data is less quality or unimportant. This method may not be recommended as the information which is important would be lost. The latter is inserting the missing values using appropriate statistical methods. We can inert values in the missing columns by finding out the mean and then filling the column.
- **Handling null values:** The values also possess the same issue as the missing values possess the null values can be avoided by simply deleting the record or filling up the entry by appropriate statistical method. In our project, we used the mean of the attribute column or the previous entries in or to deal with null values. We used an appropriate method available from the library available in python to perform the statistical operations.
- **Removing the duplicates:** The presence of the duplicate entries the dataset would have a significant effect on our model or any other machine learning

model. Hence the issue of duplicates must be resolved in order to successful execution of the model. In our project, we used the drop duplicates method in order to remove the duplicates in our dataset. This method is elaborate in the upcoming sections.

- Managing the outliers: The outliers also possess a challenge in developing a machine-learning model. For example, we take the presence of outliers in a particular model, the linear regression could be less fast or accurate than the decision-tree. It is not reasonable to have unlikely data or unreasonable data in the dataset. For example, we cannot have year 2017 entry in our dataset as it would be practically impossible and also has a negative effect to our model. Hence it is a better idea or approach to remove the outliers and work on the project. For our project, we also transformed and modified the data for improving our dataset.

Hence, we applied the methods available in the python library to study the data in the dataset so that we could identify any abnormal entry or anomalies in the data. Python libraries used have provided us with appropriate methods to visualize the data and detect the missing values and outliers and overcome the problems in our dataset.

5.1.3 Language Used for Implementation

We used Google Colab for the system's implementation. It's a cloud-based jupyter notebook environment that's free to use. Most significantly, it does not require any setup, and the notebooks you create can be modified by multiple team members at the same time, much like we do with Google Docs. Many common machine learning libraries are supported by Colab and can be quickly loaded into your notebook. This simplifies the debugging process while also improving error handling.

Some of the libraries used in our system are Pandas, Matplotlib, NumPy, Scikit-Learn and Seaborn. They provide various advantages for the process of machine learning. For example, the Pandas package for Python provides simple data structures with good speed as well as a data analysis tool.

5.2 Implementation

5.2.1 Data Preprocessing and grouping

The most important part of every machine learning based system is data pre-processing. There may be different kinds of data which has to be made homogeneous accordingly with respect to the technologies used. In the case of our model, the missing values, duplicate values and the null values are removed in the data pre-processing. The outliers are also removed also we select the data of particular region based on our requirement. As our dataset is World covid dataset it consists of covid cases records of whole world. But in our project, we are only focusing on Indian covid cases so during the preprocessing we need to extract the data of India. Also, as it is the timeseries data we need to group our data based on the observation dates. In the below code snippet we preprocess and group our data based on our requirements.

```
data=pd.read_csv("covid_19_data.csv")
# print(data.head())

data=data[data["Country/Region"]=="India"]

data=preprocess_data(data) # Group confirmed cases, Recovered, Death cases data in the form of sorted order according to dates


def preprocess_data(data):

    data=data.loc[:,["ObservationDate","Confirmed","Recovered","Deaths"]]

    # print(type(data),data)

    grouped_data=data.groupby(["ObservationDate"]).sum()

    grouped_data["date"]=grouped_data.index

    print("grouped_data ",grouped_data,sep="\n")

    b=grouped_data.reset_index(drop=True)

    b["Date"] = pd.to_datetime(b["date"])

    b=b.sort_values(by="Date")
```

Figure 5.2.2.1 Grouping of data

After grouping the data based on the observation dates, we convert each columns into

lists like date_list for observation dates, confirmed_list for confirmed covid cases etc. As we don't have the active cases attribute in the dataset, we calculate it by removing cured cases and the death cases from the list of confirmed cases.

```
def getAllLists(data):  
    global cured_list, future_confirmed_list  
    b = data  
    date_list = b.date.values  
    confirmed_list = b.Confirmed.values  
    cured_list = b.Recovered.values  
    death_list = b.Deaths.values  
    active_cases = []  
  
    for i in range(len(cured_list)):  
        active_cases.append(confirmed_list[i] - cured_list[i] - death_list[i])  
  
    return np.array(date_list), np.array(confirmed_list), np.array(death_list), np.array(cured_list), np.array(active_cases)
```

Figure 5.2.2.2 Get all lists

5.2.2 Data Preparing

LSTM series must first be prepared before it can be modelled. The LSTM model will train a function that transfers a set of previous observations as input to a new observation as output. As a result, the series of observations must be converted into several examples for the LSTM to learn from.

Take the following univariate sequence as an example: [10, 20, 30, 40, 50, 60, 70, 80, 90].

For the one-step prediction that is being learnt, we can partition the sequence into numerous input/output patterns called samples, where three time steps are used as input and one time step is used as output.

1	X	y
2	[10, 20, 30]	40
3	[20, 30, 40]	50
4	[30, 40, 50]	60
5	...	

```
def prepare_data(seq,n_steps):
    x=list()
    y=list()
    for i in range(len(seq)):
        end_idx=i+n_steps
        if end_idx>=len(seq):
            break
        seq_x,seq_y=seq[i:end_idx],seq[end_idx]
        x.append(seq_x)
        y.append(seq_y)
    return np.array(x),np.array(y)
```

Figure 5.2.3.1 Code snippet for data preparing

5.2.3 Splitting the Data

The main part in the prediction models is first we need to separate our data into training and testing before loading the model. Then we make use this training data for training the model and testing data for testing the model. During the testing of the model, we obtain the predicted output and actual output which will be later used for calculating the accuracy of the model. In our project we are taking 80% of input data for training and 20% for testing.

```
x_train,x_test,y_train,y_test=train_test_split(confirmed_x,confirmed_y,test_size=0.2) #splitting into training and testing data

X_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)

X_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)
```

Figure 5.2.4.1 Code snippet for splitting of data

5.2.4 Loading Model (LSTM)

A simple Long Short-Term Memory (LSTM) model comprising an input layer, a single hidden layer, and an output layer that makes a prediction has been built. Some neurons in the input layer are equal to 5 sequence steps. An LSTM layer with 10 hidden units (neurons) and a rectified linear unit (ReLU) as an activation function makes up the hidden layer. There was a dense layer with 1 unit fo in the output layer. The testing rate is set to 0.2, with a five-epoch decay. Furthermore, the number of epochs was 300, Adam was the optimizer, and the loss function was the mean square error. The model was then fitted to the data in order to make a forecast. Because of the stochastic nature of the LSTM model, the generated results may vary; as a result, we ran it numerous times. Finally, we enter the last sequence with output to forecast the next value in the series.

```
def loadModel(x_train,y_train):
    # model=Sequential()
    # model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(n_steps,1)))
    # model.add(LSTM(50, activation='relu'))
    # model.add(Dense(1))
    # model.compile(optimizer='adam', loss='mse')
    # model.fit(x_train, y_train, epochs=300, verbose=1)
    model = keras.models.load_model('saved_model')
    return model
```

Figure 5.2.5.1 Code snippet for loading lstm model

5.2.5 Prediction

Our project's main goal is to forecast future verified covid instances. After loading the model, we use our data to train and test it. Then we used our trained model for predicting the unknown cases.

```
def predict(X,n_steps,model):  
  
    global future_confirmed_list,num_days  
  
    last_row=X[X.shape[0]-1]  
  
    last_date=date(2021,5,2)  
    curr_date=date.today()  
    num_days=(curr_date-last_date).days  
    # print("num of days",num_days)  
  
    tmp_list=list(last_row)  
    future_confirmed_list=list()  
  
    for i in range(num_days):  
        x_input=np.asarray(tmp_list,dtype="object").astype(np.float32).reshape(1,n_steps,1)  
        # print(x_input)  
        yhat=model.predict(x_input,verbose=0)  
        # print(yhat,type(yhat))  
        tmp_list.append(yhat[0][0])  
        tmp_list=tmp_list[1:]  
        #temp list is used to store the next input and output for this input will be predicted  
        # print(tmp_list)  
        future_confirmed_list.append(yhat[0][0])  
  
    return future_confirmed_list
```

Figure 5.2.6.1 Code snippet for the prediction

5.2.6 Accuracy

Using the predicted output and the actual output we find the accuracy of our model. For this process we take the percentage differences of two values each from the predicted output and from the actual output, after getting the percentage difference of all index values we calculate the average of these percentages thus we get the accuracy of the model. In our project LSTM model gives the accuracy of around 88% which proves that It outperforms other networks and hence could be a good choice for predicting the number of COVID–19 patients in the future.

```
def displayAccuracy(pred,actual):  
    n=len(pred)  
    #print("length: ",n)  
    sum=0  
    for i in range(n):  
        x=pred[i]  
        y=actual[i]  
        diff=abs(x-y)  
        sum+=((y-diff)/y)*100  
    #print("n: ",n," sum: ",sum)  
    sum/=n  
    return sum
```

Figure 5.2.7.1 Code snippet for the accuracy of the model

5.2.7 Graph Plot

Finally, we plot the graph in which x-axis shows the dates and y-axis shows the future confirmed case. By observing the graph we get an insight on variation on cases in future.

```
plt.plot(future_dates_list,future_confirmed_list) ## Try to concatenate old dates with future_dates_list and plot a graph  
plt.xlabel("Dates")  
plt.ylabel("Total number of confirmed cases")  
plt.show()
```

Figure 5.2.8.1 Code snippet for the graph

5.3 Result

In our project, we have tried to use the LSTM algorithm having accuracy greater than the existing system to predict the future covid cases. The data pre-processing and data grouping helps in furnishing the dataset and helps in gaining better insights. After implementing the dataset on the machine learning model, we determine the accuracy of the model along with the prediction of future covid cases.

```
grouped_data
      Confirmed  Recovered   Deaths      date
ObservationDate
01/01/2021    10286709.0  9883461.0  148994.0  01/01/2021
01/02/2021    10323965.0  9927310.0  149435.0  01/02/2021
01/03/2021    10340469.0  9946867.0  149649.0  01/03/2021
01/04/2021    10356844.0  9975958.0  149850.0  01/04/2021
01/05/2021    10374932.0  9997272.0  150114.0  01/05/2021
...          ...          ...          ...
12/27/2020    10207871.0  9782669.0  147901.0  12/27/2020
12/28/2020    10224303.0  9807569.0  148153.0  12/28/2020
12/29/2020    10244852.0  9834141.0  148439.0  12/29/2020
12/30/2020    10266674.0  9860280.0  148738.0  12/30/2020
12/31/2020    10266674.0  9860280.0  148738.0  12/31/2020

[459 rows x 4 columns]
```

Fig 5.3.4 Input Details for Prediction of cases

This is our input dataset after pre-processing and grouping it based on the observation dates.

```

predicted output: [9.9078450e+06 1.9473791e+06 8.3479012e+05 1.7026664e+04 1.0843791e+07
4.7281990e+00 1.0959116e+07 9.3606900e+06 6.5361084e+03 2.5986108e+05
4.1203798e+06 1.1638695e+07 1.6715172e+05 9.8338730e+06 8.1753680e+06
6.1562300e+05 1.4064052e+06 4.2331948e+03 1.0713458e+07 8.9340940e+06
4.2104195e+06 1.2317354e+07 7.3412200e+06 1.0771965e+07 1.2586969e+06
5.3028091e+01 3.0623662e+05 1.0535939e+07 1.3056571e+06 9.4536040e+06
1.0289194e+07 5.9571025e+05 9.7976210e+06 1.0653563e+03 1.5735044e+07
1.1502496e+07 1.1294823e+07 1.1451840e+07 6.6614005e+06 4.2668047e+00
8.4603440e+06 4.2631066e+04 1.1306564e+07 1.5183792e+07 1.2716956e+07
4.9293793e+04 6.2089471e+02 4.5737160e+06 4.8668340e+06 1.1423461e+07
7.8519080e+06 5.8928590e+06 1.1823574e+07 3.5378574e+01 7.1190070e+06
3.3137580e+06 2.2004529e+03 5.0619297e+03 2.1277712e+06 1.1283145e+07
4.0097431e+05 4.4812470e+06 8.2652770e+06 5.6276680e+04 4.2668047e+00
1.9976543e+04 1.3224659e+05 1.1083917e+07 1.0123169e+04 4.2668047e+00
3.2722369e+05 1.0929007e+07 1.1030221e+07 1.2147646e+07 1.1565397e+07
5.2814273e+04 1.1200200e+07 1.2099883e+07 1.1176025e+07 1.8345919e+05
1.5816583e+02 4.1608362e+05 3.0422882e+06 1.3557859e+06 1.0652311e+07
7.4091240e+06 7.9606335e+06 1.1468427e+07 1.8992363e+02 1.8454384e+07
1.5292375e+05]
actual output [9.6772030e+06 2.0886110e+06 9.0675200e+05 2.1370000e+04 1.0512093e+07
2.8000000e+01 1.0610883e+07 9.1778400e+06 9.2050000e+03 2.7658300e+05
4.2804220e+06 1.1285561e+07 1.9060900e+05 9.6082110e+06 8.0402030e+06
6.7316500e+05 1.4831560e+06 6.7250000e+03 1.0395278e+07 8.7734790e+06
4.3701280e+06 1.2095855e+07 7.3070970e+06 1.0450284e+07 1.3856350e+06
8.2000000e+01 3.4309100e+05 1.0224303e+07 1.4356160e+06 9.2667050e+06
1.0004599e+07 6.4831500e+05 9.5715590e+06 1.9980000e+03 1.6263695e+07
1.1139516e+07 1.0925710e+07 1.1096731e+07 6.6850820e+06 3.0000000e+00
8.3138760e+06 5.2987000e+04 1.0937320e+07 1.5616130e+07 1.2589067e+07
5.9695000e+04 9.8700000e+02 4.7543560e+06 5.0203590e+06 1.1063491e+07
7.7613120e+06 5.9925320e+06 1.1514331e+07 4.3000000e+01 7.1205380e+06
3.4639720e+06 3.5880000e+03 7.5980000e+03 2.2686750e+06 1.0916589e+07
4.4021500e+05 4.6599840e+06 8.1371190e+06 6.7161000e+04 3.0000000e+00
2.4530000e+04 1.5079300e+05 1.0733130e+07 1.3430000e+04 3.0000000e+00
3.6694600e+05 1.0581823e+07 1.0676838e+07 1.1908910e+07 1.1210799e+07
6.2808000e+04 1.0838194e+07 1.1846652e+07 1.0814304e+07 2.0719100e+05
3.3000000e+02 4.5618300e+05 3.1673230e+06 1.4800730e+06 1.0340469e+07
7.3704680e+06 7.8648110e+06 1.1112241e+07 3.9600000e+02 1.9164969e+07
1.7349100e+05]

```

Accuracy: 88.78567028289827

Fig 5.3.2 Predicted and actual outputs, accuracy of the model

These are the actual cases and predicted cases which we get after testing our model. Using these values, we calculate the accuracy of our model.

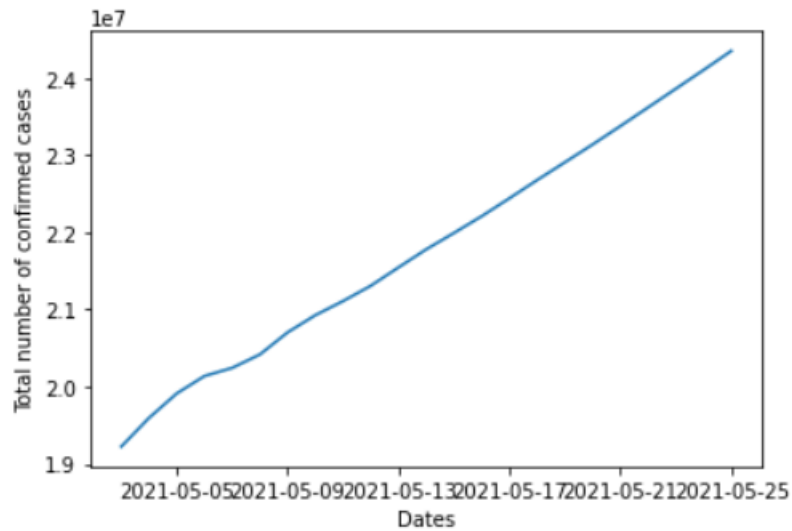


Fig 5.3.3 Output graph

As the graph is increasing exponentially indicating the second wave which involves a greater number of cases in 2021 this may be due insufficient social distance etc. By the first week of May the total number of covid effected people in India has crossed 2crores which indicates the high alarming situation so, the government should take particular measures like implementing lockdown in the states which have high positivity rate, bringing awareness among people, completing the vaccination and improving the healthcare system.

6. TESTING

6.1 Types of Testing

For our project, we have implemented Integration testing and Unit testing methodologies to test its working. In the case of unit testing, we tested every individual module in order to verify if the individual parts are correct. We have tested the individual internal code modules. In the case of Integration testing, we tested the code for each module and then integrated them based on the hierarchy to top and finally tested the entire code to verify whether it passed all of the test cases. While performing Integration testing, we identified few bugs in final code and then we modified the code and retested it again and again until it satisfied all test cases. Finally, we performed the black box testing to find whether our model is giving correct output for the input or not.

6.2 List of Test Cases

S.NO	Test Name	Input	Expected output	Actual Output	Test Case Result
1	Loading the dataset	Dataset	Dataset Loaded	Dataset read	PASS
2	Split dataset	Training and testing dataset	Splitting of the data into training and testing	Dividing data into training and testing	PASS
3	Training the model	Training set, parameters	Trained with the provided set	Trained model	PASS

4	Validation of the model	No of entries from testing data	Validation of the model with best fit	Model generated	PASS
5	Predicting the accuracy	Accuracy metrics	Predicted accuracy	Accuracy Predicted	PASS
6	Test Data	Test Column in data	Prediction for the given test case	Predicted result	PASS
7	Future Cases	Future dates	Giving the future covid cases records	Gave the future covid cases records	PASS
8	Graph Plotting	Future dates	Plotting the graph for dates vs future cases	Plotted the graph for dates vsfuture cases	PASS

Tab 6.2.1 List of Test Cases

7. CONCLUSION

The forecasting capability of the COVID-19 dataset was investigated using an LSTM network in this study. By providing a high accuracy rate, the LSTM network outperformed the other competing networks.

As a result, the LSTM network can be considered a good model for COVID–19 patient prediction.

7.2Future Work

Coronavirus, like the flu, is a contagious and infectious disease with specific growth patterns. These patterns are non-linear and dynamic in character. The data is dynamic in nature because the cases may vary depending on the seasons, population, and other factors. To effectively forecast the data, a deep learning model based on long short-term memory networks can be utilised. This could alleviate stress on health-care systems and administrations by allowing them to plan more effectively. As a result, the LSTM model could be a good contender for predicting the number of COVID–19 patients in the future.

8. BIBLIOGRAPHY

1. Coronavirus illness (COVID-19) epidemic, World Health Organization. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/index_en.html (accessed on May 14, 2020).
2. Niu, P.; Zhan, F.; Ma, X.; Wang, D.; Xu, W.; Wu, G.; Gao, G. F.; Tan, W.; Zhu, N.; Zhang, D.; Wang, W.; Li, X.; Yang, B.; Song, J.; Zhao, X.; Huang, B.; Shi, W.; Lu, R.; Niu, P.; Zhan, F In China, a new coronavirus has been discovered in pneumonia patients.
N. Engl. J. Med. 2020, 382, 727.
3. Paules, C. I...; Marston, H. D.; Fauci, A. S. Coronavirus infections more than just the common cold, JAMA 2020, and 323, 707.
4. Johns Hopkins University Centers for Systems Science and Engineering, Coronavirus COVID-19 Cases. <https://github.com/CSSEGISandData/COVIDQ3> (accessed on April 14, 2020).
5. <https://www.who.int/emergencies/disease/novelcoronavirus2019/global-research-on-novel-coronavirus-2019-ncov> (Accessed on May 14, 2020)
6. NCPERTeam. The epidemiological features of an outbreak of 2019 novel covid diseases (COVID-19) in China, China CDC Weekly 2020:, 41, 145.
7. W Jiang and HD. Schotten, "Deep Learning for Fading Channel Prediction," in IEEE Open Journal of the Communication's Society, vols. 1, pp. 320-332, 2020
8. Wikipedia, 2019-20 coronavirus severity. <https://en.wikipedia.org/wiki/2019-20coronavirusoutbreak> (accessed on May 14, 2020).
9. Connor, JT., Martin, RD., Atlas, L.E., "Recurrent Neural Networks and Robust Time Series Prediction", IEEE Transactions on Neural Networks Vol 5, Issue 3, Pages 240:-254 March, 1994
10. Sepp Hochreiterr and Jürgen Schmidhuber, Long ShortTerm Memory, Neural Computations Volume 9 ,Issue 8, November 15, p.1735:-1780, 1997
11. Sun's, Q., Jankhovic, M. V., Baully, L. Mougiakakous, SG. Predicting Blood Glucose levels With anLSTM and Bi:-LSTM Based Deep Neural Network.
12. Anuradha Toumar, Neeraj Gupta, "Prediction for the spread of COVID-19 in India,causes and efficiency of preventive measures", Science of the Total

Environment, Volume. 728, 2020

13. Global research on coronavirus spread (COVID-19)

<https://www.who.int/emergencies/diseases/novelcoronavirus2019/global::-researchs-on-novel-coronavirus-2019>. (accessed on May 14, 2020)

APPENDIX-A: Python Modules

Python:

Python is a high-level, general-purpose programming language that is interpreted. Python's design philosophy, which makes extensive use of enticing indentation, emphasises code readability. Its language elements and object-oriented approach are designed to help programmers produce clear, logical code for both small and large projects.

Python is garbage-collected and dynamically typed. It is compatible with a variety of programming languages, including structured, object-oriented, and functional programming.

Because of its extensive standard library, Python is often referred to as a "batteries included" language.

NumPy:

NumPy is a Python extension module that is occasionally built in the C language. It's a Python module for conducting various numerical computations and processing of multidimensional and single-dimensional array elements. NumPy arrays outperform regular Python arrays in terms of accuracy and speed.

It also has the ability to control a large amount of data and is useful for Matrix multiplication and data reshaping.

Pandas:

Pandas is a Python open-source toolkit that allows for high-performance data manipulation. It's built on top of the NumPy package, which means you'll need NumPy to run the Pandas. Pandas gets its name from the word Panel Data, which implies an Econometrics approach to multidimensional data. It was created by Wes McKinney in 2008 and is used for data analysis in Python. Python was used for data preparation before Pandas, but it only offered limited assistance for data analysis. As a result, Pandas arose, enhancing the possibilities of data analysis. It can conduct five amazing

stages required for data processing and analysis, namely load, modify, prepare, model, and analyse, regardless of the data's origin.

Sklearn:

Scikit-learn is undoubtedly Python's most helpful machine learning library.

Classification, regression, clustering, and dimensionality reduction are just a few of the techniques in the sklearn toolkit for machine learning and statistical modelling. Please keep in mind that sklearn is used to build machine learning models. It can't be used to read data, manipulate it, or summarise it. We have better libraries for that purpose (e.g. NumPy, Pandas etc.)

Scikit-learn provides with a lot of useful features. Algorithms for supervised learning: Consider any supervised machine learning algorithm you've heard of, and chances are it's in scikit-learn. From generalised linear models (such as linear regression) through Support Vector Machines (SVM), Decision Trees, and Bayesian approaches, scikit tool box includes them all. One of the key reasons for scikit-great learn's usability is the rapid development of machine learning techniques. I began using scikit to solve supervised learning challenges, and I would strongly advise anyone new to scikit learning to do so. Cross-validation: Using sklearn, you can check the correctness of supervised models on unrevealed data in a variety of ways. Unsupervised learning techniques: The offering includes a wide range of unsupervised learning algorithms, including clustering, unsupervised neural networks, factor analysis, and principal component analysis. Several datasets: This was quite helpful when learning scikit-learn. I learned SAS by working with several academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having these while learning a new library was extremely beneficial to learning. Extraction of features: For extracting features from photos and text, use Scikit-learn (e.g. Bag of words)

Matplotlib:

Matplotlib is a fantastic Python visualisation package for 2D array charts. Matplotlib is a multi-platform data visualisation package based on NumPy arrays and designed to operate with the SciPy stack as a whole. In the year 2002, John Hunter launched it.

One of the most significant benefits of visualisation is that it allows us to take a visual approach to large volumes of data in simple representations. Matplotlib includes a variety of graphs, including line, bar, scatter, histogram, and so on. Matplotlib includes a wide range of plots. Plots help you understand trends, patterns, and connections. They're frequently used to make decisions based on numerical data. In Python, there are thousands of libraries, and Matplotlib is one of the most powerful tools for data visualisation. Matplotlib aims to make simple things simple and difficult things feasible. With just a few lines of code, you can build plots, histograms, power spectra, bar charts, error charts, scatterplots, and more.

APPENDIX-B: UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a universal visual modelling language for specifying, visualising, constructing, and documenting a software system's work. It symbolises the judgments and understandings that must be made concerning the systems that must be built. It is used to comprehend, design, browse, configure, maintain, and regulate data related to such systems. All development approaches, lifecycle stages, application domains, and media are deemed to be compatible with it. The modelling language is used to bring together previous modelling skills and incorporate current software best practises into a new perspective.

Semantic concepts, notation, and guidelines are all covered by UML. It is divided into four sections: static, dynamic, environmental, and organisational. It is planned that interactive visual modelling tools with code generators and report writers will be used to assist it. The UML instructions do not specify a typical procedure; instead, they are designed to be useful in a repeating development process. It's designed to help with the majority of existing object-oriented development methods.

The UML understands information about a system's static structure and dynamic behaviour. A system is modelled as a set of interconnected elements that work together to execute a task that benefits the user. The static structure explains the types of items that are important to a system and how they are implemented, as well as the relationships between them. The dynamic behaviour of objects defines their evolution through time and how they communicate to attain their objectives.

Modeling a system from a variety of different but connected perspectives allows it to be understood for a variety of reasons.

In a complicated development environment, the UML also allows organisational structures for organising models into packages that enable software teams to divide big systems into usable portions, understand and handle dependencies across packages, and manage the classifying of model units. It is made of of structures for displaying implementation decisions and arranging run-time elements into components.

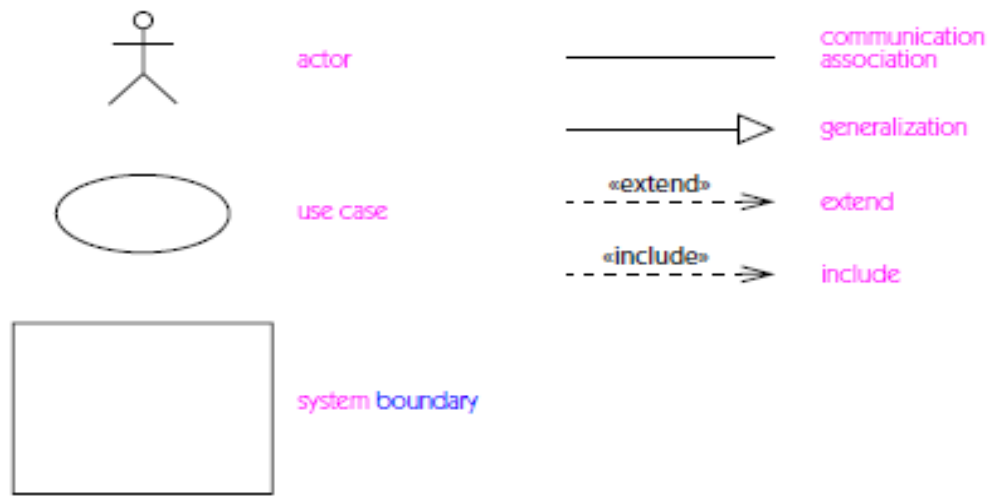
The Unified Modeling Language (UML) is not a programming language. Tools can

create code from UML in a variety of programming languages and construct reverse engineered models from existing programmes using code generators. The UML isn't a very formal language designed for proving theorems. There are a variety of such languages, but the most of them are difficult to comprehend or utilise purposes.

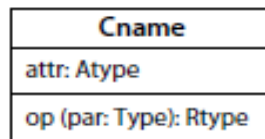
The UML is a modelling language that can be used for any purpose. A highly competent tool with a particular language might be appropriate for specialist fields like GUI layout, VLSI circuit design, or rule-based artificial intelligence. The Unified Modeling Language (UML) is a separate modelling language.

It's not meant to represent continuous systems like those found in engineering and physics. UML is intended to be a general-purpose modelling language for a variety of systems, including software, firmware, and digital logic.

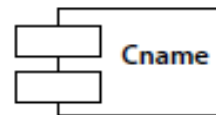
Icons on use case diagrams



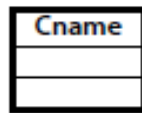
Icons on class, component, deployment, and collaboration diagrams



class



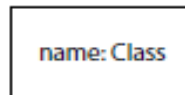
component



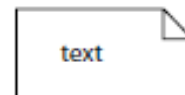
active class



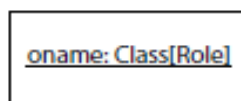
node



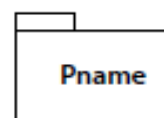
role



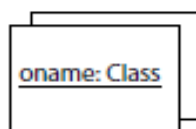
note



object



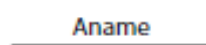
package



multiobject



interface



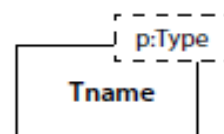
association



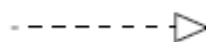
collaboration



generalization

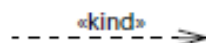


template
parameter



realization

template

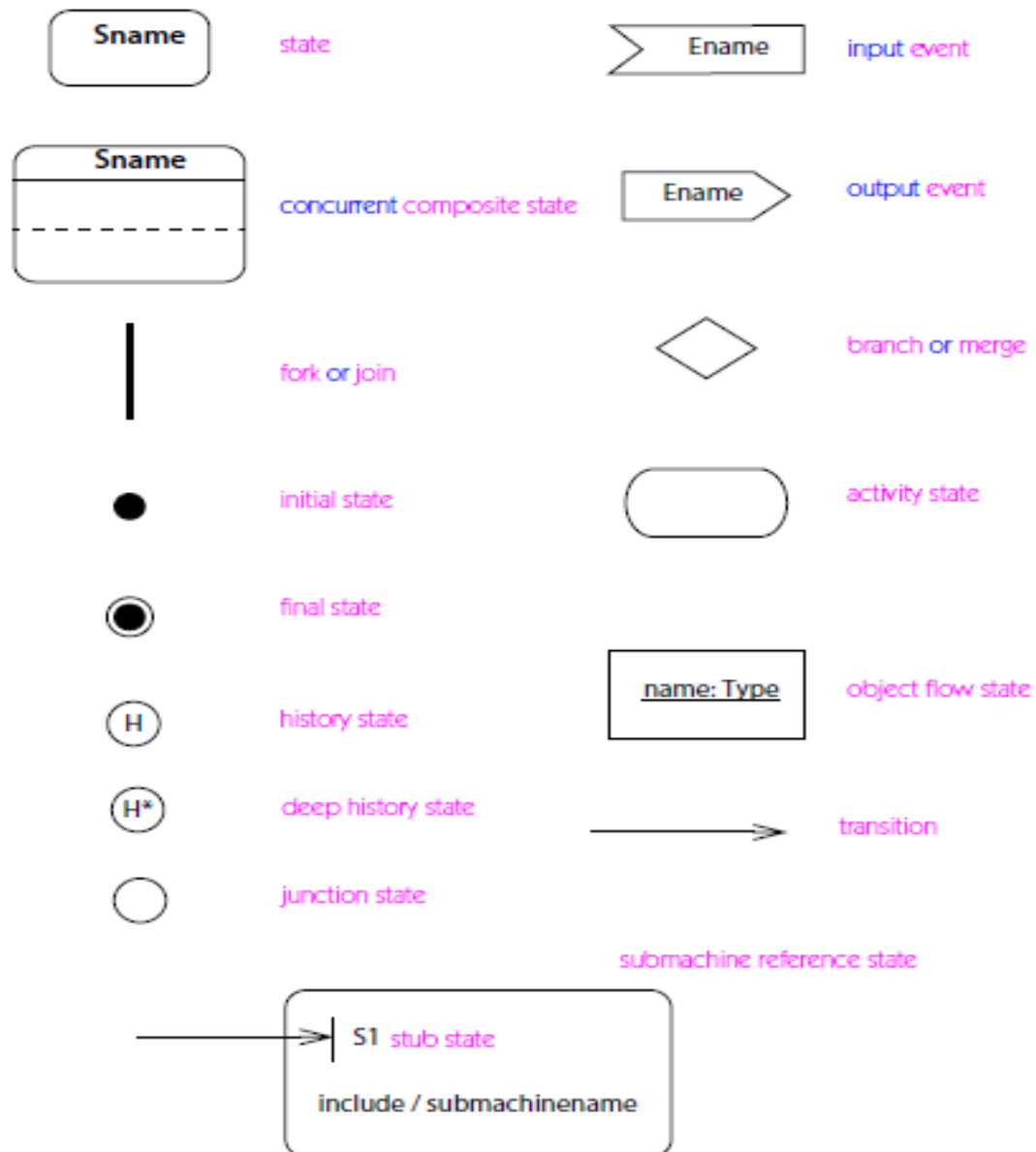


dependency



constraint

Icons on statechart and activity diagrams



C-16.pdf

ORIGINALITY REPORT

6%

SIMILARITY INDEX

3%

INTERNET SOURCES

2%

PUBLICATIONS

3%

STUDENT PAPERS

PRIMARY SOURCES

1

[sersc.org](https://www.sersc.org)

Internet Source

1%

2

Submitted to Gitam University

Student Paper

1%

3

Submitted to University of Asia Pacific

Student Paper

1%

4

ZHOU Leon Lee, WANG Zide, YANG Haotian.

"Quantitative Factors and Mathematical Modeling of CoVID-19 Pandemic Under Human Interventions", 2020 International Conference on Public Health and Data Science (ICPHDS), 2020

Publication

1%

5

"Web and Big Data", Springer Science and Business Media LLC, 2019

Publication

<1%

6

Submitted to Luton Sixth Form College, Bedfordshire

Student Paper

<1%

7

Submitted to University of Ruhuna Matara

Student Paper

		<1 %
8	scholarworks.bridgeport.edu Internet Source	<1 %
9	Submitted to KMD Computer Center Student Paper	<1 %
10	Submitted to Kingston University Student Paper	<1 %
11	pubmed.ncbi.nlm.nih.gov Internet Source	<1 %
12	docplayer.net Internet Source	<1 %
13	Submitted to AUT University Student Paper	<1 %
14	pt.scribd.com Internet Source	<1 %
15	www.mitpressjournals.org Internet Source	<1 %
16	www.umdaa.co Internet Source	<1 %
17	covid.yale.edu Internet Source	<1 %
18	en.wikipedia.org Internet Source	<1 %

19 Anuradha Tomar, Neeraj Gupta. "Prediction for the spread of COVID-19 in India and effectiveness of preventive measures", Science of The Total Environment, 2020
Publication

<1 %

20 Studies in Computational Intelligence, 2009.
Publication

<1 %

Exclude quotes On

Exclude matches

< 5 words

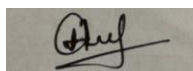
Exclude bibliography On

Team No:C20		Title	Guide Name
Roll No	Name		
17311A05D2	Badam Himabindu	A time series prediction model using Long Short Term Memory networks for prediction of Covid-19 data	Ms. J Neha
17311A05D4	Cheedu Sowmya		
17311A05F1	Mannem Nagalaxmi		

ABSTRACT

As of mid-May 2020, the current outbreak of Coronavirus disease 2019 (COVID-19), which is caused by the coronavirus 2 (SARS-CoV-2) that causes severe acute respiratory syndrome (SARS), has killed over 3,00,000 individuals and infected over 4.7 million people over the world. During this time, there have been over 1.8 million recoveries. It is critical for governments to be aware of the situation and to be able to forecast future patient numbers in order to maintain health-care readiness and plan for other essential steps. Using the Long-Sort Term Memory (LSTM) network as a driving force, a model for predicting the number of COVID–19 patients was constructed and subsequently used it for forecasting future cases. India's cases are taken into consideration. According to the findings, the LSTM network constructed in this paper outperforms other networks and hence could be a good contender for predicting the number of COVID–19 patients in the future.

Student 1



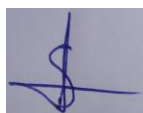
Guide

Ms. J Neha
Assistant Professor

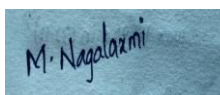
HOD

Dr. Aruna Varanasi
Department of CSE

Student 2




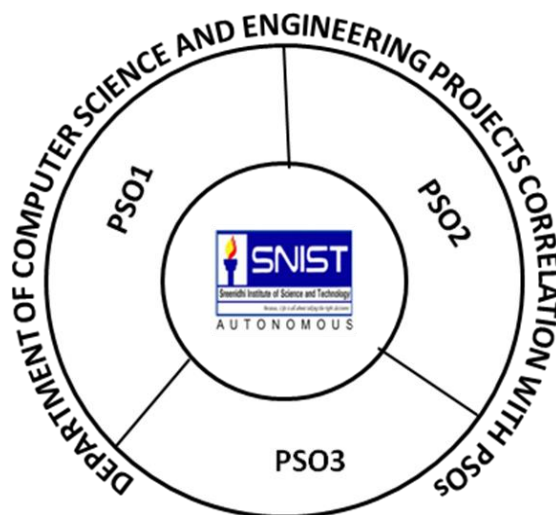
Student 3



Team No:C20		Title	Guide Name
Roll No	Name		
17311A05D2	Baddam HimaBindu	A time series prediction model using Long Short Term Memory networks for prediction of Covid-19 data	Ms. J Neha
17311A05D4	Cheedu Sowmya		
17311A05F1	Mannem Nagalaxmi		

Batch No.	Roll No.	Product/app	Ethics	research	social science	safety
C20	17311A05D2	L	M	✓	M	H
	17311A05D4					
	17311A05F1					

 SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Projects Correlation with POs											
PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
L	M	M	L	M	H	M	M	H	M	M	H



H **High**
M **Moderate**
L **Low**

Guide

Ms. J Neha
Assistant Professor

HOD

Dr. Aruna Varanasi
Department of CSE

