**A Project Report**
**on**

# COVID-19 Future Forecasting

**submitted in partial fulfillment of the requirements for the award of the degree**
**of**
**BACHELOR OF TECHNOLOGY**
**in**
**COMPUTER SICENCE AND ENGINEERING**

**by**


| | |
|---|---|
| **17WH1A0510** | **Ms. MULUMUDI NAVYA NIHITHA** |
| **18WH5A0507** | **Ms. JAGARLAMUDI MRUDULA** |
| **18WH5A0510** | **Ms. KOBBA JAGADISHWARI** |


**under the esteemed guidance of**
**Mr. Shivamurthy Hiremath**
**Assistant Professor**


**Department of Computer Science and Engineering**
**BVRIT HYDERABAD**
**College of Engineering for Women**
**(NBA Accredited – EEE, ECE, CSE and IT)**
**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**
**Bachupally, Hyderabad – 500090**


**MAY, 2021**

# DECLARATION

We hereby declare that the work presented in this project entitled **"COVID-19 Future Forecasting"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Mr. Shivamurthy Hiremath, Assistant Professor, Department of CSE.

Sign. with date:
**Ms. MULUMUDI NAVYA NIHITHA**
**(17WH1A0510)**

Sign. with date:
**Ms. JAGARLAMUDI MRUDULA**
**(18WH5A0507)**

Sign. with date:
**Ms. KOBBA JAGADISHWARI**
**(18WH5A0510)**

**BVRIT HYDERABAD**
**College of Engineering for Women**
**(NBA Accredited – EEE, ECE, CSE and IT)**
**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**
**Bachupally, Hyderabad – 500090**

**Department of Computer Science and Engineering**


## Certificate

This is to certify that the Project Work report on "**COVID-19 Future Forecasting**" is a bonafide work carried out by Ms. MULUMUDI NAVYA NIHITHA (17WH1A0510); Ms. JAGARLAMUDI MRUDULA (18WH5A0507); Ms. KOBBA JAGADISHWARI (18WH5A0510) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

**Head of the Department**
**Dr. Srinivasa Reddy Konda**
**Professor and HoD,**
**Department of CSE**

**Guide**
**Mr. Shivamurthy Hiremath**
**Assistant Professor**

**External Examiner**

## Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. Srinivasa Reddy Konda, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. Shivamurthy Hiremath, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

**Ms. MULUMUDI NAVYA NIHITHA**
**(17WH1A0510)**

**Ms. JAGARLAMUDI MRUDULA**
**(18WH5A0507)**

**Ms. KOBBA JAGADISHWARI**
**(18WH5A0510)**

# Contents

# ABSTRACT

Machine learning (ML) based forecasting mechanisms have proved their significance to anticipate in perioperative outcomes to improve the decision making on the future course of actions. The ML models have long been used in many application domains which needed the identification and prioritization of adverse factors for a threat. Several prediction methods are being popularly used to handle forecasting problems. This study demonstrates the capability of ML models to forecast the number of upcoming patients affected by COVID-19 which is presently considered as a potential threat to mankind. In particular, standard forecasting models, such as LightGBM (LGBM), least absolute shrinkage and selection operator (LASSO), and logistic regression. These have been used in this study to forecast the threatening factors of COVID-19. Three types of predictions are made by each of the models, such as the number of newly infected cases, the number of deaths, and the number of recoveries in the next 15 days. The results produced by the study proves it a promising mechanism to use these methods for the current scenario of the COVID-19 pandemic.

# LIST OF FIGURES

# 1. INTRODUCTION

Machine learning (ML) has proved itself as a prominent field of study over the last decade by solving many very complex and sophisticated real-world problems. The application areas included almost all the real-world domains such as healthcare, autonomous vehicle (AV), business applications, natural language processing (NLP), intelligent robots, gaming, climate modeling, voice, and image processing. ML algorithms' learning is typically based on trial and error method quite opposite of conventional algorithms, which follows the programming instructions based on decision statements like if-else. One of the most significant areas of ML is forecasting, numerous standard ML algorithms have been used in this area to guide the future course of actions needed in many application areas including weather forecasting, disease forecasting, stock market forecasting as well as disease prognosis. Various regression and neural network models have wide applicability in predicting the conditions of patients in the future with a specific disease. There are lots of studies performed for the prediction of different diseases using machine learning techniques such as coronary artery disease, cardiovascular disease prediction, and breast cancer prediction. In particular, the study is focused on live forecasting of COVID-19 confirmed cases and study is also focused on the forecast of COVID19 outbreak and early response. These prediction systems can be very helpful in decision making to handle the present scenario to guide early interventions to manage these diseases very effectively. This study aims to provide an early forecast model for the spread of novel coronavirus, also known as SARS-CoV-2, officially named as COVID-19 by the World Health Organization (WHO). COVID-19 is presently a very serious threat to human life all over the world. At the end of 2019, the virus was first identified in a city of China called Wuhan, when a large number of people developed symptoms like pneumonia. It has a diverse effect on the human body, including severe acute respiratory syndrome and multi-organ failure which can ultimately lead to death in a very short duration. Hundreds of thousands of people are affected by this pandemic throughout the world with thousands of deaths every coming day. Thousands of new people are reported to be positive every day from countries across the world. The virus spreads primarily through close person to person physical contacts, by respiratory droplets, or by touching the contaminated

surfaces. The most challenging aspect of its spread is that a person can possess the virus for many days without showing symptoms. The causes of its spread and considering its danger, almost all the countries have declared either partial or strict lockdowns throughout the affected regions and cities. Medical researchers throughout the globe are currently involved to discover an appropriate vaccine and medications for the disease. Since there is no approved medication till now for killing the virus so the governments of all countries are focusing on the precautions which can stop the spread. Out of all precautions, "be informed" about all the aspects of COVID-19 is considered extremely important. To contribute to this aspect of information, numerous researchers are studying the different dimensions of the pandemic and produce the results to help humanity. To contribute to the current human crisis our attempt in this study is to develop a forecasting system for COVID-19. The forecasting is done for the three important variables of the disease for the coming 10 days: 1) the number 0f New confirmed cases. 2) the number of death cases 3) the number of recoveries. This problem of forecasting has been considered as a regression problem in this study, so the study is based on some state-of-art supervised ML regression models such as LightGBM (LGBM), least absolute shrinkage and selection operator (LASSO), and logistic regression. The learning models have been trained using the COVID-19 patient stats dataset provided by Johns Hopkins. The dataset has been preprocessed and divided into two subsets: training set (85% records) and testing set (15% records). The performance evaluation has been done in terms of important measures including mean square error (MSE), mean absolute error (MAE), and root mean square error (RMSE). This study has some key findings which are listed below:

• Different ML algorithms seem to perform better in different class predictions.

• Most of the ML algorithms require an ample amount of data to predict the future, as the size of the dataset increases the model performances improve.

• ML model based forecasting can be very useful for decision-makers to contain pandemics like COVID-19.

## 1.1 OBJECTIVE

Machine learning (ML) based forecasting mechanisms have proved their significance to anticipate in perioperative outcomes to improve the decision making on the future course of actions. The ML models have long been used in many application domains which needed the identification and prioritization of adverse factors for a threat. Several prediction methods are being popularly used to handle forecasting problems. This study demonstrates the capability of ML models to forecast the number of upcoming patients affected by COVID-19 which is presently considered as a potential threat to mankind. In particular, four standard forecasting models, such as lightGBM (LGBM), least absolute shrinkage and selection operator (LASSO) and logistic regression have been used in this study to forecast the threatening factors of COVID-19.

## 1.2 ANALYSIS

### 1.2.1. Existing System :-

- The existing methods using most models assume a standard seir structure.
- Fraser and collegues to estimate size but make different changes on the nature of the different compartments and their respective residence times

### 1.2.2. Proposed System :-

- Here we proposed a forecasting system for COVID-19. The forecasting is done for the three important variables of the disease for the coming 10 days: 1) the number 0f New confirmed cases. 2) the number of death cases 3) the number of recoveries. This problem of forecasting has been considered as a regression problem in this study, so the study is based on some state-of-art supervised ML regression models such as LightGBM (LGBM), least absolute shrinkage and selection operator (LASSO) and logistic regression. The learning models have been trained using the COVID-19 patient stats dataset provided by Johns Hopkins. The dataset has been preprocessed and divided into two subsets: training set (85% records) and testing set (15% records). The performance evaluation has been done in terms of important measures including mean square error (MSE), mean absolute error (MAE), and root mean square error (RMSE)

**Advantages :**

- Different ML algorithms seem to perform better in different class predictions.
- Most of the ML algorithms require an ample amount of data to predict the future, as the size of the dataset increases the model performances improve.
- ML model based forecasting can be very useful for decision-makers to contain pandemics like COVID-19.

## 1.3 SCOPE

Hundreds of thousands of people are affected by this pandemic throughout the world with thousands of deaths every coming day. Thousands of new people are reported to be positive every day from countries across the world. The virus spreads primarily through close person to person physical contacts, by respiratory droplets, or by touching the contaminated surfaces. The most challenging aspect of its spread is that a person can possess the virus for many days without showing symptoms. The causes of its spread and considering its danger, almost all the countries have declared either partial or strict lockdowns throughout the affected regions and cities. Medical researchers throughout the globe are currently involved to discover an appropriate vaccine and medications for the disease. Since there is no approved medication till now for killing the virus so the governments of all countries are focusing on the precautions which can stop the spread. Out of all precautions, "be informed" about all the aspects of COVID-19 is considered extremely important. To contribute to this aspect of information, numerous researchers are studying the different dimensions of the pandemic and produce the results to help humanity

# 2. REQUIREMENT SPECIFICATION

## Hardware Requirements

This is an project so hardware plays an important role. Selection of hardware also plays an important role in existence and performance of any software. The size and capacity are main requirements.

Operating System supported by

- Windows 7,8,10
- Processor – Intel Core i3 and above
- RAM –2GB(RAM)

## Software Requirements

The software requirements specification is produced at the end of the analysis task. Software requirement is a difficult task, for developing the Application

- Python
- Machine Learning

## 2.1 INTRODUCTION OF TECHNOLOGIES USED

- **Python**

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

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

**Modules Used in Python :**

**Tensorflow**

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

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

**Numpy**

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

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

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code

- Useful linear algebra, Fourier transform, and random number capabilities
  Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

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

**Matplotlib**

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

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

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module

## 2.2 DATASET

The goal of this research is to forecast COVID-19 'potential distribution with an emphasis the number of positive new events, mortality and recoveries. This dataset has daily level information on the number of affected cases, deaths and recovery from 2019 novel coronavirus. Data from the Git Hub registry, supplied by Johns Hopkins University, Systems Science and Engineering Centre, were collected. The university mostly made the archive for the visual dashboard of the Novel Corona Virus in 2019 available and the ESRI Living Atlas Team helped it. A file called a Git center repository (cssecovid19) includes data collection files.

This is the link for the dataset:

https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 |
| 2 | | Afghanistan | 33.93911 | 67.709953 | 0 | 0 | 0 | 0 | 0 |
| 3 | | Albania | 41.1533 | 20.1683 | 0 | 0 | 0 | 0 | 0 |
| 4 | | Algeria | 28.0339 | 1.6596 | 0 | 0 | 0 | 0 | 0 |
| 5 | | Andorra | 42.5063 | 1.5218 | 0 | 0 | 0 | 0 | 0 |
| 6 | | Angola | -11.2027 | 17.8739 | 0 | 0 | 0 | 0 | 0 |
| 7 | | Antigua and Barbuda | 17.0608 | -61.7964 | 0 | 0 | 0 | 0 | 0 |
| 8 | | Argentina | -38.4161 | -63.6167 | 0 | 0 | 0 | 0 | 0 |
| 9 | | Armenia | 40.0691 | 45.0382 | 0 | 0 | 0 | 0 | 0 |
| 10 | Australian Capital Territory | Australia | -35.4735 | 149.0124 | 0 | 0 | 0 | 0 | 0 |
| 11 | New South Wales | Australia | -33.8688 | 151.2093 | 0 | 0 | 0 | 0 | 0 |
| 12 | Northern Territory | Australia | -12.4634 | 130.8456 | 0 | 0 | 0 | 0 | 0 |
| 13 | Queensland | Australia | -27.4698 | 153.0251 | 0 | 0 | 0 | 0 | 0 |
| 14 | South Australia | Australia | -34.9285 | 138.6007 | 0 | 0 | 0 | 0 | 0 |
| 15 | Tasmania | Australia | -42.8821 | 147.3272 | 0 | 0 | 0 | 0 | 0 |
| 16 | Victoria | Australia | -37.8136 | 144.9631 | 0 | 0 | 0 | 0 | 0 |
| 17 | Western Australia | Australia | -31.9505 | 115.8605 | 0 | 0 | 0 | 0 | 0 |

Fig 2.2.1 Recovered Dataset

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 |
| 2 | | Afghanistan | 33.93911 | 67.70995 | 0 | 0 | 0 | 0 | 0 |
| 3 | | Albania | 41.1533 | 20.1683 | 0 | 0 | 0 | 0 | 0 |
| 4 | | Algeria | 28.0339 | 1.6596 | 0 | 0 | 0 | 0 | 0 |
| 5 | | Andorra | 42.5063 | 1.5218 | 0 | 0 | 0 | 0 | 0 |
| 6 | | Angola | -11.2027 | 17.8739 | 0 | 0 | 0 | 0 | 0 |
| 7 | | Antigua and Barbuda | 17.0608 | -61.7964 | 0 | 0 | 0 | 0 | 0 |
| 8 | | Argentina | -38.4161 | -63.6167 | 0 | 0 | 0 | 0 | 0 |
| 9 | | Armenia | 40.0691 | 45.0382 | 0 | 0 | 0 | 0 | 0 |
| 10 | Australian Capital Territory | Australia | -35.4735 | 149.0124 | 0 | 0 | 0 | 0 | 0 |
| 11 | New South Wales | Australia | -33.8688 | 151.2093 | 0 | 0 | 0 | 0 | 0 |
| 12 | Northern Territory | Australia | -12.4634 | 130.8456 | 0 | 0 | 0 | 0 | 0 |
| 13 | Queensland | Australia | -27.4698 | 153.0251 | 0 | 0 | 0 | 0 | 0 |
| 14 | South Australia | Australia | -34.9285 | 138.6007 | 0 | 0 | 0 | 0 | 0 |
| 15 | Tasmania | Australia | -42.8821 | 147.3272 | 0 | 0 | 0 | 0 | 0 |
| 16 | Victoria | Australia | -37.8136 | 144.9631 | 0 | 0 | 0 | 0 | 0 |
| 17 | Western Australia | Australia | -31.9505 | 115.8605 | 0 | 0 | 0 | 0 | 0 |

Fig 2.2.2 Confirmed Dataset

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 |
| 2 | | Afghanistan | 33.93911 | 67.709953 | 0 | 0 | 0 | 0 | 0 |
| 3 | | Albania | 41.1533 | 20.1683 | 0 | 0 | 0 | 0 | 0 |
| 4 | | Algeria | 28.0339 | 1.6596 | 0 | 0 | 0 | 0 | 0 |
| 5 | | Andorra | 42.5063 | 1.5218 | 0 | 0 | 0 | 0 | 0 |
| 6 | | Angola | -11.2027 | 17.8739 | 0 | 0 | 0 | 0 | 0 |
| 7 | | Antigua and Barbuda | 17.0608 | -61.7964 | 0 | 0 | 0 | 0 | 0 |
| 8 | | Argentina | -38.4161 | -63.6167 | 0 | 0 | 0 | 0 | 0 |
| 9 | | Armenia | 40.0691 | 45.0382 | 0 | 0 | 0 | 0 | 0 |
| 10 | Australian Capital Territory | Australia | -35.4735 | 149.0124 | 0 | 0 | 0 | 0 | 0 |
| 11 | New South Wales | Australia | -33.8688 | 151.2093 | 0 | 0 | 0 | 0 | 3 |
| 12 | Northern Territory | Australia | -12.4634 | 130.8456 | 0 | 0 | 0 | 0 | 0 |
| 13 | Queensland | Australia | -27.4698 | 153.0251 | 0 | 0 | 0 | 0 | 0 |
| 14 | South Australia | Australia | -34.9285 | 138.6007 | 0 | 0 | 0 | 0 | 0 |
| 15 | Tasmania | Australia | -42.8821 | 147.3272 | 0 | 0 | 0 | 0 | 0 |
| 16 | Victoria | Australia | -37.8136 | 144.9631 | 0 | 0 | 0 | 0 | 1 |
| 17 | Western Australia | Australia | -31.9505 | 115.8605 | 0 | 0 | 0 | 0 | 0 |

Fig 2.2.3 Deaths Dataset

## 2.3 SUPERVISED MACHINE LEARNING MODELS

In the case that an undefined input instance is given, a supervised learning model can predict. Thus , data sets of input instances and its respective input instances for technique learning by the learning algorithms

The regressor is used for the regression model. A forecast for unpredictable entrants or test data is then generated in the qualified model. For the development of predictive models, Regression techniques and classification algorithms study method used here. In this COVID 19 prediction analysis , regression models of three are used:

- LGBM Regression

- LASSO Regression
- Logistic Regression

**A. LGBM Regression**

Light GBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as 'Light' because of its high speed**.** Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

**B. Lasso**

LASSO is a regression model that is part of a linear regression method that uses shrinkage. Shrinking means reducing the extreme data sample values to the key values in this case. This strengthens and stabilises LASSO and reduces the error by the shrinking process. For multi-linear situations, LASSO is considered a more fitting model. LASSO thus makes the regression smoother in terms of the amount of functions it uses. It uses a form of regularisation to penalise additional tasks automatically.

However, the LASSO regression tries one at a time, because it does not add importance of zero if the new function would not boost the penalty term's fit with that function. The

power of regularisation is therefore to automatically pick for us by adding the penalty for extra functions. Therefore, in this case of regularisation the models become sparse with few coefficients as the method removes values are zero. This regression LASSO acts to reduce the coefficient, which can be known by the square residual β slope), where, β slope is a concept of penalty.

Lasso solutions are quadratic programming problems, which are best solved with software (like Matlab). The goal of the algorithm is to minimize:

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Which is the same as minimizing the sum of squares with constraint $\Sigma\ |B_j \leq\ s$ ($\Sigma$ = summation notation). Some of the βs are shrunk to exactly zero, resulting in a regression model that's easier to interpret.

A tuning parameter, λ controls the strength of the L1 penalty. λ is basically the amount of shrinkage:

- When λ = 0, no parameters are eliminated. The estimate is equal to the one found with linear regression.
- As λ increases, more and more coefficients are set to zero and eliminated (theoretically, when λ = ∞, *all* coefficients are eliminated).
- As λ increases, bias increases.
- As λ decreases, variance increases.

If an intercept is included in the model, it is usually left unchanged.

**C.Logistic Regression**

Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multiclass problems.

Logistic regression is a linear classifier, so you'll use a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, also called the logit. The variables $b_0$, $b_1$, …, $b_r$ are the estimators of the regression coefficients, which are also called the predicted weights or just coefficients.

The logistic regression function $p(\mathbf{x})$ is the sigmoid function of $f(\mathbf{x})$: $p(\mathbf{x}) = 1 / (1 + \exp(-f(\mathbf{x}))$. As such, it's often close to either 0 or 1. The function $p(\mathbf{x})$ is often interpreted as the predicted probability that the output for a given $\mathbf{x}$ is equal to 1. Therefore, $1 - p(\mathbf{x})$ is the probability that the output is 0.

Logistic regression determines the best predicted weights $b_0$, $b_1$, …, $b_r$ such that the function $p(\mathbf{x})$ is as close as possible to all actual responses $y_i$, $i = 1, \ldots, n$, where $n$ is the number of observations. The process of calculating the best weights using available observations is called model training or fitting.

To get the best weights, you usually maximize the log-likelihood function (LLF) for all observations $i = 1, \ldots, n$. This method is called the maximum likelihood estimation and is represented by the equation $\text{LLF} = \Sigma_i(y_i \log(p(\mathbf{x}_i)) + (1 - y_i) \log(1 - p(\mathbf{x}_i)))$.

When $y_i = 0$, the LLF for the corresponding observation is equal to $\log(1 - p(\mathbf{x}_i))$. If $p(\mathbf{x}_i)$ is close to $y_i = 0$, then $\log(1 - p(\mathbf{x}_i))$ is close to 0. This is the result you want. If $p(\mathbf{x}_i)$ is far from 0, then $\log(1 - p(\mathbf{x}_i))$ drops significantly. You don't want that result because your goal is to obtain the maximum LLF. Similarly, when $y_i = 1$, the LLF for that observation is $y_i \log(p(\mathbf{x}_i))$. If $p(\mathbf{x}_i)$ is close to $y_i = 1$, then $\log(p(\mathbf{x}_i))$ is close to 0. If $p(\mathbf{x}_i)$ is far from 1, then $\log(p(\mathbf{x}_i))$ is a large negative number.

There are several mathematical approaches that will calculate the best weights that correspond to the maximum LLF.

Once you determine the best weights that define the function $p(\mathbf{x})$, you can get the predicted outputs $p(\mathbf{x}_i)$ for any given input $\mathbf{x}_i$. For each observation $i = 1, \ldots, n$, the predicted output is 1 if $p(\mathbf{x}_i) > 0.5$ and 0 otherwise. The threshold doesn't have to be 0.5, but it usually is. You might define a lower or higher value if that's more convenient for your situation.

There's one more important relationship between $p(\mathbf{x})$ and $f(\mathbf{x})$, which is that $\log(p(\mathbf{x}) / (1 - p(\mathbf{x}))) = f(\mathbf{x})$. This equality explains why $p(\mathbf{x})$ is the logit. It implies that $p(\mathbf{x}) = 0.5$ when $f(\mathbf{x}) = 0$ and that the predicted output is 1 if $f(\mathbf{x}) > 0$ and 0 otherwise.

## 2.4 EVALUATION PARAMETERS

The performance of each learning model is evaluated in the Mean Squared Error (MSE), Mean Absolute Error (MAE) and Root Means Square Error (RMSE). In this analysis, the performance of every learning model is evaluated.

**A. Mean absolute error (mae)**

This is the average of the test results between the model projections and the observed statistics of equal weight for all variations. The matrix range ranges from zero to endlessness, and fewer scores demonstrate the goodness of learning models, so it is also referred to as negative scores.

**B. Mean square error (mse)**

Another way of calculating regressive models output is a medium-square error. MSE takes and squares the regression line data points. Squared is significant if the negative sign of the value is omitted and greater weight is given to bigger variations. The lower the medium defect, the closer you find the better match. The better.

**C. Root mean square error (rmse)**

A standard deviation from the expected error may be set to RMSE. The source applies to a square failure. forecast error are also notorious as residue as the space among finest match lines and real data points. Thus, RMSE tests the best fitness of the actual data points. RMSE is For the MSE square root, this is the following error rate.

**ARIMA MODEL:-**

ARIMA is an acronym that stands for Auto Regressive Integrated Moving Average. It is a class of model that captures a suite of different standard temporal structures in time series data.

An ARIMA model is a class of statistical models 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 Auto Regressive Integrated Moving Average. It is a generalization of the simpler Auto Regressive Moving Average and adds the notion of integration.

This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

- **AR**: Auto regression. A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I**: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA**: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.
  Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA (p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

- **p**: The number of lag observations included in the model, also called the lag order.
- **d**: The number of times that the raw observations are differenced, also called the degree of differencing.
- **q**: The size of the moving average window, also called the order of moving average.

A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model.

A value of 0 can be used for a parameter, which indicates to not use that element of the model. This way, the ARIMA model can be configured to perform the function of an ARMA model, and even a simple AR, I, or MA model.

Adopting an ARIMA model for a time series assumes that the underlying process that generated the observations is an ARIMA process. This may seem obvious, but helps to motivate the need to confirm the assumptions of the model in the raw observations and in the residual errors of forecasts from the model.

The ARIMA model can be used to forecast future time steps.

We can use the predict() function on the ARIMA Results object to make predictions. It accepts the index of the time steps to make predictions as arguments. These indexes are relative to the start of the training dataset used to make predictions.
If we used 100 observations in the training dataset to fit the model, then the index of the next time step for making a prediction would be specified to the prediction function as start=101, end=101. This would return an array with one element containing the prediction.
We also would prefer the forecasted values to be in the original scale, in case we performed any differencing (d>0 when configuring the model). This can be specified by setting the typ argument to the value 'levels'*:* typ='levels'.
Alternately, we can avoid all of these specifications by using the forecast() function, which performs a one-step forecast using the model.

We can split the training dataset into train and test sets, use the train set to fit the model, and generate a prediction for each element on the test set.

# 3. DESIGN

## 3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

## 3.2 Architecture Diagram

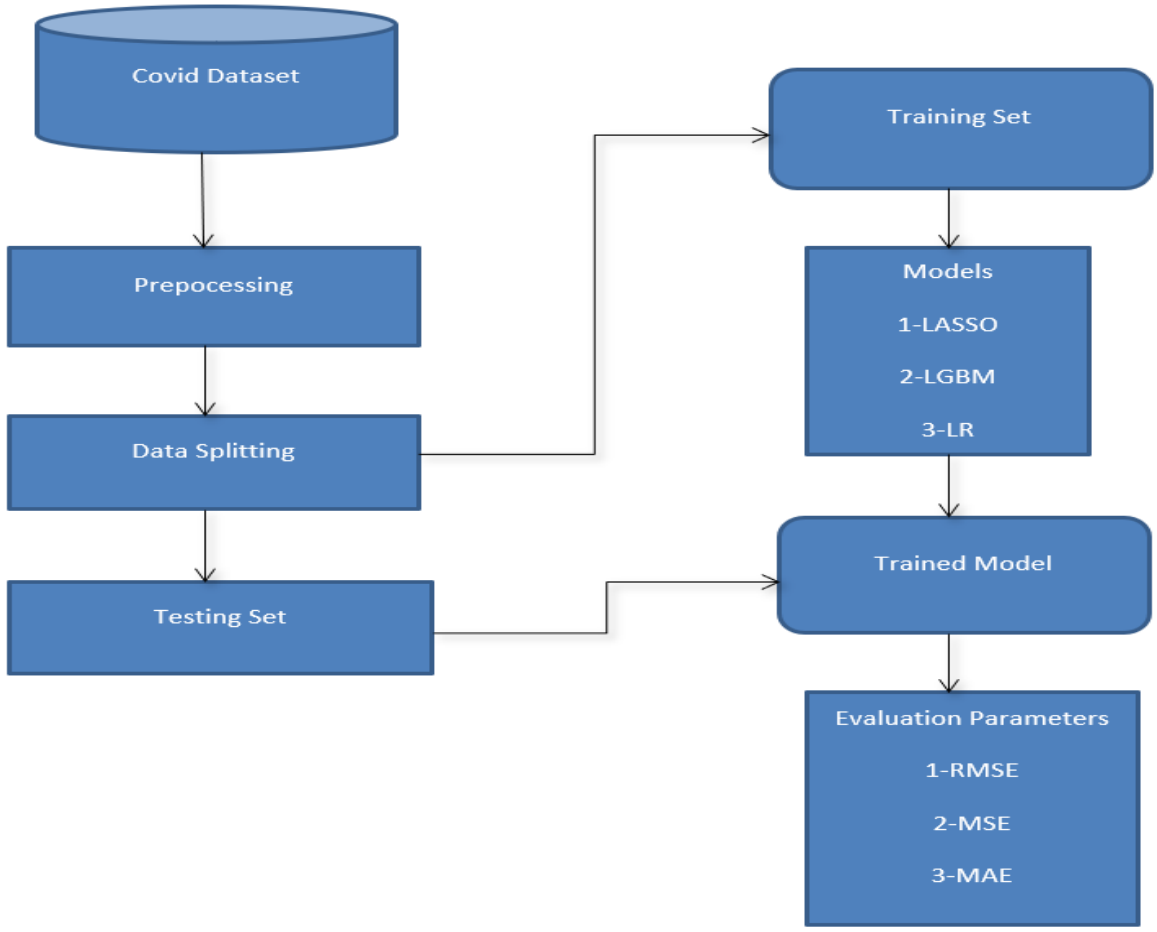


**Fig 3.2: Architecture Diagram**

## 3.3 UML DIAGRAMS

### 3.3.1 Use Case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The relationships between and among the actors and the use cases is described.

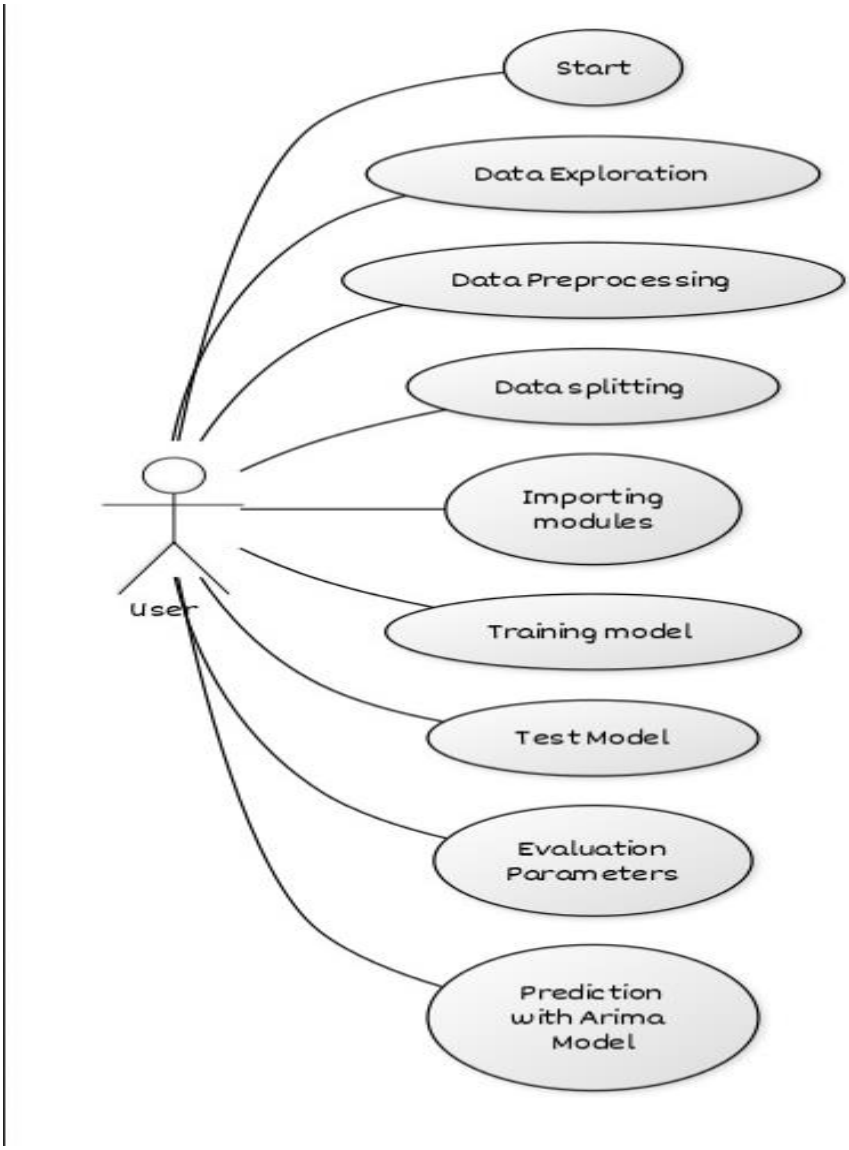


**Fig 3.3.1: Use Case Diagram**

### 3.3.2 Activity Diagram:

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.Activity diagram is basically a flowchart to represent the flow from one activity to another activity. Activity diagram has start point which is at the top of the diagram nothing but starting point of the diagram and aslo has end point at the bottom where the diagram ends. All the activities are connected to start point and also the end point..Activity diagram is basically a flowchart to represent the activity flow of the fields. The activity can be described as an operation of the system.

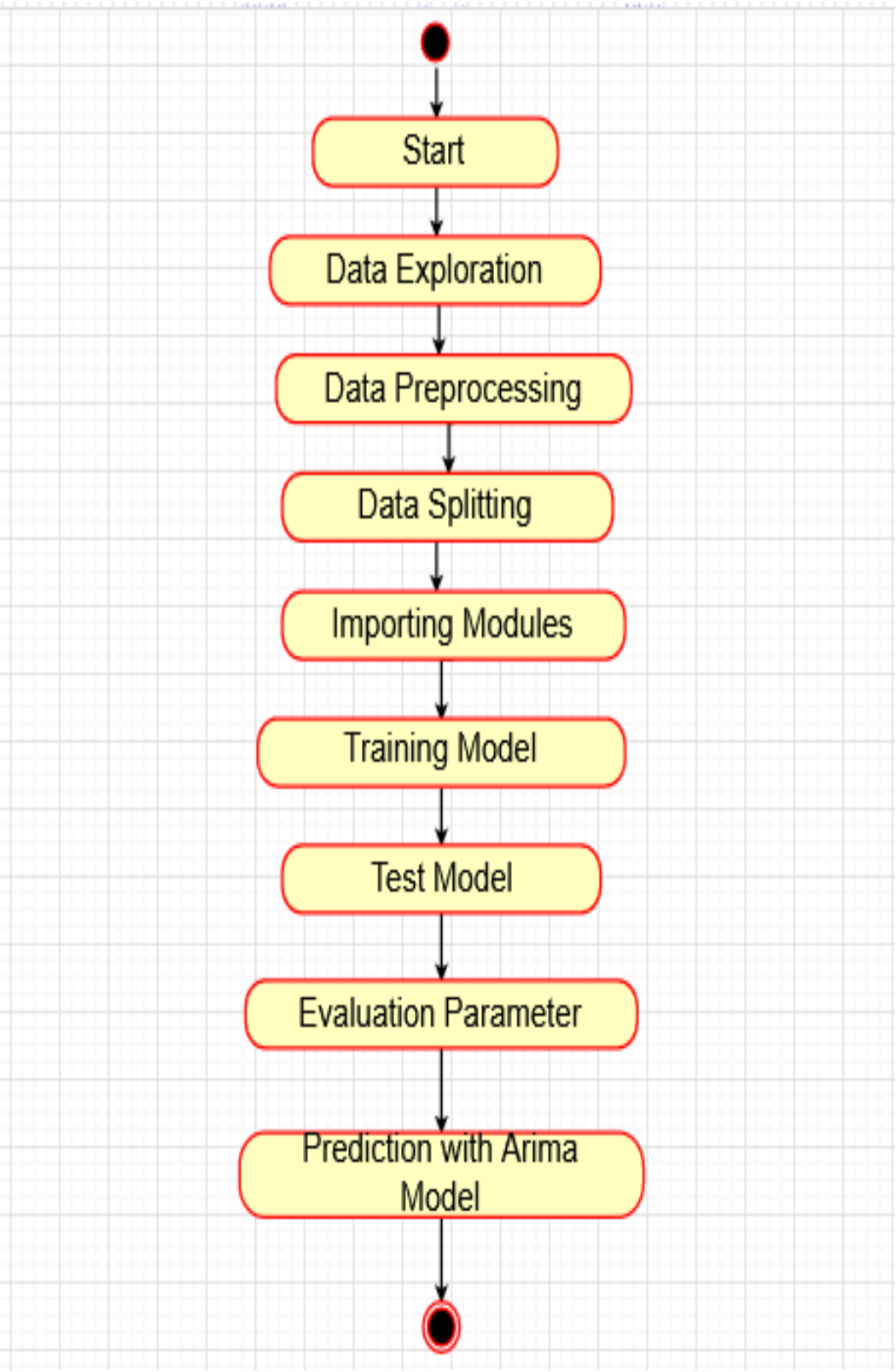


**Fig 3.3.2: Activity Diagram**

# 4. IMPLEMENTATION

## 4.1 Coding:

```
import pandas as pd
import numpy as np
import datetime
import requests
import warnings

import matplotlib.pyplot as plt
import matplotlib
import matplotlib.dates as mdates
import seaborn as sns


#Tensorflow for Multipplex Layer

import tensorflow
from keras.models import Sequential
from keras.layers import Dense

from lightgbm import LGBMRegressor

from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import OrdinalEncoder

from sklearn.model_selection import train_test_split

warnings.filterwarnings('ignore')
%matplotlib inline
from statsmodels.tsa.arima_model import ARIMA


confirmed_df = pd.read_csv('/content/drive/MyDrive/time_series_covid
19_confirmed_global.csv')
deaths_df = pd.read_csv('/content/drive/MyDrive/time_series_covid19_
deaths_global.csv')
recovered_df = pd.read_csv('/content/drive/MyDrive/time_series_covid
19_recovered_global.csv')
```

Description of Dataset Used:

```
confirmed_df.head()

deaths_df.head()

recovered_df.head()
```

**Comparing with Other Countries**

```
dates = list(confirmed_df.columns[4:])
dates = list(pd.to_datetime(dates))
dates_india = dates[8:]


df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

countries = ['China','US', 'Italy', 'Spain', 'France','India']

global_confirmed = []
global_recovered = []
global_deaths = []
global_active = []

for country in countries:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed.append(k.values.tolist()[0])

    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

plt.figure(figsize= (15,10))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Comparison with other Countries" , fontsize = 20)

for i in range(len(countries)):
    plt.plot_date(y= global_confirmed[i],x= dates_india,label = coun
tries[i],linestyle ='-')
plt.legend();


train = pd.read_csv('/content/drive/MyDrive/train.csv')
test = pd.read_csv('/content/drive/MyDrive/test.csv')
train['Date'] = pd.to_datetime(train['Date'])
test['Date'] = pd.to_datetime(test['Date'])

train['day'] = train['Date'].dt.day
```

```
train['month'] = train['Date'].dt.month
train['dayofweek'] = train['Date'].dt.dayofweek
train['dayofyear'] = train['Date'].dt.dayofyear
train['quarter'] = train['Date'].dt.quarter
train['weekofyear'] = train['Date'].dt.weekofyear
test['day'] = test['Date'].dt.day
test['month'] = test['Date'].dt.month
test['dayofweek'] = test['Date'].dt.dayofweek
test['dayofyear'] = test['Date'].dt.dayofyear
test['quarter'] = test['Date'].dt.quarter
test['weekofyear'] = test['Date'].dt.weekofyear
countries = list(train['Country_Region'].unique())
india_code = countries.index('India')
train = train.drop(['Date','Id'],1)
test =  test.drop(['Date'],1)

train.Province_State.fillna('NaN', inplace=True)
oe = OrdinalEncoder()
train[['Province_State','Country_Region']] = oe.fit_transform(train.
loc[:,['Province_State','Country_Region']])

test.Province_State.fillna('NaN', inplace=True)
oe = OrdinalEncoder()
test[['Province_State','Country_Region']] = oe.fit_transform(test.lo
c[:,['Province_State','Country_Region']])


columns = ['day','month','dayofweek','dayofyear','quarter','weekofye
ar','Province_State', 'Country_Region','ConfirmedCases','Fatalities'
]
test_columns = ['day','month','dayofweek','dayofyear','quarter','wee
kofyear','Province_State','Country_Region']
train = train[columns]
x = train.drop(['Fatalities','ConfirmedCases'], 1)
y = train['ConfirmedCases']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,ran
dom_state=0)
test = test[test_columns]
test_india = test[test['Country_Region'] == india_code]
```

LGBM Regression:

```
lgbm = LGBMRegressor(n_estimators=1300)
lgbm.fit(x_train,y_train)
pred = lgbm.predict(x_test)
lgbm_forecast = lgbm.predict(test_india)
mse=(round(mean_squared_error(pred, y_test),2))
```

```
mae=(round(mean_absolute_error(pred, y_test),2))
rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
print(mse)
print(mae)
print(rmse)
```

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(x_train,y_train)
pred = clf.predict(x_test)
LR_forecast = clf.predict(test_india)
mse=(round(mean_squared_error(pred, y_test),2))
mae=(round(mean_absolute_error(pred, y_test),2))
rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
print(mse)
print(mae)
print(rmse)
```

Lasso Regression:

```
from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit(x_train,y_train)
pred = clf.predict(x_test)
lasso_forecast = clf.predict(test_india)
mse=(round(mean_squared_error(pred, y_test),2))
mae=(round(mean_absolute_error(pred, y_test),2))
rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
print(mse)
print(mae)
print(rmse)
```

Tensorflow with Multiplex Layer:

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=
['accuracy'])

model.fit(x_train,y_train, epochs=5, batch_size=10)
```

```
_, accuracy = model.evaluate(x_train,y_train)
print('Accuracy: %.2f' % ((accuracy*30)*100))

pred = model.predict(x_test)

mse=(round(mean_squared_error(pred, y_test),2))
mae=(round(mean_absolute_error(pred, y_test),2))
rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
print(mse)
print(mae)
print(rmse)

df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
k = df1[df1['Country/Region']=='India'].loc[:,'1/22/20':]
india_confirmed = k.values.tolist()[0]
dates = list(confirmed_df.columns[4:])
dates = list(pd.to_datetime(dates))
data = pd.DataFrame(columns = ['ds','y'])
data['ds'] = dates
data['y'] = india_confirmed
```

Predicting up to march of 2021:

```
arima = ARIMA(data['y'], order=(5, 1, 0))
arima = arima.fit(trend='c', full_output=True, disp=True)
forecast = arima.forecast(steps= 30)
pred = list(forecast[0])

start_date = data['ds'].max()
prediction_dates = []
for i in range(30):
    date = start_date + datetime.timedelta(days=1)
    prediction_dates.append(date)
    start_date = date
plt.figure(figsize= (15,10))
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Predicted Values for the next 15 Days using Arima Model"
, fontsize = 20)

plt.plot_date(y= pred,x= prediction_dates,linestyle ='dashed',color
= '#ff9999',label = 'Predicted');
plt.plot_date(y=data['y'],x=data['ds'],linestyle = '-
',color = 'blue',label = 'Actual');
plt.legend();
```

Predicting upto may of 2021:

```
confirmed_df1 = pd.read_csv('/content/drive/MyDrive/time_series_covi
d_19_confirmed.csv')
deaths_df = pd.read_csv('/content/drive/MyDrive/time_series_covid_19
_deaths.csv')
recovered_df = pd.read_csv('/content/drive/MyDrive/time_series_covid
_19_recovered.csv')

df = confirmed_df1.groupby('Country/Region').sum().reset_index()
k = df[df['Country/Region']=='India'].loc[:,'1/22/20':]
india_confirmed = k.values.tolist()[0]
dates = list(confirmed_df1.columns[4:])
dates = list(pd.to_datetime(dates))
data = pd.DataFrame(columns = ['ds','y'])
data['ds'] = dates
data['y'] = india_confirmed


arima = ARIMA(data['y'], order=(5, 1, 0))
arima = arima.fit(trend='c', full_output=True, disp=True)
forecast = arima.forecast(steps= 30)
pred = list(forecast[0])

start_date = data['ds'].max()
prediction_dates = []
for i in range(30):
    date = start_date + datetime.timedelta(days=1)
    prediction_dates.append(date)
    start_date = date
plt.figure(figsize= (15,10))
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Predicted Values for the next 15 Days using Arima Model"
, fontsize = 20)

plt.plot_date(y= pred,x= prediction_dates,linestyle ='dashed',color
= '#ff9999',label = 'Predicted');
plt.plot_date(y=data['y'],x=data['ds'],linestyle = '-
',color = 'blue',label = 'Actual');
plt.legend();
```

## 4.2 OUTPUT SCREENSHOTS

```
import pandas as pd
import numpy as np
import datetime
import requests
import warnings

import matplotlib.pyplot as plt
import matplotlib
import matplotlib.dates as mdates
import seaborn as sns


#Tensorflow for Multipplex Layer

import tensorflow
from keras.models import Sequential
from keras.layers import Dense

from lightgbm import LGBMRegressor

from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import OrdinalEncoder

from sklearn.model_selection import train_test_split

warnings.filterwarnings('ignore')
%matplotlib inline
from statsmodels.tsa.arima_model import ARIMA


confirmed_df = pd.read_csv('/content/drive/MyDrive/time_series_covid19_confirmed_global.csv')
deaths_df = pd.read_csv('/content/drive/MyDrive/time_series_covid19_deaths_global.csv')
recovered_df = pd.read_csv('/content/drive/MyDrive/time_series_covid19_recovered_global.csv')
```

Fig 4.2.1 Imports

Pandas **head()** method is used to return top n (5 by default) rows of a data frame or series.

Which shows the 5 Rows and 369 Colums for the each of the Datasets.

| | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 | 1/27/20 | 1/28/20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | Afghanistan | 33.93911 | 67.709953 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | NaN | Albania | 41.15330 | 20.168300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | NaN | Algeria | 28.03390 | 1.659600 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | NaN | Andorra | 42.50630 | 1.521800 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | NaN | Angola | -11.20270 | 17.873900 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 369 columns

```
[ ] deaths_df.head()
```

| | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 | 1/27/20 | 1/28/20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | Afghanistan | 33.93911 | 67.709953 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | NaN | Albania | 41.15330 | 20.168300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | NaN | Algeria | 28.03390 | 1.659600 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | NaN | Andorra | 42.50630 | 1.521800 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | NaN | Angola | -11.20270 | 17.873900 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
[ ] recovered_df.head()
```

| | Province/State | Country/Region | Lat | Long | 1/22/20 | 1/23/20 | 1/24/20 | 1/25/20 | 1/26/20 | 1/27/20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | Afghanistan | 33.93911 | 67.709953 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | NaN | Albania | 41.15330 | 20.168300 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | NaN | Algeria | 28.03390 | 1.659600 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | NaN | Andorra | 42.50630 | 1.521800 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | NaN | Angola | -11.20270 | 17.873900 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 369 columns

Fig 4.2.2 Head of Dataset

## Comparing with Other Countries

```
[ ] dates = list(confirmed_df.columns[4:])
    dates = list(pd.to_datetime(dates))
    dates_india = dates[8:]
```

```
df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

countries = ['China','US', 'Italy', 'Spain', 'France','India']

global_confirmed = []
global_recovered = []
global_deaths = []
global_active = []

for country in countries:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed.append(k.values.tolist()[0])

    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

plt.figure(figsize= (15,10))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Comparison with other Countries" , fontsize = 20)

for i in range(len(countries)):
    plt.plot_date(y= global_confirmed[i],x= dates_india,label = countries[i],linestyle ='-')
plt.legend();
```

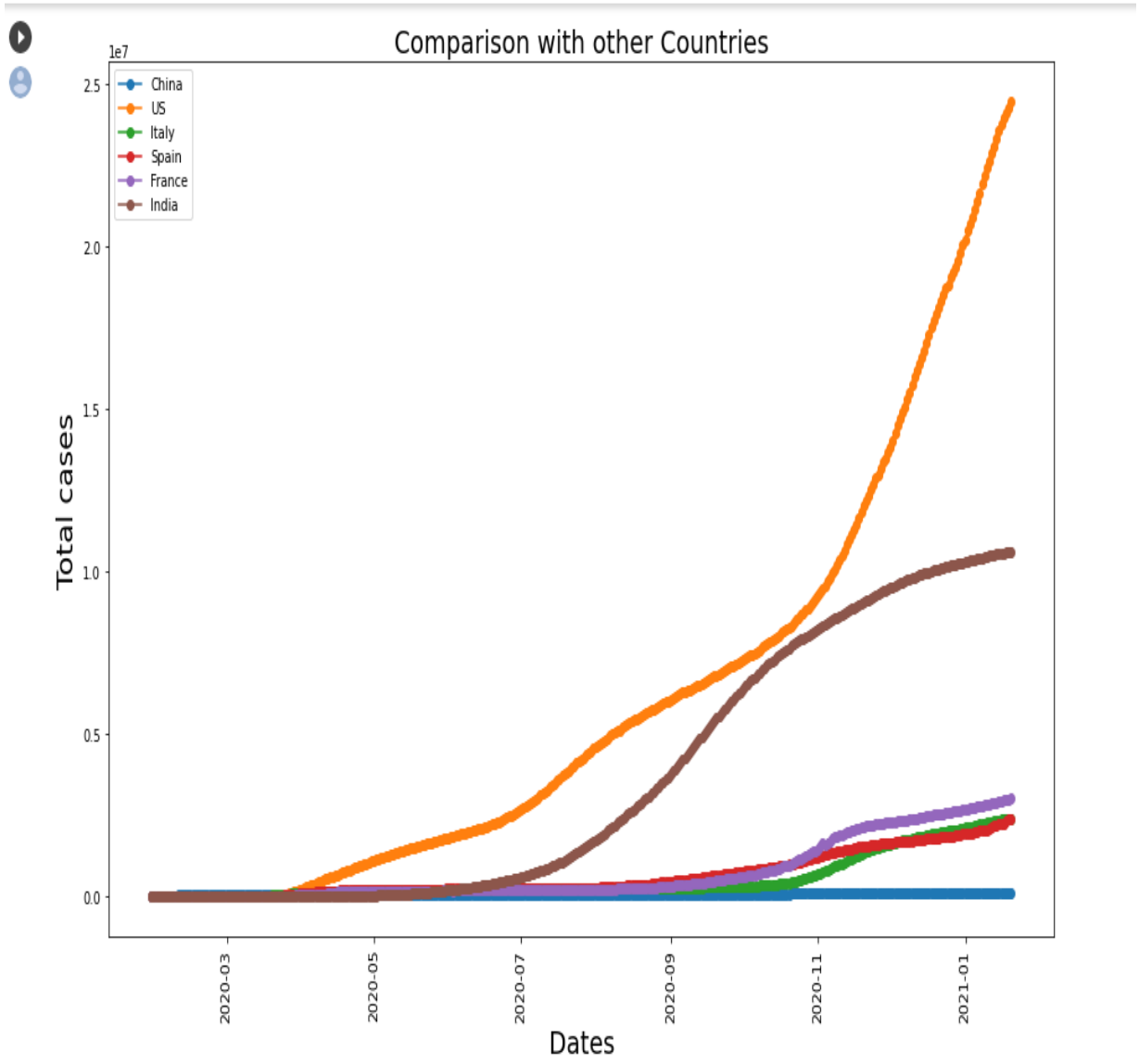


Fig 4.2.3 Comparison

Train/Test is a method to measure the accuracy of the model.

It is called Train/Test because we split the the data set into two sets: a training set and a testing set. We train the model using the training set. We test the model using the testing set.

80% for training, and 20% for testing. Train the model means create the model.
Test the model means test the accuracy of the model.
Here now we are Loading the Test and Train Data and Converting them to date and time Format also Analyzing the Parameters.

```
[ ] train = pd.read_csv('/content/drive/MyDrive/train.csv')
    test = pd.read_csv('/content/drive/MyDrive/test.csv')
    train['Date'] = pd.to_datetime(train['Date'])
    test['Date'] = pd.to_datetime(test['Date'])
```

```
[ ] train['day'] = train['Date'].dt.day
    train['month'] = train['Date'].dt.month
    train['dayofweek'] = train['Date'].dt.dayofweek
    train['dayofyear'] = train['Date'].dt.dayofyear
    train['quarter'] = train['Date'].dt.quarter
    train['weekofyear'] = train['Date'].dt.weekofyear
    test['day'] = test['Date'].dt.day
    test['month'] = test['Date'].dt.month
    test['dayofweek'] = test['Date'].dt.dayofweek
    test['dayofyear'] = test['Date'].dt.dayofyear
    test['quarter'] = test['Date'].dt.quarter
    test['weekofyear'] = test['Date'].dt.weekofyear
    countries = list(train['Country_Region'].unique())
    india_code = countries.index('India')
    train = train.drop(['Date','Id'],1)
    test =  test.drop(['Date'],1)

    train.Province_State.fillna('NaN', inplace=True)
    oe = OrdinalEncoder()
    train[['Province_State','Country_Region']] = oe.fit_transform(train.loc[:,['Province_State','Country_Region']])

    test.Province_State.fillna('NaN', inplace=True)
    oe = OrdinalEncoder()
    test[['Province_State','Country_Region']] = oe.fit_transform(test.loc[:,['Province_State','Country_Region']])
```

Training to the test set is a type of overfitting where a model is prepared that intentionally achieves good performance on a given test set at the expense of increased generalization error.

It is a type of over fitting that is common in machine learning  where a complete training dataset is provided and where only the input portion of a test set is provided. One approach to training to the test set involves constructing a training set that most resembles the test set and then using it as the basis for training a model. The model is expected to have better performance on the test set, but most likely worse performance on the training dataset and on any new data in the future.

Although overfitting the test set is not desirable, it can be interesting to explore as a thought experiment and provide more insight into both machine learning and avoiding overfitting generally.

```
[ ] columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear','Province_State', 'Country_Region','ConfirmedCases','Fatalities']
    test_columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear','Province_State','Country_Region']
    train = train[columns]
    x = train.drop(['Fatalities','ConfirmedCases'], 1)
    y = train['ConfirmedCases']
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
    test = test[test_columns]
    test_india = test[test['Country_Region'] == india_code]
```

Next, we will train the model on this new dataset and evaluate it on the test set to get the Mse, Mae and Rmse Values.

**Using LGBM  Regression**

LGBM Regression

```
[ ]  lgbm = LGBMRegressor(n_estimators=1300)
     lgbm.fit(x_train,y_train)
     pred = lgbm.predict(x_test)
     lgbm_forecast = lgbm.predict(test_india)
     mse=(round(mean_squared_error(pred, y_test),2))
     mae=(round(mean_absolute_error(pred, y_test),2))
     rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
     print(mse)
     print(mae)
     print(rmse)

     1777777.79
     441.5
     1333.33
```

Fig 4.2.4 Lgbm Regression

**Using Logistic Regression**

Logistic Regression

```
[ ]  from sklearn.linear_model import LogisticRegression
     clf = LogisticRegression(random_state=0)
     clf.fit(x_train,y_train)
     pred = clf.predict(x_test)
     LR_forecast = clf.predict(test_india)
     mse=(round(mean_squared_error(pred, y_test),2))
     mae=(round(mean_absolute_error(pred, y_test),2))
     rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
     print(mse)
     print(mae)
     print(rmse)
```

```
317892450.02
3415.65
17829.54
```

Fig 4.2.5 Logistic Regression

**Using Lasso Regression**

Lasso Regression

```
[ ]  from sklearn import linear_model
     clf = linear_model.Lasso(alpha=0.1)
     clf.fit(x_train,y_train)
     pred = clf.predict(x_test)
     lasso_forecast = clf.predict(test_india)
     mse=(round(mean_squared_error(pred, y_test),2))
     mae=(round(mean_absolute_error(pred, y_test),2))
     rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
     print(mse)
     print(mae)
     print(rmse)
```

```
290738263.65
5835.93
17051.05
```

Fig 4.2.6 Lasso Regression

Using TensorFlow With Multiplex Layer To get the Accuary

Tensorflow with Multiplex Layer

```
[ ] model = Sequential()
    model.add(Dense(12, input_dim=8, activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # compile the keras model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
[ ] model.fit(x_train,y_train, epochs=5, batch_size=10)
```

```
Epoch 1/5
2880/2880 [==============================] - 4s 1ms/step - loss: -42332207.1103 - accuracy: 0.0306
Epoch 2/5
2880/2880 [==============================] - 3s 1ms/step - loss: -1069762997.4925 - accuracy: 0.0297
Epoch 3/5
2880/2880 [==============================] - 3s 1ms/step - loss: -4374294629.5647 - accuracy: 0.0317
Epoch 4/5
2880/2880 [==============================] - 3s 1ms/step - loss: -11014226257.1552 - accuracy: 0.0330
Epoch 5/5
2880/2880 [==============================] - 3s 1ms/step - loss: -20516160398.6171 - accuracy: 0.0308
<tensorflow.python.keras.callbacks.History at 0x7f28339cd7d0>
```

```
[ ] _, accuracy = model.evaluate(x_train,y_train)
    print('Accuracy: %.2f' % ((accuracy*30)*100))
```

```
900/900 [==============================] - 1s 845us/step - loss: -32162664448.0000 - accuracy: 0.0309
Accuracy: 92.72
```

Fig 4.2.7 Accuracy

Predicting the mse, mae and rmse value for the above Model.

```
[ ] pred = model.predict(x_test)
```

```
[ ] mse=(round(mean_squared_error(pred, y_test),2))
    mae=(round(mean_absolute_error(pred, y_test),2))
    rmse=(round(np.sqrt(mean_squared_error(pred, y_test)),2))
    print(mse)
    print(mae)
    print(rmse)
```

```
317003361.1
3403.89
17804.59
```

```
[ ] df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
    k = df1[df1['Country/Region']=='India'].loc[:,'1/22/20':]
    india_confirmed = k.values.tolist()[0]
    dates = list(confirmed_df.columns[4:])
    dates = list(pd.to_datetime(dates))
    data = pd.DataFrame(columns = ['ds','y'])
    data['ds'] = dates
    data['y'] = india_confirmed
```

Now  Using the Arima Model to Predict for the future Cases

Predicting up to march of 2021

```
[ ] arima = ARIMA(data['y'], order=(5, 1, 0))
    arima = arima.fit(trend='c', full_output=True, disp=True)
    forecast = arima.forecast(steps= 30)
    pred = list(forecast[0])

    start_date = data['ds'].max()
    prediction_dates = []
    for i in range(30):
        date = start_date + datetime.timedelta(days=1)
        prediction_dates.append(date)
        start_date = date
    plt.figure(figsize= (15,10))
    plt.xlabel("Dates",fontsize = 20)
    plt.ylabel('Total cases',fontsize = 20)
    plt.title("Predicted Values for the next 15 Days using Arima Model" , fontsize = 20)

    plt.plot_date(y= pred,x= prediction_dates,linestyle ='dashed',color = '#ff9999',label = 'Predicted');
    plt.plot_date(y=data['y'],x=data['ds'],linestyle = '-',color = 'blue',label = 'Actual');
    plt.legend();
```

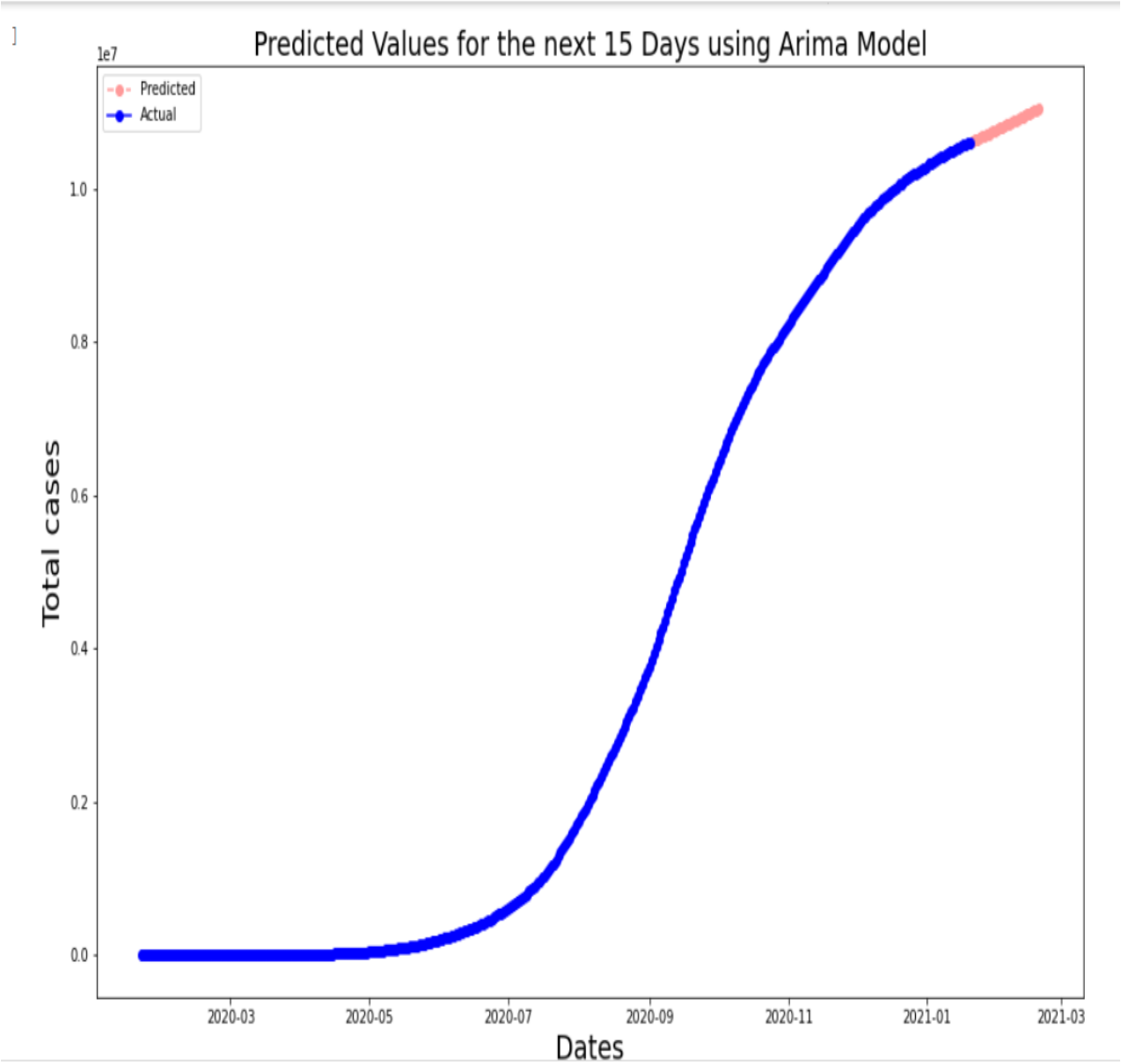


Fig 4.2.8 Forecast up to March 2021

## Predicting upto may of 2021

```
[ ] confirmed_df1 = pd.read_csv('/content/drive/MyDrive/time_series_covid_19_confirmed.csv')
    deaths_df = pd.read_csv('/content/drive/MyDrive/time_series_covid_19_deaths.csv')
    recovered_df = pd.read_csv('/content/drive/MyDrive/time_series_covid_19_recovered.csv')

    df = confirmed_df1.groupby('Country/Region').sum().reset_index()
    k = df[df['Country/Region']=='India'].loc[:,'1/22/20':]
    india_confirmed = k.values.tolist()[0]
    dates = list(confirmed_df1.columns[4:])
    dates = list(pd.to_datetime(dates))
    data = pd.DataFrame(columns = ['ds','y'])
    data['ds'] = dates
    data['y'] = india_confirmed
```

```
[ ] arima = ARIMA(data['y'], order=(5, 1, 0))
    arima = arima.fit(trend='c', full_output=True, disp=True)
    forecast = arima.forecast(steps= 30)
    pred = list(forecast[0])

    start_date = data['ds'].max()
    prediction_dates = []
    for i in range(30):
        date = start_date + datetime.timedelta(days=1)
        prediction_dates.append(date)
        start_date = date
    plt.figure(figsize= (15,10))
    plt.xlabel("Dates",fontsize = 20)
    plt.ylabel('Total cases',fontsize = 20)
    plt.title("Predicted Values for the next 15 Days using Arima Model" , fontsize = 20)
```

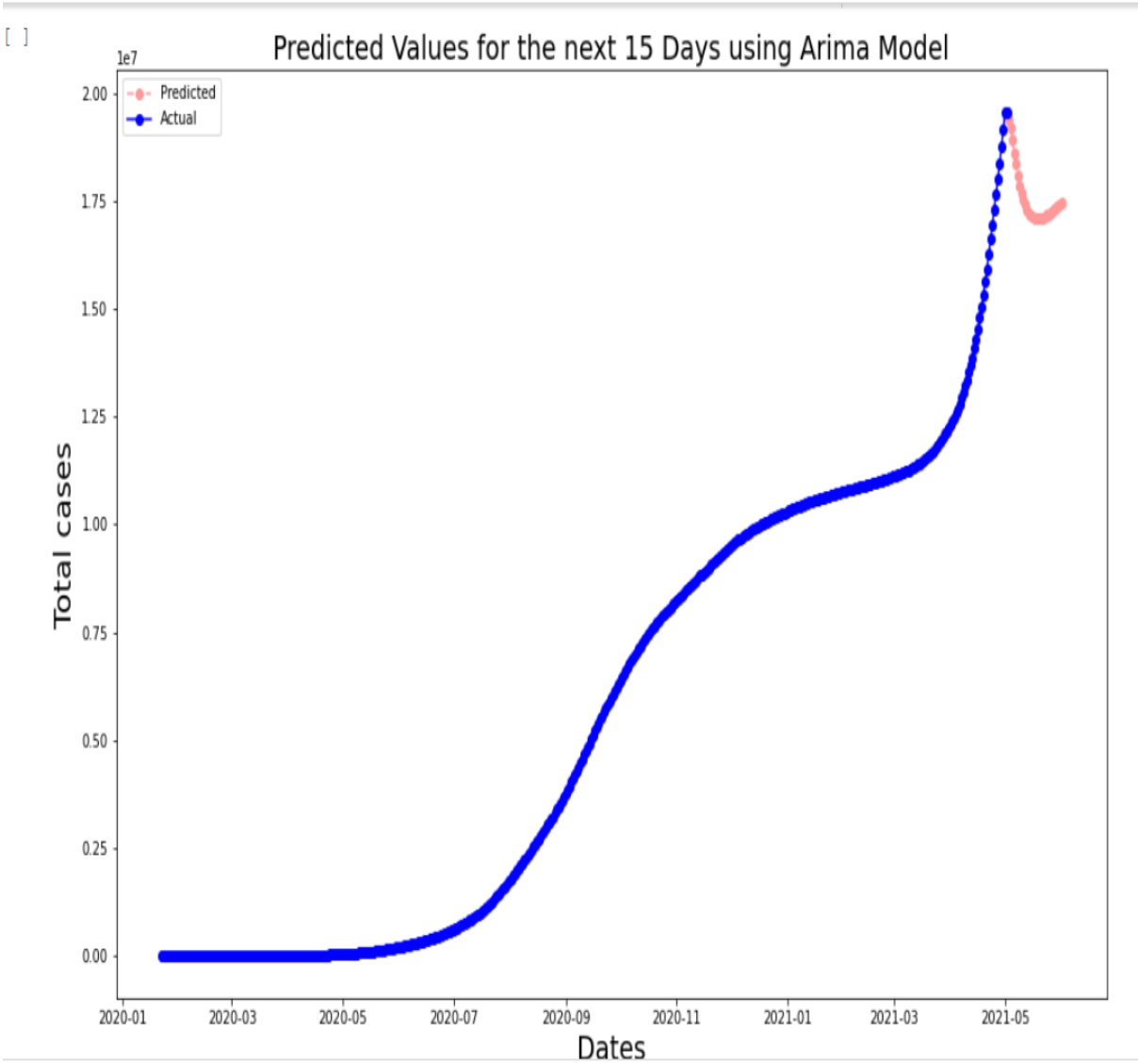


Fig 4.2.9 Forecast up to May 2021

# 5. CONCLUSION

The precariousness of the COVID-19 pandemic can ignite a massive global crisis. Some researchers and government agencies throughout the world have apprehensions that the pandemic can affect a large proportion of the world population. In this study, an ML-based prediction system has been proposed for predicting the risk of COVID19 outbreak globally. The system analyses dataset containing the day-wise actual past data and makes predictions for upcoming days using machine learning algorithms. The results of the study prove LR and LASSO perform well for forecasting to some extent to predict death rate and confirm cases. According to the results of these two models, the death rates will increase in upcoming days, and recoveries rate will be slowed down. It was very difficult to put an accurate hyper plane between the given values of the dataset. Overall we conclude that model predictions according to the current scenario are correct which may be helpful to understand the upcoming situation. The study forecasts thus can also be of great help for the authorities to take timely actions and make decisions to contain the COVID-19 crisis. This study will be enhanced continuously in the future course, next we plan to explore the prediction methodology using the updated dataset and use the most accurate and appropriate ML methods for forecasting. Real-time live forecasting will be one of the primary focuses in our future work.

# 6 REFERENCES

[1] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward,” PloS one, vol. 13, no. 3, 2018.

[2] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, “Machine learning strategies for time series forecasting,” in European business intelligence summer school. Springer, 2012, pp. 62–77.

[3] F. E. Harrell Jr, K. L. Lee, D. B. Matchar, and T. A. Reichert, “Regression models for prognostic prediction: advantages, problems, and suggested solutions.” Cancer treatment reports, vol. 69, no. 10, pp. 1071–1077, 1985.

[4] P. Lapuerta, S. P. Azen, and L. LaBree, “Use of neural networks in predicting the risk of coronary artery disease,” Computers and Biomedical Research, vol. 28, no. 1, pp. 38–52, 1995.

[5] K. M. Anderson, P. M. Odell, P. W. Wilson, and W. B. Kannel, “Cardiovascular disease risk profiles,” American heart journal, vol. 121, no. 1, pp. 293–298, 1991.

[6] H. Asri, H. Mousannif, H. Al Moatassime, and T. Noel, “Using machine learning algorithms for breast cancer risk prediction and diagnosis,” Procedia Computer Science, vol. 83, pp. 1064–1069, 2016.

[7] F. Petropoulos and S. Makridakis, “Forecasting the novel coronavirus covid-19,” Plos one, vol. 15, no. 3, p. e0231236, 2020.

[8] G. Grasselli, A. Pesenti, and M. Cecconi, “Critical care utilization for the covid-19 outbreak in lombardy, italy: early experience and forecast during an emergency response,” Jama, 2020.

[9] “Predicting Rise and Spread of COIVD-19 Epidemic using Time Series Forecasting Models in Machine Learning” in International Journal on Emerging Technologies Volume 11(4) Page 56-61(2020) (Indexed in Scopus) (ISSN: 2249-3255,0975-8364)

[10] “Statistical Analysis Providing COVID-19’s Lethalty Rate for the Elderly People using R” in International Journal of Advanced Science and Technology Volume 29, No.11s(2020) Page: 1366-1370 (Indexed in Scopus) (ISSN:2005-4238)