

COVID-19 FUTURE FORECASTING USING SUPERVISED MACHINE LEARNING MODELS

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

M.MAHESH REDDY (Reg. No. 37110476)

N.SAIPRAVEEN KUMAR(Reg. no. 37110483)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119**

MARCH – 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **MAHESH REDDY(Reg. No. 37110476)** and **PRAVEEN REDDY(Reg. No.37110483)** who carried out the project entitled "**COVID-19 FUTURE FORCASTING USING SUPERVISED MACHINE LEARNING**" under my supervision from August 2020 to March 2021.

Internal guide

S.PRAYLA SHYRY, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MAHESH REDDY** hereby declare that the Project Report entitled “**COVID-19 FUTURE FORECASTING USING SUPERVISED MACHINE LEARNING**” is done by me under the guidance of **S.PRAYLA SHYRY, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **S.PRAYLA SHYRY, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **S.PRAYLA SHYRY, M.E., Ph.D., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Machine learning (ML) based forecasting mechanisms have proved their significance to anticipate in 2perioperative 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 linear regression (LR), least absolute shrinkage and selection operator (LASSO), support vector machine (SVM), and exponential smoothing (ES) 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 10 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. The results prove that the ES performs best among all the used models followed by LR and LASSO which performs well in forecasting the new confirmed cases, death rate as well as recovery rate, while SVM performs poorly in all the prediction scenarios given the available dataset.

CONTENTS

S.NO	PAGE NO
1. Introduction	1
1.1. Motivation	2
1.2. Problem Definition	2
1.3. Objective of the Project	3
2. Literature Survey	4
3. Analysis	23
3.1. Existing System	23
3.2. Proposed System	23
3.3. Software Requirement Specification	24
3.3.1 Purpose	24
3.3.2 Scope	24
3.3.3 Overall Description	25
4. Design	28
4.1. UML diagrams	28
5. Implementation	33
5.1. Modules	33
5.2. Module description	33
5.3. Introduction of technologies used	35
5.4. Sample Code	40
6. Test cases	51
7. Screenshots	53
8. Conclusion	59
9. Future Enhancement	60

1. INTRODUCTION

1.1 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 [1]. One of the most significant areas of ML is forecasting [2], 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 [3]. There are lots of studies performed for the prediction of different diseases using machine learning techniques such as coronary artery disease [4], cardiovascular disease prediction [5], and breast cancer prediction [6]. In particular, the study [7] is focused on live forecasting of COVID-19 confirmed cases and study [8] 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) [9]. 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 [10]. 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 [11]. 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 Of 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 linear regression (LR), least absolute shrinkage and selection operator (LASSO), support vector machine (SVM), and exponential smoothing (ES). 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 R-squared score (R^2 score), Adjusted R-squared Score (R^2 adjusted), mean square error (MSE), mean absolute error (MAE), and

root mean square error (RMSE). This study has some key findings which are listed below:

- ES performs best when the time-series dataset has very limited entries.
- 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.

1.2 PROBLEM DEFINATION :-

the existing methods using Most models assume a standard SEIR structure. Fraser and colleagues to estimate size but make different changes on the nature of the different compartments and their respective residence times

1.3.Objective of the Project :-

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 linear regression (LR), least absolute shrinkage and selection operator (LASSO), support vector machine (SVM), and exponential smoothing (ES) have been used in this study to forecast the threatening factors of COVID-19.

considerably greater than those of statistical methods. The paper discusses the results, explains why the accuracy of ML models is below that of statistical ones and proposes some possible ways forward. The empirical results found in our research stress the need for objective and unbiased ways to test the performance of forecasting methods that can be achieved through sizable and open competitions allowing meaningful comparisons and definite conclusions

[2]A. Kata, “A postmodern Pandora’s box: Antivaccination misinformation on the Internet,” *Vaccine*, vol. 28, no. 7, pp. 1709–1716, Feb. 2010, doi: 10.1016/j.vaccine.2009.12.022.

The Internet plays a large role in disseminating anti-vaccination information. This paper builds upon previous research by analyzing the arguments proffered on anti-vaccination websites, determining the extent of misinformation present, and examining discourses used to support vaccine objections. Arguments around the themes of safety and effectiveness, alternative medicine, civil liberties, conspiracy theories, and morality were found on the majority of websites analyzed; misinformation was also prevalent. The most commonly proposed method of combating this misinformation is through better education, although this has proven ineffective. Education does not consider the discourses supporting vaccine rejection, such as those involving alternative explanatory models of health, interpretations of parental responsibility, and distrust of expertise. Anti-vaccination protestors make postmodern arguments that reject biomedical and scientific “facts” in favour of their own interpretations. Pro-vaccination advocates who focus on correcting misinformation reduce the controversy to merely an “educational” problem; rather, these postmodern discourses must be acknowledged in order to begin a dialogue. With morbidity and mortality from vaccine-preventable diseases [VPDs] having reached record lows [1], vaccines are one of the most successful tools for biomedical science and public health. Yet paradoxically, the effectiveness of vaccination has led to the reemergence of anti-vaccination sentiments. Vaccines may be seen as unnecessary or dangerous because incidence rates of VPDs in developed countries have plummeted. Vaccine “reactions” – negative health events following vaccination, attributed to the vaccine – then appear to be more common than the diseases themselves [2]. In this way, vaccines can be considered victims of their own success. The media plays a large role in disseminating and sensationalizing vaccine objections.

Such objections are part of what has been called the “anti-vaccination movement”, which has had a demonstrable impact on vaccination policies, and individual and community health [3]. A common sequence to vaccination scares involves scientific debate about potential vaccine risks, which communication technology transmits via a rhetoric of doubt; parents incorporate this with personal experiences and spread their views to their social groups [4]. These social groups exert considerable pressure on vaccination decisions by creating a “local vaccination culture” [5]. With the prominence of the Internet in today’s world, * Correspondence address: 110 Parkwood Cres., Hamilton, Ontario L8V 4Z7, Canada. Tel.: +1 905 387 3141. E-mail addresses: aniakata@gmail.com, kataa@mcmaster.ca. the attitudes, beliefs, and experiences of that local culture can quickly become global. Internet usage statistics show approximately 74% of Americans and 72% of Canadians are online [6]. An estimated 75–80% of users search for health information online [7]. Of these users, 70% say the information they encounter online influences their treatment decisions [8]. In 2006, 16% of users searched online for information on immunizations or vaccinations [9]. While online research is more convenient and accessible than reading medical literature or visiting health practitioners, too great a reliance on Internet-based information can be problematic. Over half (52%) of users believe “almost all” or “most” information on health websites is credible [8]; yet the availability of inaccurate and deceptive information online has labelled the Internet a “modern Pandora’s box” [10]. The nature of the Internet allows any and all opinions to spread widely and instantaneously. Individuals and groups gain exposure online without being filtered or reviewed – and anti-vaccination advocates have taken advantage of this fact. Anti-vaccination messages are more common on the Internet than in other forms of media, increasing the likelihood that vaccination decisions may be based on misleading information [11]. Indeed, parents who exempt children from vaccination are more likely to have obtained information from the Internet than parents who have their children vaccinated; they are also more likely to have used certain antivaccination websites [12]. This demonstrates the importance of understanding what messages are presented online and why they may be accepted. The body of research examining online anti-vaccinationism is not large, nor has there been a recent update [11,13–18]. Only one analysis [13] examined misinformation and deception on such sites, but was not quantitative. Prior research also acknowledged the need to understand discourses underlying anti-vaccination arguments [19,20], but

did not elaborate upon them. This analysis aims to address these issues by answering two main questions. First, what information is proffered on anti-vaccination websites, and what is its accuracy? Second, what discourses make these vaccine objections appealing?

2. Methods

2.1. Data collection

Web searches were conducted on May 21, 2009 using the terms “vaccine”, “vaccination”, and “immunization OR immunisation” input into Google.com (the American version of the search engine) and Google.ca (the Canadian version). Google was chosen as it is the most popular search engine, accounting for 73% of all Internet searches [21]. Results were classified as anti-vaccination and included for content analysis if they opposed childhood vaccination for any reason, without meeting any of the following exclusion criteria: (1) listserv or newsgroup pages; (2) pages solely containing brief notices about other website content; (3) news results, medical journals or library sites; (4) video results; (5) book previews; (6) non-English sites; (7) sites exclusively about adult immunization; (8) sites exclusively about veterinary vaccination and (9) inactive links. Criteria (see Tables 1 and 2) were applied to the anti-vaccination websites and coded as present or absent. Criteria were adapted from previous online antivaccination studies [11,13,14,17,18], as well as created by the author. Online health information seekers examine the first 10 search results 97.2% of the time [22]; therefore, only the first 10 results retrieved per term were examined. Of 30 total Google.com results, 5 of 21 immunization sites (24%) were classified as anti-vaccination. Of 30 total Google.ca results, 2 of 16 immunization sites (13%) were classified as anti-vaccination. To amass additional websites for a more meaningful study, the Canadian searches were extended to 50 results per term. Of 150 total results, 5 of 86 immunization sites (6%) were classified as anti-vaccination (two were duplicates of American results). Combining the American and Canadian results, 8 anti-vaccination websites were subjected to content analysis. Appendix A lists the sites analyzed. The proportions of pro- and anti-vaccination sites found per search term are illustrated in Fig. 1. Overall, American searches returned more anti-vaccination results (24%) than Canadian searches (6%), indicating American parents are more likely to encounter anti-vaccination sites via Google than are Canadian parents. Neither search engine returned any anti-vaccination results for “immunization OR immunisation”; this was expected based on research that found anti-vaccination groups avoid using the term “immunization” as they tend not to believe that vaccines confer immunity [16]. Although prior studies returned more search results [11], this does not necessarily

mean the number of anti-vaccination websites has decreased, but rather that their search rankings may have shifted. Nevertheless, the proportion of sites retrieved for some search terms is notable – 71% of results from the Google.com “vaccination” search were classified as antivaccination. 3.2. Content and themes Fig. 2 illustrates the percentage of analyzed sites with the listed themes present. Individual content criteria are quantified in Table 1. Individual design criteria are quantified in Table 2. 3.2.1. Safety and effectiveness “Vaccines are biological poisons, harmful to health, and a contributing factor in childhood illness.” (<http://www.vaclib.org/sites/debate/about.html>) Safety themes were present on all anti-vaccination websites analyzed. Every site claimed vaccines are poisonous and cause idiopathic illnesses. Sites stressed that vaccines contain substances poisonous to humans, including anti-freeze, ether, formaldehyde, mercury, and nanobacteria. Pertinent information was not elaborated upon – for instance, that the amount of potentially harmful substances in vaccines is not enough to produce toxic effects in humans, or that ether does not refer to the anaesthetic but to a chemical compound. Illnesses attributed to vaccines included:

[3] B. Martin, “Texas anti-vaxxers fear mandatory COVID-19 vaccines more than the virus itself,” Texas Monthly, 2020. [Online]. Available: <https://www.texasmonthly.com/news/texas-antivaxxers-fear-mandatory-coronavirus-vaccines>

Background

The World Health Organization lists vaccine hesitancy as one of 10 threats to global health. The antivaccine movement uses Facebook to promote messages on the alleged dangers and consequences of vaccinating, leading to a reluctance to immunize against preventable communicable diseases.

Objective

We would like to know more about the messages these websites are sharing via social media that can influence readers and consumers. What messages is the public receiving on Facebook about immunization? What content (news articles, testimonials, videos, scientific studies) is being promoted?

Methods

We proposed using a social media audit tool and 3 categorical lists to capture information on websites and posts, respectively. The keywords “vaccine,” “vaccine truth,” and “anti-vax” were entered in the Facebook search bar. A Facebook page was examined if it had between 2500 and 150,000 likes. Data about beliefs, calls to action, and testimonials were recorded from posts and listed under the categories Myths, Truths, and Consequences. Website data were entered in a social media audit template.

Results

Users’ posts reflected fear and vaccine hesitancy resulting from the alleged dangers of immunization featured on the website links. Vaccines were blamed for afflictions such as autism, cancer, and infertility. Mothers shared testimonies on alleged consequences their children suffered due to immunization, which have influenced other parents to not vaccinate their children. Users denied the current measles outbreaks in the United States to be true, retaliating against the government in protests for fabricating news.

Conclusions

Some Facebook messages encourage prevailing myths about the safety and consequences of vaccines and likely contribute to parents’ vaccine hesitancy. Deeply concerning is the mistrust social media has the potential to cast upon the relationship between health care providers and the public. A grasp of common misconceptions can help support health care provider practice. Many diseases have been almost, or completely, eradicated due to immunization. Immunization against disease prevents 2-3 million deaths per year internationally and could prevent even more with global vaccination improvements [1]. Immunization has vastly decreased mortality due to preventable communicable diseases. For example, before the introduction of the measles vaccine, 300,000-400,000 Canadians were infected every year, with some recoveries and many deaths [2]. Since the elimination of measles in 1998 due to vaccines, there have been very few cases in Canada [2]. Similarly, once the polio vaccine was introduced in Canada in the 1950s, cases reduced dramatically, and the current risk to the Canadian population is extremely low [3].

The World Health Organization (WHO) has declared vaccine hesitancy as one of the top 10 threats to global health [4]. Social media has helped fuel the growth of the antivaccine movement, with Facebook being identified as a key disseminator of misinformation surrounding the campaign [5-7]. Facebook is the largest social media platform, with more than 2 billion active monthly users [8]. There have been serious efforts to reduce the amount of misinformation spread on the social media site by lowering the ranking of Groups and Pages making false claims [7]. Social media administrators have been urged to remove these Pages and Groups altogether; however, counterarguments cite a violation of human rights to access uncensored information [7]. This paper exposes the messages of the antivaccine movement online and how individuals perceive immunization. We aimed to uncover the myths and truths that users of Facebook Pages observe and partake in. Health care consumers and health care providers may find themselves on opposite ends of the debate. Lack of immunization places the public at risk and decreases public health efforts to curb measles and polio and prevent outbreaks of influenza (flu) along with other communicable diseases. The shift in power between doctors and patients due to easy access to information online has led to the questioning of health care providers and increased shared decision making [6,9].

As most of the world awaits a vaccine to put an end to the COVID-19 (also known as the 2019 novel coronavirus) pandemic, “followers” of antivaccine Facebook Pages seem to fear the vaccine more than the virus itself [10]. Amid the COVID-19 pandemic, social media sites such as Facebook are unable to control the health misinformation that is spread on its Pages [11]. Antivaccine Pages have been providing conspiracy theories, safety concerns, and alternative health medication that grasp the attention of “undecided” individuals surfing the web for information on vaccines [11]. The WHO is fighting to stop the spread of misinformation online by collaborating with social media giants to find a way to regulate false claims [12]. Some examples of such claims include that COVID-19 is a bioweapon funded by the Bill & Melinda Gates Foundation or that it can simply be cured by consuming homemade concoctions (some include drinking bleach) [13]. Our aim was to know more about the messages that can influence readers and consumers that these websites are sharing via social media.

3. ANALYSIS

3.1. Existing System :-

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

3.2. 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 Of 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 linear regression (LR), least absolute shrinkage and selection operator (LASSO), support vector machine (SVM), and exponential smoothing (ES). 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 R-squared score (R² score), Adjusted R-squared Score (R² adjusted), mean square error (MSE), mean absolute error (MAE), and root mean square error (RMSE)

Advantages :-

- ES performs best when the time-series dataset has very limited entries.
- 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.

3.3. Software 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

- 1. Windows 7
- 2. Windows XP
- 3 . Windows 8
- Processor – Pentium IV or higher
- RAM -- 256 MB

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

1. Python
2. Mysql
3. Mysql client
4. WampServer 2.4

Technologies and Languages used to Develop

-- Python

3.3.1 Purpose :-

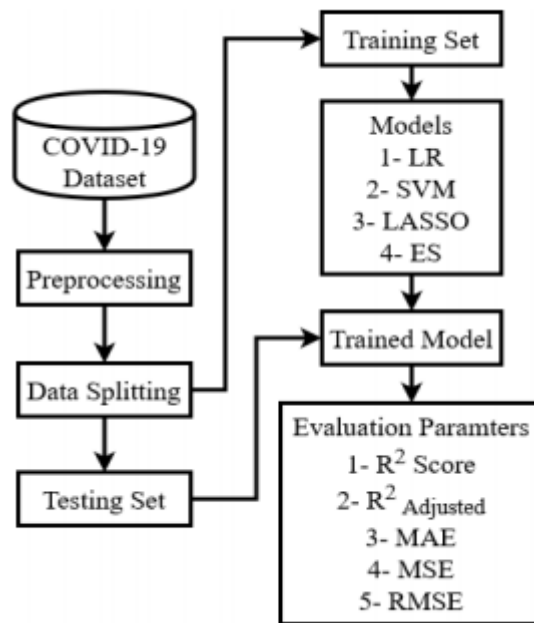
The aim of this study is the future forecasting of COVID19 spread focusing on the number of new positive cases, the number of deaths, and the number of recoveries. The dataset used in the study has been obtained from the GitHub

repository provided by the Center for Systems Science and Engineering, Johns Hopkins University [12]. The repository was primarily made available for the visual dashboard of 2019 Novel Coronavirus by the university and was supported by the ESRI Living Atlas Team. Dataset files are contained in the folder on the GitHub repository named (csse_covid_19_time_series). The folder contains daily time series summary tables, including the number of confirmed cases, deaths, and recoveries. All data are from the daily case report and the update frequency of data is one day

3.3.2 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

3.3.3 Overall Description :-



4. DESIGN

4.1. UML diagrams :-

4.1.1 CLASS DIAGRAM

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP).

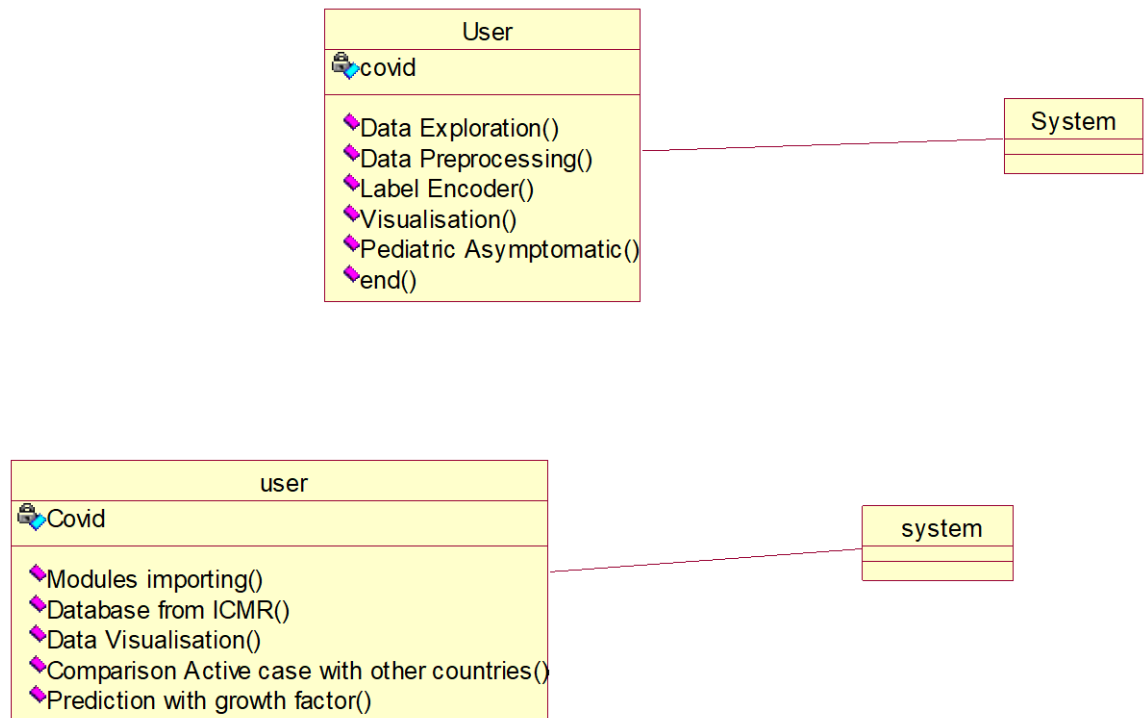


Fig 4.5 Class Diagram

In this there are three classes : data owner, data user and cloud server.

Cloud activates owner and user. Data owner uploads files and data user view and download files.

4.1.2USE 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.

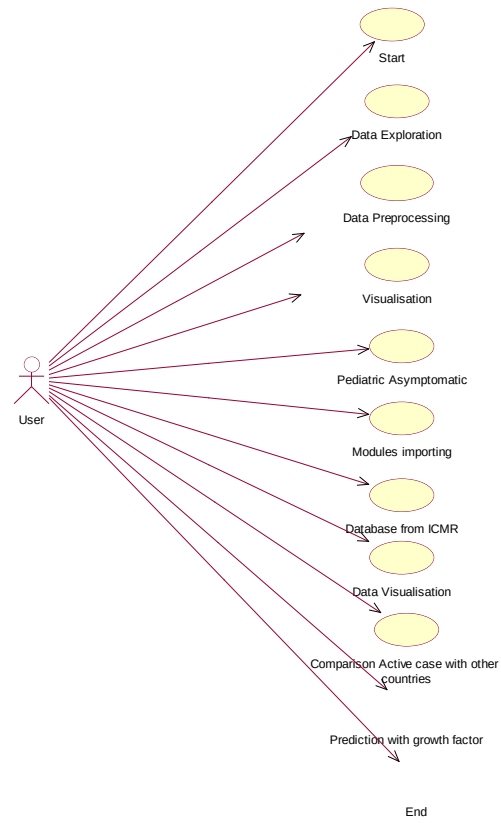


Fig: 4.6 Use case Diagram

In our Use case diagram, the actors are: User, Data Owner, Cloud.

There are ten different use cases where actors interact.

4.1.3 SEQUENCE DIAGRAM

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of collaboration. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. Sequence diagrams are time focus and they show the order of the interaction they not only have interaction but also some focus of control over the sequences. And also has time line to show from which part it is sending and which part is receiving.

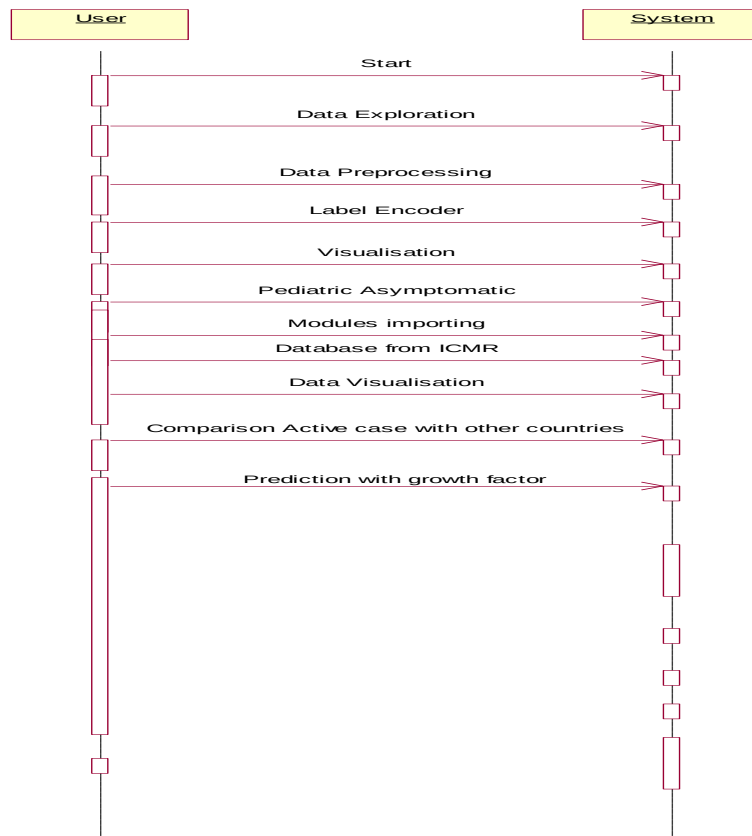
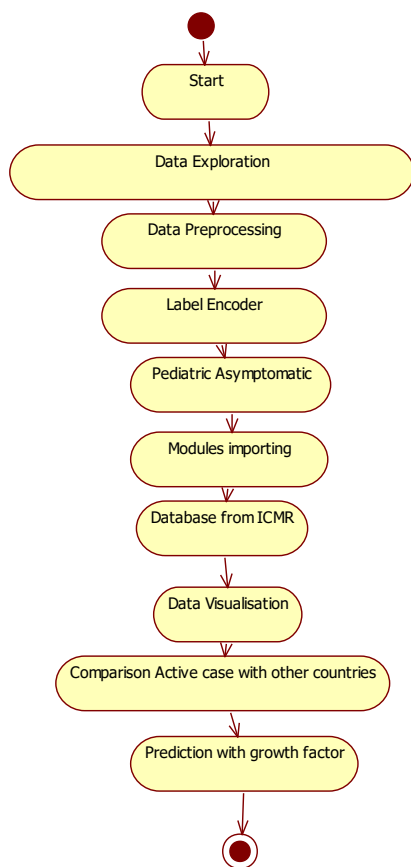


Fig: 4.7 Sequence Diagram

4.1.4ACTIVITY 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.



5. Implementation

5.1 SAMPLE CODE:

DATA LOGIN:

```
/******  
*****/  
  
from __future__ import absolute_import  
from __future__ import division  
from __future__ import print_function  
  
import argparse  
import collections  
from datetime import datetime  
import hashlib  
import os.path  
import random  
import re  
import sys  
import tarfile  
  
import numpy as np  
from six.moves import urllib  
import tensorflow as tf  
  
from tensorflow.python.framework import graph_util  
from tensorflow.python.framework import tensor_shape  
from tensorflow.python.platform import gfile  
from tensorflow.python.util import compat  
  
FLAGS = None  
MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1 # ~134M
```

```

def create_image_lists(image_dir, testing_percentage, validation_percentage):
    if not gfile.Exists(image_dir):
        tf.logging.error("Image directory '" + image_dir + "' not found.")
        return None
    result = collections.OrderedDict()
    sub_dirs = [
        os.path.join(image_dir, item)
        for item in gfile.ListDirectory(image_dir)]
    sub_dirs = sorted(item for item in sub_dirs
                       if gfile.IsDirectory(item))
    for sub_dir in sub_dirs:
        extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
        file_list = []
        dir_name = os.path.basename(sub_dir)
        if dir_name == image_dir:
            continue
        tf.logging.info("Looking for images in '" + dir_name + "'")
        for extension in extensions:
            file_glob = os.path.join(image_dir, dir_name, '*' + extension)
            file_list.extend(gfile.Glob(file_glob))
        if not file_list:
            tf.logging.warning('No files found')
            continue
        if len(file_list) < 20:
            tf.logging.warning(
                'WARNING: Folder has less than 20 images, which may cause issues.')
        elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
            tf.logging.warning(
                'WARNING: Folder {} has more than {} images. Some images will '
                'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
        label_name = re.sub(r'^a-z0-9+', '', dir_name.lower())
        training_images = []
        testing_images = []
        validation_images = []

```

```

for file_name in file_list:
    base_name = os.path.basename(file_name)
    hash_name = re.sub(r'_nohash_.*$', '', file_name)
    hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
    percentage_hash = ((int(hash_name_hashed, 16) %
                          (MAX_NUM_IMAGES_PER_CLASS + 1)) *
                       (100.0 / MAX_NUM_IMAGES_PER_CLASS))
    if percentage_hash < validation_percentage:
        validation_images.append(base_name)
    elif percentage_hash < (testing_percentage + validation_percentage):
        testing_images.append(base_name)
    else:
        training_images.append(base_name)
result[label_name] = {
    'dir': dir_name,
    'training': training_images,
    'testing': testing_images,
    'validation': validation_images,
}
return result

```

```

def get_image_path(image_lists, label_name, index, image_dir, category):
    if label_name not in image_lists:
        tf.logging.fatal('Label does not exist %s.', label_name)
    label_lists = image_lists[label_name]
    if category not in label_lists:
        tf.logging.fatal('Category does not exist %s.', category)
    category_list = label_lists[category]
    if not category_list:
        tf.logging.fatal('Label %s has no images in the category %s.',
                        label_name, category)
    mod_index = index % len(category_list)
    base_name = category_list[mod_index]

```

```

sub_dir = label_lists['dir']
full_path = os.path.join(image_dir, sub_dir, base_name)
return full_path

```

```

def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
                        category, architecture):
    return get_image_path(image_lists, label_name, index, bottleneck_dir,
                          category) + '_' + architecture + '.txt'

```

```

def create_model_graph(model_info):
    with tf.Graph().as_default() as graph:
        model_path = os.path.join(FLAGS.model_dir, model_info['model_file_name'])
        with gfile.FastGFile(model_path, 'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(
                graph_def,
                name="",
                return_elements=[
                    model_info['bottleneck_tensor_name'],
                    model_info['resized_input_tensor_name'],
                ]))
    return graph, bottleneck_tensor, resized_input_tensor

```

```

def run_bottleneck_on_image(sess, image_data, image_data_tensor,
                            decoded_image_tensor, resized_input_tensor,
                            bottleneck_tensor):
    resized_input_values = sess.run(decoded_image_tensor,
                                    {image_data_tensor: image_data})
    bottleneck_values = sess.run(bottleneck_tensor,
                                {resized_input_tensor: resized_input_values})

```

```

bottleneck_values = np.squeeze(bottleneck_values)
return bottleneck_values

```

```

def maybe_download_and_extract(data_url):
    dest_directory = FLAGS.model_dir
    if not os.path.exists(dest_directory):
        os.makedirs(dest_directory)
    filename = data_url.split('/')[-1]
    filepath = os.path.join(dest_directory, filename)
    if not os.path.exists(filepath):

        def _progress(count, block_size, total_size):
            sys.stdout.write('\r>> Downloading %s %.1f%%' %
                             (filename,
                              float(count * block_size) / float(total_size) * 100.0))
            sys.stdout.flush()

        filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
        print()
        statinfo = os.stat(filepath)
        tf.logging.info('Successfully downloaded', filename, statinfo.st_size,
                        'bytes.')
        tarfile.open(filepath, 'r:gz').extractall(dest_directory)

```

```

def ensure_dir_exists(dir_name):
    if not os.path.exists(dir_name):
        os.makedirs(dir_name)

```

```

bottleneck_path_2_bottleneck_values = {}

```

```

def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor):
    tf.logging.info('Creating bottleneck at ' + bottleneck_path)
    image_path = get_image_path(image_lists, label_name, index,
                                image_dir, category)
    if not gfile.Exists(image_path):
        tf.logging.fatal('File does not exist %s', image_path)
    image_data = gfile.FastGFile(image_path, 'rb').read()
    try:
        bottleneck_values = run_bottleneck_on_image(
            sess, image_data, jpeg_data_tensor, decoded_image_tensor,
            resized_input_tensor, bottleneck_tensor)
    except Exception as e:
        raise RuntimeError('Error during processing file %s (%s)' % (image_path,
                                                                    str(e)))
    bottleneck_string = ','.join(str(x) for x in bottleneck_values)
    with open(bottleneck_path, 'w') as bottleneck_file:
        bottleneck_file.write(bottleneck_string)


def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,
                             category, bottleneck_dir, jpeg_data_tensor,
                             decoded_image_tensor, resized_input_tensor,
                             bottleneck_tensor, architecture):
    label_lists = image_lists[label_name]
    sub_dir = label_lists['dir']
    sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
    ensure_dir_exists(sub_dir_path)
    bottleneck_path = get_bottleneck_path(image_lists, label_name, index,
                                           bottleneck_dir, category, architecture)
    if not os.path.exists(bottleneck_path):
        create_bottleneck_file(bottleneck_path, image_lists, label_name, index,

```

```

        image_dir, category, sess, jpeg_data_tensor,
        decoded_image_tensor, resized_input_tensor,
        bottleneck_tensor)
with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
did_hit_error = False
try:
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
except ValueError:
    tf.logging.warning('Invalid float found, recreating bottleneck')
    did_hit_error = True
if did_hit_error:
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
        image_dir, category, sess, jpeg_data_tensor,
        decoded_image_tensor, resized_input_tensor,
        bottleneck_tensor)
with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
return bottleneck_values

```

```

def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir,
    jpeg_data_tensor, decoded_image_tensor,
    resized_input_tensor, bottleneck_tensor, architecture):
    how_many_bottlenecks = 0
    ensure_dir_exists(bottleneck_dir)
    for label_name, label_lists in image_lists.items():
        for category in ['training', 'testing', 'validation']:
            category_list = label_lists[category]
            for index, unused_base_name in enumerate(category_list):
                get_or_create_bottleneck(
                    sess, image_lists, label_name, index, image_dir, category,
                    bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,

```

```

        resized_input_tensor, bottleneck_tensor, architecture)

    how_many_bottlenecks += 1
    if how_many_bottlenecks % 100 == 0:
        tf.logging.info(
            str(how_many_bottlenecks) + ' bottleneck files created.')

def get_random_cached_bottlenecks(sess, image_lists, how_many, category,
                                   bottleneck_dir, image_dir, jpeg_data_tensor,
                                   decoded_image_tensor, resized_input_tensor,
                                   bottleneck_tensor, architecture):
    class_count = len(image_lists.keys())
    bottlenecks = []
    ground_truths = []
    filenames = []
    if how_many >= 0:
        for unused_i in range(how_many):
            label_index = random.randrange(class_count)
            label_name = list(image_lists.keys())[label_index]
            image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
            image_name = get_image_path(image_lists, label_name, image_index,
                                         image_dir, category)
            bottleneck = get_or_create_bottleneck(
                sess, image_lists, label_name, image_index, image_dir, category,
                bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
                resized_input_tensor, bottleneck_tensor, architecture)
            ground_truth = np.zeros(class_count, dtype=np.float32)
            ground_truth[label_index] = 1.0
            bottlenecks.append(bottleneck)
            ground_truths.append(ground_truth)
            filenames.append(image_name)
    else:
        for label_index, label_name in enumerate(image_lists.keys()):

```

```

for image_index, image_name in enumerate(
    image_lists[label_name][category]):
    image_name = get_image_path(image_lists, label_name, image_index,
                                image_dir, category)
    bottleneck = get_or_create_bottleneck(
        sess, image_lists, label_name, image_index, image_dir, category,
        bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor, architecture)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
    bottlenecks.append(bottleneck)
    ground_truths.append(ground_truth)
    filenames.append(image_name)
return bottlenecks, ground_truths, filenames

```

```

def get_random_distorted_bottlenecks(
    sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
    distorted_image, resized_input_tensor, bottleneck_tensor):
    class_count = len(image_lists.keys())
    bottlenecks = []
    ground_truths = []
    for unused_i in range(how_many):
        label_index = random.randrange(class_count)
        label_name = list(image_lists.keys())[label_index]
        image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
        image_path = get_image_path(image_lists, label_name, image_index, image_dir,
                                    category)
        if not gfile.Exists(image_path):
            tf.logging.fatal('File does not exist %s', image_path)
        jpeg_data = gfile.FastGFile(image_path, 'rb').read()
        distorted_image_data = sess.run(distorted_image,
                                       {input_jpeg_tensor: jpeg_data})
        bottleneck_values = sess.run(bottleneck_tensor,

```

```

        {resized_input_tensor: distorted_image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
    bottlenecks.append(bottleneck_values)
    ground_truths.append(ground_truth)
    return bottlenecks, ground_truths

```

```

def should_distort_images(flip_left_right, random_crop, random_scale,
                          random_brightness):
    return (flip_left_right or (random_crop != 0) or (random_scale != 0) or
            (random_brightness != 0))

```

```

def add_input_distortions(flip_left_right, random_crop, random_scale,
                          random_brightness, input_width, input_height,
                          input_depth, input_mean, input_std):
    jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
    decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
    decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
    decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
    margin_scale = 1.0 + (random_crop / 100.0)
    resize_scale = 1.0 + (random_scale / 100.0)
    margin_scale_value = tf.constant(margin_scale)
    resize_scale_value = tf.random_uniform(tensor_shape.scalar(),
                                           minval=1.0,
                                           maxval=resize_scale)
    scale_value = tf.multiply(margin_scale_value, resize_scale_value)
    precrop_width = tf.multiply(scale_value, input_width)
    precrop_height = tf.multiply(scale_value, input_height)
    precrop_shape = tf.stack([precrop_height, precrop_width])
    precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
    precropped_image = tf.image.resize_bilinear(decoded_image_4d,

```

```

        precrop_shape_as_int)
precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
cropped_image = tf.random_crop(precropped_image_3d,
                               [input_height, input_width, input_depth])
if flip_left_right:
    flipped_image = tf.image.random_flip_left_right(cropped_image)
else:
    flipped_image = cropped_image
brightness_min = 1.0 - (random_brightness / 100.0)
brightness_max = 1.0 + (random_brightness / 100.0)
brightness_value = tf.random_uniform(tensor_shape.scalar(),
                                     minval=brightness_min,
                                     maxval=brightness_max)
brightened_image = tf.multiply(flipped_image, brightness_value)
offset_image = tf.subtract(brightened_image, input_mean)
mul_image = tf.multiply(offset_image, 1.0 / input_std)
distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
return jpeg_data, distort_result

```

```

def variable_summaries(var):
    with tf.name_scope('summaries'):
        mean = tf.reduce_mean(var)
        tf.summary.scalar('mean', mean)
        with tf.name_scope('stddev'):
            stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
        tf.summary.scalar('stddev', stddev)
        tf.summary.scalar('max', tf.reduce_max(var))
        tf.summary.scalar('min', tf.reduce_min(var))
        tf.summary.histogram('histogram', var)

```

```

def add_final_training_ops(class_count, final_tensor_name, bottleneck_tensor,
                           bottleneck_tensor_size):

```

```

with tf.name_scope('input'):
    bottleneck_input = tf.placeholder_with_default(

    if (i % FLAGS.eval_step_interval) == 0 or is_last_step:
        train_accuracy, cross_entropy_value = sess.run(
            [evaluation_step, cross_entropy],
            feed_dict={bottleneck_input: train_bottlenecks,
                       ground_truth_input: train_ground_truth})
        tf.logging.info('%s: Step %d: Train accuracy = %.1f%%' %
                        (datetime.now(), i, train_accuracy * 100))
        tf.logging.info('%s: Step %d: Cross entropy = %f' %
                        (datetime.now(), i, cross_entropy_value))
        validation_bottlenecks, validation_ground_truth, _ = (
            get_random_cached_bottlenecks(
                sess, image_lists, FLAGS.validation_batch_size, 'validation',
                FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
                decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
                FLAGS.architecture))
        # Run a validation step and capture training summaries for TensorBoard
        # with the `merged` op.
        validation_summary, validation_accuracy = sess.run(
            [merged, evaluation_step],
            feed_dict={bottleneck_input: validation_bottlenecks,
                       ground_truth_input: validation_ground_truth})
        validation_writer.add_summary(validation_summary, i)
        tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)' %
                        (datetime.now(), i, validation_accuracy * 100,
                         len(validation_bottlenecks)))

    # Store intermediate results
    intermediate_frequency = FLAGS.intermediate_store_frequency

    if (intermediate_frequency > 0 and (i % intermediate_frequency == 0)
        and i > 0):

```

```

        intermediate_file_name = (FLAGS.intermediate_output_graphs_dir +
                                   'intermediate_' + str(i) + '.pb')

    default=10,
    help='How often to evaluate the training results.'
)
parser.add_argument(
    '--train_batch_size',
    type=int,
    default=100,
    help='How many images to train on at a time.'
)
parser.add_argument(
    '--test_batch_size',
    type=int,
    default=-1,
    help="""\
How many images to test on. This test set is only used once, to evaluate
the final accuracy of the model after training completes.
A value of -1 causes the entire test set to be used, which leads to more
stable results across runs.\
""")
)
parser.add_argument(
    '--validation_batch_size',
    type=int,
    default=100,
    help="""\
How many images to use in an evaluation batch. This validation set is
used much more often than the test set, and is an early indicator of how
accurate the model is during training.
A value of -1 causes the entire validation set to be used, which leads to
more stable results across training iterations, but may be slower on large
training sets.\

```

5.3 Introduction of Technologies Used :-

Python :-

What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms.

Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

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

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

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

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

2. Extensible

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

3. Embeddable

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

4. Improved Productivity

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

5. IOT Opportunities

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

6. Simple and Easy

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

7. Readable

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

8. Object-Oriented

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

9. Free and Open-Source

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

10. Portable

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

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages. *Any doubts till now in the advantages of Python? Mention in the comment section.*

Advantages of Python Over Other Languages

1. Less Coding

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

2. Affordable

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

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

3. Python is for Everyone

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

Disadvantages of Python

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

1. Speed Limitations

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

2. Weak in Mobile Computing and Browsers

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

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

3. Design Restrictions

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

4. Underdeveloped Database Access Layers

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

5. Simple

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

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

History of Python : -

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

Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already

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

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

Purpose :-

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

Python

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

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

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

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

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

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

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

How to Install Python on Windows and Mac :

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

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

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

Download the Correct version into the system

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



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

Step 2: Click on the Download Tab.



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

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	Release Notes
Python 3.6.9	July 2, 2019	 Download	Release Notes
Python 3.7.3	March 25, 2019	 Download	Release Notes
Python 3.4.10	March 18, 2019	 Download	Release Notes
Python 3.5.7	March 18, 2019	 Download	Release Notes
Python 2.7.16	March 4, 2019	 Download	Release Notes
Python 3.7.2	Dec. 24, 2018	 Download	Release Notes

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

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

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Clipped source tarball	Source release		68111671e5b3db4ae77b9ab01b0f99be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae6097051c2eca45ee3604803	17133432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583da71a4c2c8a8cee08e6	34898436	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c56b2ac56cade6b47c3d2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c3cf6d8ec0b0abe83184a072ba2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76d4bdc3043a583e5a3e00	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28c91c60ff6d72aee653a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3bd19841879fda94133574129d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d8451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d3117d882c30983ea371d87c	1324608	SIG

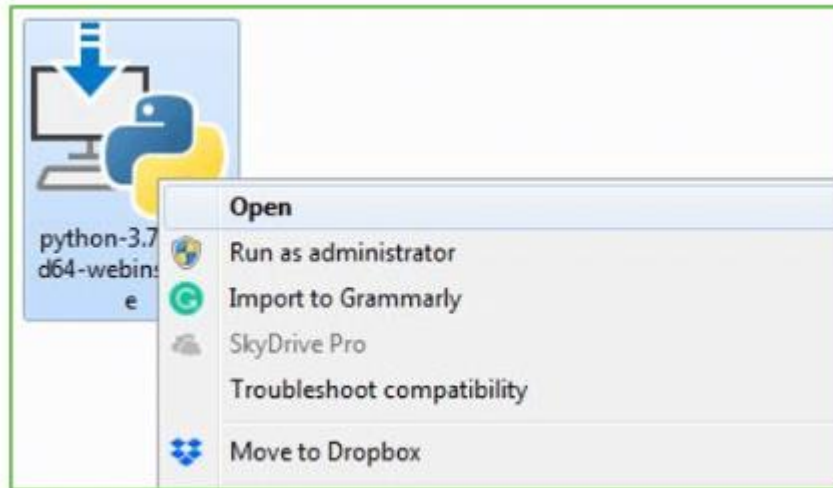
- To download **Windows 32-bit python**, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download **Windows 64-bit python**, you can select any one from the three options: Windows x86-64 embeddable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

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

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

Installation of Python

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



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



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



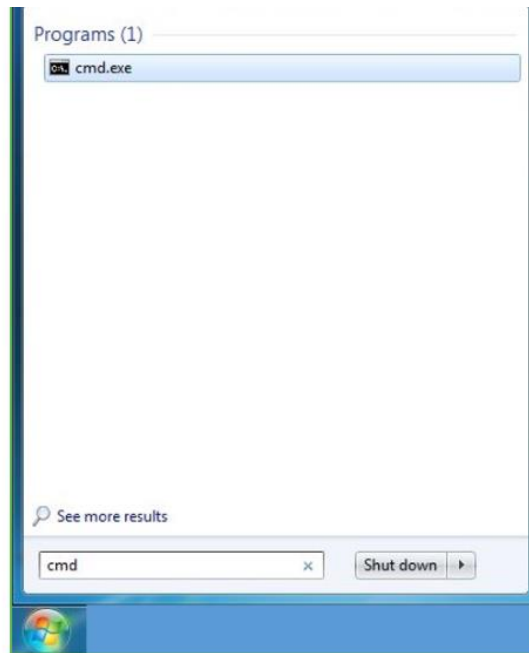
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

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

Verify the Python Installation

Step 1: Click on Start

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



Step 3: Open the Command prompt option.

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

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>python -V
Python 3.7.4
C:\Users\DELL>_
```

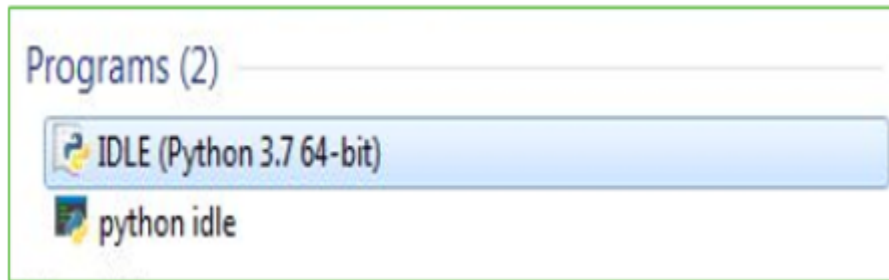
Step 5: You will get the answer as 3.7.4

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

Check how the Python IDLE works

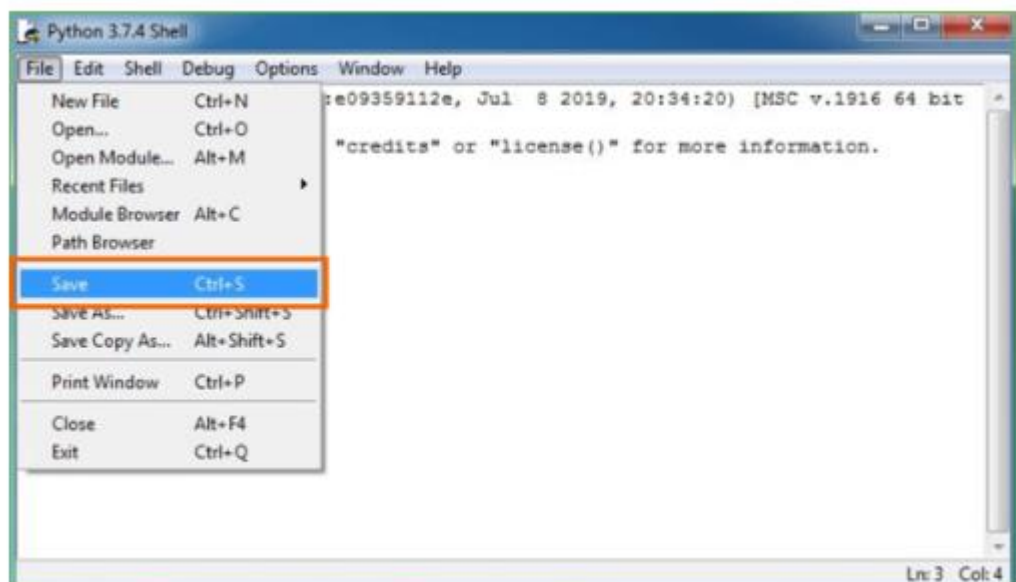
Step 1: Click on Start

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



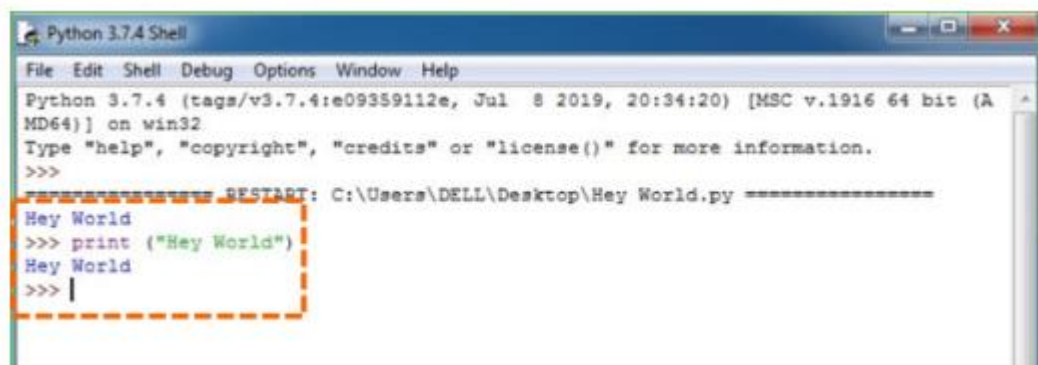
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

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



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

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



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

Note: Unlike Java, Python doesn't need semicolons at the end of the statements otherwise it won't work.

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 are conventionally named SciKits. As such, the module

Anaconda :-

What is Anaconda Python?

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages.

This work environment, Anaconda is used for scientific computing, data science, statistical analysis, and machine learning. The latest version of Anaconda 5.0.1 is released in October 2017.

The released version 5.0.1 addresses some minor bugs and adds useful features, such as updated R language support. All of these features weren't available in the original 5.0.0 release.

This package manager is also an environment manager, a Python distribution, and a collection of open source packages and contains more than 1000 R and Python Data Science Packages.

Why Anaconda for Python?

There's no big reason to switch to Anaconda if you are completely happy with your regular python. But some people like data scientists who are not full-time developers, find anaconda much useful as it simplifies a lot of common problems a beginner runs into.

Anaconda can help with –

- Installing Python on multiple platforms
- Separating out different environments
- Dealing with not having correct privileges and
- Getting up and running with specific packages and libraries

How to Download Anaconda 5.0.1?

The free version of Anaconda distribution community edition can be downloaded directly from Anaconda's website. For the enterprise edition, one needs professional support from Anaconda's sales team.

Conda treats Python the same as any other package, so it is easy to manage and update multiple installations.

Anaconda supports Python 2.7, 3.4, 3.5 and 3.6. The default is Python 2.7 or 3.6, depending on which installer you used:

- For the installers "Anaconda" and "Miniconda," the default is 2.7.
- For the installers "Anaconda 3" or "Miniconda 3," the default is 3.6.

How to Install Anaconda?

Once downloaded the .exe file, run through the installer. Do accept the terms, and finish the installation. To check, close the browser and pull up the terminal.

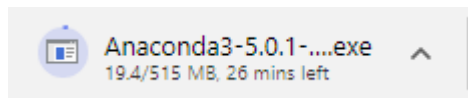
Once the installation is complete, it should have automatically added that to the path. To test this, go ahead and type 'python'.

The version of python i.e. 3 will show and also the anaconda distribution will be seen. If you install the 4 versions of Anaconda, then all the packages, which are there in the packages, can also be imported easily.

To check type `import numpy` or `import matplotlib` and run that.

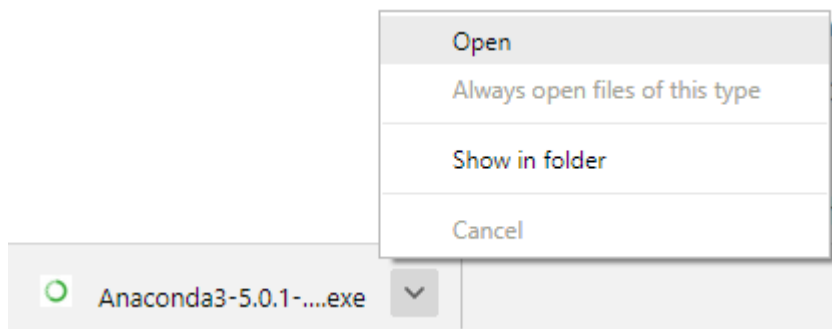
You can check out our [Python training courses](#) for more details.

The download is quite large (over 500 MB) so it may take a while for Anaconda to download.



4. Open and run the installer

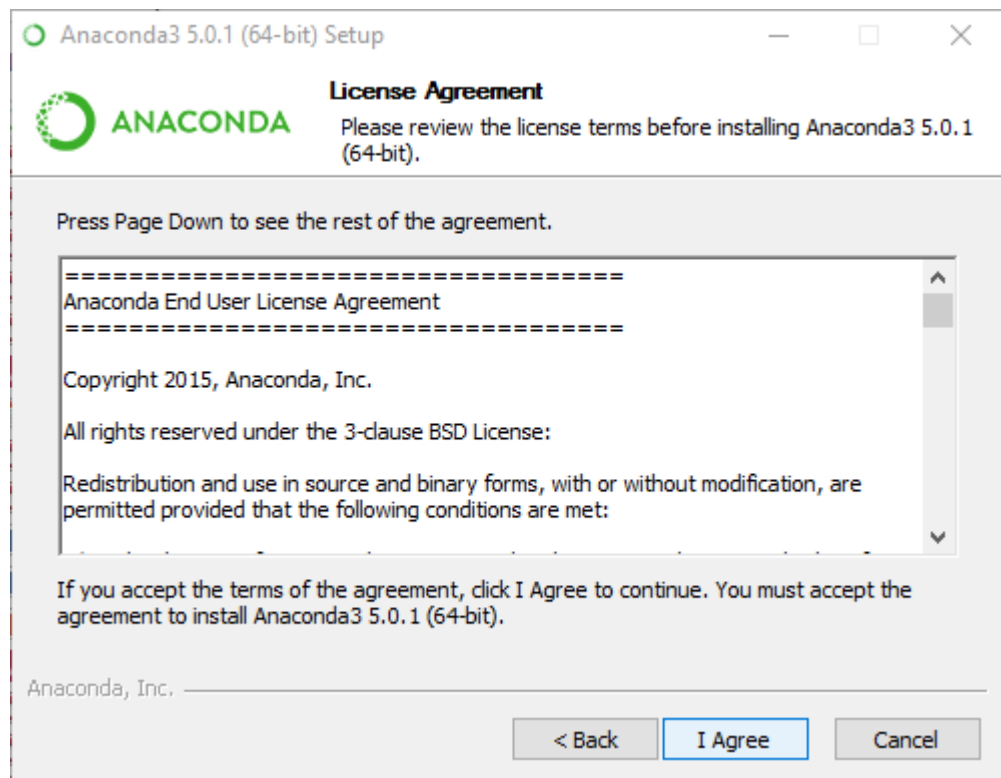
Once the download completes, open and run the **.exe** installer



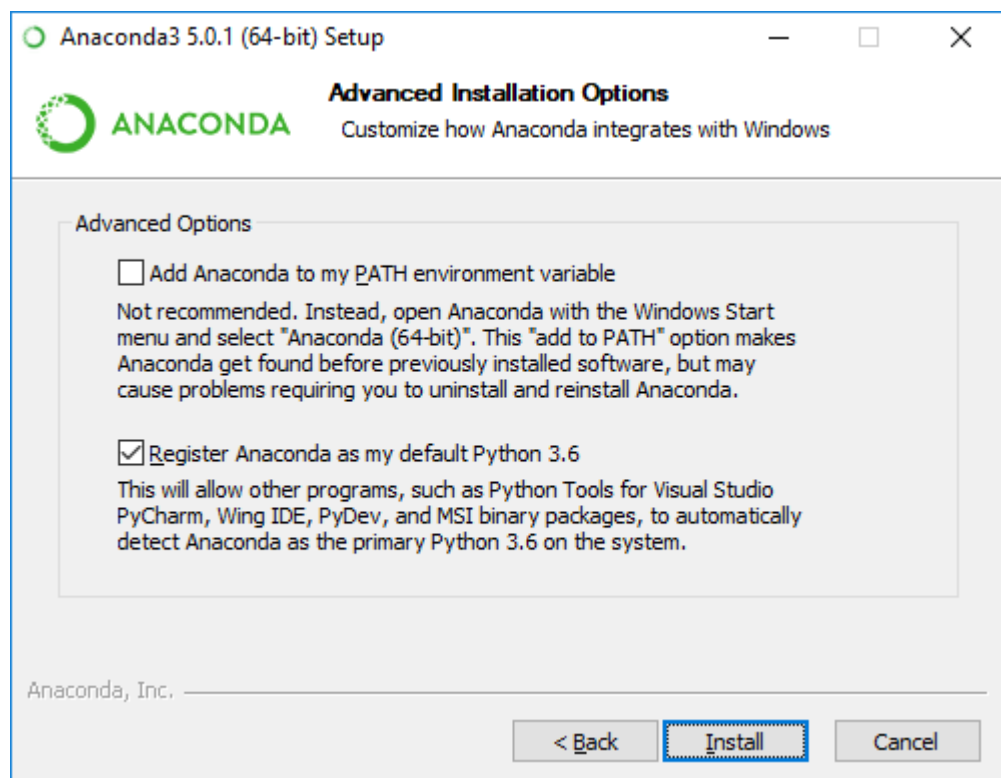
At the beginning of the install, you need to click **Next** to confirm the installation.



Then agree to the license.

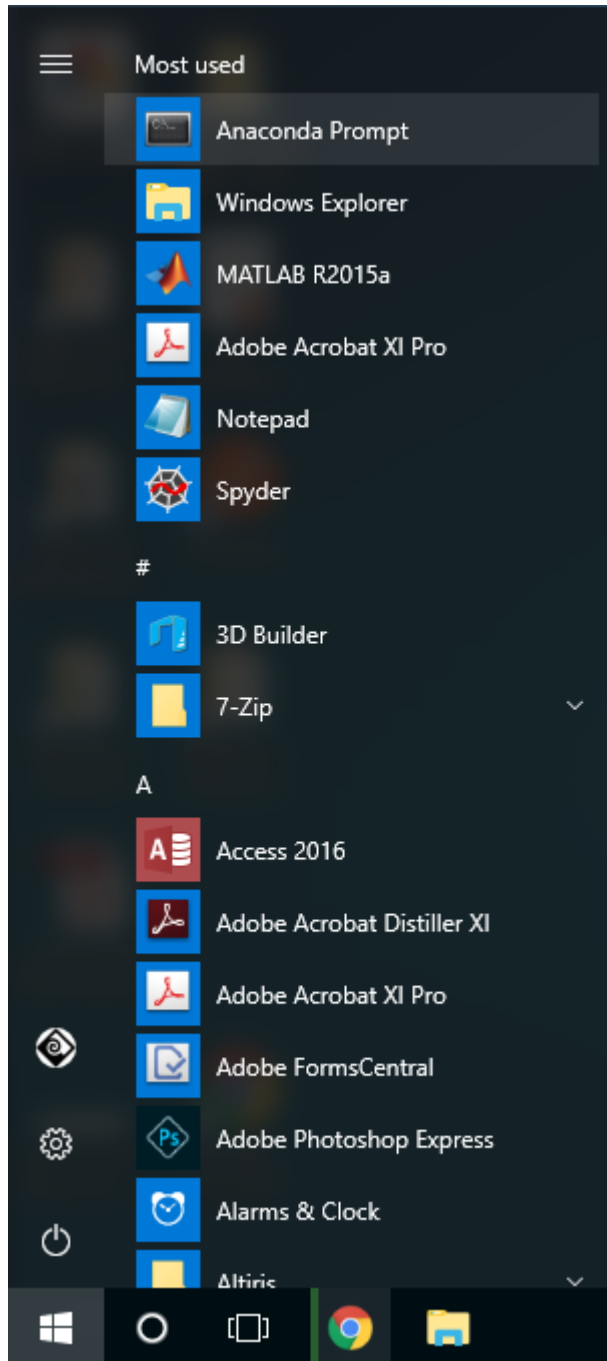


At the Advanced Installation Options screen, I recommend that you **do not check** "Add Anaconda to my PATH environment variable"



5. Open the Anaconda Prompt from the Windows start menu

After the installation of Anaconda is complete, you can go to the Windows start menu and select the Anaconda Prompt.

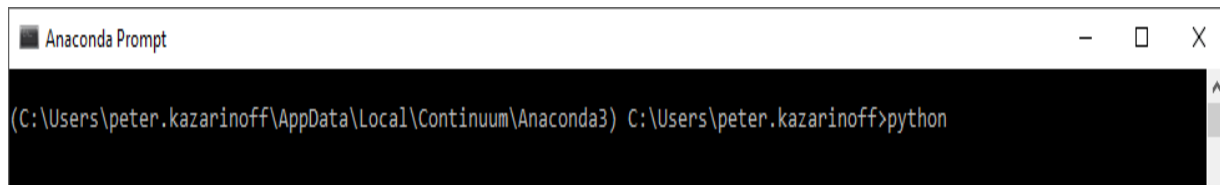


This opens the **Anaconda Prompt**. **Anaconda** is the Python distribution and the **Anaconda Prompt** is a command line shell (a

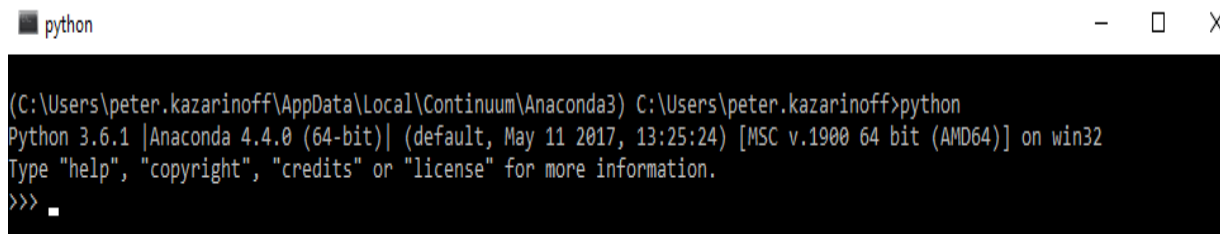
program where you type in commands instead of using a mouse). The black screen and text that makes up the **Anaconda Prompt** doesn't look like much, but it is really helpful for problem solvers using Python.

At the Anaconda prompt, type `python` and hit [Enter]. The `python` command starts the Python interpreter, also called the Python REPL (for Read Evaluate Print Loop).

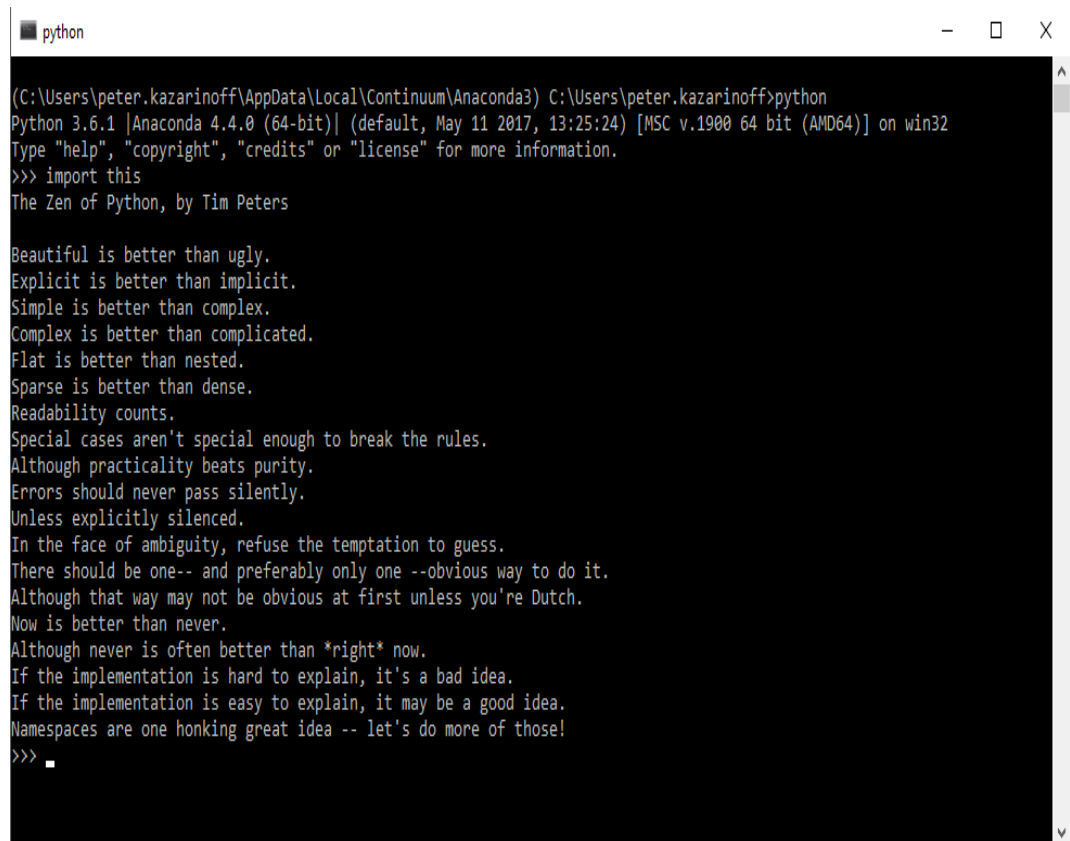
> `python`

A screenshot of the Anaconda Prompt window. The title bar says "Anaconda Prompt". The command prompt shows the path "(C:\Users\peter.kazarinoff\AppData\Local\Continuum\Anaconda3) C:\Users\peter.kazarinoff>python" with the cursor at the end of the line.

Note the Python version. You should see something like `Python 3.6.1`. With the interpreter running, you will see a set of greater-than symbols `>>>` before the cursor.

A screenshot of the Python interpreter window. The title bar says "python". The command prompt shows the path "(C:\Users\peter.kazarinoff\AppData\Local\Continuum\Anaconda3) C:\Users\peter.kazarinoff>python". Below the path, it displays: "Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32". It then says "Type 'help', 'copyright', 'credits' or 'license' for more information." and shows the prompt ">>> " with a cursor.

Now you can type Python commands. Try typing `import this`. You should see the ***Zen of Python*** by Tim Peters



```
(C:\Users\peter.kazarinoff\AppData\Local\Continuum\Anaconda3) C:\Users\peter.kazarinoff>python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>> .
```

To close the Python interpreter, type `exit()` at the prompt `>>>`. Note the double parenthesis at the end of the `exit()` command. The `()` is needed to stop the Python interpreter and get back out to the **Anaconda Prompt**.

To close the **Anaconda Prompt**, you can either close the window with the mouse, or type `exit`, no parenthesis necessary.

When you want to use the Python interpreter again, just click the Windows Start button and select the **Anaconda Prompt** and type `python`.

Algorithm used in python :-

CNN Algorithm:

Convolutional Neural Networks :-

Convolutional Neural Network are very efficient In image recognition problems ,as these are very efficient in understanding and learning spatial lfeatures.These spatial features play a key role in the decision process of image classification problems.Some CNN architectures involve complex designs and expensive to train.However,in this project,we simplified the CNN architecture in order to reduce the resource usage.

6. Test cases

Introduction

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements contrary to the actual requirements.

Types of Testing

Manual Testing

Manual testing is the process of testing software by hand to learn more about it, to find what is and isn't working. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end user's in mind. Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results, through to high-level guides that steer exploratory testing sessions. There are lots of sophisticated tools on the market to help with manual testing like Test Pad.

Automation Testing

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools. Some of the famous automation testing tools for functional testing are QTP/UFT and Selenium.

Testing Methods

Static Testing

It is also known as Verification in Software Testing. Verification is a static method of checking documents and files. Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not. Activities involved here are Inspections, Reviews, and Walkthroughs.

Dynamic Testing

It is also known as Validation in Software Testing. Validation is a dynamic process of testing the real product. Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not. Activities involved in this are testing the software application.

Testing Approaches

White Box Testing

It is also called as Glass Box, Clear Box, and Structural Testing. White Box Testing is based on the application's internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. This testing is usually done at the unit level.

Black Box Testing

It is also called as Behavioural/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure.

Grey Box Testing

The grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.

Testing Levels

Unit Testing

Unit Testing is done to check whether the individual modules of the source code are working properly. I.e. testing each and every unit of the application separately by the developer in the developer's environment. It is AKA Module Testing or Component Testing.

Integration Testing

Integration Testing is the process of testing the connectivity or data transfer between a couple of units tested modules. It is AKA I&T Testing or String Testing. It is subdivided into the Top-Down Approach, Bottom-Up Approach and Sandwich Approach (Combination of Top-Down and Bottom-Up).

System Testing

It's a black box testing. Testing the fully integrated application this is also called as the end to end scenario testing. To ensure that the software works in all intended target systems. Verify thorough testing of every input in the application to check for desired outputs. Testing of the users' experiences with the application.

Acceptance Testing

To obtain customer sign-off so that software can be delivered and payments received. Types of Acceptance Testing are Alpha, Beta & Gamma Testing.

Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Start the Application	Host the application and test if it starts making sure the required software is available	If it doesn't start	We cannot run the application.	The application hosts success.	High	High
02	Home Page	Check the deployment environment for properly loading the application.	If it doesn't load.	We cannot access the application.	The application is running successfully.	High	High
03	User Mode	Verify the working of the application in freestyle mode	If it doesn't respond	We cannot use the Freestyle mode.	The application displays the Freestyle Page	High	High
04	Data Input	Verify if the application takes input and updates the database.	If it fails to take the input or store in The Database	We cannot proceed further	The application updates the input to the database.	High	High

8. Conclusion and Future work

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 [26], [27]. 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 that ES performs best in the current forecasting domain given the nature and size of the dataset. LR and LASSO also 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. SVM produces poor results in all scenarios because of the ups and downs in the dataset values. It was very difficult to put an accurate hyperplane 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.

Bibliography :-

- [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] WHO. Naming the coronavirus disease (covid-19) and the virus that causes it. [Online]. Available: [https://www.who.int/emergencies/diseases/novelcoronavirus-2019/technical-guidance/naming-the-coronavirus-disease-\(covid-2019\)-and-the-virus-that-causes-it](https://www.who.int/emergencies/diseases/novelcoronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it)
- [10] C. P. E. R. E. Novel et al., “The epidemiological characteristics of an outbreak of 2019 novel coronavirus diseases (covid-19) in china,” *Zhonghua liu xing bing xue za zhi= Zhonghua liuxingbingxue zazhi*, vol. 41, no. 2, p. 145, 2020.

Code
snippets

for any
errors
<http://stackoverflow.com/>

Android
Development
Guide
<https://www.udemy.com/android>

Xml and
Layout
Guide
<https://www.androidhive.com/>

Connecti
ng to
Firebase
Docs
<https://firebase.google.com/>

Software Testing
http://en.wikipedia.org/wiki/Software_testing

Manual Testing
http://en.wikipedia.org/wiki/Manual_testing

Performance Testing
http://en.wikipedia.org/wiki/Software_performance_testing