

Prediction of Covid-19 Death Rate Using Machine Learning

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

By

AVINENI GNANA SAI TEJA

(Reg.No:38110060)



**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

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600119

MAY 2022



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **AVINENI GNANA SAI TEJA (Reg.No.38110060)** who carried out the project entitled “**PREDICTION OF COVID-19 DEATH RATE USING MACHINE LEARNING**” under our supervision from November 2021 to March 2022.

Internal Guide
Dr.B.U.Anu Barathi M.E.,Ph.D.

Head of the Department
Dr.L.Lakshmanan M.E.,Ph.D,
Dr. S. Vigneshwari, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **AVINENI GNANA SAI TEJA (Reg. No. 38110060)** hereby declare that Project report entitled “**PREDICTION OF COVID-19 DEATH RATE USING MACHINE LEARNING**” done under the guidance of **DR. B.U.ANU BARATH, M.E, P.hD.**, at Sathyabama 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 **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr.L.Lakshmanan M.E.,Ph.D., Dr. S.Vigneshwari,M.E., Ph. D. ,** Heads of the Department, 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 **Dr.B.U.Anu Barathi, M.E.,Ph.D.,** 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

The Covid-19 Corona virus, otherwise called SARS-CoV-2, has been killed from the world and things go severely. It is a pandemic that spreads to the world consistently. The current framework gives engine data in a cooperative manner to handily investigate and foresee the advancement of an issue in a specific locale. A perusing and composing machine (ML) can be utilized to adequately tackle an issue and its activity. Perusing and composing machine controls (SMLMs) and calculations connected with backwardness and scientific composing assist with displaying preparing. The distance is more noteworthy than the quantity of direct articles. Inversion calculation(MDR). The worldwide informational index is gathered, pre-handling, and the quantity of affirmed cases to a specific date vanishes. It assists with preparing a model for anticipating the quantity of legitimate cases around the world.

Keywords— Machine Literacy (ML), Supervised Machine Literacy Models (SMLM), Multiple direct retrogression (MDR)

TABLE OF CONTENTS

CHAPTER NO NO:	TITLE	PAGE
	ABSTRACT	--> 5
	LIST OF FIGURES	--> 8
1	INTRODUCTION	--> 9
	1.1 OUTLINE OF THE PROJECT	9
	1.2 LITERATURE REVIEW	11
	1.3 OBJECTIVE OF THE PROJECT	14
		15
2	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	2.1 AIM AND SCOPE OF THE PROJECT	15
	2.2 HARDWARE REQUIREMENTS	15
	2.3 SOFTWARE REQUIREMENTS	15
	2.4 LIBRARIES USED	15
3	EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED	
	-->16	
	3.1 SYSTEM DESIGN	16
	3.2 METHODS	17
	3.2.1 EXISTING SYSTEM	20
	3.2.2 PROBLEM STATEMENT	20
	3.2.3 PROPOSED SYSTEM	21
	3.3 TECHNOLOGY USED	22
	3.4 ALOGORITHM USED	38
	3.5 MODULES	42
4	RESULTS AND DISCUSSION	-->46
	4.1 IMPLEMENTATION DETAILS	46

5	CONCLUSIONS	-->51
6	REFERENCES	-->52
7	APPENDIX	-->55
	7.1. SOURCE CODE	55
	7.2. SCREENSHOTS	60

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
3.1	SYSTEM DESIGN AND ARCHITECTURE --> SYSTEM DESIGN	16
3.1	SYSTEM DESIGN AND ARCHITECTURE --> SYSTEM ARCHITECTURE	16
3.4	SUPERVISED ALGORITHMS	39
3.4	UNSUPERVISED ALGORITHMS	40
3.5	MODULES - FEATURE EXTRACTION	44
4.1	MODEL ACCURACY- RESULT	47
4.1	COVID-19 DEATH RATE PREDICTION	48
4.1	OUTPUT SCREEN-SHOTS (DENSITY MAP AND CORRELATION MATRIX MAP)	49
4.1	GRAPHICAL REPRESENTATION (NEW TESTS VS NEW DEATHS AND NEW CASES VS NEW DEATHS)	50
4.1	RESULTS - DEATH-RATE PREDICTION	52
7.2	SCREENSHOTS	61

1. INTRODUCTION

1.1 OUTLINE OF THE PROJECT:

As of September 6, 2021, COVID-19 has caused more than 219 million infections worldwide and resulted in more than 4.55 million deaths. Complications are more common among elderly patients and people with preexisting conditions, and the rate of intensive care unit (ICU) admission is substantially higher in these groups . ICU admissions rely on the critical care capacity of the health care system. Iran, which is the primary test bed for this study, was one of the first countries hit by COVID-19. The ICU admission rate involves about 32% of all hospitalizations, and the ICU mortality rate is about 39%. With the potential of new waves of COVID-19 infections driven by more transmissible variants, ICU hospitalization numbers are expected to rise, leading to shortages of ICU beds and critical management equipment.

There is also the risk of a global shortage of effective medical supplies, making the judicious use of these medications a top priority for healthcare systems. An individual-based prediction model is essential for tailoring treatment strategies and would aid in expanding our insights into the pathogenesis of COVID-19. A number of risk assessment scores are available to predict the severity of different diseases in ICU patients. Predictors of the need for intensive respiratory or vasopressor support in patients with COVID-19 and of mortality in COVID-19 patients with pneumonia have been identified.

To date, no general mortality prediction scores have been available for ICU admitted COVID-19 patients, irrespective of the patients' clinical presentation. Additionally, existing risk scales rely on parameters measured by health care providers such as blood pressure, respiratory rate, and oxygen saturation, which are subject to human error and operator bias especially under challenging and stressful conditions when numbers of COVID-19 patients surge. Thus, it remains vital to develop more unbiased risk-assessment tools that can predict the most likely outcomes for individual patients with COVID-19. Recent advances in artificial intelligence (AI) technology for disease screening show promise as computer-aided diagnosis and prediction tools. In the era of COVID-19, AI has played an important role in early diagnosis of infection, contact tracing, and drug and vaccine development.

Thus, AI represents a useful technology for the management of COVID-19 patients with the potential to help control the mortality rate of this disease. Nevertheless, an AI tool for making standardized and accurate predictions of outcomes in COVID-19 patients with severe disease is currently missing. Beyond the general benefits of data-driven decision-making, the pandemic has also exposed the need for computational assistance to health care providers, who under the pressure of severely ill patients may make mistakes in judgment.

Stressful conditions and burnout in health care providers can reduce their clinical performance, and a lack of accurate judgment can lead to increased mortality rates. Artificial intelligence can help healthcare professionals determine who needs a critical level of care more precisely. Indeed, the effective use of AI could mitigate the severity of this outbreak. Here, we propose a personalized machine-learning (ML) method for predicting mortality in COVID-19 patients based on routinely available laboratory and clinical data on the day of ICU admission.

1.2 LITERATURE REVIEW:

Castiglioni and others, Authors, and numerous different strategies for looking at and foreseeing Covid-19 have as of now shown positive segregation. These models have a high danger of predisposition or misconception because of exorbitant wear and ill-advised utilization of business sectors. The COVID-PRECISE group doesn't need the utilization of any type of forecast practically speaking, yet one strategy for evaluation and expectation should be approved (autonomously) in other set up regions.

Wang and Wong and their co-creators introduced COVIDNet-CT, an inside and out organization to recognize COVID-19 patients with chest CT. Utilizing COVIDX-CT test estimations, COVIDNet-CT was tried to decide reality, discernment, and PPV. We tried the standard CT picture and tracked down it to function admirably. It plans to help the far and wide use and improvement of the exploration local area. Later on, an extra investigation of the outcomes should be possible to decide the model on CT.

Xu et al., A man-made reasoning framework dependent on worldwide preparing information, gives execution estimations suitable to the chest CT-tainted with COVID-19. This inside and out research dependent on a smart arrangement of craftsmanship can be utilized for normal appraisal and during the time spent assessing illustrative data. It tends to be pretty much as helpful as an examination instrument, a proportion of clinical outcomes, or as an extra exploration device for not many individuals or an audit.

Jin et al., Co-creator of the clever COVID-19 brought about by SARS-CoV-2, have a higher rate of bleakness and mortality than SARS and MERS. Researchers have taken extraordinary steps in recognizing Covids and creating antiviral medications and antibodies. The older and individuals with medical conditions are defenseless and require care and consideration. We truly trust that the Chinese and individuals of different nations will actually want to conquer this plague quickly.

Lamiaa A.Amar a, Ashraf A.Taha a, Marwa Yu.Mohamed b, c and different scholars, The point of the review was to screen and assess the viability of the Egyptian government's COVID-19 anticipation methodology. technique. These models are essential for the Egyptian government to battle the plague before long.

Afshin Shoibi, Marjane Hodatars, Roohalla Alizadehsani and different scholars, Corona virus is an exceptionally infectious sickness that can be hazardous for the time being. It recuperates the body straight forwardly and, whenever left

untreated, can prompt genuine injury and passing. Current DL procedures are utilized to gauge, separate, and anticipate the conveyances displayed in Tables III, IV, and V.

Iman Rahimi 1, Amir H. Gandomi and different creators, COVID-19 affirms the quantity of passings in Australia, Italy and the United Kingdom. Factual innovation dependent on SIR and SEIR is given to decide the pervasiveness of the pandemic. The presumptions were far and away superior when the SIR model was presented. r^2 focuses 0.97, coordinated factors execution was the most grounded model in the UK, while Australia and Italy scored r^2 focuses 0.67 and 0.95, individually. At the point when the example rate was great, the genuine expectations of the SIR and SEIR models improved fundamentally.

Satyaki Roy¹, Shehzad Z. Sheikh^{2*} & Terrence S. Furey³, Using machine learning approaches, we conducted a detailed investigation of the clinical parameters affecting the COVID-19 results of IBD patients in this study. With Bayesian statistics and a multinomial distribution, our random sampling method is extremely generalize in terms of the number of observable features. COVID-19 mortality is connected to having UC and using 5-ASAs (sulfasalazine/mesalamine). Our findings suggest that before prescribing these drugs to patients with numerous comorbidities, gastroenterologists should balance the risks and benefits.

Ahmad Alimadadi,¹ * Sachin Aryal,¹ * Ishan Manandhar and co-authors, currently artificial intelligence, machine learning and deep learning are used to analyze COVID-19 data which includes epidemiological, clinical and genetic data. Clinical data is obtained from healthcare centers in a country. Epidemiological data pertains to the data collected by coordinating clinical data from various countries all over the world. Genetic data is the preexisting data available in various established data banks consisting genotypical and physiological makeup of patients around the world. The management and processing of this data is done by AI, ML and DL. The major problem faced in this process is the availability and collection of data. Efforts to increase collection and coordination of data between countries and should be taken up to increase the efficiency of running these algorithms. The benefits of this process can be seen in different areas of diagnosis, therapeutics, prevention of covid 19 and also in hospital operations.

Akib Mohi Ud Din Khanday¹ • Syed Tanzeel Rabani¹ and co-authors, COVID-19 has surprised the globe because of the lack of a vaccine or medicine to treat it. Various researchers are attempting to eradicate this lethal virus. TF/IDF, bag of words, and other features are derived from these clinical reports. Clinical reports are classified

into four categories using machine learning techniques. With 94 percent precision, 96 percent recall, 95 percent f1 score, and accuracy of 96.2 percent, logistic regression and multinomial Nave Bayesian classifier produce great results.

Shreyansh chordia and Yogini pawar, The data analysis, visualisation, and prediction investigation of the COVID-19 pandemic outbreak in India is presented in this report. The trend analysis reveals a quick increase in infected patients, while the prediction research reveals a significant increase in confirmed cases, recoveries, and deaths. The results show that ARIMA outperformed Prophet in predicting the counts.

Anwar Jarndal, Saddam Hussein, and different authors are concentrating on how age, number of smokers, and the quantity of individuals with diabetes add to the increment in the quantity of passings brought about by this exceptionally infectious sickness. The distributed data was found in the World Health Organization and is available to general society. GPR has observed that it works better compared to ANN in the plan/forecast of COVID-19 information.

Rajani Kumari, Sandip Kumar * and different creators, This review centers around the increment in COVID-19 in various pieces of India and gives an instance of anticipating the quantity of contaminated, rich, and biting the dust. Various direct relapse and deduction were utilized to decide the quantity of future cases. The normal qualities and the genuine qualities are all around coordinated.

Vishan Kumar Gupta, Avdhesh Gupta, and different scholars, we are concentrating on five models of AI COVID-19 and three critical components of lawful, dangerous, and rich contextual analyses in different Indian states. In light of the predominance of the memory strategy over different kinds of AI, we have observed that this is the most effective way to anticipate various things.

Sina F. Ardabili 1, Amir Mosavi 2,3, *, Pedram Gamisi 4 and different creators have observed that drawn out forecasts are wrong utilizing epidemiological models. The similar examination shows that ML from various nations ought to be the fundamental subject of future exploration. The pace of individual demise (n (passing)/ n (contamination)) additionally should be determined. The death rate is vital in looking at the quantity of beds needed in the patient consideration unit. Different examinations require the joining of the machine with the SIR/SEIR model.

1.3 OBJECTIVE OF THE PROJECT:

This study was conducted based on several objectives which are:

- Early prediction of mortality using machine learning based on typical laboratory results and clinical data registered on the day of ICU admission. To predict the covid 19 death rate.
- To evaluate and compare the performance of machine -learning techniques for detecting COVID-19 death rate based on recent data.

2. AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM AND SCOPE OF THE PROJECT:

The aim of the project is to find the covid 19 death prediction using machine learning algorithm to give the best accuracy in the project Prediction of Covid-19 Death-Rate Using Machine Learning.

- Early prediction of mortality using machine learning based on typical laboratory results and clinical data registered on the day of ICU admission. To predict the covid 19 death rate.
- To evaluate and compare the performance of machine -learning techniques for detecting COVID-19 death rate based on recent data.

2.2 HARDWARE REQUIREMENTS:

- System : Pentium i3 Processor
- Hard Disk : 500 GB
- Monitor : 15” LED
- Input Devices : Keyboard, Mouse
- Ram : 2 GB

2.3 SOFTWARE REQUIREMENTS:

- Operating system : Windows 7,10.
- Coding Language : Python

2.4 LIBRARIES USED:

The Libraries used on this project are:

- pandas Sea born
- Numpy
- Matplotlib
- Sklearn
- Scipy

3.EXPERIMENTAL OR MATERIALS, METHODS, ALGORITHMS USED

3.1 SYSTEM DESIGN AND ARCHITECTURE:

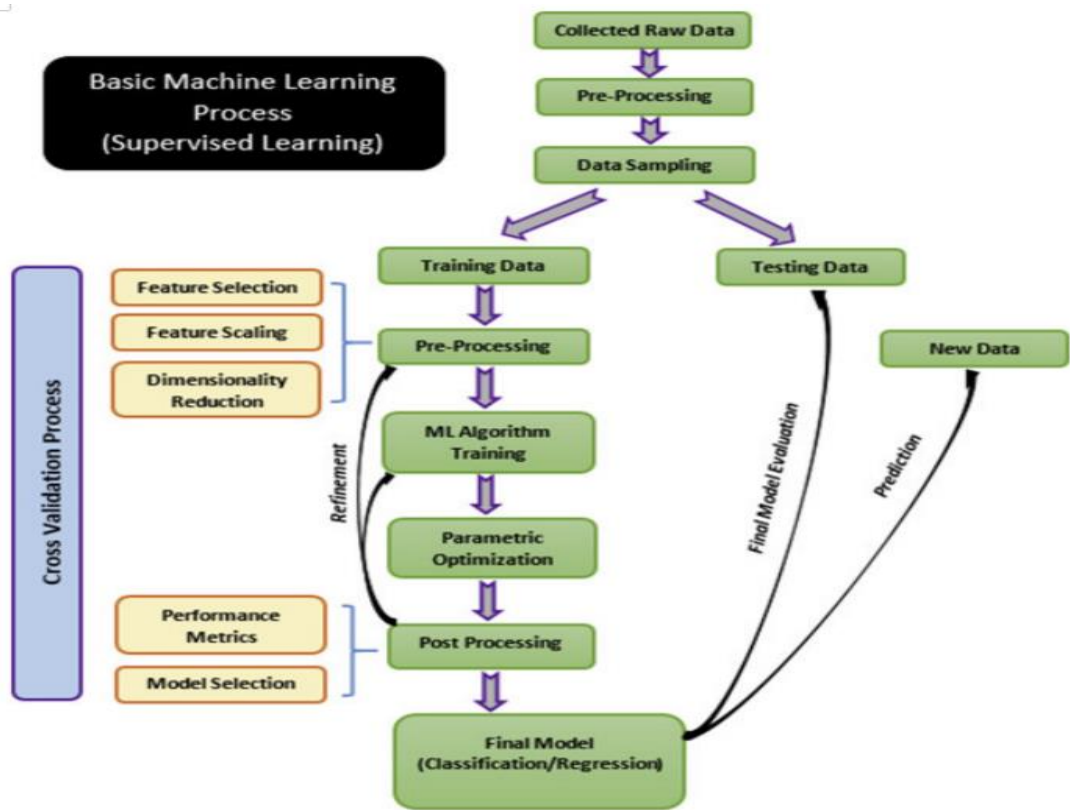


Figure 1 System Design

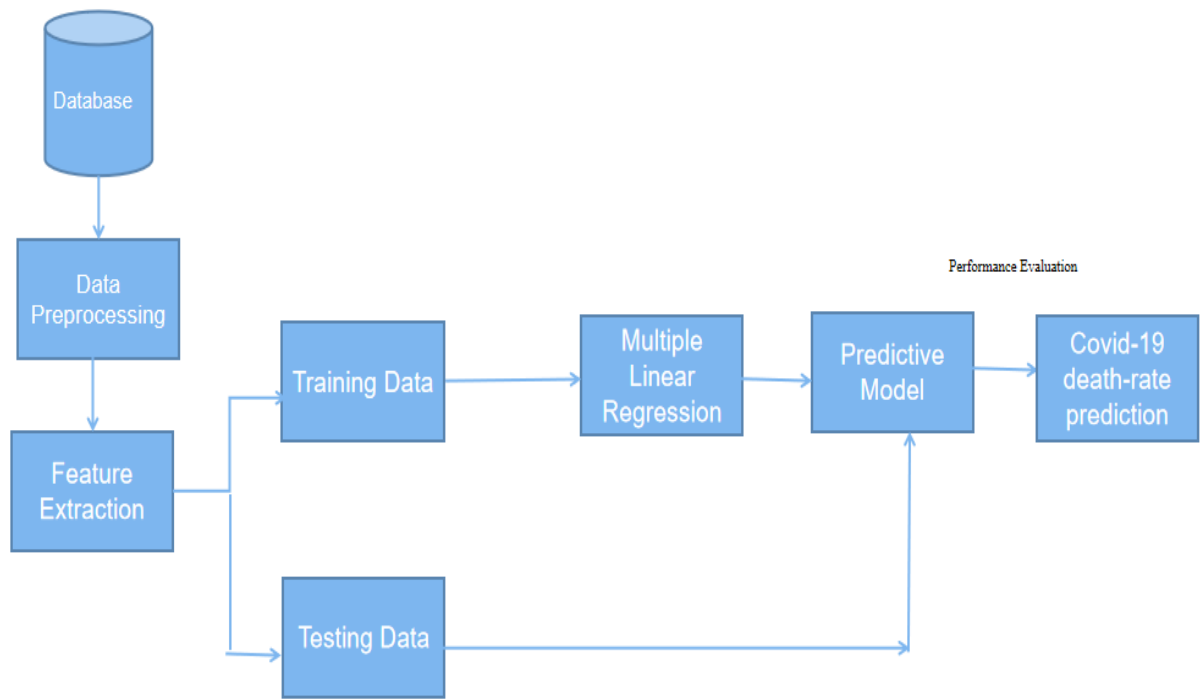


Figure 2 System Architecture

3.2 METHODS:

Setting and study data

The Israeli Ministry of Health publicly released data of individuals who were tested for SARS-CoV-2 via RT-PCR assay of a nasopharyngeal swab¹¹. The data set contains initial records, on a daily basis, of all the residents who were tested for COVID-19 nationwide. In addition to the test date and result, various information is available, including clinical symptoms, sex and a binary indication as to whether the tested individual is aged 60 years or above. Based on these data, we developed a model that predicts COVID-19 test results using eight binary features: sex, age 60 years or above, known contact with an infected individual, and five initial clinical symptoms.

The training-validation set consisted of records from 51,831 tested individuals (of whom 4769 were confirmed to have COVID-19), from the period March 22th, 2020 through March 31st, 2020. The test set contained data from the subsequent week, April 1st through April 7th (47,401 tested individuals, of whom 3624 were confirmed to have COVID-19). The training-validation set was further divided to training and validation sets at a ratio of 4:1 (Table 1).

Table 1 Characteristics of the data set and the features used by the model in this study.

The following list describes each of the dataset's features used by the model:

1. A. Basic information:
 - 1.1 Sex (male/female).
 - 1.2 Age ≥ 60 years (true/false)
- 2B. Symptoms:
 - 1.3 Cough (true/false).
 1. 4. Fever (true/false).
 2. 5. Sore throat (true/false).
 3. 6. Shortness of breath (true/false).
 4. 7. Headache (true/false).
2. C. Other information:

1. 8.Known contact with an individual confirmed to have COVID-19 (true/false).

Development of the model:

Predictions were generated using a gradient-boosting machine model built with decision-tree base-learners. Gradient boosting is widely considered state of the art in predicting tabular data and is used by many successful algorithms in the field of machine learning. As suggested by previous studies, missing values were inherently handled by the gradient-boosting predictor.

We used the gradient-boosting predictor trained with the LightGBM Python package. The validation set was used for early stopping, with AUROC as the performance measure.

To identify the principal features driving model prediction, SHAP values were calculated. These values are suited for complex models such as artificial neural networks and gradient-boosting machines.

Originating in game theory, SHAP values partition the prediction result of every sample into the contribution of each constituent feature value. This is done by estimating differences between models with subsets of the feature space. By averaging across samples, SHAP values estimate the contribution of each feature to overall model predictions.

Evaluation of the model:

The model was scored on the test set using the AUROC. In addition, plots of the PPV against the sensitivity (precision–recall curve) were drawn across different thresholds. Metrics were calculated for all the thresholds from all the ROC curves, including sensitivity, specificity, PPV and negative predictive value, false-positive rate, false-negative rate, false discovery rate and overall accuracy.

Confidence intervals (CI) for the various performance measures were derived through resampling, using the bootstrap percentile method with 1000 repetitions.

Ethics declarations:

The Tel-Aviv University review board (IRB) determined that the Israeli Ministry of Health public data set used in this study does not require IRB approval for analysis. Therefore, the IRB determined that this study is exempted from an approval.

- Methods
- Data availability
- Code availability
- References
- Acknowledgements
- Author information
- Ethics declarations
- Additional information
- Supplementary information
- Rights and permissions
- About this article

Rights and permissions:

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material.

If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit

3.2.1 EXISTING SYSTEM:

- Egypt is an African country found in the Eastern Mediterranean vicinity in line with the classification of the World Health Organization (WHO) and categorized as a lower-middle-income country with respect to a World Bank category.
- The total inhabitation of Egypt is almost 100 million individuals and almost 8% of them are exceeding 60 years old. About 1.7% of the entire inhabitation lives under the national poorness line.
- Systems of health in Egypt, like African countries, have low resources to confront the pandemic. Egypt features a physician density of 0.79 physicians/1000 individuals and a single bed capacity of 1.6 beds/1000 individuals.
- The demographic structure of Egypt highlights a particular nature that varies from other European and Asian countries where the middle age of the Egyptians is 24.6 years.
- As they were 4.23% of Egyptian individuals have almost 65 years. The infected countries' experiences appeared that elderly individuals over 60 years and individuals who have weakening maladies are most defenceless to genuine grades of COVID-19.
- In this manner, the Egyptian young may act as a defensive line to constrain the spread of the widespread around the world

3.2.2 PROBLEM STATEMENT:

- Slow process.
- The result is not efficient.
- Low accuracy
- It did not give the correct result.

3.2.3 PROPOSED SYSTEM:

- In our proposed system we have to predict the future death count by using of machine learning algorithm
- We have to collect the previous death data for the data set.
- Then we have to train the data set using of machine learning algorithm.
- In this system using the multiple linear regression algorithm.
- It will predict the perfect death rate as a result.

ADVANTAGE:

- It give perfect result.
- Good working model.
- It used for current situation.
- It basically for the common people to know the death rate.

3.3 TECHNOLOGY USED:

PYTHON

Python is a broadly useful deciphered, intelligent, object-situated, and significant level programming language. A deciphered language, Python has a structure theory that accentuates code lucidness (outstandingly utilizing white space to delimit code squares as opposed to wavy sections or catchphrases), and a punctuation that permits software engineers to communicate ideas in less lines of code than may be utilized in dialects, for example, C++ or Java. It gives builds that empower clear programming on both little and enormous scopes. Python mediators are accessible for some working frameworks. CPython, the reference usage of Python, is open source programming and has a network based advancement model, as do about the entirety of its variation executions.

CPython is overseen by the non-benefit Python Software Foundation. Python includes a powerful sort framework and programmed memory the executives. It bolsters numerous programming ideal models, including object oriented, basic, utilitarian and procedural, and has an enormous and exhaustive standard library.

Intuitive Mode Programming

Summoning the mediator without passing a content document as a parameter raises the accompanying brief –

```
$ python
```

```
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
```

```
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
```

```
Type "help", "copyright", "credits" or "permit" for more data.
```

```
>>>
```

Type the accompanying content at the Python brief and press the Enter –

```
>>> print "Hi, Python!"
```

In the event that you are running new form of Python, at that point you would need to utilize print proclamation with enclosure as in print ("Hello, Python!");. Anyway in Python rendition 2.4.3, this delivers the accompanying outcome –

```
Hi, Python!
```

Content Mode Programming

Summoning the mediator with a content parameter starts execution of the content and proceeds until the content is done. At the point when the content is done, the mediator is never again dynamic.

Let us compose a straightforward Python program in a content. Python records have augmentation .py. Type the accompanying source code in a test.py document –

Live Demo

```
print "Hi, Python!"
```

We accept that you have Python mediator set in PATH variable. Presently, attempt to run this program as follows –

```
$ python test.py
```

This creates the accompanying outcome –

```
Hi, Python!
```

Let us attempt another approach to execute a Python content. Here is the changed test.py document –

Live Demo

```
#!/usr/container/python
```

```
print "Hi, Python!"
```

We expect that you have Python translator accessible in/usr/canister index. Presently, attempt to run this program as follows –

```
$ chmod +x test.py # This is to make record executable
```

```
$/test.py
```

This creates the accompanying outcome –

```
Hi, Python!
```

Python Identifiers

A Python identifier is a name used to recognize a variable, work, class, module or other article. An identifier begins with a letter a to z or start to finish or an underscore (_) trailed by at least zero letters, underscores and digits (0 to 9).

Python doesn't permit accentuation characters, for example, @, \$, and % inside identifiers. Python is a case touchy programming language. Therefore, Manpower and labor are two unique identifiers in Python.

Here are naming shows for Python identifiers –

Class names start with a capitalized letter. Every single other identifier start with a lowercase letter.

Beginning an identifier with a solitary driving underscore shows that the identifier is private.

Beginning an identifier with two driving underscores shows an emphatically private identifier.

On the off chance that the identifier additionally finishes with two trailing underscores, the identifier is a language-characterized unique name.

Saved Words

The accompanying rundown shows the Python catchphrases. These are held words and you can't utilize them as consistent or variable or some other identifier names. All the Python catchphrases contain lowercase letters as it were.

and exec not
assert finally or
break for pass
class from print
continue global raise
def if return
del import try
elif in while
else is with
exceptlambda yield

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

if True:

 print "True"

else:

 print "False"

However, the following block generates an error –

if True:

print "Answer"

print "True"

else:

print "Answer"

print "False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks –

Note – Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –

```
days = ['Monday', 'Tuesday', 'Wednesday',  
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or ''') quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is  
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

```
print "Hello, Python!" # second comment
```

This produces the following result –

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows –

```
# This is a comment.
```

```
# This is a comment, too.
```

```
# This is a comment, too.
```

```
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
```

```
This is a multiline  
comment.
```

```
'''
```

Utilizing Blank Lines

A line containing just whitespace, potentially with a remark, is known as a clear line and Python absolutely disregards it.

In an intuitive mediator meeting, you should enter a void physical line to end a multiline explanation.

Hanging tight for the User

The accompanying line of the program shows the brief, the announcement saying "Press the enter key to exit", and trusts that the client will make a move –

```
#!/usr/container/python
```

```
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is utilized to make two new lines before showing the genuine line. When the client presses the key, the program closes. This is a pleasant stunt to keep a comfort window open until the client is finished with an application.

Different Statements on a Single Line

The semicolon (;) permits different proclamations on the single line given that neither one of the statements begins another code square. Here is an example cut utilizing the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Different Statement Groups as Suites

A gathering of individual articulations, which make a solitary code square are called suites in Python. Compound or complex articulations, for example, if, while, def, and class require a header line and a suite.

Header lines start the announcement (with the watchword) and end with a colon (:) and are trailed by at least one lines which make up the suite. For instance –

on the off chance that articulation :

```
suite
```

```
elif articulation :
```

```
suite
```

```
else :
```

```
suite
```

Order Line Arguments

Numerous projects can be hurried to furnish you with some essential data about how they ought to be run. Python empowers you to do this with - h –

```
$ python - h
```

utilization: python [option] ... [-c cmd | -m mod | document | -] [arg] ...

Choices and contentions (and comparing condition factors):

- c cmd : program went in as string (ends choice rundown)
- d : investigate yield from parser (additionally PYTHONDEBUG=x)
- E : overlook condition factors, (for example, PYTHONPATH)
- h : print this assistance message and exit

You can likewise program your content so that it ought to acknowledge different choices. Order Line Arguments is a propelled subject and ought to be concentrated somewhat later once you have experienced rest of the Python ideas.

Python Lists

The rundown is a most flexible datatype accessible in Python which can be composed as a rundown of comma-isolated qualities (things) between square sections. Significant thing about a rundown is that things in a rundown need not be of a similar kind.

Making a rundown is as basic as putting distinctive comma-isolated qualities between square sections. For instance –

```
list1 = ['physics', 'science', 1997, 2000];
```

```
list2 = [1, 2, 3, 4, 5];
```

```
list3 = ["a", "b", "c", "d"]
```

Like string lists, list lists start at 0, and records can be cut, linked, etc.

A tuple is an arrangement of permanent Python objects. Tuples are arrangements, much the same as records. The contrasts among tuples and records are, the tuples can't be changed not normal for records and tuples use enclosures, while records utilize square sections.

Making a tuple is as straightforward as putting diverse comma-isolated qualities. Alternatively you can put these comma-isolated qualities between enclosures too. For instance –

```
tup1 = ('material science', 'science', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5);
```

```
tup3 = "a", "b", "c", "d";
```

The void tuple is composed as two brackets containing nothing –

```
tup1 = ();
```

To compose a tuple containing a solitary worth you need to incorporate a comma, despite the fact that there is just one worth –

```
tup1 = (50,);
```

Like string records, tuple lists start at 0, and they can be cut, connected, etc.

Getting to Values in Tuples

To get to values in tuple, utilize the square sections for cutting alongside the list or records to acquire esteem accessible at that file. For instance –

Live Demo

```
#!/usr/container/python
tup1 = ('material science', 'science', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

At the point when the above code is executed, it delivers the accompanying outcome –

```
tup1[0]: material science
tup2[1:5]: [2, 3, 4, 5]
```

Refreshing Tuples

Getting to Values in Dictionary

To get to lexicon components, you can utilize the comfortable square sections alongside the way to acquire its worth. Following is a straightforward model –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Zara
dict['Age']: 7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result –

```
dict['Alice']:
```

Traceback (most recent call last):

```
File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
```

KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result –

```
dict['Age']: 8
dict['School']: DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();    # remove all entries in dict
del dict ;       # delete entire dictionary
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more –

```
dict['Age']:
```

Traceback (most recent call last):

File "test.py", line 8, in <module>

```
    print "dict['Age']: ", dict['Age'];
```

TypeError: 'type' object is unsubscriptable

Note – del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example –

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

Traceback (most recent call last):

```
File "test.py", line 3, in <module>
```

```
    dict = {'Name': 'Zara', 'Age': 7};
```

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

Live Demo

```
#!/usr/bin/python
```

```
tup1 = (12, 34.56);
```

```
tup2 = ('abc', 'xyz');
```

```
# Following action is not valid for tuples
```

```
# tup1[0] = 100;
```

```
# So let's create a new tuple as follows
```

```
tup3 = tup1 + tup2;
```

```
print tup3;
```

When the above code is executed, it produces the following result –

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example –

Live Demo

```
#!/usr/bin/python
```

```
tup = ('physics', 'chemistry', 1997, 2000);
```

```
print tup;
```

```
del tup;
```

```
print "After deleting tup : ";
```

```
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more –

```
('physics', 'chemistry', 1997, 2000)
```

After deleting tup :

Traceback (most recent call last):

File "test.py", line 9, in <module>

```
print tup;
```

NameError: name 'tup' is not defined

DJANGO

Django is an elevated level Python Web system that supports quick advancement and perfect, sober minded structure. Worked by experienced engineers, it deals with a great part of the problem of Web advancement, so you can concentrate on composing your application without expecting to rehash an already solved problem. It's free and open source.

Django's essential objective is to facilitate the formation of complex, database-driven sites. Django underscores reusability and "pluggability" of parts, quick improvement, and the guideline of don't rehash yourself. Python is utilized all through, in any event, for settings documents and information models.

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

Create a Project

Regardless of whether you are on Windows or Linux, simply get a terminal or a cmd incite and explore to the spot you need your task to be made, at that point utilize this code –

```
$ django-admin startproject myproject
```

This will make a "myproject" organizer with the accompanying structure –

```
myproject/  
manage.py  
myproject/  
__init__.py  
settings.py  
urls.py  
wsgi.py
```

The Project Structure

The "myproject" organizer is only your task compartment, it really contains two components –

manage.py – This document is somewhat your task neighborhood django-administrator for communicating with your venture by means of order line (start the improvement server, synchronize db...). To get a full rundown of order open through manage.py you can utilize the code –

```
$ python manage.py help
```

The "myproject" subfolder – This envelope is the real python bundle of your task. It contains four documents –

__init__.py – Just for python, treat this organizer as bundle.

settings.py – As the name demonstrates, your task settings.

urls.py – All connections of your undertaking and the capacity to call. A sort of ToC of your task.

wsgi.py – If you have to send your task over WSGI.

Setting Up Your Project

Your task is set up in the subfolder myproject/settings.py. Following are some significant alternatives you may need to set –

```
Investigate = True
```

This choice allows you to set if your undertaking is in investigate mode or not. Troubleshoot mode lets you get more data about your task's mistake. Never set it to 'Valid' for a live undertaking. In any case, this must be set to 'Genuine' on the off chance that you need the Django light server to serve static documents. Do it just in the advancement mode.

```
DATABASES = {  
'default': {  
'Motor': 'django.db.backends.sqlite3',  
'NAME': 'database.sql',  
'Client': "",  
'Secret phrase': "",  
'HOST': "",  
'PORT': "",  
}  
}
```

Database is set in the 'Database' word reference. The model above is for SQLite motor. As expressed before, Django additionally bolsters –

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql_psycopg2)

Prophet (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Prior to setting any new motor, ensure you have the right db driver introduced.

You can likewise set others choices like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE...

Since your undertaking is made and arranged ensure it's working –

```
$ python manage.py runserver
```

You will get something like the accompanying on running the above code –

Approving models...

0 blunders found

September 03, 2015 - 11:41:50

Django form 1.6.11, utilizing settings 'myproject.settings'

Beginning advancement server at http://127.0.0.1:8000/

Stop the server with CONTROL-C.

An undertaking is an aggregate of numerous applications. Each application has a target and can be reused into another undertaking, similar to the contact structure on

a site can be an application, and can be reused for other people. Consider it to be a module of your undertaking.

Make an Application

We expect you are in your venture envelope. In our fundamental "myproject" envelope, a similar organizer then manage.py –

```
$ python manage.py startapp myapp
```

You just made myapp application and like venture, Django make a "myapp" organizer with the application structure –

myapp/

__init__.py

admin.py

models.py

tests.py

views.py

admin.py – This document causes you make the application modifiable in the administrator interface.

models.py – This is the place all the application models are put away.

tests.py – This is the place your unit tests are.

views.py – This is the place your application sees are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, presently we have to enlist it with our Django venture "myproject". To do as such, update INSTALLED_APPS tuple in the settings.py document of your task (include your application name) –

```
INSTALLED_APPS = (  
'django.contrib.admin',  
'django.contrib.auth',  
'django.contrib.contenttypes',  
'django.contrib.sessions',  
'django.contrib.messages',  
'django.contrib.staticfiles',  
'myapp',
```

Making structures in Django, is extremely like making a model. Here once more, we simply need to acquire from Django class and the class properties will be the structure fields. How about we include a forms.py document in myapp envelope to contain our application structures. We will make a login structure.

myapp/forms.py

```
#-*-coding: utf-8 - *-
```

```
from django import structures
```

```
class LoginForm(forms.Form):
```

```
client = forms.CharField(max_length = 100)
```

```
secret word = forms.CharField(widget = forms.PasswordInput())
```

As observed over, the field type can take "gadget" contention for html rendering; for our situation, we need the secret word to be covered up, not showed. Numerous others gadget are available in Django: DateInput for dates, CheckboxInput for checkboxes, and so forth.

Utilizing Form in a View

There are two sorts of HTTP demands, GET and POST. In Django, the solicitation object went as parameter to your view has a characteristic called "technique" where the kind of the solicitation is set, and all information passed through POST can be gotten to by means of the request.POST word reference.

How about we make a login see in our myapp/views.py –

```
#-*-coding: utf-8 - *-
```

```
from myapp.forms import LoginForm
```

```
def login(request):
```

```
username = "not signed in"
```

```
on the off chance that request.method == "POST":
```

```
#Get the posted structure
```

```
MyLoginForm = LoginForm(request.POST)
```

```
on the off chance that MyLoginForm.is_valid():
```

```
username = MyLoginForm.cleaned_data['username']
```

```
else:
```

```
MyLoginForm = Loginform()
```

```
return render(request, 'loggedin.html', {"username" : username})
```

The view will show the consequence of the login structure posted through the loggedin.html. To test it, we will initially require the login structure layout. How about we call it login.html.

```
<html>
```

```
<body>
```

```
<form name = "structure" activity = "{% url "myapp.views.login" %}"
```

```
strategy = "POST" >{% csrf_token %}
```

```
<div style = "max-width:470px;">
```

```
<center>
```

```

<input type = "content" style = "edge left:20%;"
placeholder = "Identifiant" name = "username"/>
</center>
</div>
<br>
<div style = "max-width:470px;">
<center>
<input type = "secret phrase" style = "edge left:20%;"
placeholder = "secret phrase" name = "secret key"/>
</center>
</div>
<br>
<div style = "max-width:470px;">
<center>
<button style = "border:0px; foundation color:#4285F4; edge top:8%;
height:35px; width:80%;margin-left:19%;" type = "submit"
esteem = "Login" >
<strong>Login</strong>
</button>
</center>
</div>
</form>
</body>
</html>

```

The format will show a login structure and post the outcome to our login see above. You have most likely seen the tag in the format, which is simply to forestall Cross-site Request Forgery (CSRF) assault on your site.

```
{% csrf_token %}
```

When we have the login format, we need the loggedin.html layout that will be rendered after structure treatment.

```

<html>
<body>
You are : <strong></strong>
</body>
</html>

```

Presently, we simply need our pair of URLs to begin: myapp/urls.py

```

from django.conf.urls import designs, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
    url(r'^connection/', TemplateView.as_view(template_name = 'login.html')),
    url(r'^login/', 'login', name = 'login'))

```

While getting to `"/myapp/association"`, we will get the accompanying `login.html` format rendered –

Setting Up Sessions

In Django, empowering meeting is done in your undertaking `settings.py`, by adding a few lines to the `MIDDLEWARE_CLASSES` and the `INSTALLED_APPS` alternatives. This ought to be done while making the undertaking, however it's in every case great to know, so `MIDDLEWARE_CLASSES` ought to have –

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

Also, `INSTALLED_APPS` ought to have –

```
'django.contrib.sessions'
```

As a matter of course, Django spares meeting data in database (`django_session` table or assortment), however you can design the motor to store data utilizing different ways like: in document or in reserve.

At the point when meeting is empowered, each solicitation (first contention of any view in Django) has a meeting (dict) trait.

We should make a straightforward example to perceive how to make and spare meetings. We have manufactured a basic login framework previously (see Django structure preparing section and Django Cookies Handling part). Let us spare the username in a treat along these lines, if not marked out, while getting to our login page you won't see the login structure. Essentially, how about we make our login framework we utilized in Django Cookies taking care of progressively secure, by sparing treats server side.

For this, first lets change our login view to spare our username treat server side –

```

def login(request):
    username = 'not signed in'
    on the off chance that request.method == 'POST':
        MyLoginForm = LoginForm(request.POST)
        on the off chance that MyLoginForm.is_valid():
            username = MyLoginForm.cleaned_data['username']
            request.session['username'] = username
    else:

```

```
MyLoginForm = LoginForm()
```

```
return render(request, 'loggedin.html', {"username" : username})
```

At that point let us make formView see for the login structure, where we won't show the structure if treat is set –

```
def formView(request):
```

```
    on the off chance that request.session.has_key('username'):
```

```
        username = request.session['username']
```

```
        return render(request, 'loggedin.html', {"username" : username})
```

```
    else:
```

```
        return render(request, 'login.html', {})
```

Presently let us change the url.py record to change the url so it sets with our new view

3.4 ALGORITHMS USED:

Machine Learning: Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data.

Linear Regression: Linear Regression is a machine learning algorithm which uses a dependent variable to predict future outcomes based on one or more independent variables. It measures the association between two variables. Linear regression analysis is the most widely used of all machine learning algorithms. Simply put, linear regression is a statistical test applied to a set of data to define and quantify the relationship between the variables considered. It is simple to use and is still considered among the most powerful algorithms.

Supervised algorithms:

They require a data researcher, or data analyst, who has the knowledge of machine learning to supply the desired input and output data, in addition to delivering feedback on the accuracy of the predictions; acute during algorithm training. Data researchers determine which variables, or characteristics, should be analysed by the model and used to develop predictions. Once the training is complete, the algorithm will apply what it has learned to new data.

Supervised learning problems can be further grouped into regression and classification problems. Classification: A classification problem is when the output

variable is a category, such as “red” or “blue” or “disease” and “no disease”. Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”. Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively. Some popular examples of supervised machine learning algorithms are: Linear regression for regression problems. Random forest for classification and regression problems, Support vector machines for classification problems.

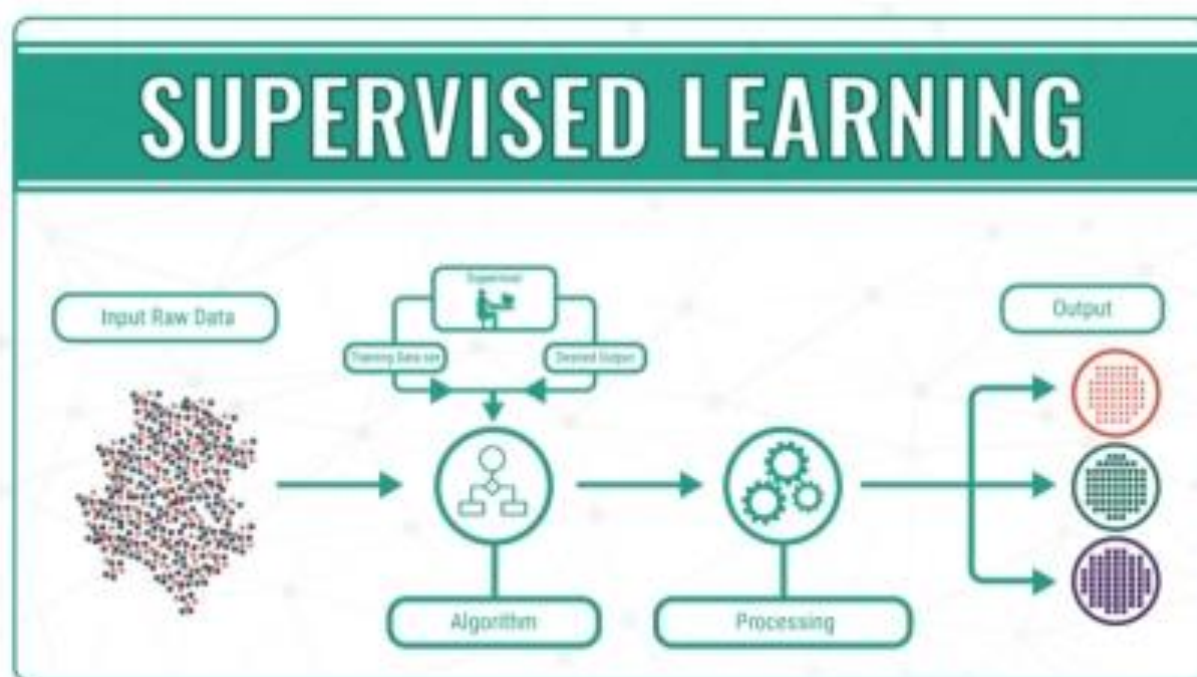


Figure 3 Supervised Learning

Unsupervised algorithms:

They do not need training with output data. Instead, they use a method called deep learning to review the data and come to conclusions. Unsupervised and learned algorithms, also known as neural networks, are used for more complex processes than supervised algorithms, which include image recognition, speech-to-text, and natural language generation. These neural networks work by first combining millions of training examples with data and automatically identifying subtle correlations between multiple variables.

Once trained, the algorithm can be used by associates to interpret new data. These algorithms become feasible only in the information age, because they require massive

amounts of data to train. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems. Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Some popular examples of unsupervised learning algorithms are: k-means for clustering problems, Apriori algorithm for association rule learning problems.

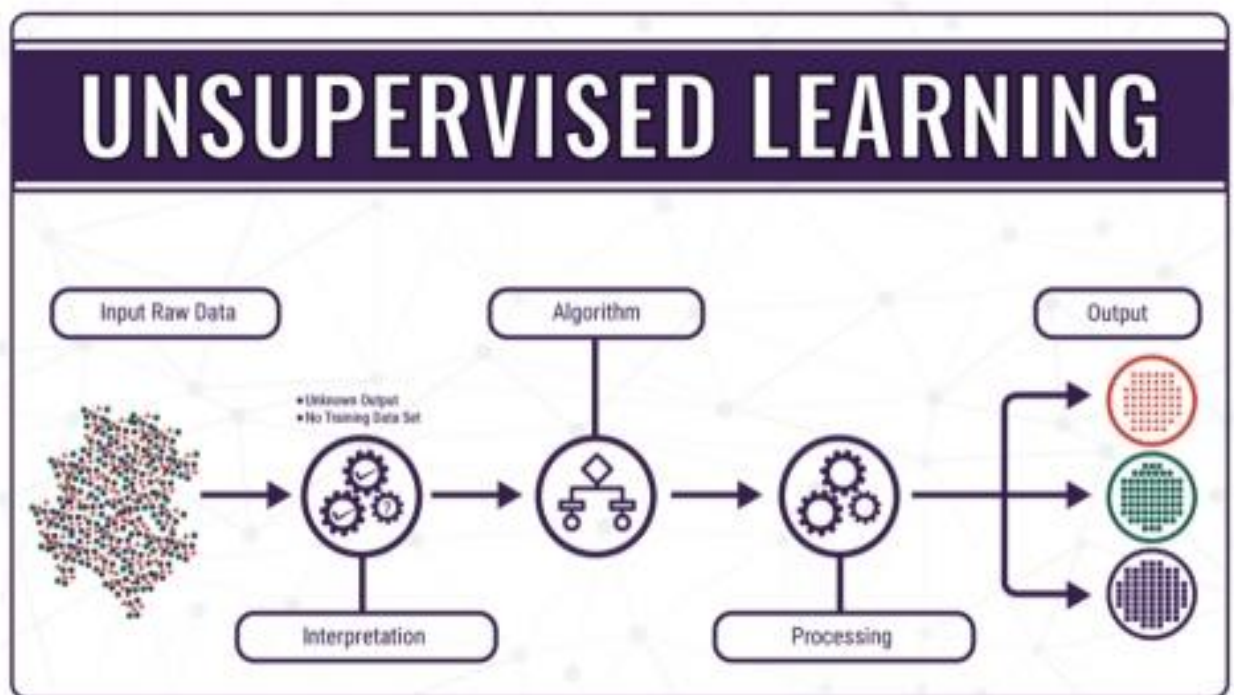


Figure 4 UN-Supervised Learning

Multiple Linear Regression:

- Multiple linear regression (MLR) is a statistical approach that is also defined as multiple regression.
- It can forecast the output of a separate variable using a variety of factors.
- Multiple regression's major purpose is to represent the linear connection among independent and dependent variables.

Random Forest

- Random Forest algorithm is derived from the random tree, which is a type of decision tree. Therefore, the first element discussed will be the Decision Tree.
- The Decision Tree creates a hierarchical division of data from the set, where a homogeneous division into classes is obtained at the tree leaf level.
- Each vertex corresponds to the selected attribute describing the instances in the set, and the edges speak about the set of values of individual attributes.
- The tree structure is usually top-down, i.e. from the root to the leaves

K-Nearest Neighbour

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

Decision tree

- Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter.
- The tree can be explained by two entities, namely decision nodes and leaves.

SVM

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning

Naive Bayes

- Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Explanation for diagram:

Algorithm steps:

- “If we have one dependent feature and multiple independent features then basically call it a multiple linear regression.”
- Dependent variable- Death rate(Y)
- Independent variable- New confirmed cases(X1)
- Independent variable2-New test cases(X2)

- $$Y=A+B_1X_1+B_2X_2$$

Step:1 Reading data-set

Step:2 Independent and Dependent variables (separate the other attributes from the predicting attribute)

Step:3 Handling categorical variables (dropping extra column)

Step:4 Splitting Data

Step:4 Applying model

Step:5 Evaluate the model

3.5 MODULES:

- Data collection
- Data pre-processing
- Feature Extraction
- Training and testing the model
- Evaluation
- Death-rate Prediction

Data set collection:

- Data set from Kaggle.
- Different cases of COVID-19 outbreak in Bangladesh.
- Date in a specific format, daily new confirmed cases, total confirmed cases, daily new deaths, total deaths, daily new recovered, total recovered, daily new tests, total tests, and active cases are the available data format for this data-set.
- All the data are available here from 1/1/21 to 19/8/21.

Data Pre-processing:

- Getting the data set - Kaggle.
- Importing libraries - pandas, numpy, matplotlib, sklearn, scipy, sea born
- Importing datasets- CSV file format.
- Finding missing data- no missing data is found in my data set.
- Dropping extra column- dropping the date, total confirmed cases, total deaths, daily new recovered, total recovered, total tests, and active cases.
- Checking if every date has only one occurrence - no duplicate rows.
- Splitting data set into training and test set- training 75% and testing 25%.

Feature Extraction:

- This is done to reduce the number of attributes in the data set hence providing advantages like speeding up the training and accuracy improvements.
- Feature extraction is the process of building a list of values (features) intended to be informative and non-redundant and facilitating the subsequent learning and generalization steps.
- In machine learning, feature extraction starts from an initial set of measured data and builds derived values that can lead to better human interpretations.

- Feature extraction is the process of transforming a large data set into a reduced set of features (also known as a feature vector) that can then be processed by an algorithm. Feature extraction is related to dimensionality reduction and involves analyzing data in terms of its shape and size.
- Date in a specific format, daily new confirmed cases, total confirmed cases, daily new deaths, total deaths, daily new recovered, total recovered, daily new tests, total tests, and active cases are the available data format for this data set.

```
date,new_confirmed_cases,new_tests,new_deaths
1/1/21,990,12103,17
2/1/21,684,9701,23
3/1/21,835,10925,27
4/1/21,910,10925,24
5/1/21,991,14462,20
6/1/21,978,15544,17
```

Figure 5 Independent and Dependent Variables

Training and testing the model:-

Model Training:

- A training model is a data set that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output.
- The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.
- This iterative process is called “model fitting”. The accuracy of the training data set or the validation data set is critical for the precision of the model.
- Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved.

- There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.
- In this module we use supervised classification algorithms like linear regression to train the model on the cleaned data set after dimensionality reduction.

Model Testing:

- In this module we test the trained machine learning model using the test data set.
- Quality assurance is required to make sure that the software system works according to the requirements. Were all the features implemented as agreed? Does the program behave as expected?
- All the parameters that you test the program against should be stated in the technical specification document.
- Moreover, software testing has the power to point out all the defects and flaws during development. You don't want your clients to encounter bugs after the software is released and come to you waving their fists.
- Different kinds of testing allow us to catch bugs that are visible only during runtime.

Splitting the data set into the Training and Test data set:

train_set_split: Split arrays or matrices into random train and

test subsets. %25 of the data set to the test set

from sklearn.model_selection import train_test_split

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2,
random_state=0)
```

Fit multiple Linear Regression model to our Train set

```
from sklearn.linear_model import LinearRegression
```

3. Create an object called regressor in the LinearRegression class...

```
regressor = linear_model.LinearRegression()
```

4. Fit the linear regression model to the training set... We use the fit method the arguments of the fit method will be training sets

```
regressor.fit(X_train,Y_train)
```

5. Predicting the Test set results:

```
y_pred= regressor.predict(X_test)
```

4.RESULTS AND DISCUSSION

4.1 MODEL ACCURACY- RESULT:

In this module, we examine the performance of the trained machine learning model in relation to F1 scores and classification error. In case the model performs poorly, we optimize the machine learning algorithms to improve its performance.

To build and deploy a generalized model we require to evaluate the model on different metrics.(Multiple linear regression and random forest).

- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error (RMSE)**
- **R squared (r2)**

```
Test Accuracy of Multiple Linear Regression: 89.16579438235044
Mean Absolute Error: 16.5242758413004
Root mean squared error test: 23.389579036512107
R squared: 0.8916579438235044
Test Accuracy of Random Forest: 87.36404954218627
Mean Absolute Error: 16.834482758620688
Root mean squared error test: 25.259678923342328
R squared: 0.8736404954218626
```

Figure 6 Accuracy Metrics

Prediction:

- "Prediction" refers to the output of an algorithm after it has been trained on a historical data set and applied to new data when forecasting the

likelihood of a particular outcome, such as whether or not a customer will churn in 30 days.

- The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.
- The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the next best action in a marketing campaign.
- Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

Covid–19 Death Rate Prediction:

```
Covid19 Death Prediction using Multiple Linear Regression Machine Learning model

Enter the confirmed_cases no. on a specific date:
620
Enter the tests no. on that date:
5240
Predicted Death no:
13
```

Figure 7 Death-Rate Prediction

Output Screenshots(Density map and Correlation Matrix map):

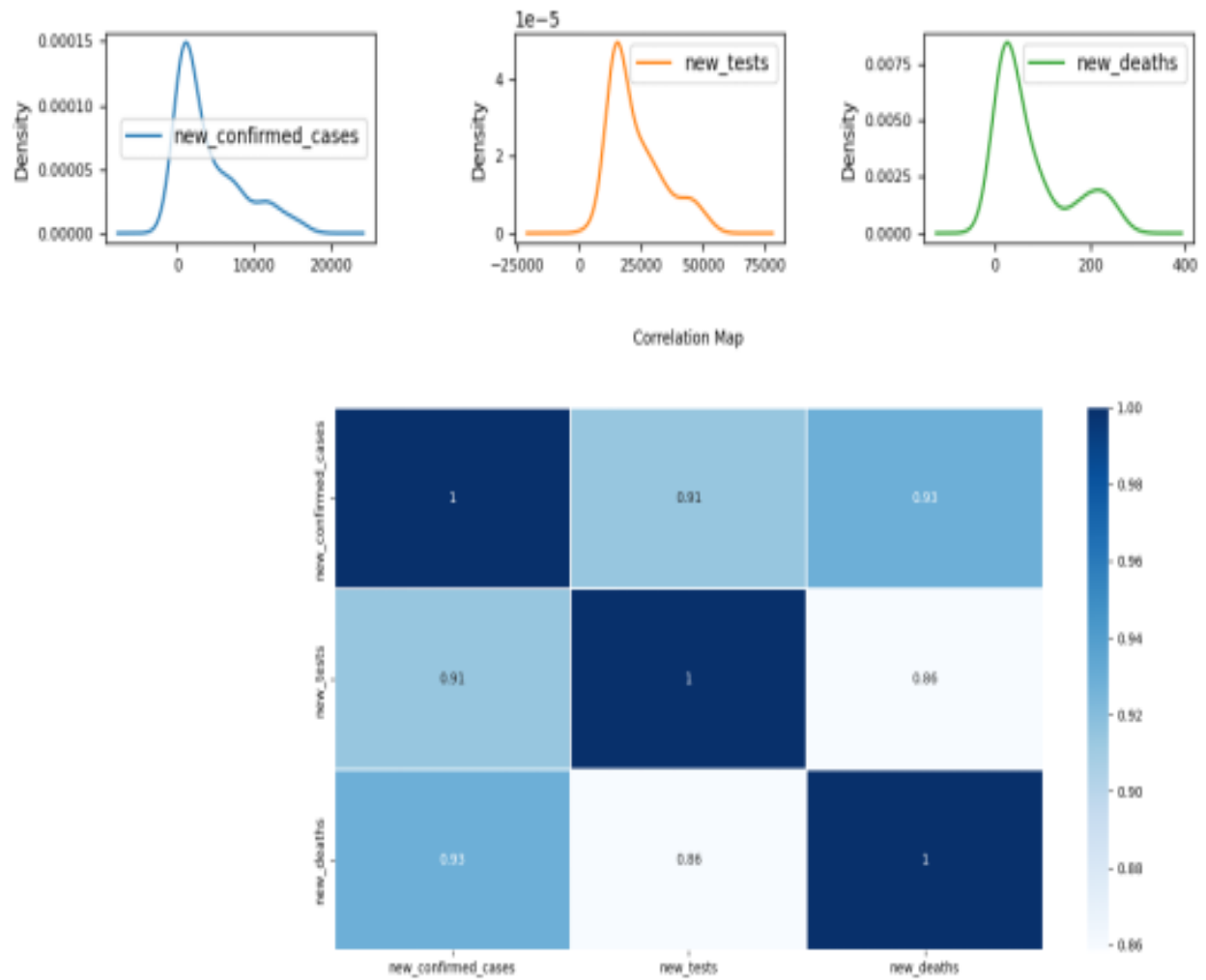


Figure 8 Density map and Correlation map

Graphical Representation(New tests vs New deaths and New cases vs New deaths):

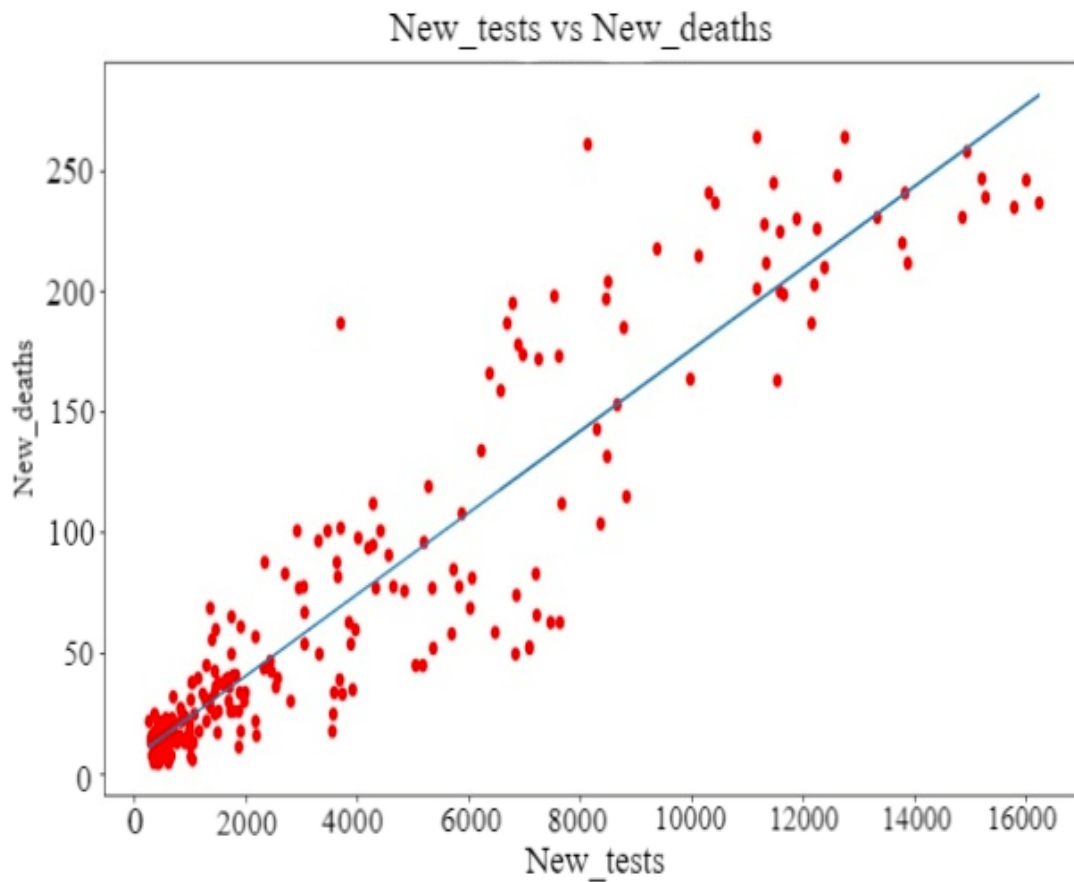


Figure 9 New tests vs New deaths

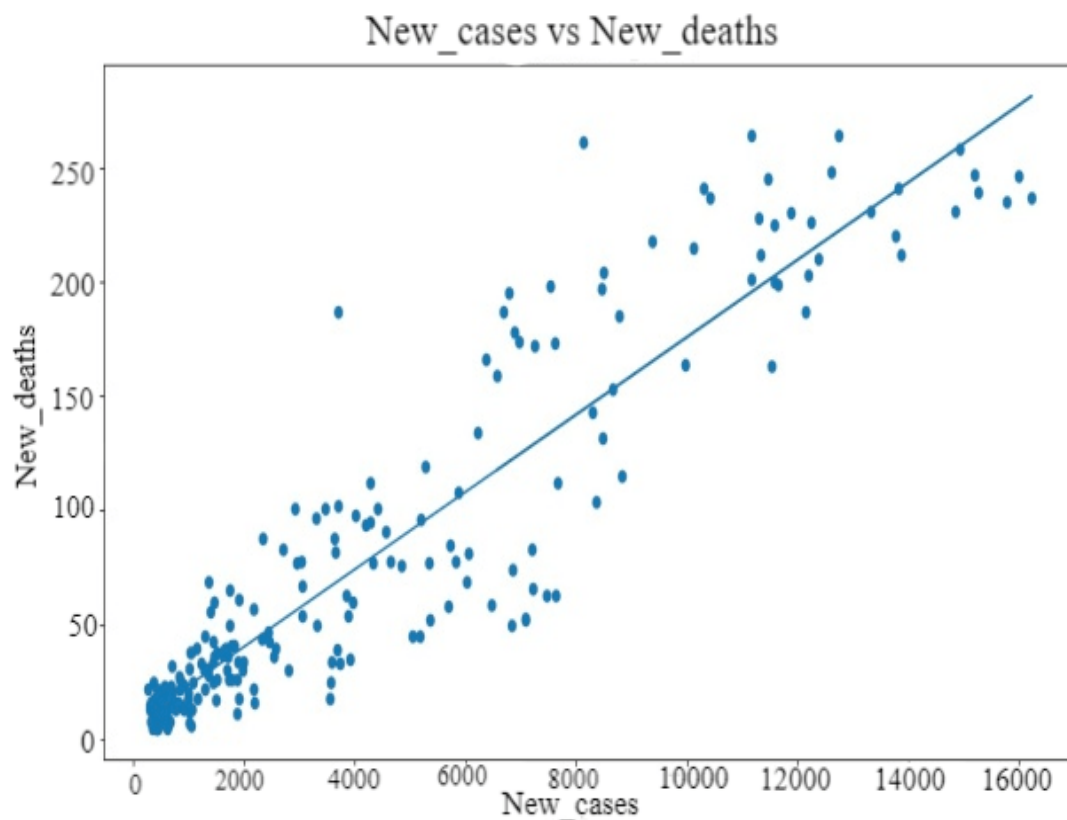


Figure 10 New cases vs New deaths

Result:

In data set 1 (Iran), all the available patient records were used to train the models. The median age of patients was 69 years with an inter-quartile range (IQR) of 54–78. The minimum and maximum ages were 20 and 98 years, respectively. One hundred fifty-three patients (65.1%) were men, and 82 (34.9%) were women. One hundred five (39.9%) were discharged from the ICU after recovery and 158 (60.1%) patients died. The most frequent comorbidities among the patients were hypertension, diabetes, and cardiovascular disorders in 94, 92, and 86 patients, respectively.

Among the 158 deceased patients, neurological disorders were the most prevalent comorbidity (42 patients, 84%). The statistical analysis and the availability of each parameter in our data-set are summarized in **Table 2**. In the RF model, the optimum point between over-fitting and efficiency was found by selecting 10 laboratory biomarkers out of 55 with the lowest KS p -values and three out of nine comorbidities with the lowest χ^2 p -values, besides demographic characteristics. The selected numerical parameters for modeling were as follows: age, blood urea nitrogen (BUN), serum creatinine level (Cr), international normalized ratio (INR), serum albumin, mean corpuscular volume (MCV), red cell distribution width (RDW), mean corpuscular hemoglobin (MCH), white blood cell count (WBC), segmented neutrophil count, and lymphocyte count.

In addition, selected categorical parameters were gender and a history of neurological, respiratory, and cardiovascular diseases. The distributions of selected numerical (age and biomarkers) and categorical (gender and preexisting conditions) variables

RESULTS - DEATH-RATE PREDICTION:

Preliminary analysis of single parameters and earlier reports of COVID-19 mortality prediction suggested that mortality outcome can be predicted by the analysis of individual biomarkers.

In this study, an approach from simple to more complex analysis was adapted. It can be argued that, compared with invasive features, mortality prediction information of non-invasive features is more independent and several major contributing features are present among them.

Conducted for the first time on Iranian patients, this study provided and compared three practical prognostic models using invasive and non-invasive data from the first day of patients' admission to predict the COVID-19 mortality.

Furthermore, the prediction power of non-invasive and invasive feature groups was evaluated across the temporal and feature number spectrum to reveal interesting results.

To forecast the COVID-19 death count, we have taken the input as the number of the new cases on a specific date and the number of the new tests on that date. We predict the outcome by using multiple linear regression, a machine learning approach.

Enter new_cases no. on a specific date:

6566

Enter new_tests no. on that date:

41014

Predicted Death no:

122

Figure 11 Final Result - Death Rate Prediction

5. CONCLUSIONS

ML-based death prediction model for COVID-19, which can aid in clinical decision making and resource allocation. Prediction of Death rate prognosis during the COVID-19 pandemic is an important concept that can reduce disease death rates by giving us insights into where and with whom to intervene. In this project we have Predicted the death no. using the data of covid 19 affected people's - 2021. The accuracy of the model is derived, Multiple Linear Regression, applied for prediction process.

6. REFERENCES

- [1] Hengyuan Kang et al, Diagnosis of Coronavirus Disease 2019 (COVID-19) with Structured Latent Multi-View Representation Learning , Published in: 2020 IEEE 10.1109/TMI.2020.2992546.
- [2] Lifang Li ; Qingpeng Zhang et al, Characterizing the Propagation of Situational Information in Social Media During COVID-19 Epidemic: A Case Study on Weibo Published in: 2020 IEEE , Accession Number 19509183 DOI: 10.1109/TCSS.2020.2980007
- [3] Di Dong et al, The role of imaging in the detection and management of COVID-19: a review , Published in: 2020 Publisher: IEEE, DOI: 10.1109/RBME.2020.2990959
- [4] Wullianallur Raghupathi et al, Big data analytics in healthcare: promise and potential, Published in: 2014.
- [5] Vinay Chamola et al, A Comprehensive Review of the COVID-19 Pandemic and the Role of IoT, Drones, AI, Blockchain and 5G in Managing its Impact , Published in: 2020 Publisher: IEEE, DOI: 10.1109/ACCESS.2020.2992341
- [6] Isabella Castiglioni, Davide Ippolito- “Machine learning applied on chest x-ray can aid in the diagnosis of COVID-19: a first experience from Lombardy, Italy” , Published on IEEE,02 February 2021.
- [7] Wang and Wong- “COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images”, Published on IEEE,11 November 2020.
- [8] Xu et al- “A Deep Learning System to Screen Novel Coronavirus Disease 2019 Pneumonia”, Published on IEEE,6 December 2020.
- [9] Jin et al- “Research on Classification of COVID-19 Chest X-Ray Image Modal Feature Fusion Based on Deep Learning”, Published on IEEE,25 August 2021.

- [10] Lamiaa A.Amar a , Ashraf A.Taha a, Marwa Yu.Mohamed b, c- “Prediction of the final size for COVID-19 epidemic using machine learning: A case study of Egypt”, Published on IEEE, 7 September 2020.
- [11] Afshin Shoibi, Marjane Hodatars, Roohalla Alizadehsani - “Automated detection and forecasting of COVID-19 using deep learning techniques: A review”, Published on IEEE, 27 July 2020.
- [12] Iman Rahimi 1, Amir H. Gandomi - “Analysis and Prediction of COVID-19 Using SIR, SEIQR, and Machine Learning Models: Australia, Italy, and UK Cases” , Published on IEEE, 3 March 2021.
- [13] Satyaki Roy¹ , Shehzad Z. Sheikh^{2*} & Terrence S. Furey³, - “A machine learning approach identifies 5-ASA and ulcerative colitis as being linked with higher COVID-19 mortality in patients with IBD” , Published on IEEE, 13 August 2021.
- [14] Ahmad Alimadadi,¹ * Sachin Aryal,¹ * Ishan Manandhar - “Artificial intelligence and machine learning to fight COVID-19” , Published on IEEE, 03 April 2020.
- [15] Akib Mohi Ud Din Khanday¹ • Syed Tanzeel Rabani¹ - “Machine learning based approaches for detecting COVID-19 using clinical text data” , Published on IEEE, 30 June 2020.
- [16] Shreyansh chordia and Yogini pawar,- “Analyzing and Forecasting COVID-19 Outbreak in India”, Published on IEEE, 28 January 2021.
- [17] Anwar Jarndal, Saddam Hussein, - “GPR and ANN based Prediction Models for COVID-19 Death Cases” , Published on IEEE, 18 November 2020.
- [18] Rajani Kumari, Sandip Kumar * - “Analysis and predictions of spread, recovery, and death caused by COVID-19 in India” , Published on IEEE, 01 February 2021.
- [19] Vishan Kumar Gupta, Avdhesh Gupta,- “Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model” , Published on IEEE, 01 February 2021.

[20] Sina F. Ardabili 1, Amir Mosavi 2,3, *, Pedram Gamisi 4- “COVID-19 Outbreak Prediction with Machine Learning” , Published on IEEE,01 October 2020.

7. APPENDIX

7.1 SOURCE CODE:

```
#import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

# Covid19deathPrediction2021UsingMultipleLinearRegression
# This Project's dataset is the covid19 affected people's data of 2021
# Our job is to predict death by building Linear regression model
# Value assigning
prev_new_cases = 0
prev_new_tests = 0
prev_new_casesCoEff = 0.0
prev_new_testsCoEff = 0.0
Prev_Death = 0
Counter= 0
decision=0

# model building
# set your input values

def setupInputValues():
    print("Covid19 Death Prediction using Multiple Linear Regression Machine
Learning model\n")
    # Taking input from the user as integer
    print("For validating the model set different new input:\n")
    new_cases = int(input("Enter the confirmed_cases no. on a specific date:\n"))
    new_tests= int(input("Enter the tests_cases no. taken on that date:\n"))
    modelBuild(new_cases, new_tests)

def modelBuild(new_cases, new_tests):
    # Reading dataset from csv file
    df = pd.read_csv("Covid19dataBD2021.csv")
```

```

X = df.iloc[:,1:3] #Getting all rows of 2nd,3rd column
y = df['new_deaths'] #target Column

regrModel = linear_model.LinearRegression() #regression model
regrModel.fit(X, y)
predictedNewDeath = int(regrModel.predict([[int(new_cases), int(new_tests)]]))
print("Predicted Death no:")
print(predictedNewDeath)

#Program starting
print("Covid19 Death Prediction using Multiple Linear Regression Machine Learning
model\n")
# Taking input from the user as integer
new_cases = int(input("Enter the confirmed_cases no. on a specific date:\n"))
new_tests = int(input("Enter the tests no. on that date:\n"))

modelBuild(new_cases, new_tests) # output checking:(6566, 41014) # 7248, 39278

```

Model accuracy code:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model, metrics
import seaborn as sn

# Reading dataset from csv file
# This dataset is the covid19 affected people's data of 2021.
# Our job is to predict death by building multivariable Linear regression model
#ACCURACY CHECKING for the model
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score

df = pd.read_csv("Covid19dataBD2021.csv")

```

```

X = df.iloc[:, 1:3] # Getting all rows of 2nd,3rd column
y = df['new_deaths'] # target Column

df.plot(kind='density' , subplots=True , layout=(4,4) , sharex=False ,
        fontsize=8 , figsize=(10,8))
plt.tight_layout()
plt.title('Density map', y=1.1)

plt.show()

sn.heatmap(df.corr(), annot=True, cmap='Blues',linewidth=0.2)
fig = plt.gcf()
fig.set_size_inches(10,8)
plt.title('Correlation Map', y=1.1)
plt.show()

plt.show()
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=2)
regrModel = linear_model.LinearRegression() # regression model
#training data fitting to the model
regrModel.fit(x_train, y_train)
# Predicting the Test set result
y_pred = regrModel.predict(x_test)

#ACCURACY Checking

print('Test Accuracy of Multiple Linear Regression: ', regrModel.score(x_test, y_test)*
100)

# importing r2_score module
from sklearn.metrics import r2_score

```

```

from sklearn.metrics import mean_squared_error
# predicting the accuracy score
score=r2_score(y_test,y_pred)
print(f'Mean Absolute Error: {metrics.mean_absolute_error(y_test, y_pred)}')
print(f'Root mean squared error test: {np.sqrt(metrics.mean_squared_error(y_test,
y_pred))}')
print(f'R squared: {(r2_score(y_test, y_pred))}')

# Fitting Random Forest Regression to the Training set
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)

regressor.fit(x_train, y_train)
# Predicting the Test set results
y_prediction = regressor.predict(x_test)
# Evaluating the Algorithm
print('Test Accuracy of Random Forest: ', regressor.score(x_test, y_test)* 100)
from sklearn import metrics
print(f'Mean Absolute Error: {metrics.mean_absolute_error(y_test, y_prediction)}')
print(f'Root mean squared error test: {np.sqrt(metrics.mean_squared_error(y_test,
y_prediction))}')
print(f'R squared: {(r2_score(y_test, y_prediction))}')

```

Data visualization :

```

#import

#This dataset is the covid19 affected people's data of 2021
#Our job is to predict death by building multiple Linear regression model
#Here is the data visualization of the new cases vs new deaths and new tests vs
new deaths separately

#Reading dataset from csv file
df = pd.read_csv("Covid19dataBD2021.csv")

```

```

#display 10 sample data of the csv file
print (df.head())

x = df['new_cases'] #Taking input column
y = df['new_deaths'] # target Column

slope, intercept, r, p, std_err = stats.linregress(x,y)
def myfunc(x):
    return slope * x + intercept

line1 = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, line1)
plt.title("new_cases vs new_deaths")
plt.xlabel("new_cases") #Taking input column
plt.ylabel("new_deaths")
plt.show()

x2 = df['new_tests'] #Taking input column
y = df['new_deaths'] # target Column

slope, intercept, r, p, std_err = stats.linregress(x, y)
def myfunc(x):
    return slope * x + intercept

line2 = list(map(myfunc, x))

plt.scatter(x, y, color="red")
plt.plot(x, line2)
plt.title("new_tests vs new_deaths")
plt.xlabel("new_tests")
plt.ylabel("new_deaths")
plt.show()

```

7.2 SCREENSHOTS:

DATA VISUALISATION:

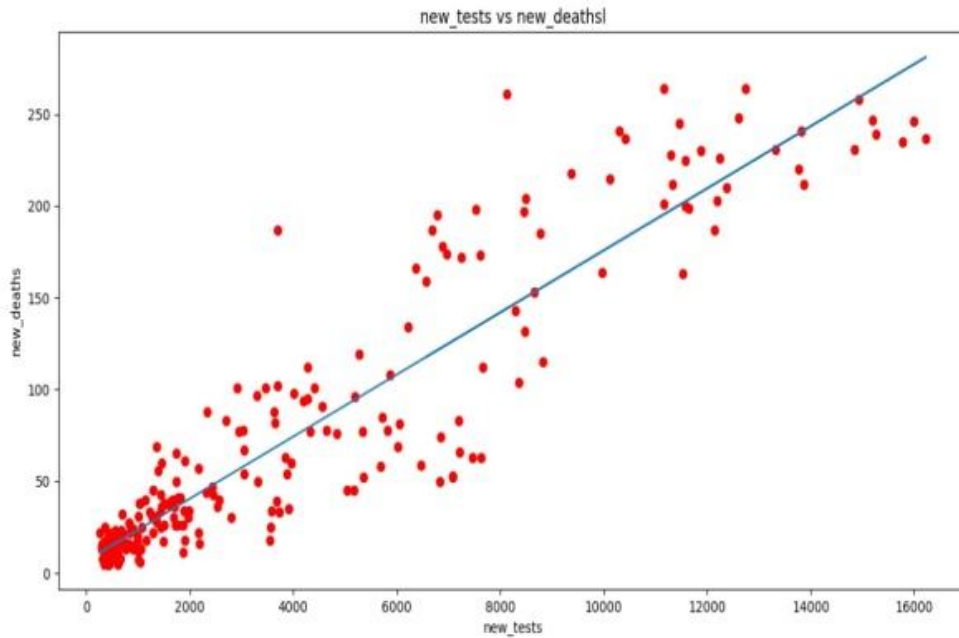


Figure.no 1 - New tests vs New deaths

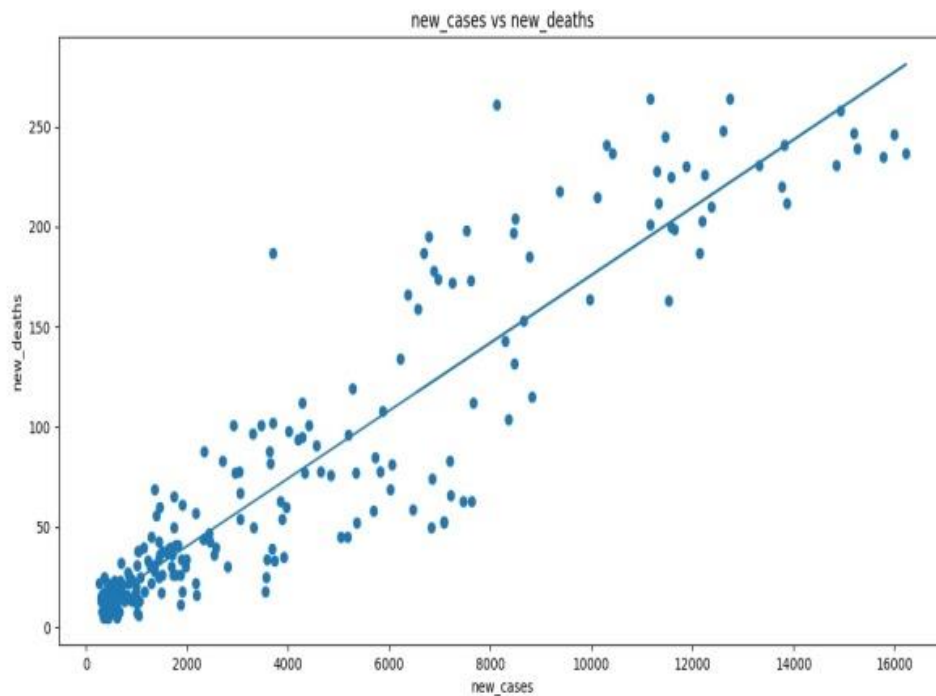


Figure.no 2 - New cases vs New deaths

MODEL ACCURACY:

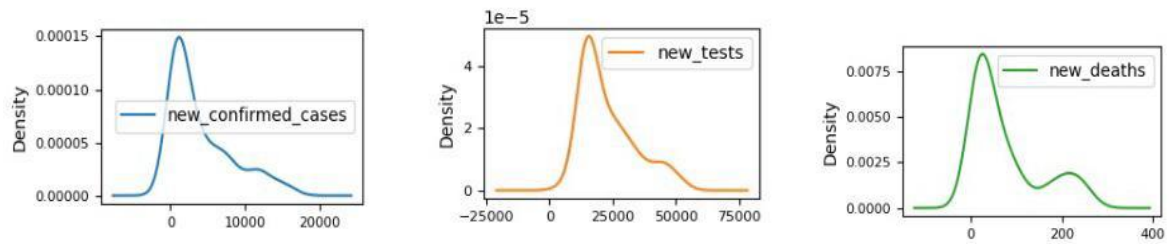


Figure.no 3 - Density map

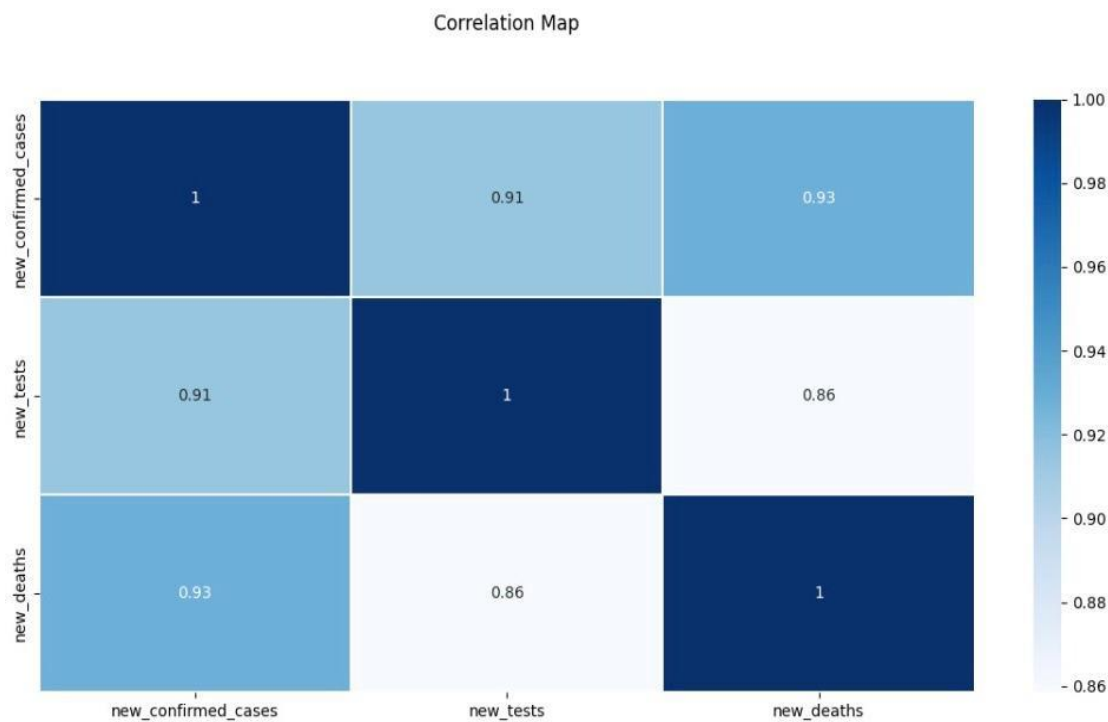


Figure.no 4 - Correlation map

```
Test Accuracy of Multiple Linear Regression: 89.16579438235044
Mean Absolute Error: 16.5242758413004
Root mean squared error test: 23.389579036512107
R squared: 0.8916579438235044
Test Accuracy of Random Forest: 87.36404954218627
Mean Absolute Error: 16.834482758620688
Root mean squared error test: 25.259678923342328
R squared: 0.8736404954218626
```

Figure.no 5 - Accuracy metrics

RESULTS:

```
Covid19 Death Prediction using Multiple Linear Regression Machine Learning model

Enter the confirmed_cases no. on a specific date:
620
Enter the tests no. on that date:
5240
Predicted Death no:
13
```

Figure.no 6 - Covid-19 Death Rate Prediction

