Analysis and Prediction of COVID-19 using Machine Learning Algorithms

A thesis submitted in partial fulfilment of

the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINNERING

By

K.LAKSHMI LASYA (180030291)

A.LAVANYA (180030307)

R.AKARSHA (180030471)

D.LAHARI (180031282)

Under the Esteemed Guidance of

Dr. K .BHANU PRAKASH

Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K L (Deemed to be) University

Green Fields, Vaddeswaram ,

Guntur District – 522502

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is certify that the minor project based report entitled **“ANALYSIS AND PREDICTION OF COVID-19 USING MACHINE LEARNING ALGORITHMS”** is a bona-fide work done and submitted by D LAHARI (180031282), K LAKSHMI LASYA (180030291), A LAVANYA (180030307), R.AKARSHA(180030471) in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in Department of Computer Science Engineering, K L (Deemed to be University), Guntur District during the academic year **2021-2022**.

Dr . K . BHANU PRAKASH
PROJECT GUIDE
DEPARTMENT OF CSE
K L (Deemed to be University)

Mr .V. HARI KIRAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE
K L (Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING

DECLARATION

This is certify that the minor project based report entitled **“ANALYSIS AND PREDICTION OF COVID-19 USING MACHINE LEARNING ALGORITHMS”** is a bona-fide work done and submitted D LAHARI (180031282), K LAKSHMI LASYA (180030291), A LAVANYA (180030307), R.AKARSHA(180030471) in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in Department of Computer Science Engineering during the academic year **2021-2022**.

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our wish to express my sincere thanks to all those who has assisted us in one way or the other for the completion of my project.

Our greatest appreciation to my guide **Dr .K .BHANU PRAKASH** Associate Professor, Department of Computer Science and Engineering which cannot be expressed in words for his tremendous support, encouragement and guidance for this project.

We express our gratitude to **Mr. V. Hari Kiran,** Head of the Department for Computer Science and Engineering for providing us with adequate facilities, ways and means by which we are able to complete this project based Lab.

We thank all the members of teaching and non-teaching staff members, and also who have assisted me directly or indirectly for successful completion of this project.

Finally, I sincerely thank my parents, friends and classmates for their kind help and co-operation during my work.

K.LAKSHMI LASYA (180030291)

A.LAVANYA (180030307)

R.AKARSHA (180030471)

D.LAHARI (180031282)

CONTENTS

ABSTRACT

The devasting spread brought about by Severe Acute Respiratory Disorder - Coronavirus (SARS-CoV-2) which is otherwise called COVID-2019 has carried worldwide danger to our general public. Every country is making immense efforts to stop the spread of the deadly disease through the use of finance, infrastructure and data sources, as well as protective devices, life-risk treatments, as well as other sources. Researchers studying artificial intelligence focus their skills to create mathematical models for studying the scourge of this disease using and shared data. In order to improve the wellbeing of our society This article proposes using model of deep and machine-learning to understand its daily exponential behavior, as well as the prediction of the future impact of the COVID-2019 across nations using the live data of the Johns Hopkins dashboard.

INTRODUCTION

In the age of automation artificial intelligence, data science play a crucial roles are able to handle their duties and provide patient treatment. Every health care organization works to create an electronic system that will be able to tackle the issues in the field of health care. Researchers are studying machines learning (ML) to come up with intelligent solutions to detect and treat diseases. ML can recognize sicknesses and viral diseases all the more exactly, to ensure that the disease can be detected in a timely manner which means that the risky phases of disease are avoided and, consequently, there are less patients. The same way ML could be used to computerize the most common way of anticipating the COVID-19 disease. It can also assist in predicting future levels of infection of COVID-19.

The chapter we're discussing present ways to forecast future cases based on current information. ML techniques are utilized and two options that can be used to predict the likelihood of contracting the disease and another for forecasting the amount of positive cases are presented. Two classifiers such as random forest and an the extra-tree classifier (ETC) have been selected with an accuracy greater than 90 percent. The one with the highest accuracy is ETC. ETC is the most accurate, with 93.62 percent accuracy. These findings are used to decide on corrective actions by various authorities. The existence of methods to predict the spread of infectious diseases can aid in the fight against diseases that cause infection, such as COVID-19.

This pandemic continues to pose a challenge to medical systems around the world in numerous ways, such as the sharp increase in the demand for hospital beds as well as critical shortages in medical equipment and many healthcare professionals have been affected themselves. This is why the ability to take rapid clinical decision-making and efficient utilization of healthcare resources is essential. The most dependable indicative test for COVID-19, using reverse transcriptase polymerase chain responses (RT-PCR) is for some time been unavailable in arising nations. This prompts higher rate of contamination and deferral in going to pivotal preventive lengths.

Effective screening allows for rapid and effective diagnosis of COVID-19, and could reduce the stress for healthcare facilities. Prediction models that incorporate a variety of aspects to assess the risk of infection have been created with the intention to assist medical personnel around the world when it comes to triaging patients particularly when faced with limited resources for healthcare. The models incorporate elements like computers tomography (CT) scans2,3,4,5,6 and clinical symptoms7, as well as lab tests8,9 and a combination of these characteristics10. But, previous models were built on hospitalized patients' data and therefore aren't effective in identifying SARS-CoV-2 within the general population.

EXISTING SYSTEM

Past works neglected to deal with long haul conduct changes. Past approaches have not considered apparatus level utilization subtleties. Activity covers has not been thought of. Determining precise forecasts has been a test up until this point. A portion of the methodologies are viewed as great at trial set up yet have not considered genuine world situations. The impact of fleeting relationship has not been thought of.

PROPOSED METHODOLOGY

This paper seeks to fill the gap left by existing healthcare systems by using the to determine an outcome that is most likely for an individual patient, based upon their health conditions, This study will allow researchers to continue to work to develop an approach that blends the processing of demographics of patients along with travel, as well as personal health information with image data (scans) to improve the estimation of COVID-19 health outcomes.

Semisupervised ML algorithms are ones that fall into the two categories of unsupervised and supervised learning. This type of algorithm employs the data of both labels and unlabeled for training , usually with a tiny amount of labeled data as well as an enormous quantity of unlabeled data. This kind of technique is utilized to increase the precision of learning [[2021] and [[22]]].

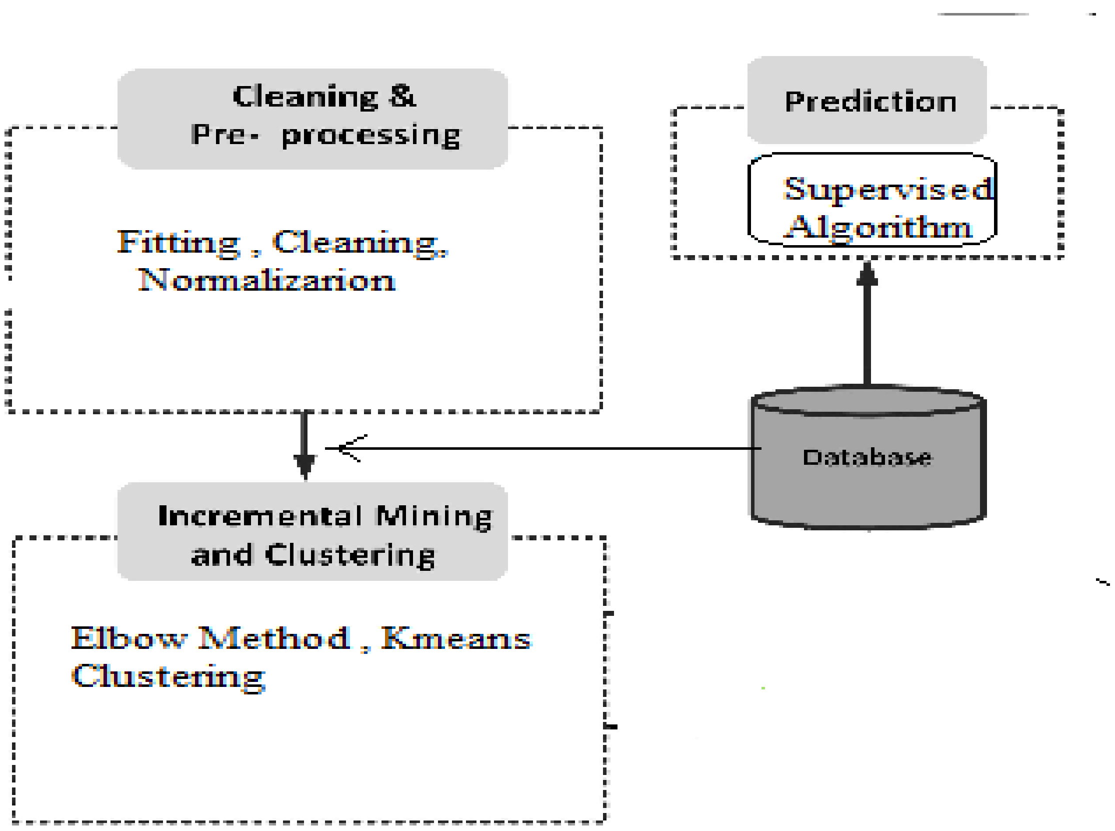
This paper is designed to fill in the gaps that existing healthcare systems have left by using machines-learning (ML) algorithm that process simultaneously both healthcare and travel information as well as other variables of COVID-19 positive patient in Wuhan to find out the outcome which is most likely to be for a particular patient in relation to their health conditions along with their travel history and duration of time required for reporting the event analysing patterns found in prior records that patients have submitted. Our contribution includes:

The study compared a variety of methods for processing patient information and found the Boosted Random Forest algorithm as the most efficient way to process the data. Furthermore, it utilized an algorithmic grid-search to refine the hyper-parameters used by the Boosted Random Forest algorithm to improve the efficiency.

Our study removes the necessity of reviewing the algorithms utilized to process the COVID-19 patient's information.

The study should enable researchers to continue to create a method which combines the analysis of demographic data of patients and travel information, as well as personal health information along with images (scans) to enhance the estimation of the COVID-19.

SYSTEM ARCHITECTURE


Cleaning & Pre- processing
Fitting , Cleaning, Normalizarion
Prediction
Supervised Algorithm
Database
Incremental Mining and Clustering
Elbow Method , Kmeans Clustering

LITERATURE SURVEY

A test set was utilized to determine the effectiveness on Piacenza. Piacenza score. To enhance the statistical value of the outcomes, bootstrapping has been utilized to generate randomly 100 testing sets from test group. Additionally an external validation group is being thought of to confirm that the Piacenza scores.

Performance of Piacenza score was assessed by comparing its accuracy and discrimination. The discrimination capability was measured by formulating the receiver operating characteristic (ROC) curve of the test group as well as the AUC along with its 95% confidence interval. Additionally the positive predictive value (NPV), the negative (NPV) and the value of positive prediction (PPV) and the sensitivity, accuracy to specificity, F1 as well as F2 scores were calculated.

The metrics were calculated using an F2 threshold that was determined by maximising F2 score. F2 score. The ability to measure calibration was discovered from the so-called calibration plots that compare expected and observed outcomes, as well as the associated uncertainties. This Brier index can be used to assess the capacity in machine learning predict and stratify the outcomes that are observed. This Brier index is the mean-squared variation between predicted and observed outcomes. It is a range of 0 to 1 with 0 being the lowest.

BasePaper:

A ssessing the seriousness of COVID-19 is vital to decide the fittingness of relief procedures and to empower getting ready for medical services needs as pestilences unfurl. Be that as it may, rough case casualty proportions acquired by partitioning the quantity of passings by the quantity of cases can be misleading.9,10 First, there can be a time of 2–3 weeks between an individual creating side effects, the case thusly being identified and announced, and perception of the last clinical result. During a developing pandemic, the last clinical result of the greater part of the announced cases is ordinarily obscure. Essentially isolating the aggregate revealed number of passings by the total number of announced cases will accordingly think little of the genuine case casualty proportion ahead of schedule in an epidemic.9–11

This impact was seen in past pandemics of respiratory microorganisms, including extreme intense respiratory disorder (SARS)12 and H1N19 flu, and as such is broadly perceived. Consequently, a considerable lot of the assessments of the case casualty proportion that have been gotten to date for COVID-19 right for this effect.13–16 Additionally, nonetheless, during

the dramatic development period of a pestilence, the noticed delays between the beginning of manifestations and result (recuperation or demise) are edited, and gullible assessments of the noticed occasions from side effect beginning to result give one-sided evaluations of the real appropriations. Overlooking this impact will in general predisposition the assessed case casualty proportion downwards during the early development period of a pandemic.

The WHO coordinates and facilitates global wellbeing, battling transferable infections through reconnaissance, readiness and reaction, and applying GIS innovation to this work. On 26 January 2020, the WHO revealed its ArcGIS Operations Dashboard for COVID-19, which additionally guides and records Covid cases and absolute number of passings by nation and Chinese territory, with educational boards about the guide and its information resour [17]. Before 18 February 2020, the WHO and JHU CSSE dashboards made them interest diferences. Each had an inconceivably diferent all out case count . Te WHO dashboard refected research facility confrmed cases, while JHU CSSE included cases analyzed utilizing a side effect exhibit in addition to chest imaging (representing approximately 18,000 extra reports). Be that as it may, starting at 19 February 2020, the two dashboards are in a state of harmony, showing comparative all out case counts.

GIS can help with refreshing and planning wellbeing occasion predominance, and is utilized as a supporting apparatus for reconnaissance and as a dynamic device by which to control medical issue and infection. GIS can be utilized to plan the geological appropriation of the pervasiveness of an illness, the patterns of infection transmission, and spatially model the ecological parts of illness event. The made geo-data set of wellbeing in Jeddah joins three kinds of ailments: diabetes, asthma, and circulatory strain. For the primary kind of spatial investigation of these information, ArcGIS (created by ESRI) information arrangement devices were utilized. These instruments can assist specialists with understanding the spatial circulation and grouping of medical issue. Checking out the yield of wellbeing occasion grouping in Figures 1–3, asthma patients are profoundly amassed in the north east of the city. These pieces of the city are thought of profoundly metropolitan created areas, delivering enormous number of asthma patients, as affirmed in examinations that have recommended that there is more asthma in metropolitan than in country regions in many regions of the planet. Early investigations from Africa (South Africa, Ethiopia, Kenya, and Ghana) revealed that populaces living in country regions (i.e., those not presented with the impacts of a metropolitan or western way of life) encountered an extremely low weight of hypersensitive illness, with a conventional, provincial method of living giving a potential defensive cover. Comparative examinations from Asia (China, Japan, Korea, India, and Saudi Arabia) affirmed the metropolitan provincial slope because of openness to various allergens, air contamination, prosperity, and diet in the advancement of asthma and hypersensitivity [29]. In the interim, hypertension and diabetes patients are found more in the focal and northern city areas. These city areas are considered as high-thickness areas, affirming that examples of diabetes and hypertension patients follow the populace thickness design in Jeddah. These are the consequences of the underlying investigation for characterizing the spatial dispersion of ailments..

Hardware and Software Requirements:

Hardware:

1. OS – Windows 7,8 or 10 (32 or 64 bit)
2. RAM – 4GB

Software:

1. Python IDLE
2. Anaconda – Jupiter Notebook

Python

Python is a programming language, which implies that it is a language that both individuals and PCs can understand. Python was created by a Dutch programmer named Guido van Rossum, who created the language to solve some problems he found in the encodings of the time.

Python is an undeniable level programming language deciphered for widely useful programming. Developed by Guido van Rossum and first published in 1991, Python has a plan theory that emphasizes the importance of code and punctuation that allows software engineers to communicate ideas in fewer lines of code. , making exceptional use of white critical spaces. It provides developments that allow clear planning on small and large areas.

Python includes a powerful type frame and a programmed memory card. It supports many ideal programming models, including established, basic, useful, and procedural objects, and has a huge and comprehensive standard library.

Python mediators are accessible for some work frameworks.C Python, the benchmark execution for Python, is open source programming and has a local area-based progression model, as well as pretty much its variation executions in general. C Python is supervised by the Python Software Foundation without any benefit.

You can use Python for just about anything

A huge advantage of learning Python is that a very useful language can be applied in a wide variety of businesses. The following are just the most normal fields of all that Python has traced its use to:

Data science

Scientific and digital recording

Web advancement

Computer illustrations

Shift core game events

Mapping and surveying (GIS programming)

The Python environment is becoming long term and is increasingly equipped for measurable scrutiny.

This is the best compromise between scale and complexity (in the formulation or management of information).

Python emphasizes utility and meaning.

Python is used by software engineers who need to further examine information or apply measurable methods (and by information science developers)

There are many Python logic packages to represent information, the AI, normal language management, examining complex information and just getting started. These elements make

Python an incredible tool for logical logging and a viable option for enterprise bundles like MatLab. The most well-known libraries and tools for information science are:

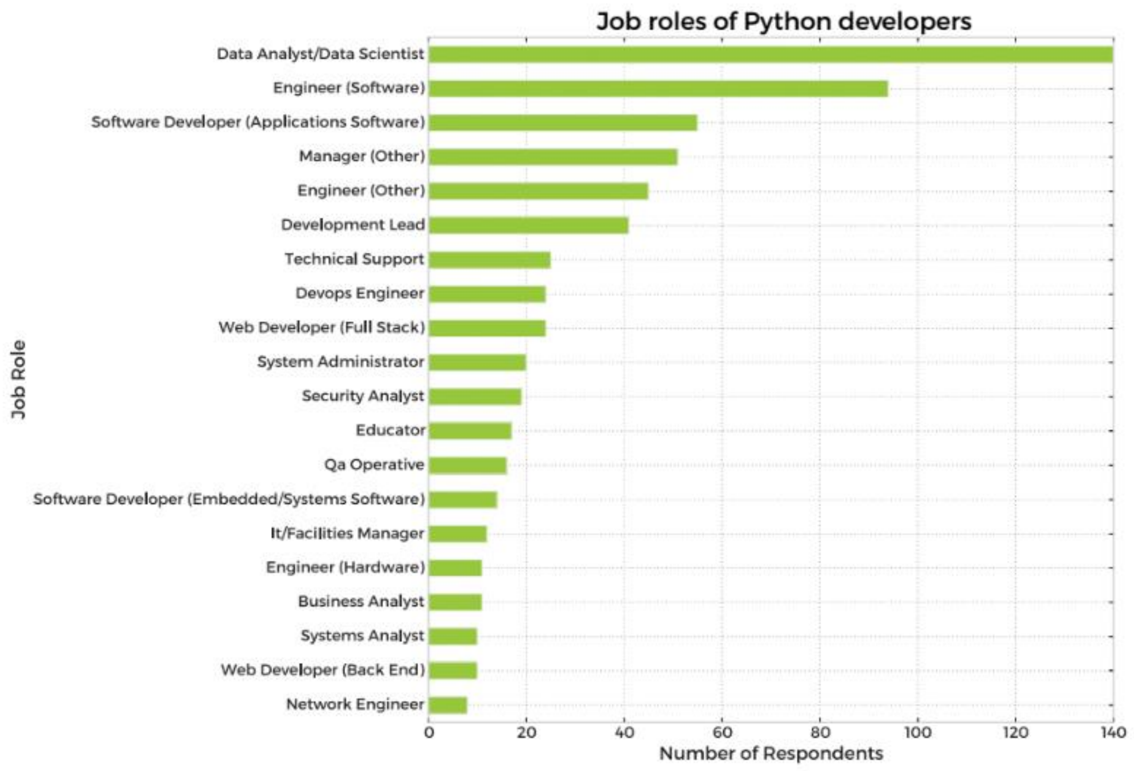
Panda: a library for information control and medical examination.The library provides information and draft tasks to control math tables and time series.

NumPy: The base package for logical figuration with Python, which adds support for huge multidimensional clusters and networks, as well as a huge library of meaningful level digital capabilities for working on these exhibits.

SciPy: A library used by researchers, researchers and architects who perform logical records and specialized calculations.

As a free, cross-sectional, widely useful, and meaningful programming language, Python has generally been adopted by mainstream researchers. Researchers value Python for its accurate and competent linguistic structure, moderate-level learning and adaptation expectations, and coordination with different dialects (eg, C / C ++).

Because of this fame, there are many Python logical sets for information representation, AI, normal language handling, examining complex information, etc. These variables make Python an incredible tool for logical processing and a viable option for enterprise bundles like MatLab.



Mlpy

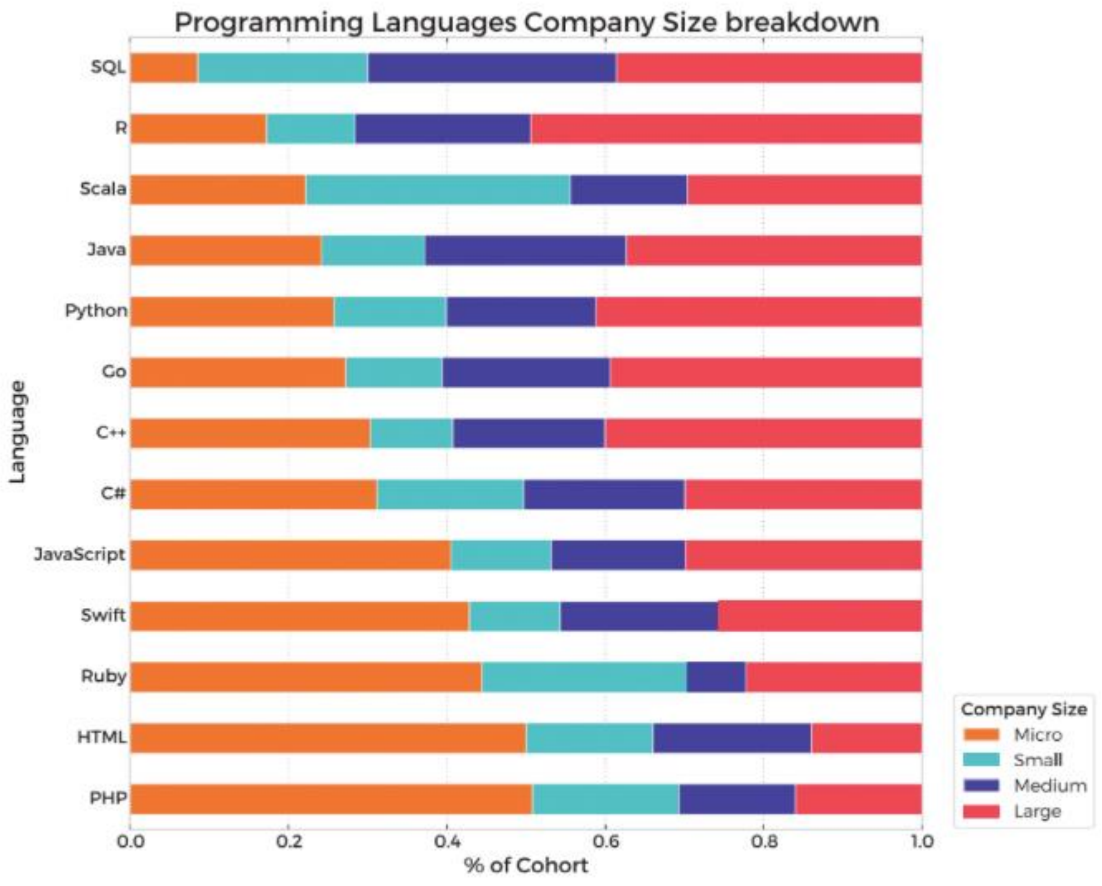
Mlpy is a machine learning library built on top of NumPy / SciPy, the GNU science libraries. Mlpy provides a wide range of machine learning methods for supervised and unsupervised problems and aims to find a reasonable compromise between modularity, maintainability, reproducibility, usability and efficiency. matplotlib Matplotlib is a Python 2D plotting library that produces publication-quality figures in a variety of paper formats and interactive environments on platforms. Matplotlib allows you to generate graphs, histograms, power spectra, bar graphs, error graphs, scatter plots and more.

NumPy

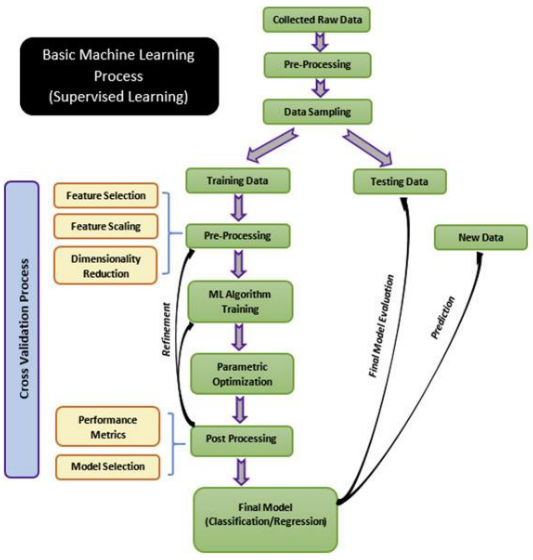
NumPy is the foundational package for scientific computing with Python, adding support for large arrays and multidimensional arrays, as well as a large library of high-level mathematical functions to operate on these arrays.

NetworkX

NetworkX is a graphics library that helps you create, manipulate, and study the structure, dynamics, and functions of complex networks.



Modules



Cough, fever fatigue, muscle pain, and malaise were the most frequently reported symptoms reported by patients with data available. The data set is comprised of columns, with the data being Date string, Numeric, and Date type. Additionally, we have categorical variables included in the data. Because it is the case that ML algorithm requires that all information that is used as inputs to be in numerical format, we carried out label-encoding on the categorical variables. This assigns a value for each categorical number within the column. The dataset is comprised of many missing values, which can result in an error when used directly as input. Therefore, we fill in the values that are missing by using "NA." Certain patient data records are missing values in as well"recov" as well as the "death" and "recov" columns. These patient records were separated from the primary dataset, and put in the test data and the rest of the records were compiled to form the training dataset. The data set also contains columns with the format of a date. Because the columns in the data format do not have direct use features engineering was applied. A new column was filled with the relevant (hosp_vis--sym_on) number. This gives us the days between the first symptoms being observed and the patient's admission to the hospital. Correlation between the various features in the dataset gives us crucial details about the specific features and the amount of influence they exert on the value of the goal. The Pearson heat map Correlation between the elements of the dataset that is clear that there is a positive correlation between the age and the condition of the patient. It also indicates whether or not the patient is native to Wuhan and the gap (in days) the time they first experienced the symptoms and then visited hospitals, as well as the time of death. However, the location of the patient shows positive correlation with the rate of recovery. This suggests that patients from abroad who traveled to China were more likely to recover at a higher rate. There is also a significant relationship between the symptom1 as well as symptoms2, as well as between symptoms 2 and 3.

CODE:

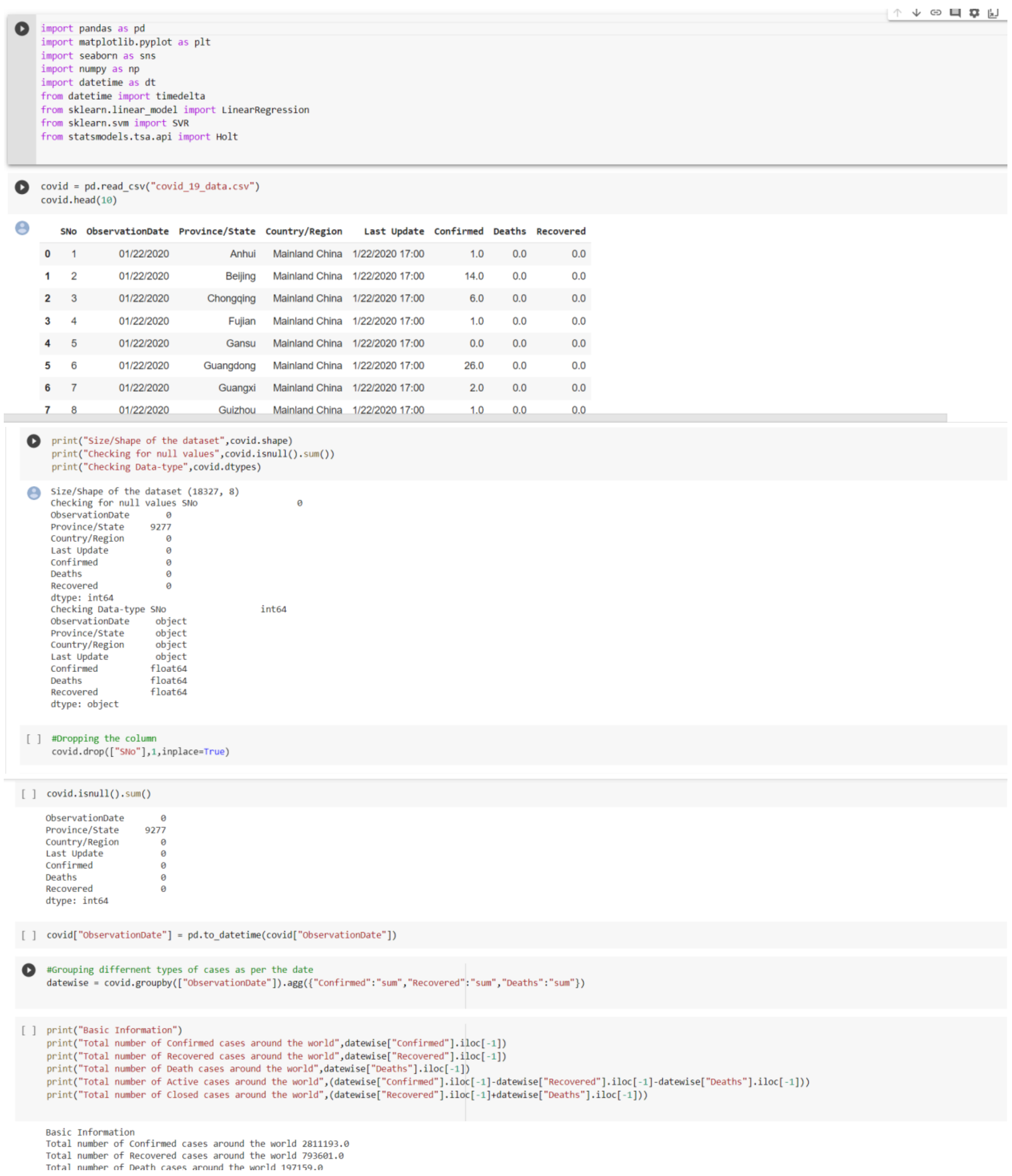
```
import pandas as pd

import matplotlib . pyplot as plt

import seaborn as sns

import numpy as np

import datetime as dt

from datetime import timedelta

from sklearn.linear_model import LinearRegression

from sklearn.svm import SVR

from statsmodels.tsa.api import Holt

covid = pd.read_csv("covid_19_data.csv")

covid.head(10)

print("Size/Shape of the dataset",covid.shape)

print("Checking for null values",covid.isnull().sum())

print("Checking Data-type",covid.dtypes)

#Dropping the column

covid.drop(["SNo"],1,inplace=True)

covid.isnull().sum()

covid["ObservationDate"] = pd.to_datetime(covid["ObservationDate"])

#Grouping differnent types of cases as per the date

datewise = covid.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum
","Deaths":"sum"})

print("Basic Information")

print("Total number of Confirmed cases around the world",datewise["Confirmed"].iloc[-1])

print("Total number of Recovered cases around the world",datewise["Recovered"].iloc[-1])

print("Total number of Death cases around the world",datewise["Deaths"].iloc[-1])
print("Total number of Active cases around the world",(datewise["Confirmed"].iloc[-1]-
datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))

print("Total number of Closed cases around the world",(datewise["Recovered"].iloc[-
1]+datewise["Deaths"].iloc[-1]))


plt.figure(figsize=(15,5))

sns.barplot(x=datewise.index.date,y=datewise["Confirmed"]-datewise["Recovered"]-
datewise["Deaths"])

plt.title("Distributions plot for Active Cases")

plt.xticks(rotation=90)

plt.figure(figsize=(15,5))

sns.barplot(x=datewise.index.date,y=datewise["Recovered"]+datewise["Deaths"])

plt.title("Distribution plot for Closed Cases")

plt.xticks(rotation=90)

datewise["WeekofYear"] = datewise.index.weekofyear

week_num = []

weekwise_confirmed = []

weekwise_recovered = []

weekwise_deaths = []

w = 1

for i in list(datewise["WeekofYear"].unique()):

   weekwise_confirmed.append(datewise[datewise["WeekofYear"]==i]["Confirmed"].iloc[-
1])

   weekwise_recovered.append(datewise[datewise["WeekofYear"]==i]["Recovered"].iloc[-
1])

   weekwise_deaths.append(datewise[datewise["WeekofYear"]==i]["Deaths"].iloc[-1])

   week_num.append(w)

   w=w+1
```

```
plt.figure(figsize=(8,5))

plt.plot(week_num,weekwise_confirmed,linewidth=3)

plt.plot(week_num,weekwise_recovered,linewidth =3)

plt.plot(week_num,weekwise_deaths,linewidth = 3)

plt.xlabel("WeekNumber")

plt.ylabel("Number of cases")

plt.title("Weekly Progress of different types of cases")

fig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))

sns.barplot(x= week_num,y=pd.Series(weekwise_confirmed).diff().fillna(0),ax=ax1)

sns.barplot(x= week_num,y=pd.Series(weekwise_deaths).diff().fillna(0),ax=ax2)

ax1.set_xlabel("Week Number")

ax2.set_xlabel("Week Number")

ax1.set_ylabel("Numberof Confirmed cases")

ax2.set_ylabel("Numberof Death cases")

ax1.set_title("Weekly increase in number of Confirmed cases")

ax2.set_title("Weekly increase in number of Death Cases")

plt.show()

print("Average increase in number of Confirmed cases everyday:",np.round(datewise["Confir
med"].diff().fillna(0).mean()))

print("Average increase in number of Recovered cases everyday:",np.round(datewise["Recov
ered"].diff().fillna(0).mean()))

print("Average increase in number of Death cases everyday:",np.round(datewise["Deaths"].di
ff().fillna(0).mean()))


plt.figure(figsize=(15,6))

plt.plot(datewise["Confirmed"].diff().fillna(0),label="Daily increase in confirmed cases",line
width=3)
```

```
plt.plot(datewise["Recovered"].diff().fillna(0),label="Daily increase in recovered cases",line
width=3)

plt.plot(datewise["Deaths"].diff().fillna(0),label="Daily increase in death cases",linewidth=3)

plt.xlabel("Timestamp")

plt.ylabel("Daily increase")

plt.title("Daily increase")

plt.legend()

plt.xticks(rotation=90)

plt.show()

#Country wise analysis

#Calculating Country wise Mortality rate

countrywise= covid[covid["ObservationDate"]==covid["ObservationDate"].max()].groupby([
"Country/Region"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"}).sort_val
ues(["Confirmed"],ascending=False)

countrywise["Mortality"]=(countrywise["Deaths"]/countrywise["Recovered"])*100

countrywise["Recovered"]=(countrywise["Recovered"]/countrywise["Confirmed"])*100

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))

top_15confirmed = countrywise.sort_values(["Confirmed"],ascending=False).head(15)

top_15deaths = countrywise.sort_values(["Deaths"],ascending=False).head(15)

sns.barplot(x=top_15confirmed["Confirmed"],y=top_15confirmed.index,ax=ax1)

ax1.set_title("Top 15 countries as per number of confirmed cases")

sns.barplot(x=top_15deaths["Deaths"],y=top_15deaths.index,ax=ax2)

ax1.set_title("Top 15 countries as per number of death cases")

#Data Anlaysis for India

india_data = covid[covid["Country/Region"]=="India"]

datewise_india = india_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recov
ered":"sum","Deaths":"sum"})

print(datewise_india.iloc[-1])
print("Total Active Cases",datewise_india["Confirmed"].iloc[-1]-
datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])

print("Total Closed Cases",datewise_india["Recovered"].iloc[-
1]+datewise_india["Deaths"].iloc[-1])

#Data Anlaysis for US

us_data = covid[covid["Country/Region"]=="US"]

datewise_us = us_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":
"sum","Deaths":"sum"})

print(datewise_us.iloc[-1])

print("Total Active Cases",datewise_us["Confirmed"].iloc[-1]-
datewise_us["Recovered"].iloc[-1]-datewise_us["Deaths"].iloc[-1])

print("Total Closed Cases",datewise_us["Recovered"].iloc[-1]+datewise_us["Deaths"].iloc[-
1])

datewise_india["WeekofYear"] = datewise_india.index.weekofyear

week_num_india = []

india_weekwise_confirmed = []

india_weekwise_recovered = []

india_weekwise_deaths = []

w = 1

for i in list(datewise_india["WeekofYear"].unique()):

   india_weekwise_confirmed.append(datewise_india[datewise_india["WeekofYear"]==i]["C
onfirmed"].iloc[-1])

   india_weekwise_recovered.append(datewise_india[datewise_india["WeekofYear"]==i]["R
ecovered"].iloc[-1])

   india_weekwise_deaths.append(datewise_india[datewise_india["WeekofYear"]==i]["Deat
hs"].iloc[-1])

   week_num_india.append(w)

   w=w+1

plt.figure(figsize=(8,5))
plt.plot(week_num_india,india_weekwise_confirmed,linewidth=3)

plt.plot(week_num_india,india_weekwise_recovered,linewidth =3)

plt.plot(week_num_india,india_weekwise_deaths,linewidth = 3)

plt.xlabel("WeekNumber")

plt.ylabel("Number of cases")

plt.title("Weekly Progress of different types of cases")

max_ind = datewise_india["Confirmed"].max()

china_data = covid[covid["Country/Region"]=="Mainland China"]

Italy_data = covid[covid["Country/Region"]=="Italy"]

US_data = covid[covid["Country/Region"]=="US"]

spain_data = covid[covid["Country/Region"]=="Spain"]

datewise_china = china_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Reco
vered":"sum","Deaths":"sum"})

datewise_Italy = Italy_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recove
red":"sum","Deaths":"sum"})

datewise_US=US_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered"
:"sum","Deaths":"sum"})

datewise_Spain=spain_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recov
ered":"sum","Deaths":"sum"})

print("It took",datewise_india[datewise_india["Confirmed"]>0].shape[0],"days in India to rea
ch",max_ind,"Confirmed Cases")

print("It took",datewise_Italy[(datewise_Italy["Confirmed"]>0)&(datewise_Italy["Confirmed
"]<=max_ind)].shape[0],"days in Italy to reach number of Confirmed Cases")

print("It took",datewise_US[(datewise_US["Confirmed"]>0)&(datewise_US["Confirmed"]<
=max_ind)].shape[0],"days in US to reach number of Confirmed Cases")

print("It took",datewise_Spain[(datewise_Spain["Confirmed"]>0)&(datewise_Spain["Confir
med"]<=max_ind)].shape[0],"days in Spain to reach number of Confirmed Cases")

print("It took",datewise_china[(datewise_china["Confirmed"]>0)&(datewise_china["Confirm
ed"]<=max_ind)].shape[0],"days in China to reach number of Confirmed Cases")

datewise["Days Since"]=datewise.index-datewise.index[0]
```

```
datewise["Days Since"] = datewise["Days Since"].dt.days

train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]

valid_ml = datewise.iloc[:int(datewise.shape[0]*0.95):]

model_scores=[]

lin_reg = LinearRegression(normalize=True)

svm = SVR(C=1,degree=5,kernel='poly',epsilon=0.001)

lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))

svm.fit(np.array(train_ml["Days Since"]).reshape(-
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))

prediction_valid_lin_reg = lin_reg.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))

prediction_valid_svm = svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))

new_date = []

new_prediction_lr=[]

new_prediction_svm=[]

for i in range(1,18):

  new_date.append(datewise.index[-1]+timedelta(days=i))

  new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days Since"].max()+i).reshap
e(-1,1))[0][0])

  new_prediction_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshap
e(-1,1))[0])

pd.set_option("display.float_format",lambda x: '%.f' % x)

model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_svm),col
umns = ["Dates","LR","SVR"])

model_predictions.head(5)

model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]

valid=datewise.iloc[int(datewise.shape[0]*0.85):]

holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=1.4,smoothing_slope
=0.2)
```

```
y_pred = valid.copy()
y_pred["Holt"]=holt.forecast(len(valid))
holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
  holt_new_date.append(datewise.index[-1]+timedelta(days=i))
  holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])
model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
```



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import datetime as dt
from datetime import timedelta
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.api import Holt
```

```
covid = pd.read_csv("covid_19_data.csv")
covid.head(10)
```

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
5	6	01/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26.0	0.0	0.0
6	7	01/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
7	8	01/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1.0	0.0	0.0

```
print("Size/Shape of the dataset",covid.shape)
print("Checking for null values",covid.isnull().sum())
print("Checking Data-type",covid.dtypes)
```

```
Size/Shape of the dataset (18327, 8)
Checking for null values SNo                  0
ObservationDate      0
Province/State    9277
Country/Region       0
Last Update          0
Confirmed            0
Deaths               0
Recovered            0
dtype: int64
Checking Data-type SNo                  int64
ObservationDate     object
Province/State      object
Country/Region      object
Last Update         object
Confirmed          float64
Deaths             float64
Recovered          float64
dtype: object
```

```
#Dropping the column
covid.drop(["SNo"],1,inplace=True)
```

```
covid.isnull().sum()
```

```
ObservationDate      0
Province/State    9277
Country/Region       0
Last Update          0
Confirmed            0
Deaths               0
Recovered            0
dtype: int64
```

```
covid["ObservationDate"] = pd.to_datetime(covid["ObservationDate"])
```

```
#Grouping differnent types of cases as per the date
datewise = covid.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
```

```
print("Basic Information")
print("Total number of Confirmed cases around the world",datewise["Confirmed"].iloc[-1])
print("Total number of Recovered cases around the world",datewise["Recovered"].iloc[-1])
print("Total number of Death cases around the world",datewise["Deaths"].iloc[-1])
print("Total number of Active cases around the world",(datewise["Confirmed"].iloc[-1]-datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))
print("Total number of Closed cases around the world",(datewise["Recovered"].iloc[-1]+datewise["Deaths"].iloc[-1]))
```

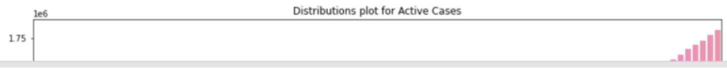
```
Basic Information
Total number of Confirmed cases around the world 2811193.0
Total number of Recovered cases around the world 793601.0
Total number of Death cases around the world 197159.0
```

```
[ ] print("Basic Information")
    print("Total number of Confirmed cases around the world",datewise["Confirmed"].iloc[-1])
    print("Total number of Recovered cases around the world",datewise["Recovered"].iloc[-1])
    print("Total number of Death cases around the world",datewise["Deaths"].iloc[-1])
    print("Total number of Active cases around the world",(datewise["Confirmed"].iloc[-1]-datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))
    print("Total number of Closed cases around the world",(datewise["Recovered"].iloc[-1]+datewise["Deaths"].iloc[-1]))
```

```
Basic Information
Total number of Confirmed cases around the world 2811193.0
Total number of Recovered cases around the world 793601.0
Total number of Death cases around the world 197159.0
Total number of Active cases around the world 1820433.0
Total number of Closed cases around the world 990760.0
```

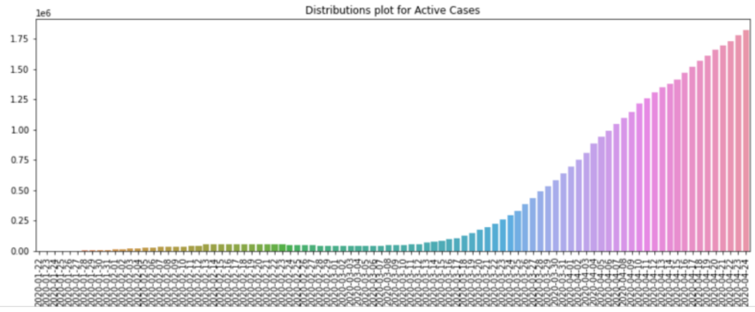
```
plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Confirmed"]-datewise["Recovered"]-datewise["Deaths"])
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 <a list of 94 Text major ticklabel objects>)
```



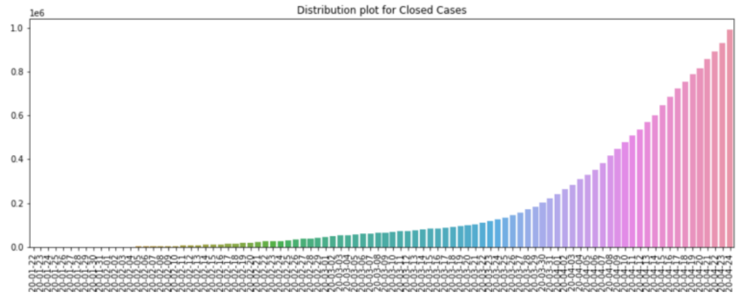
```
plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Confirmed"]-datewise["Recovered"]-datewise["Deaths"])
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 <a list of 94 Text major ticklabel objects>)
```



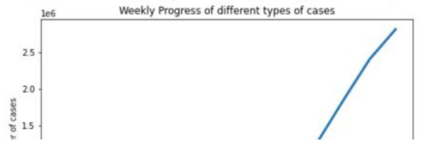
```
plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["Recovered"]+datewise["Deaths"])
plt.title("Distribution plot for Closed Cases")
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 <a list of 94 Text major ticklabel objects>)
```

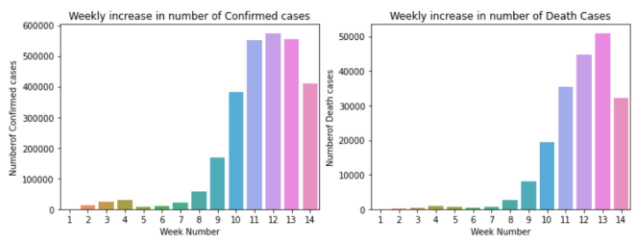


```
datewise["WeekofYear"] = datewise.index.weekofyear
week_num = []
weekwise_confirmed = []
weekwise_recovered = []
weekwise_deaths = []
w = 1
for i in list(datewise["WeekofYear"].unique()):
    weekwise_confirmed.append(datewise[datewise["WeekofYear"]==i]["Confirmed"].iloc[-1])
    weekwise_recovered.append(datewise[datewise["WeekofYear"]==i]["Recovered"].iloc[-1])
    weekwise_deaths.append(datewise[datewise["WeekofYear"]==i]["Deaths"].iloc[-1])
    week_num.append(w)
    w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num,weekwise_confirmed,linewidth=3)
plt.plot(week_num,weekwise_recovered,linewidth =3)
plt.plot(week_num,weekwise_deaths,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
```

```
Text(0.5, 1.0, 'Weekly Progress of different types of cases')
```

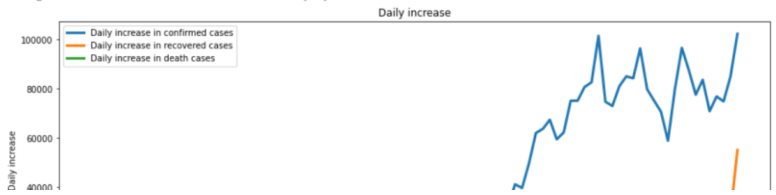


```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))
sns.barplot(x= week_num,y=pd.Series(weekwise_confirmed).diff().fillna(0),ax=ax1)
sns.barplot(x= week_num,y=pd.Series(weekwise_deaths).diff().fillna(0),ax=ax2)
ax1.set_xlabel("Week Number")
ax2.set_xlabel("Week Number")
ax1.set_ylabel("Numberof Confirmed cases")
ax2.set_ylabel("Numberof Death cases")
ax1.set_title("Weekly increase in number of Confirmed cases")
ax2.set_title("Weekly increase in number of Death Cases")
plt.show()
```



```
print("Average increase in number of Confirmed cases everyday:",np.round(datewise["Confirmed"].diff().fillna(0).mean()))
print("Average increase in number of Recovered cases everyday:",np.round(datewise["Recovered"].diff().fillna(0).mean()))
print("Average increase in number of Death cases everyday:",np.round(datewise["Deaths"].diff().fillna(0).mean()))

plt.figure(figsize=(15,6))
plt.plot(datewise["Confirmed"].diff().fillna(0),label="Daily increase in confirmed cases",linewidth=3)
plt.plot(datewise["Recovered"].diff().fillna(0),label="Daily increase in recovered cases",linewidth=3)
plt.plot(datewise["Deaths"].diff().fillna(0),label="Daily increase in death cases",linewidth=3)
plt.xlabel("Timestamp")
plt.ylabel("Daily increase")
plt.title("Daily increase")
plt.legend()
plt.xticks(rotation=90)
plt.show()
```

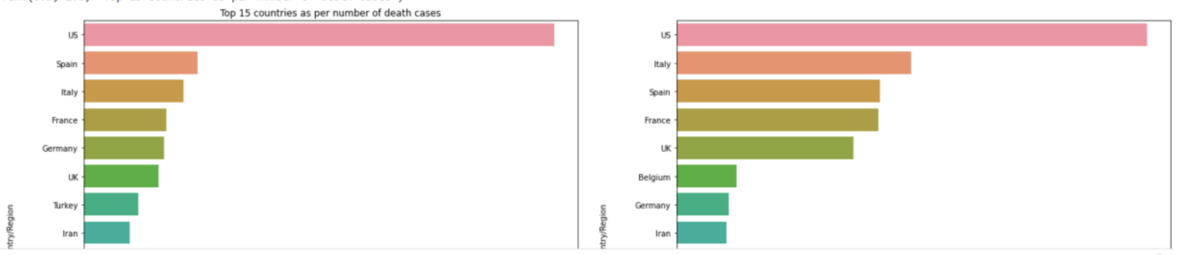
```
Average increase in number of Confirmed cases everyday: 29900.0
Average increase in number of Recovered cases everyday: 8442.0
Average increase in number of Death cases everyday: 2097.0
```



```
[ ] #Country wise analysis
    #Calculating Country wise Mortality rate
    countrywise= covid[covid["ObservationDate"]==covid["ObservationDate"].max()].groupby(["Country/Region"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"}).sort_values(["Confir
    countrywise["Mortality"]=(countrywise["Deaths"]/countrywise["Recovered"])*100
    countrywise["Recovered"]=(countrywise["Recovered"]/countrywise["Confirmed"])*100
```

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))
top_15confirmed = countrywise.sort_values(["Confirmed"],ascending=False).head(15)
top_15deaths = countrywise.sort_values(["Deaths"],ascending=False).head(15)
sns.barplot(x=top_15confirmed["Confirmed"],y=top_15confirmed.index,ax=ax1)
ax1.set_title("Top 15 countries as per number of confirmed cases")
sns.barplot(x=top_15deaths["Deaths"],y=top_15deaths.index,ax=ax2)
ax1.set_title("Top 15 countries as per number of death cases")
```

```
Text(0.5, 1.0, 'Top 15 countries as per number of death cases')
```



```
[ ] #Data Anlaysis for India
    india_data = covid[covid["Country/Region"]=="India"]
    datewise_india = india_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
    print(datewise_india.iloc[-1])
    print("Total Active Cases",datewise_india["Confirmed"].iloc[-1]-datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])
    print("Total Closed Cases",datewise_india["Recovered"].iloc[-1]+datewise_india["Deaths"].iloc[-1])
```

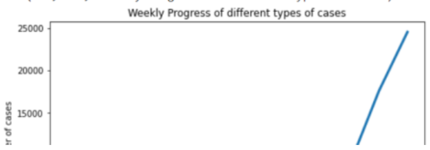
```
Confirmed    24530.0
Recovered     5498.0
Deaths         780.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 18252.0
Total Closed Cases 6278.0
```

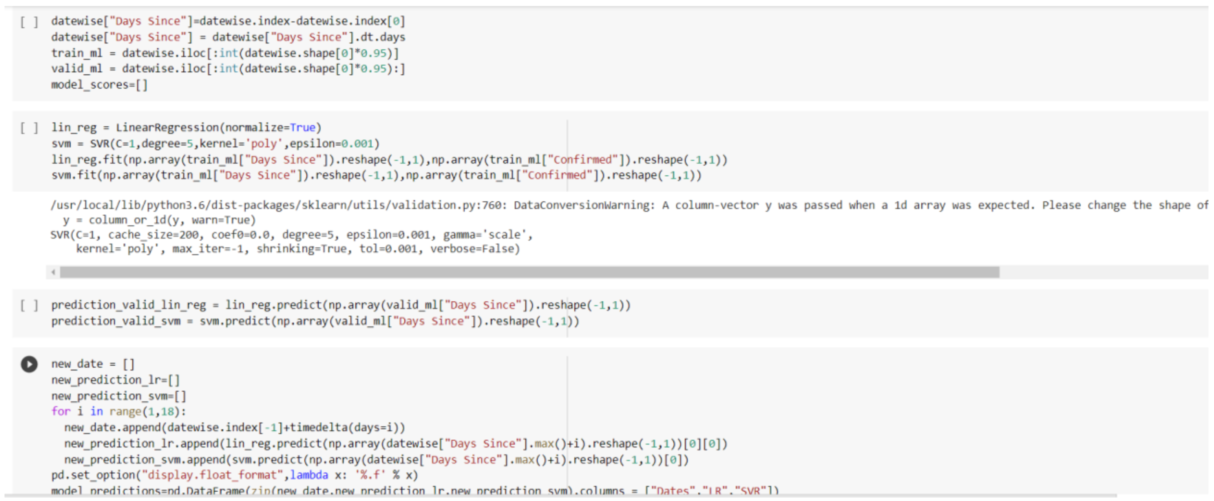
```
#Data Anlaysis for US
us_data = covid[covid["Country/Region"]=="US"]
datewise_us = us_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
print(datewise_us.iloc[-1])
print("Total Active Cases",datewise_us["Confirmed"].iloc[-1]-datewise_us["Recovered"].iloc[-1]-datewise_us["Deaths"].iloc[-1])
print("Total Closed Cases",datewise_us["Recovered"].iloc[-1]+datewise_us["Deaths"].iloc[-1])
```

```
Confirmed    905333.0
Recovered     99079.0
Deaths        51949.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 754305.0
Total Closed Cases 151028.0
```

```
datewise_india["WeekofYear"] = datewise_india.index.weekofyear
week_num_india = []
india_weekwise_confirmed = []
india_weekwise_recovered = []
india_weekwise_deaths = []
w = 1
for i in list(datewise_india["WeekofYear"].unique()):
    india_weekwise_confirmed.append(datewise_india[datewise_india["WeekofYear"]==i]["Confirmed"].iloc[-1])
    india_weekwise_recovered.append(datewise_india[datewise_india["WeekofYear"]==i]["Recovered"].iloc[-1])
    india_weekwise_deaths.append(datewise_india[datewise_india["WeekofYear"]==i]["Deaths"].iloc[-1])
    week_num_india.append(w)
    w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num_india,india_weekwise_confirmed,linewidth=3)
plt.plot(week_num_india,india_weekwise_recovered,linewidth =3)
plt.plot(week_num_india,india_weekwise_deaths,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
```

```
Text(0.5, 1.0, 'Weekly Progress of different types of cases')
```





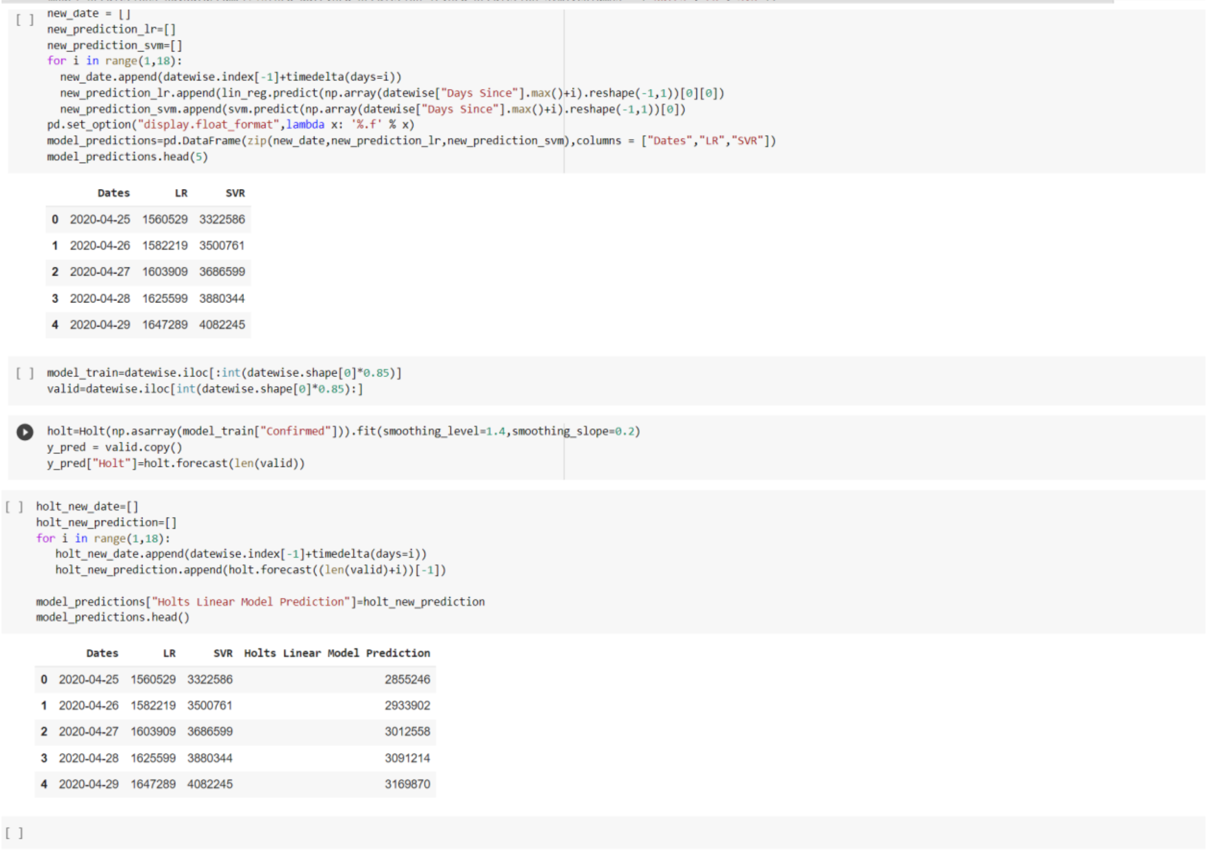
```
[ ] datewise["Days Since"]=datewise.index-datewise.index[0]
    datewise["Days Since"] = datewise["Days Since"].dt.days
    train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
    valid_ml = datewise.iloc[:int(datewise.shape[0]*0.95):]
    model_scores=[]
```

```
[ ] lin_reg = LinearRegression(normalize=True)
    svm = SVR(C=1,degree=5,kernel='poly',epsilon=0.001)
    lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
    svm.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of
  y = column_or_1d(y, warn=True)
SVR(C=1, cache_size=200, coef0=0.0, degree=5, epsilon=0.001, gamma='scale',
    kernel='poly', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
[ ] prediction_valid_lin_reg = lin_reg.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
    prediction_valid_svm = svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
```

```
    new_date = []
    new_prediction_lr=[]
    new_prediction_svm=[]
    for i in range(1,18):
      new_date.append(datewise.index[-1]+timedelta(days=i))
      new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0][0])
      new_prediction_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0])
    pd.set_option("display.float_format",lambda x: '%.f' % x)
    model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_svm),columns = ["Dates","LR","SVR"])
```



```
[ ] new_date = []
    new_prediction_lr=[]
    new_prediction_svm=[]
    for i in range(1,18):
      new_date.append(datewise.index[-1]+timedelta(days=i))
      new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0][0])
      new_prediction_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0])
    pd.set_option("display.float_format",lambda x: '%.f' % x)
    model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_svm),columns = ["Dates","LR","SVR"])
    model_predictions.head(5)
```

	Dates	LR	SVR
0	2020-04-25	1560529	3322586
1	2020-04-26	1582219	3500761
2	2020-04-27	1603909	3686599
3	2020-04-28	1625599	3880344
4	2020-04-29	1647289	4082245

```
[ ] model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]
    valid=datewise.iloc[int(datewise.shape[0]*0.85):]
```

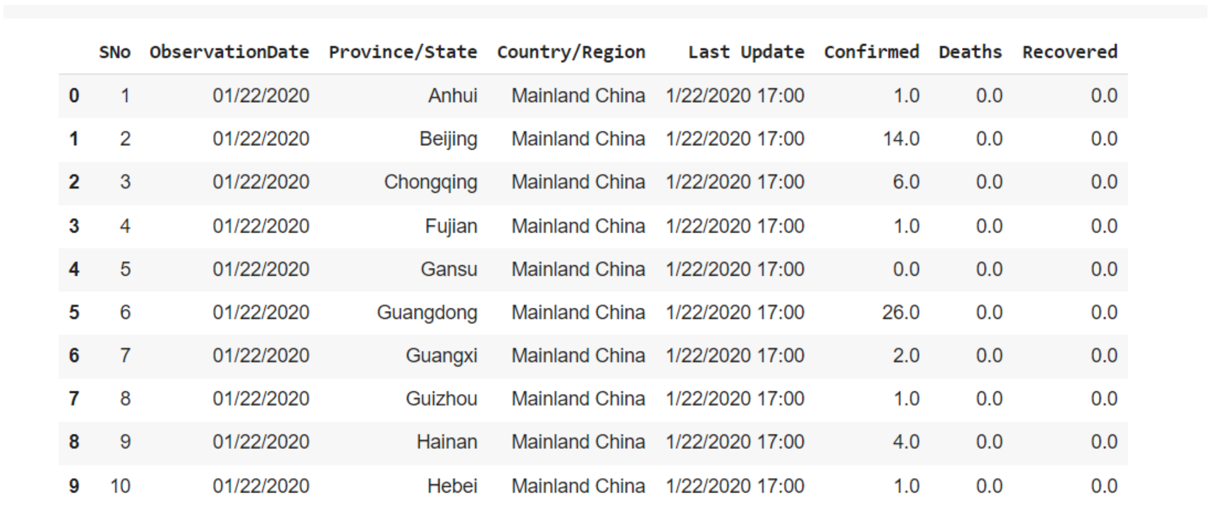
```
    holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=1.4,smoothing_slope=0.2)
    y_pred = valid.copy()
    y_pred["Holt"]=holt.forecast(len(valid))
```

```
[ ] holt_new_date=[]
    holt_new_prediction=[]
    for i in range(1,18):
        holt_new_date.append(datewise.index[-1]+timedelta(days=i))
        holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

    model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
    model_predictions.head()
```

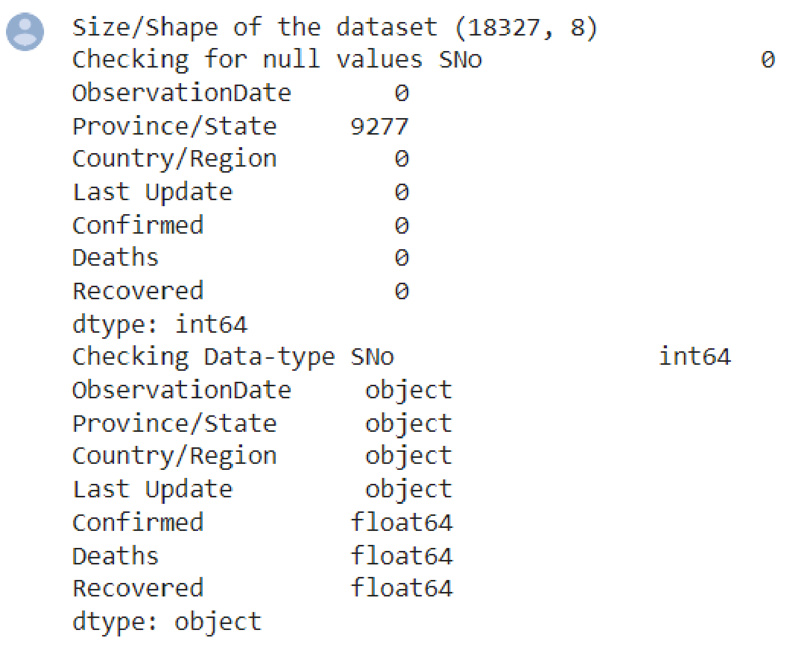
	Dates	LR	SVR	Holts Linear Model Prediction
0	2020-04-25	1560529	3322586	2855246
1	2020-04-26	1582219	3500761	2933902
2	2020-04-27	1603909	3686599	3012558
3	2020-04-28	1625599	3880344	3091214
4	2020-04-29	1647289	4082245	3169870

```
[ ]
```

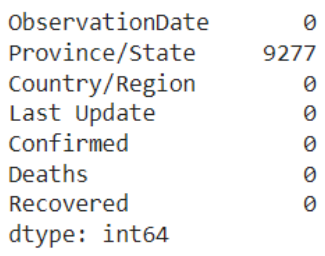
OUTPUTS



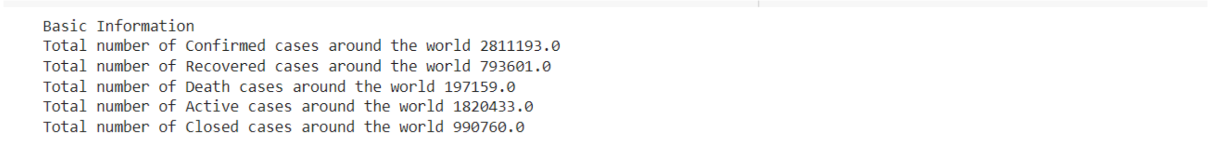
	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
5	6	01/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26.0	0.0	0.0
6	7	01/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
7	8	01/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
8	9	01/22/2020	Hainan	Mainland China	1/22/2020 17:00	4.0	0.0	0.0
9	10	01/22/2020	Hebei	Mainland China	1/22/2020 17:00	1.0	0.0	0.0



```
Size/Shape of the dataset (18327, 8)
Checking for null values SNo                  0
ObservationDate       0
Province/State     9277
Country/Region        0
Last Update           0
Confirmed             0
Deaths                0
Recovered             0
dtype: int64
Checking Data-type SNo                  int64
ObservationDate     object
Province/State      object
Country/Region      object
Last Update         object
Confirmed          float64
Deaths             float64
Recovered          float64
dtype: object
```

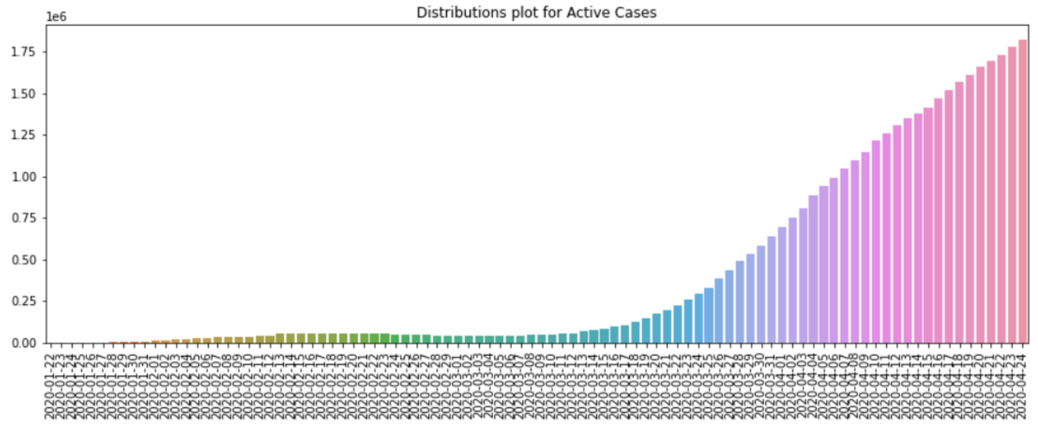


```
ObservationDate       0
Province/State     9277
Country/Region        0
Last Update           0
Confirmed             0
Deaths                0
Recovered             0
dtype: int64
```

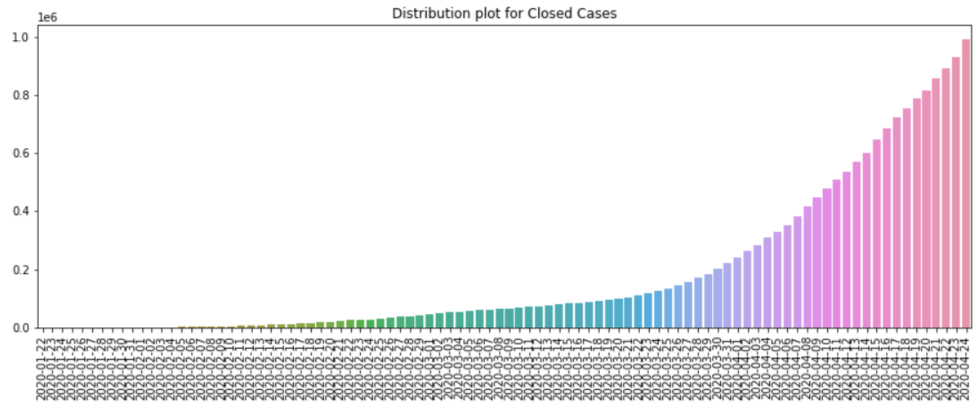


```
Basic Information
Total number of Confirmed cases around the world 2811193.0
Total number of Recovered cases around the world 793601.0
Total number of Death cases around the world 197159.0
Total number of Active cases around the world 1820433.0
Total number of Closed cases around the world 990760.0
```

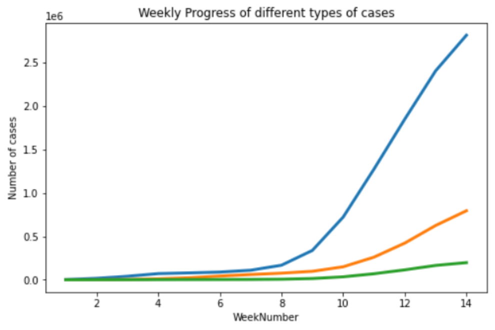
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 <a list of 94 Text major ticklabel objects>)
```

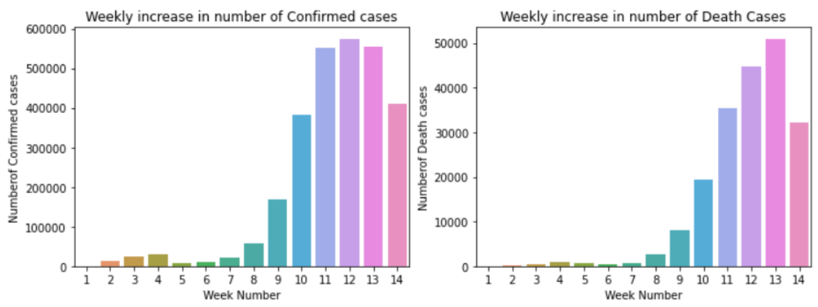


```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
        85, 86, 87, 88, 89, 90, 91, 92, 93]),
 <a list of 94 Text major ticklabel objects>)
```

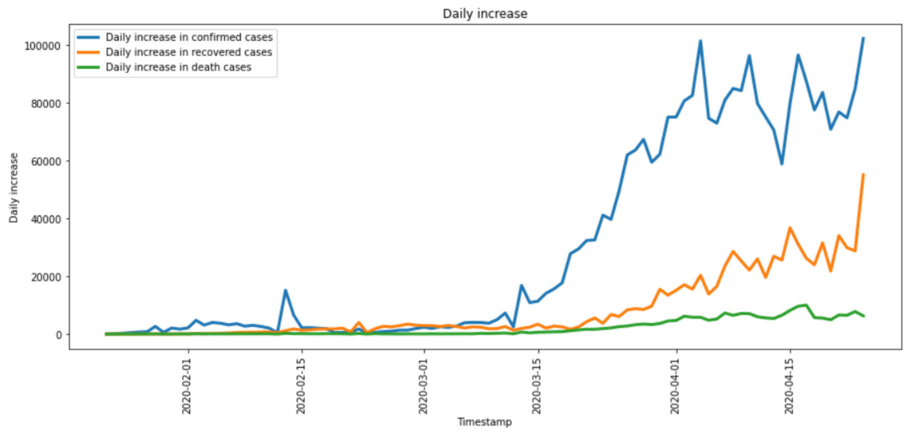


```
Text(0.5, 1.0, 'Weekly Progress of different types of cases')
```

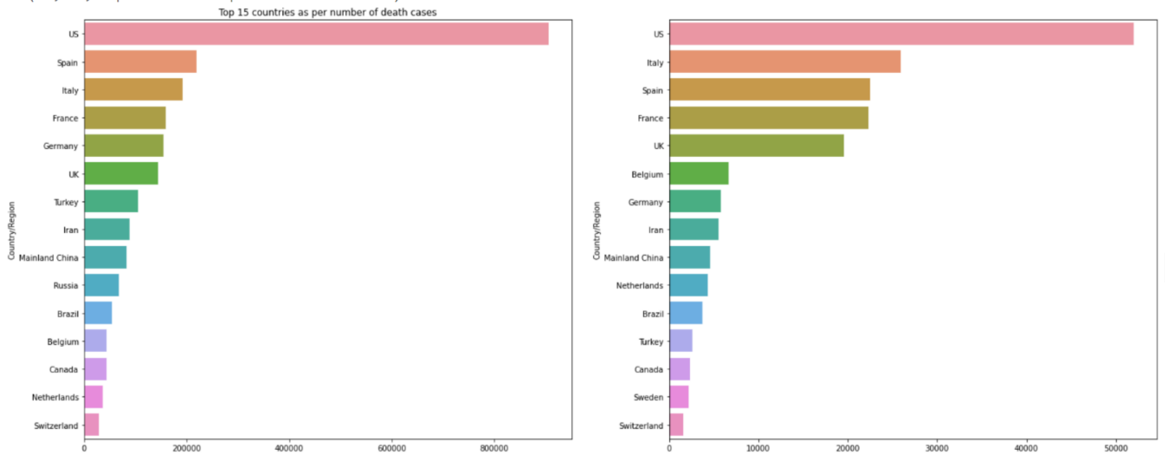




```
Average increase in number of Confirmed cases everyday: 29900.0
Average increase in number of Recovered cases everyday: 8442.0
Average increase in number of Death cases everyday: 2097.0
```



```
Text(0.5, 1.0, 'Top 15 countries as per number of death cases')
```

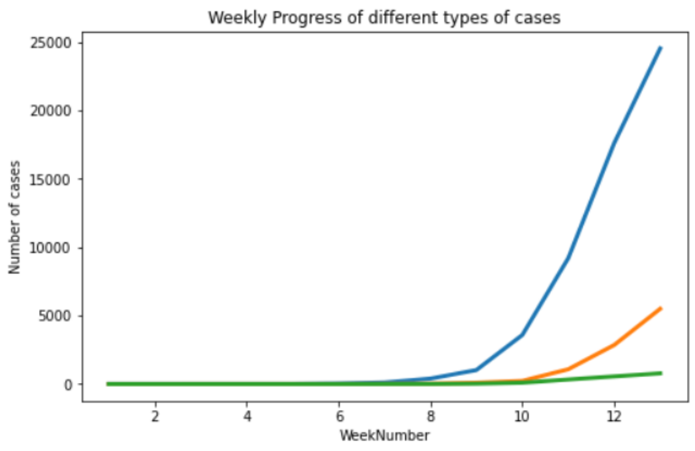


```
Confirmed    24530.0
Recovered     5498.0
Deaths         780.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 18252.0
Total Closed Cases 6278.0
```

```
[ ]  #Data Anlaysis for US
     us_data = covid[covid["Country/Region"]=="US"]
     datewise_us = us_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"})
     print(datewise_us.iloc[-1])
     print("Total Active Cases",datewise_us["Confirmed"].iloc[-1]-datewise_us["Recovered"].iloc[-1]-datewise_us["Deaths"].iloc[-1])
     print("Total Closed Cases",datewise_us["Recovered"].iloc[-1]+datewise_us["Deaths"].iloc[-1])
```

```
Confirmed    905333.0
Recovered     99079.0
Deaths        51949.0
Name: 2020-04-24 00:00:00, dtype: float64
Total Active Cases 754305.0
Total Closed Cases 151028.0
```

```
Text(0.5, 1.0, 'Weekly Progress of different types of cases')
```



```
It took 86 days in India to reach 24530.0 Confirmed Cases
It took 44 days in Italy to reach number of Confirmed Cases
It took 59 days in US to reach number of Confirmed Cases
It took 49 days in Spain to reach number of Confirmed Cases
It took 14 days in China to reach number of Confirmed Cases
```

[ ]

	Dates	LR	SVR
0	2020-04-25	1560529	3322586
1	2020-04-26	1582219	3500761
2	2020-04-27	1603909	3686599
3	2020-04-28	1625599	3880344
4	2020-04-29	1647289	4082245

	Dates	LR	SVR	Holts Linear Model Prediction
0	2020-04-25	1560529	3322586	2855246
1	2020-04-26	1582219	3500761	2933902
2	2020-04-27	1603909	3686599	3012558
3	2020-04-28	1625599	3880344	3091214
4	2020-04-29	1647289	4082245	3169870

CONCLUSION AND FUTURE ENHANCEMENT

The COVID-19 pandemic has devastated the globe, and caused an epidemic of health across the globe. There have been many initiatives to combat this epidemic. This study aimed to examine the impact of important indicators such as chronic diseases, and preliminary data on clinical and demographic data to assess the mortality and life expectancy of COVID-19 patients using automated machine-learning algorithms. Because of the reduced mortality rate in COVID-19 patients information is not balanced. SMOTE technology was used to eliminate the inequities within the data. The results showed that random forests was more effective than other models through the cross-validation of 10 times.

Grid search was utilized to improve the parameters. The research was able to attain a accuracy in the range of 0.952 with an average of 0.99. While there's a substantial outcome from this model, it is still a need to improve. The model has to be confirmed by utilizing a variety of sources of data. In coming years, we'll include and examine the impact of other clinical characteristics as well as laboratory results which were identified as significant in earlier studies.

REFERENCES

1. Geraghty, E. Why Health is so Spatial. 2016. Available online: https://www.youtube.com/watch?v=3p7OFI Cg9Ak (accessed on 30 January 2020).

2. Krieger, N. Place, space, and health: GIS and epidemiology. Epidemiology 2003, 14, 384. [CrossRef] [PubMed]

3. GISGeography. The Remarkable History of GIS. 2015. Available online: https://gisgeography.com/history-of-gis/ (accessed on 3 February 2020).

4. ESRI. GIS for Public Health Today and Tomorrow. 2020. Available online: https://www.esri.com/news/arcu ser/0499/umbrella.html (accessed on 3 February 2020).

5. Wilkinson, P.; Grundy, C.; Landon, M.; Stevenson, S. GIS in public health. In GIS and Health; Gatrell, A., Loytonen, M., Eds.; Tylor and Francis: Abingdon, UK, 1998; pp. 179–189.

6. Kirby, R.; Delmelle, E.; Eberth, J. Advances in spatial epidemiology and geographic information systems. Ann. Epidemiol. 2016, 27, 1–9. [CrossRef] [PubMed]

7. Fletcher-Lartey, S.M.; Caprarelli, G. Application of GIS technology in public health: Successes and challenges. Parasitology 2016, 143, 1–15. [CrossRef] [PubMed]

8. Lawson, A.; Browne, W.; Vidal-Rodeiro, C. Disease Mapping with Winbugs and Mlwin; John Wiley & Sons: Chichester, UK, 2003.

9. Maheswaran, R.; Craglia, M. GIS in Public Health Practice; CRC Press: New York, NY, USA, 2004; pp. 32–34.

10. Koch, T. Cartographies of Disease: Maps, Mapping, and Medicine; ESRI Press: Redlands, CA, USA, 2005.

11. American Hospital Association. Mapping Medicare Disparities. 2018. Available online: https://www.aha.or g/system/files/2018-12/mapping-medicare-disparities-issue-brief.pdf (accessed on 7 April 2020).

12. Gould, P.; Wallace, R. Spatial structures and scientific paradoxes in the AIDS pandemic. Geogr. Ann. 1994, 76, 105–116. [CrossRef]

13. Braga, M.; Cislaghi, C.; Luppi, G.; Tasco, C. A Multipurpose, interactive mortality atlas of Italy. In GIS and Health; Gatrell, A.C., Löytönen, M., Eds.; Taylor and Francis: London, UK, 1998; pp. 125–138.

14. Photis, Y.N. Disease and health care geographies: Mapping trends and patterns in a GIS. Health Sci. J. 2016, 10, 1–8.

15. Sones, M. Reveal: Mapping and Tracking the Spread of Deadly Diseases. 2019. Available online: https://ww w.esri.com/about/newsroom/blog/reveal-mapping-and-tracking-the-spread-of-deadly-diseases/ (accessed on 18 December 2019).

16. Santos, A.S.; Medeiros, N.G.; dos Santos, G.R.; Filho, J.L. Bull. Geod. Sci. 2017, 23, 405–413. [CrossRef]

17. Lawson, A.; Kleinman, K. Spatial and Syndromic Surveillance for Public Health; John Wiley & Sons: New York, NY, USA, 2005; p. 57.

18. Lawson, A.; Cressie, N. Spatial statistical methods for environmental epidemiology. Handb. Stat. 2000, 18, 357–396.

project-1

ORIGINALITY REPORT

18%
SIMILARITY INDEX

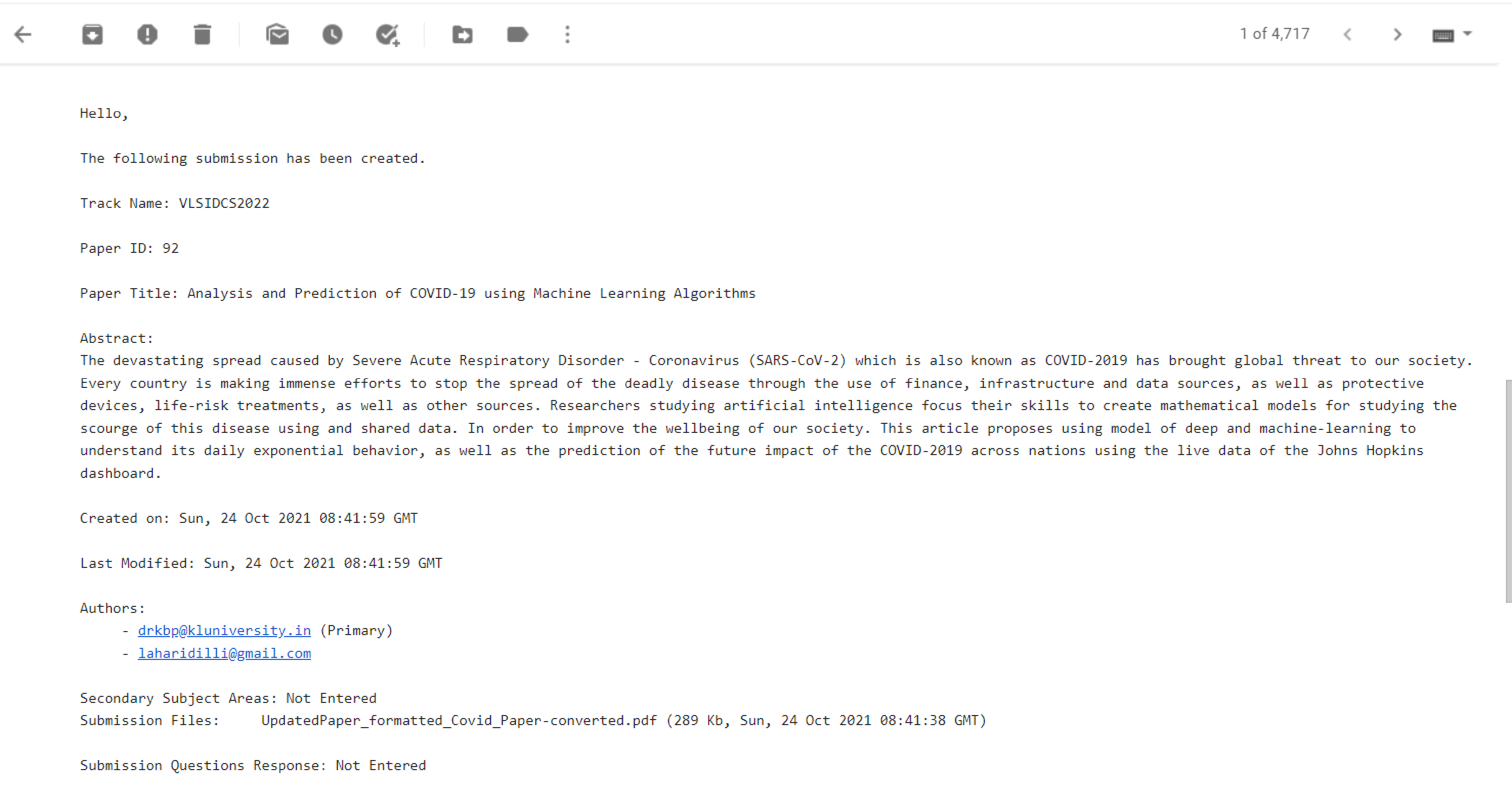
PRIMARY SOURCES

1	Celestine Iwendi, Ali Kashif Bashir, Atharva Peshkar, R. Sujatha et al. "COVID-19 Patient Health Prediction Using Boosted Random Forest Algorithm", Frontiers in Public Health, 2020 Crossref	196 words — 6%
2	V. Akila, P.K. Abhilash, P Bala Venakata Satya Phanindra, J Pavan Kumar, A. Kavitha. "Brain Tumors Classification System Using Convolutional Recurrent Neural Network", E3S Web of Conferences, 2021 Crossref	77 words — 2%
3	stxnext.com Internet	58 words — 2%
4	www.stxnext.com Internet	58 words — 2%
5	archive.org Internet	56 words — 2%
6	Yazeed Zoabi, Shira Deri-Rozov, Noam Shomron. "Machine learning-based prediction of COVID-19 diagnosis based on symptoms", npj Digital Medicine, 2021 Crossref	55 words — 2%
7	Deepak Painuli, Divya Mishra, Suyash Bhardwaj, Mayank Aggarwal. "Forecast and prediction of	47 words — 1%

COVID-19 using machine learning", Data Science for COVID-19
Internet

8	www.irjet.net Internet	26 words — 1%
9	Geza Halasz, Michela Sperti, Matteo Villani, Umberto Michelucci et al. "Predicting clinical outcomes in the Machine Learning era: The Piacenza score a purely data driven approach for mortality prediction in COVID-19 Pneumonia", Cold Spring Harbor Laboratory, 2021 Crossref Posted Content	24 words — 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		


1 of 4,717

Hello,

The following submission has been created.

Track Name: VLSIDCS2022

Paper ID: 92

Paper Title: Analysis and Prediction of COVID-19 using Machine Learning Algorithms

Abstract:
The devastating spread caused by Severe Acute Respiratory Disorder - Coronavirus (SARS-CoV-2) which is also known as COVID-2019 has brought global threat to our society. Every country is making immense efforts to stop the spread of the deadly disease through the use of finance, infrastructure and data sources, as well as protective devices, life-risk treatments, as well as other sources. Researchers studying artificial intelligence focus their skills to create mathematical models for studying the scourge of this disease using and shared data. In order to improve the wellbeing of our society. This article proposes using model of deep and machine-learning to understand its daily exponential behavior, as well as the prediction of the future impact of the COVID-2019 across nations using the live data of the Johns Hopkins dashboard.

Created on: Sun, 24 Oct 2021 08:41:59 GMT

Last Modified: Sun, 24 Oct 2021 08:41:59 GMT

Authors:
- drkbp@kluniversity.in (Primary)
- laharidilli@gmail.com

Secondary Subject Areas: Not Entered
Submission Files: UpdatedPaper_formatted_Covid_Paper-converted.pdf (289 Kb, Sun, 24 Oct 2021 08:41:38 GMT)

Submission Questions Response: Not Entered