Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

A
Project Report
On

Study on Covid-19 Prediction

M.tech-sem II

Prepared By

Faldu Vinit Girishbhai (21MCS0036)
Mehta Viraj Hiteshbhai (21MCS0045)

Under the Guidance of

Prof. BALAMURUGAN R

Abstract

COVID-19 outbreaks only affect the lives of people, they result in a negative impact on the economy of the country. On Jan. 30, 2020, it was declared as a health emergency for the entire globe by the World Health Organization (WHO). By Apr. 28, 2020, more than 3 million people were infected by this virus and there was no vaccine to prevent. The WHO released certain guidelines for safety, but they were only precautionary measures. The use of information technology with a focus on fields such as data Science and machine learning can help in the fight against this pandemic. It is important to have early warning methods through which one can forecast how much the disease will affect society, on the basis of which the government can take necessary actions without affecting its economy. In this chapter, we include methods for forecasting future cases based on existing data. Machine learning approaches are used and two solutions, one for predicting the chance of being infected and other for forecasting the number of positive cases, are discussed. A trial was done for different algorithms, and the algorithm that gave results with the best accuracy are covered in the chapter. The chapter discusses autoregressive integrated moving average time series for forecasting confirmed cases for various states of India. Two classifiers, random forest and extra tree classifiers, were selected; both have an accuracy of more than 90%. Of the two, the extra tree classifier has 93.62% accuracy. These results can be used to take corrective measures by different governmental bodies. The availability of techniques for forecasting infectious disease can make it easier to fight COVID-19.

1. Introduction

In this era of automation, artificial intelligence and data science have important role in the health care industry. These technologies are so well-connected that medical professionals can easily manage their roles and patient care. All health care organizations work hard to develop an automated system that can be used to accept the challenges faced in health care. Scientists are working on machine learning (ML) to develop smart solutions to diagnose and treat disease. ML is capable of detecting disease and virus infections more accurately so that patients' disease can be diagnosed at an early stage, the dangerous stages of diseases can be avoided, and there can be fewer patients. In the same manner, ML can be used to automate the task of predicting COVID-19 infection and help forecast future infection tallies of COVID-19. In this, we include methods for forecasting future cases based on existing data. ML approaches are used and the no of postive cases,deaths,active cases are shown. A trial was done for different algorithms, and the algorithm that gave the results with the best accuracy is covered in the model. The model discusses autoregressive integrated moving average (ARIMA) time series for forecasting. Two classifiers,support vector machine which has accuracy of more than 90% and lineare regression model, fort the improvement we have use the time series analysis. The availability of techniques for forecasting infectious disease can make it easier to fight against infectious disease such as COVID-19.

2. Problem statement

To Study on Covid19 Prediction using the linear regression model,support vector machine and timeseries analysis for better result.

3. Objective

- To view our results or visualize the data, we obtain help from different maps,graphs, charts. These visualization tools make it easy for the reader to understand the model, trends or outliers in our project results.

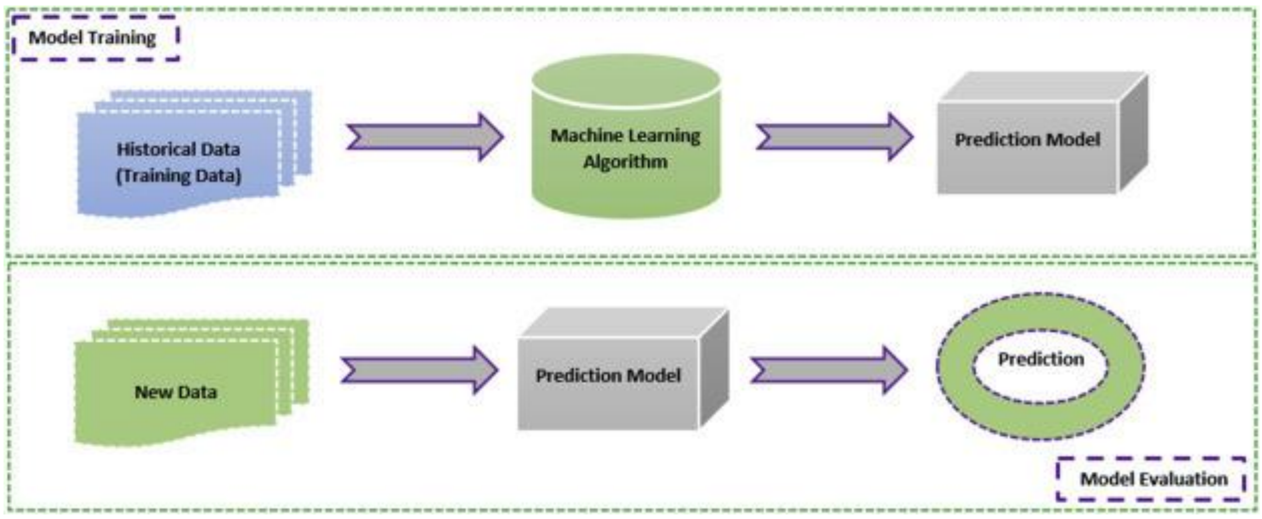
- To predict the total number of cases, retrieved cases and deaths across a given set of data based on the concept of machine learning.

4. Methodology

ML is the field of study that gives computers the ability to learn without being explicitly programmed. Thus, we can define ML as the field of computer science in which machines can be designed that can program themselves .

The process of learning is simply learning from experience or observations from previous work, such as examples, or instruction, to look for patterns in data and with the help of examples, provided the system can make better decisions. The basic aim of ML is to make computers learn automatically with no human intervention and to adjust perform actions accordingly .

Figure shows the process of ML. Past data are used to train the model, and then this trained model is used to test new data and then for prediction. The trained ML model's performance is evaluated using some portion of available past data (which is not present during training). This is usually referred as the validation process. In this process, the ML model is evaluated for its performance measure, such as accuracy. Accuracy describes the ML model's performance over unseen data in terms of the ratio of the number of correctly predicted features and total available features to be predicted.



Some of the machine learning method

- Supervised ML algorithms is a type of ML technique that can be applied according to what was previously learned to get new data using labeled data and to predict future events or labels. In this type of learning, supervisor (labels) is present to guide or correct. For this first analysis, the known training set and then the output values are predicted using the learning algorithm. The output defined by the learning system can be compared with the actual output; if errors are identified, they can be rectified and the model can be modified accordingly .

- Unsupervised ML algorithms: In this type, there is no supervisor to guide or correct. This type of learning algorithm is used when unlabeled or unclassified information is present to train the system. The system does not define the correct output, but it explores the data in such a way that it can draw inferences (rules) from datasets and can describe hidden structures from unlabeled data.

- Semisupervised ML algorithms are algorithms that are between the category of supervised and unsupervised learning. Thus, this type of learning algorithm uses both unlabeled and labeled data for training purposes, generally a small amount of labeled data and a large amount of unlabeled data. This type of method is used to improve the accuracy of learning .

- Reinforcement ML algorithms is a type of learning method that gives rewards or punishment on the basis of the work performed by the system. If we train the system to perform a certain task and it fails to do that, the system might be punished; if it performs perfectly, it will be rewarded. It typically works on 0 and 1, in which 0 indicates a punishment and 1 indicates a reward.

Why Use of machine learning in COVID-19

ML is used in various fields, including medicine to predict disease and forecast its outcome. In medicine, the right diagnosis and the right time are the keys to successful treatment. If the treatment has a high error rate, it may cause several deaths. Therefore, researchers have started using artificial intelligence applications for medical treatment. The task is complicated because the researchers have to choose the right tool: it is a matter of life or death .

For this task, ML achieved a milestone in the field of health care. ML techniques are used to interpret and analyze large datasets and predict their output. These ML tools were used to identify the symptoms of disease and classify samples into treatment groups. ML helps hospitals to maintain administrative processes and treat infectious disease ..

ML techniques were previously used to treat cancer, pneumonia, diabetes, Parkinson disease, arthritis, neuromuscular disorders, and many more diseases; they give more than 90% accurate results in prediction and forecasting .

The pandemic disease known as COVID-19 is a deadly virus that has cost the lives of many people all over the world. There is no treatment for this virus. ML techniques have been used to predict whether patients are infected by the virus based on symptoms defined by WHO and CDC .

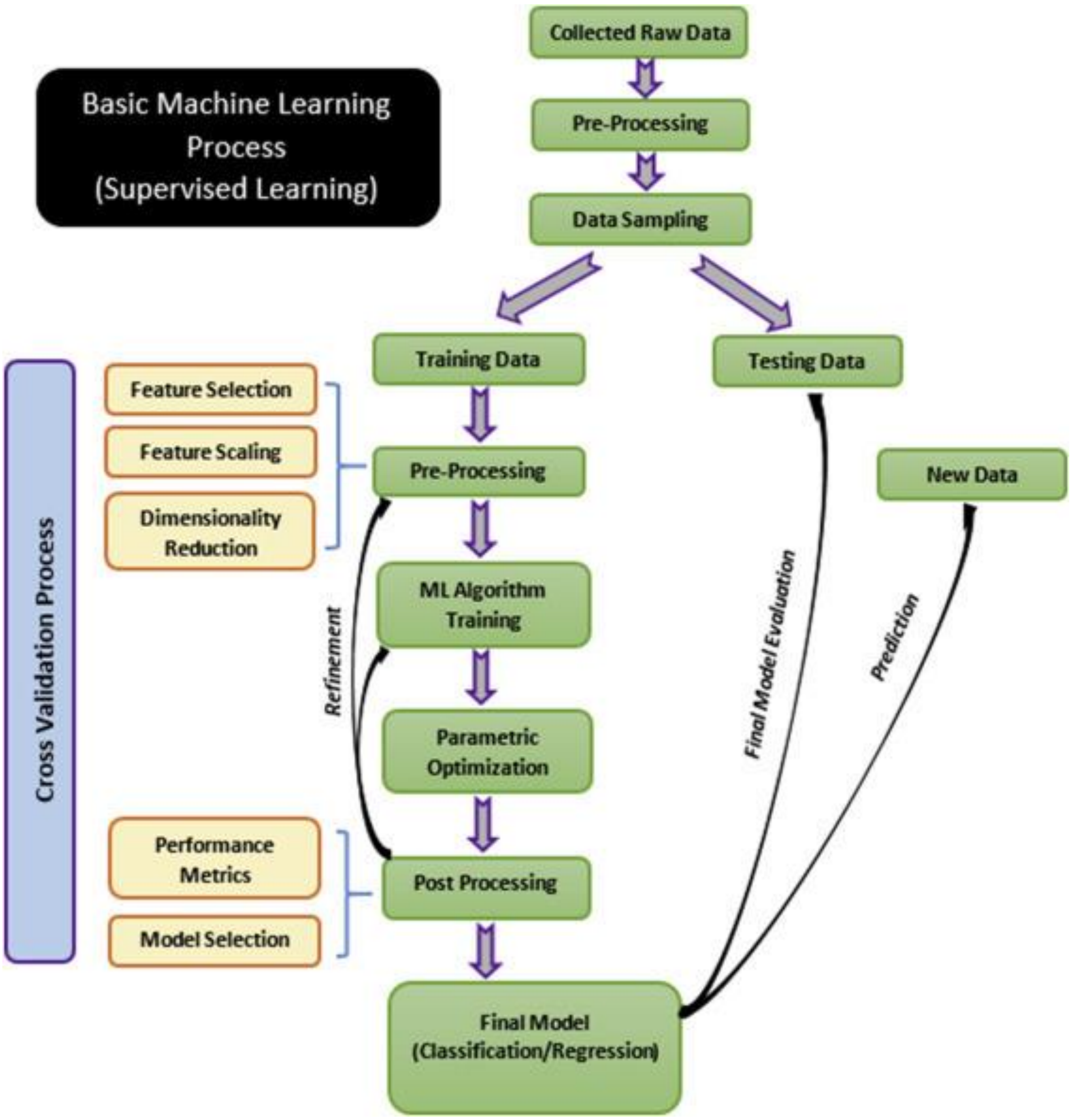
ML is also used to diagnose the disease based on x-ray images. For instance, chest images of patients can be used to detect whether a patient is infected with COVID-19 .

Moreover, social distancing can be monitored by ML; with the help of this approach, we can keep ourselves safe from COVID-19

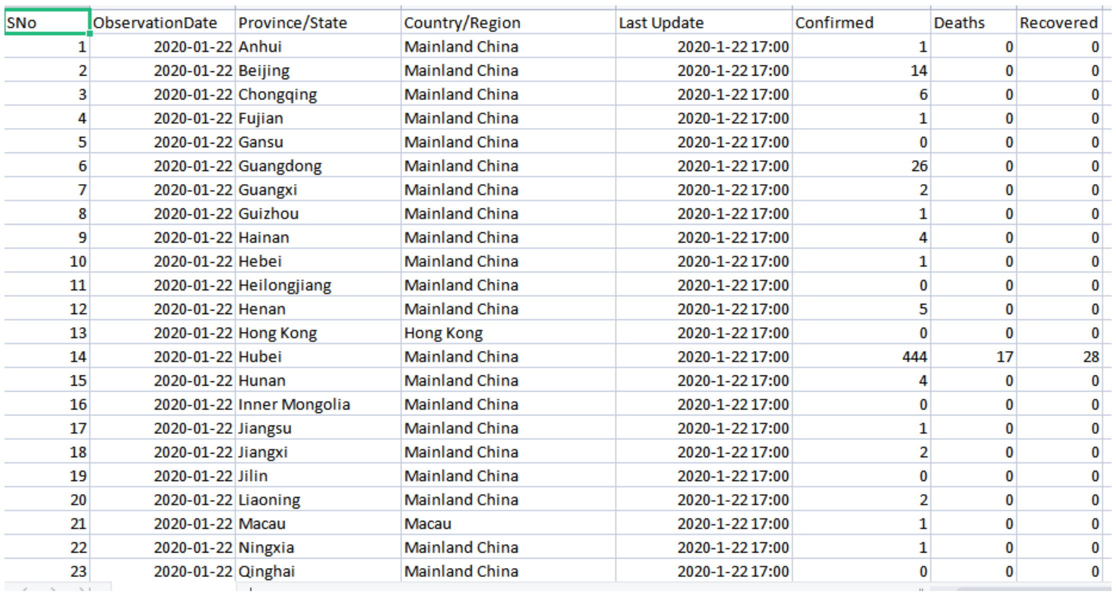
Different techniques for prediction and forecasting

Various ML techniques are used to predict and forecast future events. Some ML techniques used for prediction are support vector machine, linear regression, logistic regression, naive Bayes, decision trees (random forest and ETC), K-nearest neighbor, and neural networks (multilayer perceptron).

Similarly, some ML techniques used to forecast future events are naive approach, moving average, simple exponential smoothing, Holt's linear trend model, Holt-Winters model, Seasonal Autoregressive Integrated Moving Average Exxogenous Model (SARIMAX) and Autoregressive Integrated Moving Average Model (ARIMA).


Basic Machine Learning Process (Supervised Learning)
Collected Raw Data
Pre-Processing
Data Sampling
Training Data
Testing Data
New Data
Pre-Processing
ML Algorithm Training
Parametric Optimization
Post Processing
Final Model (Classification/Regression)
Cross Validation Process
Feature Selection
Feature Scaling
Dimensionality Reduction
Performance Metrics
Model Selection
Refinement
Final Model Evaluation
Prediction

5. DATASET Information

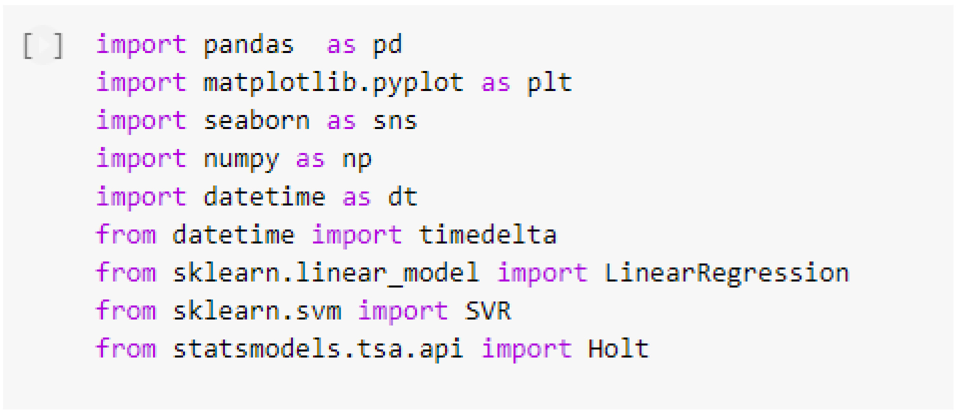


SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
1	2020-01-22	Anhui	Mainland China	2020-1-22 17:00	1	0	0
2	2020-01-22	Beijing	Mainland China	2020-1-22 17:00	14	0	0
3	2020-01-22	Chongqing	Mainland China	2020-1-22 17:00	6	0	0
4	2020-01-22	Fujian	Mainland China	2020-1-22 17:00	1	0	0
5	2020-01-22	Gansu	Mainland China	2020-1-22 17:00	0	0	0
6	2020-01-22	Guangdong	Mainland China	2020-1-22 17:00	26	0	0
7	2020-01-22	Guangxi	Mainland China	2020-1-22 17:00	2	0	0
8	2020-01-22	Guizhou	Mainland China	2020-1-22 17:00	1	0	0
9	2020-01-22	Hainan	Mainland China	2020-1-22 17:00	4	0	0
10	2020-01-22	Hebei	Mainland China	2020-1-22 17:00	1	0	0
11	2020-01-22	Heilongjiang	Mainland China	2020-1-22 17:00	0	0	0
12	2020-01-22	Henan	Mainland China	2020-1-22 17:00	5	0	0
13	2020-01-22	Hong Kong	Hong Kong	2020-1-22 17:00	0	0	0
14	2020-01-22	Hubei	Mainland China	2020-1-22 17:00	444	17	28
15	2020-01-22	Hunan	Mainland China	2020-1-22 17:00	4	0	0
16	2020-01-22	Inner Mongolia	Mainland China	2020-1-22 17:00	0	0	0
17	2020-01-22	Jiangsu	Mainland China	2020-1-22 17:00	1	0	0
18	2020-01-22	Jiangxi	Mainland China	2020-1-22 17:00	2	0	0
19	2020-01-22	Jilin	Mainland China	2020-1-22 17:00	0	0	0
20	2020-01-22	Liaoning	Mainland China	2020-1-22 17:00	2	0	0
21	2020-01-22	Macau	Macau	2020-1-22 17:00	1	0	0
22	2020-01-22	Ningxia	Mainland China	2020-1-22 17:00	1	0	0
23	2020-01-22	Qinghai	Mainland China	2020-1-22 17:00	0	0	0

the dataset consist datewise information of province based on the region for confirmed ,deaths and recovered cases.

6. implementation and result

6.1 Library used

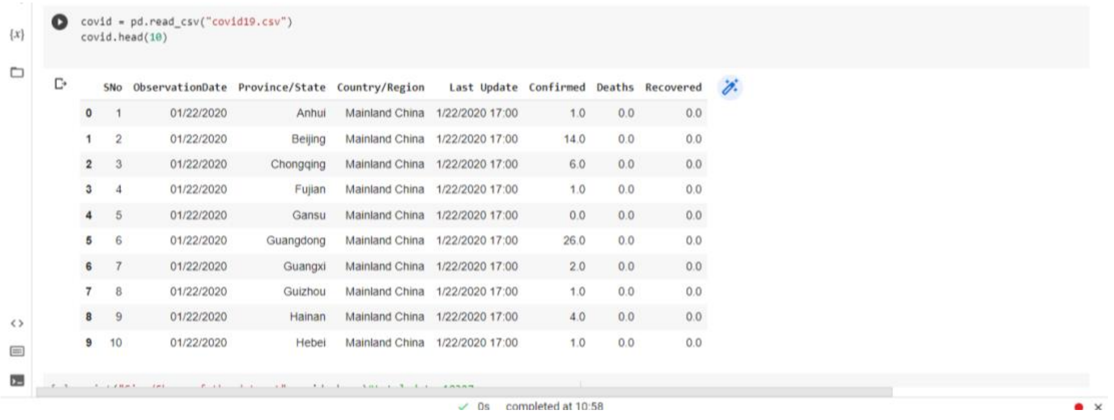


```
[ ]  import pandas  as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np
     import datetime as dt
     from datetime import timedelta
     from sklearn.linear_model import LinearRegression
     from sklearn.svm import SVR
     from statsmodels.tsa.api import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

- Pandas:Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

- NumPy: which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a Python package. It stands for 'Numerical Python'.
- Seaborn:Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper.
- Matplotlib:Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.
- Datetime:Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.
- Linear regression:linear_model is a class of the sklearn module if contain different functions for performing machine learning with linear models. The term linear model implies that the model is specified as a linear combination of features.
- Support vector machines:Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.
- Statsmodel:statsmodels. tsa contains model classes and functions that are useful for time series analysis. Basic models include univariate autoregressive models (AR), vector autoregressive models (VAR) and univariate autoregressive moving average models (ARMA).

6.2 Dataset information:

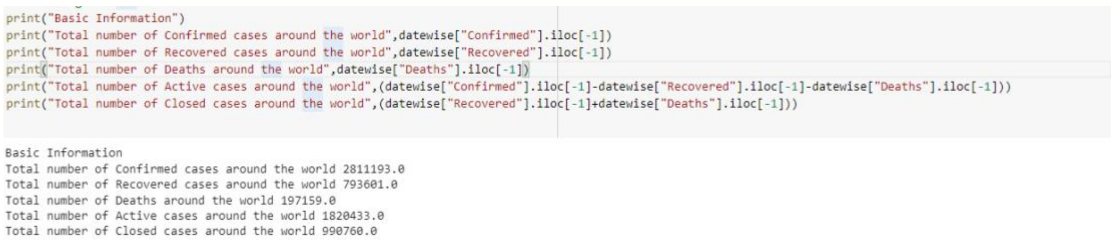


```
covid = pd.read_csv("covid19.csv")
covid.head(10)
```

	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
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

✓ 0s completed at 10:58

6.3 Information from dataset



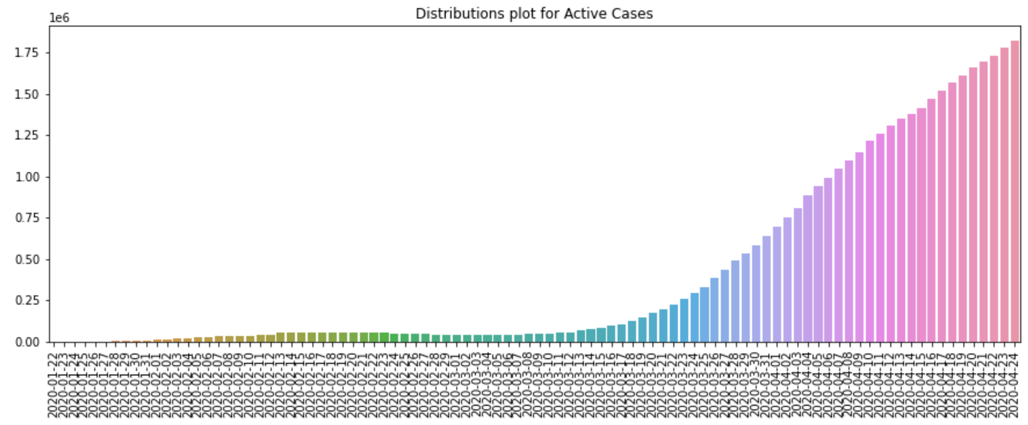
```
print("Basic Information")
print("Total number of Confirmed cases around the world",datewise["Confirmed"].iloc[-1])
print("Total number of Recovered cases around the world",datewise["Recovered"].iloc[-1])
print("Total number of Deaths around the world",datewise["Deaths"].iloc[-1])
print("Total number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

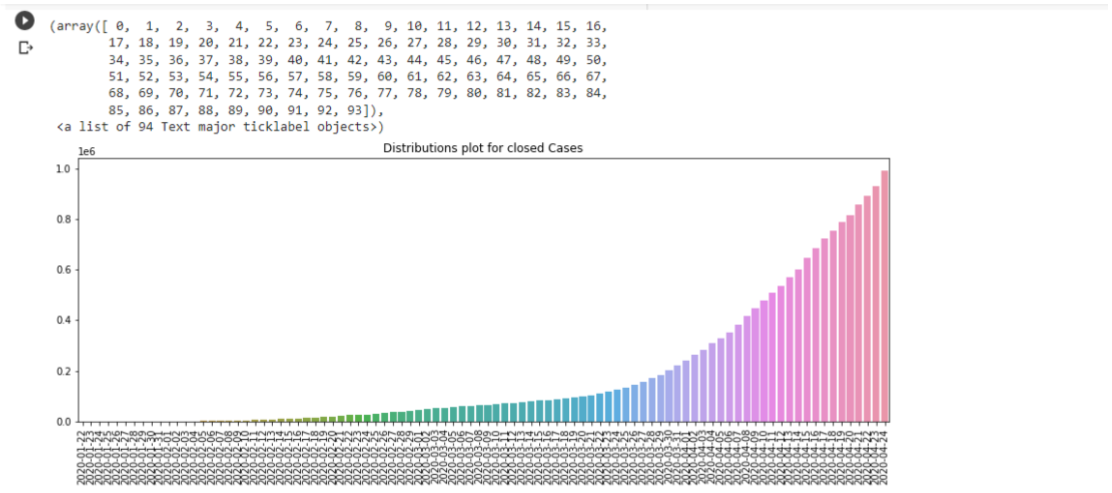
6.4 Cases Analysis

```
#datawise plotting active cases
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("Distributions plot for closed Cases")
plt.xticks(rotation=90)
```

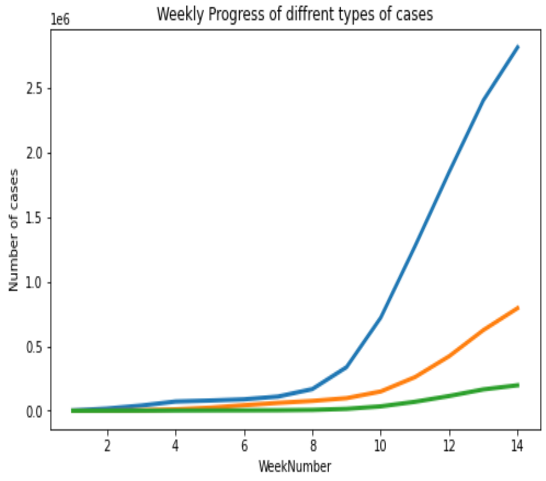


```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 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>)
```

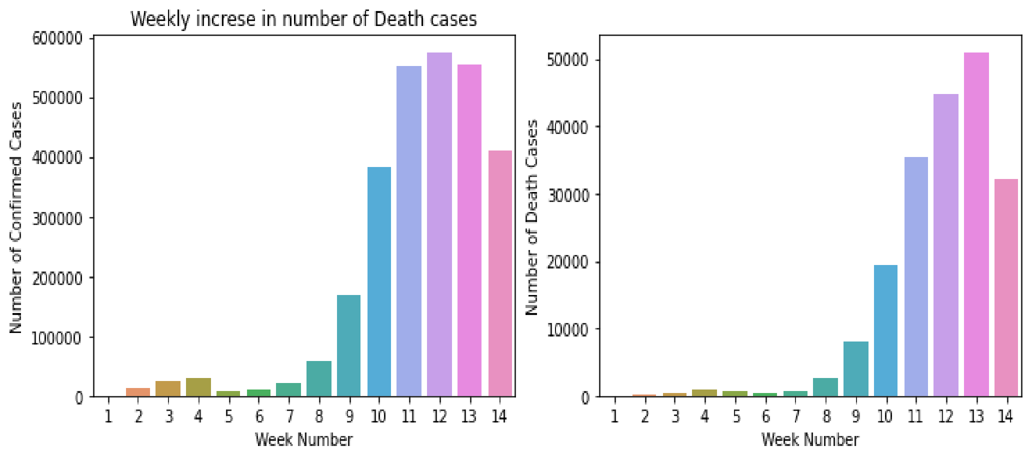
```
#weekly increse
datewise["WeekofYear"]=datewise.index.weekofyear
week_num=[]
weekwise_confirmed=[]
weekwise_recovered=[]
weekwise_deaths=[]
w=1
for i in list(datewise["WeekofYear"].unique()):
  weekwise_confirmed.append(datewise[datewise["WeekofYear"]==i]["Confirmed"].iloc[-1])
  weekwise_recovered.append(datewise[datewise["WeekofYear"]==i]["Recovered"].iloc[-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)#blue
plt.plot(week_num,weekwise_recovered,linewidth=3)#orange
plt.plot(week_num,weekwise_deaths,linewidth=3)#green
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of diffrent types of cases")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: weekofyear and week have been deprecated, please use DatetimeIndex.isocaler

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

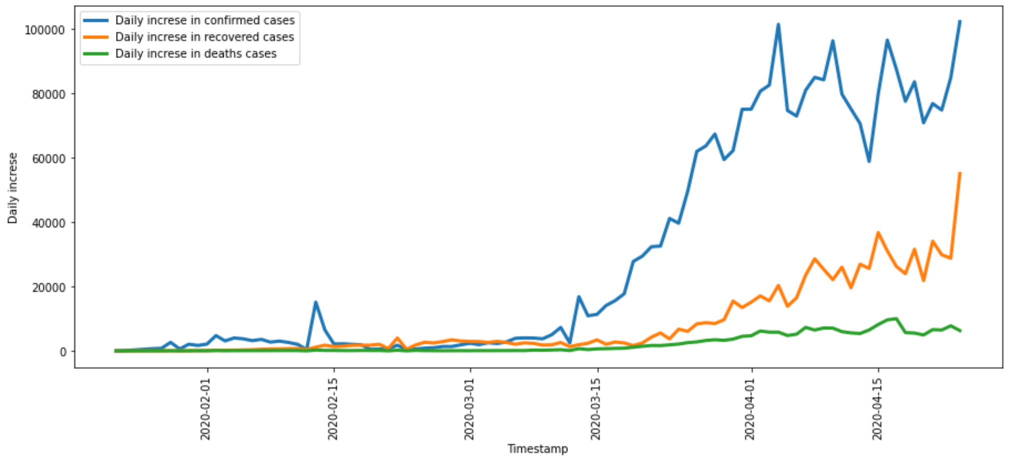


```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,4))
sns.barplot(x=week_num,y=pd.Series(weekwise_confirmed).diff().fillna(0),ax=ax1)
sns.barplot(x=week_num,y=pd.Series(weekwise_deaths).diff().fillna(0),ax=ax2)
ax1.set_xlabel("Week Number")
ax2.set_xlabel("Week Number")
ax1.set_ylabel("Number of Confirmed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



```
print("Average increse in number of Confirmed cases everyday:",np.round(datewise["Confirmed"].diff().fillna(0).mean()))
print("Average increse in number of Recovered cases everyday:",np.round(datewise["Recovered"].diff().fillna(0).mean()))
print("Average increse in number of Deaths cases everyday:",np.round(datewise["Deaths"].diff().fillna(0).mean()))
plt.figure(figsize=(15,6))
plt.plot(datewise["Confirmed"].diff().fillna(0),label="Daily increse in confirmed cases",linewidth=3)
plt.plot(datewise["Recovered"].diff().fillna(0),label="Daily increse in recovered cases",linewidth=3)
plt.plot(datewise["Deaths"].diff().fillna(0),label="Daily increse in deaths cases",linewidth=3)
plt.xlabel("Timestamp")
plt.ylabel("Daily increse")
plt.legend()#to show the labels
plt.xticks(rotation=90)
plt.show()
```

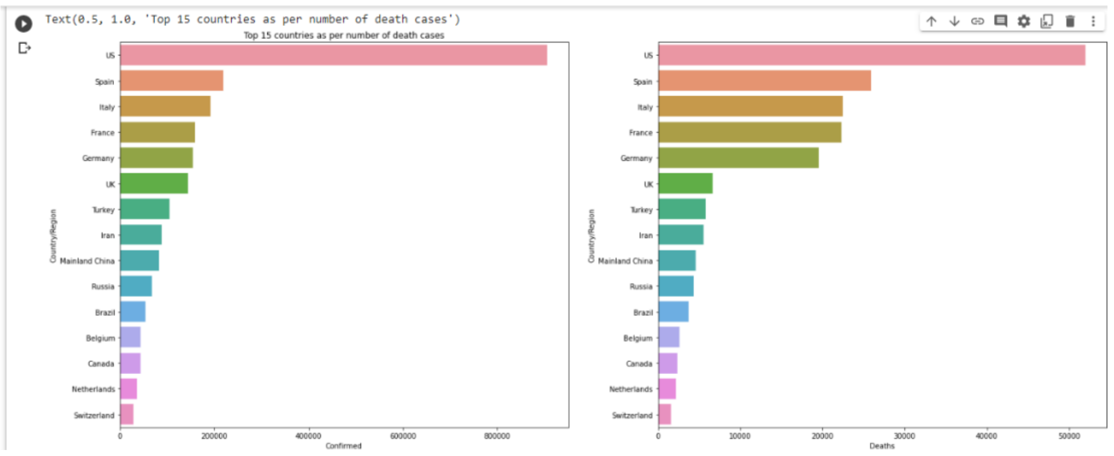
```
Average increse in number of Confirmed cases everyday: 29900.0
Average increse in number of Recovered cases everyday: 8442.0
Average increse in number of Deaths cases everyday: 2097.0
```



```
#country wise analysis

#country wise analysis
#calculating country wise mortality rate deathcase/confirmed case
countrywise=covid[covid["ObservationDate"]==covid["ObservationDate"].max()].groupby(["Country/Region"]).agg({"Confirmed":"sum","Recovered":"sum","Deaths":"sum"}).sort_values(["Confirmed"],ascending=False)
countrywise["Mortality"]=(countrywise["Deaths"]/countrywise["Recovered"])*100
countrywise["Recovery"]=(countrywise["Recovered"]/countrywise["Confirmed"]
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_15confirmed.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')
```

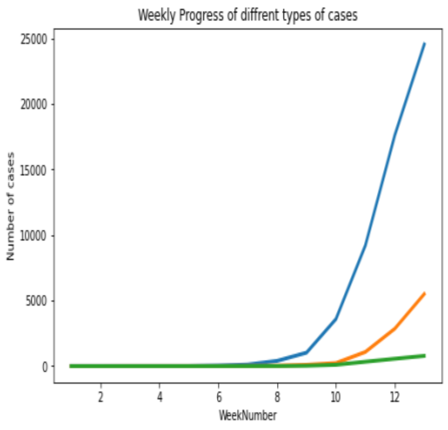
```
#india analysis
india_data=covid[covid["Country/Region"]=="India"]
datewise_india=india_data.groupby(["ObservationDate"]).agg({"Confirmed":"sum","
Recovered":"sum","Deaths":"sum"})
print(datewise_india.iloc[-1])
print("Total Active Cases",datewise_india["Confirmed"].iloc[-1]-
datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])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

```
#weekly progress for india
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 diffrent types of cases")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: weekofyear and week have been deprecated, please use DatetimeIndex.isocalendar()

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



Linear regression model:

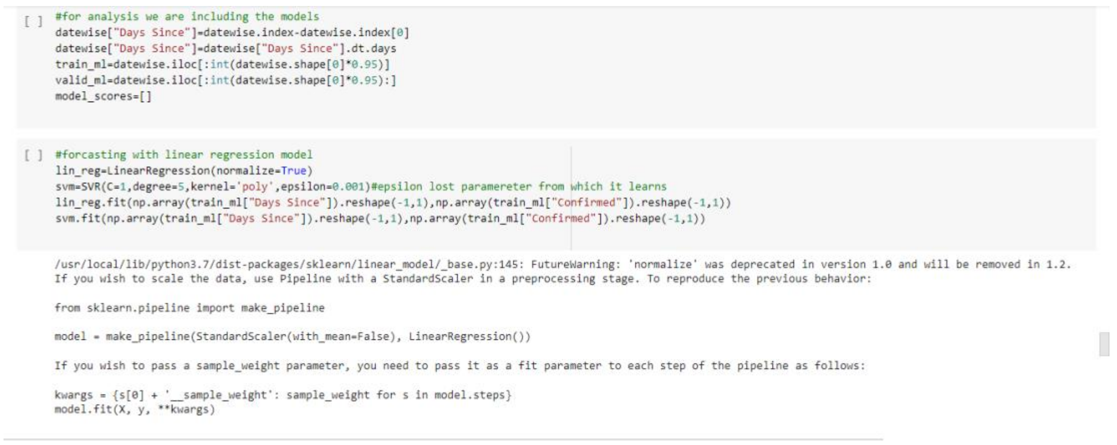
Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

Support Vector Machine:

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.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.



```
[ ] #for analysis we are including the models
    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=[]
```

```
[ ] #forcasting with linear regression model
    lin_reg=LinearRegression(normalize=True)
    svm=SVR(C=1,degree=5,kernel='poly',epsilon=0.001)#epsilon lost paramereter from which it learns
    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))
```

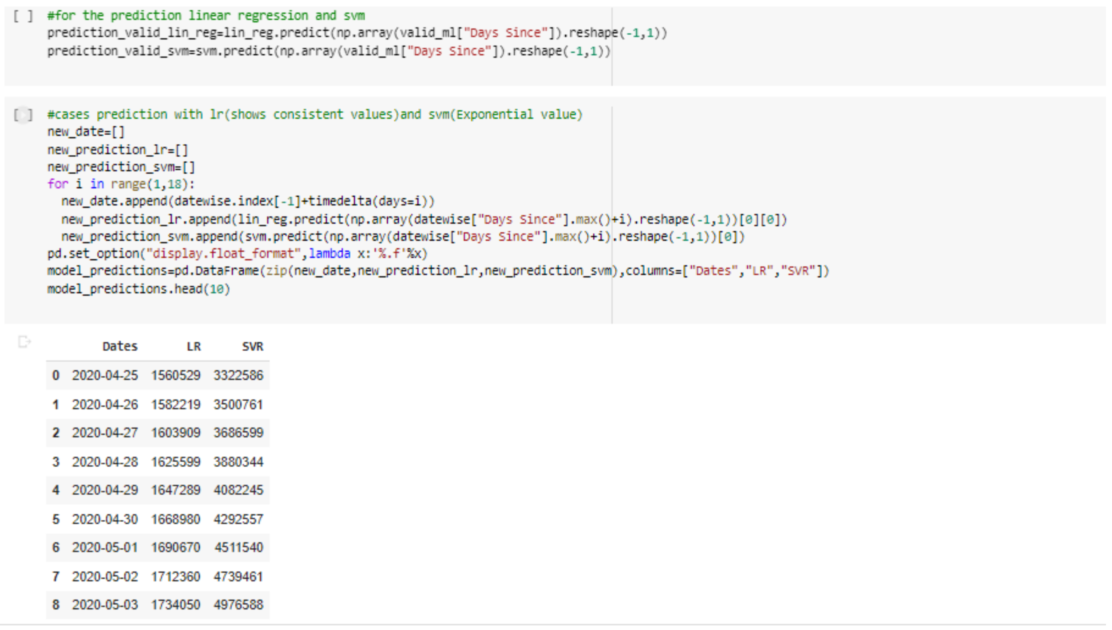
```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_base.py:145: FutureWarning: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing stage. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(with_mean=False), LinearRegression())

If you wish to pass a sample_weight parameter, you need to pass it as a fit parameter to each step of the pipeline as follows:

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
model.fit(X, y, **kwargs)
```



```
[ ] #for the prediction linear regression and svm
    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))
```

```
[ ] #cases prediction with lr(shows consistent values)and svm(Exponential value)
    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(10)
```

```
        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
5  2020-04-30  1668980  4292557
6  2020-05-01  1690670  4511540
7  2020-05-02  1712360  4739461
8  2020-05-03  1734050  4976588
```

Improvement in the model prediction using time series analysis:

- Time series forecasting is the process of analyzing time series data using statistics and modeling to make predictions and inform strategic decision-making. It's not always an exact prediction, and likelihood of forecasts can vary wildly—especially when dealing with the commonly fluctuating variables in time series data as well as factors outside our control. However, forecasting insight about which outcomes are more likely—or less likely—to occur than other potential outcomes. Often, the more comprehensive the data we have, the more accurate the forecasts can be. While forecasting and "prediction" generally mean the same thing, there is a notable distinction. In some industries, forecasting might refer to data at a specific future point in time, while prediction refers to future data in general. Series forecasting is often used in conjunction with time series analysis. Time series analysis involves developing models to gain an understanding of the data to understand the underlying causes. Analysis can provide the “why” behind the outcomes you are seeing. Forecasting then takes the next step of what to do with that knowledge and the predictable extrapolations of what might happen in the future.

- Holt-Winter’s Exponential Smoothing as named after its two contributors: Charles Holt and Peter Winter’s is one of the oldest time series analysis techniques which takes into account the trend and seasonality while doing the forecasting. This method has 3 major aspects for performing the predictions. It has an average value with the trend and seasonality. The three aspects are 3 types of exponential smoothing and hence the hold winter’s method is also known as triple exponential smoothing.
- Let us look at each of the aspects in detail.

Exponential Smoothing: Simple exponential smoothing as the name suggest is used for forecasting when the data set has no trends or seasonality.

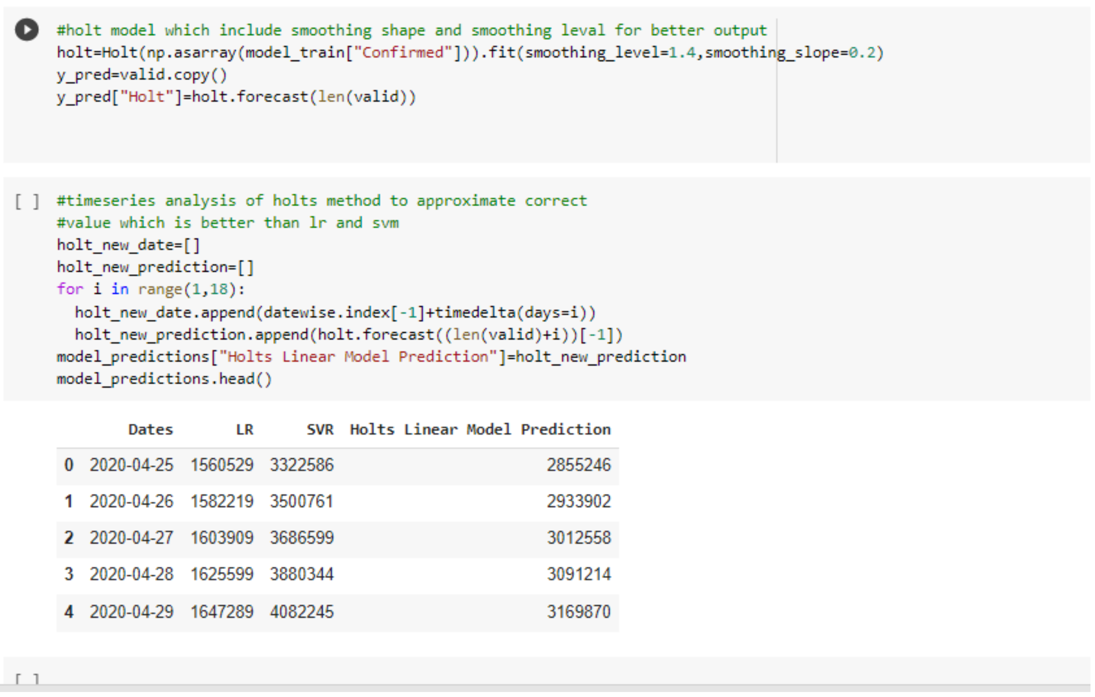
Holt’s Smoothing method: Holt’s smoothing technique, also known as linear exponential smoothing, is a widely known smoothing model for forecasting data that has a trend.

Winter’s Smoothing method: Winter’s smoothing technique allows us to include seasonality while making the prediction along with the trend.

Hence the Holt winter’s method takes into account average along with trend and seasonality while making the time series prediction.

Applocation of timeseries analysis:

Naturally, there are limitations when dealing with the unpredictable and the unknown. Time series forecasting isn’t infallible and isn’t appropriate or useful for all situations. Because there really is no explicit set of rules for when you should or should not use forecasting, it is up to analysts and data teams to know the limitations of analysis and what their models can support. Not every model will fit every data set or answer every question. Data teams should use time series forecasting when they understand the business question and have the appropriate data and forecasting capabilities to answer that question. Good forecasting works with clean, time stamped data and can identify the genuine trends and patterns in historical data. Analysts can tell the difference between random fluctuations or outliers, and can separate genuine insights from seasonal variations. Time series analysis shows how data changes over time, and good forecasting can identify the direction in which the data is changing.



```
#holt model which include smoothing shape and smoothing leval for better output
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))
```

```
[ ] #timeseries analysis of holts method to approximate correct
    #value which is better than lr and svm
    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
```

```
[ ]
```

7. Conclusion and Future work

The pandemic of COVID-19 has affected the entire globe. It has spread in more than 85 countries as of Apr. 2020. Scientists have made every effort to find solutions to it; according to claims by the United States and India, some vaccines have been made that are being trialed. The use of computers by scientists for early prediction has been widespread. A lot of research is taking place using ML to combat COVID-19. This chapter can be used by different researchers to learn how ML can be employed to forecast not only this situation but also other cases. The chapter specifically used the ARIMA method of time to forecast the stability and growth of COVID-19. Many countries have seen high totals of deaths owing to COVID-19. It is believed that the performance of the model can be improved or the model can give more accurate data if more datasets are available. The model gives results on the basis of data developed by information given by health agencies. Thus, forecasting may not be 100% accurate, but it can surely be used as a corrective measure. For future work further enhancement can be done by combining new factors and algorithms with ARIMA to get more accurate results.