# Abstract

Coronavirus disease 2019 (COVID-19) is currently destroying the world, causing more than 54 million confirmed cases and more than a million deaths, changing our lives every day, being banned globally, restricting restrictions, and improving physical hygiene. Therefore, the ability to study and monitor the spread of a disease-causing disease is now of great importance, especially as it helps to make important choices that have a significant impact on countries and their regions, their populations, and ensure effective resource management and benefits. In this project, we are developing a modeling machine designed to predict COVID-19 cases, deaths, and recoveries on the other side of a small scale using a modeling model.

# Project Ideation

Our main objective of this project is to predict the growth of COVID-19 in sub-continent countries which include Bangladesh, Bhutan, India, Maldives, Nepal, Pakistan, and Sri Lanka. The first step is to get the COVID-19 dataset of these countries. The next step is to import data in a jupyter notebook by using python library pandas then cleaning data as irrelevant data will affect our prediction. Now we will be visualizing data by using the python library Matplotlib to identify useful patterns and meaningful relationships. Now it’s time to build a machine learning model using different machine learning algorithms which includes Linear Regression, Logistic Regression, Polynomial Regression, K Nearest Neighbor, and Random Forest then we will be training our model on historical data and predict covid-19 cases for sub-continent countries and provide a predictive analytics report.

## **Motivation**

The motivation of the project is to experiment to see what are the appropriate machine learning algorithms that can be best applied to the given data and also to find out which algorithm gives us the best results in predicting COVID-19.

## **Objective**

The main objective of our project is,

* Identifying the most suitable machine learning technique for prediction.
* Preparing a machine learning model that could make accurate predictions of COVID-19.

## **Procedure**

Our approach was based on the Agile Scrum development method, which can be thought of as a series of short reflex prints. This seemed to be the most appropriate method of development due to its flexible and rigid nature, unlike ordinary sweat. way Developers have worked hard to achieve the set goals. The main reason for choosing Scrum in this method was that the problem could not be fully understood or explained at first and had to be fixed all the time, which seemed to be the best approach for our students who want to take all this development into account. To deal with experiential learning, we need to acquire new knowledge and apply general knowledge.

# Solution design

## Design

We used data from the GitHub repository of “COVID-19 Data Repository by the center for Systems Science and Engineering (CSSE) at John Hopkins University”, could be found here: <https://github.com/CSSEGISandData/COVID-19>

Detailed information of data is provided under the above link but here we give a summary of datasets and how we cleaned up the data.

The data includes seven datasets one for each subcontinent country:

Bangladesh

Bhutan

India

Maldives

Nepal

Pakistan

Sri Lanka

Each row of the datasets carries information about COVID-19 cases on daily basis.

## **Data Cleaning**

These datasets needed some cleaning and modification.

* + - Some of the countries in the start were only reporting COVID-19 cases for the whole country but after some time they reported COVID-19 cases province-wise. For Example, “Pakistan” was reporting cases as a single country after some time the report Data on province wise which we have to make the sum of all province cases reported and make a single row for Pakistan.
    - A dataset has no feature of lockdown and we also want to predict lockdown for subcontinent countries we added a feature as lockdown with binary values 0 means no lockdown 1 means lockdown.

Here is the complete list of features and a brief description

* + - Country: Name of subcontinent counties.
    - Date: Date on which COVID-19 cases were reported.
    - Confirmed: Total COVID-19 cases reported to date.
    - Deaths: Total death cases reported to date.
    - Recovered: Total recovered cases reported to date.
    - Active: Total active on that date.

Lockdown: Lockdown is imposed or not.

## **Algorithms of machine learning**

We have implemented five machine learning algorithms to determine which is best to predict COVID-19 new, deaths, and recovered cases. Following are the five ML algorithms we used:

## **Linear Regression**

A linear regression algorithm forecasts the relationship between a dependent variable and a single independent variable. The linear regression model produces a straight or diagonal connection, which is why it is called linear regression. The main idea of the variable and the linear deviation must be constant/real. However, the definition of a variable can be measured by either continuity or classification.

## **Regressions logistic**

A visual regression algorithm called logistic regression is used to estimate the six transfer goals. The expectation or shifting purpose is a two-way street, which means that only two categories are relevant. Simply put, binary variables depend on data rated 1 (meaning success / yes) or 0 (meaning failure / no).

## **Regression of polynomial**

Polynomial regression is a regression algorithm that forecasts the relationship between the nth polynomial and independence with respect to (y) (x). It is also called the Large Linear Regression ML technique. Because we add too many polynomial expressions to many classification lines to translate into polynomial regression. Here is an example of a test line with changes to improve accuracy.

## **K nearest neighbors**

The K-near-border (KNN) algorithm is a type of visual search algorithm that can be used to classify and predict problems. The K-Nearest Neighbors (KNN) algorithm uses "feature similarity" to predict the value of new information, meaning that the new information represents a value based on proximity.

## **Accidental forest**

Accidental forestry is a visually learned algorithm used for both classification and regression. However, it is still used in job classification. As we know, a forest is made up of trees, and the more trees a forest has, the more important it is. Similarly, a random jungle algorithm constructs tree points from sample data, then derives a prediction from each of them, and then selects the best answer with a query. This is a method that uses more than one option when you subtract overuse to average results.

**Required tools**

|  |  |
| --- | --- |
| Area specialisation | Data mining and machine learning |
| Total required | Python dash jupyter notebook |

## **Python**

Python provides a concise and easy-to-read user guide. Although knowledge of machine learning and programming extends beyond complex algorithms and functional changes, Python’s simplicity allows programmers to build powerful machines. Developers should do their best to solve the problem of machine learning instead of focusing on language skills. Equipped with a rich collection of technologies, Python has a wide range of libraries for intelligent programming and machine learning. Here are a few of them:

• Pandas for general-purpose data analysis

• Scikit-learn for machine learning

• NumPy for high-performance scientific computing and data analysis

• Data analysis with Matplotlib

## **Dash**

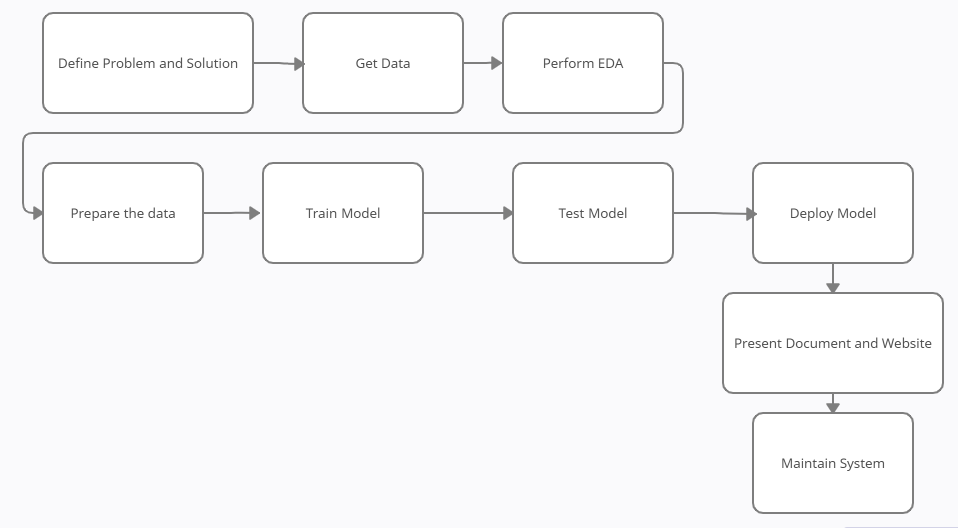
Dash is a Python based website development software. It is based on Flask, Plotly.js, React and React Js. You can create control panels using the Python plugin. Dash is open and its users are accessing the website. The dash application usually consists of two parts. The first part describes the design and definition of the application, and the second part describes the scope of the application. Dash provides HTML classes that allow us to create HTML objects in Python.

## **Jupyter Notebook**

Jupyter Notebook is open-source web software that allows you to create and share documents with live code, comparisons, visualizations, and narration. Applications: data cleaning and editing, numerical data, statistics, visual data, machine learning, etc.

## 

## **Flow diagram**



# Baseline Implementation

“import pandas as pd

from datetime import datetime,timedelta import numpy as np

import matplotlib.pyplot as plt”

“import warnings warnings.filterwarnings('ignore')

Pak\_data = pd.read\_csv("C:/Users/Snoke/Final Year Project/uk.csv") x=Pak\_data.loc[:,['Date']].values

y=Pak\_data.loc[:,['Recovered']].values”

“#Calculating for New Positive Cases per day confirmed=Pak\_data["Confirmed"] active\_day=[0]\*len(confirmed) active\_day[0]=confirmed[0]

for i in range(1,len(confirmed)): active\_day[i]=confirmed[i]-confirmed[i-1]

new\_cases= pd.DataFrame(data=active\_day, index=None, columns=["Daily\_Cases"])”

“#Calculating for New Deaths Cases per day deaths=Pak\_data["Deaths"] deaths\_day=[0]\*len(deaths) deaths\_day[0]=deaths[0]

for i in range(1,len(deaths)): deaths\_day[i]=deaths[i]-deaths[i-1]

new\_deaths= pd.DataFrame(data=deaths\_day, index=None, columns=["Daily\_Deaths"])”

“#Calculating for New Recovered Cases per day recovered=Pak\_data["Recovered"] recovered\_day=[0]\*len(recovered) recovered\_day[0]=recovered[0]

for i in range(1,len(recovered)): recovered\_day[i]=recovered[i]-recovered[i-1]

new\_recovered= pd.DataFrame(data=recovered\_day, index=None, columns=["Daily\_Recovered"])”

“Pak\_data = pd.concat([Pak\_data,new\_cases,new\_deaths,new\_recovered], axis=1) Pak\_data['Date'] = pd.to\_datetime(Pak\_data['Date'])

Pak\_data.head()

variables = Pak\_data.columns Pak\_data.isnull().sum().loc[variables]

plt.plot(Pak\_data['Date'],Pak\_data['Daily\_Cases']) plt.plot(Pak\_data['Date'],Pak\_data['Daily\_Deaths']) plt.plot(Pak\_data['Date'],Pak\_data['Daily\_Recovered']

dates= Pak\_data.Date.keys()

Pak\_data['Date'] = np.array([i for i in range(len(dates))]).reshape(-1,1)”

“train = Pak\_data[Pak\_data.index < 280] test = Pak\_data[Pak\_data.index >= 280]

x\_train = train[["Date"]].values y\_train = train["Recovered"].values x\_test = test[["Date"]].values y\_test = test["Recovered"].values

regressor=LinearRegression() regressor.fit(x\_train,y\_train) l\_pred\_pak\_r=regressor.predict(x\_test)”

“linear\_a=explained\_variance\_score(y\_test, l\_pred\_pak\_r) linear\_a=linear\_a\*100

linear\_a

linear\_e=mean\_absolute\_error(y\_test, l\_pred\_pak\_r) linear\_e

poly=PolynomialFeatures(degree=2) x\_poly=poly.fit\_transform(x\_train) reg=LinearRegression() reg.fit(x\_poly,y\_train)

p\_pred\_pak\_r=reg.predict(poly.fit\_transform(x\_test))

poly\_a=explained\_variance\_score(y\_test, p\_pred\_pak\_r) poly\_a=poly\_a\*100

poly\_a

poly\_e=mean\_absolute\_error(y\_test, p\_pred\_pak\_r) poly\_e”

“error=[]

for k in range(1,50): knn=KNeighborsRegressor(n\_neighbors=k) y\_pred=cross\_val\_predict(knn,x\_train,y\_train,cv=5) error.append(mean\_absolute\_error(y\_train,y\_pred))

plt.plot(range(1,50),error)

knn= KNeighborsRegressor(n\_neighbors=5) knn.fit(x\_train,y\_train)

knn\_pred\_pak\_r = knn.predict(x\_test)”

“knn\_a=explained\_variance\_score(y\_test,knn\_pred\_pak\_r) knn\_a=knn\_a\*100

knn\_a

knn\_e=mean\_absolute\_error(y\_test, knn\_pred\_pak\_r) knn\_e

RFReg=RandomForestRegressor(n\_estimators=100,random\_state=0) RFReg.fit(x\_train,y\_train)

r\_predict\_pak\_r=RFReg.predict(x\_test)

random\_a=explained\_variance\_score(y\_test,r\_predict\_pak\_r) random\_a=random\_a\*100

random\_a

random\_e=mean\_absolute\_error(y\_test, r\_predict\_pak\_r) random\_e”

“import matplotlib.pyplot as plt fig = plt.figure() fig.set\_figheight(5) fig.set\_figwidth(15)

ax1= fig.add\_subplot(121)

ax2 = fig.add\_subplot(122) ax1.set\_title('Model Accuracy')

algo = ['Linear', 'Polynomial', 'KNN', 'Random Forest'] accu = [linear\_a,poly\_a,knn\_a,random\_a]

err = [linear\_e,poly\_e,knn\_e,random\_e] ax1.bar(algo,accu)

ax1.set\_title('Model Accuracy') ax2.set\_title('Model Error')”

“ax1.bar(algo,acu) ax2.bar(algo,err)”