HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FALCUTY OF COMPUTER SCIENCE AND ENGINEERING

DISCRETE MATHEMATICS

COVID-19 ANALYSIS AND PREDICTION

Kamen Rider

1. Nguyen Nam Kha (2052515)
2. Nguyen Ngoc Hoa (2052485)
3. Duong Ngoc Quang Huy (2052489)
4. Nguyen Hoang Anh Thu (2053478)
5. Nguyen Hong Quan (2052228)

GROUP ASSIGNED TASK TABLE

No.	Name	StudentID	Tasks	Contribution	Results
1	Nguyen Nam Kha	2052515	build and analyse model, write content for report	20%	100%
2	Nguyen Ngoc Hoa	2052485	build model, write report in Latex, create report slide	20%	100%
3	Duong Ngoc Quang Huy	2052489	collect the data, analyse pandemic situation	20%	100%
4	Nguyen Hoang Anh Thu	2053478	analyse data, write conclusion, prepare Latex form	20%	100%
5	Nguyen Hong Quan	2052228	analyse pandemic situation, write introduction	20%	100%

Leader

Nguyen Nam Kha

Phone: 0916927299

Email: kha.nguyen02@hcmut.edu.vn

Contents

1. Introduction

At the end of 2019, the new coronavirus (COVID-19) spread widely in China, and a large number of people became infected. On March 11 2020, the World Health Organization (WHO) declared a new pneumonia outbreak a "global pandemic." At present, the pandemic has spread rapidly in almost every country and region around the world. Currently, Asia has become the center of the current outbreak of new pneumonia. The new coronavirus has caused a great threat to the health and safety of people all over the world due to its amazing spreading power and potential harm. The research on the domestic and international epidemics and the future development trend has become a hot topic of current research. That is a reason why my team conducted this research to study the trend of Covid-19 and be able to make a prediction about the pandemic situation of some countries around the world..

According to the trend of epidemic at different stages, this paper uses both Polynomial Regression and Linear Regression depending on the object of the prediction and the effectiveness of the long-term or short-term period, to construct a new model of the pandemic trend. The study points out the key factors that affect the spread of the virus in different countries, such as the percentage of people vaccinated, daily infection number, and the appearance of new pneumonia. Relevant models and data analysis may provide some basis and guidance for the related countries about epidemic prevention and control.

2. Methods

In this report, we will use Python to implement the task of collecting data as well as building models. Python is an interpreted high-level general-purpose programming language. Python's language construct emphasizes code readability and aims to help programmers write clear, logical code for small and large-scale projects.

Python's large standard library, cited as one of its greatest strengths, provides tools suited to many tasks, for example: Automation, Graphical user interfaces, Image processing, Scientific computing,... In this report, we will use some Python's libraries that can help us implement such tasks like *Data collecting, Data analytics and Machine learning :*

- `Code`

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
```

- `pandas` library offers data structures and operations for manipulating numerical tables and time series.
- `numpy` library has a large collection of high-level mathematical functions to operate on arrays and matrices.
- `matplotlib` library is used for creating interactive visualizations in Python like plotting graphs.
- `sklearn` library supports building prediction models by applying many mathematical functions, specifically *Linear Regression* and *Polynomial Regression*.

Our source code is stored in the file named "source_code.ipynb". You can open that file using Google Colab or Jupyter Notebook.

3. Data Collecting

3.1 Get raw data

Our data will be collected from an organization called "Our World in Data (OWID)". OWID is a scientific online publication that focuses on large global problems such as poverty, disease, hunger, climate change, war, existential risks, and inequality. OWID usually stores their raw data on their Github page. Here is the link of the data source that we use:

https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-data.csv

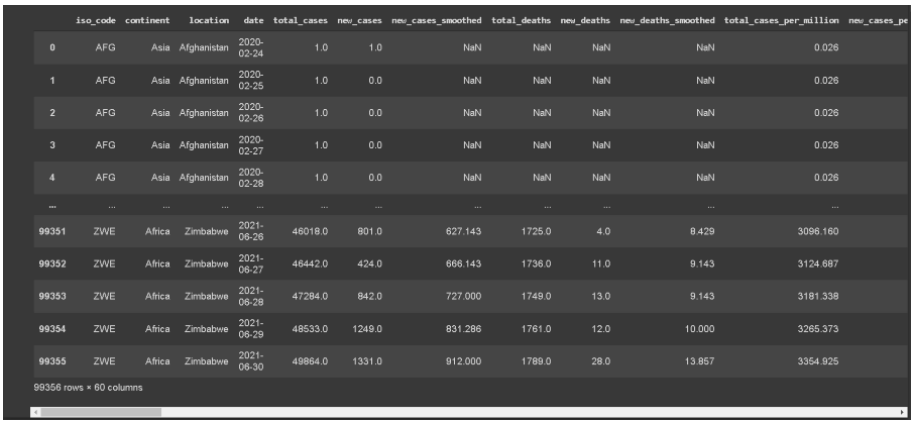
Our data will be stored in a `.csv` file. `.csv` is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format (*comma-separated value*), a `.csv` file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

We will get and read the raw `.csv` data file from the link below:

- Code

```
data_path =
'https://raw.githubusercontent.com/owid/covid-19-data/master/pu
blic/data/owid-covid-data.csv'
df = pd.read_csv(data_path)
df
```

- Result



	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_pe
0	AFG	Asia	Afghanistan	2020-02-24	1.0	1.0	NaN	NaN	NaN	NaN	0.026	
1	AFG	Asia	Afghanistan	2020-02-25	1.0	0.0	NaN	NaN	NaN	NaN	0.026	
2	AFG	Asia	Afghanistan	2020-02-26	1.0	0.0	NaN	NaN	NaN	NaN	0.026	
3	AFG	Asia	Afghanistan	2020-02-27	1.0	0.0	NaN	NaN	NaN	NaN	0.026	
4	AFG	Asia	Afghanistan	2020-02-28	1.0	0.0	NaN	NaN	NaN	NaN	0.026	
...	...	...	...	...	...	...	...	...	...	...	...	
99351	ZWE	Africa	Zimbabwe	2021-06-26	46018.0	801.0	627.143	1725.0	4.0	8.429	3096.160	
99352	ZWE	Africa	Zimbabwe	2021-06-27	46442.0	424.0	666.143	1736.0	11.0	9.143	3124.687	
99353	ZWE	Africa	Zimbabwe	2021-06-28	47284.0	842.0	727.000	1749.0	13.0	9.143	3181.338	
99354	ZWE	Africa	Zimbabwe	2021-06-29	48533.0	1249.0	831.286	1761.0	12.0	10.000	3265.373	
99355	ZWE	Africa	Zimbabwe	2021-06-30	49864.0	1331.0	912.000	1789.0	28.0	13.857	3354.925	

99356 rows × 60 columns

`describe()` will give a brief descriptive statistics about our data.

- Code

```
df.describe()
```

- Result

	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	r
count	9.578000e+04	95777.000000	94767.000000	8.565500e+04	85811.000000	94767.000000	95270.00000	
mean	1.059881e+06	6037.627238	6066.738144	2.869650e+04	146.607265	131.997344	13056.17409	
std	7.261411e+06	37783.276618	37546.533365	1.712165e+05	801.979927	745.245682	24023.81165	
min	1.000000e+00	-74347.000000	-6223.000000	1.000000e+00	-1918.000000	-232.143000	0.00100	
25%	1.257000e+03	2.000000	7.429000	5.400000e+01	0.000000	0.000000	260.21625	
50%	1.345700e+04	73.000000	91.857000	3.900000e+02	2.000000	1.429000	1787.44200	
75%	1.432235e+05	806.000000	849.214500	3.769500e+03	18.000000	14.429000	13609.57650	
max	1.822024e+08	906017.000000	826388.429000	3.947020e+06	18050.000000	14737.000000	180042.71000	

3.2 Take useful features for analysing data

The raw data contains lots of features, including *total_deaths, reproduction_rate*, positive_rate, tests_per_case, tests_units,... However, we will only use a few of these features for analyzing. So first, we will take out the useful features:

- Location (to get the country name)
- Date
- New cases

- Code

```
features_data = ['location','date','new_cases']
data_analyze = df[features_data]
data_analyze
```

- Result

	location	date	new_cases
0	Afghanistan	2020-02-24	1.0
1	Afghanistan	2020-02-25	0.0
2	Afghanistan	2020-02-26	0.0
3	Afghanistan	2020-02-27	0.0
4	Afghanistan	2020-02-28	0.0
...	...	...	...
99351	Zimbabwe	2021-06-26	801.0
99352	Zimbabwe	2021-06-27	424.0
99353	Zimbabwe	2021-06-28	842.0
99354	Zimbabwe	2021-06-29	1249.0
99355	Zimbabwe	2021-06-30	1331.0

99356 rows × 3 columns

3.3 Fix NaN cells

`NaN` cells are cells that have missing data.

There are many ways to fix `NaN` cells. In this situation, we will fix them by replacing `NaN` with 0.

- Code

```
for columns in data_analyze.columns:
    data_analyze[columns].fillna(0,inplace=True)
```

3.4 Get data of the specific object

To have an overview of the pandemic, we will analyze the general data of the entire *World*.

Therefore, we have to filter out `World` in feature `location`.

- Code

```
global_data = data_analyze[data_analyze['location']=='World']
global_data
```

- Result

	location	date	new_cases
97445	World	2020-01-22	0.0
97446	World	2020-01-23	98.0
97447	World	2020-01-24	286.0
97448	World	2020-01-25	492.0
97449	World	2020-01-26	685.0
...	...	...	...
97966	World	2021-06-26	362819.0
97967	World	2021-06-27	309889.0
97968	World	2021-06-28	331944.0
97969	World	2021-06-29	384746.0
97970	World	2021-06-30	391839.0

526 rows × 3 columns

3.5 Sort by Month

For display efficiency, we will group data into months, and calculate the total of new cases of each month.

We will take out data from `2020/07 to 2020/12` - the period *before we used the vaccine;* and from `2021/01 to 2021/05` - the period *when we started using the vaccine.*

- Code

```
global_data['year-month'] = global_data['date'].apply(lambda x:
str(x[:4]) + '-' + str(x[5:7]))
global_data = global_data.loc[:,('year-month','new_cases')]
global_data = global_data.groupby('year-month').sum()
# The period before we used the vaccine
global_data1 = global_data[6:17]
# The period when we started using the vaccine
global_data2 = global_data[12:17]
```

- Result

year-month	new_cases
2020-07	7146823.0
2020-08	7904329.0
2020-09	8502664.0
2020-10	12130624.0
2020-11	17268663.0
2020-12	19326166.0
2021-01	19469170.0
2021-02	11152787.0
2021-03	14718163.0
2021-04	22502768.0
2021-05	19676768.0

year-month	new_cases
2021-01	19469170.0
2021-02	11152787.0
2021-03	14718163.0
2021-04	22502768.0
2021-05	19676768.0

4. Data Analysis

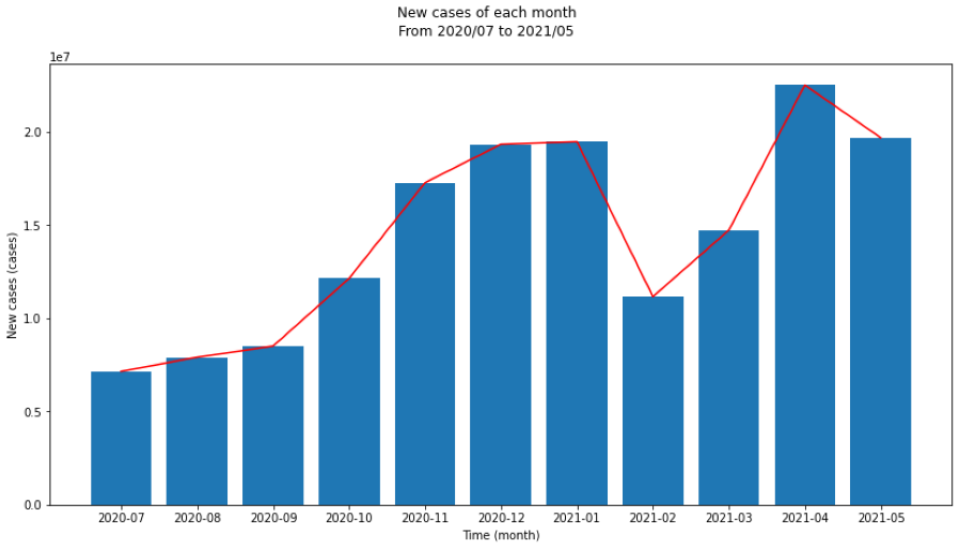
4.1 Plot the data

First we have to plot the data by drawing a *mixed chart* - a combination of a *bar chart* and a *line graph.*

- Code

```
fig_world, (ax1) = plt.subplots(1, 1,figsize=(14,7))
fig_world.suptitle("New cases of each month\nFrom 2020/07 to
2021/05")
ax1.bar(global_data1.index,global_data1['new_cases'])
ax1.plot(global_data1.index,global_data1['new_cases'],'r-')
#ax1.set_title("New cases of each month\nFrom 2020/07 to
2020/12")
ax1.set_xlabel('Time (month)')
ax1.set_ylabel('New cases (cases)')
```

- Result



4.2 Analysis data

From July 2020 to December 2020, the tendency seemingly resembles a normal distribution in which two conditions remained unchanged – the spread of the virus and the quick calls for lockdowns worldwide.

In January 2021, the figure followed the normal distribution. However, the effect of mass vaccination started to show. Expectedly, the number in February 2021 should be similar to that in November 2020. Nonetheless, the figure was significantly lower, which is a clear indicator of the effects of vaccination and lockdowns at the same time. In the first two months of 2021, the tendency follows an exponential distribution.

However, the number climbed drastically in the next two months, following the trajectory of a gamma distribution with K= 7,5 and theta=1.0. This sudden surge may result from lagging vaccination campaigns worldwide and too-soon reopens in multiple countries. However, a decline was reported again in May 2021, indicating that more lockdowns and successful vaccinations had occurred.

5. Model Prediction

5.1 About Linear Regression and Poly Regression

5.1.1 Theory

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. If the goal is prediction, forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.

So what is a *linear regression model?*

Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where T denotes the transpose, so that $\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$

Often these n equations are stacked together and written in matrix notation as $\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$, where

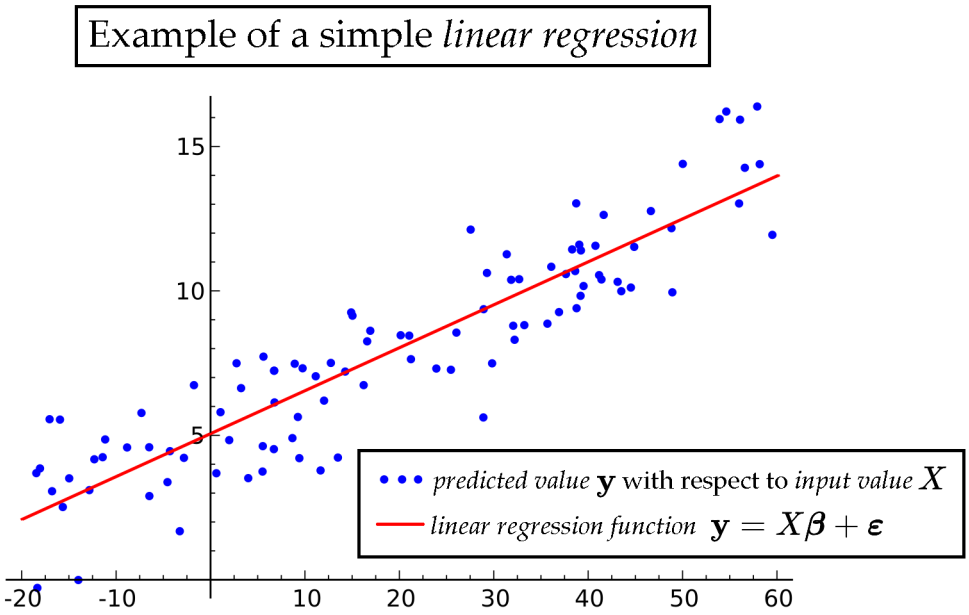
$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Some remarks on notation and terminology:

- $\boldsymbol{y}$ is a vector of observed $y_i \ (i = 1, \ldots, n)$ of the variable called *predicted variable*
- $\boldsymbol{X}$ is a matrix of vectors which are known as *input variables.*
- $\boldsymbol{\beta}$ has elements which are known as *effects or regression coefficients*. Statistical estimation and inference in *linear regression* focuses on $\boldsymbol{\beta}$. The elements of this parameter vector are interpreted as the partial derivatives of $\boldsymbol{y}$ with respect to $\boldsymbol{X}$.
- $\boldsymbol{\varepsilon}$ is a vector called *error term*. This variable captures all other factors which influence the dependent variable $\boldsymbol{y}$ other than the regressors $\boldsymbol{X}$.



So the theory of *linear regression* is very simple: we have to find the *regression coefficient* $\boldsymbol{\beta}$ so that the red-line *function* best fits the blue-dot *value*, which means the *error term* $\boldsymbol{\varepsilon} = \mathbf{y} - X\boldsymbol{\beta}$ is minimized. After that, we will use this red-line *function* to predict the desired value.

Polynomial Regression is quite similar to *linear regression*. The difference between them is the relationship between the independent variable $\boldsymbol{X}$ and the dependent variable $\boldsymbol{y}$ is modelled as an nth degree polynomial in $\boldsymbol{X}$. For example:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon.$$

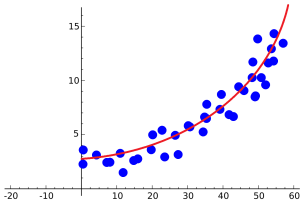
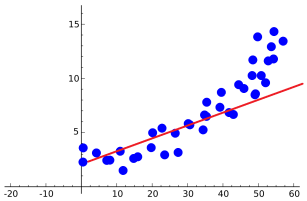
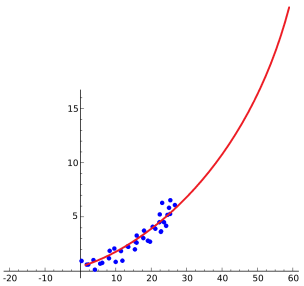
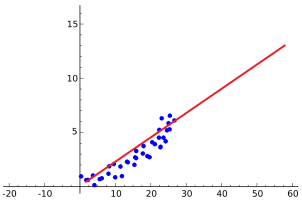
The function above is the *quadratic model* of the form. *Polynomial regression* fits a nonlinear relationship between the value of $\boldsymbol{X}$ and the corresponding conditional mean of $\boldsymbol{y}$. Therefore, we can yield a general function of polynomial regression model:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$$

The goal of both *linear regression* and *polynomial regression* analysis is to model the expected value of a *dependent variable* $\boldsymbol{y}$ in terms of the value of an *independent variable* (or vector of independent variables) $\boldsymbol{x}$. This report

will use these two models to predict the epidemic situation of some specific places around the world in the period of one-week and three-month later.

We will use *polynomial regression* for the *one-week* period. However for the case of *three-month* period, we have to use *linear regression*, here's the reason why:

	Polynomial Regression	*Linear Regression*
one week	 Predicted value is highly accurated	 Model is underfitted The accuracy of predicted value is low
three months	 Model is overfitted Predicted value might be too large or too small	 Predicted value is accurated and reasonable.

As we can see from the table above, if we use *linear regression* to predict such a short period like one-week, the model might be underfitted and our predicted value might be inaccurate. Likewise, applying *polynomial regression* on a *three-month* prediction might cause the model to be overfitted, which means our predicted value will be impractical.

5.1.2 Create doMath function

Applying the theory above, now we will build a function called `doMath` to do the math stuff.

This function receives *3 parameters:*

- The training sets (`x, y`)
- Number of days after 1 week and 3 months (`ow_norm, tm_norm`)
- Identified parameter (`check`)

And it will return *4 parameters:*

- Poly Regression function (`poly_re`)
- Linear Regression function (`lin_re`)
- Total cases predicted after 1 week and 3 months (`ow_re, tm_re`)
- Accuracy of the model(`BestAccuracy`)

- Code

```
def doMath(x,y,ow_norm,tm_norm,check):
    # Initialize accuracy
    BestAccuracy=0
    # We evaluate the accuracy by splitting the data into 2
parts:
    # 80% of them will be used for training and 20% will be
used for testing
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state = 10)
    # Create return values of Poly Regression, Linear
Regression and 2 predicted values.
    poly_re = PolynomialFeatures()
    lin_re = LinearRegression()
    ow_re = 0
    tm_re = 0
    # This algorithm uses a for-loop to find the degree 'i' of
the prediction line that best suits the training sets.
    for i in range(2,51):
        poly_reg = PolynomialFeatures(degree = i)
        # Train data
```

```
        x_poly_test = poly_reg.fit_transform(x_test)
        # Create Linear and Poly function based on the trained
data
        lin_reg = LinearRegression()
        lin_reg.fit(x_poly_train, y_train)
        lin_reg_t = LinearRegression()
        lin_reg_t.fit(x_train,y_train)
        # We will use Poly Regression model to predict the
total cases after one week, since this model can produce the
best result in the near future
        # We will use the Linear Regression model to predict
the total cases after three months, since the Poly model can
be overfitted and gives an unreasonable result.
        ow =
lin_reg.predict(poly_reg.fit_transform(ow_norm))[0]
        tm = lin_reg_t.predict(tm_norm)[0]
        # Calculate the accuracy
        temp = lin_reg.score(x_poly_test,y_test)
        if(check == 1 and temp > BestAccuracy):
            poly_re = poly_reg
            lin_re = lin_reg
            BestAccuracy=temp
            ow_re = ow
            tm_re = tm
        if(check == 2 and tm>ow and ow>1):
            poly_re = poly_reg
            lin_re = lin_reg
            BestAccuracy=temp
            ow_re = ow
            tm_re = tm
    return [poly_re, lin_re, ow_re,tm_re, BestAccuracy]
```

5.2 Global

5.2.1 Building model

Now we will build a model to predict COVID-19 situation of the *World.*

Step 1: Get data from open source, clean data by taking out useful features

First we will get raw data from open source, then clean the data by taking out the useful features.

- Code

```
df=pd.read_csv("https://raw.githubusercontent.com/owid/covid-19
-data/master/public/data/owid-covid-data.csv")
df = df[df['location'] == 'World']
df=df.loc[:,('location','date','new_cases','total_cases','peopl
e_vaccinated')]
local_model = df.copy()
n = len(local_model.index)
inde = np.array([x for x in range(n)])
local_model['indexx'] = inde
```

We need to consider data from the day we started using the vaccine until today, which is 2021/06/09, so we will slice the dataframe in this period.

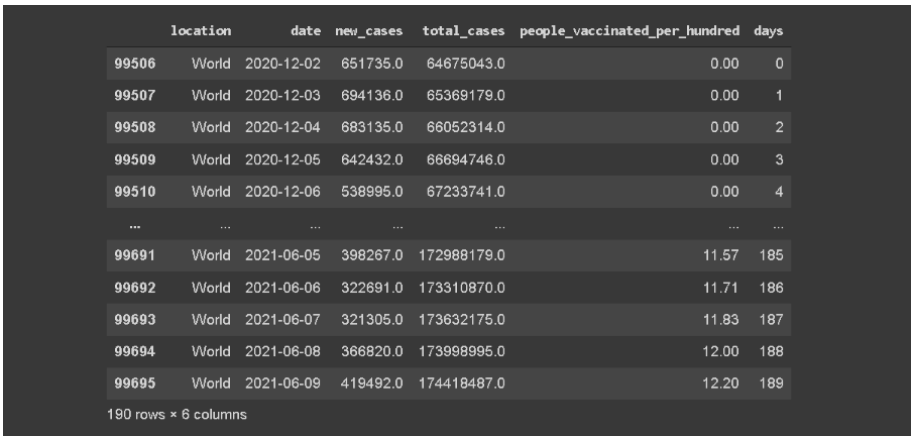
Then we will add a new column called `days`, this column presents the number of days.

- Code

```
#We will consider data from the day we started using the
vaccine until today, which is 2021-06-09.
ind = local_model.loc[local_model['date'] == '2021-06-09',
'indexx'].values[0]
local_model = local_model.iloc[:ind+1]
local_model.dropna(subset=["people_vaccinated"],axis = 0,
inplace = True)
#Add a new column called "days", this column presents the
number of days.
```

```
d = len(local_model.index)
day = np.array([x for x in range(d)])
local_model['days'] = day
nc_min = local_model['new_cases'].min()
nc_max = local_model['new_cases'].max()
dis =
local_model.loc[:,('location','date','new_cases','total_cases',
'people_vaccinated','days')]
dis
```

- Result



	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
99506	World	2020-12-02	651735.0	64675043.0	0.00	0
99507	World	2020-12-03	694136.0	65369179.0	0.00	1
99508	World	2020-12-04	683135.0	66052314.0	0.00	2
99509	World	2020-12-05	642432.0	66694746.0	0.00	3
99510	World	2020-12-06	538995.0	67233741.0	0.00	4
...	...	...	...	...	...	...
99691	World	2021-06-05	398267.0	172988179.0	11.57	185
99692	World	2021-06-06	322691.0	173310870.0	11.71	186
99693	World	2021-06-07	321305.0	173632175.0	11.83	187
99694	World	2021-06-08	366820.0	173998995.0	12.00	188
99695	World	2021-06-09	419492.0	174418487.0	12.20	189

190 rows × 6 columns

Step 2: Normalize the data

Now we will normalize data. Normalizing data is the process that adjusts the range of all training sets to 0-1.

The purpose of normalizing data is for the ease of calculating, as well as increasing the effect of visual display.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Code

```
#Step 2: Normalize the data
local_cop =
local_model.loc[:,('new_cases','total_cases','people_vaccinated
','days')]
local_cop['people_vaccinated']=(local_model['people_vaccinated'
]-local_model['people_vaccinated'].min())/(local_model['people_
vaccinated'].max()-local_model['people_vaccinated'].min())
local_cop['new_cases']=(local_model['new_cases']-local_model['n
ew_cases'].min())/(local_model['new_cases'].max()-local_model['
new_cases'].min())
local_cop['total_cases']=(local_model['total_cases']-local_mode
l['total_cases'].min())/(local_model['total_cases'].max()-local
_model['total_cases'].min())
local_cop['days']=(local_model['days']-local_model['days'].min(
))/(local_model['days'].max()-local_model['days'].min())
local_cop
```

- Result

	new_cases	total_cases	people_vaccinated_per_hundred	days
99506	0.591684	0.000000	0.000000	0.000000
99507	0.659770	0.006325	0.000000	0.005291
99508	0.642105	0.012550	0.000000	0.010582
99509	0.576746	0.018404	0.000000	0.015873
99510	0.410651	0.023315	0.000000	0.021164
...	...	...	...	...
99691	0.184675	0.986967	0.948361	0.978836
99692	0.063318	0.989907	0.959836	0.984127
99693	0.061093	0.992835	0.969672	0.989418
99694	0.134179	0.996178	0.983607	0.994709
99695	0.218758	1.000000	1.000000	1.000000

190 rows × 4 columns

Step 3: Create training sets

After normalizing data, we will create the *training sets:*

- training sets 1 `(x1,y1)` are the number of new cases per day with respect to the number of `people getting vaccinated.`

- training sets 2 `(x2,y2)` are the number of total cases with respect to the number of `days`.

- Code

```
x1 = local_cop.drop(['total_cases','new_cases','days'],axis=1)
y1 = local_cop.new_cases
x2=local_cop.drop(['total_cases','new_cases','people_vaccinated
'],axis = 1)
y2 = local_cop.total_cases
```

Step 4: Create input

`input` is the number of days from today until 1 week later and 3 months later.

We will use `input` to predict the pandemic situation after 1 week and 3 months.

- Code

```
ow = np.array([d+7])
tm = np.array([d+30*3])
#Normalize the input
ow_norm =
(ow[0]-local_model['days'].min())/(local_model['days'].max()-lo
cal_model['days'].min())
tm_norm =
(tm[0]-local_model['days'].min())/(local_model['days'].max()-lo
cal_model['days'].min())
ow_norm = ow_norm.reshape(1,-1)
tm_norm = tm_norm.reshape(1,-1)
```

Step 5: Perform Math

The data is ready, now we will use the function `doMath` that we have built above to get the Regression functions.

- Code

```
[poly_reg_1, lin_reg_1, ow_re_1,tm_re_1, accuracy_n] =
BuildModel(x1,y1,ow_norm,tm_norm,check = 1)
[poly_reg_2, lin_reg_2, ow_re_2,tm_re_2, accuracy_t] =
BuildModel(x2,y2,ow_norm,tm_norm,check = 2)
```

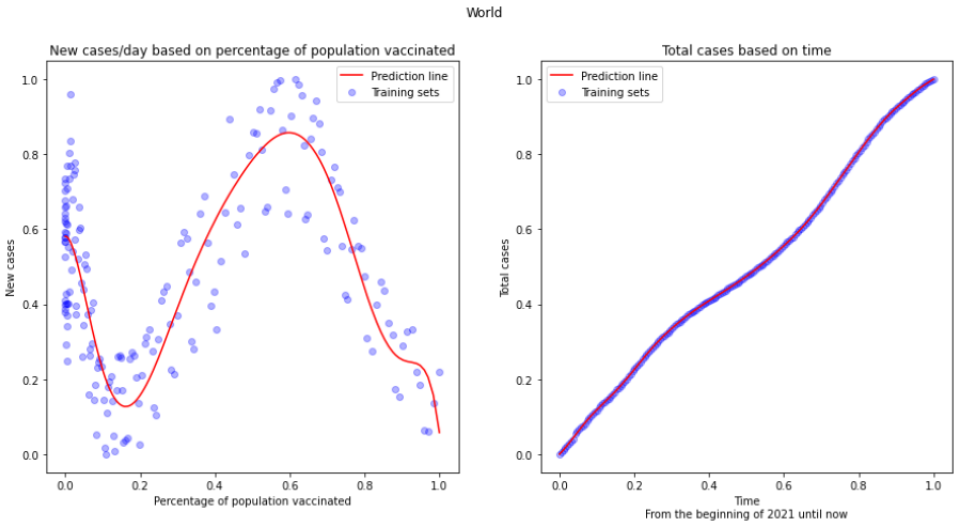
Step 6: Plot the graph

We got the Regression functions, now we will use these functions to plot graphs

- Code

```
# Step 6: In this stage, we will plot the graphs based on the
model that we have built
fig_model, (ax1, ax2) = plt.subplots(1, 2,figsize=(15,7))
fig_model.suptitle('World')
# ax1 is the graph that illustrates new cases/day based on
number of people vaccinated
ax1.scatter(x1,y1,color="blue",alpha=0.3)
ax1.plot(x1, lin_reg_1.predict(poly_reg_1.fit_transform(x1)),
color="red")
ax1.set_title("New cases/day based on number of people
vaccinated")
ax1.set_xlabel('People get vaccinated')
ax1.set_ylabel('New cases')
ax1.legend(['Prediction line','Training sets'])
# ax2 is the graph that illustrates total cases based on time.
ax2.scatter(x2,y2,color="blue",alpha=0.3)
ax2.plot(x2, lin_reg_2.predict(poly_reg_2.fit_transform(x2)),
color="red")
ax2.set_title("Total cases based on time")
ax2.set_xlabel('Time\nFrom the beginning of 2021 until now')
ax2.set_ylabel('Total cases')
ax2.legend(['Prediction line','Training sets'])
```

- `Result`



Step 7: Prediction

Finally, we predict the `total cases` of the World after one week and three months.

We also calculate the `accuracy` of the model.

- `Code`

```
# Step 7: Print on screen the number of cases that we have
predicted, as well as the accuracy of model
# First we have to bring the data back to their original state
(since we have normalized data previously)
ow_re_2 =
ow_re_2*(local_model['total_cases'].max()-local_model['total_ca
ses'].min())+local_model['total_cases'].min()
tm_re_2 =
tm_re_2*(local_model['total_cases'].max()-local_model['total_ca
ses'].min())+local_model['total_cases'].min()
# Finally, print the data
display(dis)
```

```
print("Maximum value of new cases per day: ",nc_max)
print("Minimum value of new cases per day: ", nc_min)
print("After one week, total cases of the World will
reach:",int(ow_re_2),'cases')
print("After three months, total cases of the World will
reach:",int(tm_re_2),'cases')
print("Accuracy of this prediction model:",accuracy_t*100,"%")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
99507	World	2020-12-02	651735.0	64675043.0	0.00	0
99508	World	2020-12-03	694136.0	65369179.0	0.00	1
99509	World	2020-12-04	683135.0	66052314.0	0.00	2
99510	World	2020-12-05	642432.0	66694746.0	0.00	3
99511	World	2020-12-06	538995.0	67233741.0	0.00	4
...	...	...	...	...	...	...
99692	World	2021-06-05	398267.0	172988179.0	11.57	185
99693	World	2021-06-06	322691.0	173310870.0	11.71	186
99694	World	2021-06-07	321305.0	173632175.0	11.84	187
99695	World	2021-06-08	366820.0	173998995.0	12.00	188
99696	World	2021-06-09	419492.0	174418487.0	12.20	189

190 rows × 6 columns

```
Maximum value of new cases per day:  906017.0
Minimum value of new cases per day:  283259.0
After one week, total cases of the World will reach: 196154290 cases
After three months, total cases of the World will reach: 225357379 cases
Accuracy of this prediction model: 99.99819677133316 %
```

Step 8: Analyze the pandemic situation of the World

At the beginning, there was a downward trend in the number of new cases, from 532362 to 360481, as the percentage of people who vaccinated increased by 2.4%. Then, the number of new cases reached a peak of 633189, as opposed to the increment of the proportion of the vaccinated population which was 7.28%. After that, the number of new cases experienced a dramatic decrease, with 12% of people vaccinated, since the effort of vaccinated policy was applied strictly throughout the world.

However, the world still indicated an upward trend of total cases, due to unpredictable variants of coronavirus in some countries like the United Kingdom (Alpha), Brazil (Gamma), India (Delta). These variants seem to

spread more easily and quickly than other variants, which lead to more cases of COVID-19. An increase in the number of cases will put more strain on healthcare resources, lead to more hospitalizations, and potentially more deaths.

5.2.2 Create Prediction function

We will combine all the steps above into one function that can be reused several times to predict the pandemic situation around the world.

This function is called `Prediction`

- Code

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
def Prediction(place):

    #Step 1: Get data from open source, clean data by taking
out useful features
    df =
pd.read_csv("https://raw.githubusercontent.com/owid/covid-19-d
ata/master/public/data/owid-covid-data.csv")
    df = df[df['location'] == place]
    df =
df.loc[:,('location','date','new_cases','total_cases','people_
vaccinated')]
    local_model = df.copy()
    n = len(local_model.index)
    inde = np.array([x for x in range(n)])
    local_model['indexx'] = inde
    #We will consider data from the day we started using the
vaccine until today, which is 2021-06-09.
    ind = local_model.loc[local_model['date'] == '2021-06-09',
'indexx'].values[0]
```

```
    local_model = local_model.iloc[:ind+1]
    local_model.dropna(subset=["people_vaccinated"],axis = 0,
inplace = True)
    #Add a new column called "days", this column presents the
number of days.
    d = len(local_model.index)
    day = np.array([x for x in range(d)])
    local_model['days'] = day
    dis =
local_model.loc[:,('location','date','new_cases','total_cases'
,'people_vaccinated','days')]
    #Step 2: Normalize the data
    local_cop =
local_model.loc[:,('new_cases','total_cases','people_vaccinate
d','days')]

local_cop['people_vaccinated']=(local_model['people_vaccinated
']-local_model['people_vaccinated'].min())/(local_model['peopl
e_vaccinated'].max()-local_model['people_vaccinated'].min())

local_cop['new_cases']=(local_model['new_cases']-local_model['
new_cases'].min())/(local_model['new_cases'].max()-local_model
['new_cases'].min())

local_cop['total_cases']=(local_model['total_cases']-local_mod
el['total_cases'].min())/(local_model['total_cases'].max()-loc
al_model['total_cases'].min())

local_cop['days']=(local_model['days']-local_model['days'].min
())/(local_model['days'].max()-local_model['days'].min())


    #Step 3: Create training sets
    #x1 is the number of people get vaccinated
    #y1 is the number of new cases everyday
    #x2 is the number of days
    #y2 is the number of total cases
    x1 =
```

```
local_cop.drop(['total_cases','new_cases','days'],axis = 1)
    y1 = local_cop.new_cases
    x2 =
local_cop.drop(['total_cases','new_cases','people_vaccinated']
,axis = 1)
    y2 = local_cop.total_cases


    # Step 4: Create "input feature"
    # "input feature" is the time period that we want to
predict the total cases
    # In this case, we will predict the total cases after 1
week and 3 months
    # So the "input" will be 7 days (ow) and 30*3 = 90 days
(tm)
    ow = np.array([d+7])
    tm = np.array([d+30*3])
    #Normalize the input
    ow_norm =
(ow[0]-local_model['days'].min())/(local_model['days'].max()-l
ocal_model['days'].min())
    tm_norm =
(tm[0]-local_model['days'].min())/(local_model['days'].max()-l
ocal_model['days'].min())
    ow_norm = ow_norm.reshape(1,-1)
    tm_norm = tm_norm.reshape(1,-1)

    # Step 5: Build model
    # In this stage, we will build a function called
"BuildModel", it will build a prediction model based on Linear
Regression and Poly Regression
    # This function takes 5 parameters, which are: the
training sets, the "input feature" after normalized, and a
parameter "check"
    # "check" is just an identified parameter
    # if check == 1, the function will build a model to
predict new cases/day based on number of people vaccinated
```

```
    # if check == 2, the function will build a model to
predict total cases based on days.
    # This BuildModel function will also return 5 values:
    #                   - Poly Regression function
    #                   - Linear Regression function
    #                   - Total cases predicted after 1 week
    #                   - Total cases predicted after 3 months
    #                   - Accuracy of the model
    [poly_reg_1, lin_reg_1, ow_re_1,tm_re_1, accuracy_n] =
doMath(x1,y1,ow_norm,tm_norm,check = 1)
    [poly_reg_2, lin_reg_2, ow_re_2,tm_re_2, accuracy_t] =
doMath(x2,y2,ow_norm,tm_norm,check = 2)


    # Step 6: In this stage, we will plot the graphs based on
the model that we have built
    fig_model, (ax1, ax2) = plt.subplots(1, 2,figsize=(15,7))
    fig_model.suptitle(place)

    # ax1 is the graph that illustrates new cases/day based on
number of people vaccinated
    ax1.scatter(x1,y1,color="blue",alpha=0.3)
    ax1.plot(x1,
lin_reg_1.predict(poly_reg_1.fit_transform(x1)), color="red")
    ax1.set_title("New cases/day based on number of people
vaccinated")
    ax1.set_xlabel('People get vaccinated')
    ax1.set_ylabel('New cases')
    ax1.legend(['Prediction line','Training sets'])

    # ax2 is the graph that illustrates total cases based on
time.
    ax2.scatter(x2,y2,color="blue",alpha=0.3)
    ax2.plot(x2,
lin_reg_2.predict(poly_reg_2.fit_transform(x2)), color="red")
    ax2.set_title("Total cases based on time")
    ax2.set_xlabel('Time\nFrom the beginning of 2021 until
now')
    ax2.set_ylabel('Total cases')
    ax2.legend(['Prediction line','Training sets'])
```

```
    # Step 7: Print on screen the number of cases that we have
predicted, as well as the accuracy of model
    # First we have to bring the data back to their original
state (since we have normalized data previously)
    ow_re_2 =
ow_re_2*(local_model['total_cases'].max()-local_model['total_c
ases'].min())+local_model['total_cases'].min()
    tm_re_2 =
tm_re_2*(local_model['total_cases'].max()-local_model['total_c
ases'].min())+local_model['total_cases'].min()
    # Finally, print the data
    display(dis)
    print("After one week, total cases of",place,"will
reach:",int(ow_re_2),'cases')
    print("After three months, total cases of",place,"will
reach:",int(tm_re_2),'cases')
    print("Accuracy of this prediction
model:",accuracy_t*100,"%")
    #return fig_model
```

5.3 Other countries

Now we will use the Prediction function that we have built above to predict the epidemic situation of some countries.

5.3.1 USA, India, Brazil

a/USA

- Code

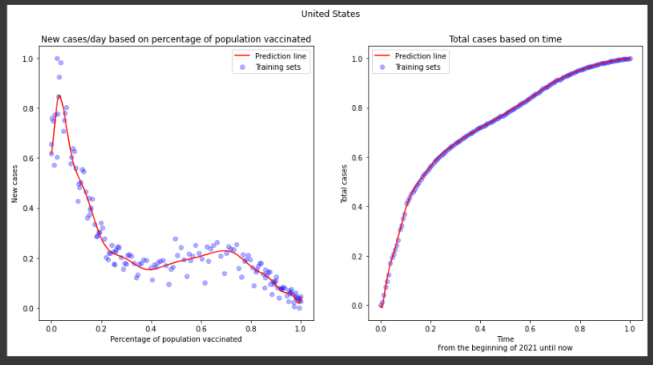
```
usa = Prediction("United States")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
96175	United States	2020-12-20	188015.0	17955653.0	0.17	0
96176	United States	2020-12-21	198703.0	18154356.0	0.18	1
96178	United States	2020-12-23	229624.0	18582087.0	0.30	2
96181	United States	2020-12-26	226416.0	19100538.0	0.58	3
96183	United States	2020-12-28	174152.0	19430535.0	0.64	4
...	...	...	...	...	...	...
96342	United States	2021-06-05	13906.0	33357285.0	50.91	151
96343	United States	2021-06-06	5395.0	33362680.0	51.08	152
96344	United States	2021-06-07	15496.0	33378176.0	51.22	153
96345	United States	2021-06-08	13013.0	33391189.0	51.35	154
96346	United States	2021-06-09	18647.0	33409836.0	51.45	155

156 rows × 6 columns

```
Maximum value of new cases per day:  300462.0
Minimum value of new cases per day:  5395.0
After one week, total cases of United States will reach: 34812918 cases
After three months, total cases of United States will reach: 42090080 cases
Accuracy of this prediction model: 99.94738597419904 %
```



After reaching a peak of 243352 new cases, it experienced a drastic drop to 76782 new cases, with 14,4% of people vaccinated. Then, it remained

constant at approximately 72023 new cases, when the percentage of population vaccinated increased by 23%, followed by a slight drop to 24432 with 50.63% vaccinated.

As the vaccinations became widespread and more and more effective, total cases in the USA had a gradual increase until now, despite a surge from 17955635 to 27790133 in the first month.

b/India

- Code

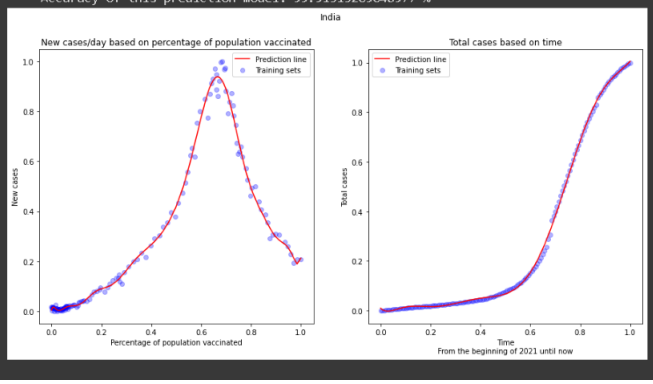
```
india = Prediction("India")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
42170	India	2021-01-15	15158.0	10542841.0	0.00	0
42171	India	2021-01-16	15144.0	10557985.0	0.01	1
42172	India	2021-01-17	13788.0	10571773.0	0.02	2
42173	India	2021-01-18	10050.0	10581823.0	0.03	3
42174	India	2021-01-19	13816.0	10595639.0	0.05	4
...	...	...	...	...	...	...
42311	India	2021-06-05	114460.0	28809339.0	13.11	136
42312	India	2021-06-06	100636.0	28909975.0	13.28	137
42313	India	2021-06-07	86498.0	28996473.0	13.44	138
42314	India	2021-06-08	92596.0	29089069.0	13.65	139
42315	India	2021-06-09	93463.0	29182532.0	13.86	140

141 rows × 6 columns

```
Maximum value of new cases per day:  414188.0
Minimum value of new cases per day:  8635.0
After one week, total cases of India will reach: 30775092 cases
After three months, total cases of India will reach: 38903814 cases
Accuracy of this prediction model: 99.91515209048977 %
```



A significant increase was observed in the number of new cases, which

was from 15158 to 40397 cases per day, with 10.1% people vaccinated. However, after reaching that peak, the trend experienced a rapid drop to 110023 cases per day as the percentage of vaccinated increased up to 13.5%.

It is noticeable that in the first two months, the number of total cases in India saw a slight increase, hovering at 10599836 cases. Then, it leapt to 29182532 total cases, which nearly tripled the initial record, due to the appearance of the Delta variant which spread more easily and quickly.

c/Brazil

- Code

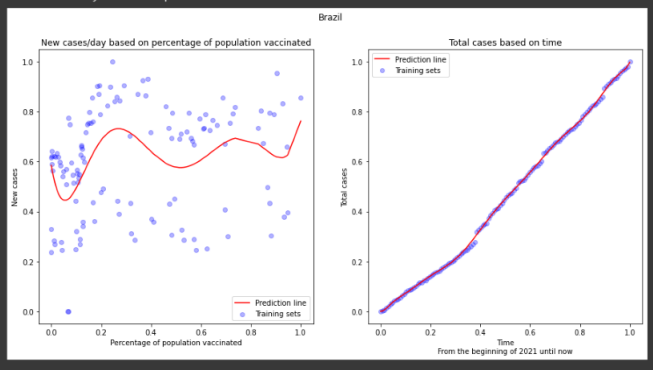
```
brazil = Prediction("Brazil")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
13183	Brazil	2021-01-16	61567.0	8455059.0	0.00	0
13184	Brazil	2021-01-17	33040.0	8488099.0	0.00	1
13185	Brazil	2021-01-18	23671.0	8511770.0	0.00	2
13186	Brazil	2021-01-19	62094.0	8573864.0	0.01	3
13187	Brazil	2021-01-20	64385.0	8638249.0	0.01	4
...	...	...	...	...	...	...
13321	Brazil	2021-06-03	83391.0	16803472.0	22.41	123
13322	Brazil	2021-06-04	37936.0	16841408.0	22.59	124
13323	Brazil	2021-06-05	66017.0	16907425.0	22.85	125
13324	Brazil	2021-06-06	39637.0	16947062.0	22.96	126
13327	Brazil	2021-06-09	85748.0	17122877.0	24.21	127

128 rows × 6 columns

```
Maximum value of new cases per day:  100158.0
Minimum value of new cases per day:  0.0
After one week, total cases of Brazil will reach: 22656820 cases
After three months, total cases of Brazil will reach: 23275536 cases
Accuracy of this prediction model: 99.92413271073445 %
```



Overall, the number of new cases fluctuated even though the percentage of people vaccinated increased.

At the beginning, after hitting the lowest point with 46740 new cases per day, it hit a peak of 74255 cases. Then, the line gained remarkable recovery at the highest point after a few fluctuations.

The appearance of the Gamma variant made the number of total cases still increased linearly, despite the endless effort of the government in preventing the quick spread of new variants.

5.3.2 Korea, Japan, Vietnam

a/Korea

- Code

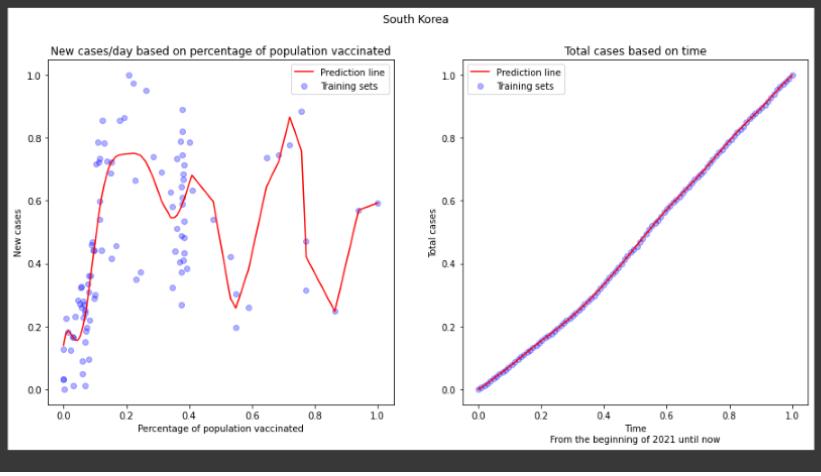
```
korea = Prediction("South Korea")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
84992	South Korea	2021-02-26	399.0	89321.0	0.04	0
84993	South Korea	2021-02-27	355.0	89676.0	0.04	1
84994	South Korea	2021-02-28	355.0	90031.0	0.04	2
84995	South Korea	2021-03-01	341.0	90372.0	0.05	3
84996	South Korea	2021-03-02	444.0	90816.0	0.18	4
...	...	...	...	...	...	...
85091	South Korea	2021-06-05	556.0	144152.0	14.82	99
85092	South Korea	2021-06-06	485.0	144637.0	14.83	100
85093	South Korea	2021-06-07	454.0	145091.0	16.58	101
85094	South Korea	2021-06-08	601.0	145692.0	18.03	102
85095	South Korea	2021-06-09	611.0	146303.0	19.19	103

104 rows × 6 columns

```
Maximum value of new cases per day:  797.0
Minimum value of new cases per day:  341.0
After one week, total cases of South Korea will reach: 191833 cases
After three months, total cases of South Korea will reach: 198307 cases
Accuracy of this prediction model: 99.99748678815476 %
```



According to the data that we have collected, the pandemic's situation in South Korea has appeared to be in a tough spot since the beginning of 2021. The proportion of total cases has been increasing rapidly. Despite the growth of the number of people who got vaccinated, the increase rate of total cases does not seem to decline.

b/Japan

- Code

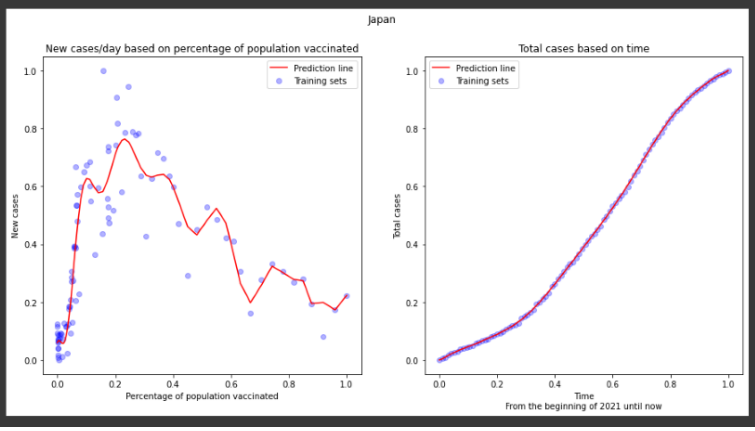
```
japan = Prediction("Japan")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
46937	Japan	2021-02-17	1461.0	419896.0	0.00	0
46938	Japan	2021-02-18	1525.0	421421.0	0.00	1
46939	Japan	2021-02-19	1297.0	422718.0	0.00	2
46942	Japan	2021-02-22	737.0	425725.0	0.01	3
46944	Japan	2021-02-24	904.0	427732.0	0.01	4
...	...	...	...	...	...	...
47045	Japan	2021-06-05	2663.0	760953.0	12.36	91
47046	Japan	2021-06-06	2027.0	762980.0	12.79	92
47047	Japan	2021-06-07	1205.0	764185.0	13.37	93
47048	Japan	2021-06-08	1883.0	766068.0	13.97	94
47049	Japan	2021-06-09	2245.0	768313.0	14.55	95

96 rows × 6 columns

```
Maximum value of new cases per day:  7914.0
Minimum value of new cases per day:  621.0
After one week, total cases of Japan will reach: 1106450 cases
After three months, total cases of Japan will reach: 1153635 cases
Accuracy of this prediction model: 99.98953230315372 %
```



In contrast to Korea, the pandemic's situation in Japan during the early days of 2021 seemed to be in a stable state where the total cases were increasing steadily. However, the situation became worse in mid-March when the total case per day began to increase considerably. During this time, the number of people who got vaccinated also had a major rise in proportion and the vaccine appeared to have a proper development.That led to a drop in the number of new cases per day from the peak of 7914 new cases per day, putting the pandemic's situation in Japan under control.

c/Vietnam

- Code

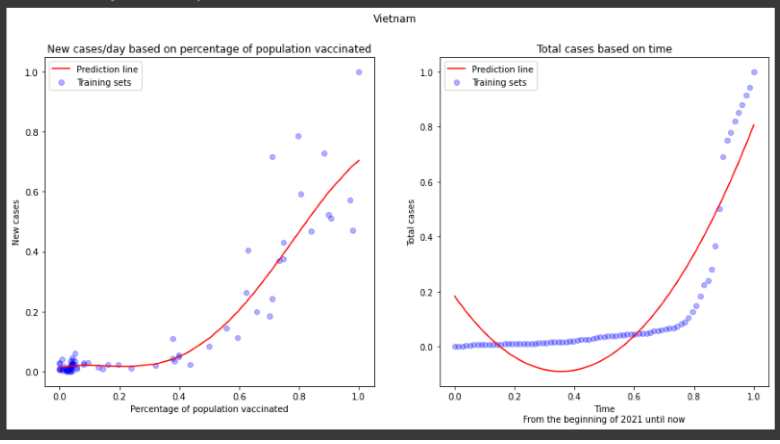
```
vietnam = Prediction("Vietnam")
```

- Result

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
98962	Vietnam	2021-03-07	11.0	2512.0	0.00	0
98963	Vietnam	2021-03-08	12.0	2524.0	0.00	1
98964	Vietnam	2021-03-09	2.0	2526.0	0.00	2
98965	Vietnam	2021-03-10	3.0	2529.0	0.00	3
98966	Vietnam	2021-03-11	4.0	2533.0	0.00	4
...	...	...	...	...	...	...
99052	Vietnam	2021-06-05	216.0	8580.0	1.24	74
99053	Vietnam	2021-06-06	211.0	8791.0	1.25	75
99054	Vietnam	2021-06-07	236.0	9027.0	1.34	76
99055	Vietnam	2021-06-08	195.0	9222.0	1.35	77
99056	Vietnam	2021-06-09	413.0	9635.0	1.38	78

79 rows × 6 columns

```
Maximum value of new cases per day:  413.0
Minimum value of new cases per day:  0.0
After one week, total cases of Vietnam will reach: 10450 cases
After three months, total cases of Vietnam will reach: 11335 cases
Accuracy of this prediction model: 70.94820112530917 %
```



In Vietnam, the pandemic's situation was under control in the early days of 2021 with approximately under 20 new cases per day. However, that was no longer the case in May. As we can see, the accuracy of this prediction is only 70%, much lower than the other region. This is because on

the National Day at the end of April, lots of Vietnamese people decided to hang out and travel around, which led to the 4th pandemic outbreak in Vietnam. The sudden drastic increase in total cases causes the machine to hardly find a regression that best suits the pandemic situation of Vietnam. Moreover, the appearance of the new COVID-19 Delta variant also led to a significant increase in total cases. The number of new people getting infected per day went up to over 200 on average. Similar to Korea, the vaccine has not yet appeared to take effect in Vietnam, which leaves the condition of pandemic uncertain.

6. Model Analysis

6.1 Ability

To sum up, our model is built based on *Linear Regression* and *Poly Regression*, which takes the form:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \text{ (for } \textit{Linear Regression})$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon. \text{ (for } \textit{Poly Regression})$$

where

- $\mathbf{y}$ is a vector *dependent variables*, also known as *predicted variables*.
- $\mathbf{X}$ is a matrix of *independent variables*, also known as *input variables*.
- $\boldsymbol{\beta}$ is the vector of *regression coefficients*, specifically the *partial derivatives of* $\mathbf{y}$ *with respect to* $\mathbf{X}$.
- $\boldsymbol{\varepsilon}$ is a vector called *error term*. This variable captures all other factors which influence the dependent variable y other than the regressors $\mathbf{X}$.

Through the graph that illustrates *number of new cases per day* based of *percentage of people vaccinated*, we can know the pandemic situation before and after vaccination, the trend (increasing or decreasing) in the spread of the pandemic at the current, as well as figure out the reason why the pandemic keep spreading although more and more people are vaccinated.

Through the graph that illustrates the *number of total cases* based on *time*, we can predict the pandemic situation a week and three months later, thereby knowing the growth rate and the peak of the pandemic.

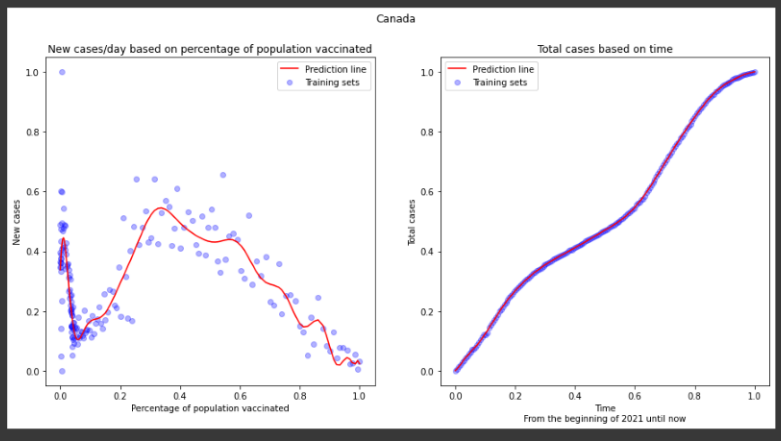
6.2 Flexibility

Overall, this model can give predictions to almost every country in the world. For example, here is a prediction about the pandemic situation in Canada:

	location	date	new_cases	total_cases	people_vaccinated_per_hundred	days
16668	Canada	2020-12-14	8377.0	472820.0	0.00	0
16669	Canada	2020-12-15	6244.0	479064.0	0.00	1
16670	Canada	2020-12-16	6512.0	485576.0	0.01	2
16671	Canada	2020-12-17	7006.0	492582.0	0.02	3
16672	Canada	2020-12-18	6700.0	499282.0	0.03	4
...	...	...	...	...	...	...
16841	Canada	2021-06-05	1349.0	1398719.0	61.09	173
16842	Canada	2021-06-06	1374.0	1400093.0	61.61	174
16843	Canada	2021-06-07	1819.0	1401912.0	62.28	175
16844	Canada	2021-06-08	1071.0	1402983.0	62.72	176
16845	Canada	2021-06-09	1460.0	1404443.0	63.15	177

178 rows × 6 columns

```
Maximum value of new cases per day:  16141.0
Minimum value of new cases per day:  991.0
After one week, total cases of Canada will reach: 1691747 cases
After three months, total cases of Canada will reach: 1892216 cases
Accuracy of this prediction model: 99.99577791787893 %
```

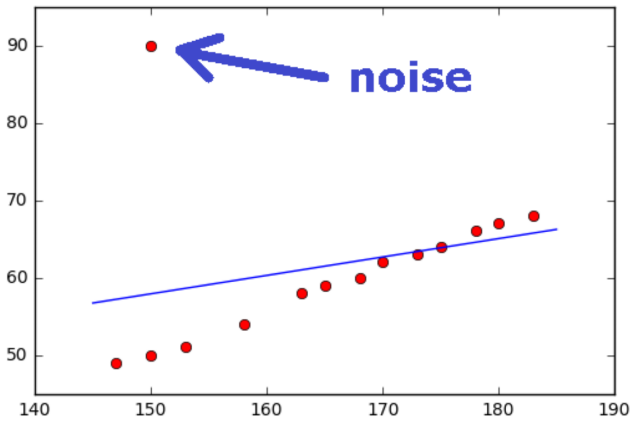


However, this model cannot be used for the prediction of some countries such as China. The reason is that our model operates by collecting data of 4 features: *total cases*, *new cases*, *percentage of population vaccinated* and *time*, while China does not publish data about their *percentage of population vaccinated*. Therefore, our model cannot operate due to lack of data. That is one drawback of our model.

Linear regression, as well as *Poly regression* also has some advantages and limitations as below:

Advantages	Disadvantages
Linear Regressions - Simple to implement - Gives information about the relevance of features. - Performs well when the dataset is linearly separable. - Works well irrespective of the dataset size (thanks to the *normalization* process) - Overfitting can be reduced by using some dimensionality reduction techniques, such as *regularization.*	**Linear Regressions** - Assumes there is a linear relationship (a straight line) between dependent and independent variables - Oversimplifies the problems since in the real world, data rarely has a linear relationship. between dependent and independent variables. - Really prone to underfitting, specifically it is very sensitive to *noise*.
Polynomial Regressions Provides a wide range of functions that best fit the relationship between the dependent and independent variable.	**Polynomial Regressions** Very sensitive to noise and easy to get overfitted.

As we can see, both regression have the same limitation which is the sensitivity to *noise. Noise* is the values that deviate from the other data points of the distribution:



As we can see, noises can damage the performance of the model drastically and can often lead to low accuracy.

Additionally, we should know how to flexibly switch between models in order to yield the best results. For example, *linear regression* model is a good choice to predict the *total cases* of one specific country, because the number of *total cases* always increases (which means the partial derivative at every point of the function is always greater than zero), so we can use *linear regression* in this case. On the other hand, *polynomial regression* is a better choice to model the relationship between *new cases per day* and *percentage of population vaccinated*, since the dependent value, which is *new cases per day*, does not always increase or decrease. There's a period of time when this value decreases, and vice versa, increases if the pandemic situation is uncontrollable. Moreover, knowing to switch models while making predictions for one week later or three months later is also necessary, because as we have said in *3.1.1. Theory*, *polynomial regression* is suitable to make predictions for a short period ahead of us such as one week later. However, to predict the pandemic situation three months later, then *linear regression* is a better choice since it will reduce the possibility of overfitting.

Therefore, both *linear regression* and *polynomial regression* model can be used in reality, if the distribution of the data points is simple enough for the regressions to find the best curve that fits the distribution. Additionally, knowing to flexibly switch between models is also very important to yield the best results closest to reality.

7. Conclusion

To conclude, by taking this great assignment, our group has the chance to study new knowledge such as Model prediction and know more about the impact of COVID-19.

This study has undergone multiple steps to understand how Python could be utilized as a tool for analyzing and predicting the pandemic's tendency.

The study begins by analyzing the past distribution of the COVID-19 cases worldwide, and the effect vaccination has on the population. The finding is that in 2020, cases skyrocketed exponentially, representing a clear and pronounced positive kurtosis (leptokurtic) distribution. However, the effect of vaccination was present as it was one of the key factors driving the pandemic's tendency in 2021.

The study then attempted to predict the behaviors of the pandemic both worldwide and in Vietnam. Internationally, as vaccination continues to show its effect, cases are expected to reach a saturation level at the end of 2021. However, with only about 10% of people being fully vaccinated, cases are still expected to increase substantially in some regions while stabilizing in others. The model successfully showed this hypothesis.

Besides, in Vietnam, the picture is much more complicated. The pattern of the pandemic in Vietnam is different from some of the Western countries, which experienced dramatic damages of the pandemic in early 2020 before managing to stabilize it with vaccines and lockdowns. In Vietnam, the worst situation of the pandemic has just begun as cases climbed unprecedentedly by the new variant compared with the extremely low level of vaccination.

Our model predicts that the cases in Vietnam may continue to increase exponentially, not following a mesokurtosis distribution but a more positive kurtosis. The increase would only be halted when a sufficient portion of the population is vaccinated, which would not happen earlier than the end of 2021. In this study, we utilize Linear regression and Polynomial regression and determine which type of regression is the best for a specific period of time. Based on the number of data plots available, our study successfully finds the most appropriate model for each prediction. Also, our study creates mathematical functions to satisfy the initial goals.

8. References

[1] Our World In Data, *Covid-19 Data*,
https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-data.csv

[2] Wikipedia, *Linear Regression*,
https://en.wikipedia.org/wiki/Linear_regression

[3] Wikipedia, *Polynomial Regression*,
https://en.wikipedia.org/wiki/Polynomial_regression

[4] Wikipedia, *Normalization (statistics)*,
https://en.wikipedia.org/wiki/Normalization_(statistics)

[5] Math is fun, *Normal distribution*,
https://www.mathsisfun.com/data/standard-normal-distribution.html

[6] Christoph Molnar, *Linear Regression*,
https://christophm.github.io/interpretable-ml-book/limo.html

[7] Naman Singh, *Advantages and Disadvantages of Linear Regression*,
https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/

[8] Palakur Eshwitha Sai, *Advantages and Disadvantages of different Regression models*,
https://www.geeksforgeeks.org/advantages-and-disadvantages-of-different-regression-models/

[9] Javatpoint, *Machine Learning Polynomial Regression*,
https://www.javatpoint.com/machine-learning-polynomial-regression

[10] codebasics, *Machine Learning Tutorial Python - 7: Training and Testing Data*,
https://www.youtube.com/watch?v=fwY9Qv96DJY