HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
OFFICE FOR INTERNATIONAL STUDY PROGRAM
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT REPORT

DISCRETE STRUCTURES FOR COMPUTING

Semester: 202 - Group name: 404 NOT FOUND

Under the guidance of: PROF.Trần Tuấn Anh

Submitted by: Nguyễn Nho Gia Phúc – 2052214
Nguyễn Văn Quốc Chương – 1950004
Nguyễn Trọng Chuẩn – 2052046
Vũ Đồng Tuệ Quyên – 2052234
Thái Tài – 2052246

Ho Chi Minh City, 7/2021

Contents

1 Introduction **5**

2 Tools and environments **9**
2.1 Python . . . 9
2.2 Google Colaboratory . . . 9
2.3 Pandas . . . 10
2.4 Matplotlib . . . 10
2.5 Seaborn . . . 10
2.6 Numpy . . . 11
2.7 SkLearn . . . 11
2.8 Pmdarima . . . 11

3 Overview of Machine Learning **12**
3.1 What is Machine Learning? . . . 12
3.1.1 Comparison between Machine Learning, AI and Deep Learning . . . 12
3.2 Supervised learning and Unsupervised learning . . . 13
3.2.1 Supervised learning . . . 13
3.2.2 Unsupervised learning . . . 14
3.3 Reinforcement learning . . . 15
3.4 Ensemble learning . . . 17
3.5 Deep learning . . . 18

4 Data collection **21**
4.1 How many data sets are enough? . . . 21
4.2 Importing data into Google Colab . . . 21
4.3 Data pre-processing . . . 21
4.4 Data description . . . 22
4.4.1 Cases dataset: Source . . . 22
4.4.2 Vaccinations by age group: Source . . . 23
4.4.3 Vaccinations by manufacturers: Source . . . 24
4.4.4 Dedicated data for Vietnam: Source . . . 24
4.4.5 The first dataset for SIR model: Source . . . 24
4.4.6 The second dataset for SIR model: Source . . . 25

5 Data Analysis **26**
5.1 Vaccination in history and in COVID-19 pandemic . . . 26
5.2 Vaccine Analysis . . . 30
5.2.1 COVID-19 vaccines prevent the spread of virus . . . 30
5.2.2 Vaccination affects the deaths . . . 36
5.3 Other Analysis . . . 39
5.3.1 Delta variant of Coronavirus . . . 39
5.3.2 Covid situation in Viet Nam . . . 39

5.3.3 Correlation Matrix . . . 41
5.4 Theory of Distribution . . . 42
5.4.1 What is data distribution? . . . 42
5.4.2 What data distribution can do? . . . 43
5.4.3 Normal Distribution . . . 44
5.4.4 Bimodal Distribution . . . 47
5.4.5 Poisson Distribution . . . 49
5.4.6 Gamma Distribution . . . 50
5.4.7 Exponential Distribution . . . 52
5.5 Distribution Analysis . . . 53
5.5.1 Distribution of new infections . . . 53
5.5.2 Distribution of vaccinated age . . . 55
5.5.3 Distribution of new infections affected by Delta variant . . . 56

6 Prediction Models 58
6.1 Linear regression algorithm . . . 58
6.1.1 Regression summary . . . 58
6.1.2 Linear regression and its algorithm . . . 58
6.2 Linear Regression Model . . . 62
6.2.1 Vietnam . . . 62
6.2.2 Korea . . . 64
6.2.3 Japan . . . 64
6.2.4 Brazil . . . 65
6.2.5 India . . . 65
6.2.6 USA . . . 66
6.3 Decision Tree Model . . . 66
6.4 Support Vector Machine . . . 67
6.4.1 What is Support Vector Machine (SVM)? . . . 67
6.4.2 Soft-margin SVM . . . 70
6.4.3 Constrained-Optimization Problem (COP) . . . 70
6.4.4 Kernel SVM . . . 73
6.4.5 Pros and cons of SVM algorithm . . . 74
6.4.6 Application of SVM . . . 75
6.5 Support Vector Machine Model . . . 76
6.6 Time Series . . . 78
6.6.1 Time series . . . 78
6.6.2 Decomposing Time Series . . . 82
6.7 ARIMA . . . 84
6.7.1 Auto-Regressive Model . . . 84
6.7.2 Moving Average Model . . . 85
6.7.3 Non-seasonal ARIMA model . . . 86
6.7.4 Estimation and order selection . . . 87
6.7.5 Forecasting . . . 88
6.7.6 Seasonal ARIMA models . . . 88
6.8 ARIMA MODEL . . . 90
6.8.1 Installing and importing necessary tools . . . 90

6.8.2 Data loading and pre-processing . . . 90
6.8.3 The USA . . . 91
6.9 Holt-Winters . . . 103
6.10 Holt-Winters MODEL . . . 105
6.11 Time Series Results (ARIMA + Holt-Winters) . . . 109
6.11.1 Korea . . . 109
6.11.2 Brazil . . . 110
6.11.3 India . . . 111
6.11.4 Japan . . . 112
6.11.5 Vietnam . . . 113
6.12 SIR model theory . . . 114
6.13 Some other compartmental models of SIR . . . 116
6.13.1 SIS Model . . . 116
6.13.2 SIRD Model . . . 116
6.13.3 MSIR Model . . . 116
6.13.4 SICR Model . . . 117
6.13.5 SEIR Model . . . 117
6.13.6 SEIS Model: . . . 118
6.13.7 MSEIR Model . . . 118
6.13.8 MSEIRS Model . . . 119
6.14 SIR MODEL . . . 119
6.14.1 Global . . . 119
6.14.2 Vietnam . . . 123
6.14.3 Korea . . . 126
6.14.4 Japan . . . 127
6.14.5 USA . . . 128
6.14.6 India . . . 129
6.14.7 Brazil . . . 130

7 Model Evaluation 131
7.1 Testing . . . 131
7.2 Model Analysis . . . 131
7.2.1 Time Series models . . . 131
7.2.2 Linear Regression and Decision Tree . . . 132
7.2.3 SIR Model . . . 132
7.2.4 Support-vector Machine . . . 133
7.3 Influencing factors . . . 134
7.4 Is it possible to use a cross-country model? . . . 134

8 Conclusion 135

9 Our code for analysis and model prediction 136
9.1 Time-based Analysis . . . 136
9.2 Location-based Analysis . . . 136
9.3 Time Series Model . . . 136
9.4 Linear Regression, Support Vector Machine, Decision Tree Models . . . 136
9.5 SIR Model . . . 136

10 Some demo codes for other machine learning algorithms **136**
10.1 Linear Regression demo code . . . 136
10.2 Logistic Regression demo code . . . 136
10.3 K-means demo code . . . 136
10.4 Neural Network demo code . . . 136
10.5 Principal Component Analysis demo code . . . 136

11 APPENDIX **137**

A Data preprocessing **137**
A.1 What is data preprocessing? . . . 137
A.2 Z-score and P-value . . . 137
A.2.1 What is a Z-Score? . . . 137
A.2.2 What is a P-value? . . . 138
A.3 Outlier . . . 138
A.3.1 What is the Outlier? . . . 138
A.3.2 How to determine the outliers? . . . 139
A.4 Data Cleaning/Cleansing: . . . 140
A.5 Data Integration . . . 140
A.6 Data Transformation . . . 141
A.7 How to do the data transforming? . . . 142
A.8 Data Reduction . . . 144

B Other basic algorithms in Machine Learning **145**
B.1 Supervised learning . . . 145
B.1.1 Logistic regression(Classification problem) . . . 145
B.1.1.a What is Logistic regression? . . . 145
B.1.1.b Type of Logistic regression . . . 145
B.1.1.c Overfitting and Underfitting . . . 147
B.1.1.d Bias and Variance . . . 148
B.2 Unsupervised Learning . . . 150
B.2.1 K-Means Clustering in Machine Learning . . . 150
B.2.2 How the K-means algorithm works? . . . 150
B.2.3 Choosing the Appropriate Number of Cluster . . . 151
B.2.4 Advantages and Disadvantages of K-means . . . 152
B.2.5 Applications of K-means . . . 153

1 Introduction

Figure 1: COVID-19 (Image source: gov.vn)

Coronavirus disease 2019 (COVID-19) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). It was first identified in Wuhan, China in December 2019, and has since spread globally, and become an ongoing pandemic.

It emerges as one of the deadliest diseases mankind has ever faced. The virus has taken the lives of nearly four million people worldwide. Consequently, productions are hindered and businesses are forced to shut down.

Figure 2: People now have meetings online (image source: BBC).

The virus was confirmed to have spread to Vietnam in January 2020, when two Chinese people in Ho Chi Minh City tested positive. The first death in Vietnam was on July 31st, 2020. Vietnam is known to the world to have well-organized pandemic control programs, including good public health system, a decisive central government, and a proactive containment strategy.

For the past few months, vaccines are being rolled out to the public in well-developed countries and also distributed to every other nation in the world. We are all hoping for herd immunity amongst countries, and, as a result, a victory against the plague. Vietnam is also working on its own vaccine, Nano Covax, and the third testing phase is expected to take place this June.

Figure 3: A person being vaccinated (image source: nature.com).

COVID-19 is no doubt one of the worst things that happened to us in this decade. But then again, it opens up the opportunity for us ourselves to study from, and take better countermeasures. The data from the plague is a gold mine for the said purposes.

The figures collect from SARS-CoV-2 is a valuable source for data enthusiasts. While we can take a look back at how lethal it has been, and observe the trend in the number of cases, deaths, and recoveries, we can also use the data to predict future outcomes of this pandemic. Furthermore, with public vaccination being carried out by some governments, we can assess whether it has a close relationship with the number of new cases and deaths everyday.

As part of the Discrete Structures for Computing assignment, our team carried out this research in order to evaluate the impact of the pandemic on continents, pivotal countries, and the effectiveness of vaccinations. At the same time, we are going to build some models to predict the future figures, and then compare the results to find out which is best. This report will demonstrate the procedures in details about how we collected, processed, and manipulated the COVID-19 data.

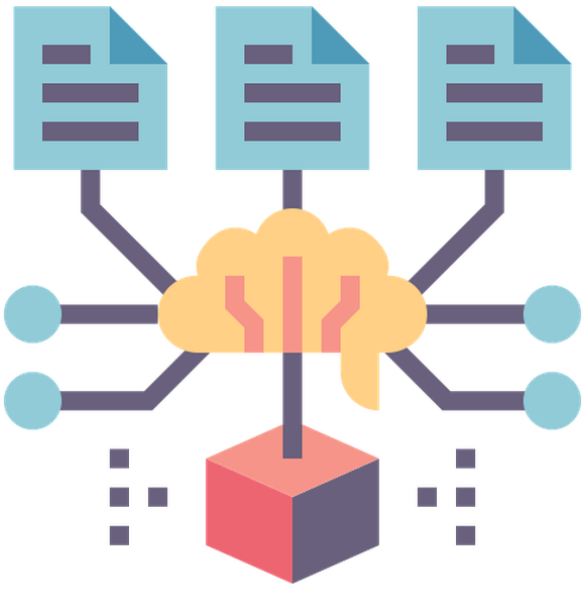


Figure 4: Prediction model.

Problem

1. **Data Collection - 2 pts**
   Collect data on the disease situations of countries and around the world.

2. **Data Analysis - 3 pts**
   Provide analysis and developments in the period from July 2020 to December 2020, and from January 2021 to May 2021.

   (a) Do the statistics agree to any distribution ? (Normal, Poisson, Gamma, Exponential, etc.)

   (b) We want to see what the difference is before we use the vaccine and after that.

3. **Model prediction - 3 pts**
   Build a model to predict the epidemic situation in the next 1 week - 3 months for:

   (a) Global

   (b) 3 countries with the highest number of infections: USA, India, Brazil

   (c) Some other Asian countries: Korea, Japan, Vietnam
   Note: We can use many different mathematical models (e.g linear regress) or use machine learning tools, deep learning... or whatever source you find.

4. **Model Analysis - 2 pts**

   (a) Testing, model analysis, influencing factors, and the ability to use the cross-country model.

   (b) According to the perception and evaluation of each group, which model do you think is the best and closest to reality? Further analysis of the influencing factors.

2 Tools and environments

In order to work with the collected data about this pandemic, we are going to use Python and Google Colaboratory. Built on top of Python are some key packages that handle specific tasks so as to help us progress with the research, such as pandas, matplotlib, seaborn, and numpy.

2.1 Python

Python is an easy-to-use programming language packed with a great deal of tools for data analysis and model building. The tools, called 'packages', are capable of visualizing, analyzing, and manipulating datasets.

Figure 5: Python programming language.

The syntax for a Python program is simple.

```python
# demo python program that determines even/odd number
n = int(input())
if (n % 2 == 0):
    print ("even number")
else:
    print ("odd number")
```

Listing 1: Demo code for Python.

As presented, the program skips all the hassles seen in other languages, for example, having to declare variable types. This makes the code more elegant while being simple enough for those who know little code to understand well.

2.2 Google Colaboratory

With the programming language settled, we need a stable and effective environment to run codes and store our outputs. Usually, we will use Jupyter Notebook, an open-source web application that allows creating and sharing documents with live code, equations, visualizations, and narrative text. But for the purpose of team work, Google Colaboratory comes out on top at our first choice. It functions identically to a Jupyter Notebook, but allows users to collaborate and run codes together.

Later in this report, it will be referred to as Google Colab.

Figure 6: Google Colaboratory.

2.3 Pandas

Pandas is a very powerful tool in data analysis and manipulation. Our research relies a lot on pandas, as it is one of the fastest and robust tools built on top of Python that are capable of handling data.

All our acts of data reading, parsing, and so on, are done with pandas.

Figure 7: Python pandas package.

2.4 Matplotlib

Matplotlib is another Python package with the ability to visualize data. With this package, we can create eye-catching and informative graphs which help a lot in understanding the data, and in assessing the effectiveness of the models.

Figure 8: Python matplotlib package.

2.5 Seaborn

Based on matplotlib, seaborn is a statistical data visualization tool. It also integrates well with pandas, and that is the main reason for its selection. Seaborn proves to be a good package because it helps users focus on what different aspects of the data represent, rather than *how* they are drawn and represented. In practical situations, there is no universally

perfect plotting for a set of data, so different kinds of graphs must be used. With seaborn's consistent dataset-oriented API, that matter is done with relative ease.

Figure 9: Python seaborn package.

2.6 Numpy

Numpy comes up as one of the best packages for numerical calculations. Model prediction requires a handful of mathematics, which is a field for numpy to shine.

Figure 10: Python numpy package.

2.7 SkLearn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Figure 11: SkLearn numpy package.

2.8 Pmdarima

Pmdarima has binary and source distributions for Windows, Mac and Linux (manylinux) on pypiunder the package name pmdarima and can be downloaded via pip.

The six mentioned packages are the core of our project, and we are going to use some additional ones in our work for the best result possible.

3 Overview of Machine Learning

In solving this problem, we mainly utilize the concepts of Machine Learning in analysing the dataset founded, building up prediction models as well as model comparison. It would be handy if we spend some time to briefly summarize some elementary concepts in advance, and we also add to the Appendix the Data pre-processing step and some fundamental algorithms in Machine Learning.

3.1 What is Machine Learning?

- Machine learning is a field of artificial intelligence involved in the research and construction of techniques that allow systems to "learn" automatically from data to solve specific problems.

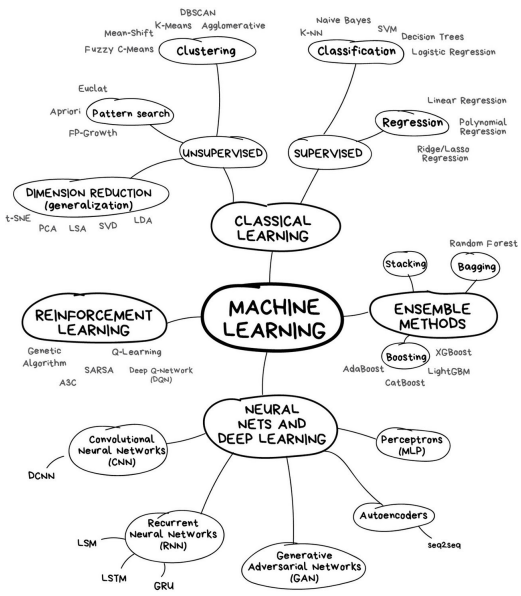


Figure 12: Machine Learning

3.1.1 Comparison between Machine Learning, AI and Deep Learning

- The simplest way to visualize the relationship of the above 3 concepts is to use the Venn diagram. AI - the first idea - the biggest, then machine learning, and finally deep learning - is what drives the explosion of AI today.

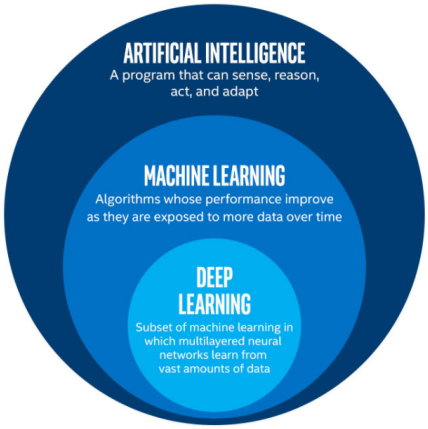


Figure 13: AI, ML and DL

3.2 Supervised learning and Unsupervised learning

3.2.1 Supervised learning

- Supervised learning is an algorithm that predicts the outcome of a new data (outcome) based on previously known pairs (input, outcome).

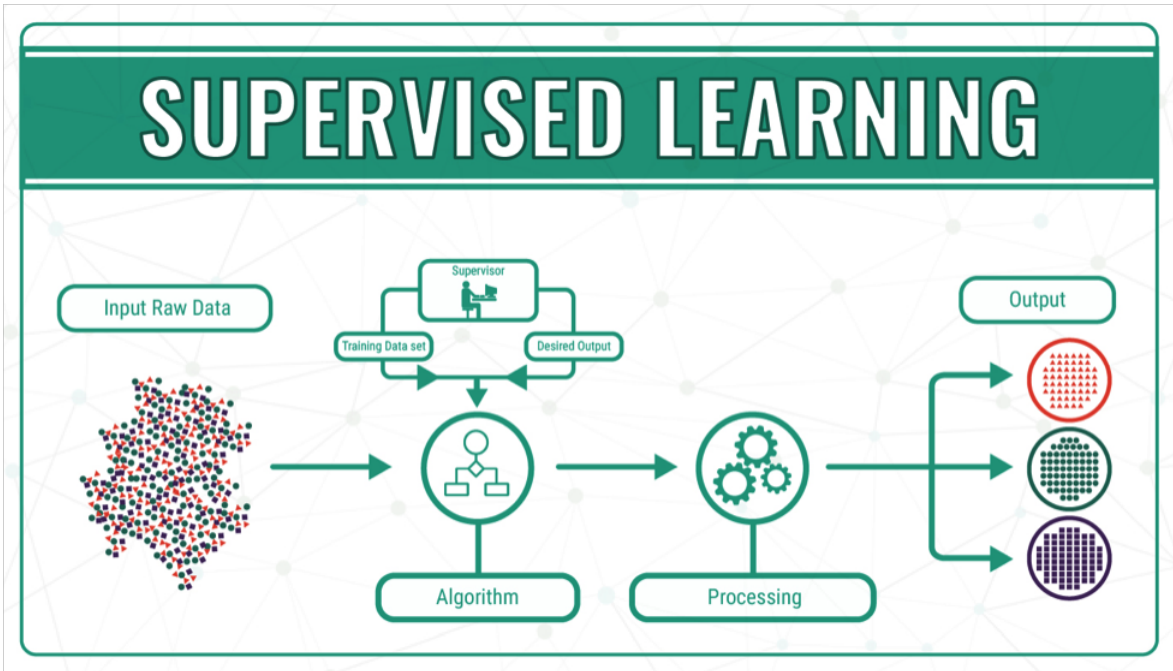


Figure 14: Supervised learning

- **The supervised learning algorithm:**
  - There are two main categories:
    * Classification: A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease"
    * Regression: A regression problem is when the output variable is a real value, such as "dollars" or "weight"

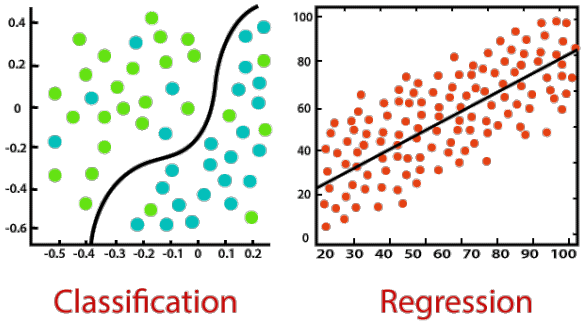


Figure 15: Classification and Regression

3.2.2 Unsupervised learning

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

- **Unsupervised learning problems:**
  - There are two main problems in Unsupervised learning:
    * Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y
    * Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

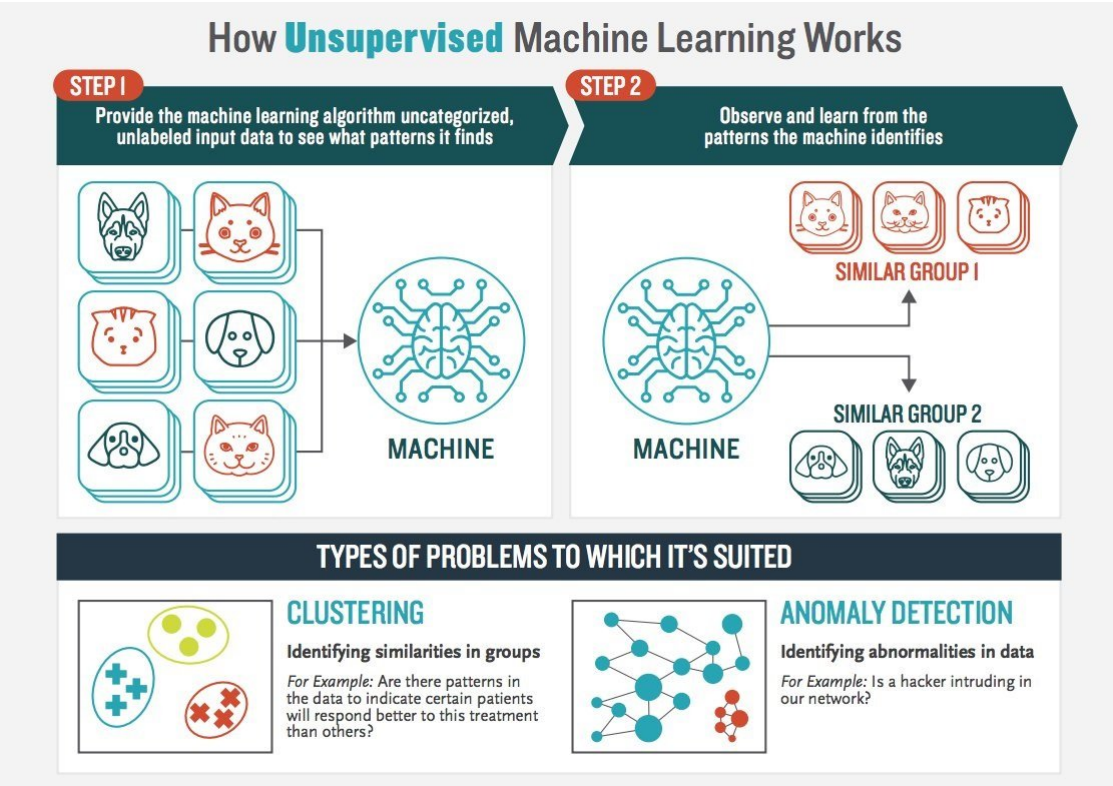


Figure 16: Works of Unsupervised learning

3.3 Reinforcement learning

- **What is reinforcement learning? How does it relate with other ML techniques?**
  - Reinforcement Learning(RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.

    Though both supervised and reinforcement learning use mapping between input and output, unlike supervised learning where feedback provided to the agent is correct set of actions for performing a task, reinforcement learning uses rewards and punishment as signals for positive and negative behavior.
  - As compared to unsupervised learning, reinforcement learning is different in terms of goals. While the goal in unsupervised learning is to find similarities and differences between data points, in reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent. The figure below represents the basic idea and elements involved in a reinforcement learning model.

- **Some applications of Rainforcement learning in real life:**
  - Reinforcement Learning in news recommendation:

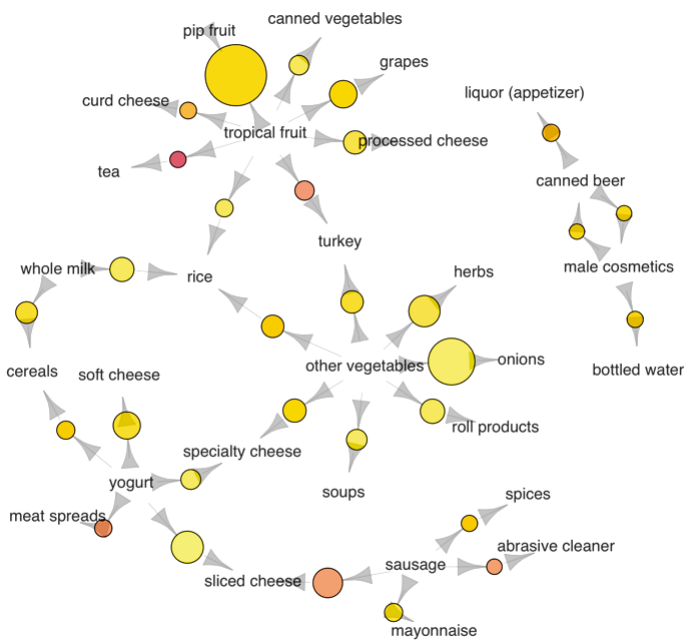


Figure 17: Clustering

- User preferences can change frequently, therefore recommending news to users based on reviews and likes could become obsolete quickly. With reinforcement learning, the RL system can track the reader's return behaviors. Construction of such a system would involve obtaining news features, reader features, context features, and reader news features. News features include but are not limited to the content, headline, and publisher. Reader features refer to how the reader interacts with the content e.g clicks and shares. Context features include news aspects such as timing and freshness of the news. A reward is then defined based on these user behaviors.

– Real-time bidding—Reinforcement Learning applications in marketing and advertising

  - In this paper, the authors propose real-time bidding with multi-agent reinforcement learning. The handling of a large number of advertisers is dealt with using a clustering method and assigning each cluster a strategic bidding agent. To balance the trade-off between the competition and cooperation among advertisers, a Distributed Coordinated Multi-Agent Bidding (DCMAB) is proposed.
  - In marketing, the ability to accurately target an individual is very crucial. This is because the right targets obviously lead to a high return on investment. The study in this paper was based on Taobao—the largest e-commerce platform in China. The proposed method outperforms the state-of-the-art single-agent reinforcement learning approaches.

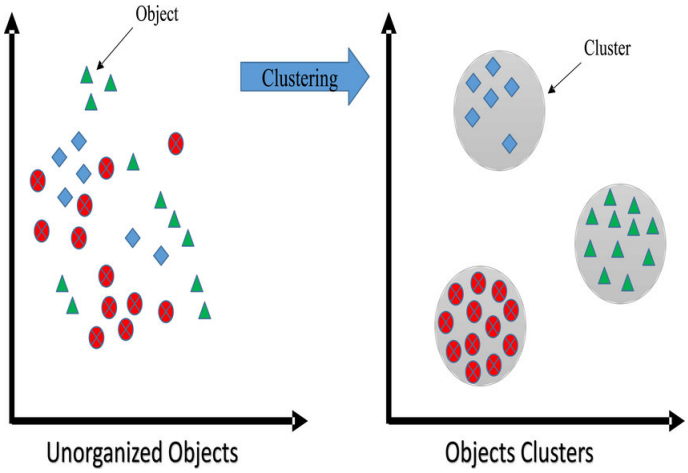


Figure 18: Association

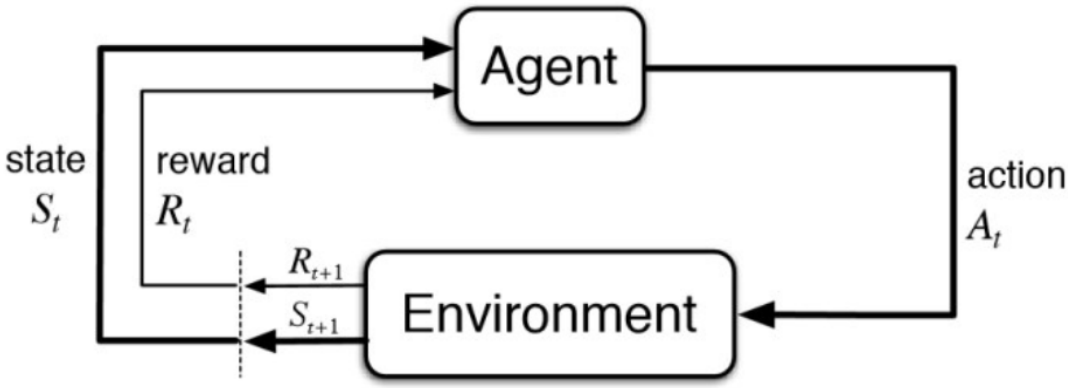


Figure 19: Structure of Rainforcement learning

3.4 Ensemble learning

- **What is Ensemble learning ?**
  - Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting. This article focuses on classification related applications of ensemble learning, however, all principle ideas described below can be easily generalized to function approximation or prediction type problems as well.

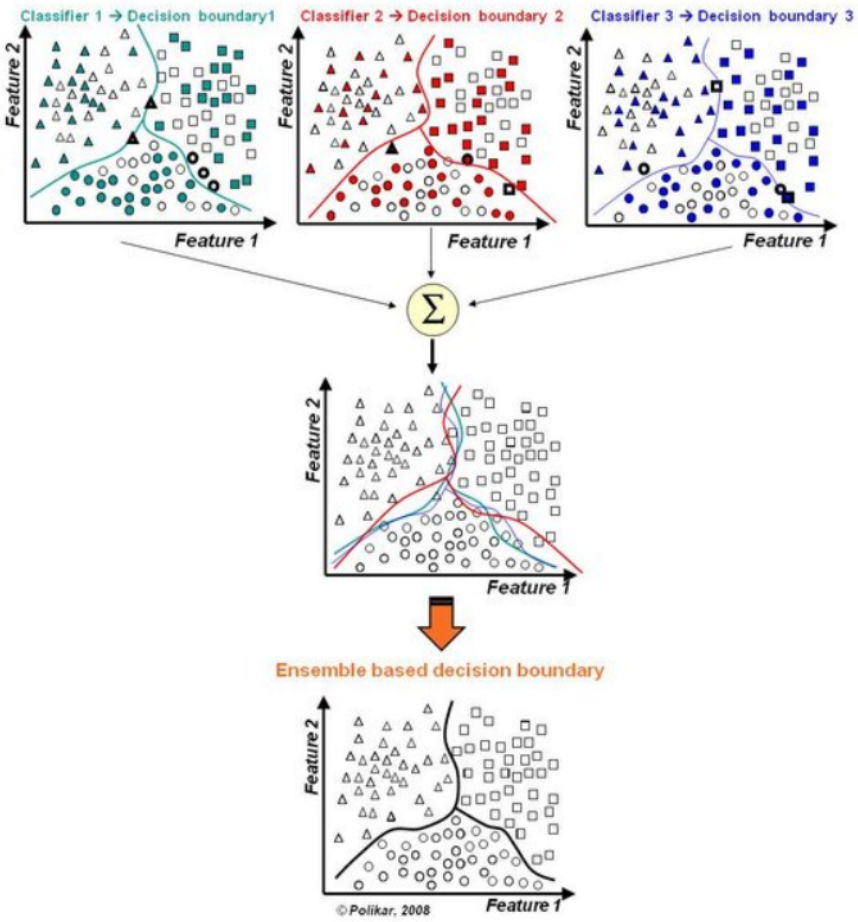


Figure 20: Combining classifiers with different decision boundaries reduce error

3.5 Deep learning

- **What is deep learning?**
  - Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.
  - In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.
- **Examples of Deep Learning at Work:**
  - Deep learning applications are used in industries from automated driving to

medical devices.

  * Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
  * Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.
  * Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
  * Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.
  * Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

- **How Deep Learning Works?**
  - Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.
  - The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.
  - Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

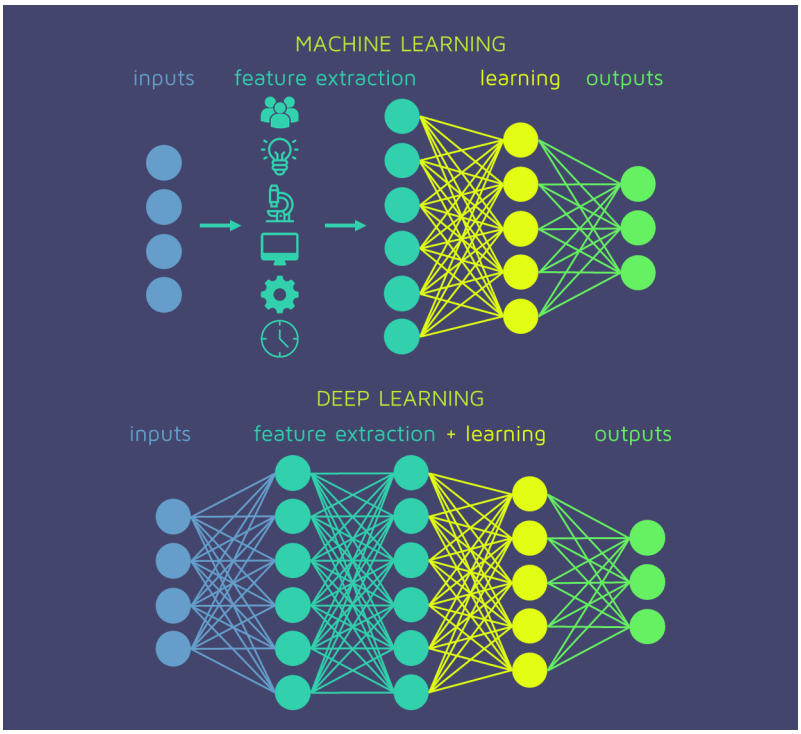


Figure 21: Deep learning process

4 Data collection

Most of the data about the virus is publicly available, and is updated daily. They often come in CSV (comma-separated values, much like an Excel Spreadsheet) and JSON (JavaScript object notation) files. Sites like GitHub, Our World In Data, Kaggle, and so on, provide a wide range of choices for users when it comes to data like these.

4.1 How many data sets are enough?

With the large variety of choices available, we can proceed to choosing only one or many data sets. Say, you need to know exactly at a specific location and a moment of time how many cases were witnessed, so you acquire a data file that contains exactly what you want. On the other hand, you also care about the effectiveness of vaccinations, but the initial file does not cover it. This is when another data set comes into the picture.

It is actually safe to choose many sources for our data, because they remain relevant to each other. They cover different aspects of a singular topic, which is SARS-CoV-2, so we need not worry about the risk of overlapping figures or mismatched numbers.

4.2 Importing data into Google Colab

There are many ways to import the data into Google Colab. We can create a Google Drive folder, then get the authentication key from Google to let Colab access the data within it. Also, we can upload the files directly from our computers every time we run our programs. But then again, the mentioned procedures take quite a lot of time.
To counter that, thanks to GitHub, we can directly import the necessary data from there in only a line of code.

4.3 Data pre-processing

Data pre-processing is an essential procedure in data analysis and model prediction. A preferred dataset which contains our designated data may contain other unnecessary attributes, proving no use to the overall work. Those attributes need to be removed for better readability. What is more, a desired data column, for example, may contain null data - entries that were not collected during the corresponding time. If we do not exclude it, our work will also be affected.

First, we need to load the data into the coding environment.

```
df_new = pd.read_csv("https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv", parse_dates=['date'], index_col='date')
df_new.tail()
# df is short for data frame
```

Listing 2: Data loading.

In the code script, read_csv is a pandas function that let us read CSV files. Then, the tail() function is used to return the last n rows from the object based on position, useful

for quick data verification.

Then, not all data columns are necessary. The pre-processing phase also deals with this, as we can keep only the meaningful data relative to our research.

```
df = df_new[["continent", "location", "population", "total_cases", "new_cases", "new_cases_smoothed", "total_deaths", "new_deaths", "new_deaths_smoothed", "total_tests", "new_tests", "positive_rate"]]
df.tail()
```

Listing 3: Keeping necessary columns.

There is a handful of functions built into pandas that let us manipulate the dataset. We can use *dropna()* to filter null values for a smooth coverage of figures.

Finally, we can use the *info()* function to see the information of data.

```
df.info()
```

Listing 4: Verification step.

which produces the following output:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 98011 entries, 2020-02-24 to 2021-06-24
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   continent             93405 non-null  object
 1   location              98011 non-null  object
 2   population            97373 non-null  float64
 3   total_cases           94574 non-null  float64
 4   new_cases             94571 non-null  float64
 5   new_cases_smoothed    93561 non-null  float64
 6   total_deaths          84491 non-null  float64
 7   new_deaths            84647 non-null  float64
 8   new_deaths_smoothed   93561 non-null  float64
 9   total_tests           43918 non-null  float64
 10  new_tests             44245 non-null  float64
 11  positive_rate         48007 non-null  float64
dtypes: float64(10), object(2)
memory usage: 9.7+ MB
```

We see that the data is not null and it is pre-processed, so we will skip this step.

4.4 Data description

4.4.1 Cases dataset: Source

This dataset is collected by Our World In Data (OWID) and hosted on GitHub. We are going to work with this dataset in the CSV format. It contains information related to confirmed deaths, cases, hospitalizations, and testing, as well as some other miscellaneous figures.

We will mostly focus on **Confirmed cases and deaths**, specifically the daily figures collected from July 2020 - December 2020 and January 2021 - May 2021. Our predictions

also rely on the mentioned fields.

Some key factors to take into considerations are:

- iso_code: collected from International Organization for Standardization that represents countries. It is the ISO 3166-1 alpha-3 - three-letter country code.
- continent, location, date: collected by Our World In Data, which represent continents of the geographical locations, geographical locations and observation date, respectively. The date is in the format of **yyyy-mm-dd**.
- new_cases, new_deaths: collected by the Center of Systems Science and Engineering (CSSE) at Johns Hopkins University.
- new_cases_smoothed, and other *smoothed* data columns: 7-day smoothed data, derived from the plain attributes.

4.4.2 Vaccinations by age group: Source

Also collected by OWID, this dataset focuses on vaccination figures from available sources around the world. The metrics are calculated in the following fashion: For example, 4 people are in for a vaccination program, which requires 2 doses for full immunity effect. A has received 2 doses, B has received 1 dose, C has received 1 dose, and D has not received any. The total number of doses will be **4**: 2+1+1, the total number of vaccinated people will be **3**: A, B, and C, and the total of people *fully* vaccinated will be 1 (A).

Some key factors to take into considerations are:

- iso_code: collected from International Organization for Standardization that represents countries. It is the ISO 3166-1 alpha-3 - three-letter country code.
- continent, location, date: collected by Our World In Data, which represent continents of the geographical locations, geographical locations and observation date, respectively. The date is in the format of **mm-dd-yyyy**.
- total_vaccinations: the total number of administered vaccine doses. The unit is single dose, so it may not be equal to the total people vaccinated, so if a person receives a dose, the figure increases by 1, and if that person receives the second dose, the figure increases by 1 again.
- daily_vaccinations: new doses given per day (7-day smoothed). For countries that do not report daily vaccination data, the doses are assumed to be changed equally on a daily basis over any periods where no data was reported.
- people_vaccinated: the total number of people who received at least a dose of vaccine. If a person receives the first dose, this figure increases by 1, and if this person takes the second dose, the figure is not affected.
- people_fully_vaccinated: the total number of people who received all doses. The figure increases by 1 only if a person complete their second vaccination.

4.4.3 Vaccinations by manufacturers: Source

This dataset is within the OWID folder that contains the vaccinations by age groups dataset.

There is a handful of COVID-19 vaccines to date, namely Pfizer/BioNTech, Johnson&Johnson, Moderna, and Oxford/AstraZeneca. It is interesting to take a look at how countries around the world use different products from manufacturers.
This dataset has 8291 lines and 4 columns:

- location: geographical locations of countries.
- date: described in **yyyy-mm-dd** format.
- vaccine: shows the manufacturer.
- total_vaccinations: the total number of administered vaccine doses. The unit is single dose, so it may not be equal to the total people vaccinated, so if a person receives a dose, the figure increases by 1, and if that person receives the second dose, the figure increases by 1 again.

4.4.4 Dedicated data for Vietnam: Source

We intend to visualize cases on the map of Vietnam, so this dataset proves to be useful. Available in geojson file type, it contains coordinates of provinces in the country and their COVID-19 figures. Later, we are going to use the **geopandas** package in Python to plot out individual, province-wise cases.

Each province comes with the following properties:

- Code: the dedicated code for each province.
- infected: the confirmed infected COVID-19 patients.
- death: the total deaths caused by the virus.
- recovered: the total recovered patients.
- active: the number of patients who have not yet recovered.
- coordinates: comes with attributes 0 and 1, showing the longitude and latitude of the province.

4.4.5 The first dataset for SIR model: Source

This in a daily-updated dataset, collected and distributed by Johns Hopkins University Center for Systems Science and Engineering (JHU CSEE).

The notable data featured in this dataset consists of:

- Country_Region: shows the country/region with its local COVID-19 data.

- Last_Update: the last date at which the data is updated.
- Confirmed: total cases recorded of the region.
- Deaths: official death record of the region.
- Recovered: the total number of patients who recovered from SARS-CoV-2.
- Active: the total number of patients that have not recovered from the disease.
- Incident_Rate: describes how fast the disease occurs in the region.
- Mortality_Rate: describes the death/total ratio.

4.4.6 The second dataset for SIR model: Source

This dataset contains total population of countries and regions all around the globe. The time recorded is the year 2020.

5 Data Analysis

5.1 Vaccination in history and in COVID-19 pandemic

These days, we are pretty much aware of the flu vaccine and the vaccine of several other diseases. Where does the concept of vaccination come from ?

In 1796, Edward Jenner was the first to do his study of a new method to protect against smallpox which is taking material from a blister of someone infected and inoculating it into another skin (called arm to arm inoculation). His approach to the problem in a scientific manner had led to large scale vaccine production throughout the 20th century when scientific knowledge had developed enough. The very first recommended vaccines were pertussis (1914), diphtheria (1926), tetanus (1938). Up to now, with the help of the World Health Organization, many vaccines for children and adults have been intensively researched and developed. Over the years, vaccines have become a key factor in helping people overcome many major pandemics in history.

Infact, vaccines contain the same antigens that cause the disease but they were killed or weakened. Vaccines will stimulate a person's immune system to produce immunity to a specific disease by allowing the immune system to take time to produce antibodies designed to fight the virus and the immune system will remember that antigen the next time it enters (called immunity).

In 2019, coronavirus was first detected in Wuhan, China and had promptly set off a global pandemic. Not long after, because of the rapid spread of the virus, the World Health Organization declared a public health emergency of international concern, the highest level of alarm under international law.

In COVID-19 as well as any pandemic, there is often five trends of number of daily new cases to monitor the epidemic situation:

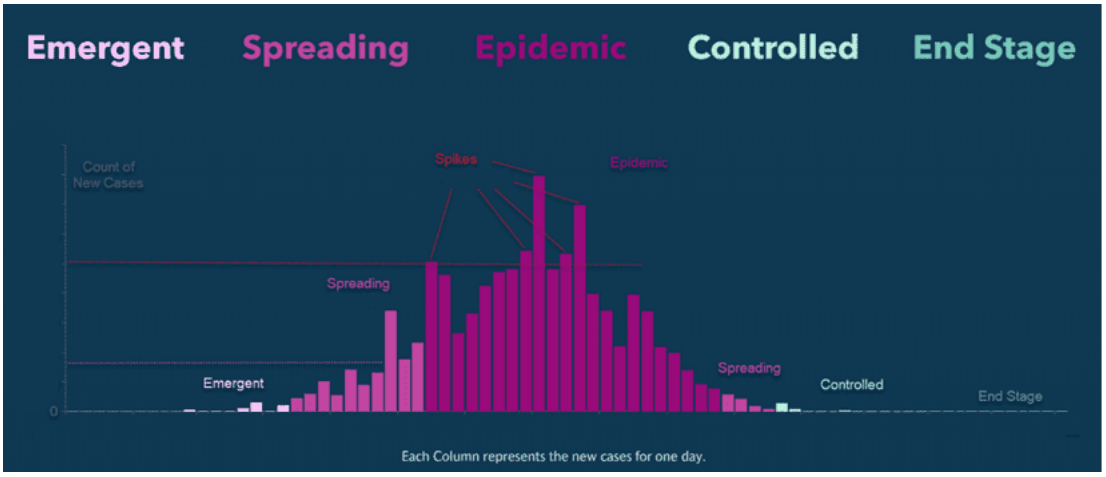


Figure 22: Trends commonly seen in an epidemic (source: esri.com)

+ Emergent: Early stages of outbreak.
+ Spreading: Early stages and depending on an administrative area's capacity, this may represent a manageable rate of spread.
+ Epidemic: Uncontrolled spread.
+ Controlled: Average less than x new cases per y people for a period of time.
+ End Stage: Average less than one new case every some days in a long time.

So far, to reach the controlled stage as soon as possible,countries around the globe are racing to roll out COVID-19 vaccines.

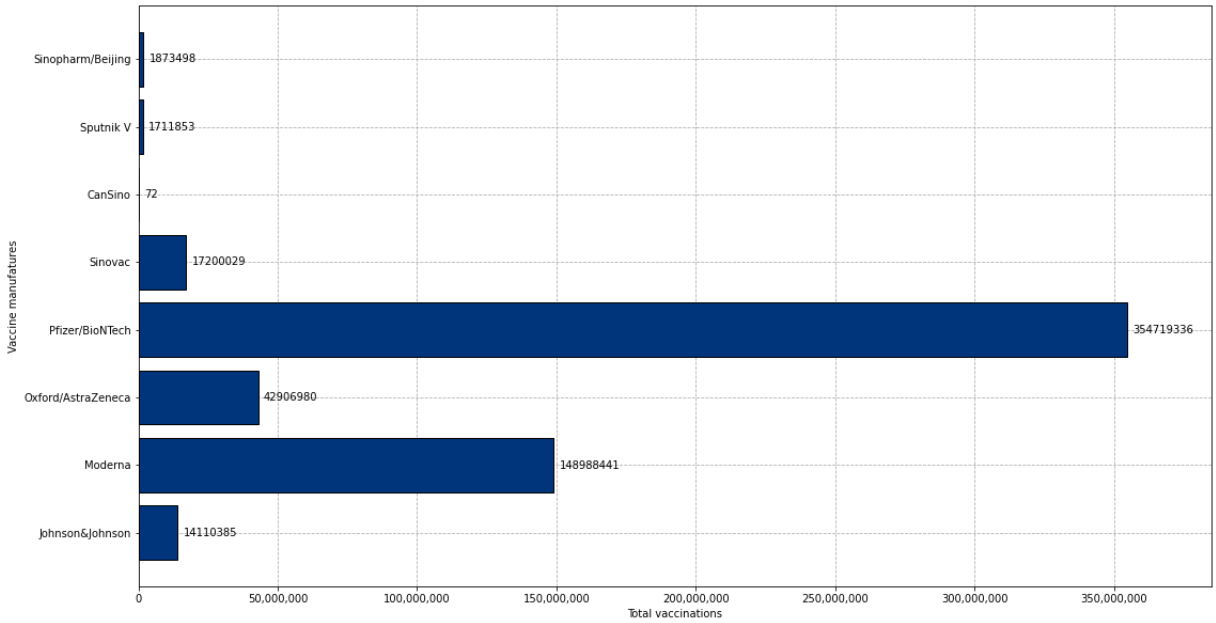


Figure 23: Globally used vaccines dose till 31/5/2021

Besides, a wide variety of vaccines does not mean that they are evenly distributed around the world. An essential pillar of the vaccination challenge is ensuring that enough doses are available globally. Uneven access to vaccines would not be unprecedented in history and COVID-19 pandemic as well. Many low-income and middle-income countries will face barriers in delivering vaccination programmes to their citizens.

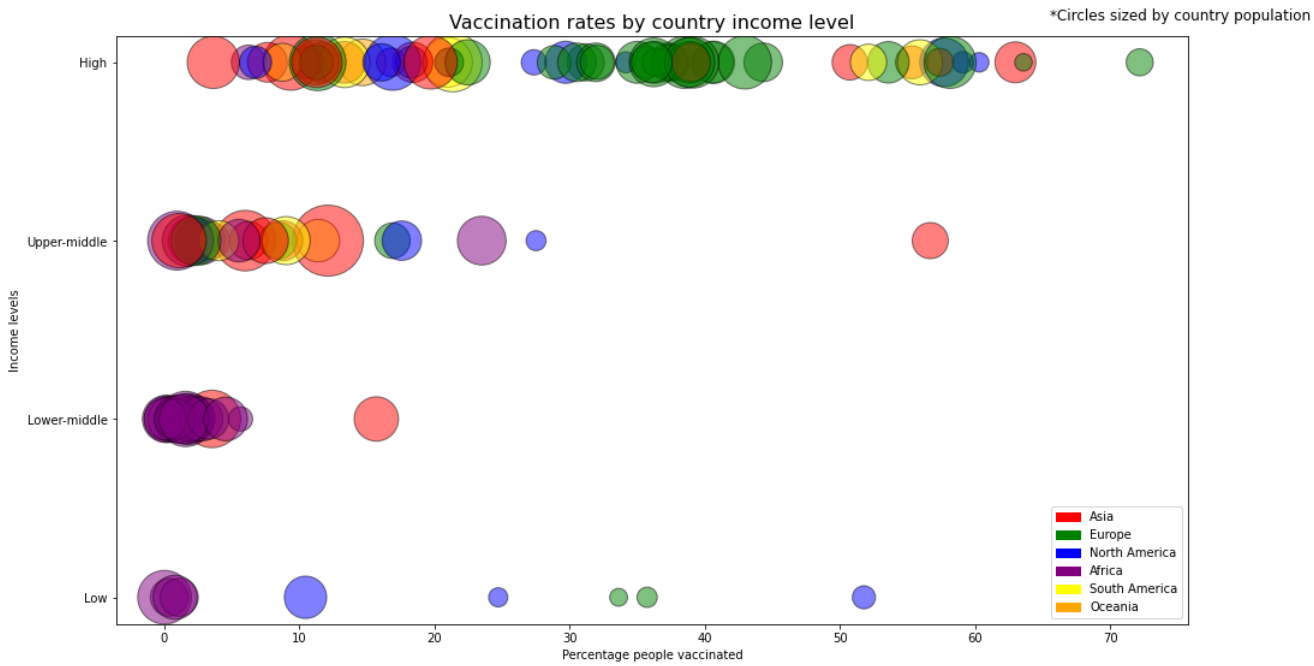


Figure 24: Vaccines accessibility of countries in 6 continents

+ Income levels (gdp per capita) : low (under 1026 USD), lower-middle (1026-3995 USD), upper-middle (3996-12375 USD), high (above 12376 USD).

Code of figure 24:

```
plt.figure(figsize=(16,9))

colors = {"Asia" : "red", "Europe" : "green", "North America" : "Blue",
    "Africa" : "purple", "South America" : "yellow", "Oceania" : "orange"
    }
data_continent = data[data['continent'].notna()]
pop = np.log10((data_continent["population"])) ** 4 / 2

data_continent["income_level"] = 0
for country in data_continent.index:
  if ((data_continent.loc[country]["gdp_per_capita"] >= 1026) & (
    data_continent.loc[country]["gdp_per_capita"] <= 3995)):
     data_continent.at[country, "income_level"] = 1
  elif ((data_continent.loc[country]["gdp_per_capita"] >= 3996) & (
    data_continent.loc[country]["gdp_per_capita"] <= 12375)):
     data_continent.at[country, "income_level"] = 2
  elif ((data_continent.loc[country]["gdp_per_capita"] >= 12376)):
     data_continent.at[country, "income_level"] = 3

plt.scatter(data_continent["people_vaccinated_per_hundred"],
    data_continent["income_level"], s = pop,
            c = data_continent["continent"].map(colors), edgecolors='k',
     alpha = 0.5)
```

```
plt.xlabel("Percentage of people vaccinated")
plt.ylabel("Income levels (gdp per capita)")
plt.title("Vaccination rates by country income level", fontsize=16)
pop = []
for continent in colors.keys():
  pop.append(mpatches.Patch(color=colors[continent], label=continent))
plt.legend(handles=pop)

income_levels = ["Low", "Lower-middle", "Upper-middle", "High"]
plt.yticks(range(len(income_levels)), income_levels)
plt.figtext(0.8,0.9,'*Circles sized by country population',fontsize=12,
    ha='left')

plt.show()
```

In the following section, by analyzing the real data of the current pandemic, we would have a more general view of inoculation.

5.2 Vaccine Analysis

5.2.1 COVID-19 vaccines prevent the spread of virus

According to the data source of Our World In Data, since most countries start vaccinating from mid-December 2020 to the end of January 2021, we decided to divide the data into 2 time periods in order to analyze them specifically. The first period lasts from July 1, 2020 to December 30, 2020 (we call it unvaccinated phase) and the second period lasts from January 1, 2021 to May 31, 2021 (vaccinated phase).

To approach analysing this dataset , we choose China (first country to identify the disease) and three countries with the highest deaths and confirmed cases : United States, India, Brazil. These four nations also have the highest percentage of immunized populations by the end of the second phase.

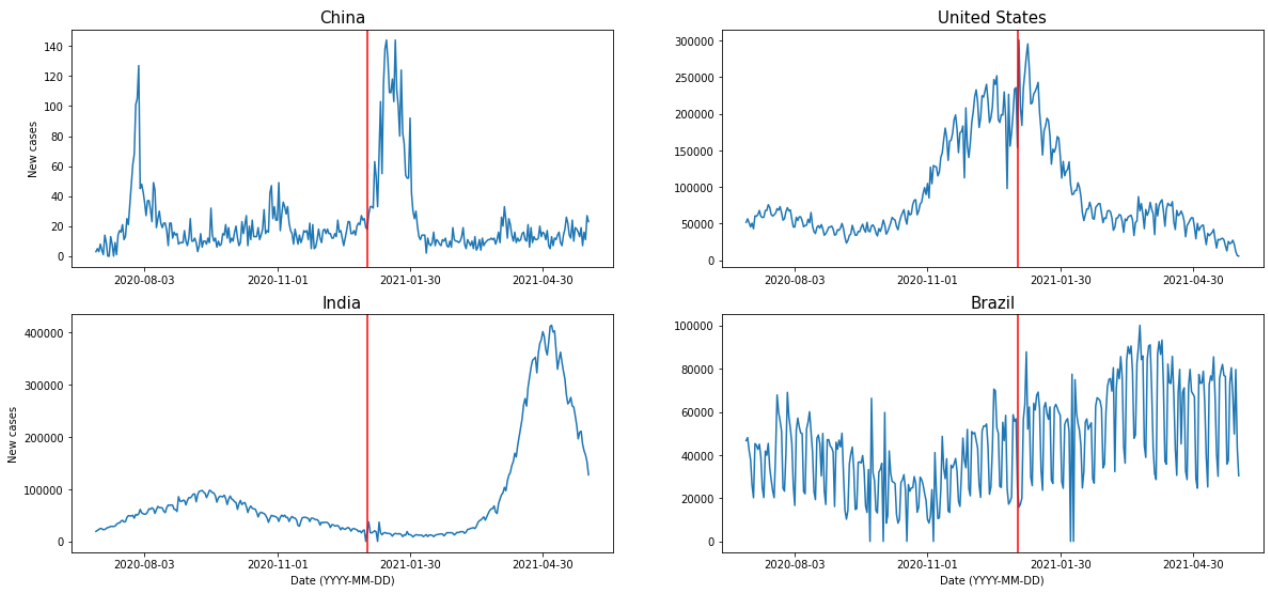


Figure 25: Daily new case of 4 countries - 2 phases separated by red line.

In China daily new cases stay low (under 150 cases per day) after and even before being vaccinated. The United States significantly increases in trend in the first period and witnesses a controlled trend just after a few months being vaccinated. India was seen to decrease for 4 months but unexpectedly uncontrolled spread and reach a peak in May, then decline again. Brazil reports nonstop new infections. Hence, with a brief look at this information, we intend to divide them temporarily into 4 groups:

+ Group1 (down_down): low/going down of new cases in both period.
+ Group2 (up_up): going up of new cases in both period.
+ Group3 (up_down): going up of new cases in 1st period and going down of new cases in 2nd period.

+ Group4 (down_up): going down of new cases in 1st period and going up of new cases in 2nd period.

From the above groups, we will compare the percentage of total vaccinated people over populations between group's members and between 4 groups to check out how the vaccines affect daily new cases. 21 countries following was picked by population size with the percentage of vaccinated people:

```
Total: 189

Increase then increase: 51
Afghanistan, Angola, Argentina, Armenia, Azerbaijan, Belgium, Bhutan, Bosnia and Herzegovina, Botswana, Bulgaria, Cabo Verde, Canada,
Colombia, Costa Rica, Croatia, Cuba, Cyprus, Fiji, France, Georgia, Greece, Guyana, Hungary, Iran, Islamic Republic of, Japan,
Jordan, Kenya, Korea, Republic of, Lao People's Democratic Republic, Malaysia, Mali, Mauritius, Mongolia, Nepal, Netherlands,
Norway, Pakistan, Palestine, State of, Paraguay, Saint Kitts and Nevis, Seychelles, Somalia, Sri Lanka, Syrian Arab Republic, Taiwan, Province of China,
Thailand, Timor-Leste, Turkey, Uganda, Ukraine, Uruguay,

Increase then decrease: 73
Albania, Algeria, Andorra, Antigua and Barbuda, Austria, Barbados, Belarus, Belize, Brunei Darussalam, Burkina Faso, Chad, Comoros,
Congo, The Democratic Republic of the, Czechia, Denmark, Dominica, Eritrea, Estonia, Eswatini, Finland, Germany, Grenada,
Holy See (Vatican City State), Hong Kong, Iceland, Indonesia, Ireland, Israel, Italy, Jamaica, Latvia, Lebanon, Lesotho, Libya,
Liechtenstein, Lithuania, Luxembourg, Malta, Mauritania, Mexico, Moldova, Republic of, Monaco, Montenegro, Morocco, Mozambique,
Myanmar, Namibia, New Zealand, Niger, North Macedonia, Panama, Poland, Portugal, Romania, Russian Federation, Rwanda,
Saint Lucia, Saint Vincent and the Grenadines, San Marino, Serbia, Slovakia, Slovenia, South Sudan, Spain, Sudan, Sweden,
Switzerland, Togo, Tunisia, United Arab Emirates, United Kingdom, United States, Zimbabwe,

Decrease then increase: 43
Australia, Bahamas, Bahrain, Bangladesh, Bolivia, Plurinational State of, Brazil, Burundi, Cambodia, Cameroon, Central African Republic, Chile,
Congo, Djibouti, Ecuador, Egypt, Equatorial Guinea, Ethiopia, Guatemala, Guinea, Honduras, India, Iraq, Kazakhstan, Kuwait, Kyrgyzstan,
Madagascar, Maldives, Nicaragua, Oman, Papua New Guinea, Peru, Philippines, Qatar, Saudi Arabia, Singapore, Solomon Islands, Suriname,
Trinidad and Tobago, Uzbekistan, Vanuatu, Venezuela, Bolivarian Republic of, Viet Nam, Yemen,

Decrease then decrease: 22
Benin, China, Côte d'Ivoire, Dominican Republic, El Salvador, Gabon, Gambia, Ghana, Guinea-Bissau, Haiti, Liberia, Malawi, Marshall Islands, Nigeria,
Samoa, Sao Tome and Principe, Senegal, Sierra Leone, South Africa, Tajikistan, Tanzania, United Republic of, Zambia,
```

Figure 26: List of countries divided into four groups.

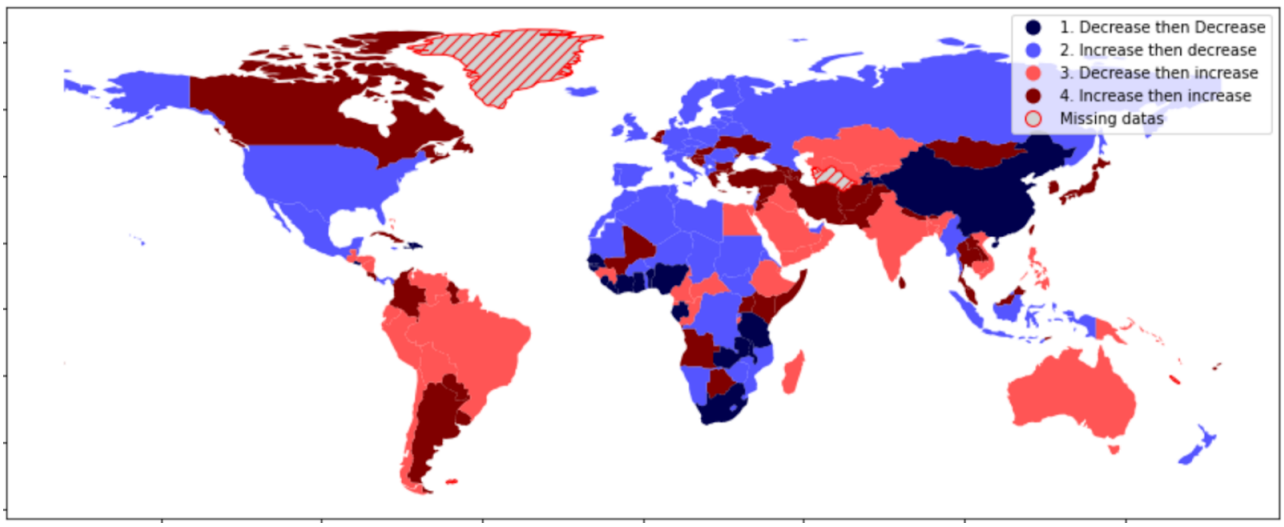


Figure 27: World map plot of 4 groups.

Code of figure 27:

```
data_two_periods_raw = []
```

```
nan_country_list = np.setdiff1d(data_covid["iso_code"].unique(), np.array(new_cases_up_up + new_cases_down_down + new_cases_up_down + new_cases_down_up))

for country in new_cases_up_up:
  data_two_periods_raw.append([str(country), '4. Increase then increase'])
for country in new_cases_up_down:
  data_two_periods_raw.append([str(country), '2. Increase then decrease'])
for country in new_cases_down_up:
  data_two_periods_raw.append([str(country), '3. Decrease then increase'])
for country in new_cases_down_down:
  data_two_periods_raw.append([str(country), '1. Decrease then Decrease'])
for country in nan_country_list:
  data_two_periods_raw.append([str(country), ])
data_two_periods = pd.DataFrame(data_two_periods_raw, columns= ["Country", "Trend"])

world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
world = data_two_periods.merge(world, left_on=  "Country", right_on= "iso_a3")
world = geopandas.GeoDataFrame(world)
world = world[(world.pop_est>0) & (world.name!="Antarctica")]
world['trend'] = world.Trend
world.plot(column="trend", legend=True, cmap="seismic", figsize=(15, 10), missing_kwds={
    "color": "lightgrey",
    "edgecolor": "red",
    "hatch": "///",
    "label": "Missing datas",
    },
);
```

From the above groups, we will compare the percentage of total vaccinated people over populations between group's members and between 4 groups to check out how the vaccines affect daily new cases. 21 countries following was picked by population size with the percentage of vaccinated people:

+ Over 1,000,000,000: China (58%), India (12%).
+ 100,000,000-350,000,000: US (50.2%), Indonesia (6%), Pakistan (2.5%), Brazil (21.2%), Nigeria (1%), Russia (11.3%), Japan (7.2%), Philippines (3.6%).
+ 50,000,000-100,000,000: Vietnam (1%), Germany (42.8%), Thailand (3.6%), UK (58%), South Africa (1.2%), South Korea (10.5%).
+ under 50,000,000: UAE (about 60%), Canada (56.9%), Malaysia (5.9%), Cambodia (15.4%), Singapore (39.0%).

—> Four groups after sorting, nations with low immunized rate all witness an increase in the number of new infections in the second phase:

+ Up_up (6) : Pakistan (2.5%), Japan (7.2%), Thailand (3.6%), South Korea (10.5%), Canada (56.9%), Malaysia (5.9%)
  -> Exception : Canada (small population size, vaccinated rate high and new infections still increase in 2nd phase however the rate slowed down)

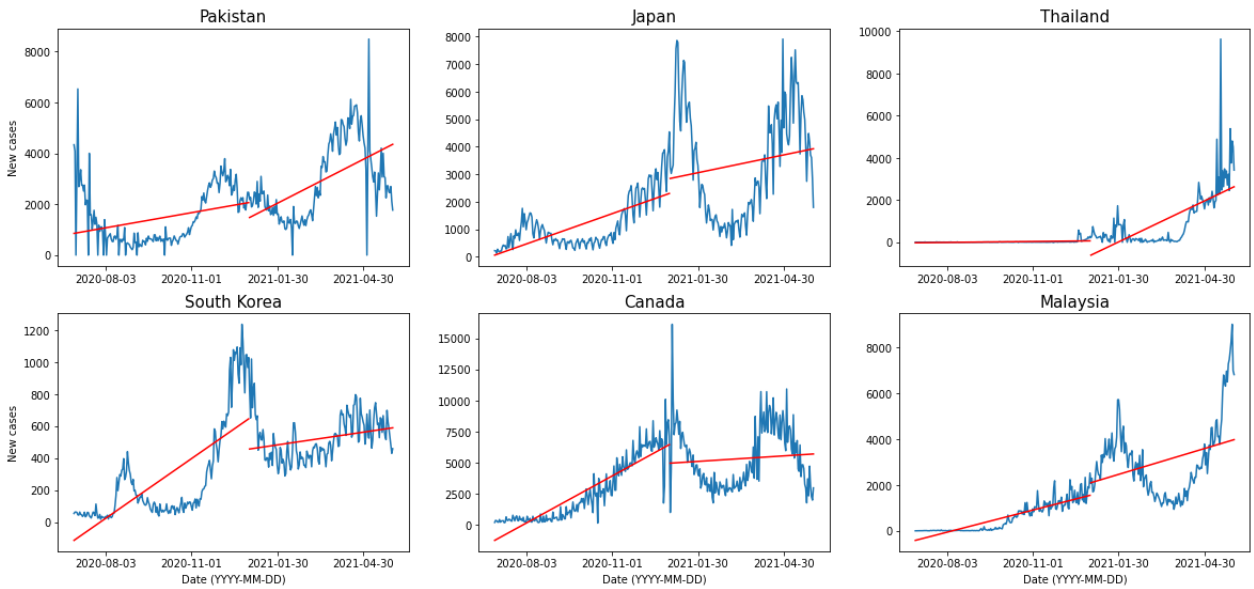


Figure 28: General trend of up_up group.

Code of figure 28:

```
loc = matplotlib.ticker.MultipleLocator(base=90)
ax.xaxis.set_major_locator(loc)

fig, axs = plt.subplots(2, 3, figsize = (20, 9))
fig.suptitle('Increase then increase', fontsize = 20)

inspecting_countries = [["Pakistan", "Japan", "Thailand"], ["South Korea", "Canada", "Malaysia"]]

for row in range(2):
  for col in range(3):
    axs[row, col].plot(data_countries[data_countries["location"] == inspecting_countries[row][col]]["new_cases"])
    axs[row, col].axes.get_xaxis().get_label().set_visible(row == 1)
    axs[row, col].axes.get_yaxis().get_label().set_visible(col == 0)
    axs[row, col].set_title(inspecting_countries[row][col], fontsize = 15)
    data_total_cases_single_country = data_covid[data_covid["location"] == inspecting_countries[row][col]]["new_cases"].loc[start_date:end_date].dropna()

```

```
    first = data_total_cases_single_country.loc[start_date:"2020-12-31"]
    x1 = (first.index - first.index[0]).days
    res1 = linregress(x1, first)

    second = data_total_cases_single_country.loc["2021-01-01":end_date]
    x2 = (second.index - second.index[0]).days
    res2 = linregress(x2, second)

    axs[row, col].plot(first.index, res1.intercept + res1.slope*x1, 'r')
    axs[row, col].plot(second.index, res2.intercept + res2.slope*x2, 'r'
   )

for ax in axs.flat:
    ax.set(xlabel='Date (YYYY-MM-DD)', ylabel='New cases')
    ax.xaxis.set_major_locator(loc)
```

+ Up_down (6) : United States (50.2%), Indonesia (6%), Russia (11.3%), Germany (42.8%), United Kingdom (58%), United Arab Emirates (about 60%).
  -> Exception: Indonesia, Russia (large population size, vaccinated rate low but new infections still plunge in period 2).

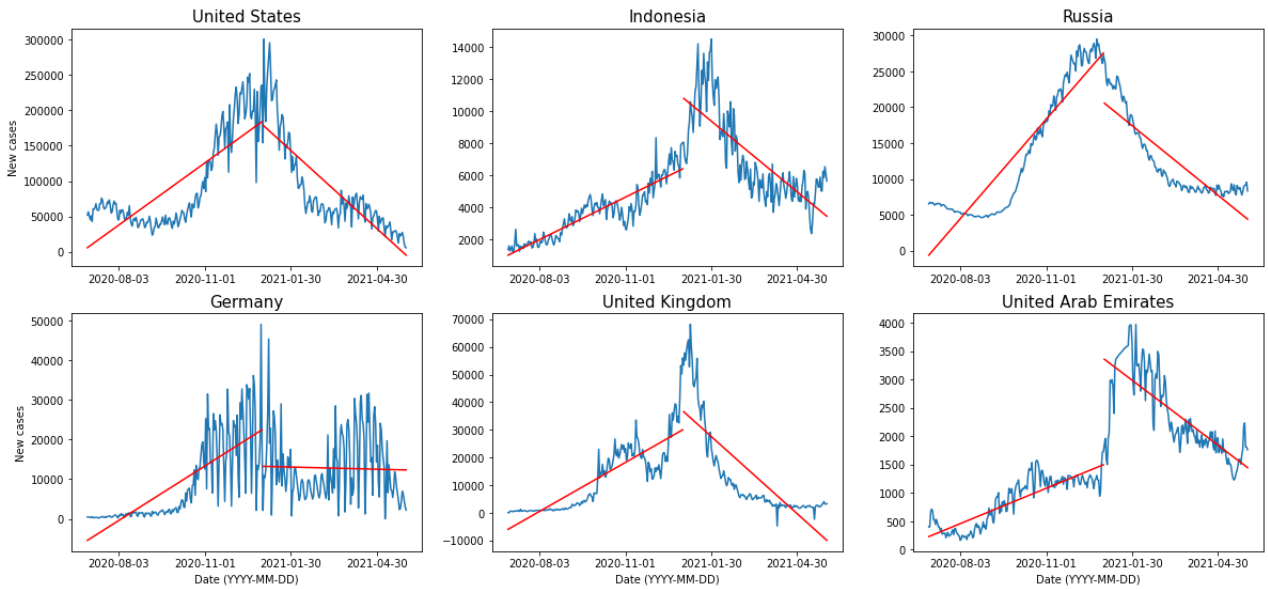


Figure 29: General trend of up_down group.

+ Down_up (6) : India (12%), Brazil (21.2%), Philippines (3.6%), Vietnam (1%), Cambodia (15.4%), Singapore (39.0%).
  -> Exception: Singapore (stay unchanged in vaccinated phase).

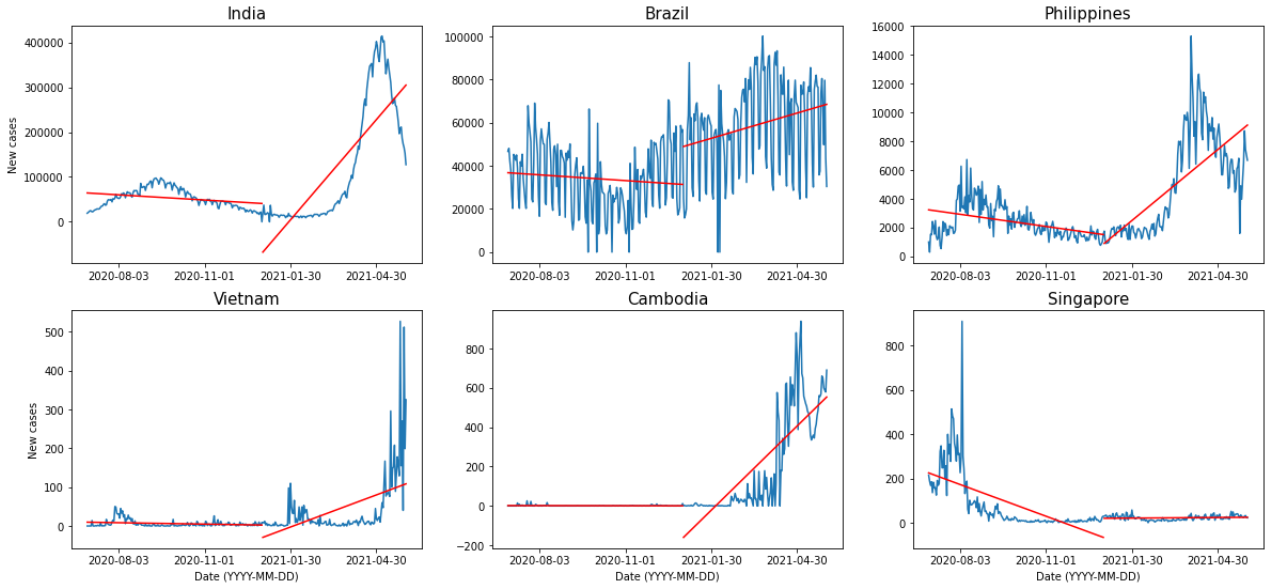


Figure 30: General trend of down_up group.

+ Down_down (3) : China (58%), Nigeria (1%), South Africa (1.2%).
  -> Exception : Nigeria, South Africa (medium population, vaccinated rate low but new infections still decrease in both period).

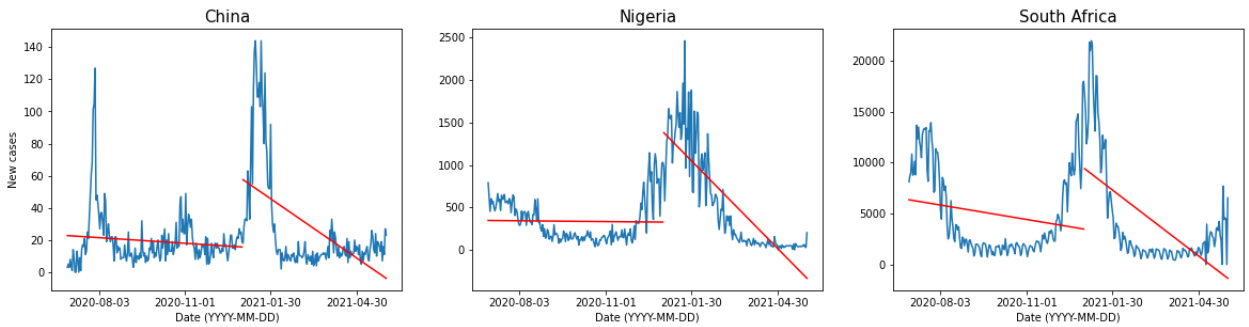


Figure 31: General trend of down_down group.

As can be seen, without vaccination in phase 1, whenever population size is small or large, over half of those countries showed an increased trend. Predictably, the number of daily new cases tends to decrease in most countries with high rates of vaccination in period 2 and briefly, COVID-19 vaccines rollout has helped reduce new cases. However, the down_down trend and those exceptions have proved that vaccination is the main but not only factor to successfully control the pandemic.

Therefore, although vaccination plays an irreplaceable role in the age of planetary health, the other factors should be considered carefully in order to have a holistic look.

5.2.2 Vaccination affects the deaths

In the previous section, we already know since vaccines COVID19 began rolling out, they have reduced transmission of the coronavirus. High vaccination rates in countries resulted in a significant reduction in the number of new infections leading to a decrease in the number of deaths as well by comparing their trend in each stage. The 2 charts below indicated the same trend in most countries:

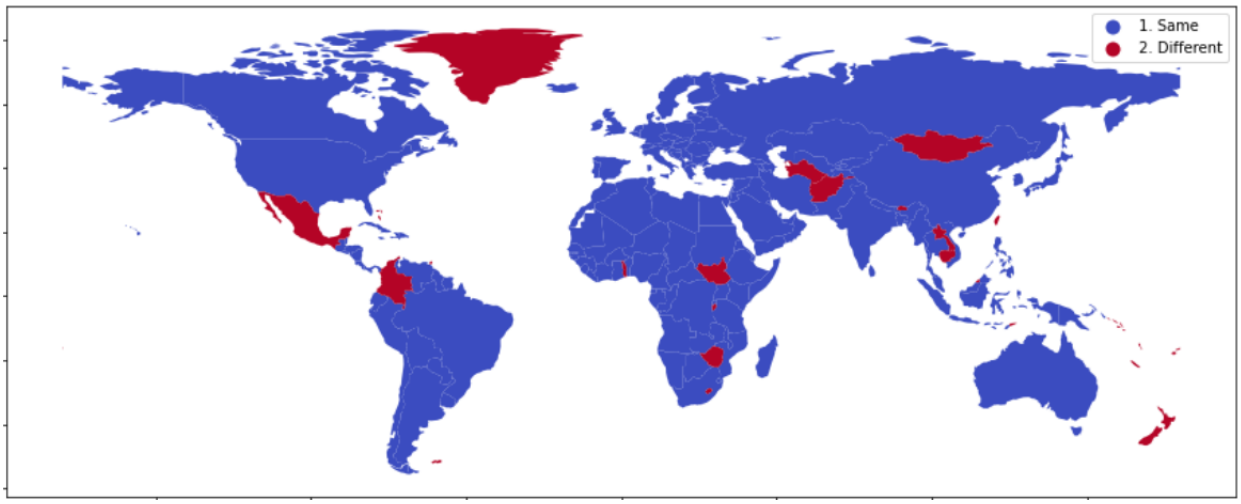


Figure 32: Trend comparison in unvaccinated phase.

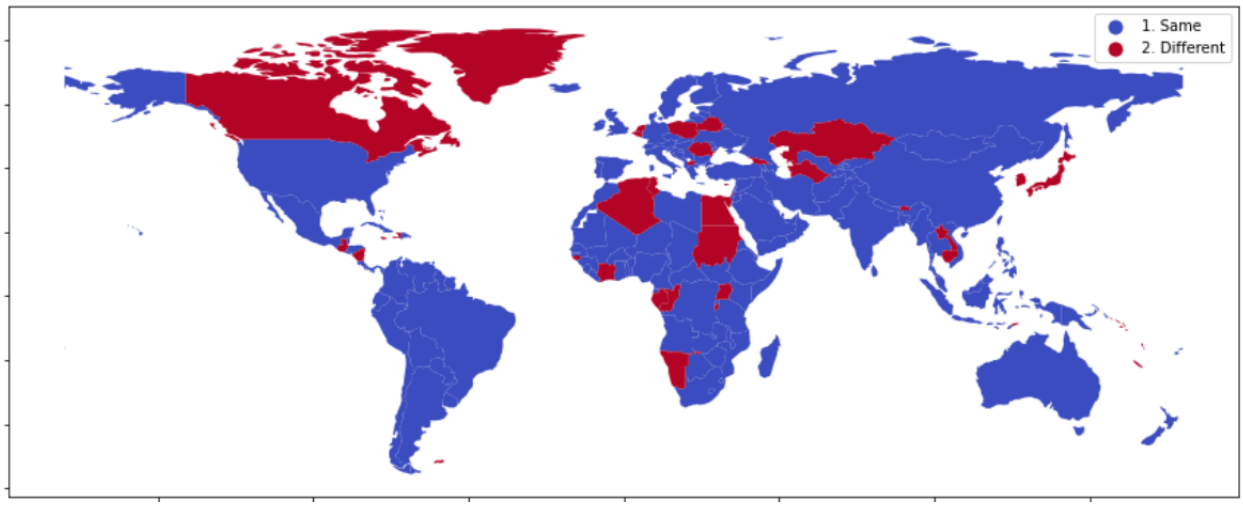


Figure 33: Trend comparison in vaccinated phase.

Vaccines' primary purpose is to prevent you from getting really sick with the viruses. Obviously, not all the vaccines perform 100% effective. So the citizen questioned the need

to receive vaccines if there is still a possibility of contracting the coronavirus. In fact, after inoculation, the chance of contracting the virus is smaller and even significantly lower the risk of severe illness or death.

We outlined the preferred age ranges for average vaccination in 20 countries. The data source only provides information of people from 18 years old to over 80 years old, so the analysis of children under 18 years old will be added when the data source is more comprehensive.

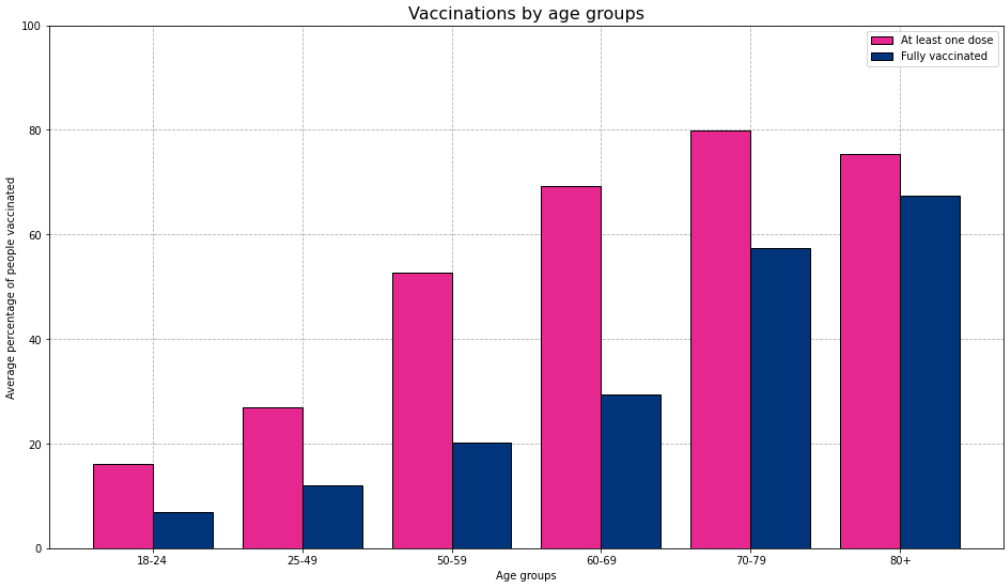


Figure 34: Average percentage of people vaccinated by age groups.

Code of figure 34:

```
age_groups = ["18-24", "25-49", "50-59", "60-69", "70-79", "80+"]
data_vaccinations_by_age = [["18-24", 0], ["25-49", 0], ["50-59", 0], ["60-69", 0], ["70-79", 0], ["80+", 0]]
data_vaccinations_by_age_fully = [["18-24", 0], ["25-49", 0], ["50-59", 0], ["60-69", 0], ["70-79", 0], ["80+", 0]]

for i in range(6):
  data_vaccinations_by_age[i][1] = data_vaccinations_by_age_group[data_vaccinations_by_age_group["age_group"] == age_groups[i]]["people_vaccinated_per_hundred"].mean()
  data_vaccinations_by_age_fully[i][1] = data_vaccinations_by_age_group[data_vaccinations_by_age_group["age_group"] == age_groups[i]]["people_fully_vaccinated_per_hundred"].mean()

data_vaccinations_by_age = pd.DataFrame(data_vaccinations_by_age, columns = ["age_groups", "average"])
data_vaccinations_by_age_fully = pd.DataFrame(data_vaccinations_by_age_fully, columns = ["age_groups", "average"])

n=6
```

```
r = np.arange(n)
width = 0.4

plt.figure(figsize=(16,9))
plt.bar(r, data_vaccinations_by_age["average"], color='#e62790',
        width = width, edgecolor = 'black',
        label='At least one dose', zorder=5)
plt.bar(r + width, data_vaccinations_by_age_fully["average"], color='#00347b',
        width = width, edgecolor = 'black',
        label='Fully vaccinated', zorder=5)

plt.xlabel("Age groups")
plt.ylabel("Average percentage of people vaccinated")
plt.title("Vaccinations by age groups", fontsize=16)
plt.ylim(0, 100)
plt.grid(linestyle='--', zorder=0)
plt.xticks(r + width/2, age_groups)
plt.legend()

plt.show()
```

Most elderly and middle-aged people have higher vaccination rates than adolescents. This priority means the older people are (more background disease when they get older), the more vulnerable they are to the coronavirus. Hence, directly reducing severe disease by giving the vaccine to those most at risk, as elderly, could have a big impact on the overall burden of death and medical facilities.

5.3 Other Analysis

5.3.1 Delta variant of Coronavirus

In the last month of the vaccinated phase, many nations in control stage such as Japan, Viet Nam, India, Malaysia, United Kingdom, etc witnessed an unexpected increase in new cases.

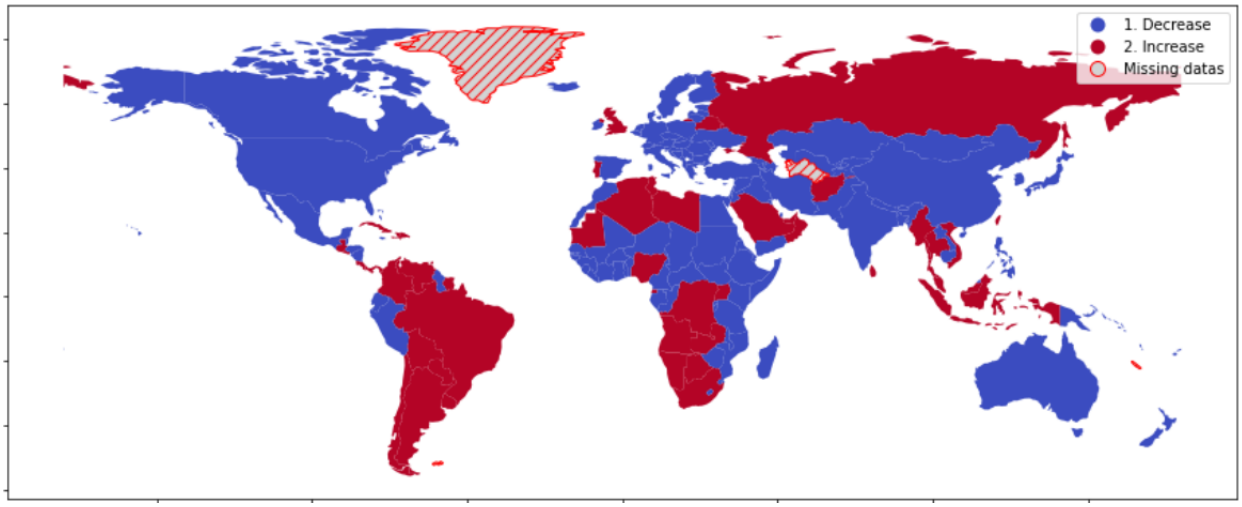


Figure 35: Trend of countries in May.

Those rising trends mainly appear in Latin America, Eastern Europe, West Asia, South Asia, Southeast Asia and some countries in Africa which have to face infection surges because of new variants. The more a virus circulates in a population, the more opportunities it has to transform itself and can reduce the effectiveness of our vaccines. Recently, the Delta variant (known as B.1.617.2 to scientists) was found to represent a majority of active cases in these areas. This variant is more transmissible than any variant (Alpha in England (B.1.1.7), Gamma in Americas (P.1), Beta in Brazil (B.1.351)) detected before and widely spread to other countries. It is predicted to create a new wave of covid in 2021 in countries that are slow to inoculate.

The Delta variant had come to dominate the headlines, however, in late June, a mutation of that variant emerged in India, called Delta Plus. The Indian government labelled it as a "variant of concern (VOC)" and there are fears that it could potentially be more transmittable.

5.3.2 Covid situation in Viet Nam

Since the first case was detected, Vietnam has undergone fierce battle to control the outbreak with stringent measures from the government to mitigate the adverse impacts. As of the end of 2020, Vietnam reported about 1500 confirmed cases and 35 deaths. Vietnam's government has had relative success in controlling the virus in a long period.

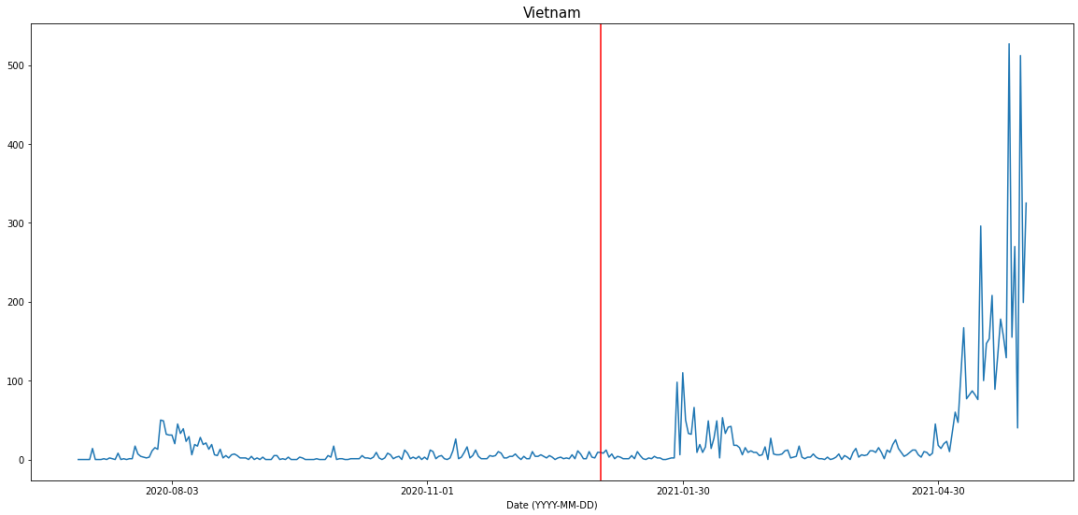


Figure 36: Daily new case of Viet Nam - 2 phases separated by red line.

In May, Viet Nam witnessed a strong outbreak of infections across the northern provinces. According to information, most of the infections appeared in a new variant of Covid-19 which originated from the existing Delta variant, formerly referred to as the Indian variant, causing increased transmission while the inoculation campaign is limited.

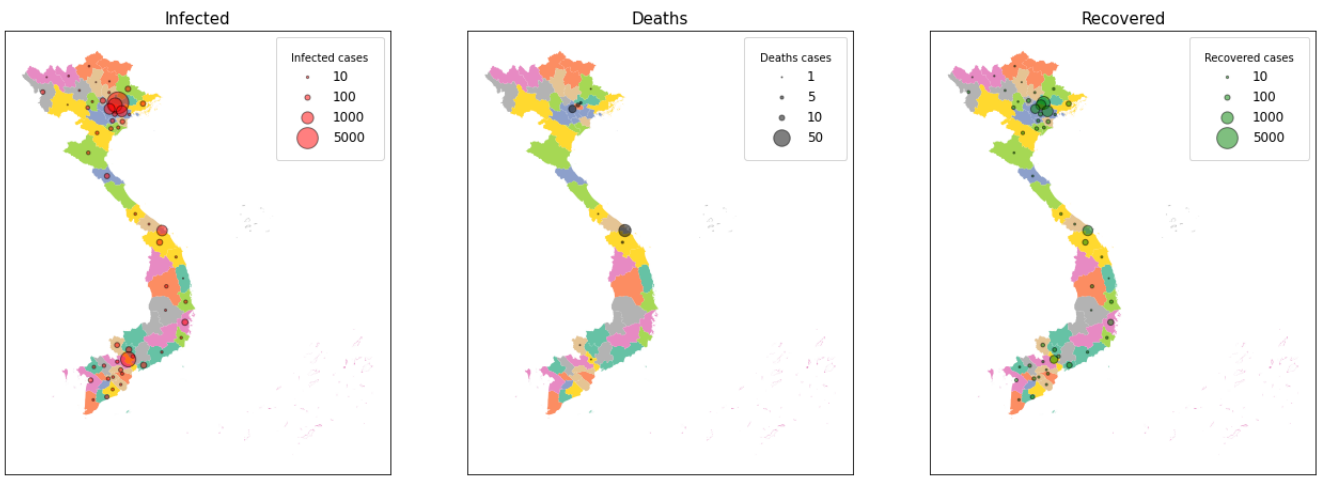
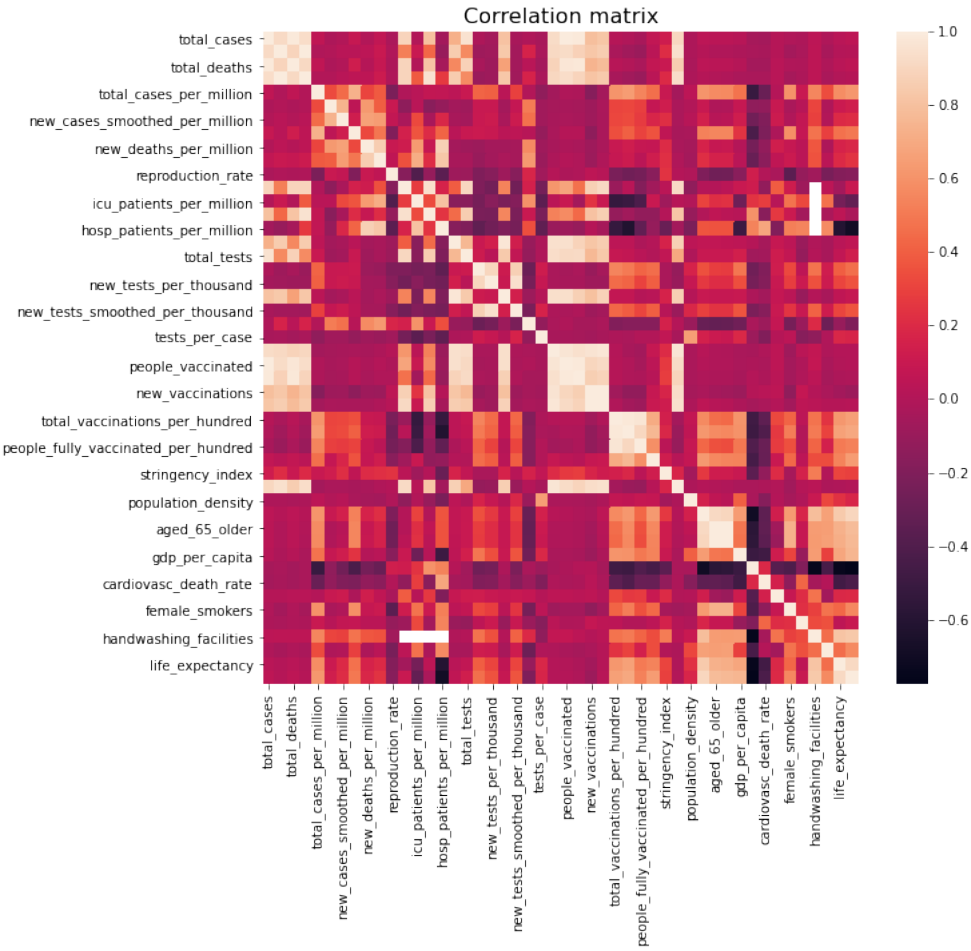


Figure 37: Covid in Viet Nam (total cases until 30/06/2021).

From the northern provinces, the virus quickly spread throughout the country and rapidly rose in densely populated cities (Ho Chi Minh, Ha Noi, Da Nang,...). In the following weeks, the number of new infections is likely to continue to increase until about 70% of the population is vaccinated.

5.3.3 Correlation Matrix



The correlation matrix represents how each aspect of the given data acts as a factor that influences the other aspects in the dataset. The bar chart on the right side of the matrix demonstrates the level of these correlations, where a positive coefficient indicates that both variables move in the same direction, a negative coefficient signifies that the variables move in the opposite direction, and a coefficient close to zero shows that both variables are independent.

For example, the small square at the top left corner illustrates the correlation between the total cases and itself. The matrix tells us that when we compare something to itself, the correlative result is always 1.0 as demonstrated by the white color.

When we have a good understanding of how these factors may interact with each other, it is easier for us to come up with the model prediction step and also model analysing step.

5.4 Theory of Distribution

5.4.1 What is data distribution?

Most of us are familiar with charts like dot plot and histogram in daily life like mean annual temperature histogram, weekly purchase per customer histogram, number of siblings dot plot or even preferred snack choice dot plot. They are approximate representations of the distribution of numerical data and tell us detailedly how frequency things occur in data we actually have. But in many circumstances, the data we have mentioned usually is not all the data that exists. For example, if we want to know information about numbers of kids per family in Viet Nam, we are not definitely going to ask over one hundred millions Vietnamese. Instead, we can ask about ten million people and the shape of the gathered data is a shadow of what all the data look like. We collected those samples because we think they could give us a glimpse of a bigger picture and tell us something about the overall data. Let's check out the following figure to get clearlier:

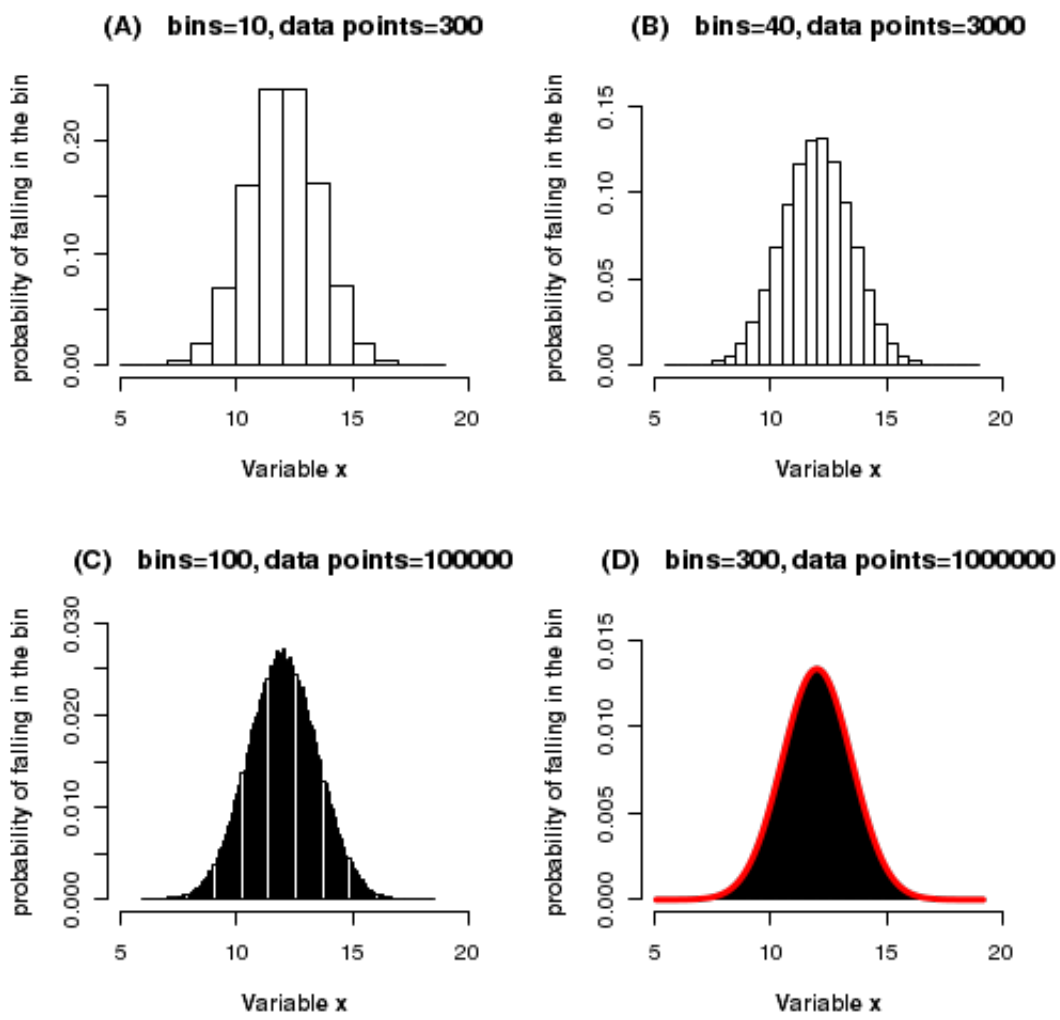


Figure 38: Four plots in an experiment (source: countbio.com)

Imagine we have a histogram of 300 variables x put in 10 bins (figure A). We see that as the number of bins and data points increases (figure B,C), the bin width decreases and the histogram starts looking like a continuous curve (shown as red line in the figure D). From discrete data, it is getting thinner and thinner, then the outline of our histogram looks like a smooth line. It's also known as distribution of data.

Data distributions are used often in statistics. They are graphical methods of organizing and displaying useful information of any datasets. Like the histogram, the distribution tells us about the shape and spread of the data.

5.4.2 What data distribution can do?

In addition to showing the information of the data, the distribution will help us compare the data in a more meaningful way.

Samples of data are not all. However, when we compare two samples we may wonder if the same distribution could have produced two different shapes as the image below ?

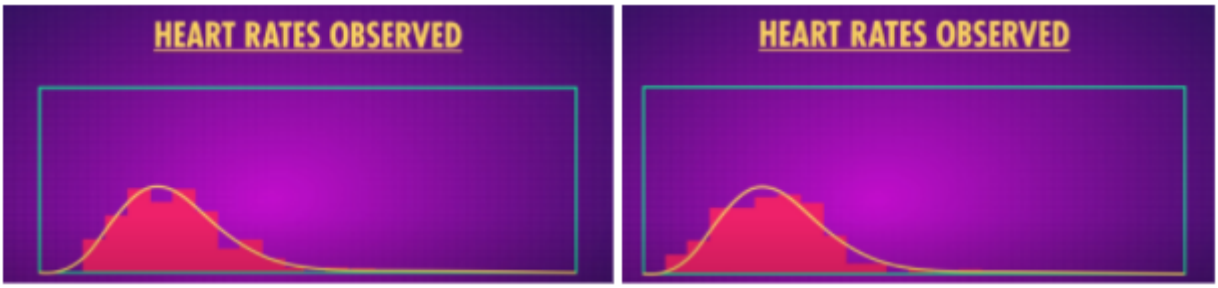


Figure 39: Same distribution with two different histograms (source: CrashCourse)

To answer this question, now think of a real life example, it's monthly electricity bills. If we get an expensive electricity bill this month, we will want to check out histogram of our daily energy consumption this month, and the last month or even the same month last year side by side. It's not realistic we have consumed exactly the same amount of energy, there definitely be some differences. But whenever there's enough difference between them to conclude that our energy consumption has changes in behaviors or not, the distribution of energy helps us compare the general data. That's what it really does.

Back to the question, because we know the samples approximate some theoretical shapes, we could draw connections between samples and other actual data which we will collect in the near future.

While data comes in all sorts of shapes, in this report we can take a look at a few of the most common of them.

5.4.3 Normal Distribution

It is otherwise known as Gaussian Distribution (named for Carl Friedrich Gauss or Symmetric Distribution or The bell curve.)

For a deeper understanding of the Normal distribution, let's get started with these 3 concepts : mean, median, mode. What are they?

Type	Description	Example	Result
Arithmetic mean	*Sum of values of a data set divided by number of values*	(1+2+2+3+4+7+9) / 7	**4**
Median	*Middle value separating the greater and lesser halves of a data set*	1, 2, 2, **3**, 4, 7, 9	**3**
Mode	*Most frequent value in a data set*	1, **2**, **2**, 3, 4, 7, 9	**2**

Figure 40: Mean, Median and Mode definition (source: Wikipedia)

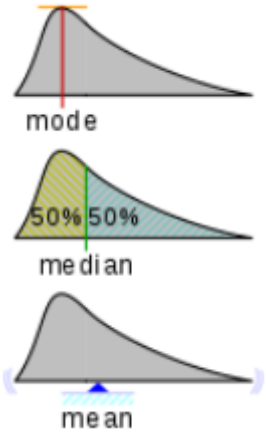


Figure 41: Mean, Median and Mode figure (source: Wikipedia)

Normal distribution's mean, median and mode are the same so the distribution is symmetric and it's unimodal (meaning there's only one peak).

The shape of the Normal Distribution curve is defined by the mean and standard deviation of the sample; the curve will be centered and symmetric around the mean and stretched by the standard deviation. The PDF curve never crosses the x-axis; therefore, it is non-zero across the entire real line. This means that the normal distribution can give you the probability of any event happening, but as it gets farther from the mean, its probability of happening will be closer and closer to zero.

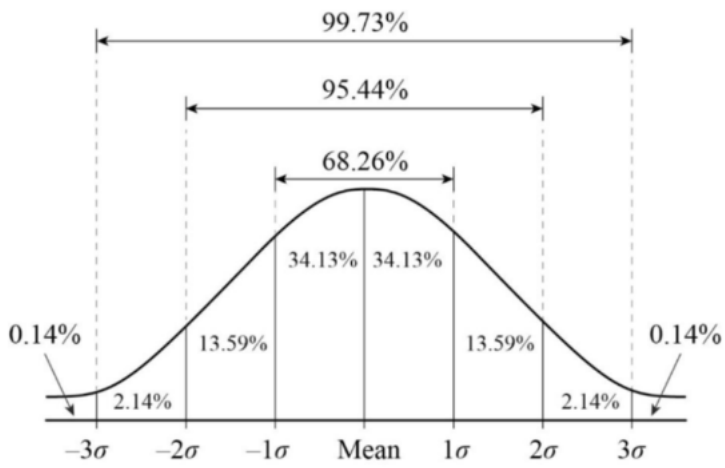


Figure 42: Normal Distribution (source: ml-science.com)

The Empirical Rule (68-95-99.7% rule) states that, in a normal distribution, almost all of the data lies within 3 standard deviations of the mean. This comes very handy when you are trying to identify outliers in your data or even as a way to check the distribution's normality. Fig 4 shows the Empirical Rule and how 99.7% of the data in a Normal Distribution lies within 3 standard deviations.

Formula of normal distribution:

$$f(x|\mu, \delta^2) = \frac{1}{2\sqrt{2\pi\delta^2}} e^{-\frac{(x-\mu)^2}{2\delta^2}} \tag{1}$$

where:

- δ: Standard Deviation.
- μ: Mean (in normal distribution: mean = mode = median).

There is another point that data is not always symmetric, oftentimes it has some extreme values on one side making it a little bit skewed. In probability theory and statistics, skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, zero, negative, or undefined.

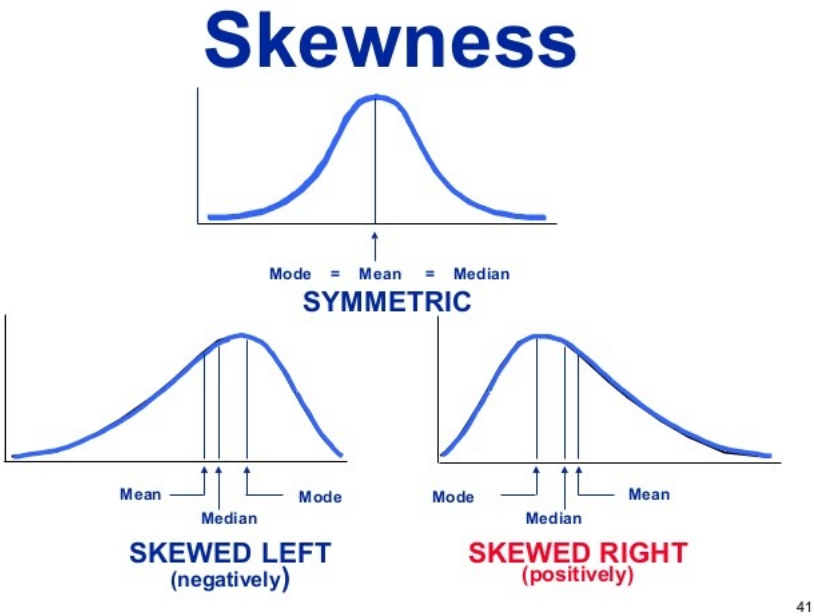


Figure 43: Skewness of Normal distribution (source: opentextbc.ca)

Boxplot of normal distribution:

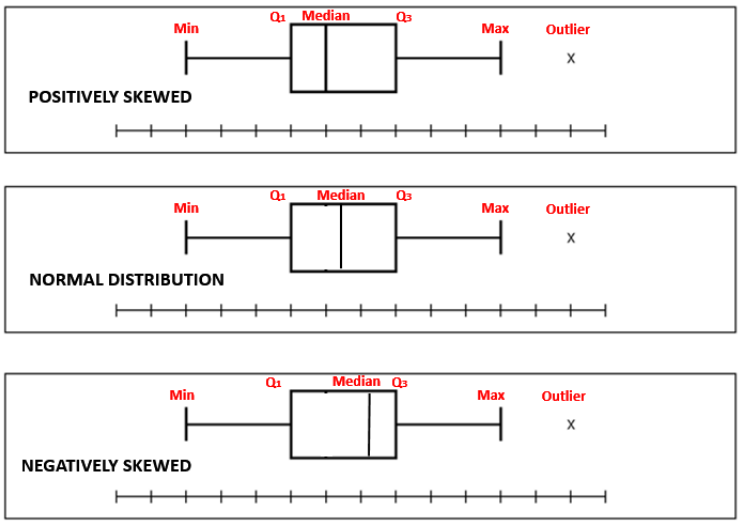


Figure 44: Boxplot of Normal distribution (source: studywalk.com)

Normal distribution comes up a lot when we look at groups of things like IQ or in tests such as SAT. Therefore, it is widely used in statistics, business, government bodies and commonly found in measurement errors, points on a test, salaries, height of people, blood pressure, IQ scores. Skew is also a useful way to compare data. For example, teachers often look at the distribution of test scores to see how difficult they were. Really hard tests tend to generate negative skewed scores, with most students doing poorly then they can evaluate many things gained from the distribution.

5.4.4 Bimodal Distribution

In the previous part, the distribution we have seen so far is unimodal - they have only one peak. However, there are many times data would have two or more peaks so we call it bimodal or multimodal data.

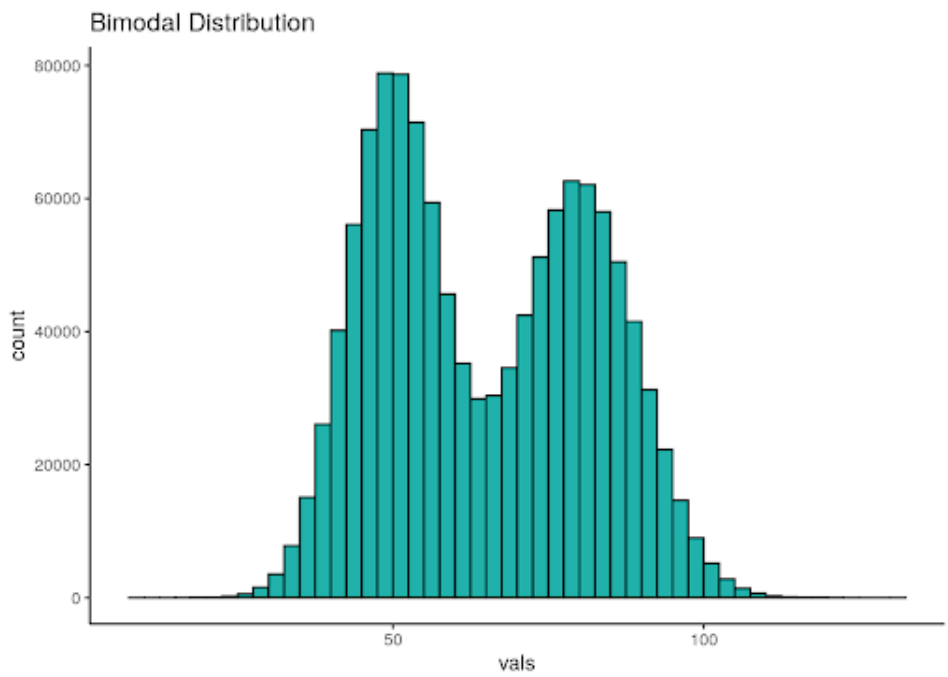


Figure 45: Histogram of bimodel distribution (source: MathBootCamps)

The “bi” in “bimodal” refers to “two” and the model refers to the peaks. . The two peaks in a bimodal distribution also represent two local maximums; these are points where the data points stop increasing and start decreasing.

Often when we could see multimodal data in the world it is because there are two independent trials in an experiment, and each of them has two results (usually known as success and failure). Hence, whenever we generate a bimodal, we get two peaks of data which would indicate we have got two different groups.

Formula of bimodal distribution:

$$P(x) = \binom{n}{x} p^x q^{n-x} = \frac{n!}{(n-x)!x!} p^x q^{n-x} \tag{2}$$

where:

- n: the number of trials (or the number being sampled).
- x: the number of successes desired.
- p: probability of getting a success in one trial.
- $q = 1 - p$: the probability of getting a failure in one trial.

One possible example of this is the scores distribution of a class. Most of them are 30 or 90 then there are two different mechanisms behind the data. These two kinds of score ranges look like two groups of students, one is underprepared for the test for some reason and the other is well prepared (as two results of success and failure we gained from collecting data).

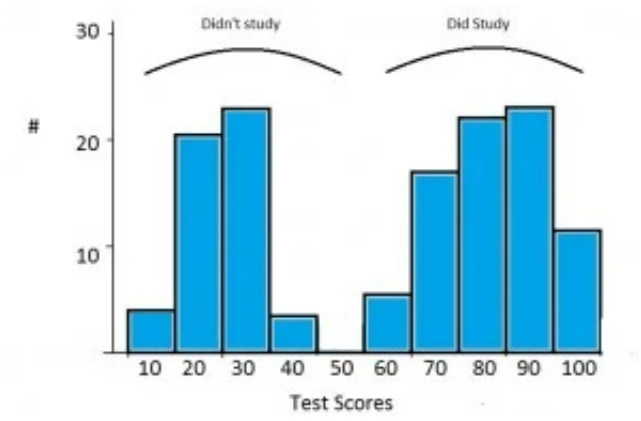


Figure 46: A testing histogram (source: statisticshowto.com)

Sometimes, what appears to be a bimodal distribution is actually two unimodal (one-peaked) distributions graphed on the same axis and being measured at the same times.

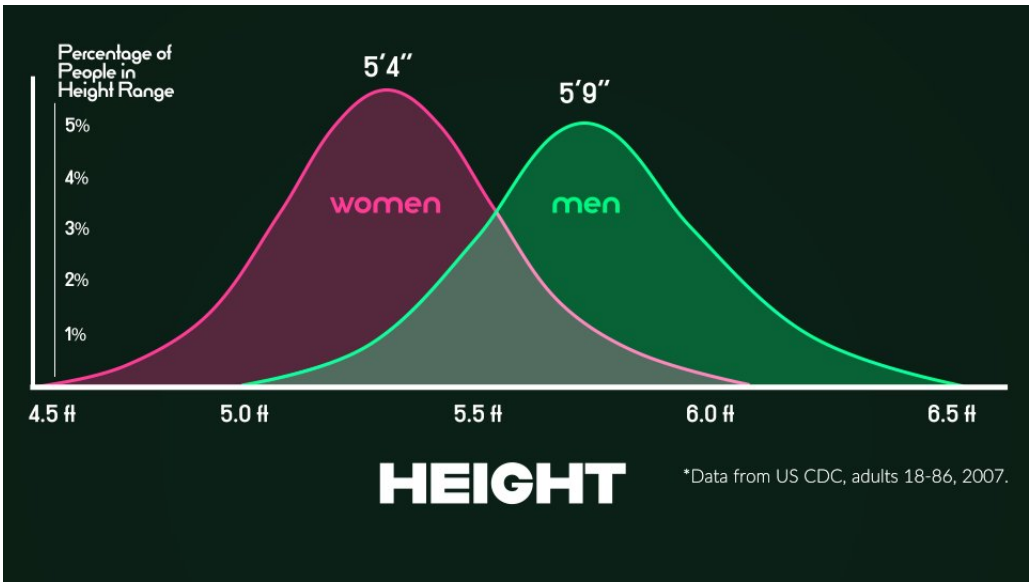


Figure 47: A histogram of a height survey (source: @lecanardnoir)

5.4.5 Poisson Distribution

Poisson distribution is a discrete distribution which describes a number of events occurring in a fixed time interval or region of opportunity. The occurrences in each interval can range from zero to infinity (0 to ∞). Specifically, the rate at which an event occurs is constant and the occurrence of one event does not affect the occurrence of a subsequent event (meaning these two events are independent).

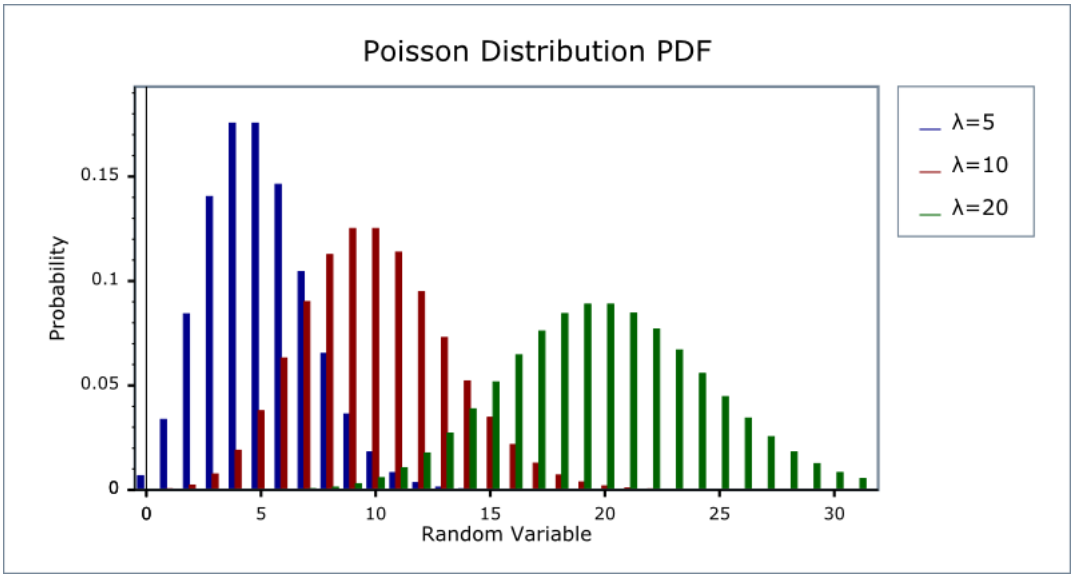


Figure 48: Poisson distribution (source: boots.org)

Formula of Poisson distribution:

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{3}$$

where:

- X: 0, 1, 2, 3...
- λ: mean number of occurences in the interval
- e: Euler's constant (≈ 2.71828).

Because the Poisson distribution measures the frequency of events under the assumption of statistical randomness, the agreement of the expected distribution with the actual data suggests that the actual data was indeed due to randomness.

For example, if an office averages 12 calls per hour, they can calculate that the probability of receiving at least 20 calls in an hour is:

$$\sum_{k=20}^{\infty} \frac{12^k e^{-12}}{k!} \approx 2.12\%$$

which means they can generally feel comfortable keeping only enough staff on hand to handle 20 calls. There are other excitingly practical applications of Poisson distribution such as the number of bankruptcies that are filed in a month, the number of arrivals at a car wash in one hour, the number of births, deaths, marriages, divorces, suicides, and homicides over a given period of time, etc.

5.4.6 Gamma Distribution

It deals with continuous variables which take on a wide range of values. Based on which we can model probabilities across any range of possible values using a gamma distribution function. First one is the shape parameter (α) and the second one is the scale parameter (β).

Formula of gamma distribution:

If X is a continuous random variable then the probability distribution function is:

$$f(n) = \begin{cases} \frac{1}{\beta\Gamma(\alpha)} x^{\alpha-1} e^{\frac{-x}{\beta}} & \text{if } x \geqslant 0 \\ 0 & \text{if otherwise} \end{cases}$$

where:

- $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$: the Gamma function.
- α: the shape parameter.
- β (sometimes θ is used instead): the rate parameter (the reciprocal of the scale parameter).

α and β are both greater than 1.
When $\alpha = 1$, this becomes the exponential distribution.
When $\beta = 1$ this becomes the standard gamma distribution.

The Erlang distribution, and chi-square distribution are also special cases of the gamma distribution

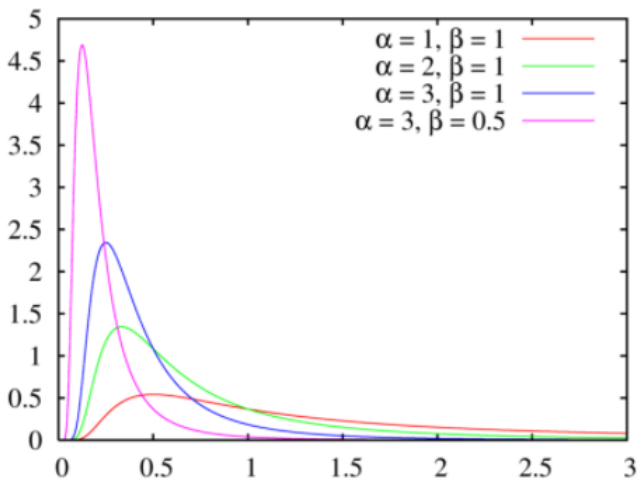


Figure 49: Gamma distribution (source: statisticshowto.com)

If we think of α as the number of events we are waiting for (although α can be any positive number,not just integers), and β as the mean waiting time until the first event. If α (number of events) remains unchanged but β (mean time between events) increases, it makes sense that the graph shifts to the right as waiting times will lengthen. Similarly, if the mean waiting time (β) stays the same but the number of events (α) increases, the graph will also shift to the right. As α approaches infinity, the gamma closely matches the normal distribution. Therefore, gamma distribution application is the amount of rainfall accumulated in a reservoir, the load on web servers, etc.

5.4.7 Exponential Distribution

As mentioned above, exponential distribution is a particular case of gamma distribution. Exponential distributions are commonly used in calculations of product reliability, or the length of time a product lasts.

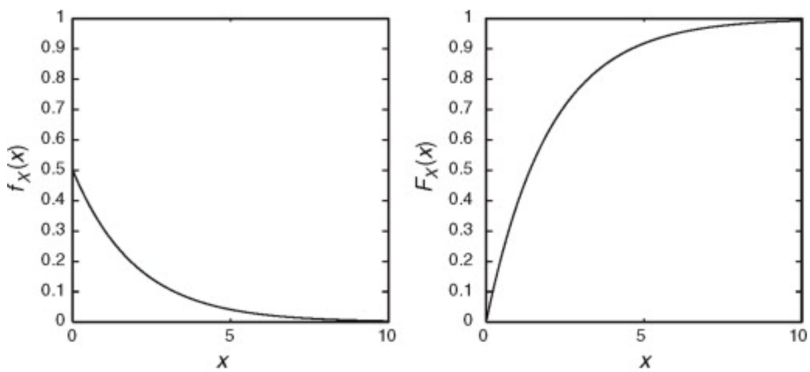
Formula of exponential distribution:

The exponential random variable has a probability density function and cumulative distribution function given for any b > 0 by:

$$f_X(x) = \frac{1}{b} exp\left(\frac{-x}{b}\right) u(x)$$

$$f_X(x) = \left[1 - exp\left(\frac{-x}{b}\right)\right] u(x)$$

A plot of its PDF and CDF (respectively) are shown below :



The exponential distribution is often concerned with the amount of time until some specific event occurs. For example, the amount of time (beginning now) until an earthquake occurs has an exponential distribution. Other examples include the length of time, in minutes, of long distance business telephone calls, and the amount of time, in months, a car battery lasts, etc.

5.5 Distribution Analysis

5.5.1 Distribution of new infections

In this section, we plot the histograms of new confirmed cases and check out their distribution.

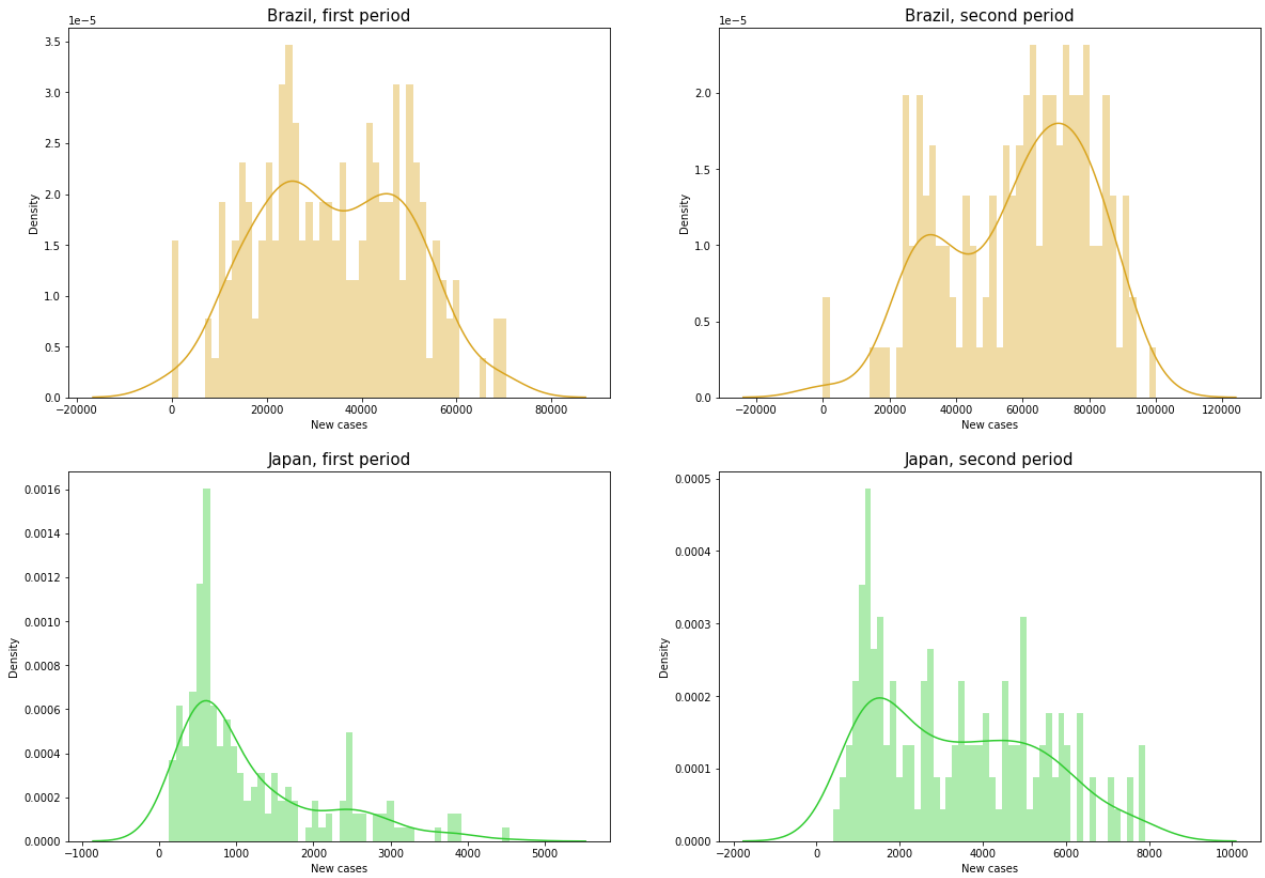


Figure 50: Increase in new cases in second phase.

To the rising trend group, the data followed by bimodal distribution which can be observed clearlier in Brazil's histogram than Japan's. The first mode represents the number of infections commonly seen in a country and the second mode when there is at least one outbreak of covid (often called "waves"). The more modes there are (multimodal distribution), the more likely there will be more waves. Besides, in the second phase, more days with a high number of cases detected.

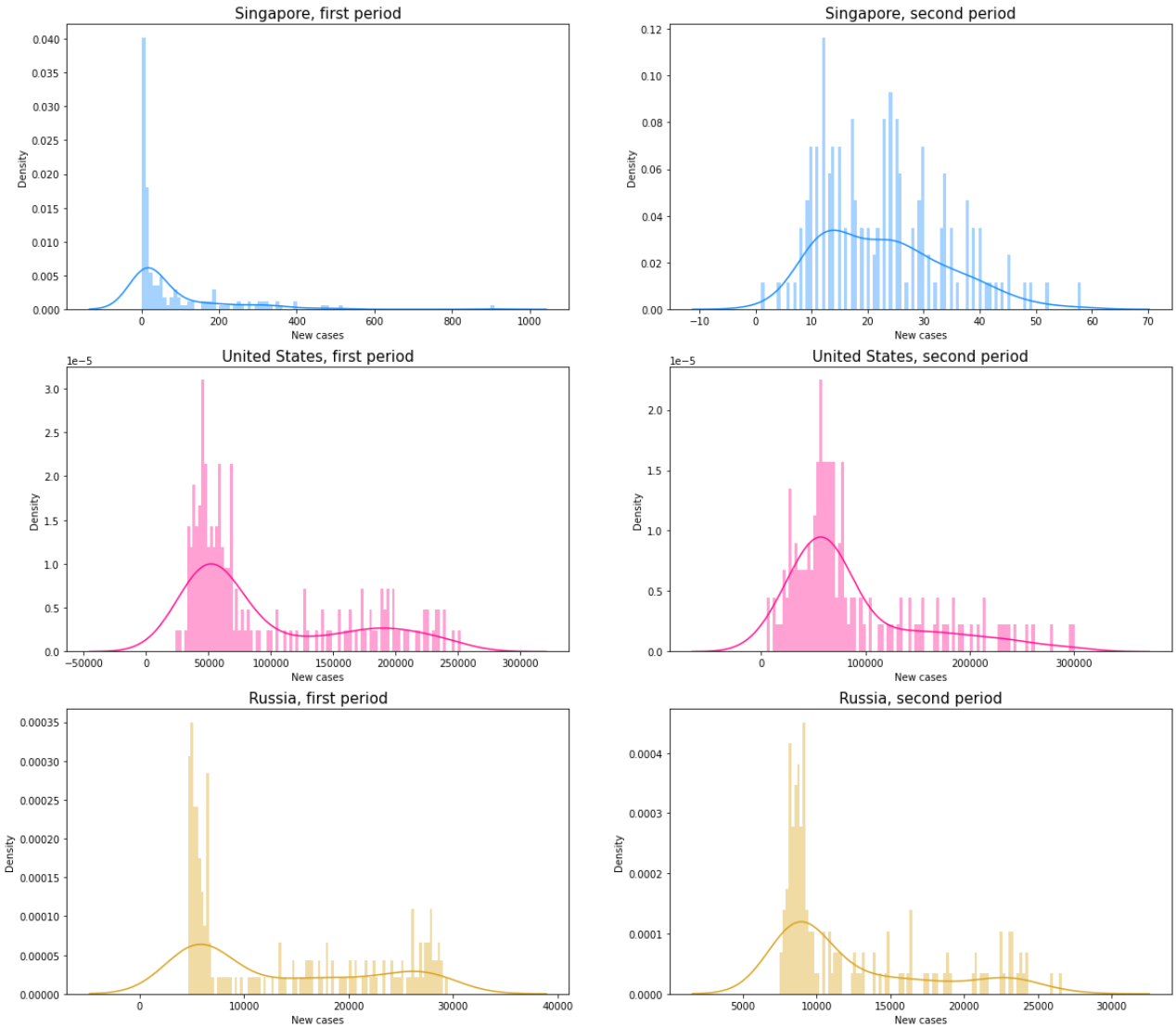


Figure 51: Decrease in new cases in second phase.

=> All in all, the distribution of the COVID-19 new cases does not belong to any particular distribution. However, if we divide into 2 periods (July - Dec 2020 and Jan - June 2021), then the distribution of some countries might look similar to the bimodal distribution. Code of figure 52:

```
fig, axs = plt.subplots(3, 2, figsize = (20, 18))

colors = ["dodgerblue", "deeppink", "goldenrod", "limegreen"]
inspecting_countries = ["Singapore", "United States", "Russia"]
periods = [[start_date, "2020-12-31"], ["2021-01-01", end_date]]

for row in range(3):
  for col in range(2):
    sns.distplot(data_countries[data_countries["location"] ==
   inspecting_countries[row]]["new_cases"].loc[periods[col][0]:periods[
   col][1]], color=colors[row], label="Compact", ax=axs[row, col], bins
   =100)
```

```
    axs[row, col].set_title(inspecting_countries[row] + (", first " if col == 0 else ", second ") + "period", fontsize = 15)
    axs[row, col].set(xlabel='New cases')
```

In general, infections tend to level off or decrease at the second stage in countries with high vaccination rates. The frequency of data starts to appear more in the first mode, the data distribution from bimodal through each stage is similar to the right skewed distribution.

5.5.2 Distribution of vaccinated age

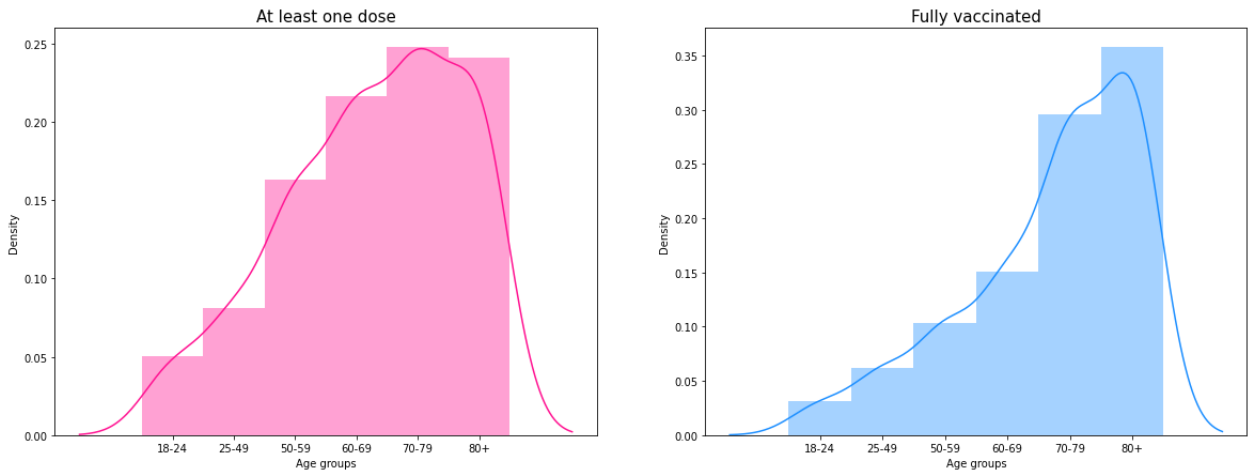


Figure 52: Histogram of distribution by vaccinated age.

Left skewed distribution of both “fully vaccinated” group and “at least one dose” group.

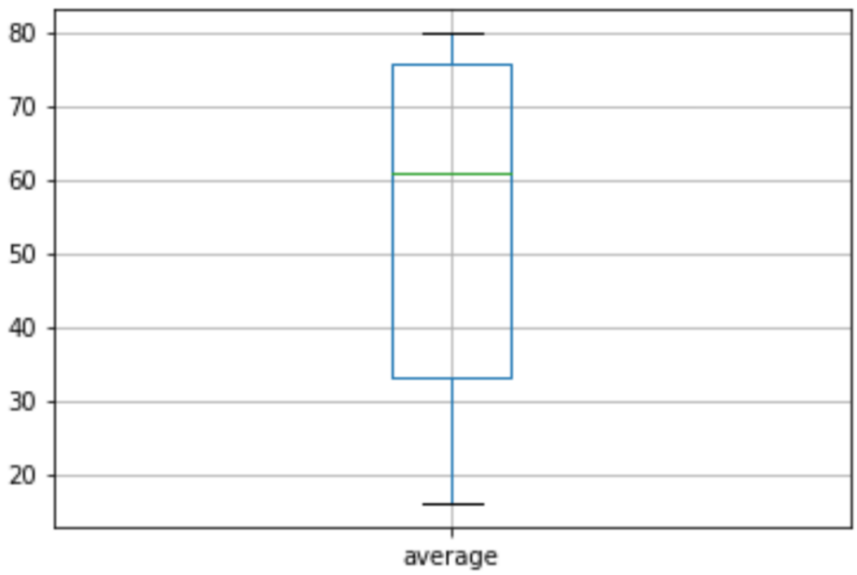


Figure 53: Boxplot of average inoculation age.

The median is in the “60-69" age group.

5.5.3 Distribution of new infections affected by Delta variant

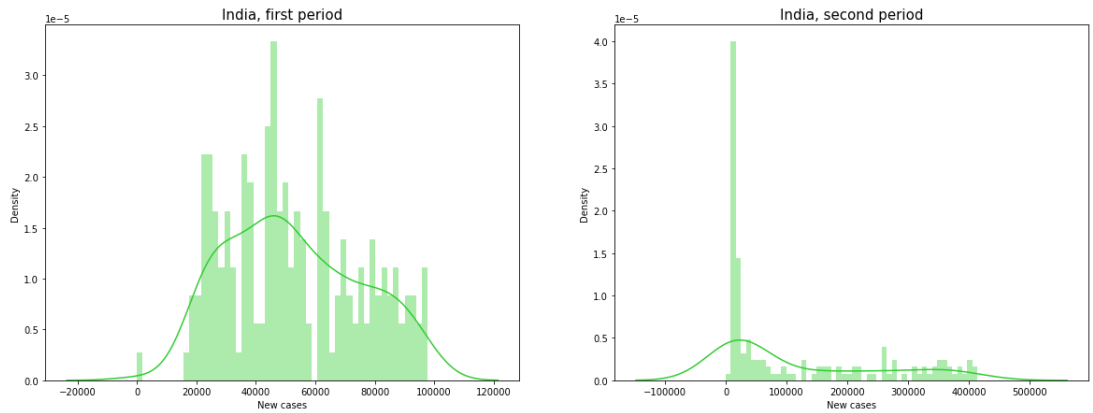


Figure 54: New waves causing by Delta variant in first detected nation.

The epidemic situation in India was already in a serious condition, after the new strain was found, within a few months the epidemic became out of control with the number of infections spiked in the second phase. In the first period, the right skewed distribution seem to have a short tail but it is much longer in the second phase.

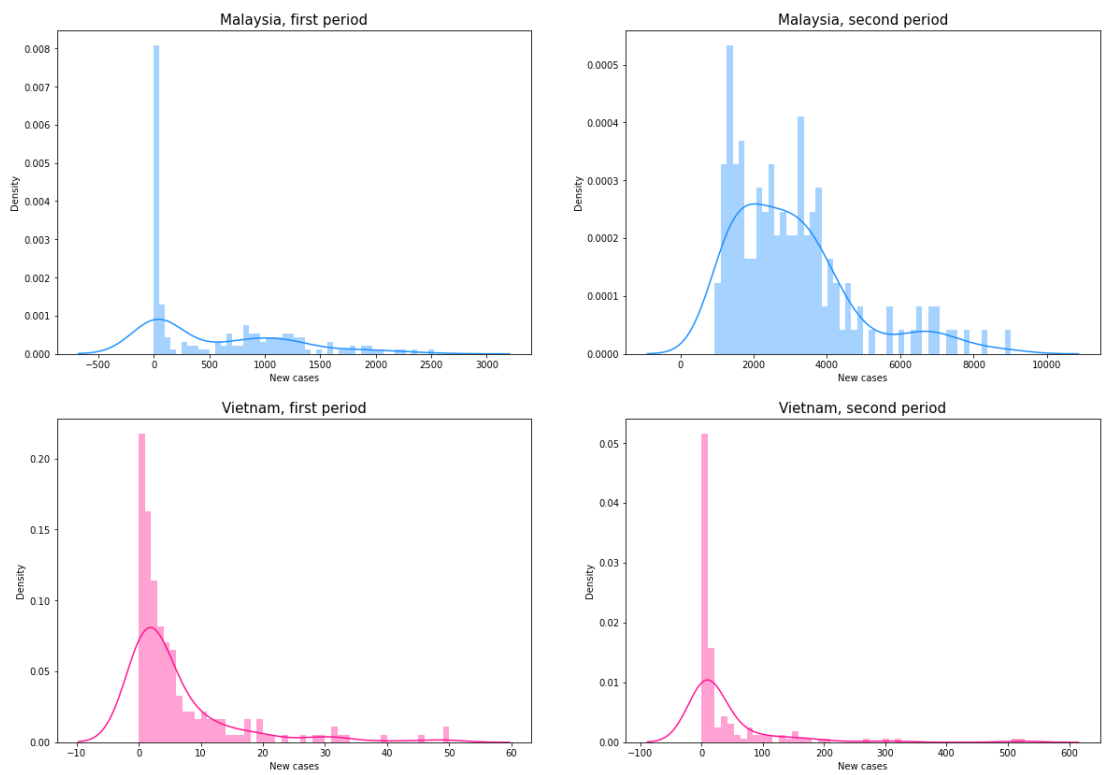


Figure 55: New waves causing by Delta variant in South East Asia area.

The two Southeast Asian countries seriously affected by the new strain are Vietnam and Malaysia. Although the vaccination rate in this area is not high, the number of infections is still in a controllable state before the delta strain enters. The mean values of these distributions become progressively larger over the 2 periods.

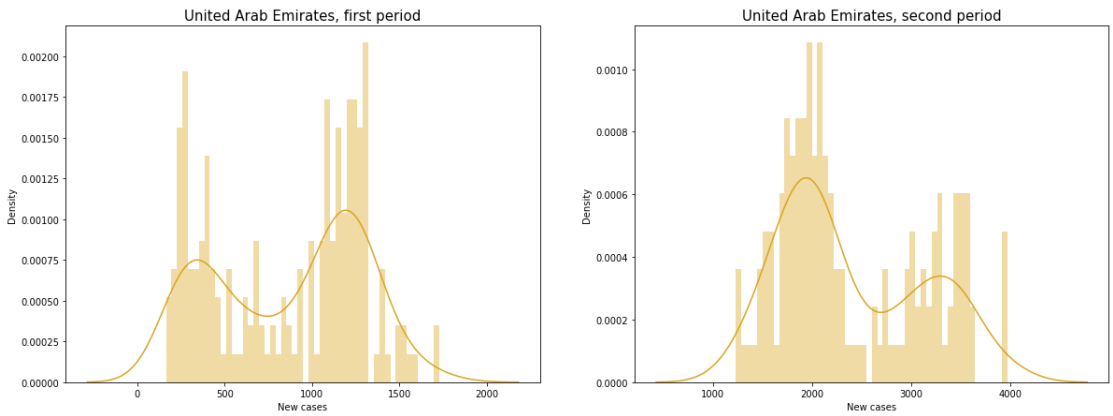


Figure 56: New waves causing by Delta variant in high rate vaccinated nation.

The United Arab Emirates currently has the highest vaccination rate with a low population and high GDP per capita also inevitably increasing the number of infections due to delta mutations, but the rate of increase is slow in this country which demonstrates the obvious effectiveness of vaccination against different strains and a slow in transmission of the virus means fewer opportunities for it to mutate. Both periods are represented by bimodal distribution which mode and tail values between them without the substantial difference like India or South East Asia.

6 Prediction Models

In this chapter, we will build some models to predict the cases of the COVID-19 pandemic. Our models can be classified into three sub-categories:

- Supervised Learning(Linear Regression, Support Vector Machine, Decision Tree) from 6.1 to 6.5
- Time Series (ARIMA, Holt-Winters) from 6.6 to 6.11
- SIR from 6.12 to 6.14

6.1 Linear regression algorithm

6.1.1 Regression summary

- **What is Regression?**
  - Regression is a statistical technique used in the field of data analysis and experimental modeling, allowing the newly discovered regression model to be used for the purpose of predicting, controlling, and learning the creation mechanism data
  - Regression model allow us to estimate relationships between variables including one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).
- **Regression equation**

$$Y = f(X, \theta)$$

  - Y: the variable that you are trying to predict (dependent variable).
  - X: the variable that you are using to predict Y (independent variable).
  - θ : regression coefficient (describes the relative influence of X on Y).

6.1.2 Linear regression and its algorithm

- Linear regression is a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).
- Linear regression algorithm is a supervised learning algorithm. This algorithm is sometimes referred to as Linear Fitting (in statistics) or Linear Least Square:
  - Hypothesis function
    * This is a mathematical function that helps us to convert the value from x to y so that the predicted value of y is correct or approximate to the actual value of the data samples we have collected.
    * The formula of the hypothesis function:

$$h_\theta(x) = \theta_1 \times x + \theta_0$$

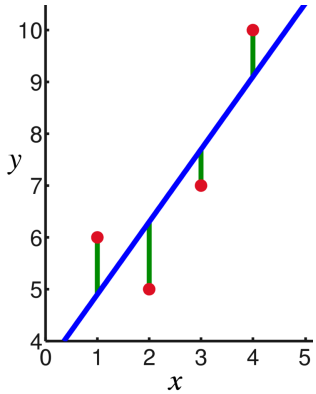


Figure 57: Linear regression model

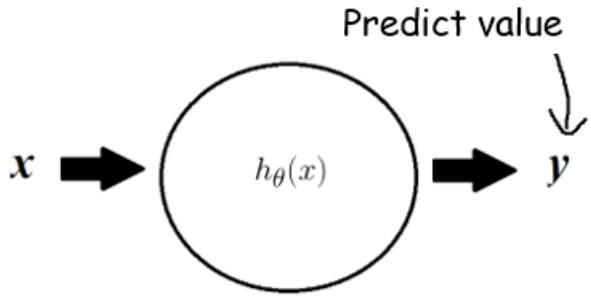


Figure 58: Hypothesis function role

- $h_\theta(x)$: Hypothesis function
- x: independent variable
- θ_1, θ_0: coefficients

* The Hypothesis function can be extended with even more weights:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

– Cost function

* After we have the formula of our predictive function, our next job is to find out the weights θ_1 and θ_0 to be able to draw the line that is as close to all the points on our graph as possible (the distance from the line to the points is minimal) to minimize the error.
* In Machine Learning, Cost Function is a name for a function used to accurately measure the level of prediction, which helps to find the important numerical optimizations for our function prediction. This function is defined as follows:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

- m: number of data samples collected
- x: input value (the facts that we use to make predictions)

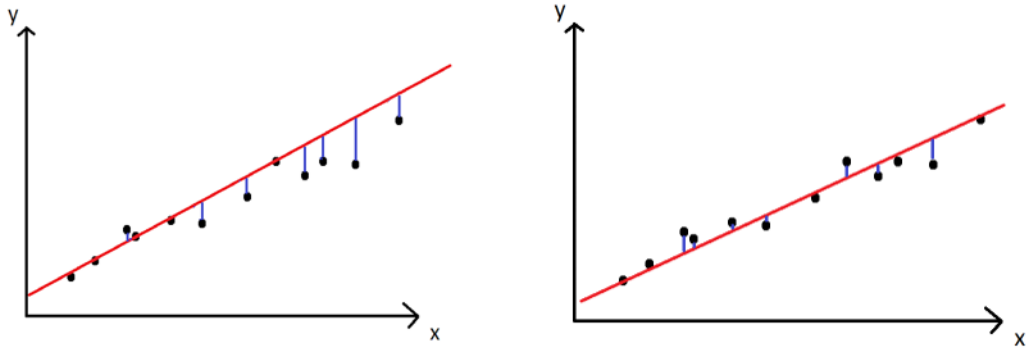


Figure 59: minimize errors

- y: the real value obtained in the data
- $\left(x^{(i)}\right)$: the i(th) input value in the data samples
- $\left(y^{(i)}\right)$: the i(th) real value in the data samples
- (x, y): a complete record of the data we collect

* The term $h_\theta\left(x^{(i)}\right) - y^{(i)}$ is called error term or residual analysis (in statistics). This term represents the distance between the line and the points in the graph.

– Gradient descent technique:

* Gradient descent technique purpose: Finding the optimal point (minimize J(θ_0, θ_1))
* In general:
  - Start with random values (θ_0, θ_1), ex. (θ_0= 0, θ_1= 0)
  - Change the set (θ_0, θ_1) to decrease J (θ_0, θ_1)
  - Repeat the above step until we "believe" J (θ_0, θ_1) is small most (minimum)

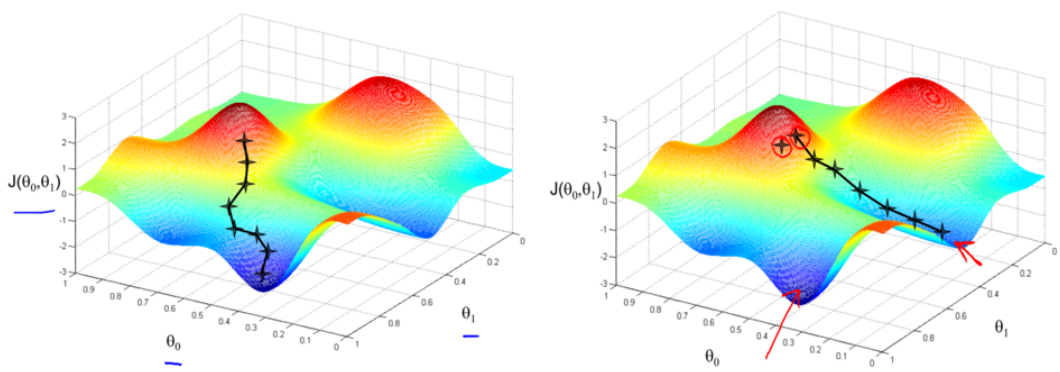


Figure 60: Gradient descent technique simulation

– Gradient descent algorithm:

∗ Example code

Repeat until convergence{

$$\theta_j = \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{// for j=0 and j=1, simultaneously}$$

}

Learning rate

∗ Result

True: update simultaneously	Fall:
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$	$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$	$\theta_0 := \text{temp0}$
$\theta_0 := \text{temp0}$	$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_1 := \text{temp1}$	$\theta_1 := \text{temp1}$

∗ Minimize $J(\theta_0, \theta_1))$

$$J(\theta_0, \theta_1) = \frac{1}{2N} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

Repeat until convergence{

$$\theta_0 = \theta_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

}

Update θ_0 and θ_1 simultaneously

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

6.2 Linear Regression Model

6.2.1 Vietnam

```
df_new = pd.read_csv("https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv", parse_dates=['date'], index_col='date')
df_new.tail()
```

	iso_code	continent	location	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_per_million
date											
2021-06-30	ZWE	Africa	Zimbabwe	49864.0	1331.0	912.000	1789.0	28.0	13.857	3354.925	89.552
2021-07-01	ZWE	Africa	Zimbabwe	51221.0	1357.0	987.857	1808.0	19.0	14.143	3446.226	91.301
2021-07-02	ZWE	Africa	Zimbabwe	52663.0	1442.0	1063.714	1841.0	33.0	17.143	3543.246	97.020
2021-07-03	ZWE	Africa	Zimbabwe	53665.0	1002.0	1092.429	1859.0	18.0	19.143	3610.662	67.416
2021-07-04	ZWE	Africa	Zimbabwe	54474.0	809.0	1147.429	1878.0	19.0	20.286	3665.092	54.431

```
df = df_new[["new_cases_smoothed"]].where(df_new['location']=="Vietnam").dropna()
df
```

```
start_time = '2021-04-15'
end_time = '2021-06-15'
train = df.loc[start_time:end_time]  # train & test split
test = df.loc[end_time:]
```

```
Y_train = np.array(train.values)
Y_test = np.array(test.values)
X_train = []
X_test = []
index = 0
while(index < train.shape[0]):
  X_train.append(index)
  index += 1

while(index < test.shape[0]+train.shape[0]):
  X_test.append(index)
  index += 1
```

```
x0_train = np.array(X_train).reshape(-1,1)
x0_test = np.array(X_test).reshape(-1,1)
y0_train = np.array(Y_train).reshape(-1,1)
y0_test = np.array(Y_test).reshape(-1,1)
```

```
model = LinearRegression()
model.fit(x0_train,y0_train)
y_predict = model.predict(x0_test)
```

```
plt.figure(figsize=(15,10))
plt.title("Predict Covid-19 with Linear Regression")
plt.scatter(test.index,y_predict, label='Prediction')
```

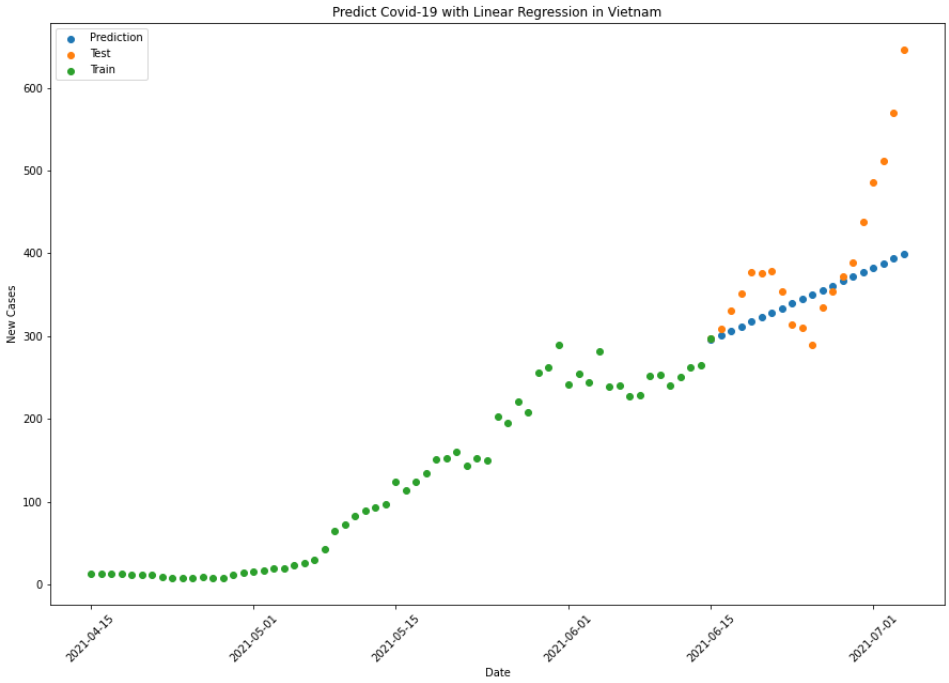
```
[8] print(model.score(x0_train, y0_train))
    print(model.score(x0_test, y0_test))
    print(model.score(y_predict, y0_test))
```

```
0.9166733643126905
0.19534160250202684
-247.79808922349767
```

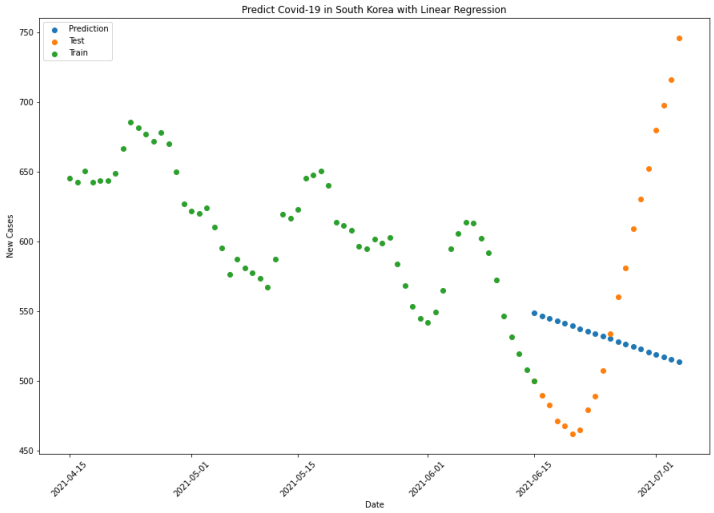
```
[9] print(np.sqrt(mean_squared_error(y_predict, y0_test)))
    print(mean_squared_error(y_predict, y0_test))
```

```
83.38327845398885
6952.771125735441
```

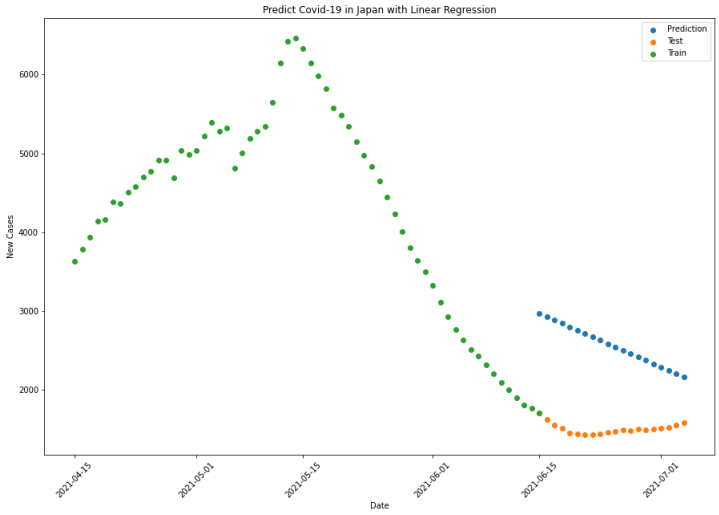
```
plt.scatter(test.index,y0_test, label='Test')
plt.scatter(train.index,y0_train, label='Train')
plt.xlabel('Date')
plt.ylabel('New Cases')
plt.legend(loc="best")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



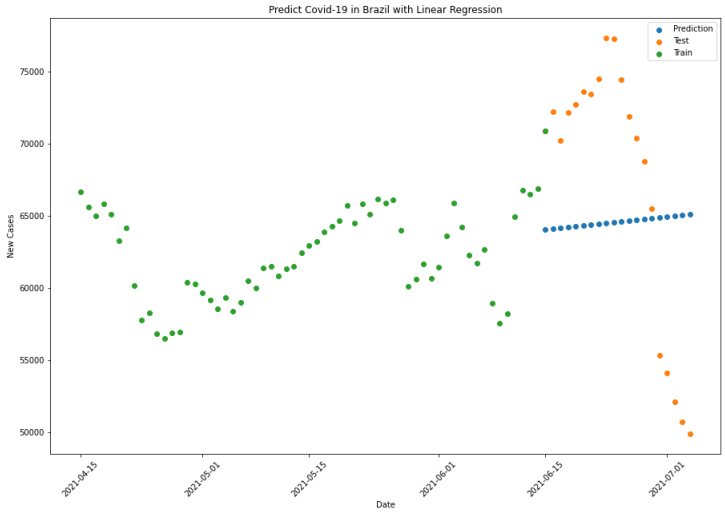
6.2.2 Korea



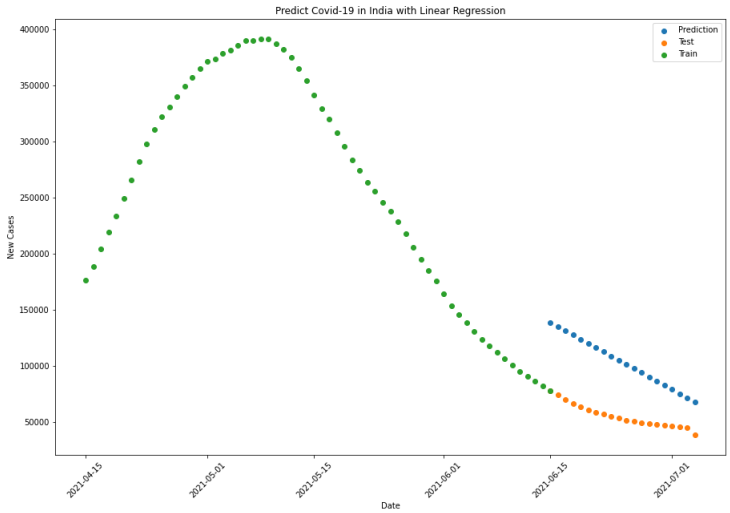
6.2.3 Japan



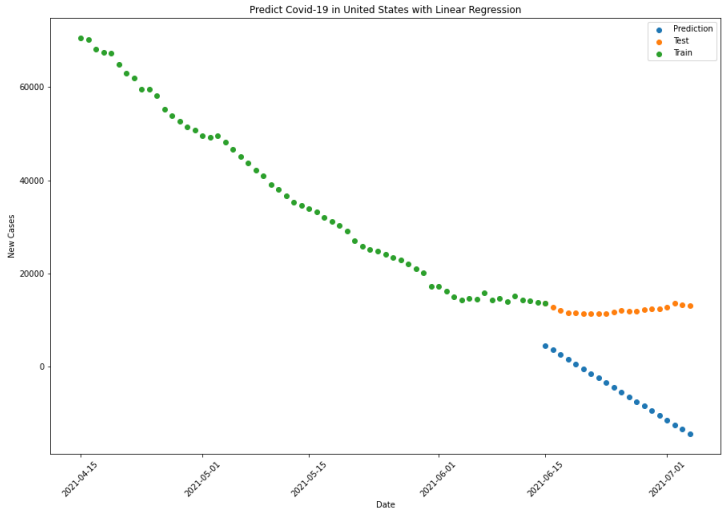
6.2.4 Brazil



6.2.5 India



6.2.6 USA



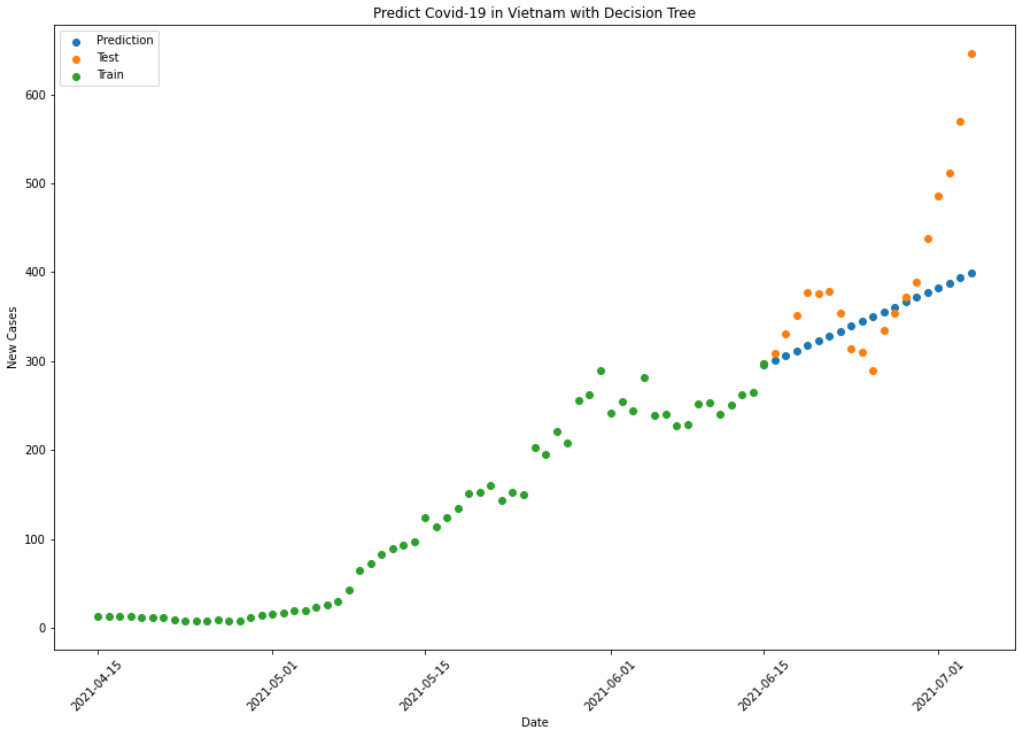
6.3 Decision Tree Model

```
dtr = RandomForestRegressor()
model_dtr = dtr.fit(x0_train, y0_train)
y_dtr_pred = model.predict(x0_test)
y_dtr_pred
```

```
df_cpare_dtr = pd.DataFrame({'Actual': pd.DataFrame(y0_test)[0].values,'Prediction': pd.DataFrame(y_dtr_pred)[0].values})
df_cpare_dtr.head()
```

```
plt.figure(figsize=(15,10))
plt.title("Predict Covid-19 with Random Forest")
plt.scatter(test.index,y_dtr_pred, label='Prediction')
plt.scatter(test.index,y0_test, label='Test')
plt.scatter(train.index,y0_train, label='Train')
plt.xlabel('Date')
plt.ylabel('New Cases')
plt.legend(loc="best")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

=> The Decision Tree model in this case is similar to the Linear Regression Model. Hence, photography repetition should be avoided in this situation.



6.4 Support Vector Machine

6.4.1 What is Support Vector Machine (SVM)?

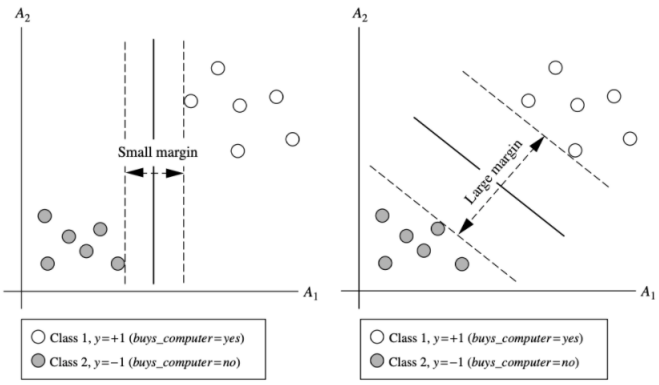
- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

- To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes.

  => An SVM approaches this problem by searching for the maximum marginal hyperplane.

What is the margin?

- Getting to an informal definition of margin, we can say that the shortest distance from a hyperplane to on side of its margin is equal to the shortest distance from the hyperplane to the other side of its margin, where the "sides" of the margin are parallel to the the hyperplane. **The margin is determined by:** margin = $\min_n \frac{y_n\left(\mathbf{w}^T\mathbf{x}_n+b\right)}{\|\mathbf{w}\|_2}$



Mathematical base

- The distance from a point to a hyperplane
  - In the 2-dimensional space, we know that the distance from a point with the coordinates
    $$(x_0, y_0)$$
    to the line whose equation
    $$w_1x + w_2y + b = 0$$
    is determined by
    $$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$
  - In the 3-dimensional space, the distance from a point with the coordinates
    $$(x_0, y_0, z_0)$$
    to the plane whose equation
    $$w_1x + w_2y + w_3z + b = 0$$
    is determined by
    $$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$
  - If we omit the absolute value sign on the numerator, we can determine which side of the line or plane it is on. The points that make the expression in the absolute sign positive lie on the same side (I temporarily call this the positive side of the line), the points that make the expression in the absolute sign the negative sign lie on the other side (I call the negative side).The points on the line / plane will cause the numerator equals zero, or the distance equals 0.

    This can be generalized to multidimensional space: The distance from a point (vector) with coordinates
    $$x_0$$

to a hyperplane with equation

$$\mathbf{w}^T\mathbf{x} + b = 0$$

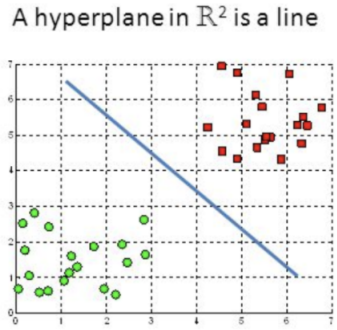
is determined by:

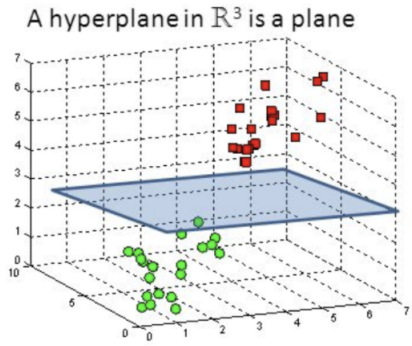
$$\frac{\left|\mathbf{w}^T\mathbf{x}_0 + b\right|}{\|\mathbf{w}\|_2}$$

with

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$$

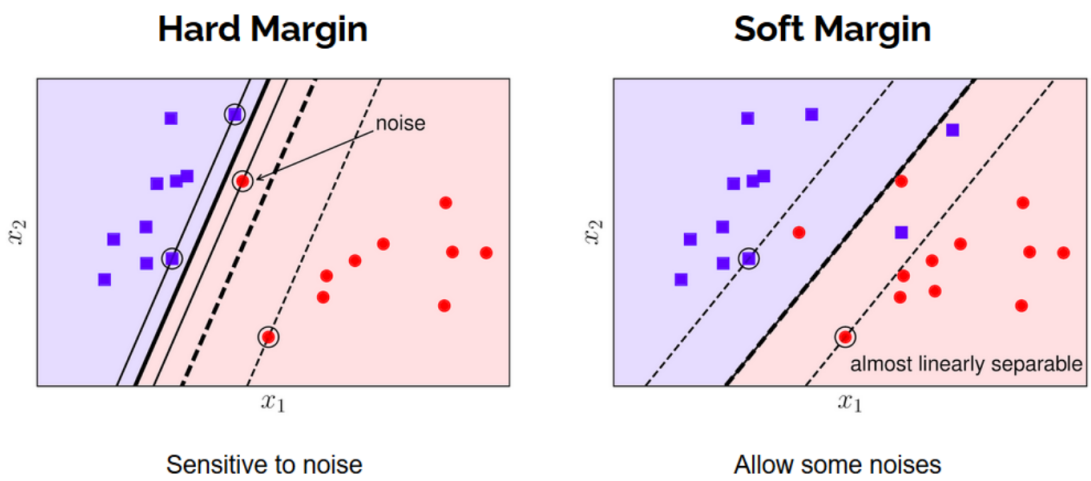
with d is the number of dimensions of the space.





So, hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

6.4.2 Soft-margin SVM



- There are two easy cases where SVM is not working effectively or even not:
  - Case 1: The data is still linearly separable as shown in the image on the left but there is a red noise point too close to the blue class. In this case, if we use the original SVM it will generate a very small margin. In addition, the decision boundary is too close to the blue class and away from the "real" red class. Whereas if we sacrifice this noise point we get a much better margin described by the dashed lines. Therefore, the original SVM is also considered sensitive to noise.
  - Case 2: Data is not linearly separable but almost linearly separable as the right image. In this case, if we use the original SVM then it is clear that the optimization problem is infeasible, ie the feasible set is an empty set, so the SVM optimization problem becomes null. However, if we sacrifice a few points near the decision boundary between the two classes, we can still create a pretty good decision boundary like the dotted line. The dashed supported line still helps to create a large margin for this classifier. For each point overlapping to the other side of the respective support lines (or margin line, or border), we call that point into an unsafe area. Note that the safety zone of the two classes is different, intersecting the part between the two support lines.
- In both cases, the margin created by the decision boundary and the dashed line is also known as a Soft Margin. Also by this designation, the original SVM is also known as Hard Margin SVM.

6.4.3 Constrained-Optimization Problem (COP)

- Constrained optimization is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables.

The objective function is either

- a cost function or energy function which is to be minimized, or
- a reward function or utility function, which is to be maximized.

Constraints can be either

- hard constraints which set conditions for the variable that are required to be satisfied, or
- soft constraints which have some variable values that are penalized in the objective function if the conditions on the variables are not satisfied.

Max margin classifier

- We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\underbrace{\max_{\mathbf{w},b} \underbrace{\frac{1}{\|\mathbf{w}\|} \min_{2} \mathbf{x}_1 \in D}} \left|\mathbf{w}^T\mathbf{x}_i + b\right| \text{ s.t. } \underbrace{\forall i y_i \left(\mathbf{w}^T x_i + b \geq 0\right)}_{\text{maximize margin}}$$
$$\gamma(\mathbf{w},b)$$

Because the hyperplane is scale invariant, we can fix the scale of w,b anyway we want. Let's be clever about it, and choose it such that

$$\min_{\mathbf{x}\in D} \left|\mathbf{w}^T\mathbf{x} + b\right| = 1$$

Then the objective becomes:

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2} \cdot 1 = \min_{\mathbf{w},b} \|\mathbf{w}\|_2 = \min_{\mathbf{w},b} \|\mathbf{w}\|_2^2 0$$

The new optimization problem becomes:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 \text{ subject to: } 1 - y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \leq 0, \forall n = 1, 2, \ldots, N$$

Here, we took the inverse of the objective function, squared it to get a differential function, and multiplied by 1/2 for a nicer derivative expression.

Lagrange Multiplier for SVM (the linearly separable case)

- Original Optimization Problem:

$$\begin{array}{ll} \text{minimize}_{\mathbf{w},b} & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} \\ (\mathbf{w}\cdot\mathbf{x}_j + b)\, y_j & \geq 1, \forall j \end{array}$$

Lagrangian:

$$\begin{array}{l} L(\mathbf{w},\alpha) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_j \alpha_j \left[(\mathbf{w}\cdot\mathbf{x}_j + b)\, y_j - 1\right] \\ \alpha_j \geq 0, \forall j \end{array}$$

Our goal is to solve

$$\min_{\vec{w},b} \max_{\vec{\alpha}\geq 0} L(\vec{w},\vec{\alpha})$$

Now, we have the Primal SVM

$$\min_{\vec{w},b} \max_{\vec{\alpha}\geq 0} \frac{1}{2}\|\vec{w}\|^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

and we change it to the Dual SVM by swapping min and max

$$\max_{\vec{\alpha}\geq 0} \min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Slater's condition from convex optimization guarantees that these two optimization problems are equivalent!

We can solve for optimal w, b as function of α:

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j y_j x_j \rightarrow \quad \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \rightarrow \sum_j \alpha_j y_j = 0$$

Substituting these values back in (and simplifying), we obtain:

(Dual)

$$\max_{\vec{\alpha}\geq 0, \sum_j \alpha_j y_j = 0} \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

=>This is the most important function in SVM

Lagrange Multiplier for SVM (the non-separable case, skipping the detail)

- Primal:

$$\begin{gathered}\text{minimize}_{\mathbf{w},b} \quad \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_j \xi_j \\ (\mathbf{w}\cdot\mathbf{x}_j + b)\, y_j \geq 1 - \xi_j, \forall j \\ \xi_j \geq 0, \forall j\end{gathered}$$

  Solve for w, b, a:

$$\begin{aligned}&\mathbf{w} = \textstyle\sum_i \alpha_i y_i \mathbf{x}_i \\ &b = y_k - \mathbf{w}\cdot\mathbf{x}_k \\ &\text{for any } k \text{ where } C > \alpha_k > 0\end{aligned}$$

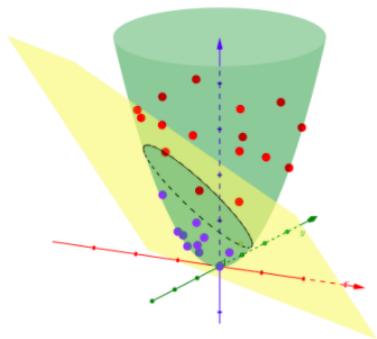
  Dual:

$$\begin{gathered}\text{maximize}_{\alpha} \quad \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j \\ \textstyle\sum_i \alpha_i y_i = 0 \\ C \geq \alpha_i \geq 0\end{gathered}$$

  Why do we learn about the dual SVM? => There are some quadratic programming algorithms that can solve the dual faster than the primal (at least for small datasets)

6.4.4 Kernel SVM

- The basic idea of the Kernel SVM and kernel methods, in general, is to find a transformation so that the original, non-linear and non-separable data, is converted to the new space. In this new dimension, the data becomes the linearly separable data.



- In short, the Kernel SVM is finding a function that converts data x from the original feature space into data in a new space by the function Φ(x). In this example, the function Φ() simply introduces a new data dimension (a new feature) that is a function of the known features. This function needs to fulfill our purpose: in a new space, the data between two classes is either linearly separable or almost linearly separable. Then, we can use common linear classifiers such as PLA, Logistic Regression, or Hard / Soft Margin SVM.

- The $\Phi()$ functions often create new data with a higher number of dimensions than the original, even an infinite number of dimensions. If we calculate these functions directly, we will inevitably run into memory and computational performance issues. One approach is to use kernel functions that describe the relationship between any two data points in the new space, instead of going to compute each data point directly in the new space. This technique is built on the observation of SVM's duality problem.

Popular Kernel Functions

- Linear:
$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$$
- Polynomial:
$$k(\mathbf{x}, \mathbf{z}) = \left(r + \gamma\mathbf{x}^T\mathbf{z}\right)^d$$
- Radial Basis Function (Gaussian):
$$k(\mathbf{x}, \mathbf{z}) = e^{-\gamma\|\mathbf{x}-\mathbf{z}\|_2^2}, \gamma > 0$$
- Sigmoid:
$$k(\mathbf{x}, \mathbf{z}) = \tanh\left(\gamma\mathbf{x}^T\mathbf{z} + r\right)$$

6.4.5 Pros and cons of SVM algorithm

- Pros:
  - SVM works relatively well when there is a clear margin of separation between classes.
  - SVM is more effective in high dimensional spaces.
  - SVM is effective in cases where the number of dimensions is greater than the number of samples.
  - SVM is relatively memory efficient.
- Cons:
  - SVM algorithm is not suitable for large data sets.
  - SVM does not perform very well when the data set has more noise.
  - In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
  - As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

6.4.6 Application of SVM

- As we have seen, SVMs depends on supervised learning algorithms. The aim of using SVM is to correctly classify unseen data. SVMs have a number of applications in several fields.
  - Face Detection
  - Text and Hypertext Categorization
  - Classification of Images
  - Bioinformatics
  - Protein Fold and Remote Homology Detection
  - Handwriting Recognition
  - Geo and Environmental Sciences
  - Generalized Predictive Control
- Thus, we conclude that the SVMs can not only make the reliable prediction but also can reduce redundant information. The SVMs also obtained results comparable with those obtained by other approaches.

6.5 Support Vector Machine Model

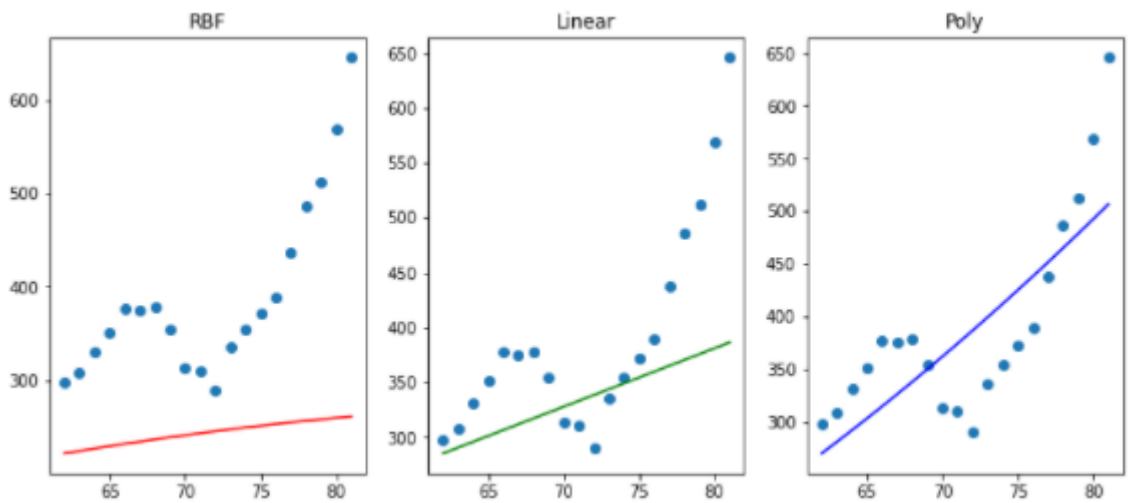
```
svr_rbf = SVR(kernel='rbf', C=20, gamma=0.0001)
svr_lin = SVR(kernel='linear', C=20)
svr_poly = SVR(kernel='poly', C=20, degree=2)
```

```
svr_rbf.fit(x0_train, y0_train)
yhat_rbf = svr_rbf.predict(x0_test)
yhat_rbf
```

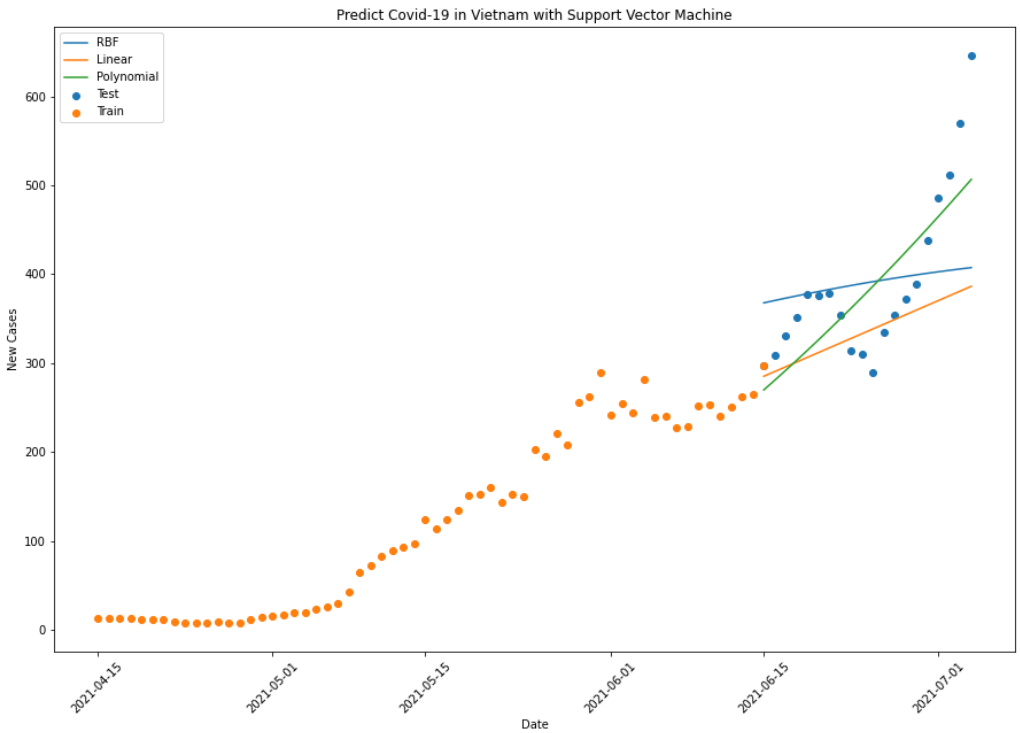
```
svr_lin.fit(x0_train,y0_train)
yhat_lin = svr_lin.predict(x0_test)
yhat_lin
```

```
svr_poly.fit(x0_train,y0_train)
yhat_poly = svr_poly.predict(x0_test)
yhat_poly
```

```
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
plt.plot(x0_test,yhat_rbf, color='r')
plt.scatter(x0_test,y0_test)
plt.title("RBF")
plt.subplot(1,3,2)
plt.plot(x0_test,yhat_lin, color='g')
plt.scatter(x0_test,y0_test)
plt.title("Linear")
plt.subplot(1,3,3)
plt.plot(x0_test,yhat_poly - mean_absolute_error(yhat_poly,y0_test),
 color='b')
plt.scatter(x0_test,y0_test)
plt.title("Poly")
plt.show()
```



```
plt.figure(figsize=(15,10))
plt.title("Predict Covid-19 with Support Vector Machine")
plt.plot(test.index,yhat_rbf + mean_absolute_error(yhat_rbf,y0_test)
, label='RBF')
plt.plot(test.index,yhat_lin, label='Linear')
plt.plot(test.index,yhat_poly - mean_absolute_error(yhat_poly,
y0_test) , label='Polynomial')
plt.scatter(test.index,y0_test, label='Test')
plt.scatter(train.index,y0_train, label='Train')
plt.xlabel('Date')
plt.ylabel('New Cases')
plt.legend(loc="best")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

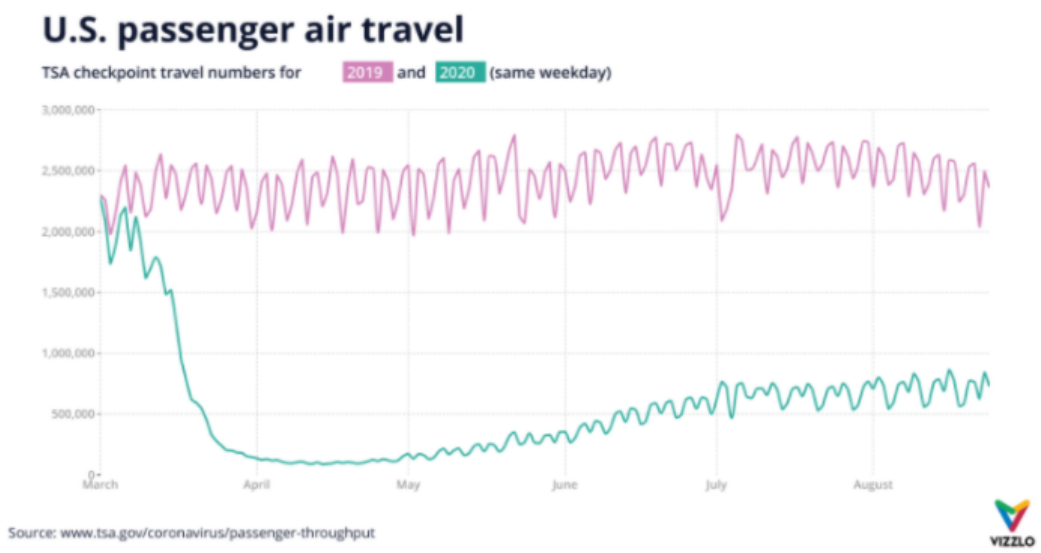


In this problem, we found that the SVM algorithm was not suitable when testing the predicted values against the actual values on the test set by plotting data. Therefore, we don't use this model for the future prediction.

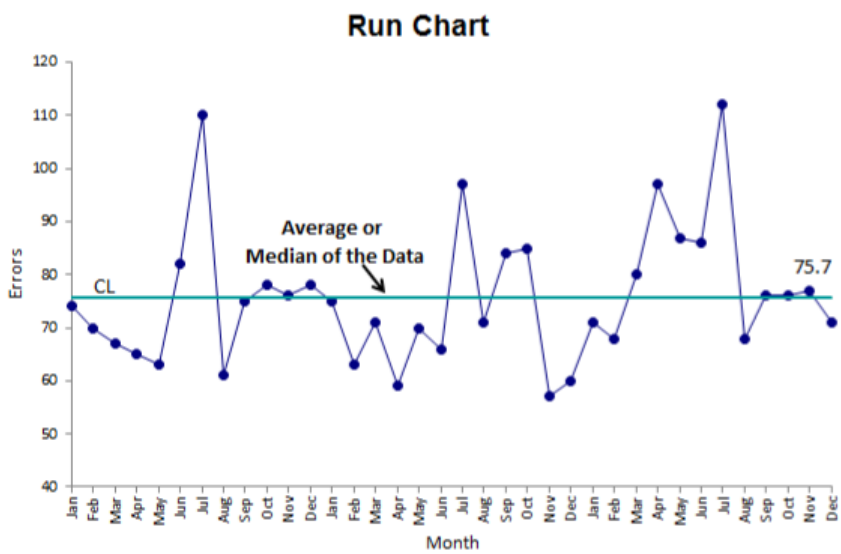
6.6 Time Series

6.6.1 Time series

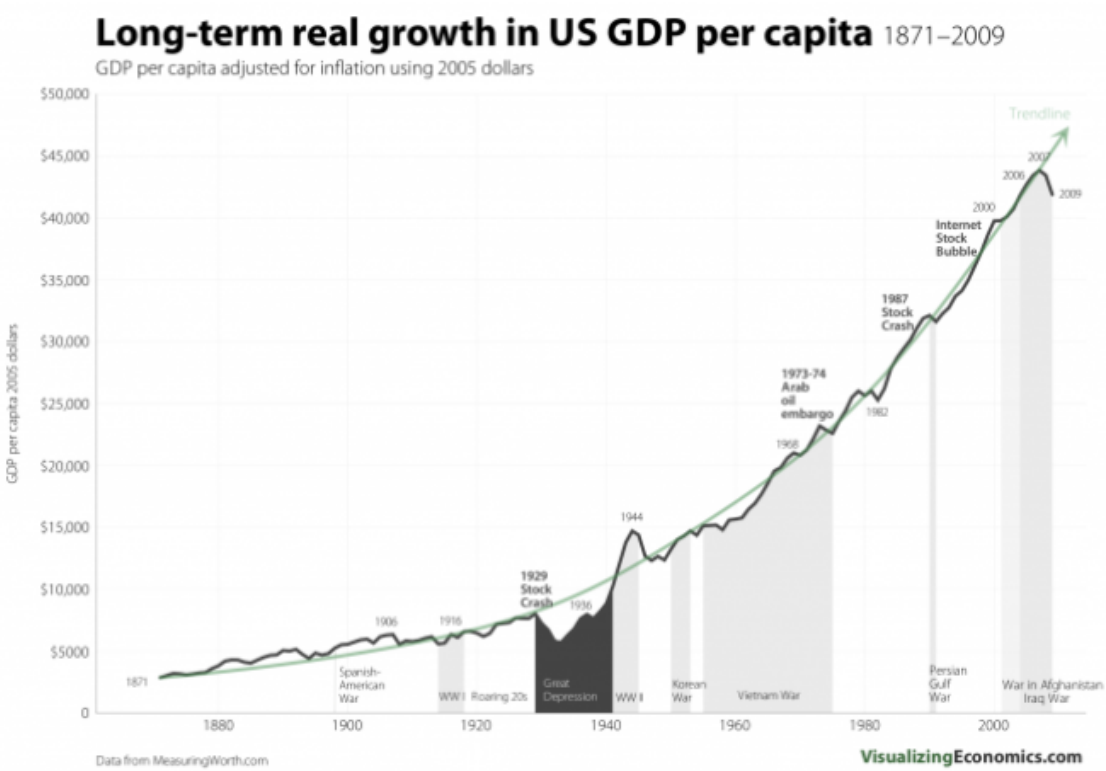
A time series is an arrangement of data points indexed, listed, or graphed in their chronological order.For the most part, a time series looks like a sequence of successive equally spaced points in time, which is called the sequence of discrete-time data. The following time series, for example, depicts the number of people who opt to travel by plane.



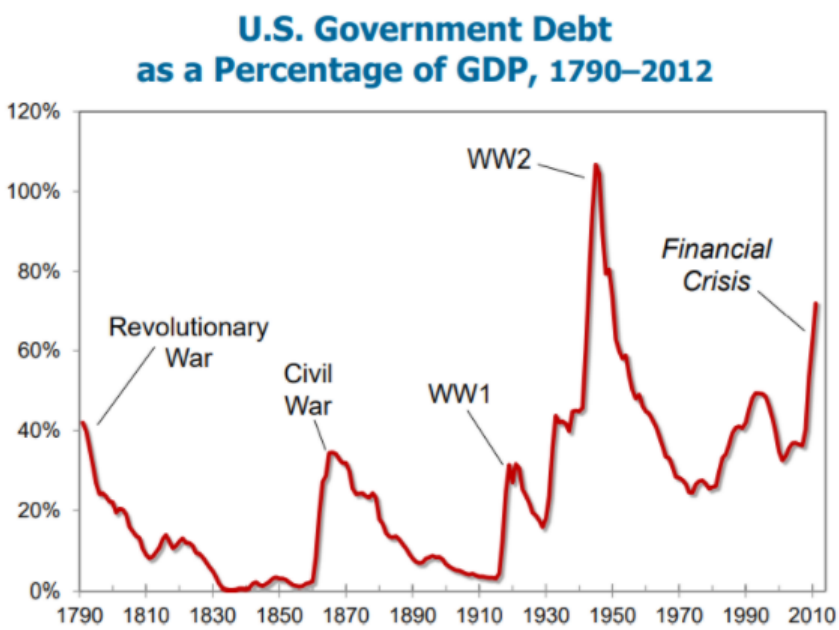
Time series are of great importance because they are very frequently plotted via run charts.



Time series can be used in a variety of ways. This means that statistics, signal processing, pattern identification, econometrics, mathematical finance, weather forecasting, earthquake prediction, and even electroencephalography, control engineering, astronomy, and communications engineering are all examples of applications. More importantly, this is used in the domain of applied science and engineering that involve temporal measurements a majority of the time.
For example, the following figures use time series to demonstrate the GDP and the government debt of the United States since 1950 and 1790. Time series help users to easily identify the trends of the graph to make concise analysis and prediction.

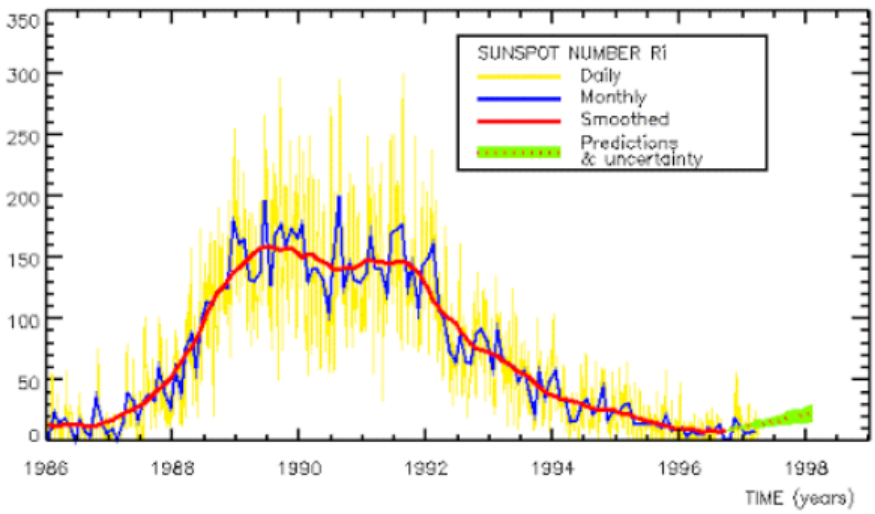


US GDP over time



US Government debt since 1750

The time series can also be used in astronomy to predict the sunspot.

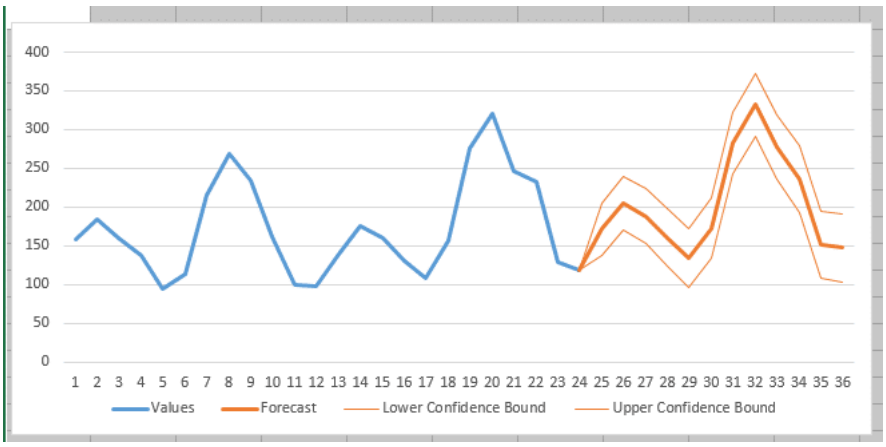


Time Series in Astronomy

Now we move on another aspect of time series - called time series analysis. In fact, time series analysis consists of various methods that can be used for analyzing time series data, which would serve to extract some meaningful statistics and many other intrinsic characteristics of the data. Time series forecasting is the use of a model to predict the future values of the datasets based on the previously obtained values. While regression analysis is often employed in such a way as to test relationships between one or more different time series, this type of analysis is not usually called "time series analysis", which refers

in particular to relationships between different points in time within a single series. We often use interrupted time series analysis to detect changes in the evolution of a time series from before to after some intervention which may affect the underlying variable.

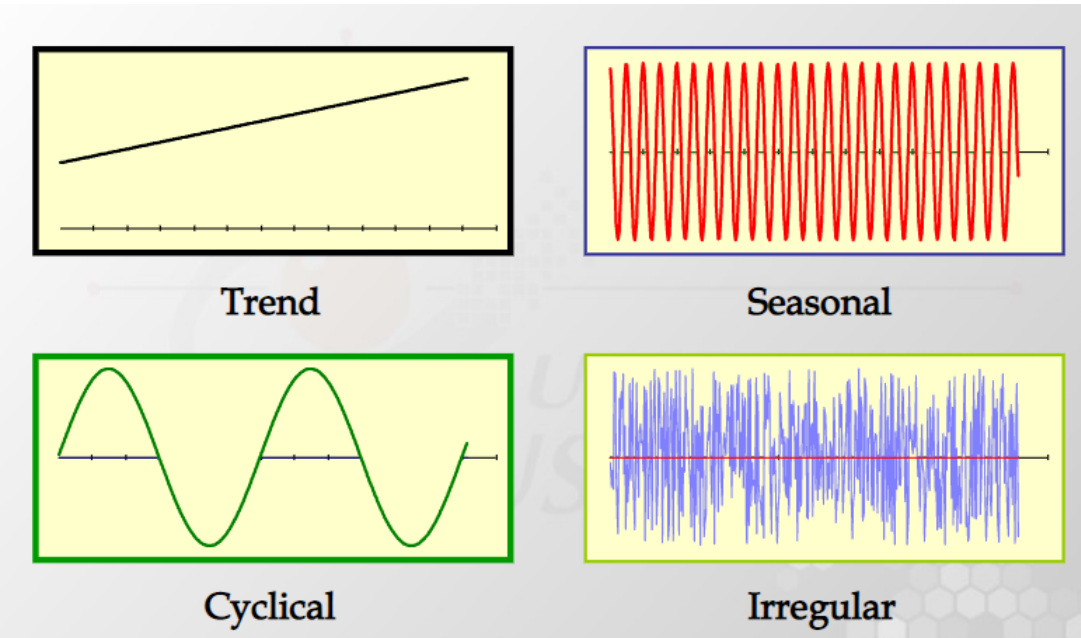
Next, time series data have a natural temporal ordering. This helps differentiate time series analysis from cross-sectional studies, in which there is no natural ordering of the observations (e.g. explaining people's wages by reference to their respective education levels, where the individuals' data could be entered in any order). Time series analysis is also different from spatial data analysis where the observations typically relate to geographical locations (e.g. accounting for house prices by the location as well as the intrinsic characteristics of the houses). A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values (such as time reversibility).

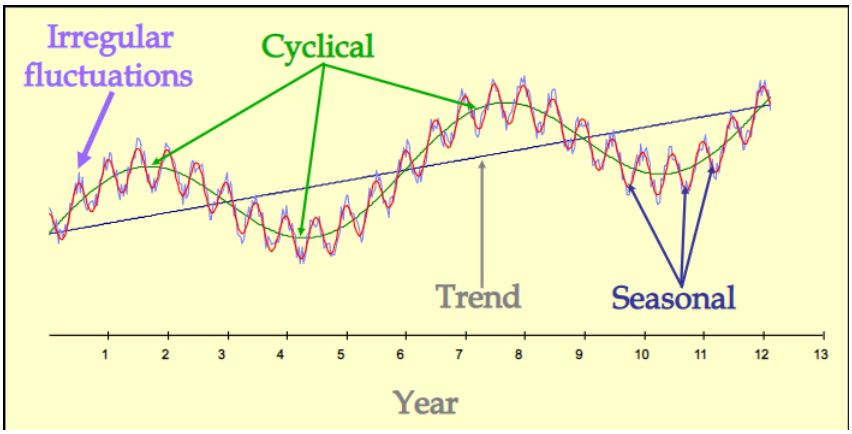


6.6.2 Decomposing Time Series

Generally, there are two methods for decomposing a time series, they are decomposition based on rates of change and decomposition based on probability.
This is a fundamental technique for all kinds of time series analysis, especially useful for seasonal adjustment. From an observed time series, it will construct a number of component series (that could be used to reconstruct the original by additions or multiplications) where each of these has a certain characteristic or type of behavior. For example, time series are usually decomposed into:

- T, the trend component at time t, which reflects the long-term progression of the series (secular variation). A trend exists when there is a persistent increasing or decreasing direction in the data. The trend component does not have to be linear.
- C, the cyclical component at time t, which reflects repeated but non-periodic fluctuations. The duration of these fluctuations depend on the nature of the time series.
- S, the seasonal component at time t, reflecting seasonality (seasonal variation). A seasonal pattern exists when a time series is influenced by seasonal factors. Seasonality occurs over a fixed and known period (e.g., the quarter of the year, the month, or day of the week).
- I, the irregular component (or "noise") at time t, which describes random, irregular influences. It represents the residuals or remainders of the time series after the other components have been removed.





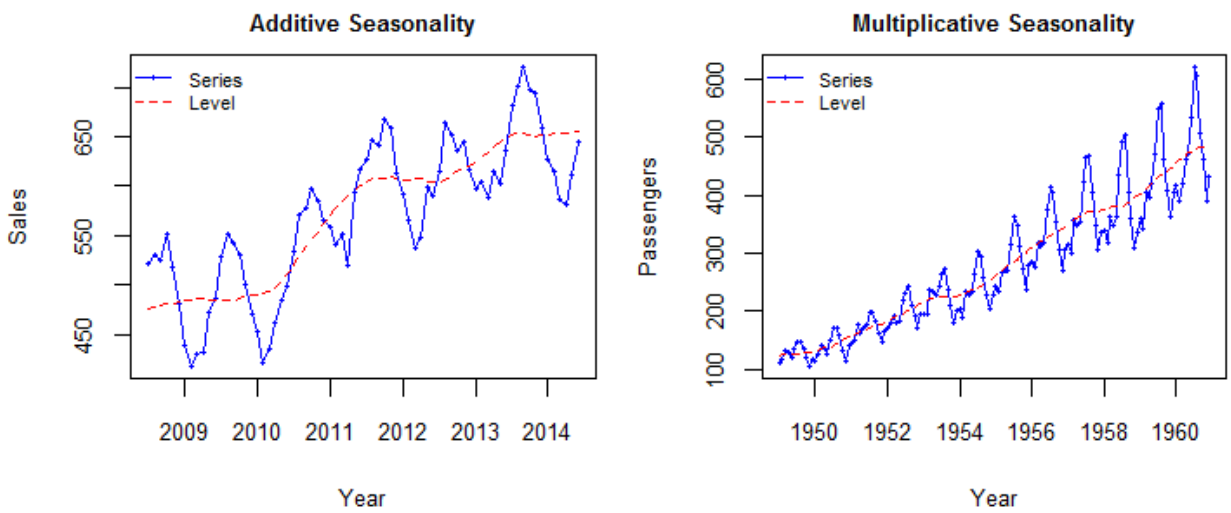
Hence a time series using an additive model can be thought of as

$$y = T + C + S + I$$

whereas a multiplicative model would be

$$y = T.C.S.I$$

An additive model would be used when the variations around the trend do not vary with the level of the time series whereas a multiplicative model would be appropriate if the trend is proportional to the level of the time series.
Sometimes the trend and cyclical components are grouped into one, called the trend-cycle component. The trend-cycle component can just be referred to as the "trend" component, even though it may contain cyclical behavior.



6.7 ARIMA

6.7.1 Auto-Regressive Model

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself.

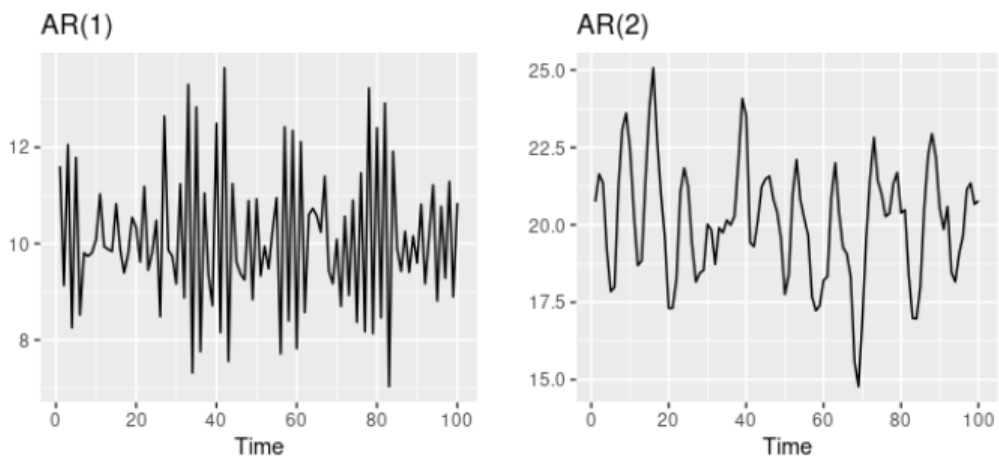
Thus, an autoregressive model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

where ε_t is white noise. This is like a multiple regression but with lagged values of y_t as predictors. We refer to this as an AR(p) model, an autoregressive model of order p.

Autoregressive models are remarkably flexible at handling a wide range of different time series patterns. The two series in the figure show series from an AR(1) model and an AR(2) model. Changing the parameters $\phi_1, \ldots, \phi_p$ results in different time series patterns. The variance of the error term ε_t will only change the scale of the series, not the patterns.

Figure 8.5: Two examples of data from autoregressive models with different parameters.



Left: $AR(1)$ with $y = 18 - 0.8y_{t-1} + \varepsilon_t$. Right: $AF(2)$ with $y = 8 + 1.3yt - 1 - 0.7y_{t-2} + \varepsilon_t$. In both cases, ε_t is normally distributed white noise with mean zero and variance one. For an AR(1) model:

- when $\phi_1 = 0, y_t$ is equivalent to white noise
- when $\phi_1 = 1$ and $c = 0, y_1$ is equivalent to a random walk
- when $\phi_1 = 1$ and $c \neq 0, y_t$ is equivalent to a random walk with drift
- when $\phi_1 < 0, y_t$ tends to oscillate around the mean.

We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required. For an AR(t) model: $-1 < \phi_1 < 1$. For an AR(2) model: $-1 < \phi_2 < 1, \phi_1 + \phi_2 < 1, \phi_2 - \phi_1 < 1$. When $p \geq 3$, the restrictions are much more complicated. R takes care of these restrictions when estimating a model.

6.7.2 Moving Average Model

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where ε_t is white noise. We refer to this as an MA(q) model, a moving average model of order q. Of course, we do not observe the values of ε_t, so it is not really a regression in the usual sense.
Notice that each value of y_t can be thought of as a weighted moving average of the past few forecast errors. However, moving average models should not be confused with the moving average smoothing. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

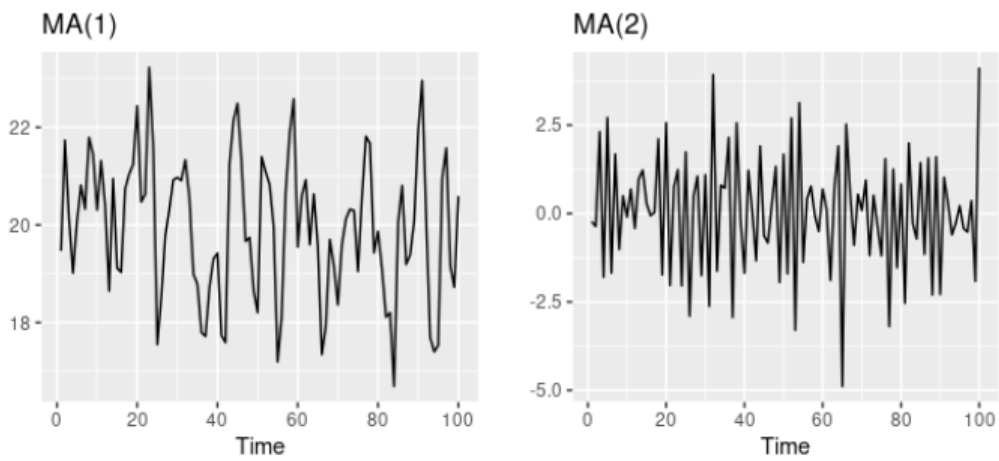


Figure 8.6: Two examples of data from moving average models with different parameters. Left: MA(1) with $y_t = 20 + \varepsilon_t + 0.8\varepsilon_{t-1}$. Right: MA(2) with $y_t = \varepsilon_t - \varepsilon_{t-1} + 0.8\varepsilon_{t-2}$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

Figure 8.6 shows some data from an MA(1) model and an MA(2) model. Changing the parameters $\theta_1, \ldots, \theta_q$ results in different time series patterns. As with autoregressive models, the variance of the error term ε_t will only change the scale of the series, not the patterns. It is possible to write any stationary AR (p) model as an MA(∞) model. For example, using repeated substitution, we can demonstrate this for an AR(1) model:

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + \varepsilon_t \\ &= \phi_1 \left(\phi_1 y_{t-2} + \varepsilon_{t-1}\right) + \varepsilon_t \\ &= \phi_1^2 y_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \\ &= \phi_1^3 y_{t-3} + \phi_1^2 \varepsilon_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \end{aligned}$$

etc.
Provided $-1 < \phi_1 < 1$, the value of ϕ_1^k will get smaller as k gets larger. So eventually we obtain

$$y_t = \varepsilon_t + \phi_1\varepsilon_{t-1} + \phi_1^2\varepsilon_{t-2} + \phi_1^3\varepsilon_{t-3} + \cdots$$

an MA(∞) process.
The reverse result holds if we impose some constraints on the MA parameters. Then the MA model is called invertible. That is, we can write any invertible MA(q) process as an AR(∞) process. Invertible models are not simply introduced to enable us to convert from MA models to AR models. They also have some desirable mathematical properties.
For example, consider the MA(1) process, $y_t = \varepsilon_t + \theta_1\varepsilon_{t-1}$. In its AR($\infty$) representation, the most recent error can be written as a linear function of current and past observations:

$$\varepsilon_t = \sum_{j=0}^{\infty}(-\theta)^j y_{t-j}$$

When $|\theta| > 1$, the weights increase as lags increase, so the more distant the observations the greater their influence on the current error. When $|\theta| = 1$, the weights are constant in size, and the distant observations have the same influence as the recent observations. As neither of these situations make much sense, we require $|\theta| < 1$, so the most recent observations have higher weight than observations from the more distant past. Thus, the process is invertible when $|\theta| < 1$.
The invertibility constraints for other models are similar to the stationarity constraints.

- For an MA(1) model: $-1 < \theta_1 < 1$.
- For an MA(2) model: $-1 < \theta_2 < 1, \theta_2 + \theta_1 > -1, \theta_1 - \theta_2 < 1$

More complicated conditions hold for $q \geq 3$. Again, R will take care of these constraints when estimating the models.

6.7.3 Non-seasonal ARIMA model

If we combine differencing with autoregression and a moving average model, we obtain a nonseasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average (in this context, "integration" is the reverse of differencing). The full model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} + \varepsilon_t$$

where y'_t is the differenced series (it may have been differenced more than once). The "predictors" on the right hand side include both lagged values of y_t and lagged errors.
We call this an ARIMA (p, d, q) model, where:

- $p =$ order of the autoregressive part.
- $d =$ degree of first differencing involved.
- $q =$ order of the moving average part.

The same stationarity and invertibility conditions that are used for autoregressive and moving average models also apply to an ARIMA model.
Many of the models we have already discussed are special cases of the ARIMA model, as shown in the following table.

Table of some special cases of ARIMA models.

White noise	ARIMA$(0,0,0)$
Random walk	ARIMA$(0,1,0)$ with no constant
Random walk with drift	ARIMA$(0,1,0)$ with a constant
Autoregression	ARIMA$(p,0,0)$
Moving average	ARIMA$(0,0,q)$

Once we start combining components in this way to form more complicated models, it is much easier to work with the backshift notation. For example, Equation (8.1) can be written in backshift notation as

$$(1-\phi_1 B-\cdots-\phi_p B^p) \quad (1-B)^d y_t = c+(1+\theta_1 B+\cdots+\theta_q B^q)\,\varepsilon_t$$

6.7.4 Estimation and order selection

Maximum likelihood estimation
Once the model order has been identified (i.e., the values of p, d and q), we need to estimate the parameters $c, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$. When R estimates the ARIMA model, it uses maximum likelihood estimation (MLE). This technique finds the values of the parameters which maximise the probability of obtaining the data that we have observed. For ARIMA models, MLE is similar to the least squares estimates that would be obtained by minimising

$$\sum_{t=1}^{T} \varepsilon_t^2$$

(For the regression models considered earlier, MLE gives exactly the same parameter estimates as least squares estimation.) Note that ARIMA models are much more complicated to estimate than regression models, and different software will give slightly different answers as they use different methods of estimation, and different optimisation algorithms.

In practice, R will report the value of the log likelihood of the data; that is, the logarithm of the probability of the observed data coming from the estimated model. For given values of p, d and q, R will try to maximise the log likelihood when finding parameter estimates.

Information Criteria
Akaike's Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. It can be written as

$$\text{AIC} = -2\log(L) + 2(p+q+k+1)$$

where L is the likelihood of the data, $k=1$ if $c \neq 0$ and $k=0$ if $c=0$. Note that the last term in parentheses is the number of parameters in the model (including σ^2, the variance

of the residuals). For ARIMA models, the corrected AIC can be written as

$$\text{AICc} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2}$$

and the Bayesian Information Criterion can be written as

$$\text{BIC} = \text{AIC} + [\log(T) - 2](p+q+k+1)$$

Good models are obtained by minimising the AIC, AICc or BIC. Our preference is to use the AICc. It is important to note that these information criteria tend not to be good guides to selecting the appropriate order of differencing (d) of a model, but only for selecting the values of p and q. This is because the differencing changes the data on which the likelihood is computed, making the AIC values between models with different orders of differencing not comparable. So we need to use some other approach to choose d, and then we can use the AICc to select p and q.

6.7.5 Forecasting

Point forecasts
Although we have calculated forecasts from the ARIMA models in our examples, we have not yet explained how they are obtained. Point forecasts can be calculated using the following three steps.

1. Expand the ARIMA equation so that y_t is on the left hand side and all other terms are on the right.
2. Rewrite the equation by replacing t with $T+h$.
3. On the right hand side of the equation, replace future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals.

Beginning with $h=1$, these steps are then repeated for $h=2,3,\ldots$ until all forecasts have been calculated.

6.7.6 Seasonal ARIMA models

So far, we have restricted our attention to non-seasonal data and non-seasonal ARIMA models. However, ARIMA models are also capable of modelling a wide range of seasonal data. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\uparrow}\underbrace{(P,D,Q)_m}_{\uparrow}$$

(p,d,q) is the non-seasonal part of the model, while (P,D,Q) is the seasonal part of the model.
where m = number of observations per year. We use uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model.

The seasonal part of the model consists of terms that are similar to the non-seasonal components of the model, but involve backshifts of the seasonal period. For example, an ARIMA $(1,1,1)(1,1,1)_4$ model (without a constant) is for quarterly data ($m = 4$), and can be written as

$$(1-\phi_1 B)\left(1-\Phi_1 B^4\right)(1-B)\left(1-B^4\right)y_t = (1+\theta_1 B)\left(1+\Theta_1 B^4\right)\varepsilon_t$$

The additional seasonal terms are simply multiplied by the non-seasonal terms.

ACF/PACF
The seasonal part of an AR or MA model will be seen in the seasonal lags of the PACF and ACF. For example, an ARIMA$(0,0,0)(0,0,1)_{12}$ model will show:

- a spike at lag 12 in the ACF but no other significant spikes.
- exponential decay in the seasonal lags of the PACF.

Similarly, an ARIMA $(0,0,0)(1,0,0)_{12}$ model will show:

- exponential decay in the seasonal lags of the ACF
- a single significant spike at lag 12 in the PACF

6.8 ARIMA MODEL

6.8.1 Installing and importing necessary tools

We start by installing *pmdarima*.

```
!pip install pmdarima
```

Listing 5: Installing pdarima package.

Then, we move on to importing packages.

```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from pmdarima import auto_arima
from sklearn.metrics import r2_score,mean_absolute_error,
    mean_squared_error, accuracy_score, classification_report,
    confusion_matrix, precision_recall_fscore_support # Used to evaluate
    the model
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.mixture import GaussianMixture
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
from pandas_profiling import ProfileReport
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB, GaussianNB
from scipy.stats.stats import pearsonr
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn import svm
from scipy.integrate import odeint
import xgboost as xgb
from wordcloud import WordCloud,STOPWORDS
# Import model building tools
import os, sys, glob, datetime, warnings
from google.colab import files, drive
warnings.filterwarnings("ignore")
%matplotlib inline
```

Listing 6: Importing necessary packages.

6.8.2 Data loading and pre-processing

As previously mentioned, we are going to import the dataset straight from GitHub.

```
df_new = pd.read_csv("https://raw.githubusercontent.com/owid/covid-19-
    data/master/public/data/owid-covid-data.csv",  parse_dates=['date'],
    index_col='date')
df_new.tail()
# here, the pandas package is in charge of reading the CSV file.
# the tail() command is used to return the last n rows from the object
    based on position, useful for quick data verification.
```

Listing 7: Data loading script.

We then pre-process the data, keeping only necessary columns for the prediction phase.

```
df = df_new[["continent", "location", "population", "total_cases", "
    new_cases", "new_cases_smoothed", "total_deaths", "new_deaths", "
    new_deaths_smoothed", "total_tests", "new_tests", "positive_rate"]]
df.tail()
```

Listing 8: Data pre-processing script.

6.8.3 The USA

Let us begin by examining the total cases in the USA. First, we extract the data related to this country with the following script.

```
new_cases_USA = df["new_cases"].where(df["location"] == "United States")
    .dropna()
total_cases_USA = df["total_cases"].where(df["location"] == "United
    States").dropna()
new_cases_USA_smoothed = df["new_cases_smoothed"].where(df["location"]
    == "United States").dropna()
```

Listing 9: USA data extraction.

Let us plot the total cases in the USA with the following code script.

```
plt.figure(figsize=(10,8))
plt.title("Total Cases in USA")
plt.plot(total_cases_USA)
plt.xlabel('Date')
plt.ylabel('New cases')
plt.xticks(rotation=45)
plt.show()
```

Listing 10: Code script for data plotting.

Which produces Figure 61

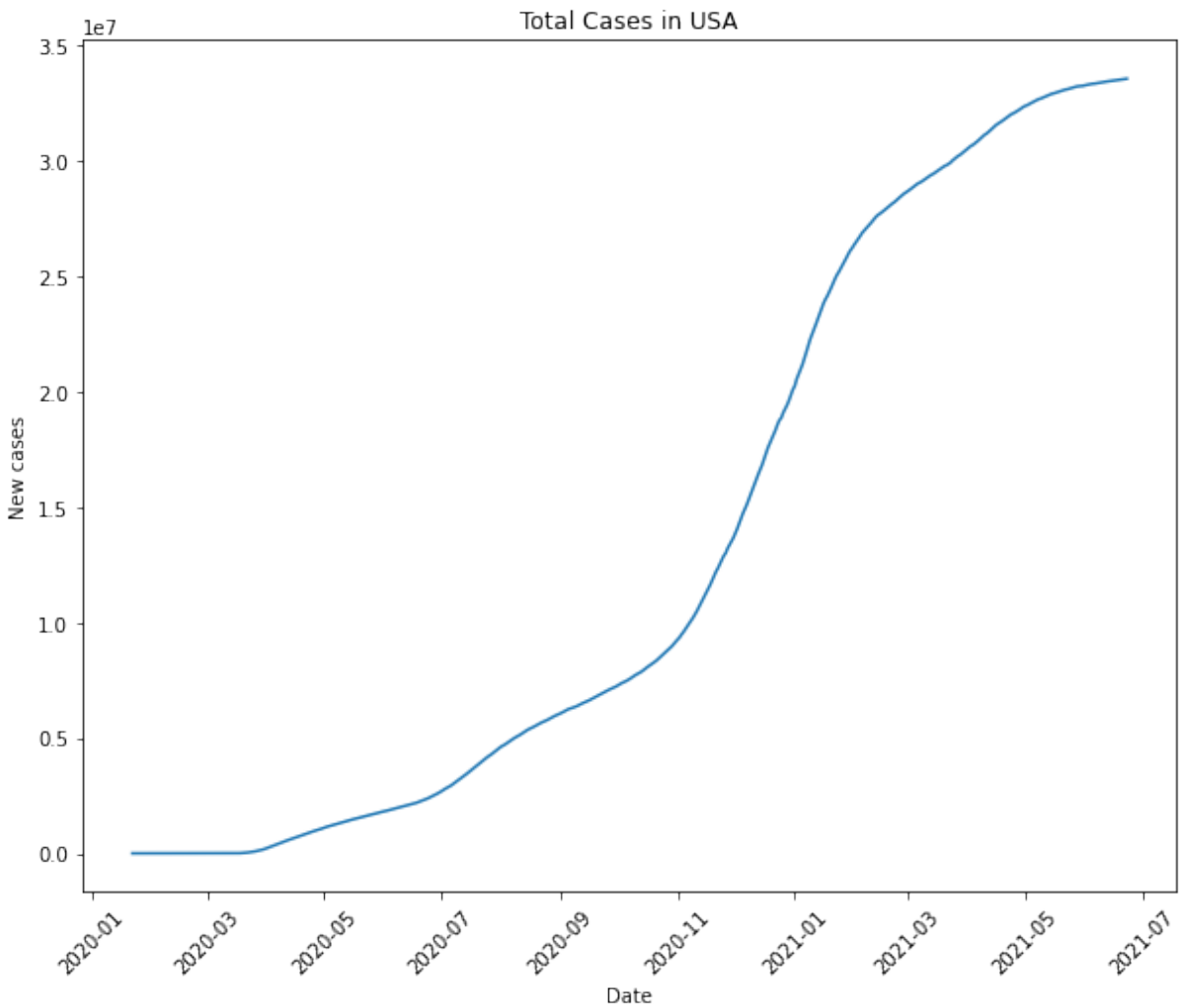


Figure 61: Total cases in the USA

Next, let us plot the new cases here as raw data as well as smoothed data.

```
plt.figure(figsize=(10,8))
plt.title("New cases in USA")
plt.plot(new_cases_USA, label="raw")
plt.plot(new_cases_USA_smoothed, label="smooothed")
plt.xlabel('Date')
plt.ylabel('New cases')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

Listing 11: Code script for raw and smoothed data plotting in the USA.

Which gives Figure 62.

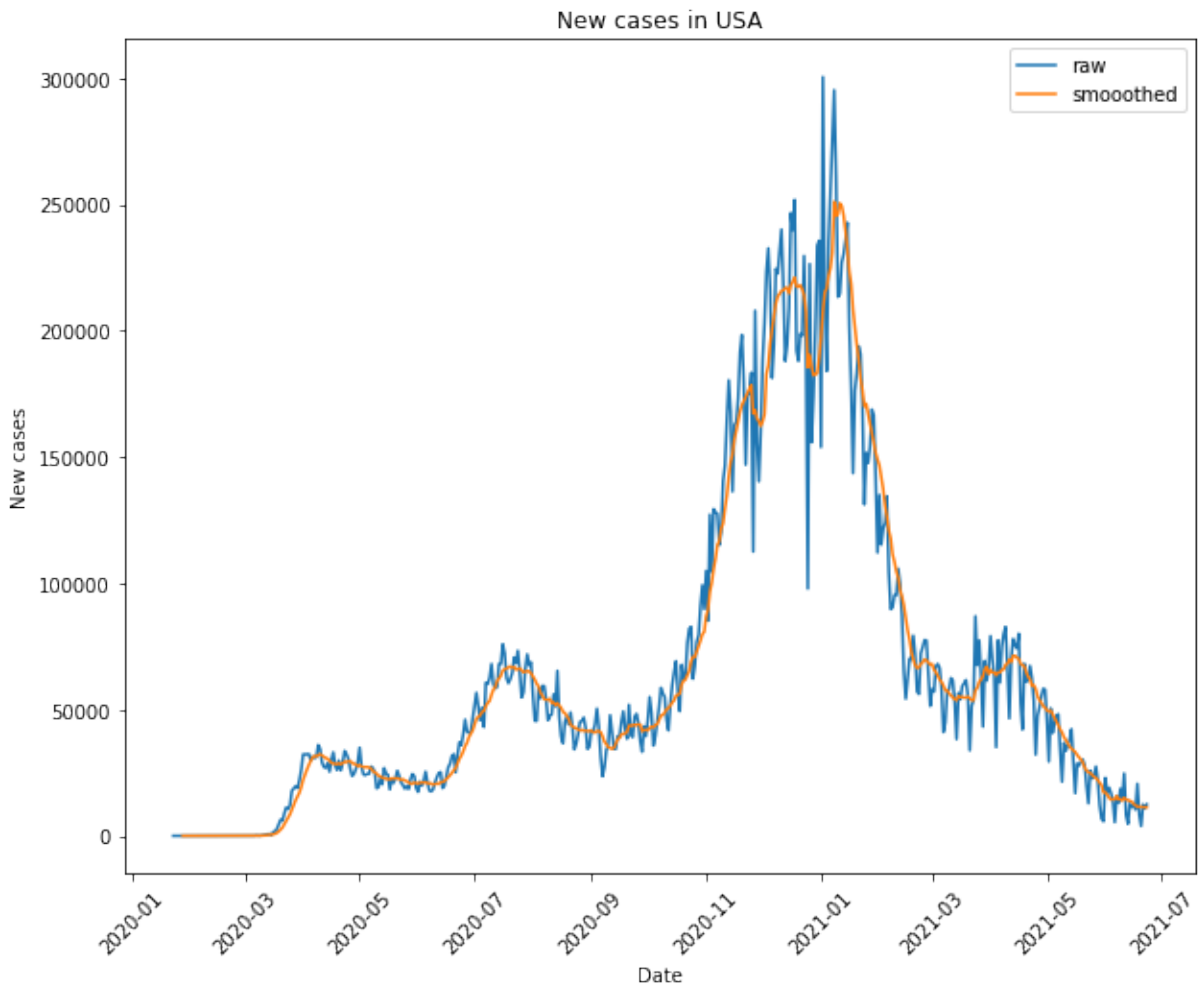


Figure 62: New cases in the USA as raw and smoothed data.

What we can deduce from the figures:

- The number of new COVID-19 infections rocketed from the beginning of 2020 until January 2021, then dropped sharply.
- Expected time is from 1 week to 12 weeks (3 months). For 1 week, we can get the data from January to May or June to train, because the figures were falling sharply. As for 12 weeks, it is very difficult to predict properly.
- We will divide the data to the ratio of 80% for training and 20% for testing. This ratio will be preserved for the next locations. Also, the source code from now on will be repeated a lot, so for the next countries, we will only show the resulted graphs and predictions.

Let us first observe the data from January 2021 to the end of June 2021 and plot it with the following code script, producing Figure 63

```
#Select Time to train
start_time = '2021-02-01'
end_time = '2021-06-01'
```

```
# start_time->end_time (train),  End_time -> now (test)

plt.figure(figsize=(10,8))
plt.title("New cases in USA from " + start_time + " to now")
plt.plot(new_cases_USA[start_time:], label="raw")
plt.plot(new_cases_USA_smoothed[start_time:], label="smoothed")
plt.ylabel("Count")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.legend()
```

Listing 12: Code script for splitting data.

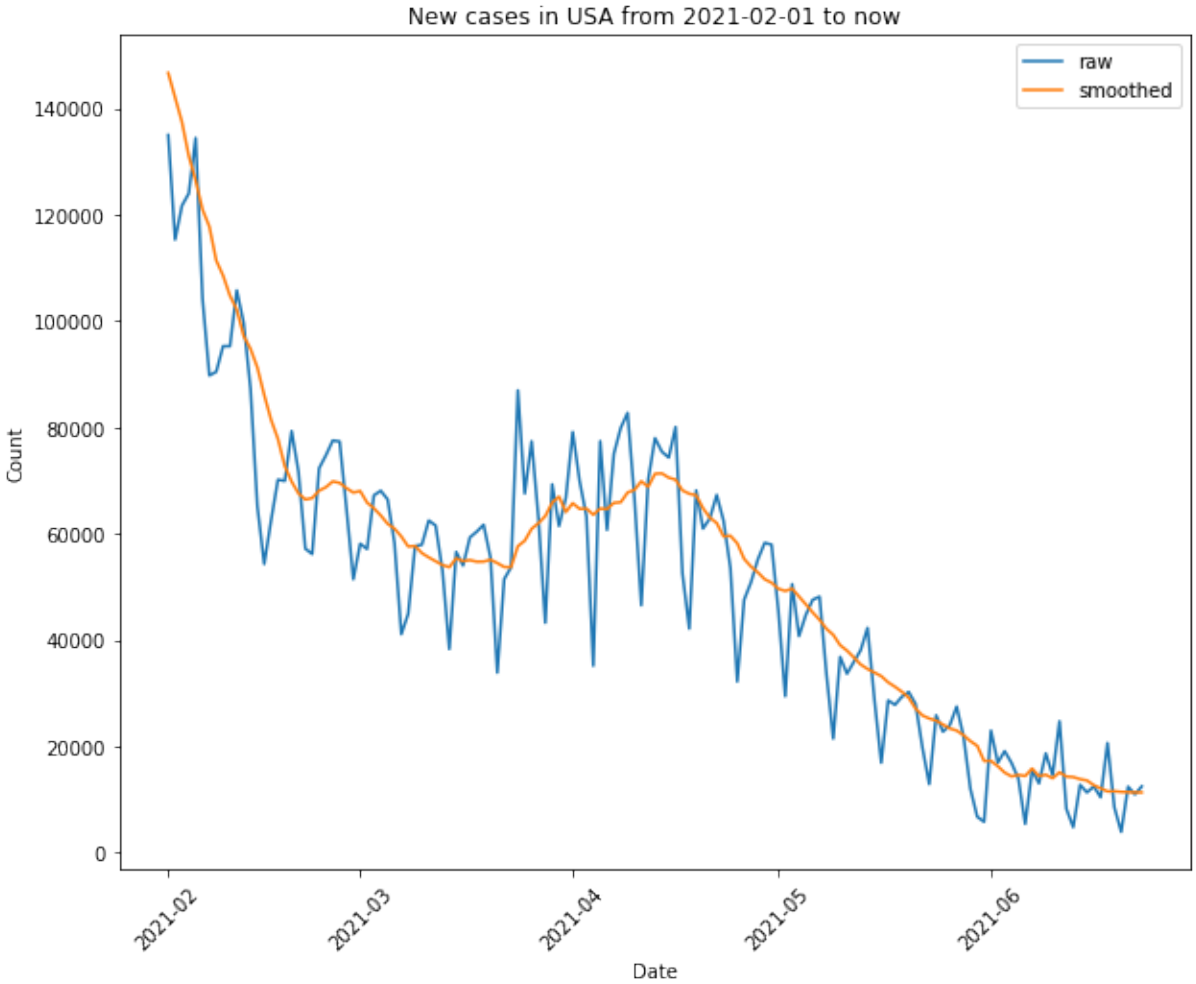


Figure 63: New cases in the USA which are going to be used for training and testing.

Now, we are going to seasonally decompose the data by additive and multiplicative models, respectively.

```
result = seasonal_decompose(new_cases_USA[start_time:], model='multiplicative')
result_add = seasonal_decompose(new_cases_USA[start_time:], model='additional')
```

```
result
```

Listing 13: Code script for seasonal decomposition.

Figure 64 and 65 demonstrate whether our data is seasonal.

```
result_add.plot()
plt.show()
```

Listing 14: additive seasonal decomposition.

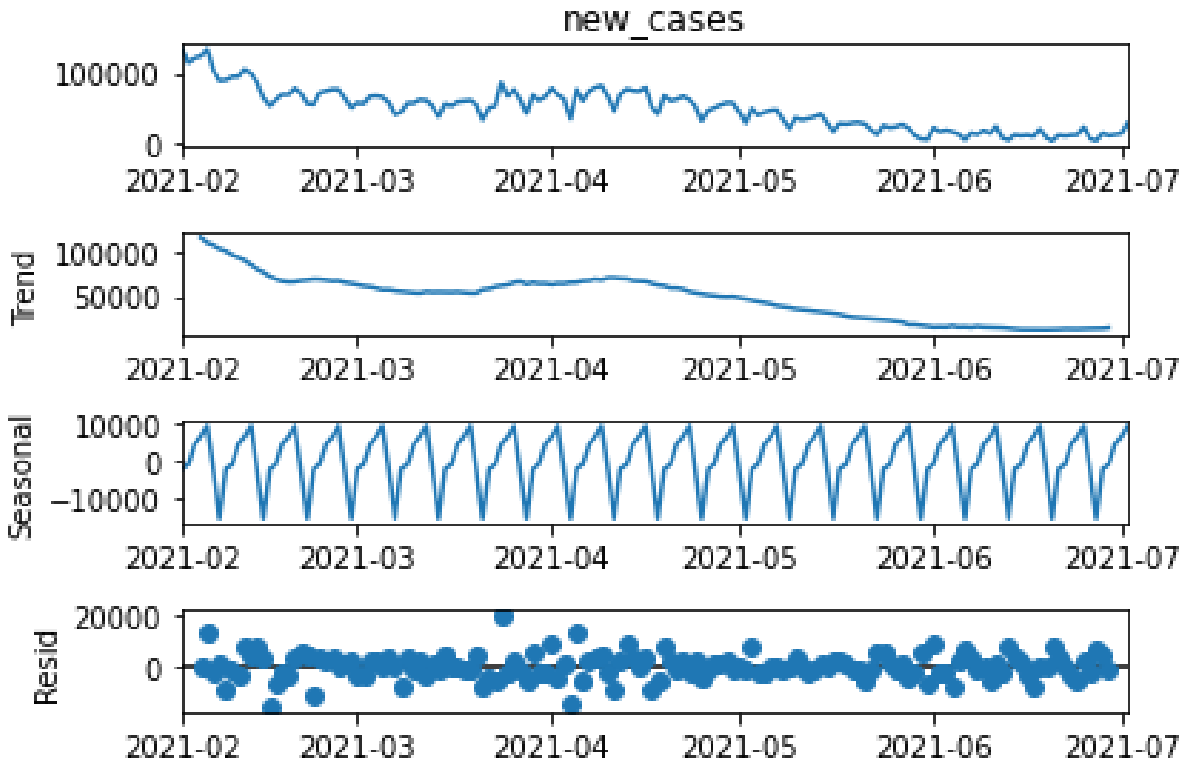


Figure 64: Additive seasonal decomposition.

```
result.plot()
plt.show()
```

Listing 15: Multiplicative seasonal decomposition.

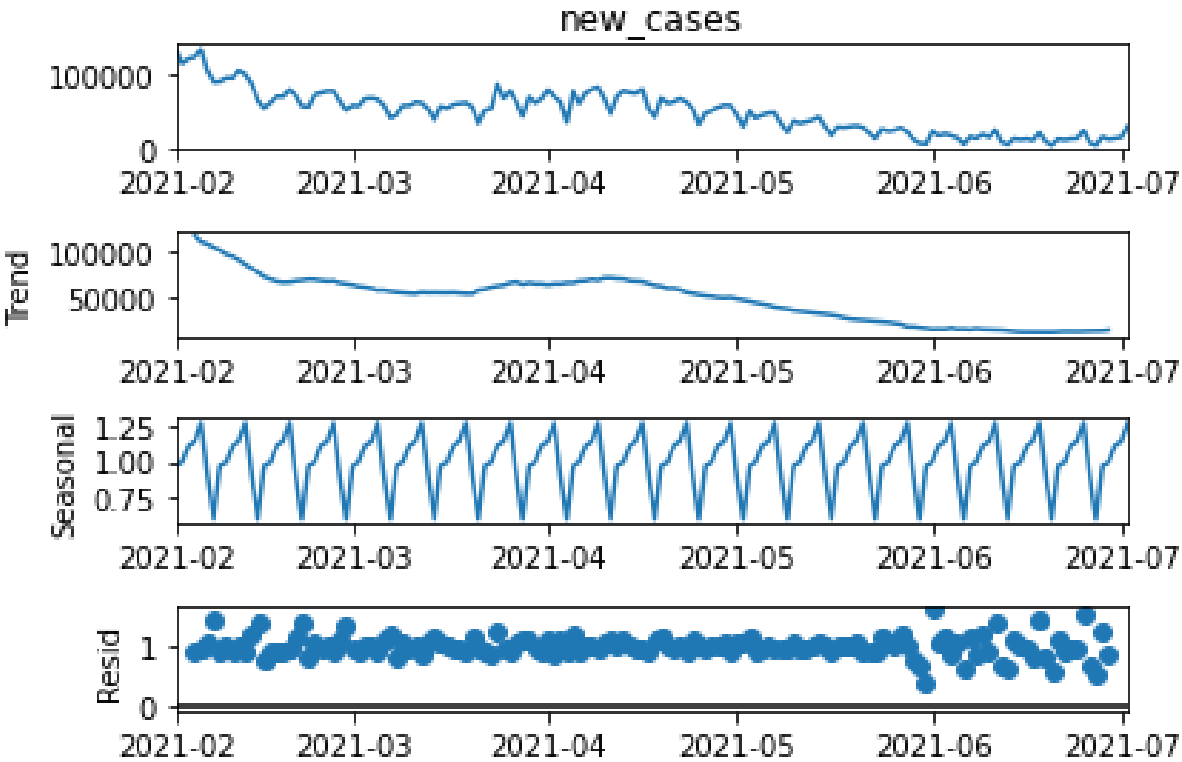


Figure 65: Multiplicative seasonal decomposition.

Next, we will observe the trend and seasonality of the data, as described in Figure 66 and Figure 67.

```
plt.figure(figsize=(15,4))
plt.plot(result.trend)
plt.title("Multiplicative seasonal decomposition trend")
```

Listing 16: Code script for multiplicative seasonal decomposition trend.

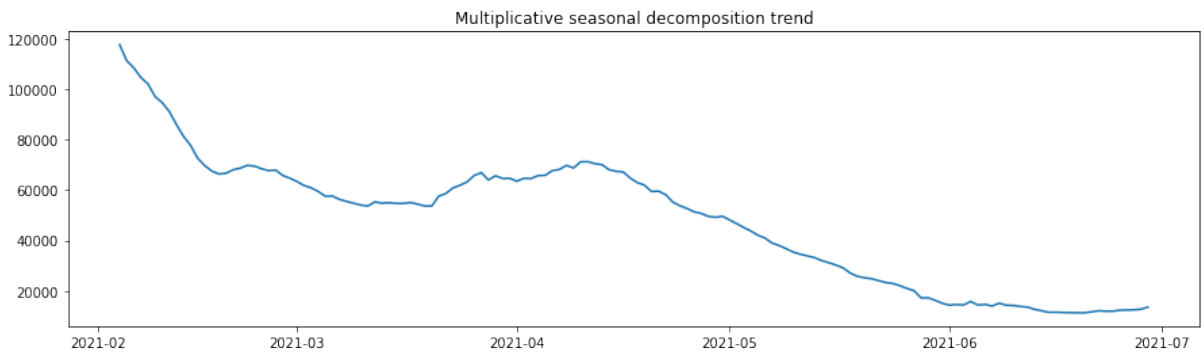


Figure 66: Data trend.

```
plt.figure(figsize=(15,4))
plt.plot(result.seasonal)
plt.title("Seasonality verification")
```

Listing 17: Code script for seasonality verification.

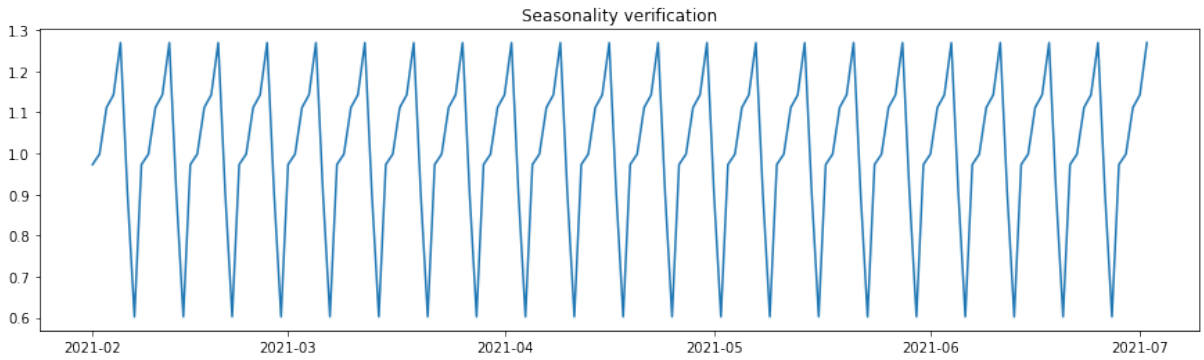


Figure 67: The seasonality of the given data.

After checking if the data is seasonal, we move on to see the fluctuations of it in Figure 68

```
plt.figure(figsize=(15,4))
plt.plot(result.resid)
```

Listing 18: Code script for checking fluctuations.

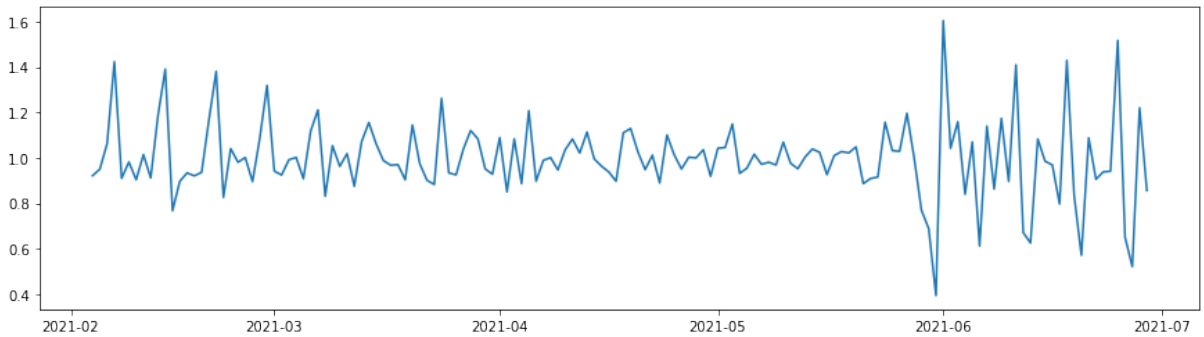


Figure 68: Relative data fluctuations, with some anomalies (June 2021).

Now, let us split the data for training and testing.

```
train_USA = new_cases_USA.loc[start_time:end_time]  # train & test split
test_USA = new_cases_USA.loc[end_time:]
print("Train length: ", len(train_USA))
print("Test length: ",len(test_USA))
print("Ratio: ", (len(test_USA))/ (len(test_USA)+len(train_USA)))
```

Listing 19: Code script for data splitting.

```
Train length:  121
Test length:  32
Ratio:  0.20915032679738563
```

We now select parameters for the ARIMA model and use the aic index for finding the most suitable parameters. The lower the aic, the better the parameters.

```
stepwise_model = auto_arima(new_cases_USA[start_time:], start_p=2,
    start_q=2, max_p=5, max_q=5, m=7, seasonal=True, d=None, D=None,
    trace=True, error_action='ignore', suppress_warnings=True, stepwise=
    True)
```

Listing 20: Code script for ARIMA model building.

The above script's output is:

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[7] intercept   : AIC=3211.592, Time=0.53 sec
 ARIMA(0,1,0)(0,0,0)[7] intercept   : AIC=3260.964, Time=0.01 sec
 ARIMA(1,1,0)(1,0,0)[7] intercept   : AIC=3214.989, Time=0.09 sec
 ARIMA(0,1,1)(0,0,1)[7] intercept   : AIC=3223.843, Time=0.08 sec
 ARIMA(0,1,0)(0,0,0)[7]             : AIC=3259.504, Time=0.01 sec
 ARIMA(2,1,2)(0,0,1)[7] intercept   : AIC=3224.847, Time=0.41 sec  ARIMA
    (2,1,2)(1,0,0)[7] intercept   : AIC=3209.304, Time=0.62 sec
 ARIMA(2,1,2)(0,0,0)[7] intercept   : AIC=inf, Time=0.56 sec
 ARIMA(2,1,2)(2,0,0)[7] intercept   : AIC=3212.588, Time=0.76 sec
 ARIMA(2,1,2)(2,0,1)[7] intercept   : AIC=3210.196, Time=0.83 sec
 ARIMA(1,1,2)(1,0,0)[7] intercept   : AIC=3202.055, Time=0.28 sec
 ...

Best model:  ARIMA(3,1,2)(1,0,1)[7]
Total fit time: 48.510 seconds
```

Select the aic index.

```
stepwise_model.aic()
```

Listing 21: Aic index selection.

```
3159.2872374206736
```

Now we fit the aic index to fit into our training dataset.

```
stepwise_model.fit(train_USA)
```

The output of the above script is:

```
ARIMA(maxiter=50, method='lbfgs', order=(3, 1, 2), out_of_sample_size=0,
          scoring='mse', scoring_args={}, seasonal_order=(1, 0, 1, 7),
          start_params=None, suppress_warnings=True, trend=None,
          with_intercept=False)
```

Let us now put the predicted data into an array and data frame.

```
future_forcast_USA = stepwise_model.predict(n_periods=len(test_USA))
future_forcast_USA
```

Listing 22: Code script for generating data array.

```
array([24341.40468779, 28328.80235144, 28152.73651492, 20187.28481612,
        7133.46277874, 23062.5796477 , 22870.05947277, 27702.26232137,
           29132.74697268, 30077.35816796, 20352.54894829,
    8320.44014901,
           22656.67785665, 23279.78854026, 26850.99206493,
    28978.68808144,
           28900.13423983, 19850.30262055,  7054.85530971,
    21823.52287947,
```

```
            21743.69167061, 25744.30510366, 27277.62445604,
    27588.11823221,
            18115.28322964,  5736.65160615, 20006.56789184,
    20211.56130514,
            23851.29489882, 25617.13328978, 25646.61944152,
    16442.84205489])
```

The array's plot is shown in Figure 69

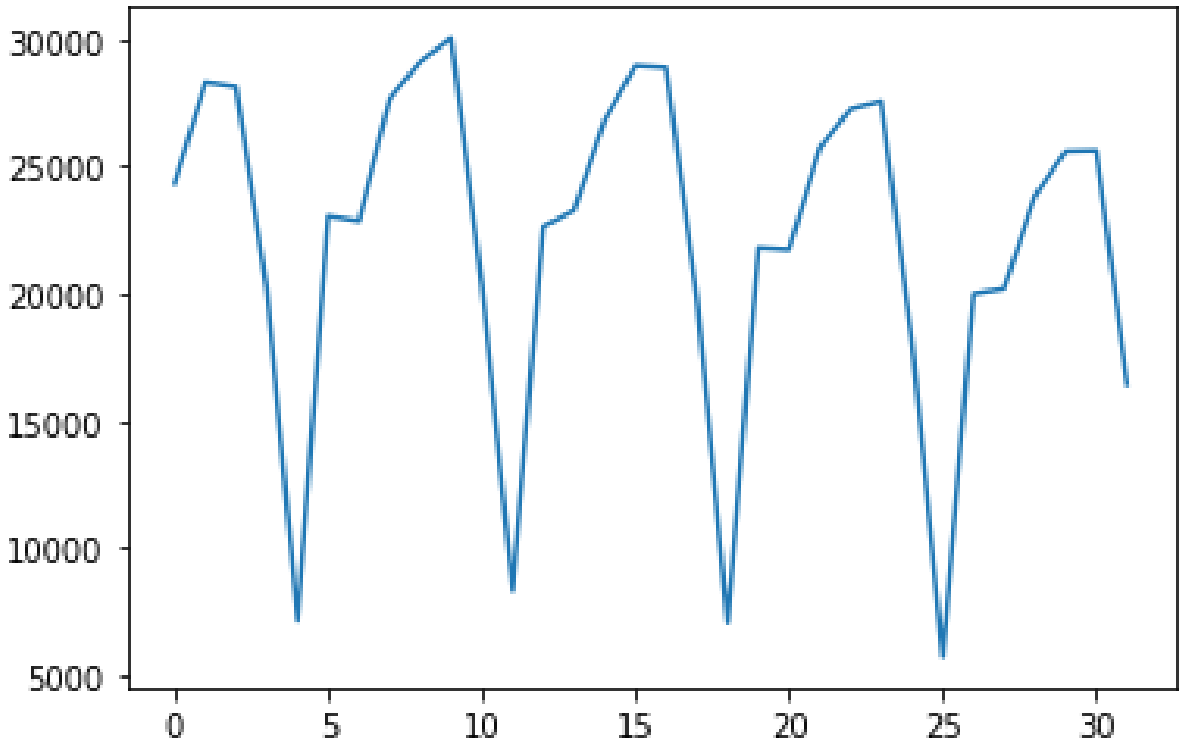


Figure 69: The future forecast in the USA.

```
future_forcast_USA_dataframe = pd.DataFrame(future_forcast_USA, index=
    test_USA.index, columns=['Prediction'])
future_forcast_USA_dataframe.head()
```

Listing 23: Code script for transform predicted data to data frame with the time index

```
     Prediction
date
2021-06-01  24341.404688
2021-06-02  28328.802351
2021-06-03  28152.736515
2021-06-04  20187.284816
2021-06-05  7133.462779
```

Let us calculate the mean absolute, mean squared, and root mean squared errors of the ARIMA prediction. The lower the MSE or MAE, the better the model fits the data, additionaly, we need to consider the overfitting situation by using some other techniques to avoid overfitting or underfitting like cross-validation test set, but in this excercise, we

have not implemented this technique because the time is short and the data is still not big enough.

```
mae_USA = mean_absolute_error(test_USA, future_forcast_USA_dataframe[0:
    len(test_USA)])
mse_USA = mean_squared_error(test_USA, future_forcast_USA_dataframe[0:
    len(test_USA)])
print("Mean absolute error: ", mae_USA)
print("Mean squared error: ", mse_USA)
print("Root mean squared error: ", np.sqrt(mse_USA))
```

Listing 24: Code script for calculating errors.

```
Mean absolute error:  10405.982765811981
Mean squared error:  141106979.36990243
Root mean squared error:  11878.845877016101
```

Next, we compare the predicted data with the test data as graphs, shown in Figure 70

```
plt.figure(figsize=(15,5))
plt.plot(test_USA, label="Actual")
plt.plot(future_forcast_USA_dataframe, label="Prediction")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

Listing 25: Data comparison.

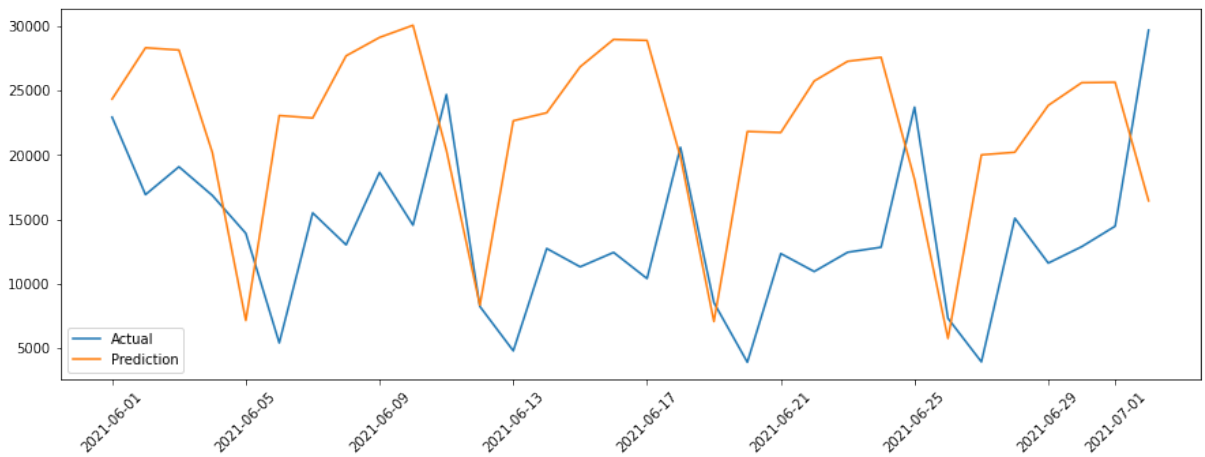


Figure 70: Comparison between test and predicted data.

We've noticed that Arima's prediction values are a day ahead of test values, but we haven't fixed this small issue.
From June 23th to July 23th, the figures would be:

```
start_time_future_arima = datetime.datetime(2021,6, 23) #2021,6, 22
end_time_future_arima = datetime.datetime(2021, 7, 23)  #2021, 6, 29
start_end_index = pd.date_range(start=start_time_future_arima,end=
    end_time_future_arima)
pred_next_N_days_Arima = stepwise_model.predict(len(start_end_index))
pred_N_days_Arima = pd.DataFrame(pred_next_N_days_Arima, index=
    start_end_index, columns=['Prediction'])
```

```
print(pred_N_days_Arima)
```

Listing 26: Predicted data for a month, from June 23th, 2021 to July 23th, 2021.

```
            Prediction
2021-06-23  24341.404688
2021-06-24  28328.802351
2021-06-25  28152.736515
2021-06-26  20187.284816
2021-06-27   7133.462779
2021-06-28  23062.579648
2021-06-29  22870.059473
2021-06-30  27702.262321
2021-07-01  29132.746973
2021-07-02  30077.358168
2021-07-03  20352.548948
2021-07-04   8320.440149
2021-07-05  22656.677857
2021-07-06  23279.788540
2021-07-07  26850.992065
2021-07-08  28978.688081
2021-07-09  28900.134240
2021-07-10  19850.302621
2021-07-11   7054.855310
2021-07-12  21823.522879
2021-07-13  21743.691671
2021-07-14  25744.305104
2021-07-15  27277.624456
2021-07-16  27588.118232
2021-07-17  18115.283230
2021-07-18   5736.651606
2021-07-19  20006.567892
2021-07-20  20211.561305
2021-07-21  23851.294899
2021-07-22  25617.133290
2021-07-23  25646.619442
```

Now, we compare everything together, which is seen in Figure 71.

```
plt.figure(figsize=(25,4))
plt.title("The prediction of new cases in USA with Arima")
plt.plot(new_cases_USA[start_time:end_time], label="Train")
plt.plot(new_cases_USA[end_time:], label="Test", color="orange")
plt.plot(future_forcast_USA_dataframe, label="Prediction", color="green"
    )
plt.plot(future_forcast_USA_dataframe - mae_USA, label="Prediction -
    mae_USA", color="red")
plt.plot(pred_N_days_Arima - mae_USA, label="Future Prediction", color="
    brown")
plt.xlabel("DATE")
plt.ylabel("NEW CASES")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

Listing 27: Code script for final comparison plotting.

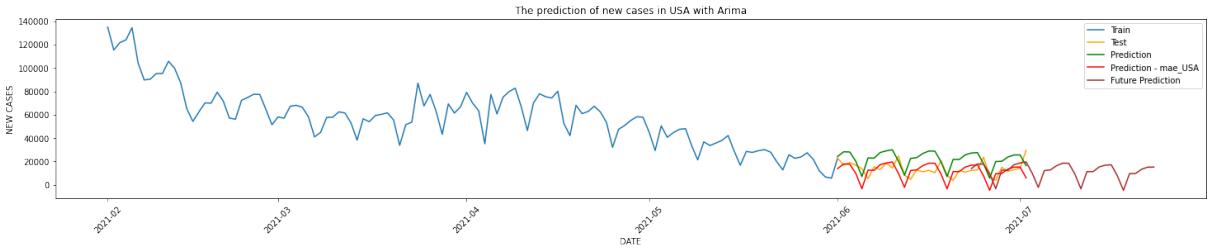


Figure 71: The comparison of the prediction with the actual data.

A close-up view is in Figure 72.

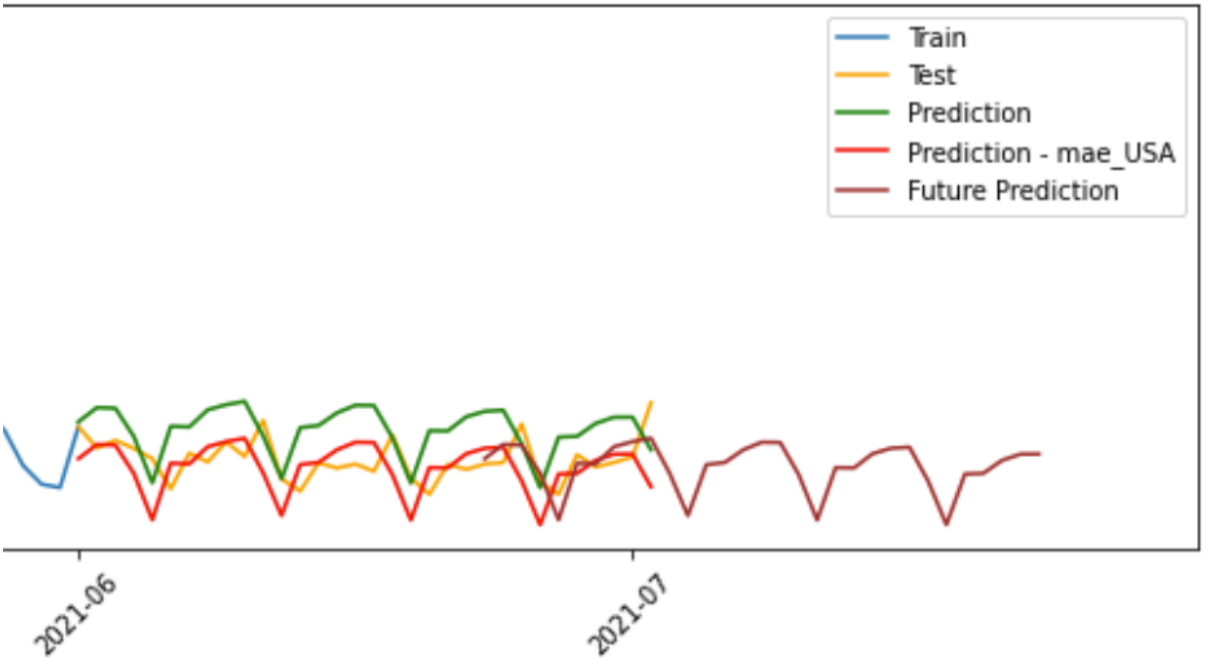
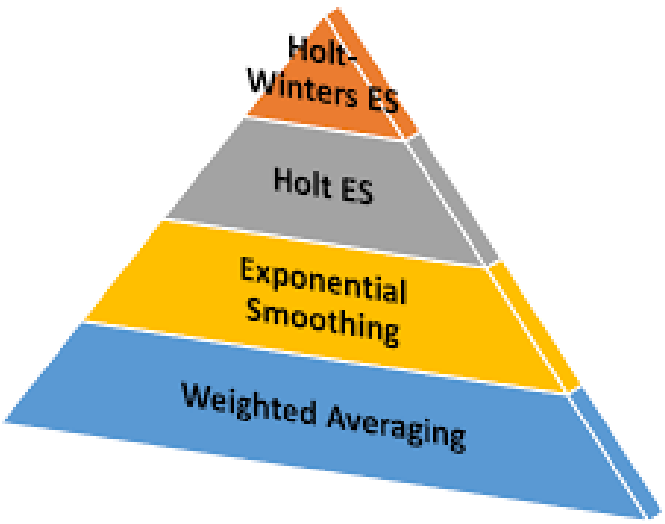


Figure 72: Close-up view of the comparison.

Based on personal experience, we can add or subtract Mean absolute error with predict values to give a more suitable prediction value.
This is only a small demonstration of how to implement the ARIMA model.
=> We will implement and demonstrate the results of together with the Holt Winters algorithms in chapter 6.10

6.9 Holt-Winters

Holt-Winter algorithm was first developed by Holt(1957) and was extended by Winters(1960) to capture the seasonality. This seasonal method consists of the forecast formula together with three smoothing equations, including one for the level l, one for the trend b, and one for the seasonal component s, with three corresponding smoothing parameters α, β, and γ. Most of the time, we will use m to denote the frequency of the seasonality or the number of seasons in a year. For example, for quarterly data $m = 4$, and for monthly data $m = 12$.



In using this method, there are, in fact, two variations in the nature of the seasonal component: additive and multiplicative. It is encouraged that the additive method be preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. With the additive method, the seasonal component is expressed in absolute terms in the scale of the observed series, and in the level equation the series is seasonally adjusted by subtracting the seasonal component. Within each year, the seasonal component will add up to approximately zero. With the multiplicative method, the seasonal component is expressed in relative terms (percentages), and the series is seasonally adjusted by dividing through by the seasonal component. Within each year, the seasonal component will sum up to approximately m.
Holt-Winters' additive method:
The component form for the additive method is:

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha\left(y_t - s_{t-m}\right) + (1-\alpha)\left(\ell_{t-1} + b_{t-1}\right) \\
b_t &= \beta^*\left(\ell_t - \ell_{t-1}\right) + \left(1-\beta^*\right) b_{t-1} \\
s_t &= \gamma\left(y_t - \ell_{t-1} - b_{t-1}\right) + (1-\gamma) s_{t-m}
\end{aligned}
$$

where k is the integer part of $(h-1)/m$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample. The level equation shows a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the nonseasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation is identical to Holt's linear method. The seasonal equation shows a weighted average between the current seasonal index, $(y_t - \ell_{t-1} - b_{t-1})$, and the seasonal index of the same season last year (i.e., m time periods ago). The equation for the seasonal component is often expressed as

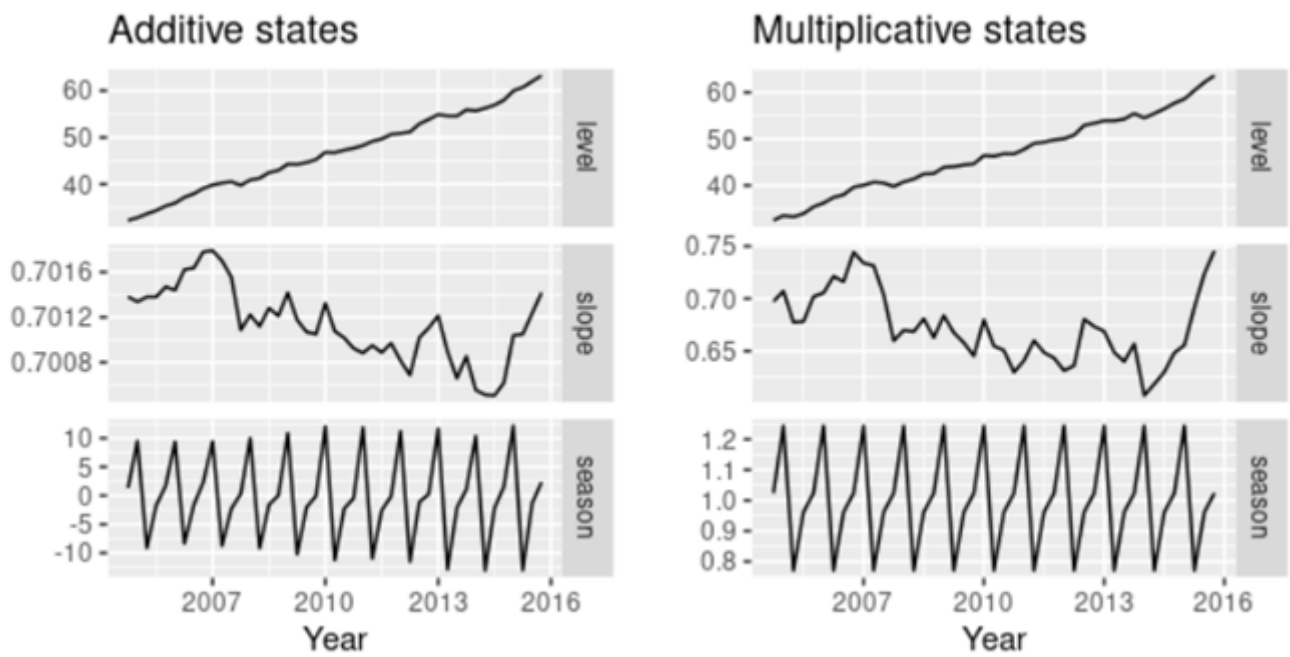
$$s_t = \gamma^* (y_t - \ell_t) + (1 - \gamma^*) s_{t-m}$$

If we substitute ℓ_t from the smoothing equation for the level of the component form above, we get

$$s_t = \gamma^*(1-\alpha)(y_t - \ell_{t-1} - b_{t-1}) + [1 - \gamma^*(1-\alpha)] s_{t-m}$$

which is identical to the smoothing equation for the seasonal component we specify here, with $\gamma = \gamma^*(1-\alpha)$. The usual parameter restriction is $0 \leq \gamma^* \leq 1$, which translates to $0 \leq \gamma \leq 1 - \alpha$ Holt-Winters' multiplicative method The component form for the multiplicative method is:

$$\begin{aligned}
\hat{y}_{t+h|t} &= (\ell_t + hb_t) s_{t+h-m(k+1)} \\
\ell_t &= \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^* (\ell_t - \ell_{t-1}) + (1-\beta^*) b_{t-1} \\
s_t &= \gamma \frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma) s_{t-m}
\end{aligned}$$



6.10 Holt-Winters MODEL

We will build our the Holt-Winters Model to predict global figures. All the steps of data loading, pre-processing, and seasonality decomposition are pretty much the same as ARIMA, so we will go straight to the train-test splitting phase and model building.

```
train_World = new_cases_World.loc[start_time_world:end_time_world]  # train & test split
test_World = new_cases_World.loc[end_time_world:]
print("Train length: ", len(train_World))
print("Test length: ",len(test_World))
print("Ratio: ", (len(test_World))/ (len(test_World)+len(train_World)))
```

Listing 28: Code script for splitting data.

The result of the above code script is:

```
Train length:  121
Test length:  32
Ratio:  0.20915032679738563
```

Now we proceed to building the Holt-Winters model.

```
HoltWinter_model_mul_World = ExponentialSmoothing(train_World, seasonal='mul', seasonal_periods=7).fit()
HoltWinter_model_add_World = ExponentialSmoothing(train_World, seasonal='add', seasonal_periods=7).fit()
pred_mul_World = HoltWinter_model_mul_World.predict(start=test_World.index[0], end= test_World.index[-1])
pred_add_World = HoltWinter_model_add_World.predict(start=test_World.index[0], end= test_World.index[-1])
```

Listing 29: Code script for train-test data splitting.

Let us calculate the mean squared errors and mean absolute errors for the additive and multiplicative method.

```
mae_World_mul_hw = mean_absolute_error(test_World, pred_mul_World[0:len(test_World)])
mse_World_mul_hw = mean_squared_error(test_World, pred_mul_World[0:len(test_World)])
mae_World_add_hw = mean_absolute_error(test_World, pred_add_World[0:len(test_World)])
mse_World_add_hw = mean_squared_error(test_World, pred_add_World[0:len(test_World)])
print("MAE Mul:  ", mae_World_mul_hw)
print("MSE Mull: ",mse_World_mul_hw)
print("MAE Add:  ", mae_World_add_hw)
print("MSE Add:  ", mse_World_add_hw)
```

Listing 30: Code script for calculating mean absolute errors and mean squared errors.

Which returns the following result:

```
MAE Mul:   69077.94465613799
MSE Mull:  5626836803.418269
MAE Add:   84357.75692708419
MSE Add:   8020232372.650858
```

Now we can graphically illustrate the prediction of multiplicative and additive Holt-Winters and compare them to the test data.

```
plt.figure(figsize=(20,4))
plt.title("The prediction of global new cases with Holt Winters")
plt.plot(train_World.index, train_World, label='Train')
plt.plot(test_World.index, test_World, label='Test')
plt.plot(pred_mul_World.index, pred_mul_World - mae_World_mul_hw, label=
    'Holt-Winters (mul)')
plt.plot(pred_add_World.index, pred_add_World - mae_World_add_hw, label=
    'Holt-Winters (add)')
plt.legend(loc="best")
plt.xticks(rotation=45)
plt.show()
```

Listing 31: Code script for comparison with test data.

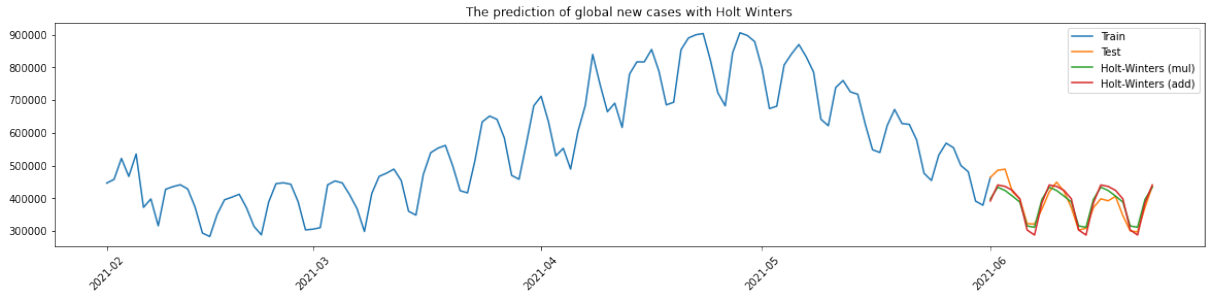


Figure 73: Holt-Winters with multiplicative and additive seasonality.

Next up, the Holt-Winters model is used to predict future figures.

```
#SELECT TIME TO PREDICT
start_time_future = datetime.datetime(2021,6, 22) #2021,6, 22
end_time_future = datetime.datetime(2021, 7, 22)  #2021, 6, 29
pred_next_N_days_add_World = HoltWinter_model_add_World.predict(start=
    start_time_future, end=end_time_future)
pred_next_N_days_mul_World = HoltWinter_model_mul_World.predict(start=
    start_time_future, end=end_time_future)
pred_N_add_World = pd.Series(pred_next_N_days_add_World)
pred_N_mul_World = pd.Series(pred_next_N_days_mul_World)
```

Listing 32: Code script for future prediction with Holt-Winters model.

A comparison between additive and multiplicative seasonality is conducted with the following code block.

```
plt.figure(figsize=(10,5))
plt.title("COMPARE FUTURE PREDICTION BETWEEN Additive Seasonality AND
    Multiplicative Seasonality")
plt.plot(pred_next_N_days_add_World, label="Holt-Winters Additive
    Seasonality")
plt.plot(pred_next_N_days_mul_World, label="Holt-Winters Multiplicative
    Seasonality")
plt.xlabel("DATE")
plt.ylabel("NEW CASES")
plt.legend()
plt.xticks(rotation=45)
```

```
plt.show()
```

Listing 33: Code script to generate graphic illustration of two seasonality choices of Holt-Winters model.

The given code script produces Figure 74.

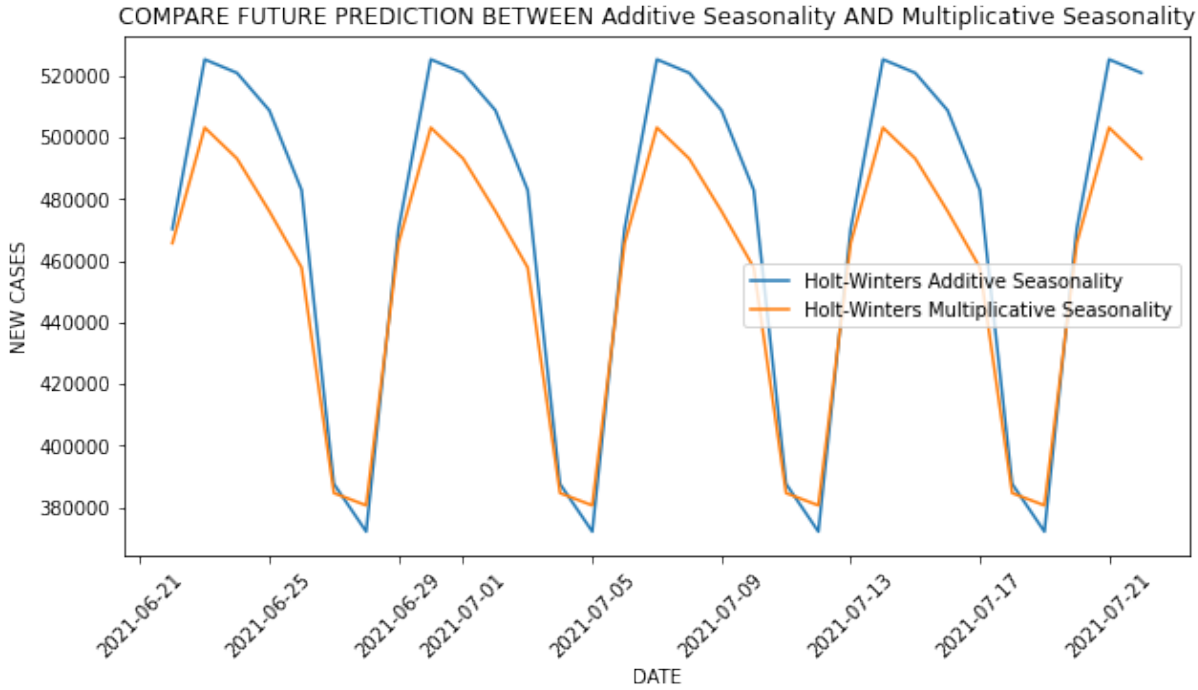
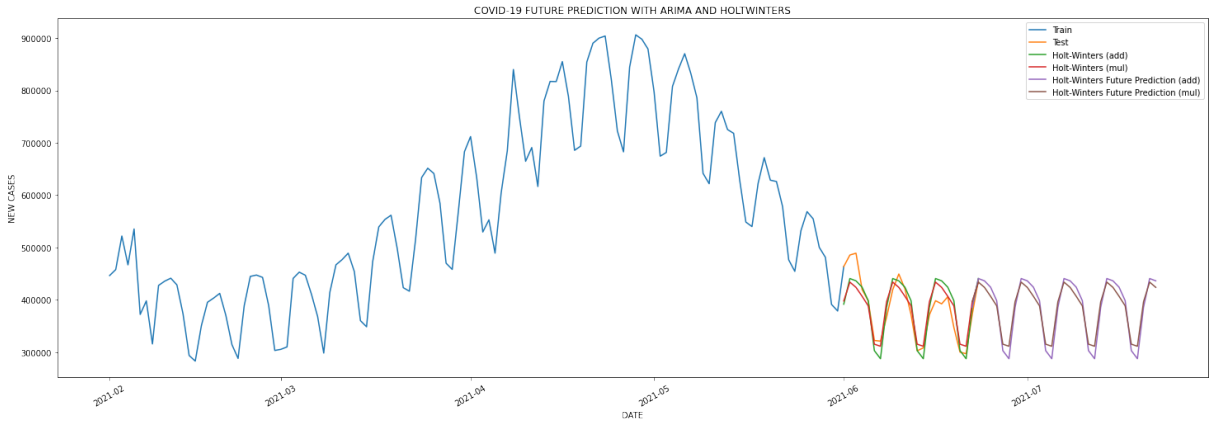


Figure 74: Graphic illustration of two seasonality choices of Holt-Winters model.

```
plt.figure(figsize=(25,8))
plt.title("COVID-19 FUTURE PREDICTION WITH ARIMA AND HOLTWINTERS")
plt.plot(train_World.index, train_World, label='Train')
plt.plot(test_World.index, test_World, label='Test')
plt.plot(pred_add_World.index, pred_add_World - mae_World_add_hw,  label
    ='Holt-Winters (add)')
plt.plot(pred_mul_World.index, pred_mul_World - mae_World_mul_hw,  label
    ='Holt-Winters (mul)')
plt.plot(pred_next_N_days_add_World.index, pred_next_N_days_add_World.
    values - mae_World_add_hw, label="Holt-Winters Future Prediction (add
    )")
plt.plot(pred_next_N_days_mul_World.index, pred_next_N_days_mul_World.
    values - mae_World_mul_hw, abel="Holt-Winters Future Prediction (mul)
    ")
plt.xlabel("DATE")
plt.ylabel("NEW CASES")
plt.legend(loc="best")
plt.xticks(rotation=30)
plt.show()
```



=> Because the Holt Winters algorithm gives the best results according to the updated data, we will only build the world pandemic prediction using this algorithm. For other algorithms, we only build model to predict the pandemic situation of only countries.

6.11 Time Series Results (ARIMA + Holt-Winters)

6.11.1 Korea

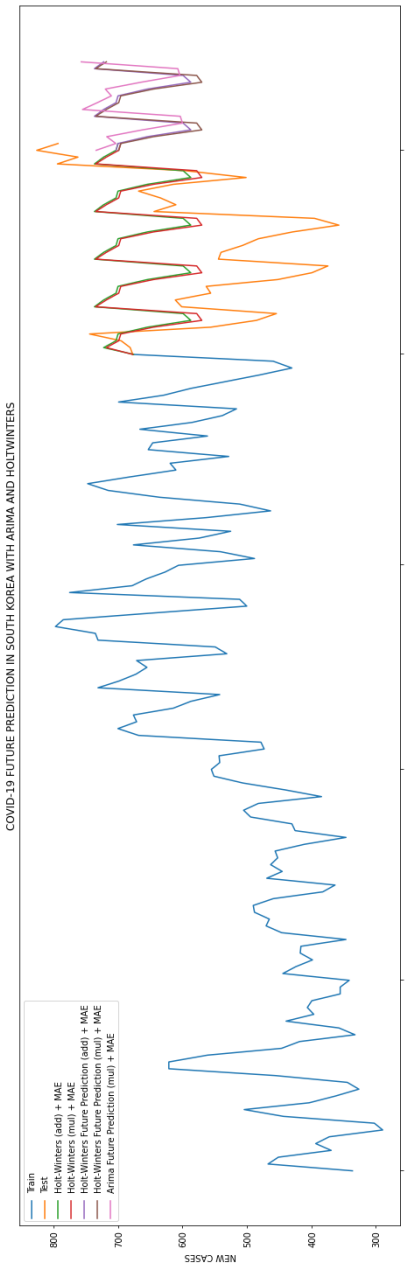


Figure 75: Prediction for Korea using ARIMA and Holt Winters

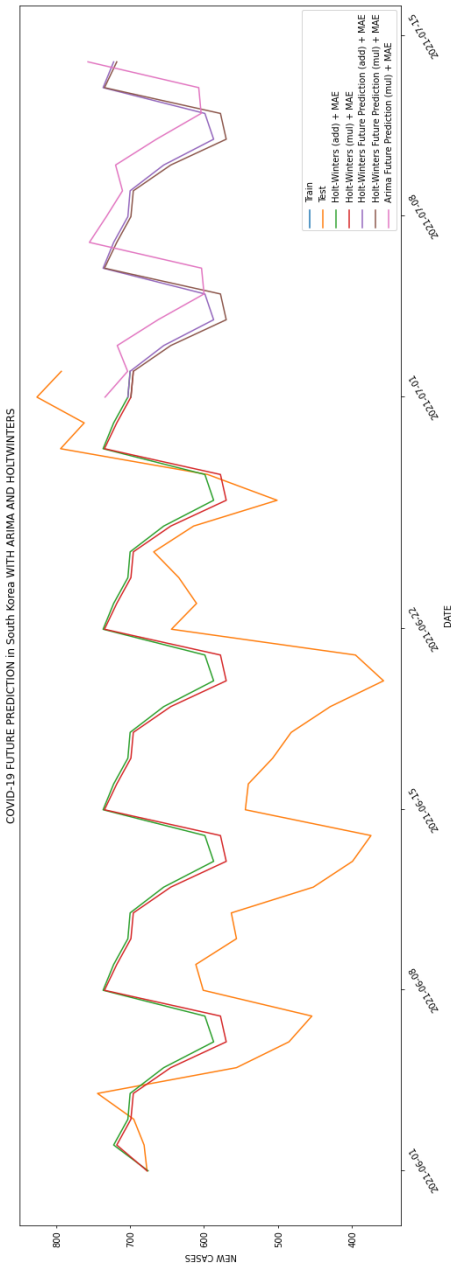


Figure 76: Korea zoom out

6.11.2 Brazil

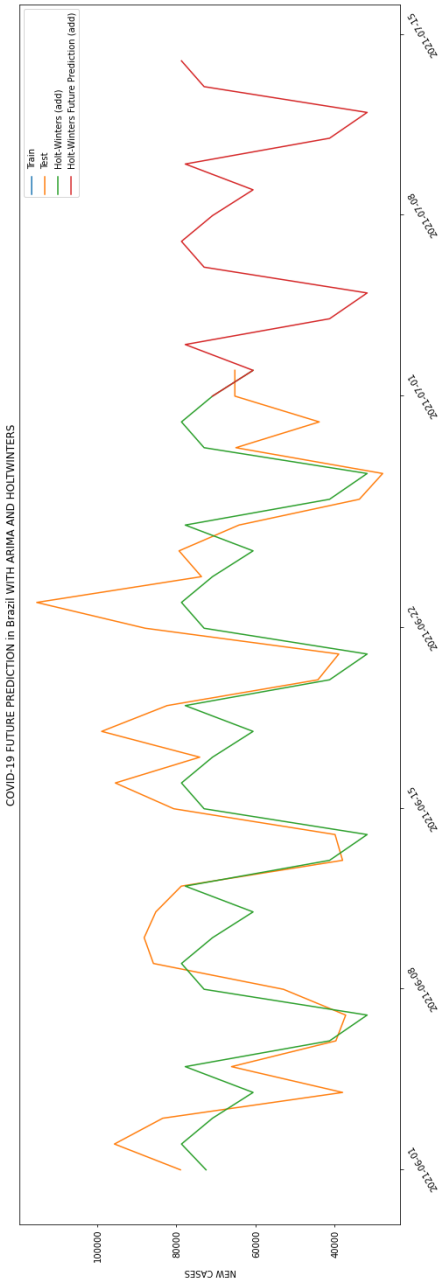


Figure 77: Prediction for Brazil using ARIMA and Holt Winters

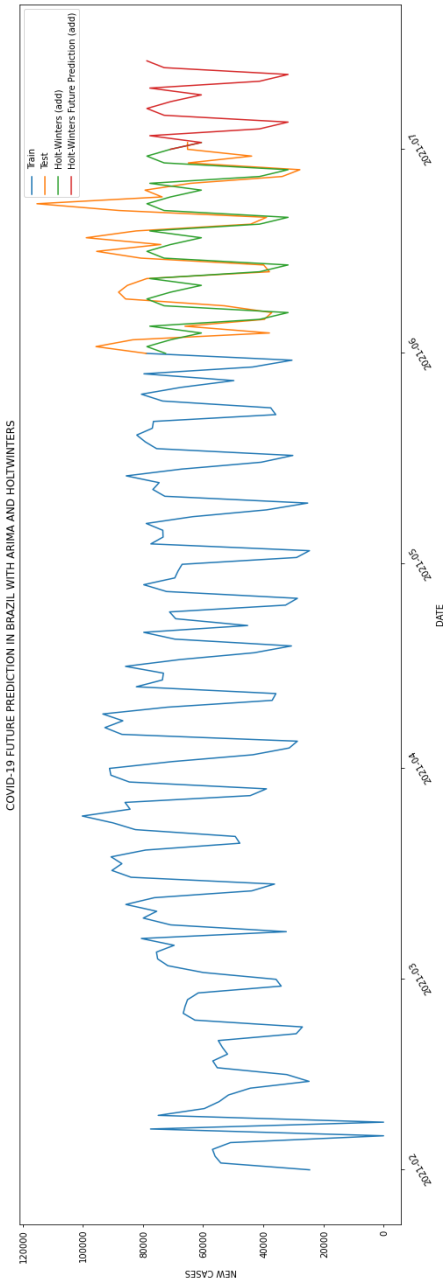


Figure 78: Brazil zoom out

6.11.3 India

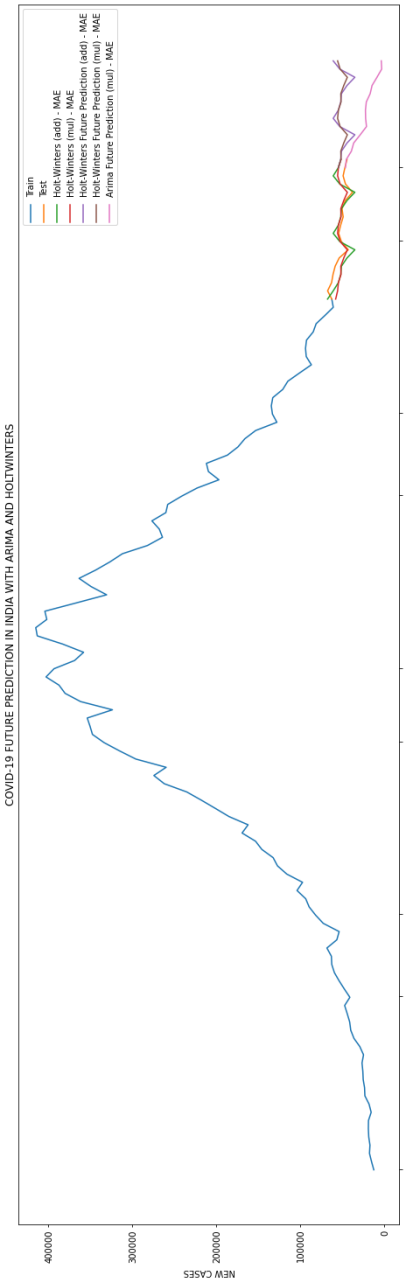


Figure 79: Prediction for India using ARIMA and Holt Winters

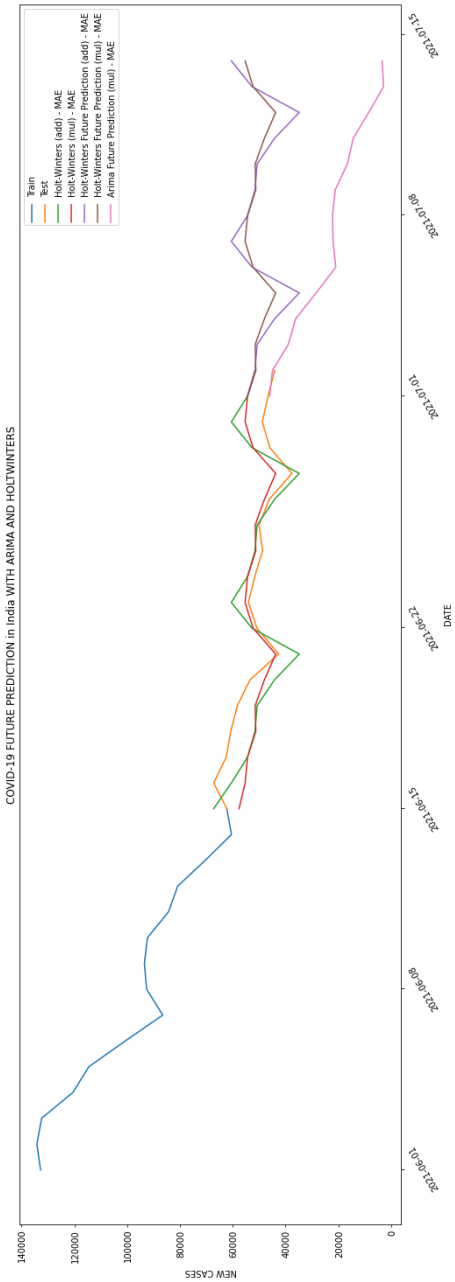


Figure 80: India zoom out

6.11.4 Japan

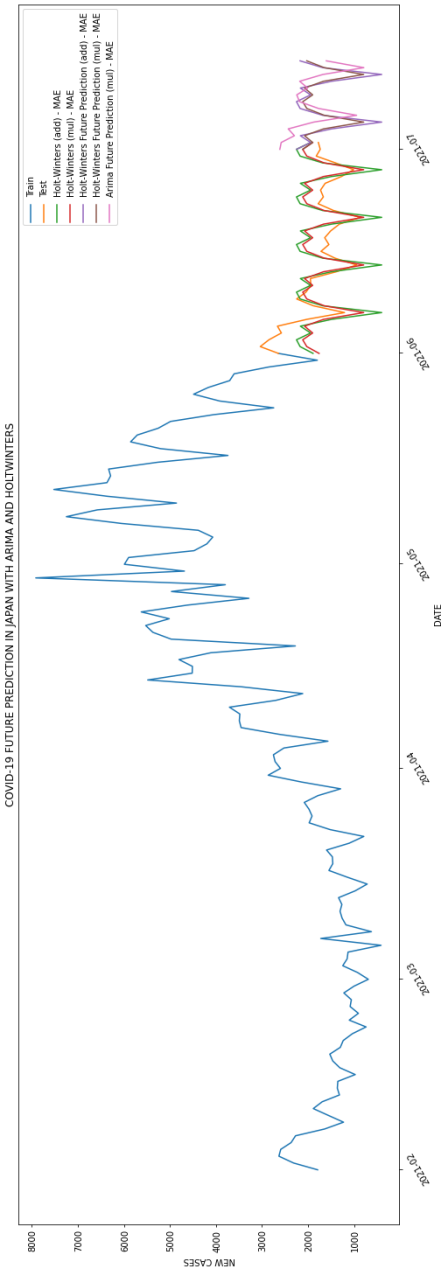


Figure 81: Prediction for Japan using ARIMA and Holt Winters

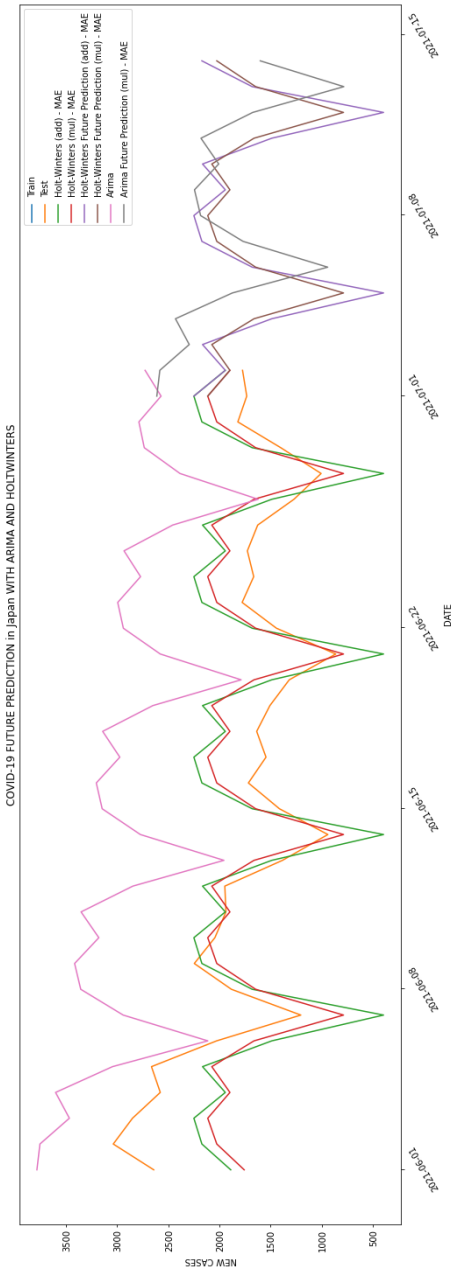


Figure 82: Japan zoom out

6.11.5 Vietnam

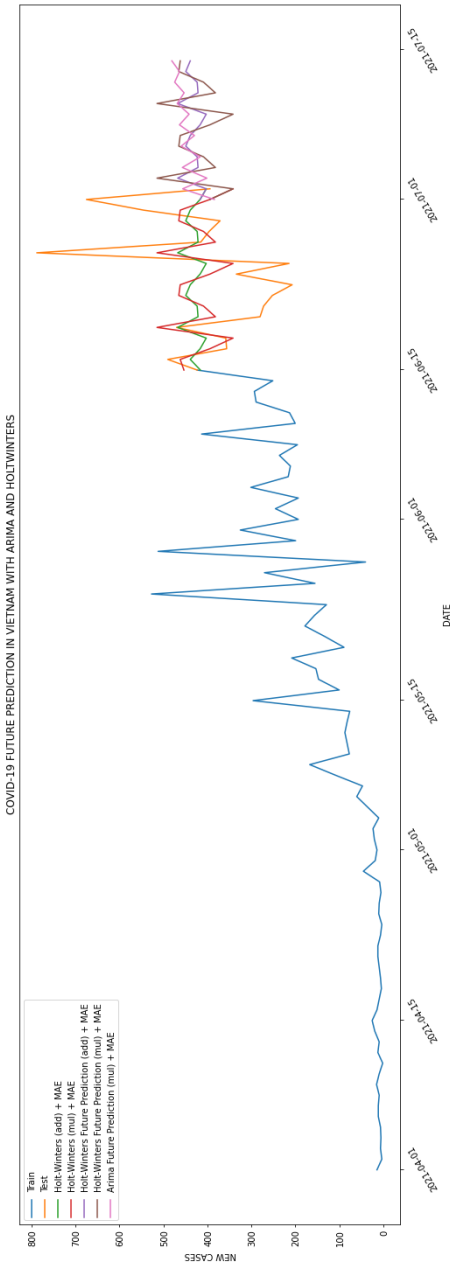


Figure 83: Prediction for Vietnam using ARIMA and Holt Winters

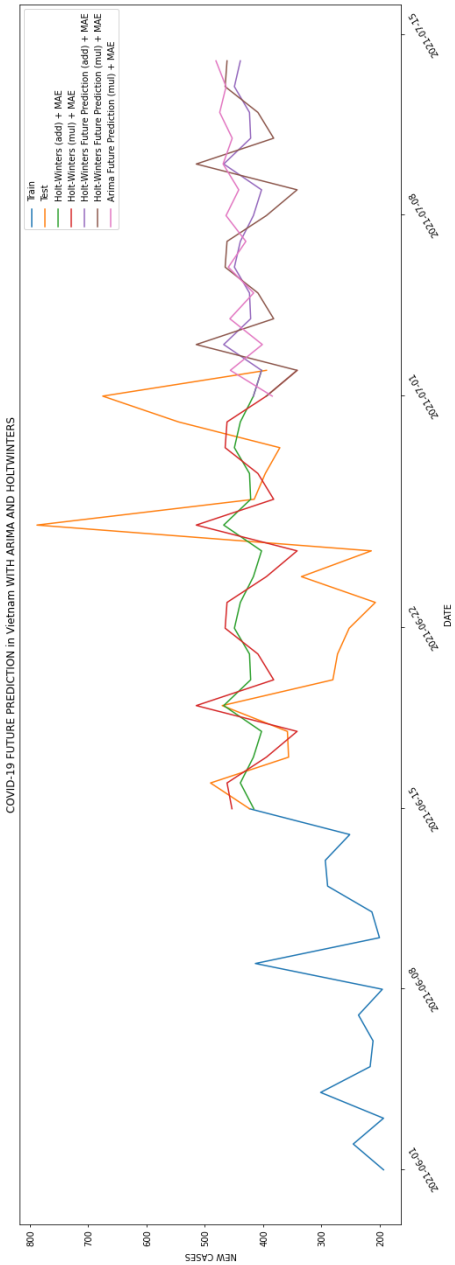


Figure 84: Vietnam zoom out

6.12 SIR model theory

The SIR model is one of the popular mathematical models that utilize the differential equations to describe the dynamics of the coronavirus. The SIR model is also known as the "compartmental model" because it classifies the population into groups named S(Susceptible), I(Infected), and R (Recovered).

Susceptible includes a group of population that are vulnerable to the exposure with the disease. We assume that when a nation has one or more citizens infected with the coronavirus, then the rest of the population will be considered to be susceptible because before the infectious people are confirmed, they might travel across the country and meet various other people.

Infected is the group of people who have been confirmed to be a victim of the coronavirus. These people will be treated under the surveillance of doctors. Also, we should note that this group does not include the people who are suspected to have caught the disease but testing over the next 21 days yields negative results.

Recovered group is the people who either once belonged to the group Infected but got the immunity or those who passed away because of serious illness. Once they are fatal or recover, they will not be counted in the infected group anymore.

In summary:

- S: Susceptible = Population – Confirmed
- I : Infected = Confirmed – Recovered – Fatal
- R : Recovered or Fatal (Recovered + Fatal)

The framework of SIR model:

Here, B stands for the coefficient of how much the disease can be transmitted through exposure and is calculated by the chance of contact and the probability of disease. In short, B is the chance that a person catches the disease. Y, as much the same as B, demonstrates the probability that an infected person will recover or become fatal. A noticeable remark is that we do not have any transmission from R back to S because when we recover, we are immune from the disease, so we are no longer considered to be susceptible, the same

is applied for fatal situations.

Having compartmentalized the population, we now move on to develop a set of differential equations to determine the sizes of each group over time.

First, we have the relationship between groups as the following systems:

$$\begin{cases} S(n+1) = S(n) - \beta \cdot S(n) \cdot \dfrac{I(n)}{N} \\ I(n+1) = I(n) - \gamma \cdot I(n) + \beta(n) \cdot \dfrac{I(n)}{N} \\ R(n+1) = R(n) + \gamma \cdot I(n) \end{cases}$$

where:

- S(n) is the susceptible population in week n.
- I(n) is the infected population in week n.
- R(n) is the recovered population in week n.
- β: The number of people that each infectious person may meet.
- γ: the chance of recovering.
- N: the nation population.

From the listed equations, when we take derivatives of both sides with respect to the changes in time, we obtain:

$$\begin{cases} \frac{dS}{dt} = \dfrac{-\beta SI}{N} \\ \frac{dI}{dt} = -\gamma I + \dfrac{\beta SI}{N} \\ \frac{dR}{dy} = \gamma I \end{cases}$$

which means that

$$\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dy} = 0 \tag{4}$$

Note:

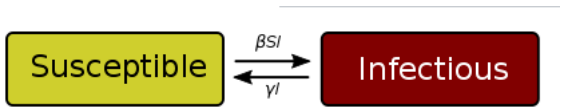
- The minus mark on the right side of the first derived equation demonstrates the downsizing susceptible group when they meet the people in the infected group.
- Following that, the next function illustrates the population of the infected group changes to a number equal to those who are susceptible and catch the disease and those who are infected but recovered.

6.13 Some other compartmental models of SIR

As the ultimate goal of models is to correctly predict such behavior patterns of the disease, the total number of infected patients, or the epidemic duration, and to estimate some epidemiological parameters like the reproductive number, numerous other public health interventions should be taken into account to ensure the correctness of the SIR model such as:

6.13.1 SIS Model

This model considers some other infections that are likely to be viewed as coronavirus but turn out just to be the common cold or influenza and do not confer any long-lasting immunity. Because of the lack of immunity, these once infected patients can still be susceptible to the coronavirus.
The differential equations of this model are:



$$\begin{cases} \frac{dS}{dt} = \dfrac{-\beta SI}{N} + \gamma I \\ \frac{dI}{dt} = \dfrac{\beta SI}{N} - \gamma I \end{cases}$$

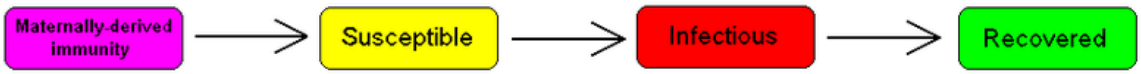
6.13.2 SIRD Model

This model is just the same as our original model with just a small differentiation in the Recovered group between those recovered and those that become fatal.

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta IS}{N} \\ \frac{dI}{dt} = -\frac{\beta IS}{N} - \gamma I - \mu I \\ \frac{dR}{dt} = \gamma I \\ \frac{dD}{dt} = \mu I \end{cases}$$

6.13.3 MSIR Model

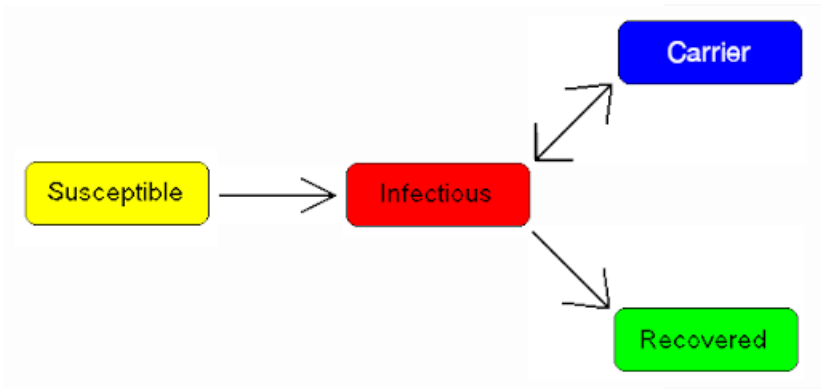
The letter M in this model represents 'maternal', this means that the model also considers that new babies are not born to the susceptible compartment. However, they do receive the immunity generated by the mothers to survive for the first few months of life.



$$
\begin{aligned}
\frac{dM}{dt} &= \Lambda - \delta M - \mu M \\
\frac{dS}{dt} &= \delta M - \frac{\beta S I}{N} - \mu S \\
\frac{dI}{dt} &= \frac{\beta S I}{N} - \gamma I - \mu I \\
\frac{dR}{dt} &= \gamma I - \mu R
\end{aligned}
\tag{5}
$$

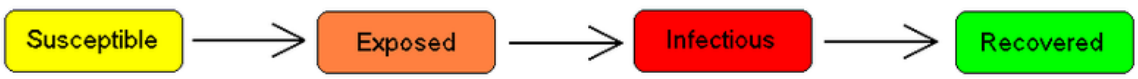
6.13.4 SICR Model

The Carriers are the people who have caught respiratory diseases but never completely recover. Consequently, they carry the infection, while they themselves do not suffer from the disease, they can be infected again or infect others people in the carrier state.



6.13.5 SEIR Model

The additional 'exposed' group illustrates the period when the people have been infected, but they are not infectious themselves.



$$
\begin{aligned}
\frac{dS}{dt} &= \mu N - \mu S - \frac{\beta I S}{N} \\
\frac{dE}{dt} &= \frac{\beta I S}{N} - (\mu + a)E \\
\frac{dI}{dt} &= aE - (\gamma + \mu)I \\
\frac{dR}{dt} &= \gamma I - \mu R
\end{aligned}
\tag{6}
$$

6.13.6 SEIS Model:

The SEIS is basically the same as SEIR, except that the infected people will not get the immunity, so he or she might be re-infected and classified as a susceptible person again.

$$\mathcal{S} \to \mathcal{E} \to \mathcal{I} \to \mathcal{S}$$

$$
\begin{aligned}
\frac{dS}{dt} &= \Lambda - \frac{\beta S I}{N} - \mu S \dashv \cdot \gamma I \\
\frac{dE}{dt} &= \frac{\beta S I}{N} - (\epsilon - \mu)E \\
\frac{dI}{dt} &= \varepsilon E - (\gamma + \mu)I
\end{aligned}
\tag{7}
$$

6.13.7 MSEIR Model

This is the combination of MSIR and SEIR models.

$$\mathcal{M} \to \mathcal{S} \to \mathcal{E} \to \mathcal{I} \to \mathcal{R}$$

$$
\begin{aligned}
\frac{dM}{dt} &= \Lambda - \delta M - \mu M \\
\frac{dS}{dt} &= \delta M - \frac{\beta S I}{N} - \mu S \\
\frac{dE}{dt} &= \frac{\beta S I}{N} - (\varepsilon + \mu)E \\
\frac{dI}{dt} &= \varepsilon E - (\gamma + \mu)I \\
\frac{dR}{dt} &= \gamma I - \mu R
\end{aligned}
$$

6.13.8 MSEIRS Model

This is the combination of MSEIR and SEIS models.

$$\mathcal{M} \rightarrow \mathcal{S} \rightarrow \mathcal{E} \rightarrow \mathcal{I} \rightarrow \mathcal{R} \rightarrow \mathcal{S}$$

6.14 SIR MODEL

6.14.1 Global

First, we will import some packages that we will use to build the SIR model, including pandas, numpy, matplotlib, odeint, and scipy.stats

```
# Import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
import scipy.stats
```

Next, we connect the Github link to our Colab as the data can be updated daily automatically instead of manually.

```
# Upload the link of GitHub
country_cases_file_link = 'https://raw.githubusercontent.com/
    CSSEGISandData/COVID-19/web-data/data/cases_country.csv'
population_file_link = 'https://raw.githubusercontent.com/owid/covid-19-
    data/master/scripts/input/un/population_2020.csv'
```

At this step, we transform the given link csv file into a dataframe by using pd.Dataframe

```
# Create dataframes based on the 2 links uploaded
cases_country_df = pd.read_csv(country_cases_file_link)
print(cases_country_df.shape) # Print the shape

population_df = pd.read_csv(population_file_link)
print(population_df.shape) # Print the shape
```

```
# Create dataframes based on the 2 links uploaded
cases_country_df = pd.read_csv(country_cases_file_link)
print(cases_country_df.shape) # Print the shape

population_df = pd.read_csv(population_file_link)
print(population_df.shape) # Print the shape
```

```
(195, 14)
(252, 4)
```

Next, we will try to print the dataframe to have an overview about how our data is arranged initally.

```
cases_country_df = cases_country_df.replace(np.nan, '', regex=True) # Preprocess data
cases_country_df # Print to view the dataframe
```

[5] cases_country_df # Print to view the dataframe

	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Incident_Rate
0	Afghanistan	2021-07-03 08:21:18	33.9391	67.71	120216.0	4962.0	72700	42554	308.814
1	Albania	2021-07-03 08:21:18	41.1533	20.1683	132526.0	2456.0	130018	52	4605.12
2	Algeria	2021-07-03 08:21:18	28.0339	1.6596	140550.0	3735.0	97701	39114	320.517
3	Andorra	2021-07-03 08:21:18	42.5063	1.5218	13918.0	127.0	13721	70	18013.3
4	Angola	2021-07-03 08:21:18	-11.2027	17.8739	39089.0	906.0	33375	4808	118.933
...	...	...	...	...	...	...	...	...	...
190	Vietnam	2021-07-03 08:21:18	14.0583	108.277	18360.0	84.0	7395	10881	18.862
191	West Bank and Gaza	2021-07-03 08:21:18	31.9522	35.2332	314409.0	3570.0	308413	2426	6163.17
192	Yemen	2021-07-03 08:21:18	15.5527	48.5164	6924.0	1361.0	4082	1481	23.2147
193	Zambia	2021-07-03 08:21:18	-13.1339	27.8493	160613.0	2341.0	137819	20453	873.659
194	Zimbabwe	2021-07-03 08:21:18	-19.0154	29.1549	52663.0	1841.0	39568	11254	354.325

195 rows × 14 columns

Here, we pre-process the data and take only necessary columns to build our model, which are Country/Region, Confirmed, Deaths, Recovered, Active, Incident Rate, and Mortality Rate. Then, since the four columns Recovered, Active, Incident Rate, and Mortality Rate is recorded as objects instead of floating point numbers, we call pd.to numeric() to turn them into float.

```
# Take necessary columns to create the global_data dataframe
global_data = cases_country_df[['Country_Region','Confirmed','Deaths','Recovered','Active', 'Incident_Rate', 'Mortality_Rate']]
global_data['Recovered'] = pd.to_numeric(global_data['Recovered'], errors='ignore')
global_data['Active'] = pd.to_numeric(global_data['Active'], errors='ignore')
global_data['Incident_Rate'] = pd.to_numeric(global_data['Incident_Rate'], errors='ignore')
global_data['Mortality_Rate'] = pd.to_numeric(global_data['Mortality_Rate'], errors='ignore')

# Global_summary is the sum of global_data
global_summary = pd.DataFrame(global_data.sum()).transpose()
global_summary
```

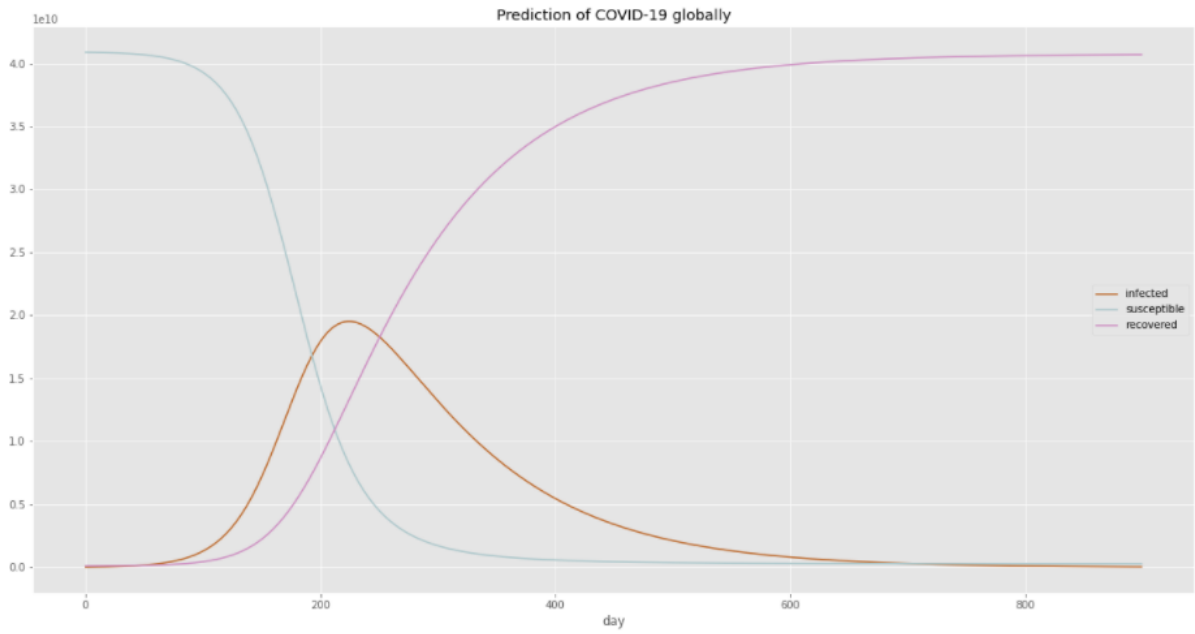
Next, we build the three differential equations of the SIR model by implementing a function and named it ODE. The input parameters include state, t, N for population, beta as the transmission rate, and gamma as the recovery rate.

```
# SIR model differentiation equations function:
def ode(state, t, N, beta, gamma):
    S, I, R = state
    # Change in S population over time
    dSdt = -beta * S * I / N
```

```
    # Change in I population over time
    dIdt = beta * S * I / N - gamma * I
    # Change in R population over time
    dRdt = gamma * I
    return dSdt, dIdt, dRdt
```

Here is the main calculation of our model, as we research online, the beta and gamma parameters of the COVID-19 disease is 0.05 and 0.01 respectively. Then we use data to find the susceptible, infected, and recovered group members, after which we put these three parameters to the ode function we have just built. Finally, we calculation is done, we use df.plot to plot the diagram.

```
effective_contact_rate = 0.05 # = transmission rate * contact rate
recovery_rate = 0.01      # = 1 / days

print("R0 is", effective_contact_rate / recovery_rate)

# State population in SIR model
# Everyone not infected or recovered is susceptible
total_pop = population_df['population'].sum()
recovered = (global_summary['Recovered'][0] + global_summary['Deaths'
    ][0])
infected = global_summary['Active'][0]
susceptible = total_pop - infected - recovered

# A list of days, 0 -> 160
days = range(0,900)

# Use differential equations with the model
ret = odeint(ode, [susceptible,infected, recovered],days,args=(total_pop
    , effective_contact_rate, recovery_rate))
S, I, R = ret.T

# Build dataframe
df = pd.DataFrame({
        'susceptible': S,
        'infected': I,
        'recovered': R,
        'day': days
    })

plt.style.use('ggplot')
df.plot(title = 'Prediction of COVID-19 globally',
            figsize=(10,5),
            x='day',
            y=['infected', 'susceptible', 'recovered'],
            color=['#bb6424', '#aac6ca', '#cc8ac0'],
            kind='line',
            stacked=False)
```



We use this block to output the result by days.

```
df
```

```
df
```

	susceptible	infected	recovered	day
0	4.087767e+10	2.538780e+07	1.242670e+08	0
1	4.087638e+10	2.641906e+07	1.245260e+08	1
2	4.087504e+10	2.749216e+07	1.247956e+08	2
3	4.087364e+10	2.860880e+07	1.250760e+08	3
4	4.087219e+10	2.977074e+07	1.253679e+08	4
...	...	...	...	...
895	2.914280e+08	4.753234e+07	4.068837e+10	895
896	2.914112e+08	4.707610e+07	4.068884e+10	896
897	2.913946e+08	4.662424e+07	4.068931e+10	897
898	2.913781e+08	4.617672e+07	4.068977e+10	898
899	2.913618e+08	4.573349e+07	4.069023e+10	899

900 rows × 4 columns

6.14.2 Vietnam

```
country_list = np.array(cases_country_df['Country_Region'])
country_list
```

```
country_list = np.array(cases_country_df['Country_Region'])
country_list
```

```
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
       'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
       'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia',
       'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica',
       "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
       'Diamond Princess', 'Djibouti', 'Dominica', 'Dominican Republic',
       'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
       'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France',
       'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
       'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana',
       'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
       'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
       'Korea, South', 'Kosovo', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia',
       'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Liechtenstein',
       'Lithuania', 'Luxembourg', 'MS Zaandam', 'Madagascar', 'Malawi',
       'Malaysia', 'Maldives', 'Mali', 'Malta', 'Marshall Islands',
       'Mauritania', 'Mauritius', 'Mexico', 'Micronesia', 'Moldova',
       'Monaco', 'Mongolia', 'Montenegro', 'Morocco', 'Mozambique',
       'Namibia', 'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua',
       'Niger', 'Nigeria', 'North Macedonia', 'Norway', 'Oman',
       'Pakistan', 'Palau', 'Panama', 'Papua New Guinea', 'Paraguay',
       'Peru', 'Philippines', 'Poland', 'Portugal', 'Qatar', 'Romania',
       'Russia', 'Rwanda', 'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
       'Spain', 'Sri Lanka', 'Sudan', 'Summer Olympics 2020', 'Suriname',
       'Sweden', 'Switzerland', 'Syria', 'Taiwan*', 'Tajikistan',
       'Tanzania', 'Thailand', 'Timor-Leste', 'Togo',
       'Trinidad and Tobago', 'Tunisia', 'Turkey', 'US', 'Uganda',
       'Ukraine', 'United Arab Emirates', 'United Kingdom', 'Uruguay',
       'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',
       'West Bank and Gaza', 'Yemen', 'Zambia', 'Zimbabwe'], dtype=object)
```

```
country = input("Which countries would you like? : ")
```

```
df_country = global_data[global_data['Country_Region'] == country]
df_country
```

```
[11] country = input("Which countries would you like? : ")
```

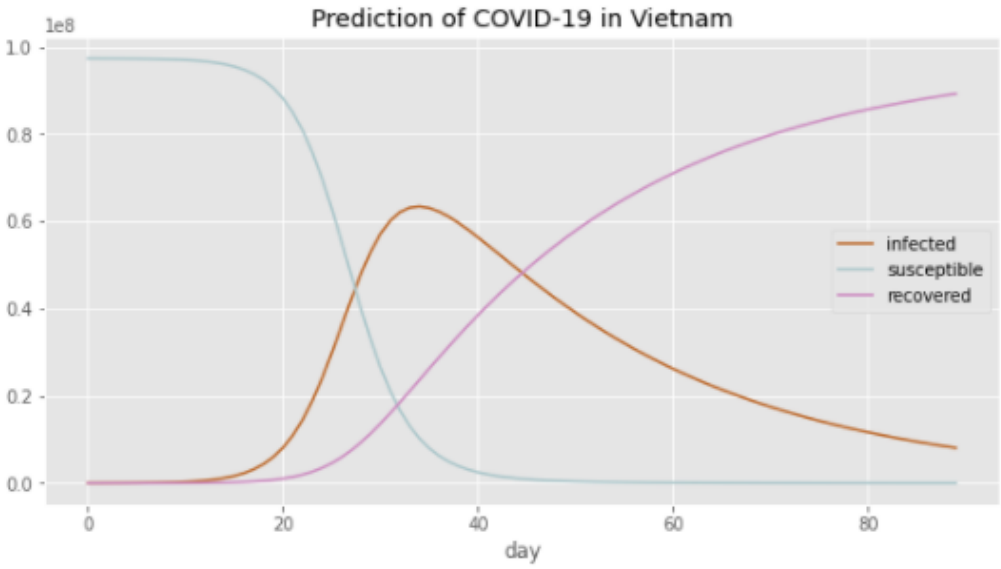
```
Which countries would you like? : Vietnam
```

```
[12] df_country = global_data[global_data['Country_Region'] == country]
     df_country
```

	Country_Region	Confirmed	Deaths	Recovered	Active	Incident_Rate	Mortality_Rate
190	Vietnam	18360.0	84.0	7395.0	10881.0	18.861996	0.457516

```
effective_contact_rate = float(df_country['Incident_Rate']) * 0.05 # = transmission rate * contact rate
recovery_rate =  float(df_country['Mortality_Rate'] * (df_country['Deaths']  + df_country['Recovered']) / df_country['Deaths']) * 0.01 # = 1 / days

print("R0 is", effective_contact_rate / recovery_rate)

# State population in SIR model
# Everyone not infected or recovered is susceptible
total_pop = float(population_df[population_df['entity'] == country]['population'])
recovered = float(df_country['Recovered']) + float(df_country['Deaths'])
infected = df_country['Active']
susceptible = total_pop - infected - recovered

# A list of days, 0 -> 160
days = range(0,90)

# Use differential equations with the model
ret = odeint(ode, [susceptible,infected, recovered],days,args=(total_pop, effective_contact_rate, recovery_rate))
S, I, R = ret.T

# Build dataframe
df = pd.DataFrame({
        'susceptible': S,
        'infected': I,
        'recovered': R,
        'day': days
    })

plt.style.use('ggplot')
df.plot(figsize=(10,5),
            title='Prediction of COVID-19 in ' + country,
            x='day',
            y=['infected', 'susceptible', 'recovered'],
            color=['#bb6424', '#aac6ca', '#cc8ac0'],
            kind='line',
            stacked=False)
```

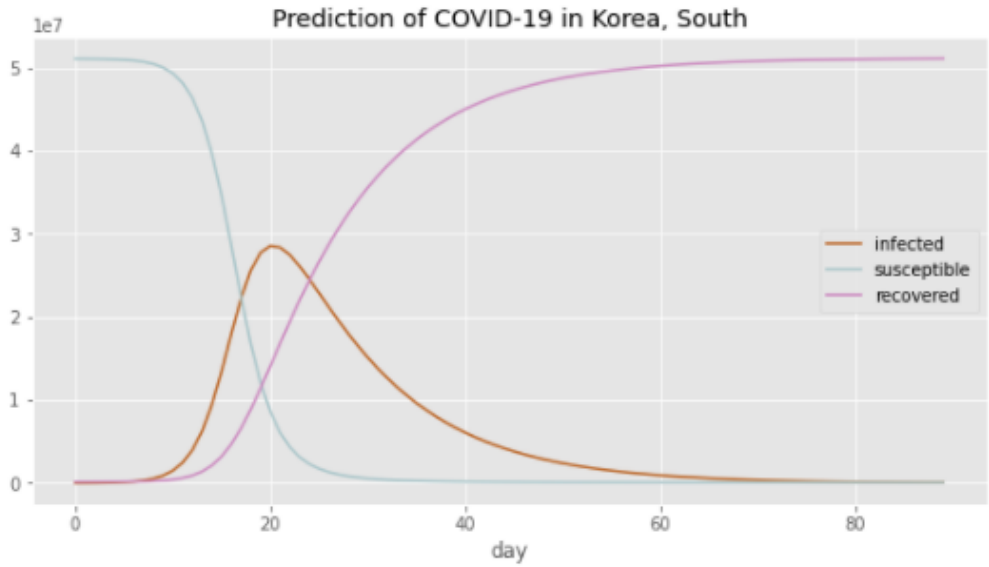
```
df
```



	susceptible	infected	recovered	day
0	9.732022e+07	1.088100e+04	7.479000e+03	0
1	9.731534e+07	1.523267e+04	8.005915e+03	1
2	9.730852e+07	2.132425e+04	8.743551e+03	2
3	9.729896e+07	2.985091e+04	9.776154e+03	3
4	9.728558e+07	4.178519e+04	1.122162e+04	4
...	...	...	...	...
85	2.294937e+04	9.516503e+06	8.779913e+07	85
86	2.213475e+04	9.137433e+06	8.817901e+07	86
87	2.137979e+04	8.773436e+06	8.854377e+07	87
88	2.067916e+04	8.423915e+06	8.889399e+07	88
89	2.002805e+04	8.088298e+06	8.923026e+07	89

90 rows × 4 columns

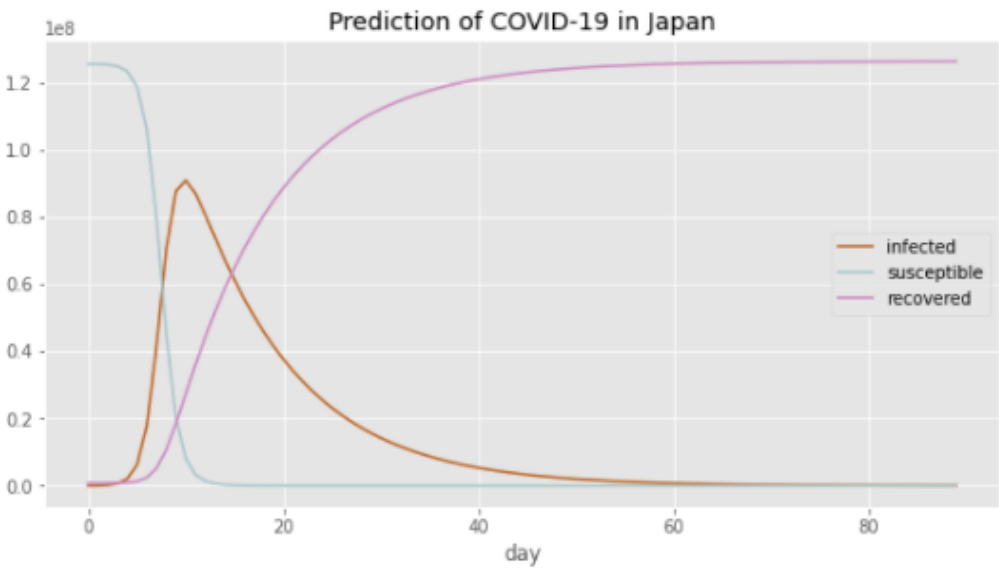
6.14.3 Korea



	susceptible	infected	recovered	day
0	5.110984e+07	8185.000000	1.511570e+05	0
1	5.110317e+07	13833.054190	1.521780e+05	1
2	5.109190e+07	23376.060200	1.539035e+05	2
3	5.107287e+07	39495.402002	1.568192e+05	3
4	5.104073e+07	66709.859776	1.617446e+05	4
...	...	...	...	...
85	7.592013e+04	89278.561614	5.110398e+07	85
86	7.584174e+04	81273.345850	5.111207e+07	86
87	7.577044e+04	73985.855455	5.111943e+07	87
88	7.570560e+04	67351.752745	5.112613e+07	88
89	7.564662e+04	61312.465302	5.113222e+07	89

90 rows × 4 columns

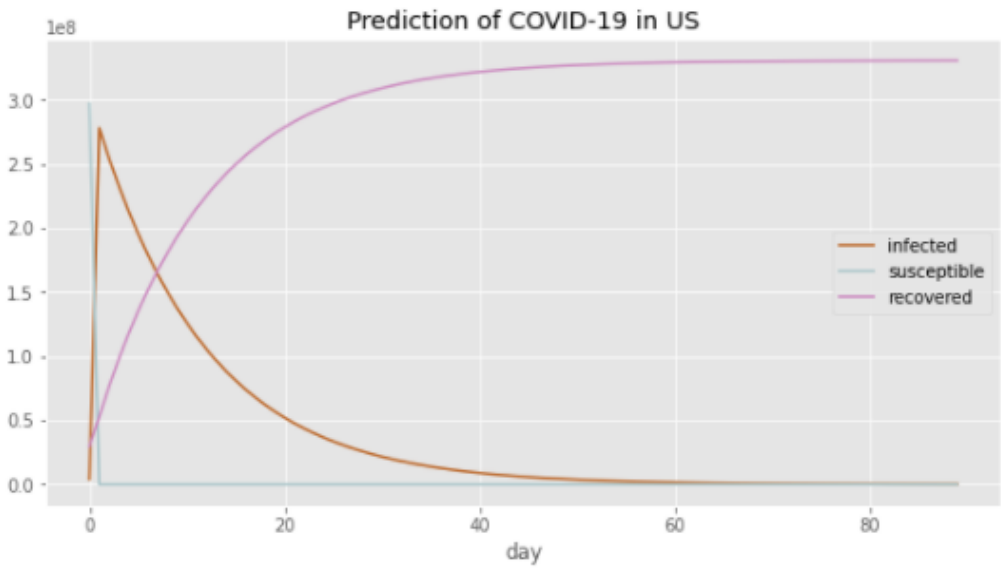
6.14.4 Japan



	susceptible	infected	recovered	day
0	1.256724e+08	1.944200e+04	7.846350e+05	0
1	1.256259e+08	6.237005e+04	7.882288e+05	1
2	1.254768e+08	1.999064e+05	7.997528e+05	2
3	1.250009e+08	6.389214e+05	8.366368e+05	3
4	1.234987e+08	2.023782e+06	9.539925e+05	4
...	...	...	...	...
85	3.009060e+02	6.645611e+04	1.264097e+08	85
86	3.007145e+02	6.027772e+04	1.264159e+08	86
87	3.005410e+02	5.467374e+04	1.264215e+08	87
88	3.003836e+02	4.959076e+04	1.264266e+08	88
89	3.002410e+02	4.498034e+04	1.264312e+08	89

90 rows × 4 columns

6.14.5 USA

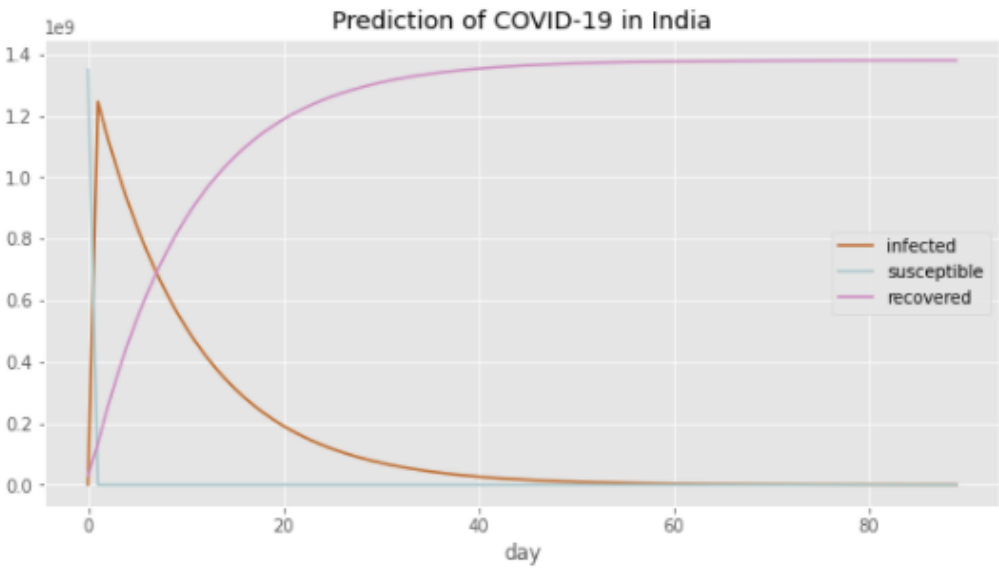



```
df
```

	susceptible	infected	recovered	day
0	2.972935e+08	4.015134e+06	2.969404e+07	0
1	8.514363e-10	2.781708e+08	5.283182e+07	1
2	9.742312e-11	2.547153e+08	7.628733e+07	2
3	-4.476177e-12	2.332376e+08	9.776506e+07	3
4	-1.101043e-11	2.135709e+08	1.174318e+08	4
...	...	...	...	...
85	-1.352898e-12	1.701233e+05	3.308325e+08	85
86	-1.320191e-12	1.557784e+05	3.308469e+08	86
87	-1.289799e-12	1.426431e+05	3.308600e+08	87
88	-1.262732e-12	1.306153e+05	3.308720e+08	88
89	-1.238722e-12	1.196018e+05	3.308830e+08	89

90 rows × 4 columns

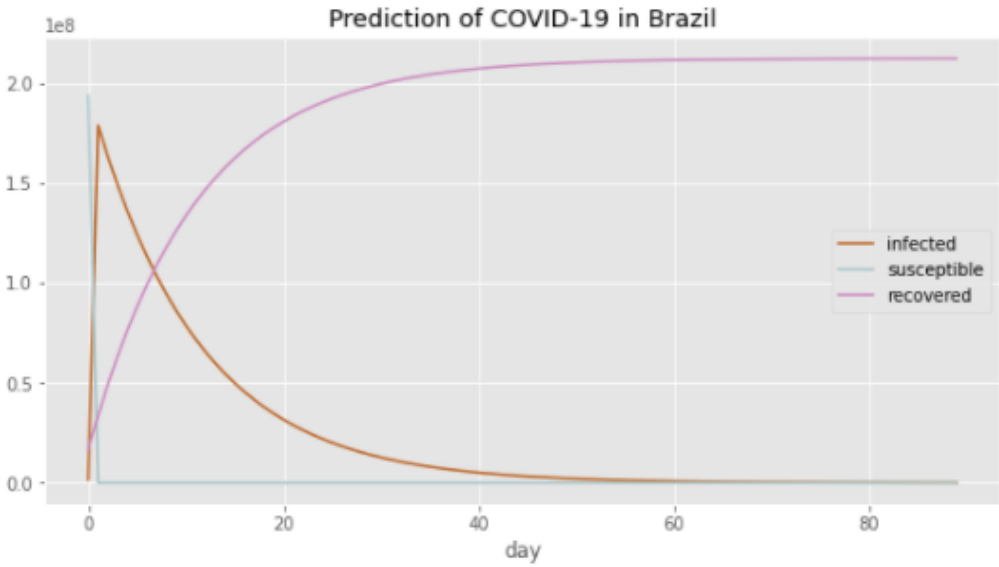
6.14.6 India



	susceptible	infected	recovered	day
0	1.349502e+09	4.955330e+05	3.000683e+07	0
1	2.474973e-06	1.245791e+09	1.342131e+08	1
2	-2.304745e-10	1.129071e+09	2.509330e+08	2
3	9.938390e-11	1.023287e+09	3.567173e+08	3
4	1.912153e-11	9.274138e+08	4.525905e+08	4
...	...	...	...	...
85	1.481382e-11	3.210978e+05	1.379683e+09	85
86	1.466941e-11	2.910137e+05	1.379713e+09	86
87	1.453974e-11	2.637482e+05	1.379741e+09	87
88	1.442322e-11	2.390373e+05	1.379765e+09	88
89	1.431842e-11	2.166415e+05	1.379788e+09	89

90 rows × 4 columns

6.14.7 Brazil



	susceptible	infected	recovered	day
0	1.938719e+08	1.655696e+06	1.703177e+07	0
1	1.363386e-11	1.789761e+08	3.358330e+07	1
2	1.343517e-11	1.633855e+08	4.917394e+07	2
3	-6.046314e-12	1.491529e+08	6.340647e+07	3
4	-7.551417e-12	1.361602e+08	7.639921e+07	4
...	...	...	...	...
85	-4.624993e-12	8.471071e+04	2.124747e+08	85
86	-4.325366e-12	7.733155e+04	2.124821e+08	86
87	-4.068831e-12	7.059518e+04	2.124888e+08	87
88	-3.847948e-12	6.444562e+04	2.124950e+08	88
89	-3.656793e-12	5.883176e+04	2.125006e+08	89

90 rows × 4 columns

7 Model Evaluation

7.1 Testing

Evaluating the accuracy of the models is a crucial step in determining how well they perform.

So as to test our models, we opted for mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and R-squared. They are evaluation metrics that compare the predicted values with the test samples. In other words, we look for mistakes in our numbers using the mentioned metrics. Therefore, **the lower the error, the better the model**, which signifies the accuracy and usability of a particular prediction method.

- Mean absolute error tells the difference between the prediction and test sample by calculating the average of absolute difference over the dataset.
- Mean squared error calculates the average difference, which is then squared, over the dataset.
- Root mean squared error takes the square root of mean squared error.
- R-squared, also known as coefficient of determination, calculates how well predicted values fit in with the test sample, and is presented in percentage. The higher this value is, the better the prediction.

As seen through out our research, we mainly use MAE and MSE to find the optimal accuracy of the models. What is more, based on personal experience and other observation, we can alter these figures for the best result.

7.2 Model Analysis

In order to complete this Discrete Structures for Computing assignment, we used a handful of prediction models to gather some very interesting results.

7.2.1 Time Series models

First, the Time Series models, ARIMA and Holt-Winters, are perfect for short-term predictions. On the long run, however, these models fail to generate proper results. This is due to the minimal input of data. As COVID-19 is still a rather new disease, there is actually not enough data to be used for long-term predictions of time series models, and they also depend on the trends of countries' data, which in our case, a month is possible but a 3-month prediction is likely impossible. That raised the demand of having to train again for the next prediction. Also, we encountered a problem with ARIMA where its results are one day ahead of the actual flow of time. Anyway, this problem is minor and does not negatively affect the overall picture.

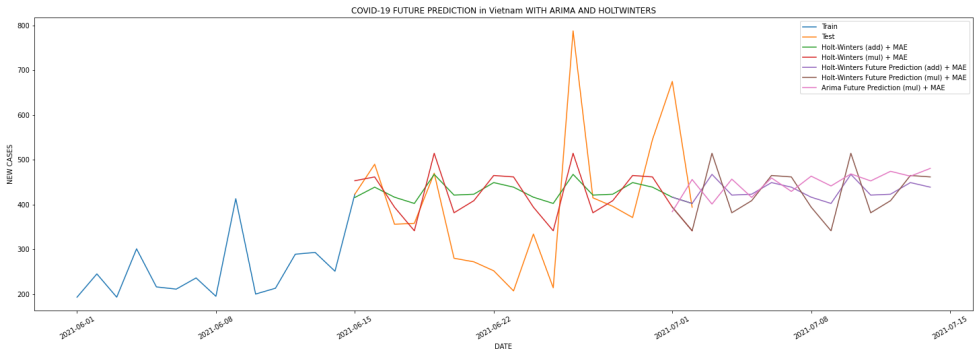


Figure 85: Comparison between ARIMA and Holt-Winters and their predicting abilities.

7.2.2 Linear Regression and Decision Tree

Next, the Linear Regression model and Decision Tree produce quite similar results. In particular, Decision Tree relies a lot on other features, probabilities as we put it, and unfortunately the data does not cover a lot of features for Decision Tree to work at its best.

7.2.3 SIR Model

SIR is an interesting talk. There are a number of reasons why SIR-based models failed to predict the COVID-19 pandemic. These oversimplified models neglect the factors that have a significant impact on illness progression. Various studies, for example, show that air pollution is a major contributor to observed variations in disease transmission. Another factor influencing virus spread is behavioral changes that are strongly linked to the population's social and cultural context. Because SIR models often examine only a single R0 value, unanticipated changes in population social behaviors are ignored, and the model is unable to keep up with the changes.

Ceremonies and national gatherings do, in fact, have a significant impact on disease transmission. For example, a religious gathering in Malaysia from February 27 to March 3 is suspected of spreading the virus to India and Pakistan42. Similar social activities in Isfahan may, at least in part, explain the unexpected rise that began in July. And SIR do not take these national holidays into account, which in turn become the second weakness of SIR model.

Another reason for the failure of SIR-based models to predict COVID-19 epidemics is that they are based on assumptions that are not always true; in SIR models, the population is considered closed, whereas in most regions, complete isolation was not followed, leaving them vulnerable to changes in neighboring communities. Furthermore, in the SIR models, the recovered individuals are thought to be immune, making them no longer susceptible. This idea is at odds with recent discoveries that imply the virus could be reactivated or already infected people could get infected again.

7.2.4 Support-vector Machine

This model is not suitable for COVID-19 prediction. When we tried many parameters C and γ and degree to train many times with three different kernel types (Linear, RBF and Poly), we found that the predictability of SVM deviated quite a lot from that on the test set, line prediction is almost straight line like Linear Regression and the higher we adjust the *degree*, the bigger the error. So we think maybe SVM is more suitable for classification problem or maybe we do not have too many features while training model. The illustration for the problem is described in Figure 86 and Figure 87.

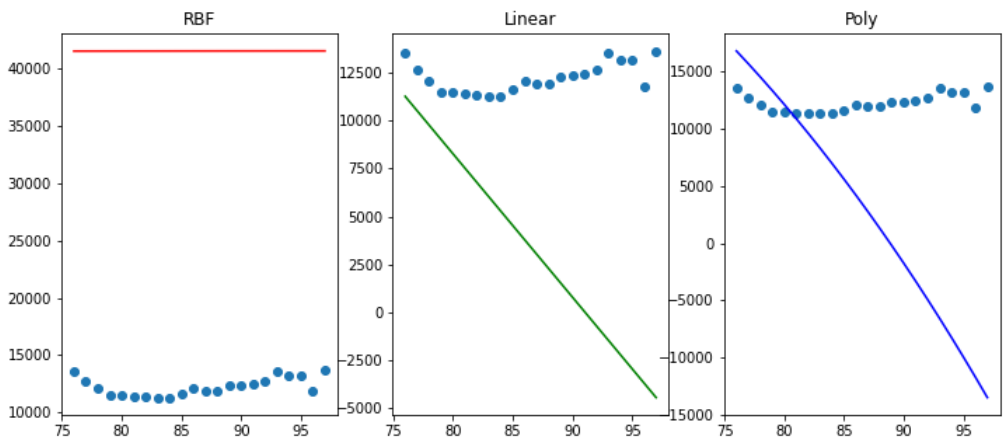


Figure 86: RBF - Linear - Poly Errors.

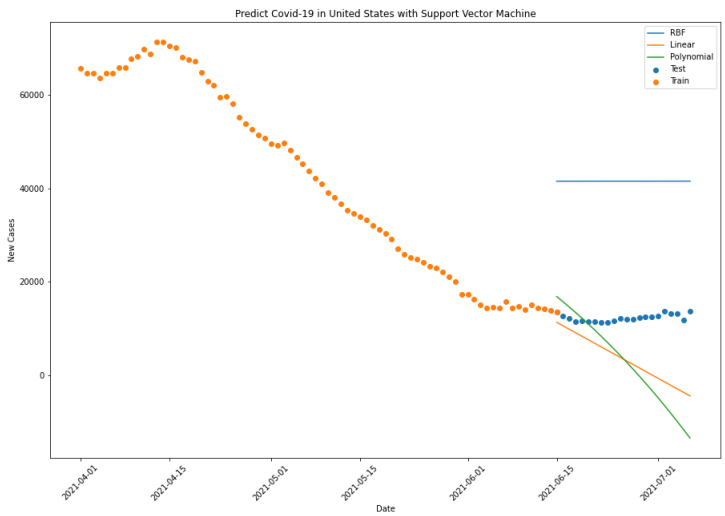


Figure 87: SVM Model Error.

7.3 Influencing factors

For respiratory viral diseases like COVID-19, the influencing factors can be those related to hospital beds, the number of patients with other underlying diseases, smoking habits, gender, and so on. The real struggle to apply them into our model is that not all regions collect these features for the observation data. Also, unlike total deaths, the recovered patients, or daily new cases, those figures are not updated frequently, and have the potential of stalling our research, or even completely halt our progress. Furthermore, the datasets we reached out for had very few options when it comes to miscellaneous information like that.

But then again, a thorough observation of influencing factors can be found in our data analysis part, notably when we found out if the popular events like New Years or Christmas have anything to do with the recorded information.

7.4 Is it possible to use a cross-country model?

A prediction model, if usable across different countries and regions, which means we use the model that had been trained in a country's data set to predict other country/region, is called cross-country.

In fact, this is almost Utopian because each country has its own data set, meaning that the different range of values and variability is visible. But if two regions have similar trends in a certain figure, the same prediction can be applied, even though the error between the actual data and the predicted data is still very large. It is possible to slightly adjust the range by adding or subtracting the MAE for a better fit, but not the best.

8 Conclusion

Even after reading a 150-page report, you are still bound to have a lot of questions concerning the COVID-19 conditions due to the complexity in the way this disease is spreading. Although there are numerous models that can be used to forecast the COVID-19 epidemic in the near future, these models have flaws since they oversimplify external elements. As a result, everyone of us has a responsibility to take care of ourselves and improve our health, as well as to take precautions such as wearing masks and avoiding congested areas for the greater good of the community. Furthermore, governments should urgently provide incentives for vaccine development in order to quickly return our world to its original situation.

This concludes our group effort; it is our delight that you took the time to read our work, and we hope that you will be safe in the COVID-19 circumstance as well.

For further question, please contact us via this email: phuc.nguyenng@hcmut.edu.vn.

9 Our code for analysis and model prediction

9.1 Time-based Analysis

LINK 1

9.2 Location-based Analysis

LINK 2

9.3 Time Series Model

LINK 3

9.4 Linear Regression, Support Vector Machine, Decision Tree Models

LINK 4

9.5 SIR Model

LINK 5

10 Some demo codes for other machine learning algorithms

10.1 Linear Regression demo code

LINK 6

10.2 Logistic Regression demo code

LINK 7

10.3 K-means demo code

LINK 8

10.4 Neural Network demo code

LINK 9

10.5 Principal Component Analysis demo code

LINK 10

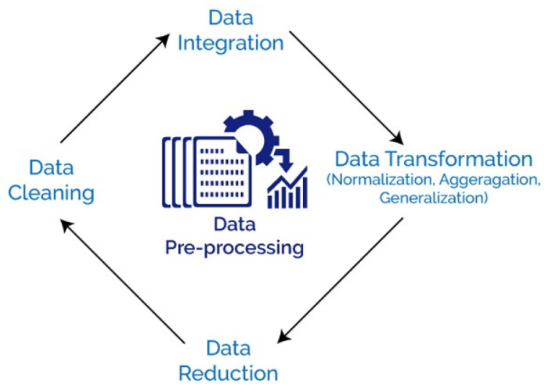
11 APPENDIX

A Data preprocessing

A.1 What is data preprocessing?

- Data preprocessing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, and missing values, etc.

  => We need to do data preprocessing before train the machine learning model.

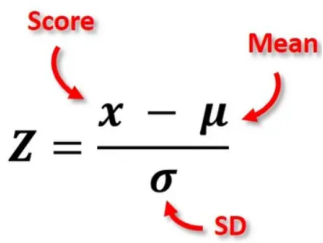


A.2 Z-score and P-value

A.2.1 What is a Z-Score?

- A z-score describes the position of a raw score in terms of its distance from the mean, when measured in standard deviation units. The z-score is positive if the value lies above the mean, and negative if it lies below the mean.

- We have the formula for Z-score:



A.2.2 What is a P-value?

- A p-value is a measure of the probability that an observed difference could have occurred just by random chance. The lower the p-value, the greater the statistical significance of the observed difference. P-value can be used as an alternative to or in addition to pre-selected confidence levels for hypothesis testing

P-value	Description	Hypothesis Interpretation
P-value ≤ 0.05	It indicates the null hypothesis is very unlikely.	Rejected
P-value > 0.05	It indicates the null hypothesis is very likely.	Accepted or it "fails to reject".
P-value > 0.05	The P-value is near the cut-off. It is considered as marginal	The hypothesis needs more attention.

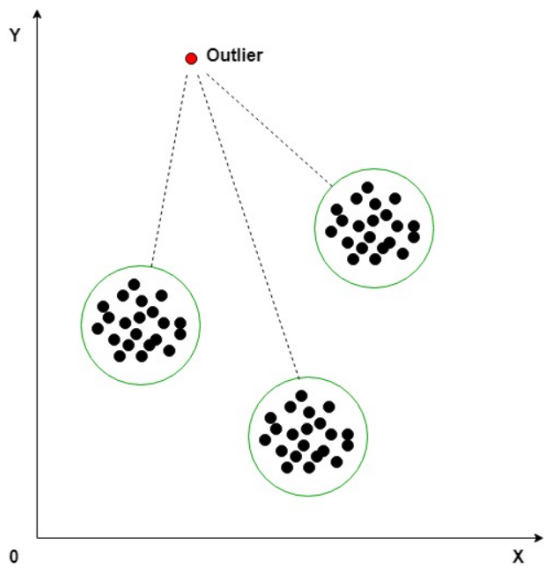
Figure 88: P-value table

Why do we need to know the normal distribution, z-score and p-value? =>To determine the outliers when we clean the data.

A.3 Outlier

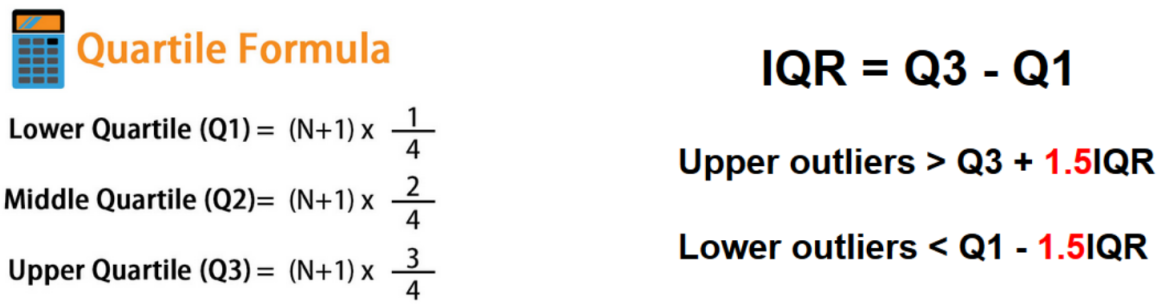
A.3.1 What is the Outlier?

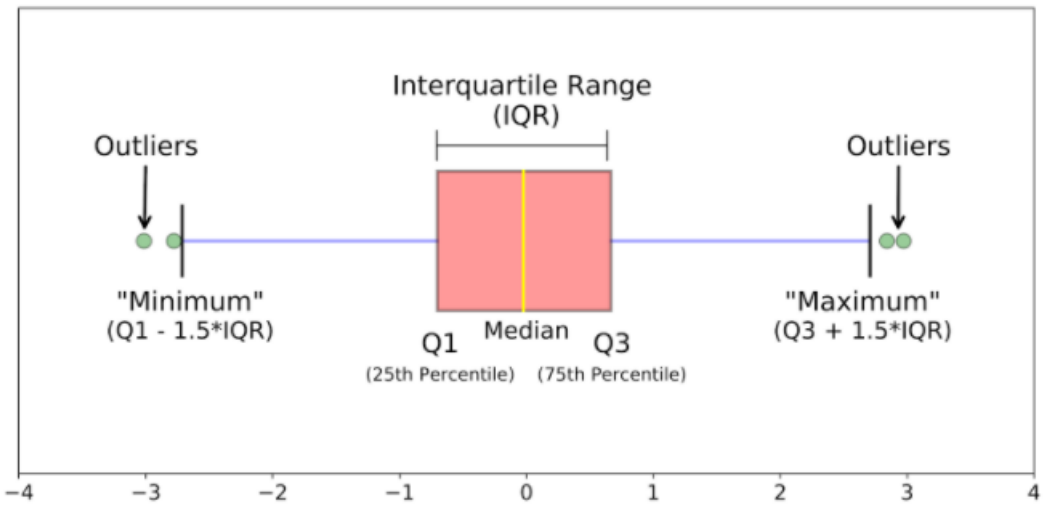
- In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.



A.3.2 How to determine the outliers?

- **Using Z-score:**
  - As we mentioned above, we can use z-score, z-table in Normal Distribution to determine outliers. If the z score of a data point is more than 3, it indicates that the data point is quite different from the other data points. Such a data point can be an outlier.
- **Using The Interquartile Range (IQR):**
  - In descriptive statistics, the interquartile range, also called the midspread, middle 50%, or H-spread, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, $\text{IQR} = Q_3 - Q_1$.





- **Using box-plot (in Python)**

A.4 Data Cleaning/Cleansing:

- Real-world data tend to be incomplete, noisy, and inconsistent. Data Cleaning/-Cleansing routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.
- Data can be noisy, having incorrect attribute values. Owing to the following, the data collection instruments used may be fault. Maybe human or computer errors occurred at data entry. Errors in data transmission can also occur.
- "Dirty" data can cause confusion for the mining procedure. Although most mining routines have some procedures, they deal incomplete or noisy data, which are not always robust. Therefore, a useful Data Preprocessing step is to run the data through some Data Cleaning/Cleansing routines.

A.5 Data Integration

- Data Integration is involved in data analysis task which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. The issue to be considered in Data Integration is schema integration. It is tricky.
- How can real-world entities from multiple data sources be 'matched up'? This is referred as entity identification problem. For example, how can a data analyst be sure that customer-id in one database and cust-number in another refer to the same entity? The answer is metadata. Databases and data warehouses typically have metadata. Simply, metadata is data about data.
- Metadata is used to help avoiding errors in schema integration. Another important issue is redundancy. An attribute may be redundant, if it is derived from another table. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

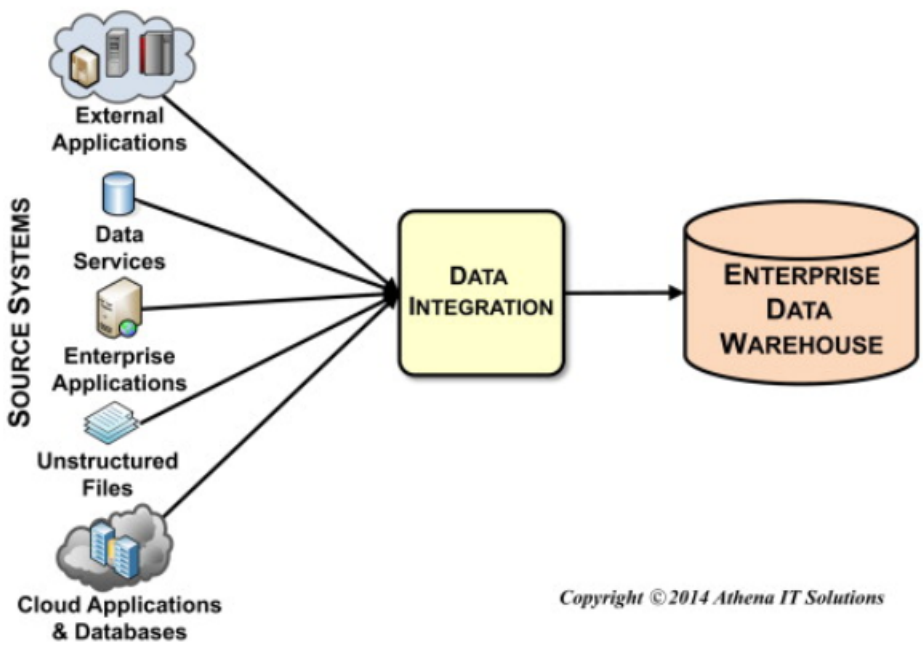
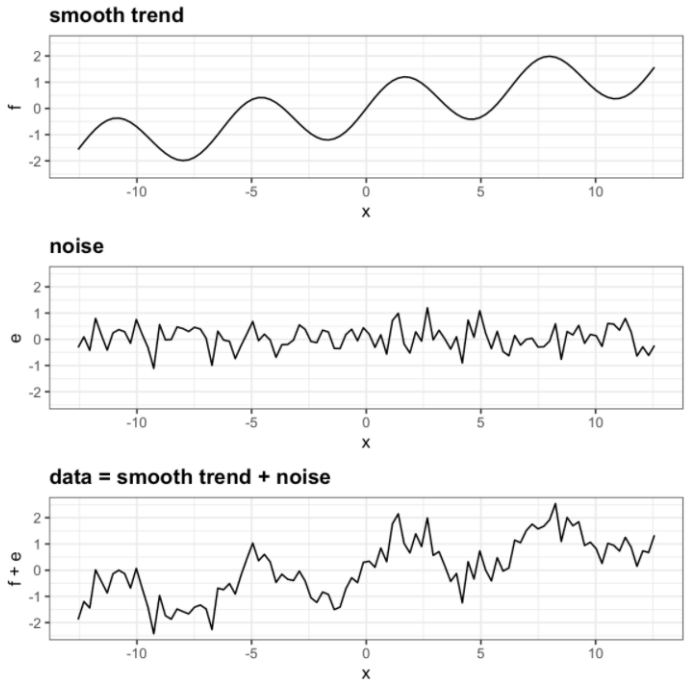


Figure 89: Role of Data Integration

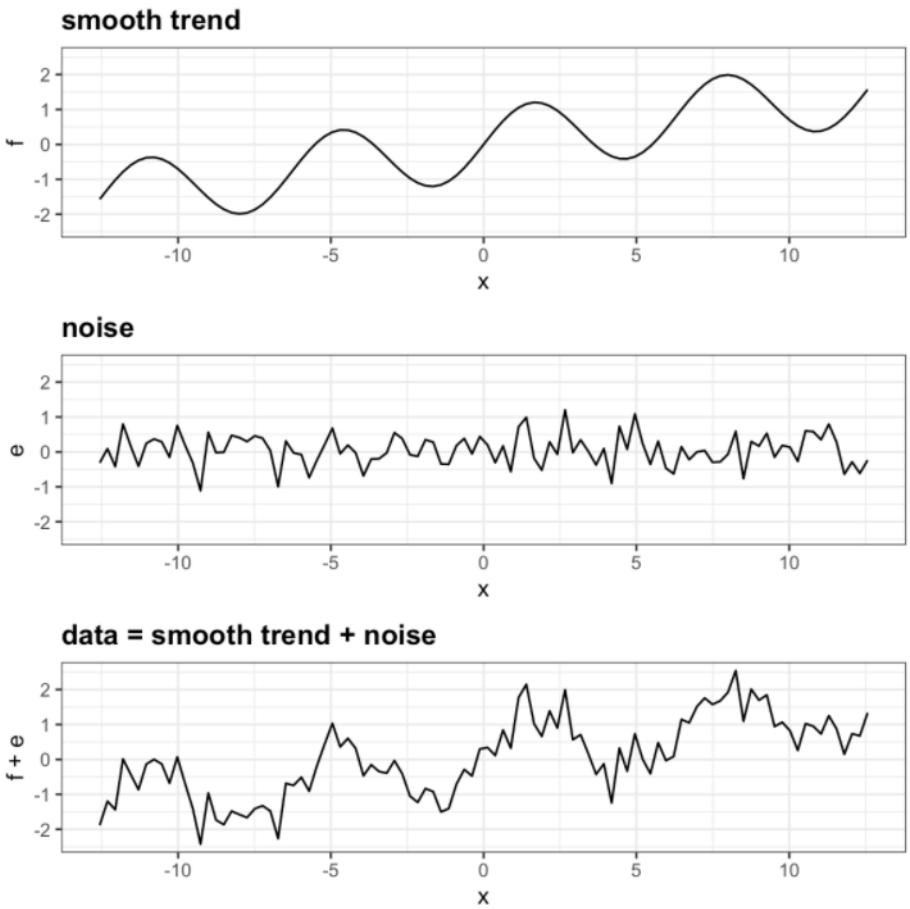
A.6 Data Transformation

- Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system. Data transformation is a component of most data integration and data management tasks, such as data wrangling and data warehousing.



A.7 How to do the data transforming?

- **Smoothing:**
  - It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form.
  - The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn't see otherwise.
- **Aggregation:**
  - It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes.
  - Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values.
  - For example here:
- **Attribute Construction:**
  - Where new attributes are created and applied to assist the mining process from the given set of attributes. This simplifies the original data and makes



the mining more efficient.

- **Generalization:**
  - Data generalization is the process that abstracts a large set of task-relevant data in a database from a low conceptual level to higher ones. It is a summarization of general features of objects in a target class and produces what is called characteristic rules => Data reduction
- **Normalization:**
  - Some types of normalization
    * Min-Max Normalization
    * Z-Score Normalization
    * Decimal Scaling

Company	Name	Age	Wages	Education.University	Productivity
A	`Wayne	26	50000	1	100
A	Duane	27	70000	1	120
B	William	28	70000	1	120
C	Rafael	32	60000	0	95
A	John	28	50000	0	88
B	Eric	24	70000	1	115
B	James	34	65000	1	100
C	Pablo	30	50000	0	90
C	Tammy	25	55000	1	120

Company	average Age	average Wages	Sum. Education.Unive	average Productivity
A	27	56600	2	102,6
B	28,6	68333	3	111,6
C	29	55000	1	101,6

A.8 Data Reduction

- Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data Reduction techniques are helpful in analysing the reduced representation of the data set without compromising the integrity of the original data and yet producing the qualitative knowledge. Strategies for data reduction include the following:
  - In Data Cube Aggregation, aggregation operations are applied to the data in the construction of a data cube.
  - In Dimension Reduction, irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
  - In Data Compression, encoding mechanisms are used to reduce data set size. The methods used for Data Compression are Wavelet Transform and Principle Component Analysis.
  - In Numerosity Reduction, data is replaced or estimated by alternative and smaller data representations such as parametric models (which store only the model parameters instead of the actual data, e.g. Regression and Log-Linear Models) or non-parametric methods (e.g. Clustering, Sampling, and the use of histograms).
  - In Discretisation and Concept Hierarchy Generation, raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are powerful tools for data mining.

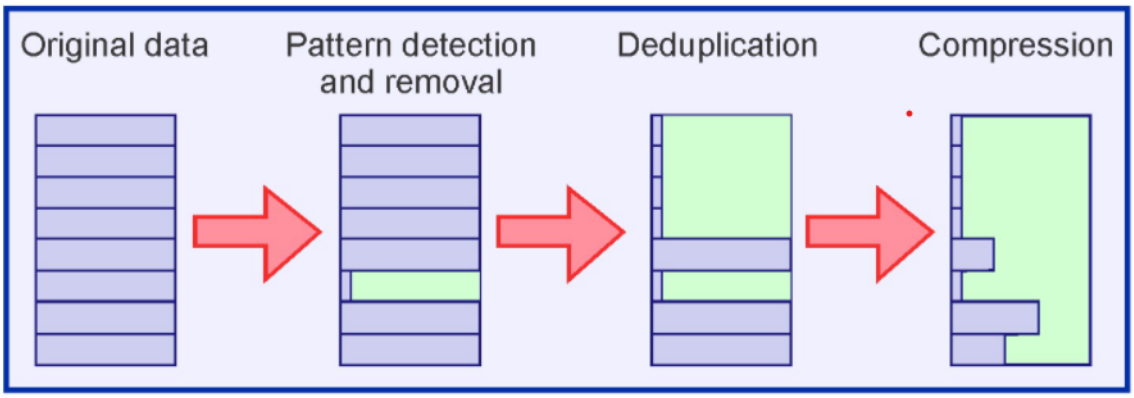


Figure 90: Caption

B Other basic algorithms in Machine Learning

B.1 Supervised learning

B.1.1 Logistic regression(Classification problem)

B.1.1.a What is Logistic regression?

- Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes:
  - 1: positive class
  - 0: negative class

B.1.1.b Type of Logistic regression

- Binomial: target variable can have only 2 possible types: "0" or "1" which may represent "win" vs "loss", "pass" vs "fail", "dead" vs "alive", etc.
- Multinomial: target variable can have 3 or more possible types which are not ordered(i.e. types have no quantitative significance) like "disease A" vs "disease B" vs "disease C".
- Ordinal: it deals with target variables with ordered categories. For example, a test score can be categorized as : "very poor", "poor", "good", "very good". Here, each category can be given a score like 0, 1, 2, 3.

Sigmoid function

- We use the sigmoid function to map predicted values to probabilities. The function maps any real value into another value between 0 and 1.
- The logistic regression hypothesis is defined as:

$$h_\theta(x) = g\left(\theta^T x\right)$$

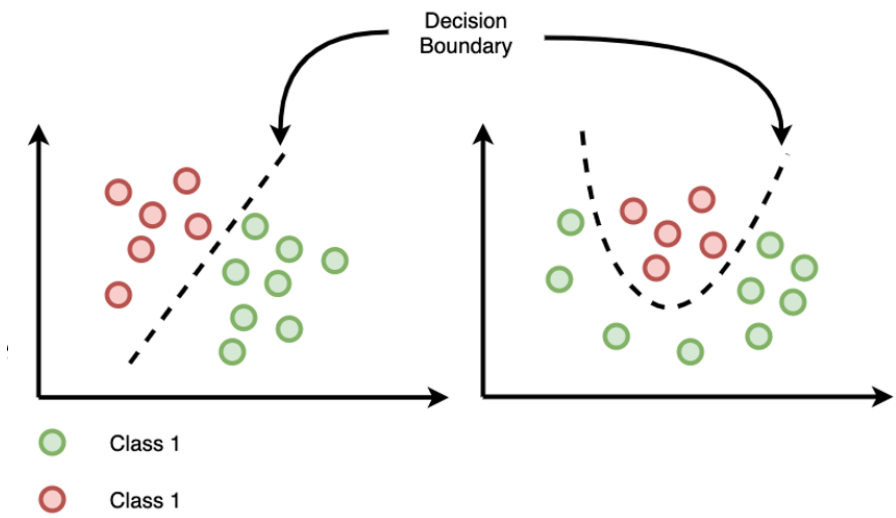


Figure 91: Decision boundary

- where function g is the sigmoid function. The sigmoid function is defined as:

$$g(z) = \frac{1}{1 + e^{-z}}$$

  - g(z):output between 0 and 1
  - Z: input to the function
  - e: base of natural log

Decision boundary

- The prediction function returns a probability score between 0 and 1. If we want to map the discrete class (true/false, yes/no), we will have to select a threshold value above which you will be classifying values into class 1 and below the threshold value into class 2.
  - p $\geq$ 0.5, class=1
  - p $<$ 0.5, class=0

Cost function

- A cost function is a function of input prices and output quantity whose value is the cost of making that output given those input prices, often applied through the use of the cost curve by companies to minimize cost and maximize production efficiency.
- For logistic regression, the cost function is defined as:

$$\text{Cost}\left(h_\theta(x), y\right) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

- The above two functions can be compressed into a single function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

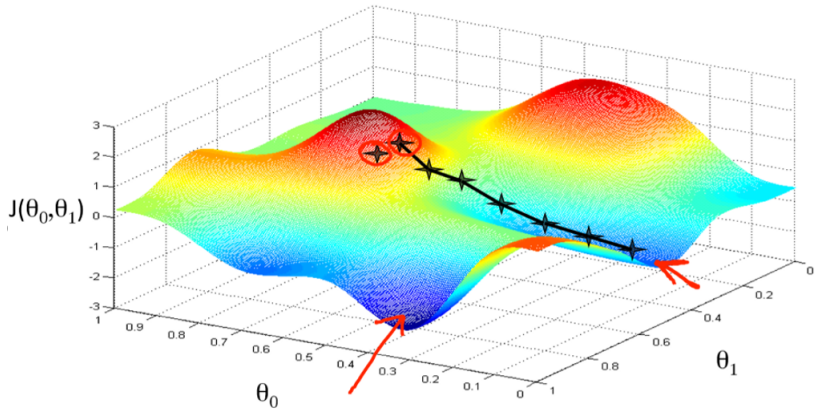


Figure 92: Gradient descent

Gradient descent

- The main goal of Gradient descent is to minimize the cost value. In order to minimize our cost function we need to run the gradient descent function on each parameter.

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \quad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1$$

B.1.1.c Overfitting and Underfitting

Underfitting

- The model contains high bias is simpler than it should be and hence tends to underfit. In other words, model fails to learn on the given data and acquire the intricate pattern of the dataset.
- It Means a biased model will not fit on the Training Dataset properly and hence will have low training accuracy (or high training loss).
- In a nutshell, Underfitting – High bias and low variance.
- Techniques to reduce underfitting :
  - Increase model complexity
  - Increase number of features, performing feature engineering
  - Remove noise from the data.
  - Increase the number of epochs or increase the duration of training to get better results.

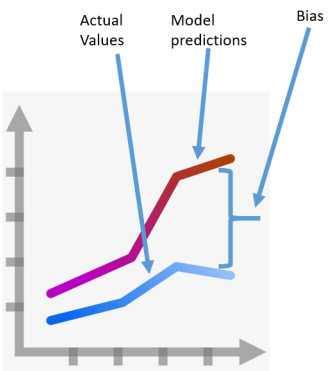
Overfitting

- A model with high Variance will have a tendency to be overly complex. This causes the overfitting of the model.
- Suppose the model with high Variance will have very high training accuracy (or very low training loss), but it will have a low testing accuracy (or a low testing loss).
- Overcomplicating simpler problems: A model with a high variance means the model is overly complex and tries to fit a much more complex curve to a relatively simpler data. The model is thus capable of solving complex problems but incapable of solving simple problems efficiently.
- In a nutshell, Overfitting – High variance and low bias.
- Techniques to reduce overfitting :
  - Increase training data.
  - Reduce model complexity.
  - Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
  - Ridge Regularization and Lasso Regularization
  - Use dropout for neural networks to tackle overfitting.

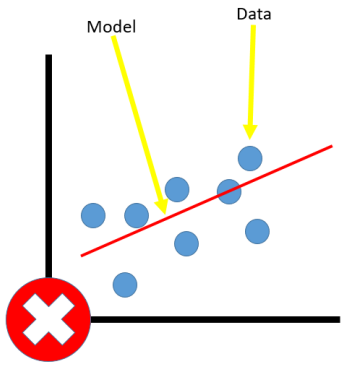
B.1.1.d Bias and Variance

Bias and its effects

- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.



- When the BIAS IS HIGH, assumptions made by our model are too basic, the model can't capture the important features of our data.



Variance and its effects

- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.
- Overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy datasets. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.

Bias-Variance Trade-off

- When we are facing this underfitting and Overfitting into our predictive model we need to fix this problem generally used bias-variance trade-off.

Why is Bias Variance Tradeoff?

- If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.
- This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

B.2 Unsupervised Learning

B.2.1 K-Means Clustering in Machine Learning

- Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes. It is proposed by Jame McQueen in 1967.

  The mission of K-Means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset.

  A cluster refers to a collection of data points aggregated together because of certain similarities.

- You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.

  Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

  In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

  The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

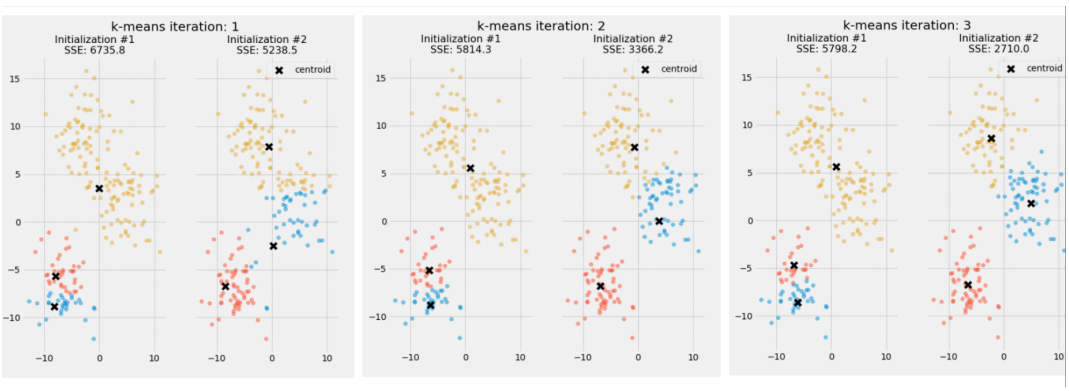
  Spaceflights cannot be stopped. This is not the work of any one man or even a group of men. It is a historical process which mankind is carrying out in accordance with the natural laws of human development.

B.2.2 How the K-means algorithm works?

- To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids. Here's what the conventional version of the k-means algorithm looks like:

  1. Specify the number k of clustering to assign.
  2. Randomly initialize k centroids.
  3. **Repeat**
  4. **Expectation:**Assign each points to each closest centroid.
  5. **Maximization:** Comput the new centroid (mean) of each cluster.
  6. **Until** The centroid positions do not change

The purpose of this figure is to show that the initialization of the centroids is an important step. It also highlights the use of SSE as a measure of clustering performance. After



choosing a number of clusters and the initial centroids, the expectation-maximization step is repeated until the centroid positions reach convergence and are unchanged.
The random initialization step causes the k-means algorithm to be non-deterministic , meaning that cluster assignments will vary if you run the same algorithm twice on the same dataset. Researchers commonly run several initialization of the entire k-means algorithm and choose the cluster assignments from the initialization with the lowest SSE.

B.2.3 Choosing the Appropriate Number of Cluster

- A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The Elbow Method is one of the most popular methods to determine this optimal value of k.
- The optimal number of clusters can be defined as follow:

Step 1: Creating and visualizing the data

- From the above visualization, we can see that the optimal number of clusters should be around 3. But visualizing the data alone cannot always give the right answer. Hence we demonstrate the following steps.

- We now define the following:
  - **Distortion:**It is calculated as the average of the squared distances from the cluster centers of the respective clusters. Typically, the Euclidean distance metric is used.
  - **Inertia:**It is the sum of squared distances of samples to their closest cluster center.

Step 2: Calculating the values of the Distortion and Inertia

Step 3: Tabulating and Visualizing the results

- Using different values of Disortion
- Using different values of inertia
  - To determine the optimal number of clusters, we have to select the value of k at the "elbow" ie the point after which the distortion/inertia start decreasing in a linear fashion. Thus for the given data, we conclude that the optimal number of clusters for the data is 3.
  - The cluster data points for different value of k:
    1. k = 1
    2. k = 2
    3. k = 3
    4. k = 4

B.2.4 Advantages and Disadvantages of K-means

- **Advantages of K-means:**
  - Relatively simple to implement.
  - Scales to large data sets.
  - Guarantees convergence.
  - Can warm-start the positions of centroids.
  - Easily adapts to new examples.
  - Generalizes to clusters of different shapes and sizes, such as elliptical clusters.
- **Disadvantages of k-means:**
  - Choosing k manually.
    - Use the "Loss vs. Clusters" plot to find the optimal (k), as discussed in Interpret Results.
  - Being dependent on initial values.

    * For a low k, you can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of k-means to pick better values of the initial centroids (called k-means seeding). For a full discussion of k-means seeding see, A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm by M. Emre Celebi, Hassan A. Kingravi, Patricio A. Vela.
  - Clustering data of varying sizes and density.
    * k-means has trouble clustering data where clusters are of varying sizes and density. To cluster such data, you need to generalize k-means as described in the Advantages section.
  - Clustering outliers.
    * Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.
  - Scaling with number of dimensions.
    * As the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples. Reduce dimensionality either by using PCA on the feature data, or by using "spectral clustering" to modify the clustering algorithm as explained below.

B.2.5 Applications of K-means

- K-means algorithm is very popular and used in a variety of applications such as market segmentation, document clustering, image segmentation and image compression, etc.
- The goal usually when we undergo a cluster analysis is either:
  1. Get a meaningful intuition of the structure of the data we're dealing with.
  2. Cluster-then-predict where different models will be built for different subgroups if we believe there is a wide variation in the behaviors of different subgroups. An example of that is clustering patients into different subgroups and build a model for each subgroup to predict the probability of the risk of having heart attack.

References

[1] **Overview of Machine Learning**
https://www.expert.ai/blog/machine-learning-definition/
https://www.sas.com/en_us/insights/analytics/machine-learning.html
https://www.investopedia.com/terms/d/deep-learning.asp
https://en.wikipedia.org/wiki/Reinforcement_learning

[2] **Data preprocessing**
https://www.upgrad.com/blog/methods-of-data-transformation-in-data-mining/
https://t4tutorials.com/decimal-scaling-normalization-in-data-mining/
https://byjus.com/maths/p-value/
https://en.wikipedia.org/wiki/Data_transformation
https://en.wikipedia.org/wiki/Data_pre-processing
https://medium.com/easyread/basics-of-data-preprocessing-71c314bc7188

[3] **Linear Regression**
https://www.hackerearth.com/blog/developers/
https://developers.google.com/machine-learning/crash-course/
reducing-loss/gradient-descent
https://www.coursera.org/lecture/machine-learning/cost-function-rkTp3
http://datapandas.com/
http://onlinestatbook.com/2/regression/intro.html

[4] **Logistic Regression**
https://lionbridge.ai/articles/7-types-of-data-bias-in-machine-learning/
https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b
https://www.fireblazeaischool.in/blogs/overfitting-and-underfitting/
https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[5] **Support Vector Machine**
https://machinelearningcoban.com/2017/04/13/softmarginsmv/
https://medium.com/bite-sized-machine-learning/
https://medium.com/machine-learning-101/
http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/
lecturenote09.html

[6] **Unsupervised Learning**
https://www.geeksforgeeks.org/k-means-clustering-introduction/
https://towardsdatascience.com/
https://realpython.com/k-means-clustering-python/

[7] **ARIMA**
https://machinelearningcoban.com/
https://www.machinelearningplus.com/time-series/
arima-model-time-series-forecasting-python/
https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
https://otexts.com/fpp2/AR.html
https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

[8] **Holt Winters**
https://otexts.com/fpp2/holt-winters.html
https://orangematter.solarwinds.com/2019/12/15/
holt-winters-forecasting-simplified/
https://en.wikipedia.org/wiki/Exponential_smoothing

[9] **Time Series**
https://otexts.com/fpp2/regression.html
https://vizzlo.com/gallery/time-series-graph/example/
u-s-passenger-air-travel
https://www.qimacros.com/control-chart/run-and-control-charts/
http://astro.uni-tuebingen.de/groups/time/
https://en.wikipedia.org/wiki/Time_series

[10] **Decomposing time serires**
https://otexts.com/fpp2/decomposition.html
https://en.wikipedia.org/wiki/Decomposition_of_time_series

[11] **SIR theory**
https://www.maa.org/press/periodicals/loci/joma/
the-sir-model-for-spread-of-disease-the-differential-equation-model
https://www.degruyter.com/document/doi/10.1515/em-2020-0044/html
https://www.researchgate.net/publication/349593155_Inefficiency_of_SIR_
models_in_forecasting_COVID-19_epidemic_a_case_study_of_Isfahan
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7570398/
https://www.maa.org/press/periodicals/loci/joma/
the-sir-model-for-spread-of-disease-relating-model-parameters-to-data
https://mathinsight.org/discrete_sir_infectious_disease_model
https://kingaa.github.io/clim-dis/parest/parest.html

[12] **SIR compartmental models**
https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7270519/

[13] **SIR model**
https://www.youtube.com/watch?v=LQCGeboXZkc&t=4s
https://www.youtube.com/watch?v=NKMHhm2Zbkw
https://www.youtube.com/watch?v=ITUiX5Fpmm4&t=860s
https://in.springboard.com/blog/data-modelling-covid/

[14] **Data Analysis**
https://otexts.com/fpp2/
https://www.machinelearningplus.com/plots/matplotlib-histogram-python-examples/
#5seabornhistogramanddensitycurveonthesameplot
https://en.wikipedia.org/wiki/Normal_distribution
https://opentextbc.ca/introbusinessstatopenstax/chapter/
skewness-and-the-mean-median-and-mode/
https://www.khanacademy.org/math/probability/data-distributions-a1

Duty roster of our team

Student's name	Student ID	Work
All Members		• Learnt necessary tools and environment for the project • Met every Tuesday and Saturday to report the individual progress • Checked the grammatical errors in the report, discussed and gave advice to each other
Nguyễn Văn Quốc Chương	1950004	• Built ARIMA, Holt-Winters, Linear Regression, Support Vector Machine models • Did the model evaluation step • Wrote latex chapter 6.1 and the Appendix
Nguyễn Nho Gia Phúc	2052214	• Managed team • Did the Introduction, Tool and Environment, Data collection, and Conclusion • Wrote Latex Chapter 1,2,3,4,6.8,6.10,7,8
Thái Tài	2052246	• Wrote theory for SIR, Times series, ARIMA and Holt-Winters • Wrote latex chapter 6 and the short report • Built SIR model
Nguyễn Trọng Chuẩn	2052046	• Built analysis coding: time-based and location-based and other analysis • Developed analysis strategies
Vũ Đồng Tuệ Quyên	2052234	• Developed analysis strategies • Wrote Latex chapter 5